MOTOROLA

MC68306

Integrated EC000 Processor
User's Manual

PREFACE

The complete documentation package for the MC68306 consists of the MC68306UM/AD, *MC68306 EC000 Integrated Processor User's Manual*, M68000PM/AD, *MC68000 Family Programmer's Reference Manual,* and the MC68306P/D, *MC68306 EC000 Integrated Processor Product Brief*.

The *MC68306 EC000 Integrated Processor User's Manual* describes the programming, capabilities, registers, and operation of the MC68306; the *MC68000 Family Programmer's Reference Manual* provides instruction details for the MC68306; and the *MC68306 EC000 Integrated Processor Product Brief* provides a brief description of the MC68306 capabilities.

This user's manual is organized as follows:

Section 1	Introduction
Section 2	Signal Descriptions
Section 3	68000 Bus Operation Description
Section 4	EC000 Core Processor
Section 5	System Operation
Section 6	Serial Module
Section 7	IEEE 1149.1 Test Access Port
Section 8	Electrical Specifications
Section 9	Ordering Information and Mechanical Data

68K FAX-IT – Documentation Comments

FAX 512-891-8593—Documentation Comments Only

The Motorola High-End Technical Publications Department provides a fax number for you to submit any questions or comments about this document or how to order other documents. We welcome your suggestions for improving our documentation. Please do not fax technical questions.

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.

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

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 —

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) 753-7360
CALIFORNIA, Rosevllle	(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, Marborough	(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, Fairport	(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) 690-5593
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

Field Applications Engineering Available
Through All Sales Offices

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 761921
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 Whltely	(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, lnc.	(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

TABLE OF CONTENTS

Paragraph Number	Title	Page Number

Section 1
Introduction

1.1 MC68EC000 Core processor 1-2
1.2 On-Chip Peripherals 1-3
1.2.1 Serial Module 1-3
1.2.2 DRAM Controller 1-4
1.2.3 Chip Selects 1-4
1.2.4 Parallel Ports 1-4
1.2.5 Interrupt Controller 1-4
1.2.6 Clock 1-5
1.2.7 Bus Timeout Monitor 1-5
1.2.8 Mode Controller 1-5
1.2.9 IEEE 1149.1 Test 1-5

Section 2
Signal Descriptions

2.1 Bus Signals 2-5
2.1.1 Address Bus (A23–A1) 2-5
2.1.2 Address Strobe ($\overline{AS}$) 2-5
2.1.3 Bus Error ($\overline{BERR}$) 2-5
2.1.4 Bus Request ($\overline{BR}$) 2-5
2.1.5 Bus Grant ($\overline{BG}$) 2-6
2.1.6 Bus Grant Acknowledge ($\overline{BGACK}$) 2-6
2.1.7 Data Bus (D15–D0) 2-6
2.1.8 Data Transfer Acknowledge ($\overline{DTACK}$) 2-6
2.1.9 DRAM Multiplexed Address Bus ($\overline{DRAMA14}$–$\overline{DRAMA0}$) 2-6
2.1.10 Processor Function Codes (FC2–FC0) 2-6
2.1.11 Halt ($\overline{HALT}$) 2-7
2.1.12 Read/Write (R/$\overline{W}$) 2-7
2.1.13 Upper And Lower Data Strobes ($\overline{UDS}$, $\overline{LDS}$) 2-7
2.1.14 Upper Byte Write ($\overline{UW}$) 2-8
2.1.15 Lower Byte Write ($\overline{LW}$) 2-8
2.1.16 Output Enable ($\overline{OE}$) 2-8
2.1.17 Reset ($\overline{RESET}$) 2-8
2.2 Chip Select Signals 2-9

TABLE OF CONTENTS (Continued)

Paragraph Number	Title	Page Number
2.3	DRAM Controller Signals	2-9
2.3.1	Column Address Strobe ($\overline{CAS1}$–$\overline{CAS0}$)	2-9
2.3.2	Row Address Strobe ($\overline{RAS1}$–$\overline{RAS0}$)	2-9
2.3.3	DRAM Write Signal ($\overline{DRAMW}$)	2-9
2.4	Interrupt Control and Parallel Port Signals	2-9
2.4.1	Interrupt Request (IRQ7–IRQ1)	2-9
2.4.2	Interrupt Acknowledge (IACK7–IACK1)	2-9
2.4.3	Port A Signals (PA7–PA0)	2-9
2.4.4	Port B (PB7–PB0)	2-9
2.5	Clock and Mode Control Signals	2-10
2.5.1	Crystal Oscillator (EXTAL, XTAL)	2-10
2.5.2	Clock Out (CLKOUT)	2-10
2.5.3	Address Mode (AMODE)	2-10
2.6	Serial Module Signals	2-10
2.6.1	Channel A Receiver Serial-Data Input (RxDA)	2-10
2.6.2	Channel A Transmitter Serial-Data Output (TxDA)	2-10
2.6.3	Channel B Receiver Serial-Data Input (RxDB)	2-10
2.6.4	Channel B Transmitter Serial-Data Output (TxDB)	2-10
2.6.5	CTSA	2-11
2.6.6	RTSA	2-11
2.6.7	CTSB	2-11
2.6.8	RTSB	2-11
2.6.9	Crystal Oscillator (X1, X2)	2-11
2.6.10	IP2	2-11
2.6.11	OP3	2-11
2.7	JTAG Port Test Signals	2-11
2.7.1	Test Clock (TCK)	2-12
2.7.2	Test Mode Select (TMS)	2-12
2.7.3	Test Data In (TDI)	2-12
2.7.4	Test Data Out (TDO)	2-12
2.7.5	Test Reset (TRST)	2-12

Section 3
68000 Bus Operation Description

3.1	Data Transfer Operations	3-1
3.1.1	Read Cycle	3-1
3.1.2	Write Cycle	3-4
3.1.3	Read-Modify-Write Cycle	3-7
3.1.4	CPU Space Cycle	3-11

TABLE OF CONTENTS (Continued)

Paragraph Number	Title	Page Number
3.2	Bus Arbitration	3-12
3.2.1	Requesting the Bus	3-15
3.2.2	Receiving the Bus Grant	3-16
3.2.3	Acknowledgment of Mastership (3-Wire Bus Arbitration Only)	3-16
3.3	Bus Arbitration Control	3-16
3.4	Bus Error and Halt Operation	3-24
3.4.1	Bus Error Operation	3-24
3.4.2	Retrying the Bus Cycle	3-25
3.4.3	Halt Operation	3-26
3.4.4	Double Bus Fault	3-27
3.5	Reset Operation	3-28
3.6	The Relationship of $\overline{\text{DTACK}}$, $\overline{\text{BERR}}$, and $\overline{\text{HALT}}$	3-28
3.7	Asynchronous Operation	3-30
3.8	Synchronous Operation	3-33

Section 4
EC000 Core Processor

4.1	Features	4-1
4.2	Processing States	4-1
4.3	Programming Model	4-2
4.3.1	Data Format Summary	4-3
4.3.2	Addressing Capabilities Summary	4-4
4.3.3	Notation Conventions	4-5
4.4	EC000 Core Instruction Set Overview	4-7
4.5	Exception Processing	4-12
4.5.1	Exception Vectors	4-14
4.6	Processing of Specific Exceptions	4-16
4.6.1	Reset Exception	4-17
4.6.2	Interrupt Exceptions	4-17
4.6.3	Uninitialized Interrupt Exception	4-18
4.6.4	Spurious Interrupt Exception	4-18
4.6.5	Instruction Traps	4-18
4.6.6	Illegal and Unimplemented Instructions	4-18
4.6.7	Privilege Violations	4-19
4.6.8	Tracing	4-19
4.6.9	Bus Error	4-20
4.6.10	Address Error	4-21
4.6.11	Multiple Exceptions	4-21

TABLE OF CONTENTS (Continued)

Paragraph Number	Title	Page Number

Section 5
System Operation

5.1 MC68306 Address Space 5-1
5.2 Register Description 5-3
5.2.1 System Register 5-3
5.2.2 Timer Vector Register 5-4
5.2.3 Bus Timeout Period Register 5-4
5.2.4 Interrupt Registers 5-5
5.2.4.1 Interrupt Control Register 5-5
5.2.4.2 Interrupt Status Register 5-6
5.2.5 I/O Port Registers 5-6
5.2.5.1 Port Pins Register 5-7
5.2.5.2 Port Direction Register 5-7
5.2.5.3 Port Data Register 5-8
5.2.6 Chip Selects 5-8
5.2.6.1 Chip Select Configuration Registers (High Half) 5-9
5.2.6.2 Chip Select Configuration Registers (Low Half) 5-10
5.2.7 DRAM Control Registers 5-12
5.2.7.1 DRAM Refresh Register 5-13
5.2.7.2 DRAM Bank Configuration Register (High Half) 5-14
5.2.7.3 DRAM Bank Configuration Register (Low Half) 5-14
5.2.8 Automatic DTACK Generation 5-16
5.3 Crystal Oscillator 5-16

Section 6
Serial Module

6.1 Module Overview 6-2
6.1.1 Serial Communication Channels A and B 6-3
6.1.2 Baud Rate Generator Logic 6-3
6.1.3 Timer/Counter 6-3
6.1.4 Interrupt Control Logic 6-3
6.1.5 Comparison of Serial Module to MC68681 6-4
6.2 Serial Module Signal Definitions 6-4
6.2.3 Channel A Transmitter Serial Data Output (TxDA) 6-4
6.2.4 Channel A Receiver Serial Data Input (RxDA) 6-5
6.2.5 Channel B Transmitter Serial Data Output (TxDB) 6-5
6.2.6 Channel B Receiver Serial Data Input (RxDB) 6-6
6.2.7 Channel A Request-To-Send (RTSA/OP0) 6-6
6.2.7.1 RTSA 6-6
6.2.7.2 OP0 6-6
6.2.8 Channel B Request-To-Send (RTSB/OP1) 6-6

TABLE OF CONTENTS (Continued)

Paragraph Number	Title	Page Number
6.2.8.1	RTSB	6-6
6.2.8.2	OP1	6-6
6.2.9	Channel A Clear-To-Send (CTSA/IP0)	6-6
6.2.9.1	CTSA	6-6
6.2.9.2	IP0	6-6
6.2.10	Channel B Clear-To-Send (CTSB/IP1)	6-6
6.2.10.1	CTSB	6-6
6.2.10.2	IP1	6-6
6.3	Operation	6-7
6.3.1	Baud Rate Generator	6-7
6.3.2	Transmitter and Receiver Operating Modes	6-7
6.3.2.1	Transmitter	6-9
6.3.2.2	Receiver	6-10
6.3.2.3	FIFO Stack	6-11
6.3.3	Looping Modes	6-13
6.3.3.1	Automatic Echo Mode	6-13
6.3.3.2	Local Loopback Mode	6-13
6.3.3.3	Remote Loopback Mode	6-13
6.3.4	Multidrop Mode	6-14
6.3.5	Counter/Timer	6-16
6.3.5.1	Counter Mode	6-16
6.3.5.2	Timer Mode	6-16
6.3.6	Bus Operation	6-17
6.3.6.1	Read Cycles	6-17
6.3.6.2	Write Cycles	6-17
6.3.6.3	Interrupt Acknowledge Cycles	6-17
6.4	Register Description and Programming	6-17
6.4.1	Register Description	6-17
6.4.1.1	Mode Register 1 (DUMR1)	6-18
6.4.1.2	Mode Register 2 (DUMR2)	6-20
6.4.1.3	Status Register (DUSR)	6-22
6.4.1.4	Clock-Select Register (DUCSR)	6-24
6.4.1.5	Command Register (DUCR)	6-26
6.4.1.6	Receiver Buffer (DURB)	6-29
6.4.1.7	Transmitter Buffer (DUTB)	6-29
6.4.1.8	Input Port Change Register (DUIPCR)	6-29
6.4.1.9	Auxiliary Control Register (DUACR)	6-30
6.4.1.10	Interrupt Status Register (DUISR)	6-31
6.4.1.11	Interrupt MASK Register (DUIMR)	6-33
6.4.1.12	Count Register Current MSB of Counter (DUCUR)	6-33
6.4.1.13	Count Register Current LSB of Counter (DUCLR)	6-33
6.4.1.14	Counter/Timer Upper Preload Register (CTUR)	6-34

TABLE OF CONTENTS (Continued)

Paragraph Number	Title	Page Number
6.4.1.15	Counter/Timer Lower Pimer Register (CTLR)	6-34
6.4.1.16	Interrupt Vector Register (DUIVR)	6-34
6.4.1.17	Input Port Register	6-34
6.4.1.18	Output Port Control Register (DUOPCR)	6-35
6.4.1.19	Output Port Data Register (DUOP)	6-35
6.4.1.20	Start Counter Command Register	6-36
6.4.1.21	Stop Counter Command Register	6-36
6.4.2	Programming	6-36
6.4.2.1	Serial Module Initialization.	6-36
6.4.2.2	I/O Driver Example	6-37
6.4.2.3	Interrupt Handling	6-37
6.5	Serial Module Initialization Sequence	6-43

Section 7
IEEE 1149.1 Test Access Port

7.1	Overview	7-1
7.2	TAP Controller	7-3
7.3	Boundary Scan Register	7-3
7.4	Instruction Register	7-9
7.4.1	EXTEST (000)	7-10
7.4.2	SAMPLE/PRELOAD (110)	7-10
7.4.3	BYPASS (010, 101, 111)	7-11
7.4.4	CLAMP (011)	7-11
7.5	MC68306 Restrictions	7-11
7.6	Non-IEEE 1149.1 Operation	7-12

Section 8
Electrical Specifications

8.1	Maximum Ratings	8-1
8.2	Thermal Characteristics	8-1
8.3	Power Considerations	8-2
8.4	AC Electrical Specification Definitions	8-2
8.5	DC Electrical Specifications	8-4
8.6	AC Electrical Specifications—Clock Timing	8-4
8.7	AC Electrical Specifications—Read and Write	8-5
8.8	AC Electrical Specifications—Chip Selects	8-9
8.9	AC Electrical Specifications—Bus Arbitration	8-10
8.10	Bus Operation—DRAM Accesses AC Timing Specifications	8-12
8.11	Serial Module Electrical Characteristics	8-15
8.12	Serial Module AC Electrical Characteristics—Clock Timing	8-16
8.13	AC Electrical Characteristics—Port Timing	8-16
8.14	AC Electrical Characteristics—Interrupt Reset	8-16

TABLE OF CONTENTS (Continued)

Paragraph Number	Title	Page Number
8.14	AC Electrical Characteristics—Interrupt Reset	8-16
8.15	AC Electrical Characteristics—Transmitter Timing	8-17
8.16	AC Electrical Characteristics—Receiver Timing	8-18
8.17	IEEE 1149.1 Electrical Characteristics	8-19

Section 9
Ordering Information and Mechanical Data

9.1	Standard Ordering Information	9-1
9.2	Pin Assignments	9-2

LIST OF ILLUSTRATIONS

Figure Number	Title	Page Number
Figure 1-1.	MC68306 Simplified Block Diagram	1-1
Figure 2-1.	MC68306 Detailed Block Diagram	2-2
Figure 3-1.	Word Read Cycle Flowchart	3-2
Figure 3-2.	Byte Read Cycle Flowchart	3-2
Figure 3-3.	Read and Write Cycle Timing Diagram	3-3
Figure 3-4.	Word and Byte Read Cycle Timing Diagram	3-3
Figure 3-5.	Word Write Cycle Flowchart	3-5
Figure 3-6.	Byte Write Cycle Flowchart	3-6
Figure 3-7.	Word and Byte Write Cycle Timing Diagram	3-6
Figure 3-8.	Read-Modify-Write Cycle Flowchart	3-8
Figure 3-9.	Read-Modify-Write Cycle Timing Diagram	3-9
Figure 3-10.	Interrupt Acknowledge Cycle	3-11
Figure 3-11.	Interrupt Acknowledge Cycle Timing Diagram	3-12
Figure 3-12.	Three-Wire Bus Arbitration Cycle Flowchart	3-13
Figure 3-13.	Two-Wire Bus Arbitration Cycle Flowchart	3-14
Figure 3-14.	Three-Wire Bus Arbitration Timing Diagram	3-15
Figure 3-15.	Two-Wire Bus Arbitration Timing Diagram	3-15
Figure 3-16.	External Asynchronous Signal Synchronization	3-17
Figure 3-17.	Bus Arbitration Unit State Diagrams	3-19
Figure 3-18.	Three-Wire Bus Arbitration Timing Diagram—Processor Active	3-20
Figure 3-19.	Three-Wire Bus Arbitration Timing Diagram—Bus Inactive	3-21
Figure 3-20.	Three-Wire Bus Arbitration Timing Diagram—Special Case	3-22
Figure 3-21.	Two-Wire Bus Arbitration Timing Diagram—Processor Active	3-23
Figure 3-22.	Two-Wire Bus Arbitration Timing Diagram—Bus Inactive	3-24
Figure 3-23.	Two-Wire Bus Arbitration Timing Diagram—Special Case	3-25
Figure 3-24.	Bus Error Timing Diagram	3-26
Figure 3-25.	Retry Bus Cycle Timing Diagram	3-27
Figure 3-26.	Halt Operation Timing Diagram	3-28
Figure 3-27.	Reset Operation Timing Diagram	3-29
Figure 3-28	Fully Asynchronous Read Cycle	3-32
Figure 3-29.	Fully Asynchronous Write Cycle	3-32
Figure 3-30.	Pseudo-Asynchronous Read Cycle	3-33
Figure 3-31.	Pseudo-Asynchronous Write Cycle	3-34
Figure 3-32.	Synchronous Read Cycle	3-36
Figure 3-33.	Synchronous Write Cycle	3-37

LIST OF ILLUSTRATIONS (Continued)

Figure Number	Title	Page Number
Figure 4-1.	Programmer's Model	4-2
Figure 4-2.	Status Register	4-3
Figure 4-3.	General Exception Processing Flowchart	4-13
Figure 4-4.	General Form of Exception Stack Frame	4-14
Figure 4-5.	Exception Vector Format	4-15
Figure 4-6.	Address Translated from 8-Bit Vector Number	4-15
Figure 4-7.	Supervisor Stack Order for Bus or Address Error Exception	4-21
Figure 5-1.	Chip Select Expansion	5-12
Figure 5-2.	Oscillator Circuit Diagram	5-17
Figure 6-1.	Simplified Block Diagram	6-1
Figure 6-2.	External and Internal Interface Signals	6-5
Figure 6-3.	Baud Rate Generator Block Diagram	6-7
Figure 6-4.	Transmitter and Receiver Functional Diagram	6-8
Figure 6-5.	Transmitter Timing Diagram	6-9
Figure 6-6.	Receiver Timing Diagram	6-11
Figure 6-7.	Looping Modes Functional Diagram	6-14
Figure 6-8.	Multidrop Mode Timing Diagram	6-15
Figure 6-9.	Serial Module Programming Model	6-18
Figure 6-10.	Serial Module Programming Flowchart	6-38
Figure 7-1.	Test Access Port Block Diagram	7-2
Figure 7-2.	TAP Controller State Machine	7-3
Figure 7-3.	Output Cell (O.Cell)	7-7
Figure 7-4.	Input Cell (I.Cell)	7-7
Figure 7-5.	Output Control Cell (En.Cell)	7-8
Figure 7-6.	Bidirectional Cell (IO.Cell)	7-8
Figure 7-7.	Bidirectional Cell (IOx0.Cell)	7-9
Figure 7-8.	General Arrangement for Bidirectional Pins	7-9
Figure 7-9.	Bypass Register	7-11
Figure 8-1.	Drive Levels and Test Points for AC Specifications	8-3
Figure 8-2.	Clock Output Timing	8-4
Figure 8-3.	Read Cycle Timing Diagram	8-7
Figure 8-4.	Write Cycle Timing Diagram	8-8
Figure 8-5.	Chip Select and Interrupt Acknowledge Timing Diagram	8-9
Figure 8-6.	Bus Arbitration Timing Diagram	8-10
Figure 8-7.	Bus Arbitration Timing Diagram	8-11
Figure 8-8.	DRAM Timing – 0-Wait Read, No Refresh	8-13
Figure 8-9.	DRAM Timing – 1-Wait Write, No Refresh	8-14
Figure 8-10.	DRAM Timing – 0- and 1-Wait Refresh	8-14

LIST OF ILLUSTRATIONS (Continued)

Figure Number	Title	Page Number
Figure 8-11.	DRAM Timing – 1-Wait, Test and Set	8-15
Figure 8-12.	Clock Timing	8-16
Figure 8-13.	Port Timing	8-16
Figure 8-14.	Interrupt Reset Timing	8-17
Figure 8-15.	Transmit Timing	8-17
Figure 8-16.	Receive Timing	8-18
Figure 8-17.	Test Clock Input Timing Diagram	8-19
Figure 8-18.	Boundary Scan Timing Diagram	8-20
Figure 8-19.	Test Access Port Timing Diagram	8-20

LIST OF TABLES

Table Number	Title	Page Number
Table 2-1.	Bus Signal Summary	2-3
Table 2-2.	Chip Select Signal Summary	2-3
Table 2-3.	DRAM Controller Signal Summary	2-3
Table 2-4.	Interrupt and Parallel Port Signal Summary	2-4
Table 2-5.	Clock and Mode Control Signal Summary	2-4
Table 2-6.	Serial Module Signal Summary	2-4
Table 2-7.	JTAG Signal Summary	2-5
Table 2-8.	Function Code Outputs	2-7
Table 2-9.	Data Strobe Control of Data Bus	2-8
Table 3-1.	DTACK, BERR, and HALT Assertion Results	3-24
Table 3-2.	BERR and HALT Negation Results	3-25
Table 4-1.	Processor Data Formats	4-3
Table 4-2.	Effective Addressing Modes	4-4
Table 4-3.	Notation Conventions	4-5
Table 4-4.	EC000 Core Instruction Set Summary	4-8
Table 4-5.	Exception Vector Assignments	4-16
Table 4-6.	Exception Grouping and Priority	4-22
Table 5-1.	MC68306 Memory Map	5-2
Table 5-2.	Chip Select Match Bits	5-11
Table 5-3.	DRAM Address Multiplexer	5-13
Table 5-4.	DRAM Bank Match Bits	5-15
Table 6-1.	PMx and PT Control Bits	6-20
Table 6-2.	B/Cx Control Bits	6-20
Table 6-3.	CMx Control Bits	6-21
Table 6-4.	SBx Control Bits	6-22
Table 6-5.	RCSx Control Bits	6-25
Table 6-6.	TCSx Control Bits	6-26
Table 6-7.	MISCx Control Bits	6-27
Table 6-8.	TCx Control Bits	6-28
Table 6-9.	RCx Control Bits	6-28
Table 6-10.	Counter/Timer Mode and Source Select Bits	6-30

LIST OF TABLES (Continued)

Table Number	Title	Page Number
Table 7-1.	Boundary Scan Control Bits	7-4
Table 7-2.	Boundary Scan Bit Definitions	7-5
Table 7-3.	Instructions	7-10

SECTION 1
INTRODUCTION

The MC68306 is an integrated processor containing an MC68EC000 processor and elements common to many MC68000- and MC68EC000-based systems. Designers of virtually any application requiring MC68000-class performance will find that the MC68306 reduces design time by providing valuable system elements integrated in one chip. The combination of peripherals offered in the MC68306 can be found in a diverse range of microprocessor-based systems, including embedded control and general computing. Systems requiring serial communication and dynamic random access memory (DRAM) can especially benefit from using the MC68306.

The MC68306's high level of functional integration results in significant reductions in component count, power consumption, board space, and cost while yielding much higher system reliability and shorter design time. Complete code compatibility with the MC68000 affords the designer access to a broad base of established real-time kernels, operating systems, languages, applications, and development tools, many of which are oriented towards embedded control. Figure 1-1 shows a simplified block diagram of the MC68306.

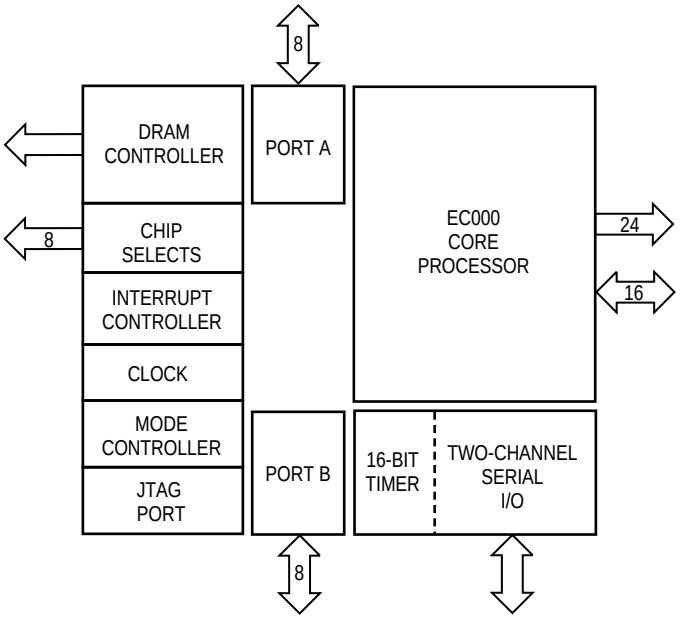


Figure 1-1. MC68306 Simplified Block Diagram

The primary features of the MC68306 are as follows:

- Functional Integration on a Single Piece of Silicon
- EC000 Core—Identical to MC68EC000 Microprocessor
  - — Complete Code Compatibility with MC68000 and MC68EC000
  - — High Performance—2.4 MIPS
  - — Extended Internal Address Range – to 4 Gbyte
- Two-Channel Universal Synchronous/Asynchronous Receiver/Transmitter (DUART)
  - — Baud Rate Generators
  - — Modem Control
  - — Compatible with MC68681/MC2681
  - — Integrated 16-Bit Timer/Counter
- DRAM Controller
  - — Supports up to 16 Mbytes using 4M x 1 DRAMs, 64 Mbytes using 16M x 1 DRAMs
  - — Provides Zero Wait State Interface to 80-ns DRAMs
  - — Programmable Refresh Timer Provides $\overline{\text{CAS}}$-before-$\overline{\text{RAS}}$ Refresh
- Chip Selects
  - — Eight Programmable Chip Select Signals
  - — Provide Eight Separate 1-Mbyte Spaces or Four Separate 16-Mbyte Spaces
  - — Programmable Wait States
- Programmable Interrupt Controller
- Bus Timeout
- 24 Address Lines, 16 Data Lines
- 16.67 MHz, 5 Volt Operation
- 144-Pin Thin Quad Flat Pack (TQFP)or 132-Pin Plastic Quad Flat Pack (PQFP)

1.1 MC68EC000 CORE PROCESSOR

The MC68EC000 is a core implementation of the MC68000 32-bit microprocessor architecture. The programmer can use any of the eight 32-bit data registers for fast manipulation of data and any of the eight 32-bit address registers for indexing data in memory. Flexible instructions support data movement, arithmetic functions, logical operations, shifts and rotates, bit set and clear, conditional and unconditional program branches, and overall system control.

The MC68EC000 core can operate on data types of single bits, binary-coded decimal (BCD) digits, and 8, 16, and 32 bits. The integrated chip selects allow peripherals and data in memory to reside anywhere in the 4-Gbyte linear address space. A supervisor operating mode protects system-level resources from the more restricted user mode, allowing a true virtual environment to be developed. Many addressing modes complement

these instructions, including predecrement and postincrement, which allow simple stack and queue maintenance and scaled indexing for efficient table accesses. Data types and addressing modes are supported orthogonally by all data operations and with all appropriate addressing modes. Position-independent code is easily written.

Like all M68000 family processors, the MC68EC000 core recognizes interrupts of seven different priority levels and allows either an automatic vector or a peripheral-supplied vector to direct the processor to the desired service routine. Internal trap exceptions ensure proper instruction execution with good addresses and data, allow operating system intervention in special situations, and permit instruction tracing. Hardware signals can either terminate or rerun bad memory accesses before instructions process data incorrectly. The EC000 core provides 2.4 MIPS at 16.67 MHz.

1.2 ON-CHIP PERIPHERALS

To improve total system throughput and reduce part count, board size, and cost of system implementation, the M68300 family integrates on-chip, intelligent peripheral modules and typical glue logic. The functions on the MC68306 include two serial channels, a timer/counter, a DRAM controller, a parallel port, and system glue logic.

1.2.1 Serial Module

Most digital systems use serial I/O to communicate with host computers, operator terminals, or remote devices. The MC68306 contains a two-channel, full-duplex UART with an integrated timer. An on-chip baud rate generator provides standard baud rates up the 38.4K baud to each channel's receiver and transmitter. The serial module is identical to the MC68681/MC2681 DUART.

Each communication channel is completely independent. Data formats can be 5, 6, 7, or 8 bits with even, odd, or no parity and stop bits up to 2 in 1/16 increments. Four-byte receive buffers and two-byte transmit buffers minimize CPU service calls. Each channel provides a wide variety of error detection and maskable interrupt capability. Full-duplex, autoecho loopback, local loopback, and remote loopback modes can be selected. Multidrop applications are also supported.

A 3.6864 MHz crystal drives the baud rate generators. Each transmit and receive channel can be programmed for a different baud rate. Full modem support is provided with separate request-to-send (RTS) and clear-to-send (CTS) signals for each channel.

The integrated 16-bit timer/counter can operate in a counter mode or a timer mode. The timer/counter can function as a system stopwatch, a real-time single interrupt generator, or a device watchdog when in counter mode. In timer mode, the timer/counter can be used as a programmable clock source for channels A and B, a periodic interrupt generator, or a variable duty cycle square-wave generator.

1.2.2 DRAM Controller

DRAM is used in many systems since it is the least expensive form of high-speed storage available. However, considerable design effort is often spent designing the interface

between the processor and DRAM. The MC68306 contains a full DRAM controller, greatly reducing design time and complexity.

The DRAM controller provides row address strobe ($\overline{RAS}$) and column address strobe ($\overline{CAS}$) signals for two separate banks of DRAMs. Each bank can include up to 16 devices; up to 15 multiplexed address lines are also available. Thus, using 4M x 1 DRAMs, up to 16 Mbytes of DRAM are supported; with 16M x 1 DRAMs, up to 64 Mbytes of DRAM are supported. A programmable refresh timer provides $\overline{CAS}$-before-$\overline{RAS}$ refreshes at designated intervals.

The DRAM controller has its own address registers that control the address range selected by each $\overline{RAS}$ and $\overline{CAS}$ signal, leaving the eight integrated chip selects free for other system peripherals. DRAM accesses are zero wait states using 80-ns DRAMs.

1.2.3 Chip Selects

The MC68306 provides up to eight programmable chip select outputs, in most cases eliminating the need for external address decoding. All handshaking and timing signals are provided, with up to 950-ns access times. Each chip select can access a 16 Mbyte address space located anywhere in the 4-Gbyte address range. Internal registers allow the base address, range, and cycle duration of each chip select to be independently programmed. After reset, chip select ($\overline{CS0}$) responds to all accesses until the chip selects have been properly programmed. Four of the chip selects are multiplexed with the most significant address bits (A23–A20). The address mode (AMODE) input determines the functions of these outputs.

1.2.4 Parallel Ports

Two 8-bit parallel ports are provided. The port pins can be individually programmed to be inputs or outputs. If the pins are programmed to be inputs, the value on those pins can be read by accessing an on-board register. If the pins are programmed to be outputs, the pins will reflect the value programmed into another on-board register. The port B pins are multiplexed with four interrupt request and four interrupt acknowledge lines. The function of these pins is controlled by the internal registers.

1.2.5 Interrupt Controller

Seven input signals are provided to trigger an external interrupt, one for each of the seven priority levels supported. Each input can be programmed to be active high or active low. Seven separate outputs indicate the priority level of the interrupt being serviced. Interrupts at each priority level can be pre-programmed to go to the default service routine. For maximum flexibility, interrupts can be vectored to the correct service routine by the interrupting device.

1.2.6 Clock

To save on system costs, the MC68306 has an on-board oscillator that can be driven with a 16.67-MHz crystal. A bus clock output is provided by a CLKOUT pin. Alternatively, an

external 16.67-MHz oscillator can be used, with a tight skew between the input clock signal and the bus clock on the CLKOUT pin.

1.2.7 Bus Timeout Monitor

A bus timeout monitor is provided to automatically terminate and report as erroneous any bus cycle that is not normally terminated after a pre-programmed length of time. The user can program this timeout period to be up to 4096 clocks.

1.2.8 IEEE 1149.1 Test

To aid in system diagnostics, the MC68306 includes dedicated user-accessible test logic that is fully compliant with the IEEE 1149.1 standard for boundary scan testability, often referred to as JTAG (Joint Test Action Group).

SECTION 2
SIGNAL DESCRIPTION

This section contains a brief description of the input and output signals, with reference (if applicable) to other sections which give greater detail on its use. Figure 2-1 provides a detailed diagram showing the integrated peripherals and signals, and Tables 2-1–2-7 provides a quick reference for determining a signal's name, mnemonic, its use as an input or output, active state, and type identification.

NOTE

The terms **assertion** and **negation** will be used extensively. This is done to avoid confusion when dealing with a mixture of “active low” and “active high” signals. The term assert or assertion is used to indicate that a signal is active or true, independent of whether that level is represented by a high or low voltage. The term negate or negation is used to indicate that a signal is inactive or false.

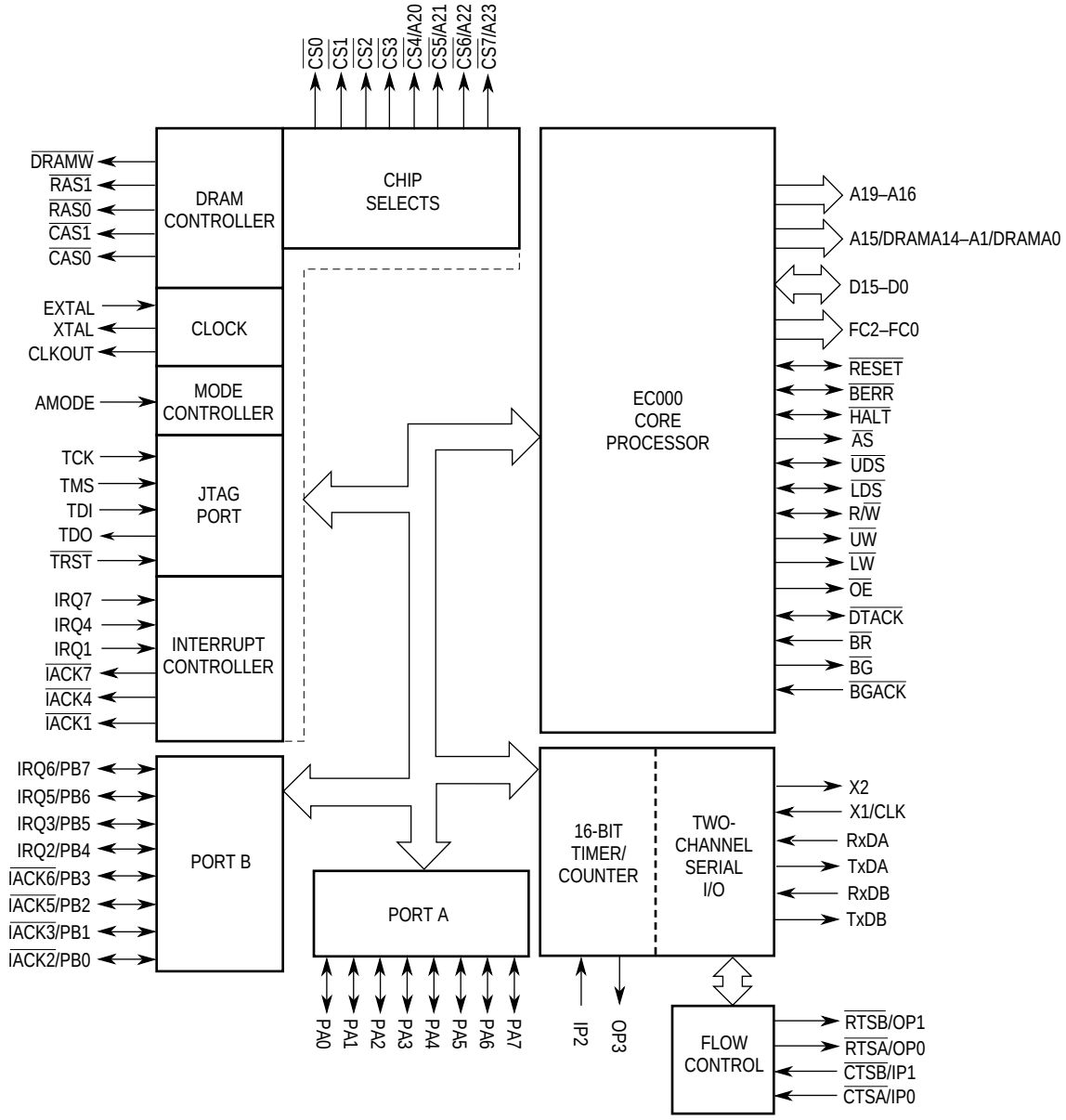


Figure 2-1. MC68306 Detailed Block Diagram

Table 2-1. Bus Signal Summary

Signal Name	Mnemonic	Input/ Output	Three-State During Bus Arbitration	Pullup Required
Address Signals	A23–A1	Output	Yes	
Address Strobe	$\overline{AS}$	Output	Yes	4.7 K
Bus Error	$\overline{BERR}$	I/O	—	2.2 K
Bus Grant	$\overline{BG}$	Output	No	
Bus Grant Acknowledge	$\overline{BGACK}$	Input	—	(1)
Bus Request	$\overline{BR}$	Input	—	(1)
Data Bus	D15–D0	I/O	Yes	
Data Transfer Acknowledge	$\overline{DTACK}$	I/O	—	2.2 K
DRAM Multiplexed Address14–0	DRAMA14–DRAMA0	Output	Yes	
Function Codes	FC2–FC0	Output	Yes	
Halt	$\overline{HALT}$	I/O	—	2.2 K
Lower Data Strobe	$\overline{LDS}$	I/O	Yes	4.7 K
Upper Data Strobe	$\overline{UDS}$	I/O	Yes	4.7 K
Lower-Byte Write Strobe	$\overline{LW}$	Output	No	
Upper-Byte Write Strobe	$\overline{UW}$	Output	No	
Output Enable	$\overline{OE}$	Output	No	
Read/Write	R/$\overline{W}$	Output	Yes	
Reset	$\overline{RESET}$	I/O	—	2.2 K

NOTES:

1. Pullup may be required, value depends on individual application. Must not be left floating.

Table 2-2. Chip Select Signal Summary

Signal Name	Mnemonic	Input/ Output	Three-State During Bus Arbitration	Pullup Required
Chip Select	$\overline{CS3}$–$\overline{CS0}$	Output	Yes	4.7 K
Chip Select 4–7/Address Port 23–20	$\overline{CS7}$–$\overline{CS4}$/ A23–A20	Output	Yes	4.7 K

Table 2-3. DRAM Controller Signal Summary

Signal Name	Mnemonic	Input/ Output	Three-State During Bus Arbitration	Pullup Required
Column Address Strobe	$\overline{CAS1}$–$\overline{CAS0}$	Output	Yes	4.7 K
Row Address Strobe	$\overline{RAS1}$–$\overline{RAS0}$	Output	Yes	4.7 K
DRAM Write Signal	$\overline{DRAMW}$	Output	Yes	

Table 2-4. Interrupt and Parallel Port Signal Summary

Signal Name	Mnemonic	Input/ Output	Three-State During Bus Arbitration	Pullup Required
Interrupt Request Level 7, 4, 1	IRQ7, IRQ4, IRQ1	Input	—	(2)
Interrupt Request Level 6/Port B 7	IRQ6/PB7	I/O	—	(2)
Interrupt Request Level 5/Port B 6	IRQ5/PB6	I/O	—	(2)
Interrupt Request Level 3/Port B 5	IRQ3/PB5	I/O	—	(2)
Interrupt Request Level 2/Port B 4	IRQ2/PB4	I/O	—	(2)
Interrupt Acknowledge 7, 4, 1	$\overline{\text{IACK7}}$, $\overline{\text{IACK4}}$, $\overline{\text{IACK1}}$	Output	—	
Interrupt Acknowledge 6/Port B 7	$\overline{\text{IACK6}}$/PB3	I/O	—	(2)
Interrupt Acknowledge 5/Port B 6	$\overline{\text{IACK5}}$/PB2	I/O	—	(2)
Interrupt Acknowledge 3/Port B 5	$\overline{\text{IACK3}}$/PB1	I/O	—	(2)
Interrupt Acknowledge 2/Port B 4	$\overline{\text{IACK2}}$/PB0	I/O	—	(2)
Port A	PA7–PA0	I/O	—	(2)

NOTES:
2. Pullup or pulldown may be required, value depends on individual application.

Table 2-5. Clock and Mode Control Signal Summary

Signal Name	Mnemonic	Input/ Output	Three-State During Bus Arbitration	Pullup Required
Crystal Oscillator or External Clock	EXTAL	Input	—	
Crystal Oscillator	XTAL	Output	—	
System Clock	CLKOUT	Output	No	
Address Mode	AMODE	Input	—	

Table 2-6. Serial Module Signal Summary

Signal Name	Mnemonic	Input/ Output	Three-State During Bus Arbitration	Pullup Required
Channel A Receiver Serial Data	RxDA	Input	—	
Channel A Transmitter Serial Data	TxDA	Output	No	
Channel B Receiver Serial Data	RxDB	Input	—	
Channel B Transmitter Serial Data	TxDB	Output	No	
Channel A Clear-to-Send	$\overline{\text{CTSA}}$/IP0	Input	—	(1)
Channel A Request-to-Send	$\overline{\text{RTSA}}$/OP0	Output	No	
Channel B Clear-to-Send	$\overline{\text{CTSB}}$/IP1	Input	—	(1)
Channel B Request-to-Send	$\overline{\text{RTSB}}$/OP1	Output	No	
Crystal Output	X2	Output	No	
Crystal Input or External Clock	X1/CLK	Input	—	
Parallel Input 2	IP2	Input	—	(1)
Parallel Output 3	OP3	Output	No	

NOTES:

1. Pullup may be required, value depends on individual application. Must not be left floating.

Table 2-7. JTAG Signal Summary

Signal Name	Mnemonic	Input/ Output	Three-State During Bus Arbitration	Pulldown Required
Test Clock	TCK	Input	—	
Test Data Input	TDI	Input	—	
Test Data Output	TDO	Output	—	
Test Mode Select	TMS	Input	—	
Test Reset	$\overline{\text{TRST}}$	Input	—	4.7 K (3)

NOTES:

3. Pin has internal pullup, but external pulldown may be required for correct initialization.

2.1 BUS SIGNALS

The following signals are used for the MC68306 bus.

2.1.1 Address Bus (A23–A1)

This 23-bit, unidirectional, three-state bus is capable of addressing 16 Mbytes of data. This bus provides the address for bus operation during all cycles except interrupt acknowledge cycles. During interrupt acknowledge cycles, address lines A1, A2, and A3 provide the level number of the interrupt being acknowledged, and address lines A23–A4

are driven to logic high. A23–A20 are only available in address mode (AMODE=0). A15–A1 are multiplexed with DRAM address.

2.1.2 Address Strobe ($\overline{AS}$)

Assertion of this three-state signal indicates that the information on the address bus is a valid address.

2.1.3 Bus Error ($\overline{BERR}$)

Assertion of this bi-directional, open-drain signal indicates a problem in the current bus cycle. The MC68306 can assert this signal to terminate a bus cycle when no external response is received. An external source can assert $\overline{BERR}$ to indicate a problem such as:

1. No response from a device
2. No interrupt vector number returned
3. An illegal access request rejected by a memory management unit
4. Some other application-dependent error

Either the processor retries the bus cycle or performs exception processing, as determined by interaction between the bus error signal and the halt signal.

2.1.4 Bus Request ($\overline{BR}$)

This input can be wire-ORed with bus request signals from all other devices that could be bus masters. Assertion of this signal indicates to the processor that some other device needs to become the bus master. Bus requests can be issued at any time during a bus cycle or between cycles.

2.1.5 Bus Grant ($\overline{BG}$)

This output signal indicates to all other potential bus master devices that the processor will relinquish bus control at the end of the current bus cycle.

2.1.6 Bus Grant Acknowledge ($\overline{BGACK}$)

Assertion of this input indicates that some other device has become the bus master. This signal should not be asserted until the following conditions are met:

1. A bus grant has been received.
2. Address strobe is inactive, which indicates that the microprocessor is not using the bus.
3. Data transfer acknowledge is inactive, which indicates that neither memory nor peripherals are using the bus.
4. Bus grant acknowledge is inactive, which indicates that no other device is claiming bus mastership.

$\overline{\text{BGACK}}$ can be negated (pulled high), and the MC68306 will operate in a two-wire bus arbitration system.

2.1.7 Data Bus (D15–D0)

This bi-directional, three-state bus is the general-purpose data path. It is 16 bits wide and can transfer and accept data of either word or byte length. During an interrupt acknowledge cycle, an external device can supply the interrupt vector number on data lines D7–D0.

2.1.8 Data Transfer Acknowledge ($\overline{\text{DTACK}}$)

Assertion of this bi-directional, open-drain signal indicates the completion of the data transfer. When the processor recognizes $\overline{\text{DTACK}}$ during a read cycle, data is latched, and the bus cycle is terminated. When $\overline{\text{DTACK}}$ is recognized during a write cycle, the bus cycle is terminated. The MC68306 generates $\overline{\text{DTACK}}$ for all internal cycles, DRAM cycles, and autovector IACK cycles, and can be programmed to generate $\overline{\text{DTACK}}$ for any chip select cycle. (Refer to **3.7 Asynchronous Operation** and **3.8 Synchronous Operation**.)

2.1.9 DRAM Multiplexed Address Bus (DRAMA14–DRAMA0)

These signals provide fifteen multiplexed address bits used during row address strobe.

2.1.10 Processor Function Codes (FC2–FC0)

These function code outputs indicate the mode (user or supervisor) and the address space type currently being accessed, as shown in Table 2-8. The function code outputs are valid whenever $\overline{\text{AS}}$ is asserted.

Table 2-8. Function Code Outputs

Function Code Output			
FC2	**FC1**	**FC0**	**Address Space Type**
Low	Low	Low	(Undefined, Reserved)
Low	Low	High	User Data
Low	High	Low	User Program
Low	High	High	(Undefined, Reserved)
High	Low	Low	(Undefined, Reserved)
High	Low	High	Supervisor Data
High	High	Low	Supervisor Program
High	High	High	CPU Space

2.1.11 Halt ($\overline{\text{HALT}}$)

External assertion of this bi-directional signal causes the processor to stop bus activity at the completion of the bus cycle for which the input met set-up time requirements (i.e., current or next cycle). This operation places all control signals in the inactive state. For

additional information about the interaction between $\overline{HALT}$ and $\overline{RESET}$, refer to **3.5 Reset Operation** and for more information on $\overline{HALT}$ and $\overline{BERR}$, refer to **3.4 Bus Error and Halt Operation**.

Processor assertion of $\overline{HALT}$ indicates a double bus fault condition. This condition is unrecoverable; the MC68306 must be externally reset to resume operation.

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

This three-state, bi-directional signal defines the data bus transfer as a read or write cycle. The R/$\overline{W}$ signal relates to the data strobe signals described in the following paragraphs.

2.1.13 Upper And Lower Data Strobes ($\overline{UDS}$, $\overline{LDS}$)

These three-state, bi-directional signals and R/$\overline{W}$ control the flow of data on the data bus. Table 2-9 lists the combinations of these signals, the corresponding data on the bus, and the $\overline{OE}$, $\overline{LW}$, and $\overline{UW}$ signals. When the R/$\overline{W}$ line is high, the processor reads from the data bus. When the R/$\overline{W}$ line is low, the processor drives the data bus. When another bus master controls the bus, the $\overline{UDS}$, $\overline{LDS}$, and R/$\overline{W}$ pins become inputs and the $\overline{OE}$, $\overline{LW}$, and $\overline{UW}$ signals are still decoded as shown in Table 2-9.

Table 2-9. Data Strobe Control of Data Bus

$\overline{UDS}$	$\overline{LDS}$	R/$\overline{W}$	D8–D15	D0–D7	$\overline{OE}$	$\overline{UW}$	$\overline{LW}$
High	High	—	No Valid Data	No Valid Data	High	High	High
Low	Low	High	Valid Data Bits 15–8	Valid Data Bits 7–0	Low	High	High
High	Low	High	No Valid Data	Valid Data Bits 7–0	Low	High	High
Low	High	High	Valid Data Bits 15–8	No Valid Data	Low	High	High
Low	Low	Low	Valid Data Bits 15–8	Valid Data Bits 7–0	High	Low	Low
High	Low	Low	Valid Data Bits 7–0*	Valid Data Bits 7–0	High	High	Low
Low	High	Low	Valid Data Bits 15–8	Valid Data Bits 15–8*	High	Low	High

*These conditions are a result of current implementation and may not appear on future devices.

2.1.14 Upper-Byte Write ($\overline{UW}$)

This signal is a combination of R/$\overline{W}$ low and $\overline{UDS}$ low for writing the upper-byte of a 16-bit port. This signal simplifies memory system design by explicitly signalling that data is valid on the upper portion of the data bus on a write operation. $\overline{UW}$ is also decoded for external bus masters.

2.1.15 Lower-Byte Write ($\overline{LW}$)

This signal is a combination of R/$\overline{W}$ low and $\overline{LDS}$ low for writing the lower-byte of a 16-bit port. This signal simplifies memory system design by explicitly signalling that data is valid on the lower portion of the data bus on a write operation. $\overline{LW}$ is also decoded for external bus masters.

2.1.16 Output Enable ($\overline{OE}$)

$\overline{OE}$ is a combination of R/$\overline{W}$ high and an active data strobe ($\overline{UDS}$ or $\overline{LDS}$). $\overline{OE}$ is also decoded for external bus masters.

2.1.17 Reset ($\overline{RESET}$)

The external assertion of this bi-directional, open-drain signal can start a system initialization sequence by resetting the processor. The processor assertion of $\overline{RESET}$ (from executing a RESET instruction) resets all external devices of a system without affecting the internal state of the processor. The interaction of internal and external $\overline{RESET}$, and the $\overline{HALT}$ signal is described in paragraph **3.5 Reset Operation.**

2.2 CHIP SELECT SIGNALS

These eight three-state signals provide address decodes with programmable base and range. $\overline{CS7}$–$\overline{CS4}$ are only available in chip select mode (AMODE bit =1). $\overline{CS3}$–$\overline{CS0}$ are always available.

2.3 DRAM CONTROLLER SIGNALS

The following signals are used to control an external DRAM for the MC68306.

2.3.1 Column Address Strobe ($\overline{CAS1}$–$\overline{CAS0}$)

These three-state signals provide column address strobe timing for external DRAM. CAS0 controls data lines D15–D8 and CAS1 controls D7–D0.

2.3.2 Row Address Strobe ($\overline{RAS1}$–$\overline{RAS0}$)

These three-state signals provide row address strobe timing for external DRAM. Each RAS controls a separate bank of DRAM.

2.3.3 DRAM Write Signal ($\overline{DRAMW}$)

This signal provides write control for external DRAM.

2.4 INTERRUPT CONTROL AND PARALLEL PORT SIGNALS

The following signals are used for interrupt control on the MC68306.

2.4.1 Interrupt Request (IRQ7–IRQ1)

Three input signals (IRQ7, IRQ4, IRQ1) notify the core processor of an interrupt request. Four additional interrupt request lines (IRQ6, IRQ5, IRQ3, and IRQ2) are shared with parallel port B pins and may be individually programmed as interrupts.

2.4.2 Interrupt Acknowledge ($\overline{IACK7}$–$\overline{IACK1}$)

Three output signals ($\overline{IACK7}$, $\overline{IACK4}$, $\overline{IACK1}$) indicate an interrupt acknowledge cycle. Four additional interrupt acknowledge lines ($\overline{IACK6}$, $\overline{IACK5}$, $\overline{IACK3}$, and $\overline{IACK2}$) are shared with parallel port B pins and may be individually programmed as interrupt acknowledges.

2.4.3 Port A Signals (PA7–PA0)

These eight pins serve as port A parallel input/output signals.

2.4.4 Port B (PB7–PB0)

These eight pins are shared with IRQ6, IRQ5, IRQ3, IRQ2 and $\overline{IACK6}$, $\overline{IACK5}$, $\overline{IACK3}$, $\overline{IACK2}$, and can be individually programmed to serve as port B parallel input/output signals.

2.5 CLOCK AND MODE CONTROL SIGNALS

These four pins are used to connect an external crystal to the on-chip oscillator and define the four multifunction pins.

2.5.1 Crystal Oscillator (EXTAL, XTAL)

These two pins are the connections for an external crystal to the internal oscillator circuit. If an external oscillator is used, it should be connected to EXTAL, with XTAL left open, and must drive CMOS levels. A crystal or clock input must be supplied at all times.

2.5.2 Clock Out (CLKOUT)

This output signal is the system clock output and is used as the bus timing reference by external devices.

2.5.3 Address Mode (AMODE)

This input signal provides mode control for the multi-function chip select pins. When set to zero, A23–A20 is selected and when set to one, $\overline{CS7}$–$\overline{CS4}$ is selected. The mode selection is static: AMODE is latched at the end of any system reset.

2.6 SERIAL MODULE SIGNALS

The following paragraphs describe the signals used by the serial module on the MC68306.

2.6.1 Channel A Receiver Serial-Data Input (RxDA)

This signal is the receiver serial-data input for channel A. The least-significant bit is received first. Data on this pin is sampled on the rising edge of the programmed clock source.

2.6.2 Channel A Transmitter Serial-Data Output (TxDA)

This signal is the transmitter serial-data output for channel A. The least-significant bit is transmitted first. This output is held high (mark condition) when the transmitter is disabled, idle, or operating in the local loopback mode. (Mark is high and space is low). Data is shifted out this pin on the falling edge of the programmed clock source.

2.6.3 Channel B Receiver Serial-Data Input (RxDB)

This signal is the receiver serial-data input for channel B. The least-significant bit is received first. Data on this pin is sampled on the rising edge of the programmed clock source.

2.6.4 Channel B Transmitter Serial-Data Output (TxDB)

This signal is the transmitter serial-data output for channel B. The least-significant bit is transmitted first. This output is held high (mark condition) when the transmitter is disabled, idle, or operating in the local loopback mode. Data is shifted out of this pin on the falling edge of the programmed clock source.

2.6.5 $\overline{\text{CTSA}}$

This input can be used as the channel A clear-to-send active low input ($\overline{\text{CTSA}}$) or general-purpose input (IP0). A change-of-state detector is also associated with this input.

2.6.6 $\overline{\text{RTSA}}$

This output can be used as the channel A active low request-to-send ($\overline{\text{RTSA}}$) output, or a general-purpose output (OP0). When used as $\overline{\text{RTSA}}$, it is automatically negated and reasserted by either the receiver or transmitter.

2.6.7 $\overline{\text{CTSB}}$

This input can be used as the channel B clear-to-send active low input ($\overline{\text{CTSB}}$) or general-purpose input. A change-of-state detector is also associated with this input.

2.6.8 $\overline{\text{RTSB}}$

This output can be used as a general-purpose output or the channel B active low request-to-send ($\overline{\text{RTSB}}$) output. When used for this function, it is automatically negated and reasserted by either the receiver or transmitter.

2.6.9 Crystal Oscillator (X1/CLK, X2)

These two pins are the connections for an external crystal to the internal oscillator circuit. If an external oscillator is used, it should be connected to X1/CLK, with X2 left floating, and must drive CMOS levels. A crystal or clock input must be supplied at all times.

2.6.10 IP2

This input can be used as a general-purpose input, the channel B receiver external clock input (RxCB), or the counter/timer external clock input. When this input is used as the external clock by the receiver, the received data is sampled on the rising edge of the clock. A change-of-state detector is also associated with this input.

2.6.11 OP3

This output can be used as a general-purpose output, the open-drain active low counter-ready output, the open-drain timer output, the channel B transmitter 1X-clock output, or the channel B receiver 1X-clock output.

2.7 JTAG PORT TEST SIGNALS

The following signals are used with the on-chip test logic defined by the IEEE 1149.1 standard. See **IEEE 1149.1 Test Access Port** for more information on the use of these signals.

2.7.1 Test Clock (TCK)

This input provides a clock for on-chip test logic defined by the IEEE 1149.1 standard.

2.7.2 Test Mode Select (TMS)

This input controls test mode operations for on-chip test logic defined by the IEEE 1149.1 standard. Connecting TMS to V_{CC} disables the test controller, making all JTAG circuits transparent to the system.

2.7.3 Test Data In (TDI)

This input is used for serial test instructions and test data for on-chip test logic defined by the IEEE 1149.1 standard.

2.7.4 Test Data Out (TDO)

This output is used for serial test instructions and test data for on-chip test logic defined by the IEEE 1149.1 standard.

2.7.5 Test Reset ($\overline{\text{TRST}}$)

This input is the master reset for on-chip test logic defined by the IEEE 1149.1 standard.

SECTION 3
68000 BUS OPERATION DESCRIPTION

This section describes control signal and bus operation during data transfer operations, bus arbitration, bus error and halt conditions, and reset operation.

NOTE

The terms **assertion** and **negation** are used extensively in this manual to avoid confusion when describing a mixture of "active-low" and "active-high" signals. The term assert or assertion is used to indicate that a signal is active or true, independently of whether that level is represented by a high or low voltage. The term negate or negation is used to indicate that a signal is inactive or false.

3.1 DATA TRANSFER OPERATIONS

Transfer of data between devices involves the following signals:

1. Address bus A1 through A31
2. Data bus D0 through D7 and/or D8 through D15
3. Control signals

The address and data buses are separate parallel buses used to transfer data using an asynchronous bus structure. In all cases, the bus master must deskew all signals it issues at both the start and end of a bus cycle. In addition, the bus master must deskew the acknowledge and data signals from the slave device.

The following paragraphs describe the read, write, read-modify-write, and CPU space cycles. The indivisible read-modify-write cycle implements interlocked multiprocessor communications. A CPU space cycle is a special processor cycle.

3.1.1 Read Cycle

During a read cycle, the processor receives either one or two bytes of data from the memory or from a peripheral device. If the instruction specifies a word or long-word operation, the processor reads both upper and lower bytes simultaneously by asserting both upper and lower data strobes. A long-word read is accomplished by two consecutive word reads. When the instruction specifies byte operation, the processor uses the internal A0 bit to determine which byte to read and issues the appropriate data strobe. When A0 is zero, the upper data strobe is issued; when A0 is one, the lower data strobe is issued. When the data is received, the processor internally positions the byte appropriately.

The word read cycle flowchart is shown in Figure 3-1. The byte read cycle flowchart is shown in Figure 3-2. The read and write cycle timing is shown in Figure 3-3. Figure 3-4 shows the word and byte read cycle timing diagram.

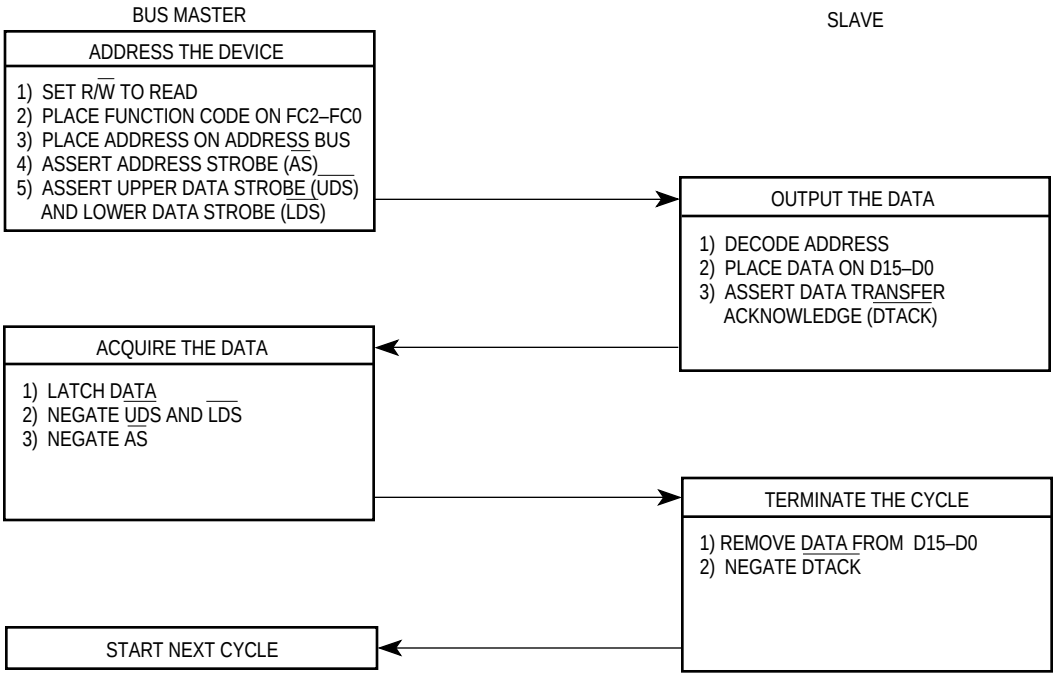


Figure 3-1. Word Read Cycle Flowchart

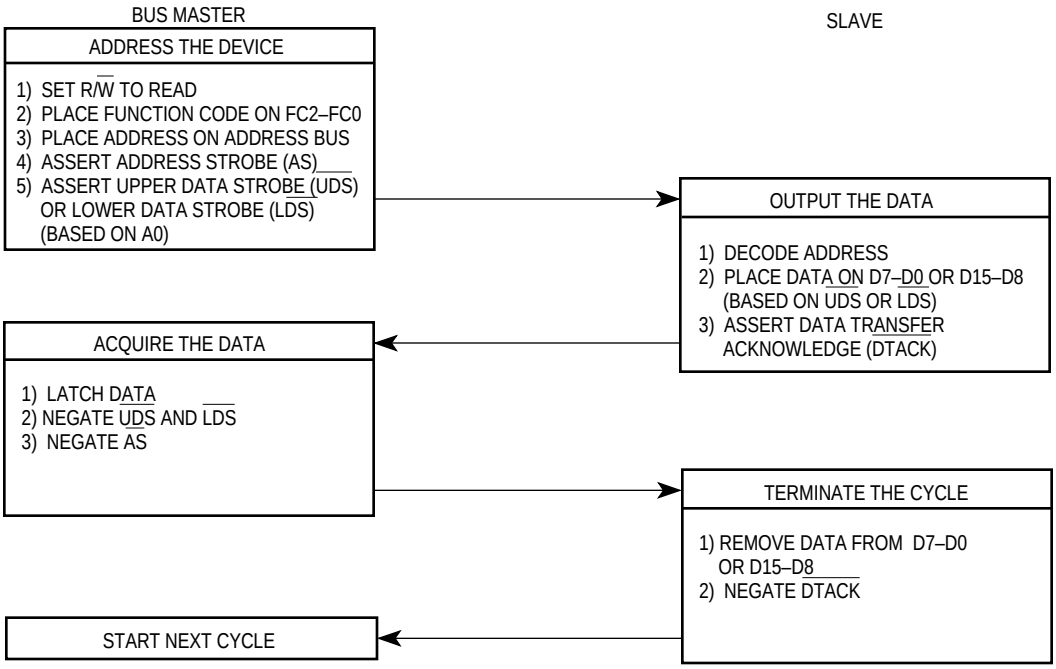


Figure 3-2. Byte Read Cycle Flowchart

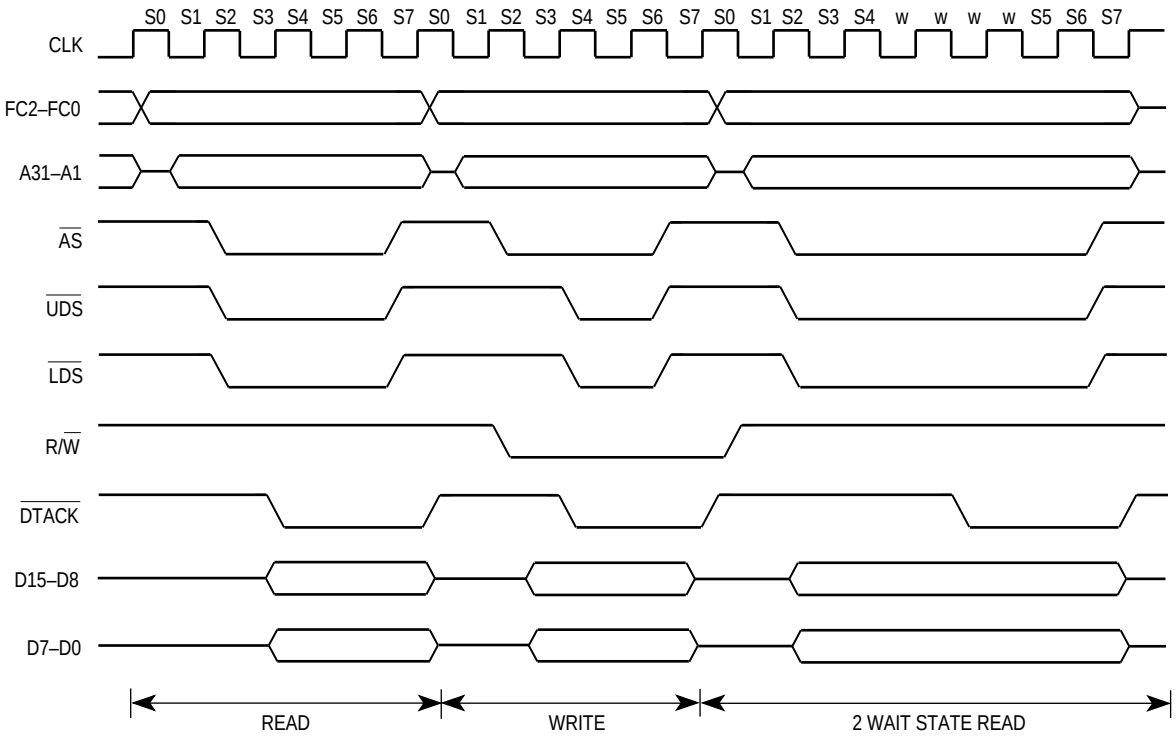


Figure 3-3. Read and Write Cycle Timing Diagram

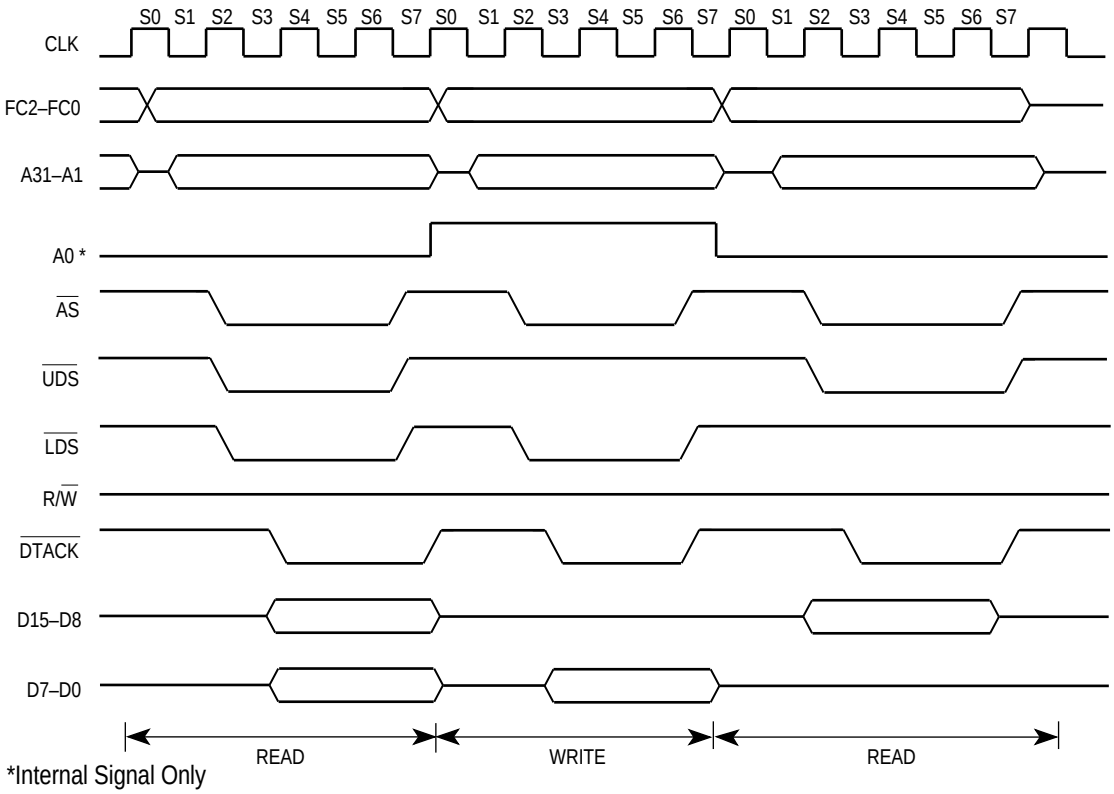


Figure 3-4. Word and Byte Read Cycle Timing Diagram

A bus cycle consists of eight states. 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 processor places valid function codes on FC0–FC2, a valid address on the bus, and drives R/$\overline{W}$ high to identify a read cycle.

STATE 1 During state 1 (S1), no bus signals are altered.

STATE 2 On the rising edge of state 2 (S2), the processor asserts $\overline{AS}$ and $\overline{UDS}$/$\overline{LDS}$.

STATE 3 During state 3 (S3), no bus signals are altered.

STATE 4 During state 4 (S4), the processor waits for a cycle termination signal ($\overline{DTACK}$ or $\overline{BERR}$). If neither termination signal is asserted before the falling edge at the end of S4, the processor inserts wait states (full clock cycles) until either $\overline{DTACK}$ or $\overline{BERR}$ is asserted.

Case 1: $\overline{DTACK}$ received, with or without $\overline{BERR}$.

STATE 5 During state 5 (S5), no bus signals are altered.

STATE 6 Sometime between state 2 (S2) and state 6 (S6), data from the device is driven onto the data bus.

STATE 7 On the falling edge of the clock entering state 7 (S7), the processor latches data from the addressed device and negates $\overline{AS}$ and $\overline{UDS}$, $\overline{LDS}$. The device negates $\overline{DTACK}$ or $\overline{BERR}$ at this time.

Case 2: $\overline{BERR}$ received without $\overline{DTACK}$.

STATE 5 During state 5 (S5), no bus signals are altered.

STATE 6 During state 6 (S6), no bus signals are altered.

STATE 7 During state 7 (S7), no bus signals are altered.

STATE 8 During state 8 (S8), no bus signals are altered.

STATE 9 $\overline{AS}$ and $\overline{UDS}$/$\overline{LDS}$ negated. Slave negates $\overline{BERR}$.

3.1.2 Write Cycle

During a write cycle, the processor sends bytes of data to the memory or peripheral device. If the instruction specifies a word or long-word operation, the processor issues both $\overline{UDS}$ and $\overline{LDS}$ and writes both bytes. A long-word write is accomplished by two consecutive word writes. When the instruction specifies a byte operation, the processor uses the internal A0 bit to determine which byte to write and issues the appropriate data

strobe. When the A0 bit equals zero, $\overline{UDS}$ is asserted; when the A0 bit equals one, $\overline{LDS}$ is asserted.

The word write cycle flowchart is shown in Figure 3-5. The byte write cycle flowchart is shown in Figure 3-6. The word and byte write cycle timing is shown in Figure 3-7.

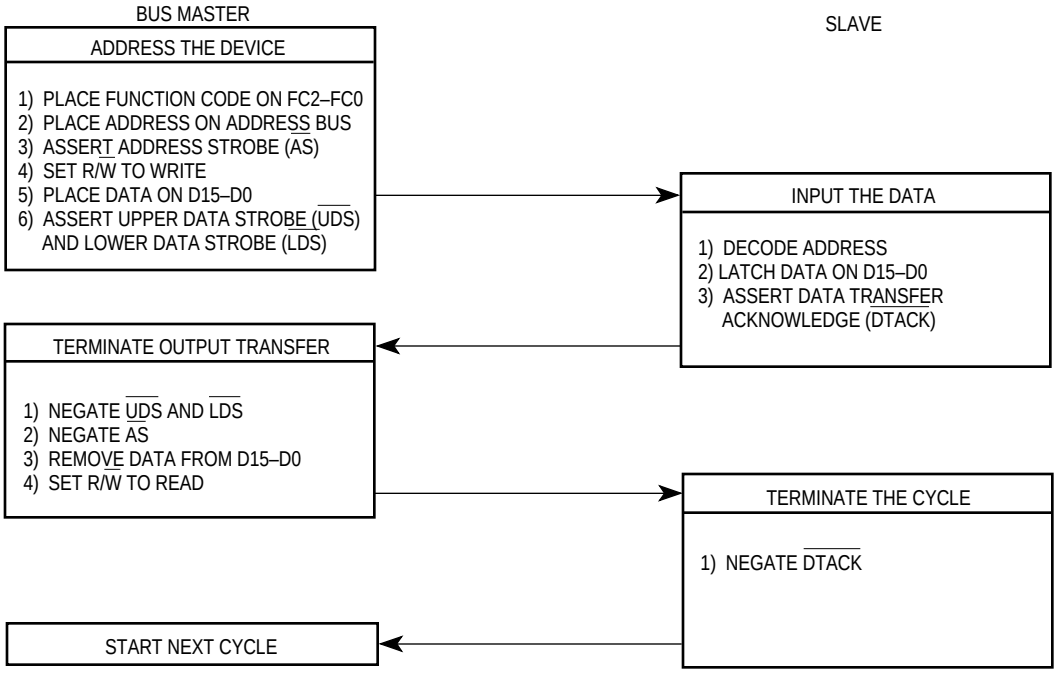


Figure 3-5. Word Write Cycle Flowchart

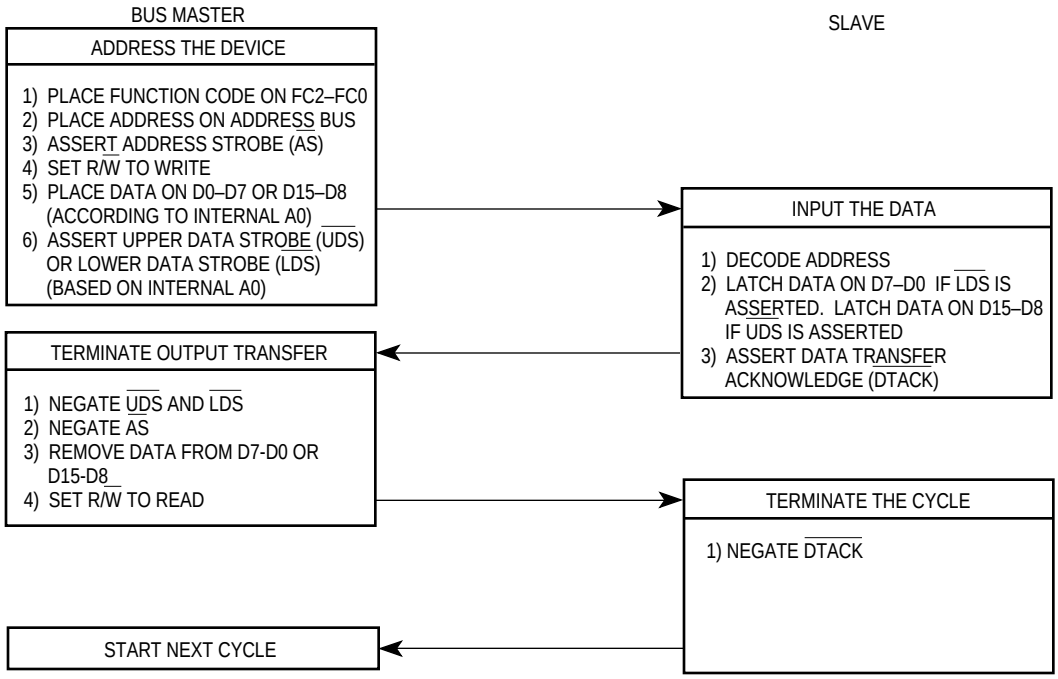


Figure 3-6. Byte Write Cycle Flowchart

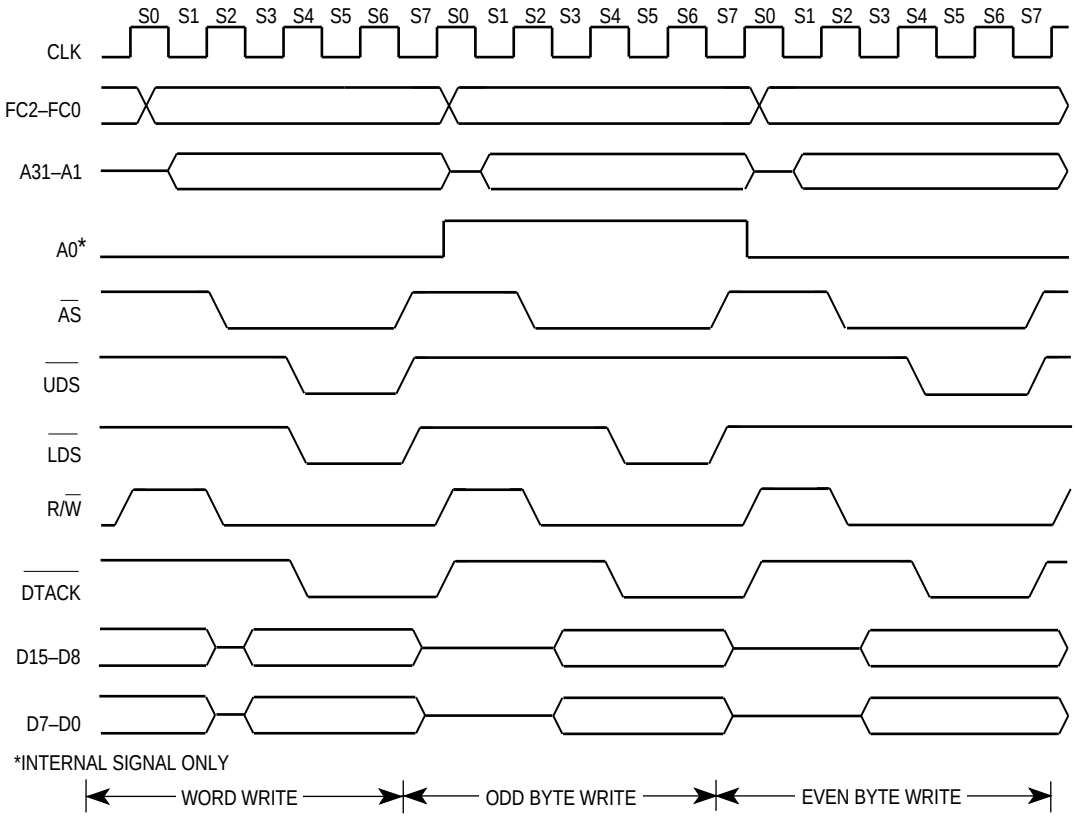


Figure 3-7. Word and Byte Write Cycle Timing Diagram

The descriptions of the eight states of a write cycle are as follows:

STATE 0 The write cycle starts in S0. The processor places valid function codes on FC2–FC0, a valid address on the address bus, and drives R/$\overline{W}$ high (if a preceding write cycle has left R/$\overline{W}$ low).

STATE 1 During S1, no bus signals are altered.

STATE 2 On the rising edge of S2, the processor asserts $\overline{AS}$ and drives R/$\overline{W}$ low.

STATE 3 During S3, the data bus is driven out of the high-impedance state as the data to be written is placed on the bus.

STATE 4 At the rising edge of S4, the processor asserts $\overline{UDS}$ and/or $\overline{LDS}$;. The processor waits for a cycle termination signal ($\overline{DTACK}$ or $\overline{BERR}$). If neither termination signal is asserted before the falling edge at the end of S4, the processor inserts wait states (full clock cycles) until either $\overline{DTACK}$ or $\overline{BERR}$ is asserted.

Case 1: $\overline{DTACK}$ received, with or without $\overline{BERR}$.

STATE 5 During S5, no bus signals are altered.

STATE 6 During S6, no bus signals are altered.

STATE 7 On the falling edge of the clock entering S7, the processor negates $\overline{AS}$, $\overline{UDS}$, and/or $\overline{LDS}$. As the clock rises at the end of S7, the processor places the data bus in the high-impedance state, and drives R/$\overline{W}$ high. The device negates $\overline{DTACK}$ or $\overline{BERR}$ at this time.

Case 2: $\overline{BERR}$ received without $\overline{DTACK}$.

STATE 5 During state 5 (S5), no bus signals are altered.

STATE 6 During state 6 (S6), no bus signals are altered.

STATE 7 During state 7 (S7), no bus signals are altered.

STATE 8 During state 8 (S8), no bus signals are altered.

STATE 9 $\overline{AS}$ and $\overline{UDS}$/$\overline{LDS}$ negated. Slave negates $\overline{BERR}$. At the end of S9, three-state data and drive R/$\overline{W}$ high.

3.1.3 Read-Modify-Write Cycle

The read-modify-write cycle performs a read operation, modifies the data in the arithmetic logic unit, and writes the data back to the same address. The address strobe ($\overline{AS}$) remains asserted throughout the entire cycle, making the cycle indivisible. The test and set (TAS) instruction uses this cycle to provide a signaling capability without deadlock between processors in a multiprocessing environment. The TAS instruction (the only instruction

that uses the read-modify-write cycle) only operates on bytes. Thus, all read-modify-write cycles are byte operations. The read-modify-write flowchart is shown in Figure 3-8 and the timing diagram is shown in Figure 3-9.

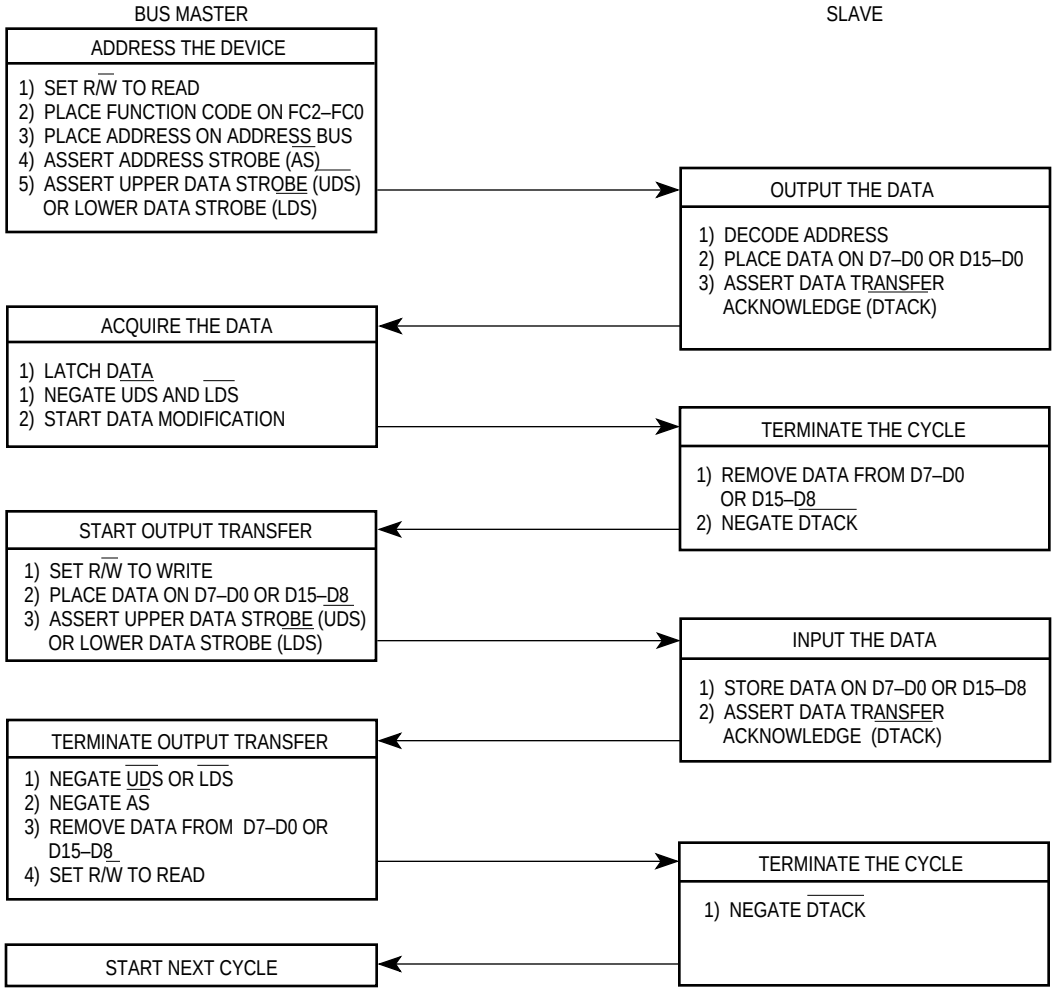


Figure 3-8. Read-Modify-Write Cycle Flowchart

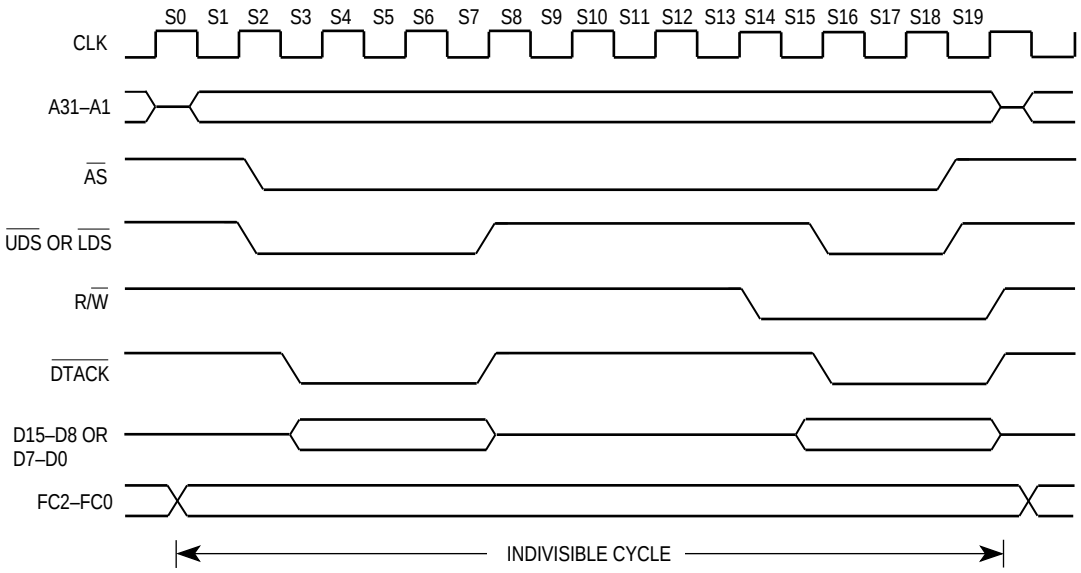


Figure 3-9. Read-Modify-Write Cycle Timing Diagram

The descriptions of the read-modify-write cycle states are as follows:

STATE 0 — The read cycle starts in S0. The processor places valid function codes on FC2–FC0, a valid address on the address bus, and drives R/$\overline{W}$ high to identify a read cycle.

STATE 1 — During S1, no bus signals are altered.

STATE 2 — On the rising edge of S2, the processor asserts $\overline{AS}$ and $\overline{UDS}$/$\overline{LDS}$.

STATE 3 — During S3, no bus signals are altered.

STATE 4 — During S4, the processor waits for a cycle termination signal ($\overline{DTACK}$ or $\overline{BERR}$). If neither termination signal is asserted before the falling edge at the end of S4, the processor inserts wait states (full clock cycles) until either $\overline{DTACK}$ or $\overline{BERR}$ is asserted.

Case R1: $\overline{DTACK}$ only.

STATE 5 — During S5, no bus signals are altered.

STATE 6 — During S6, data from the device are driven onto the data bus.

STATE 7 — On the falling edge of the clock entering S7, the processor accepts data from the device and negates $\overline{UDS}$/$\overline{LDS}$. The device negates $\overline{DTACK}$ at this time.

STATES 8–11 — The bus signals are unaltered during S8–S11, during which the arithmetic logic unit makes appropriate modifications to the data.

STATE 12 The write portion of the cycle starts in S12. The valid function codes on FC2–FC0, the address bus lines, $\overline{AS}$, and R/$\overline{W}$ remain unaltered.

STATE 13 During S13, no bus signals are altered.

STATE 14 On the rising edge of S14, the processor drives R/$\overline{W}$ low.

STATE 15 During S15, the data bus is driven out of the high-impedance state as the data to be written are placed on the bus.

STATE 16 At the rising edge of S16, the processor asserts $\overline{UDS}$/$\overline{LDS}$. The processor waits for $\overline{DTACK}$ or $\overline{BERR}$. If neither termination signal is asserted before the falling edge at the close of S16, the processor inserts wait states (full clock cycles) until either $\overline{DTACK}$ or $\overline{BERR}$ is asserted.

Case W1: $\overline{DTACK}$ with or without $\overline{BERR}$.

STATE 17 During S17, no bus signals are altered.

STATE 18 During S18, no bus signals are altered.

STATE 19 On the falling edge of the clock entering S19, the processor negates $\overline{AS}$ and $\overline{UDS}$/$\overline{LDS}$. As the clock rises at the end of S19, the processor places the data bus in the high-impedance state, and drives R/$\overline{W}$ high. The device negates $\overline{DTACK}$ or $\overline{BERR}$ at this time.

Case R2: $\overline{DTACK}$ and $\overline{BERR}$ on read.

STATE 5 During S5, no bus signals are altered.

STATE 6 During S6, no bus signals are altered, and data from the device is ignored.

STATE 7 $\overline{AS}$ and $\overline{UDS}$/$\overline{LDS}$ are negated. The cycle terminates without the write portion.

Case R3: $\overline{BERR}$ only on read.

STATE 5 During S5, no bus signals are altered.

STATE 6 During S6, no bus signals are altered..

STATE 7 During S7, no bus signals are altered.

STATE 8 During S8, no bus signals are altered.

STATE 9 $\overline{AS}$ and $\overline{UDS}$/$\overline{LDS}$ are negated. The cycle terminates without the write portion.

Case W2: $\overline{BERR}$ only on write.

STATE 17 During S17, no bus signals are altered.

STATE 18 During S18, no bus signals are altered.

STATE 19 During S19, no bus signals are altered.

STATE 20 During S20. no bus signals are altered.

STATE 21 The processor negates $\overline{AS}$ and $\overline{UDS}$/$\overline{LDS}$.

3.1.4 CPU Space Cycle

A CPU space cycle, indicated when the function codes are all high, is a special processor cycle. In the 68EC000 core, CPU space is used only for interrupt acknowledge cycles. Figure 3-10 shows the encoding of an interrupt acknowledge cycle.

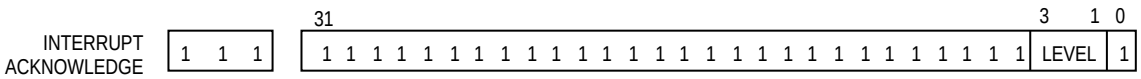
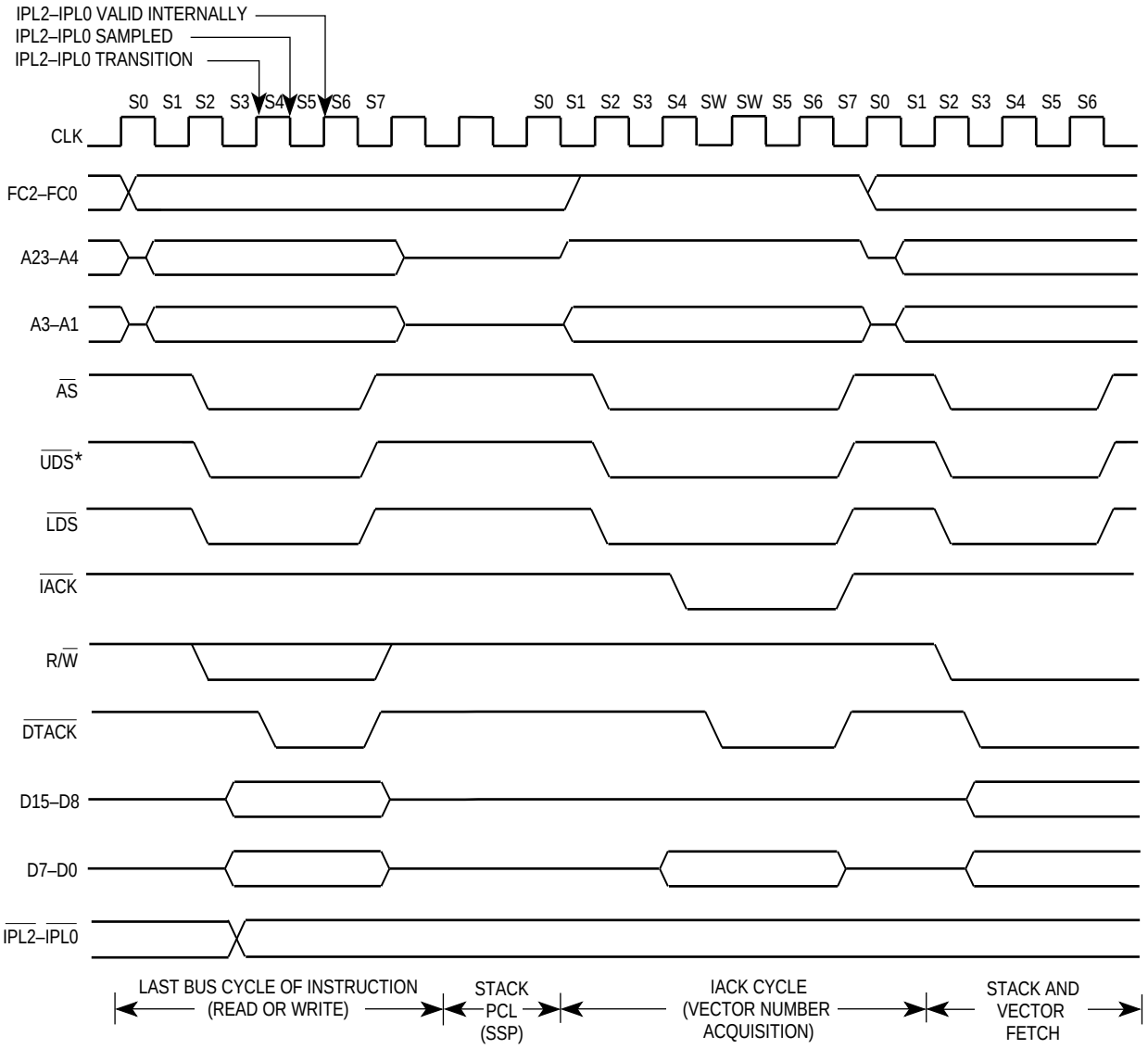


Figure 3-10. Interrupt Acknowledge Cycle

The interrupt acknowledge cycle places the level of the interrupt being acknowledged on address bits A3–A1 and drives all other address lines high. The interrupt acknowledge cycle reads a vector number when the device places a vector number on the data bus.

The timing diagram for an interrupt acknowledge cycle is shown in Figure 3-11.



*Although a vector number is one byte, both data strobes are asserted due to the microcode used for exception processing. The processor does not recognize anything on data lines D8 through D15 at this time.

Figure 3-11. Interrupt Acknowledge Cycle Timing Diagram

3.2 BUS ARBITRATION

Bus arbitration is a technique used by bus master devices to request, to be granted, and to acknowledge bus mastership. Bus arbitration consists of the following:

1. Asserting a bus mastership request
2. Receiving a grant indicating that the bus is available at the end of the current cycle
3. Acknowledging that mastership has been assumed

Figure 3-12 is a flowchart showing the bus arbitration cycle of the EC000 core. Figure 3-13 is a timing diagram of the bus arbitration cycle charted in Figure 3-12. This technique allows processing of bus requests during data transfer cycles.

There are two ways to arbitrate the bus, 3-wire and 2-wire bus arbitration. The EC000 core can do either 2-wire or 3-wire bus arbitration. Figures 3-12 and 3-14 show 3-wire bus arbitration and Figures 3-13 and 5-15 show 2-wire bus arbitration. $\overline{\text{BGACK}}$ must be pulled high for 2-wire bus arbitration.

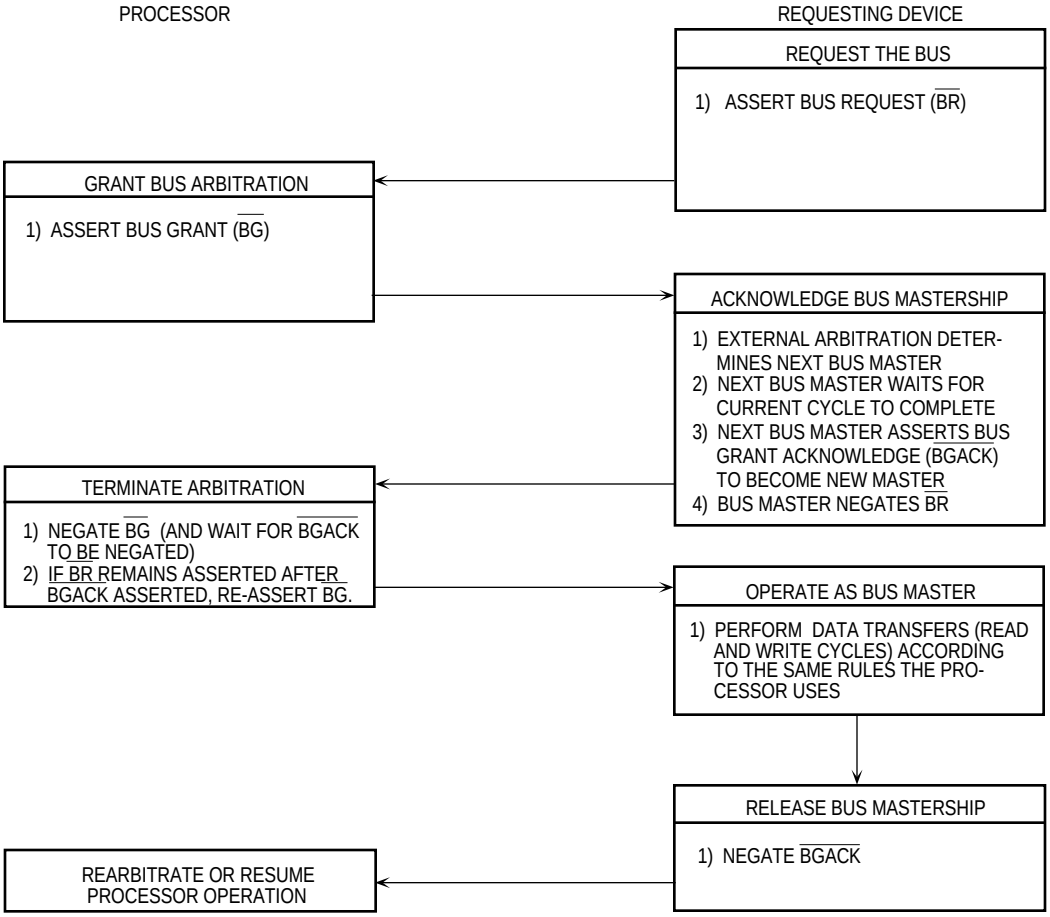


Figure 3-12. Three-Wire Bus Arbitration Cycle Flowchart

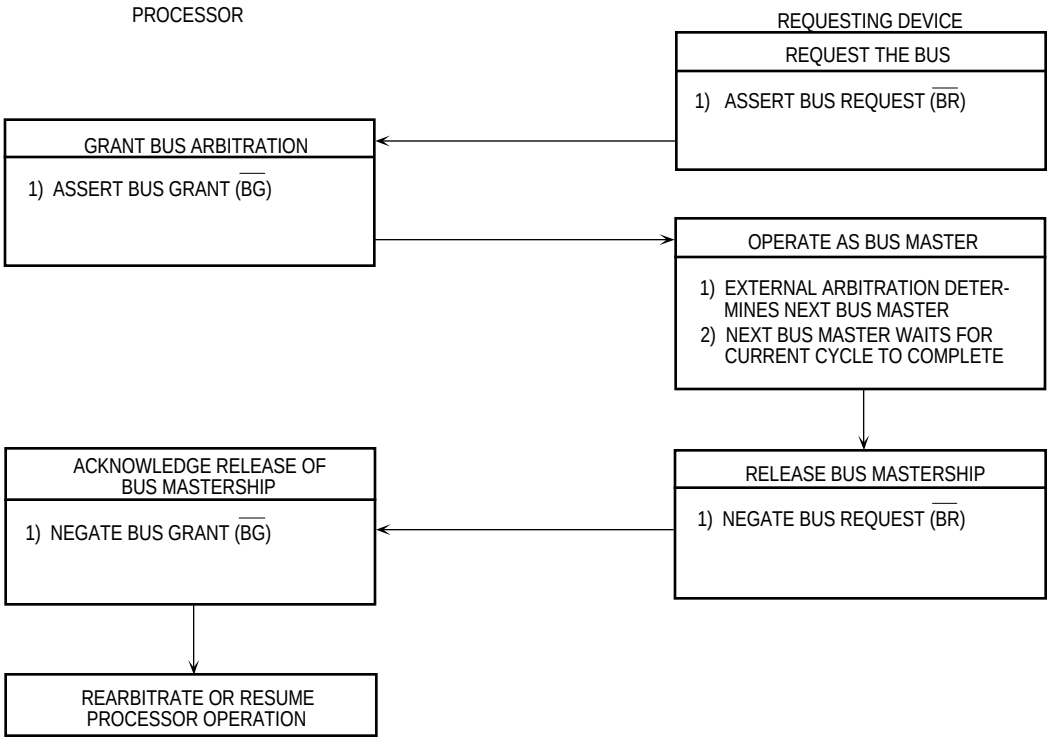


Figure 3-13. Two-Wire Bus Arbitration Cycle Flowchart

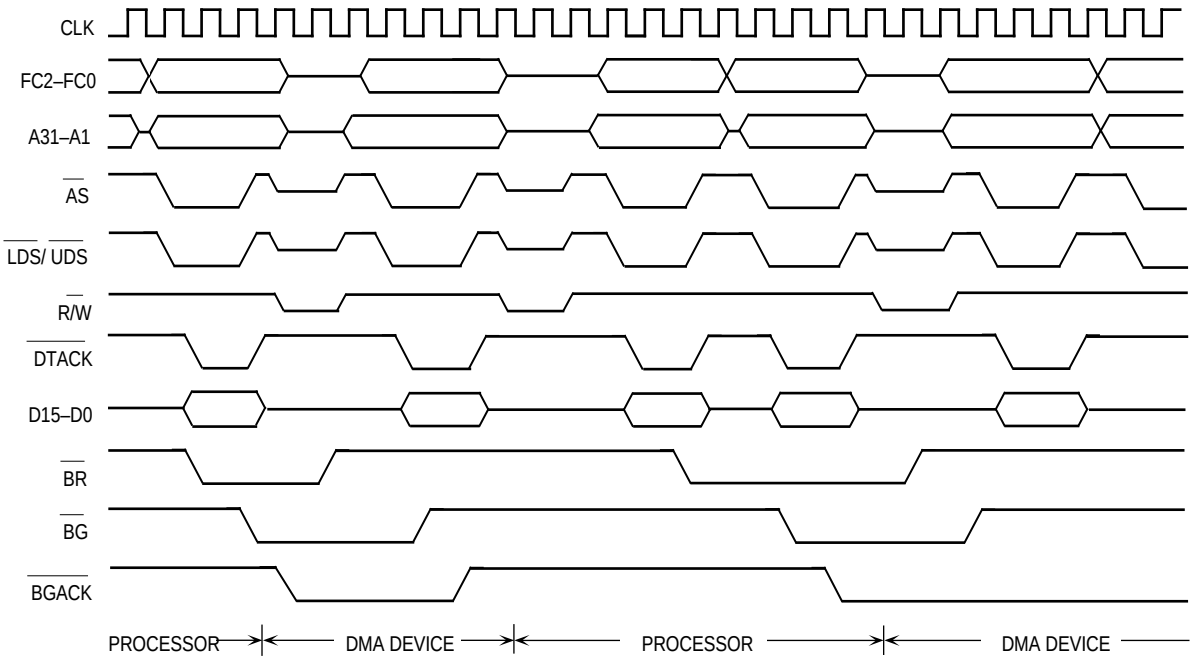


Figure 3-14. Three-Wire Bus Arbitration Timing Diagram

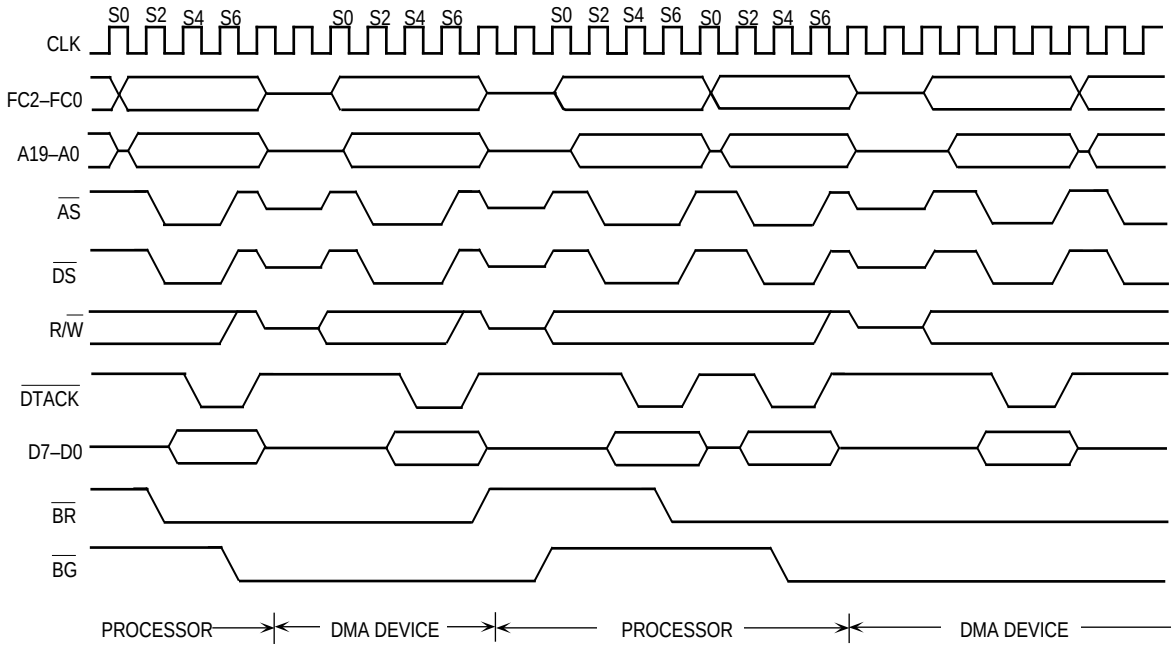


Figure 3-15. Two-Wire Bus Arbitration Timing Diagram

The timing diagram in Figure 3-14 shows that the bus request is negated at the time that an acknowledge is asserted. This type of operation applies to a system consisting of a processor and one other device capable of becoming bus master. In systems having several devices that can be bus masters, bus request lines from these devices can be wire-ORed at the processor, and more than one bus request signal could occur.

The bus grant signal is negated a few clock cycles after the assertion of the bus grant acknowledge signal. However, if bus requests are pending, the processor reasserts bus grant for another request a few clock cycles after bus grant (for the previous request) is negated. In response to this additional assertion of bus grant, external arbitration circuitry selects the next bus master before the current bus master has completed the bus activity.

The timing diagram in Figure 3-15 also applies to a system consisting of a processor and one other device capable of becoming bus master. Since the 2-wire bus arbitration scheme does not use a bus grant acknowledge signal, the external master must continue to assert $\overline{BR}$ until it has completed its bus activity. The processor negates bus grant when $\overline{BR}$ is negated.

3.2.1 Requesting the Bus

External devices capable of becoming bus masters assert $\overline{BR}$ to request the bus. This signal can be wire-ORed (not necessarily constructed from open-collector devices) from any of the devices in the system that can become bus master. The processor, which is at a lower bus priority level than the external devices, relinquishes the bus after it completes the current bus cycle.

3.2.2 Receiving the Bus Grant

The processor asserts $\overline{BG}$ as soon as possible. Normally, this process immediately follows internal synchronization, except when the processor has made an internal decision to execute the next bus cycle but has not yet asserted $\overline{AS}$ for that cycle. In this case, $\overline{BG}$ is delayed until $\overline{AS}$ is asserted to indicate to external devices that a bus cycle is in progress.

$\overline{BG}$ can be routed through a daisy-chained network or through a specific priority-encoded network. Any method of external arbitration that observes the protocol can be used.

3.2.3 Acknowledgment of Mastership (3-Wire Bus Arbitration Only)

Upon receiving $\overline{BG}$, the requesting device waits until $\overline{AS}$, $\overline{DTACK}$, and $\overline{BGACK}$ are negated before asserting $\overline{BGACK}$. The negation of $\overline{AS}$ indicates that the previous bus master has completed its cycle. (No device is allowed to assume bus mastership while $\overline{AS}$ is asserted.) The negation of $\overline{BGACK}$ indicates that the previous master has released the bus. The negation of $\overline{DTACK}$ indicates that the previous slave has terminated the connection to the previous master. (In some applications, $\overline{DTACK}$ might not be included in this function; general-purpose devices would be connected using $\overline{AS}$ only.) When $\overline{BGACK}$ is asserted, the asserting device is bus master until it negates $\overline{BGACK}$. $\overline{BGACK}$ should not be negated until after the bus cycle(s) is complete. A device relinquishes control of the bus by negating $\overline{BGACK}$.

The bus request from the granted device should be negated after $\overline{BGACK}$ is asserted. If another bus request is pending, $\overline{BG}$ is reasserted within a few clocks, as described in **3.3 Bus Arbitration Control**. The processor does not perform any external bus cycles before reasserting $\overline{BG}$.

3.3 BUS ARBITRATION CONTROL

All asynchronous bus arbitration signals to the processor are synchronized before being used internally. As shown in Figure 3-16, synchronization requires a maximum of one and a half cycles of the system clock. The input asynchronous signal is sampled on the falling edge of the clock and is valid internally after the next rising edge.

This synchronization scheme is used for all other asynchronous inputs also: $\overline{RESET}$, $\overline{HALT}$, $\overline{DTACK}$, $\overline{BERR}$, $\overline{IPL2}$–$\overline{IPL0}$.

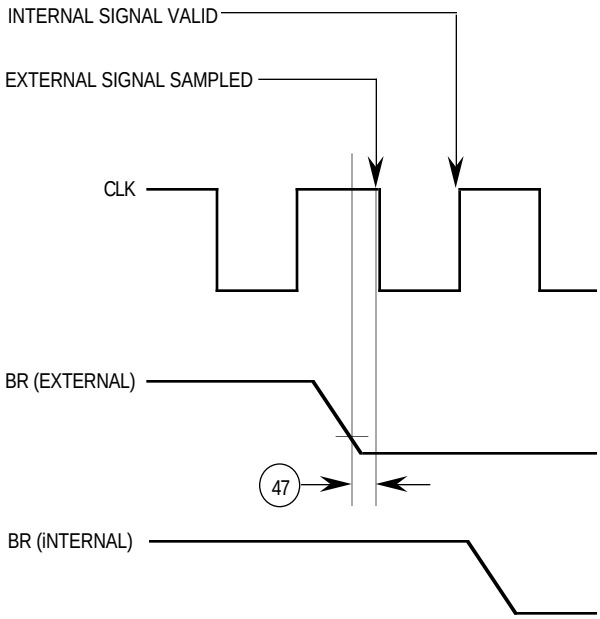
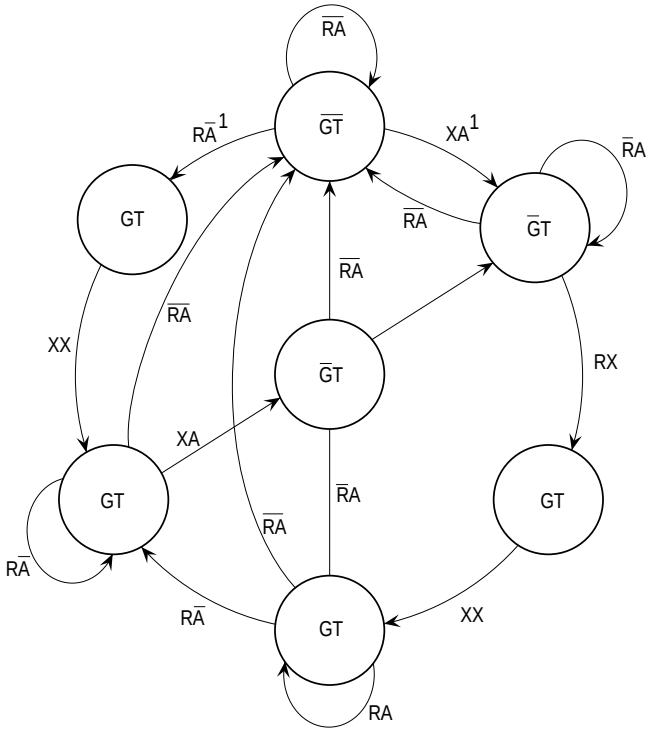


Figure 3-16. External Asynchronous Signal Synchronization

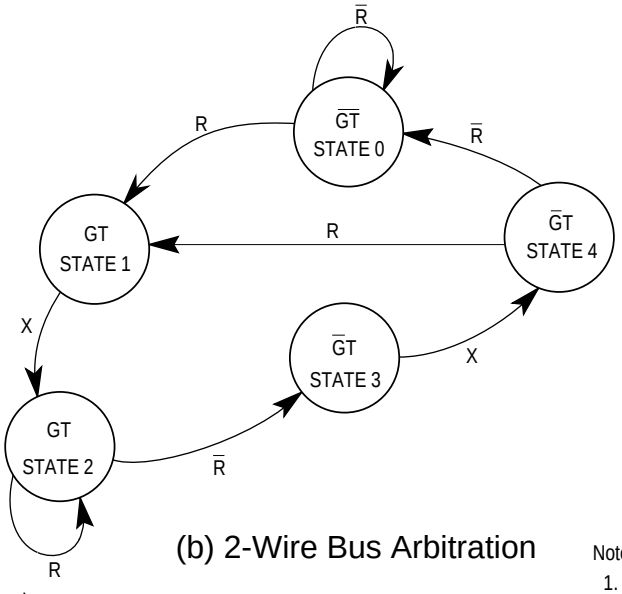
Bus arbitration control is implemented with a finite state machine (see Figure 3-17). In Figure 3-17, input signals R and A are the internally synchronized versions of $\overline{BR}$ and $\overline{BGACK}$. The $\overline{BG}$ output is shown as G, and the internal three-state control signal is shown as T. If T is true, the address, data, and control buses are placed in the high-impedance state when $\overline{AS}$ is negated. All signals are shown in positive logic (active high), regardless of their true active voltage level. State changes (valid outputs) occur on the next rising edge of the clock after the internal signal is valid.

A timing diagram of the bus arbitration sequence during a processor bus cycle is shown in Figure 3-18. The bus arbitration timing while the bus is inactive (e.g., the processor is performing internal operations for a multiply instruction) is shown in Figure 3-19.

When a bus request is made after the MPU has begun a bus cycle and before $\overline{AS}$ has been asserted (S0), the special sequence shown in Figure 3-20 applies. Instead of being asserted on the next rising edge of clock, $\overline{BG}$ is delayed until the second rising edge following its internal assertion.



(a) 3-Wire Bus Arbitration



(b) 2-Wire Bus Arbitration

R = Bus Request Internal
A = Bus Grant Acknowledge Internal
G = Bus Grant
T = Three-state Control to Bus Control Logic
X = Don't Care

Notes:

1. State machine will not change if the bus is S0 or S1. Refer to **BUS ARBITRATION CONTROL.** 5.2.3.
2. The address bus will be placed in the high-impedance state if T is asserted and $\overline{AS}$ is negated.

Figure 3-17. Bus Arbitration Unit State Diagrams

Figures 3-18, 3-19, and 3-20 apply to processors using 3-wire bus arbitration. Figures 3-21, 3-22, and 3-23 apply to processors using 2-wire bus arbitration.

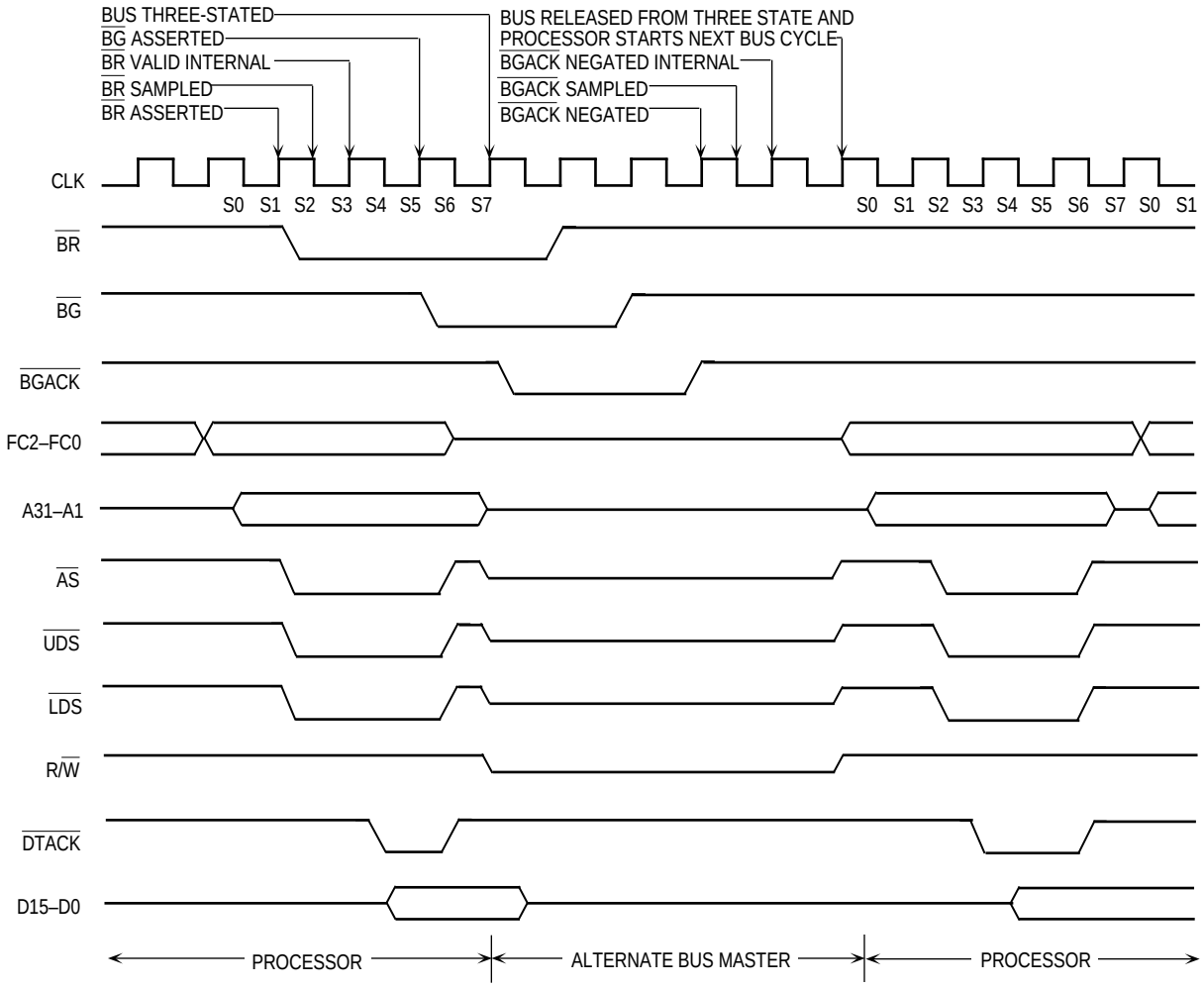


Figure 3-18. Three-Wire Bus Arbitration Timing Diagram—Processor Active

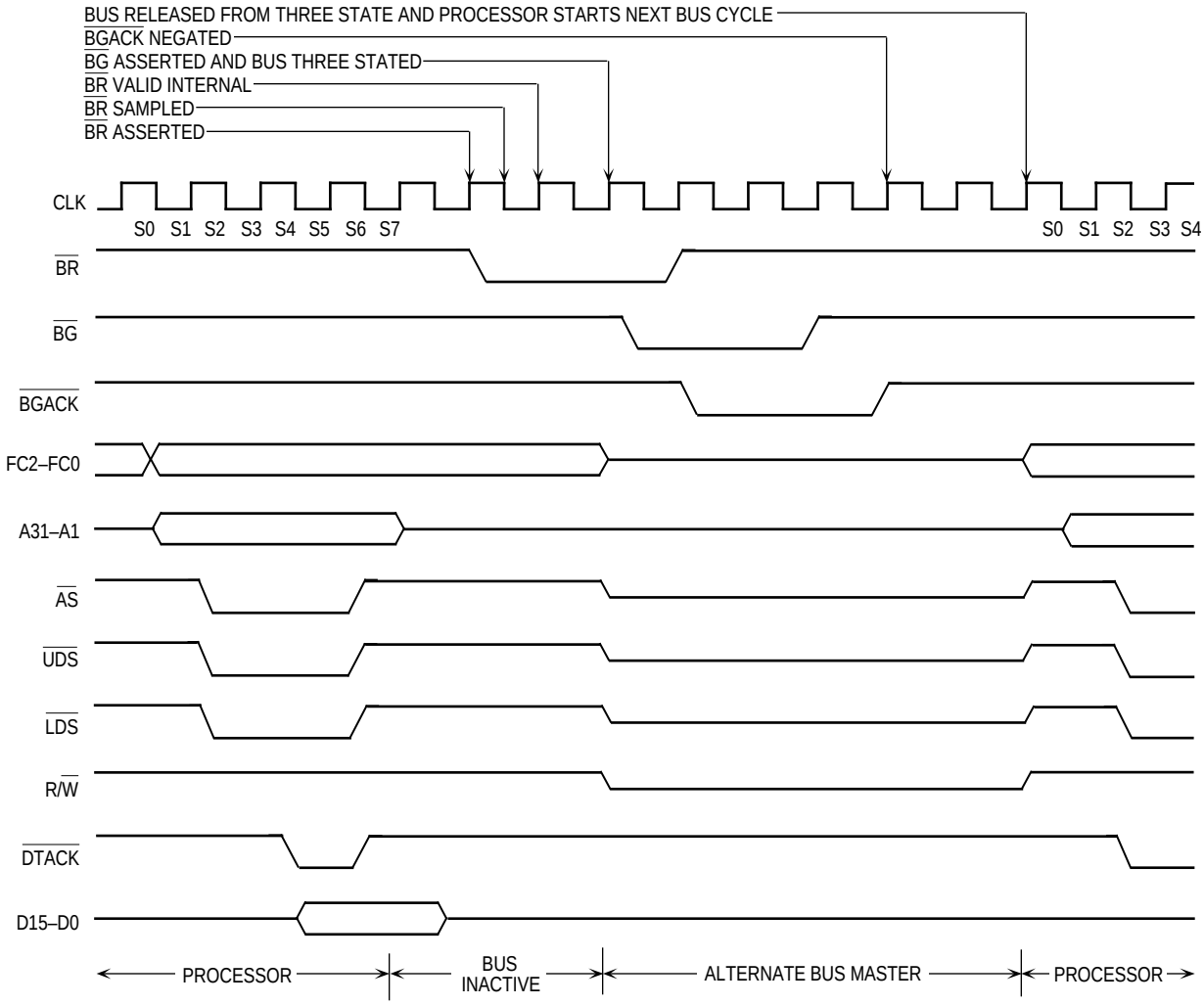


Figure 3-19. Three-Wire Bus Arbitration Timing Diagram—Bus Inactive

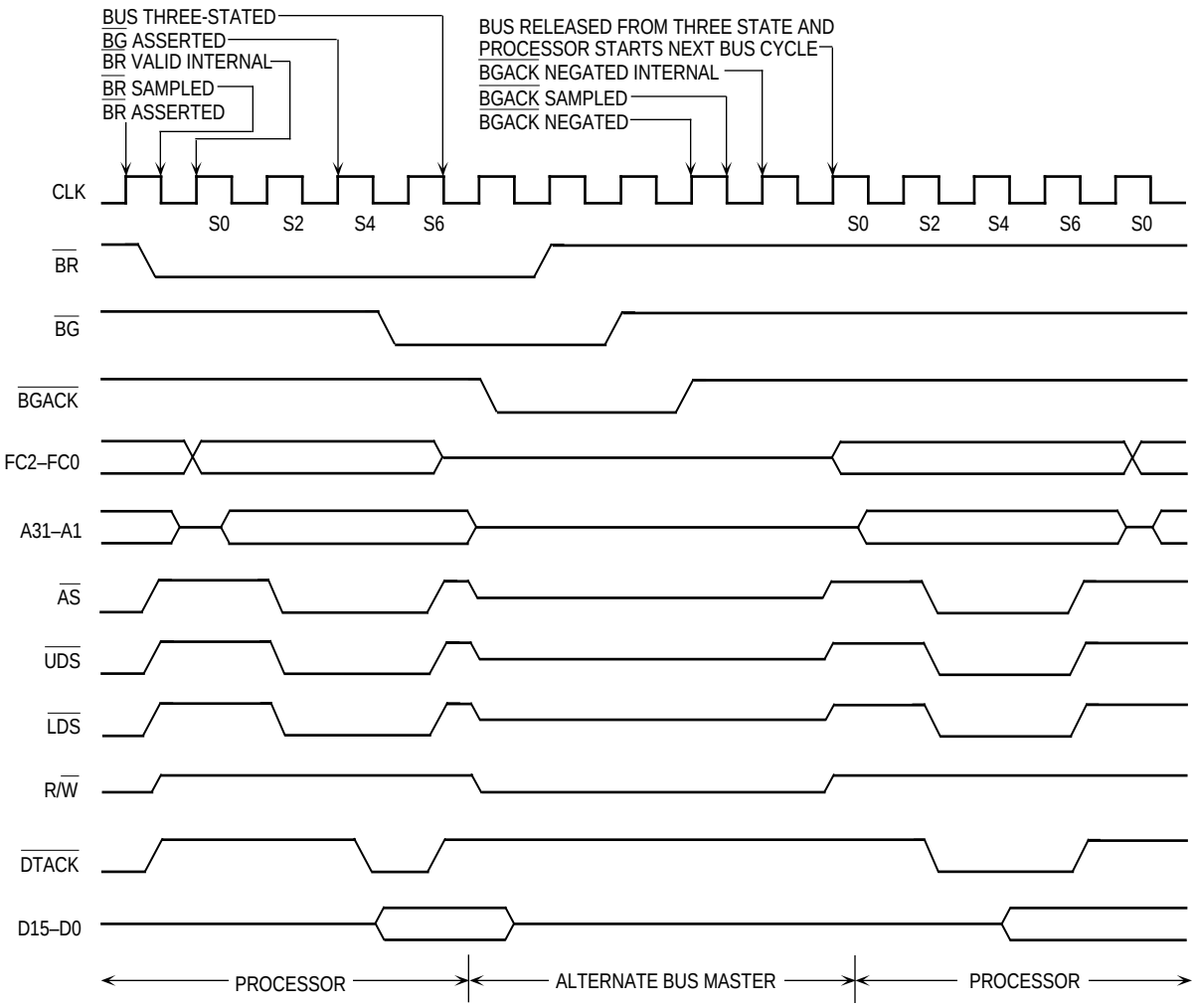


Figure 3-20. Three-Wire Bus Arbitration Timing Diagram—Special Case

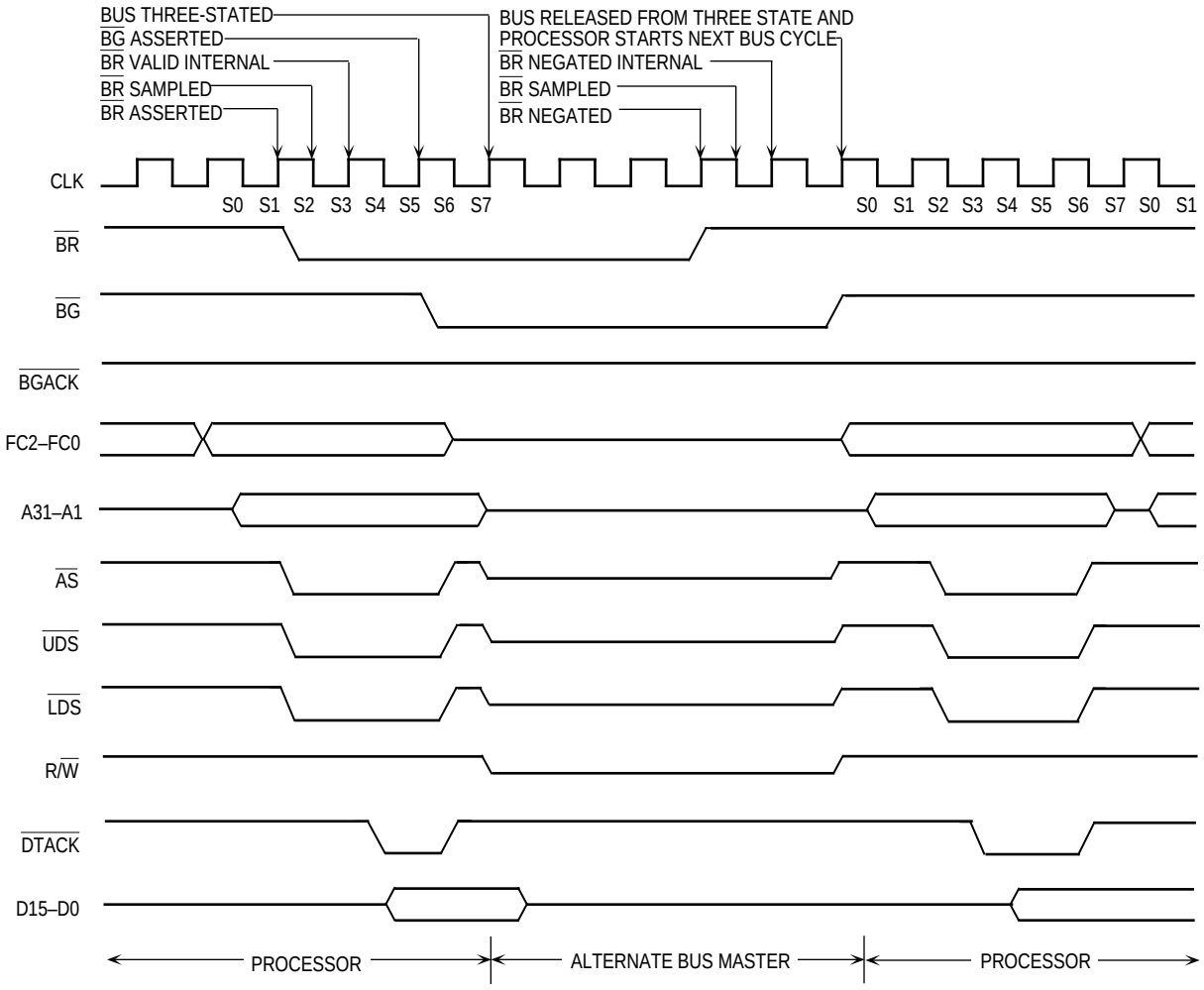


Figure 3-21. Two-Wire Bus Arbitration Timing Diagram—Processor Active

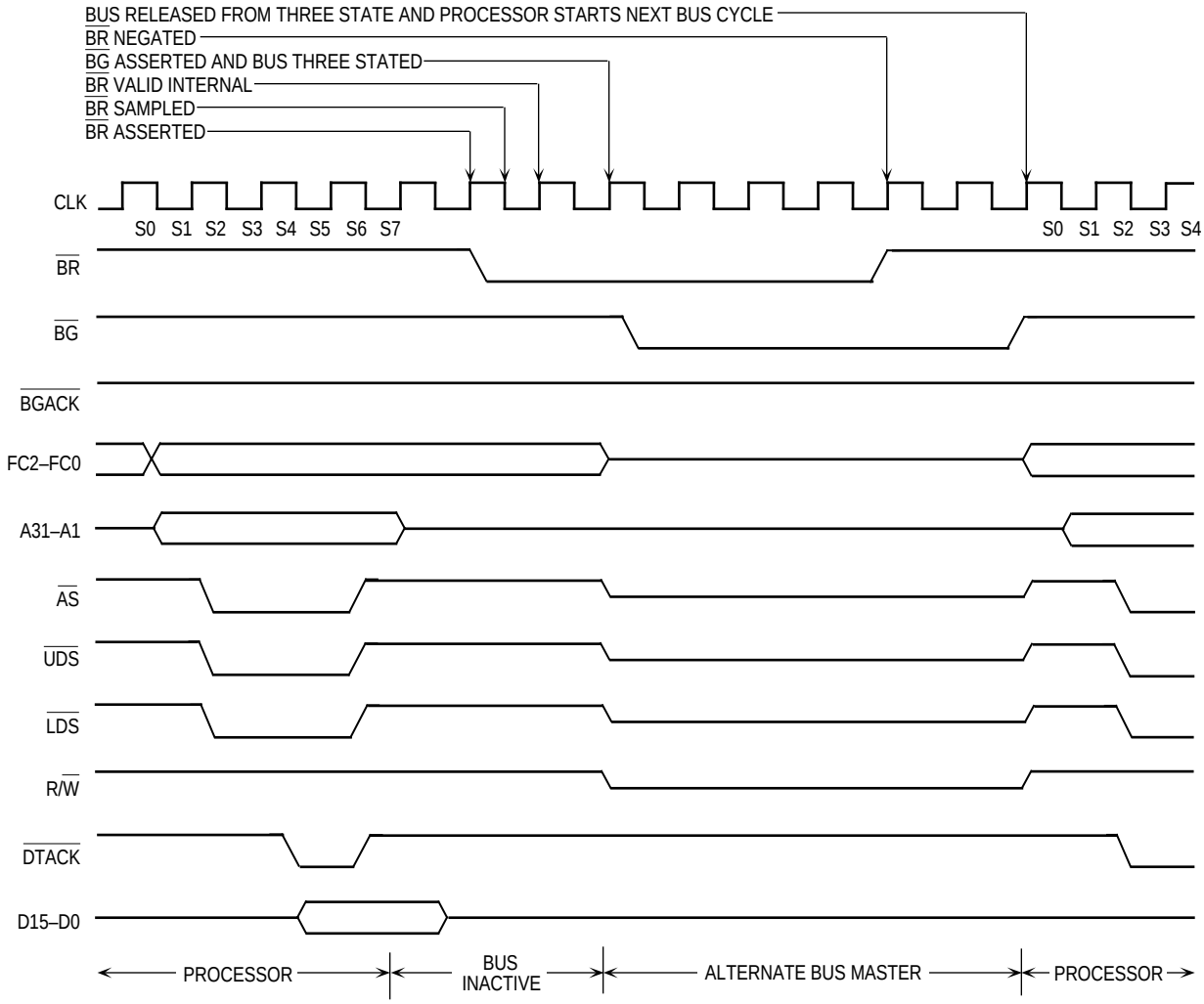


Figure 3-22. Two-Wire Bus Arbitration Timing Diagram—Bus Inactive

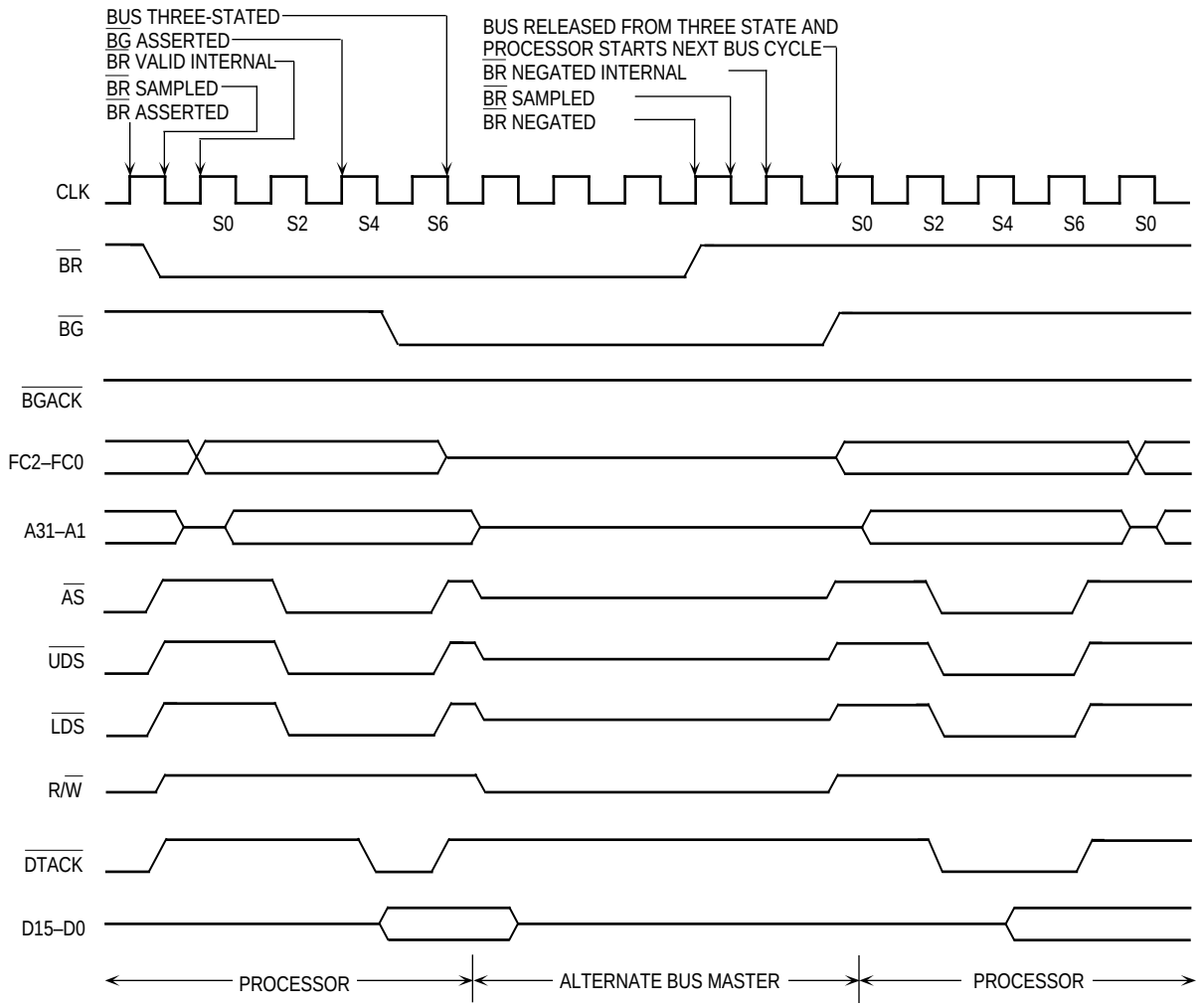


Figure 3-23. Two-Wire Bus Arbitration Timing Diagram—Special Case

3.4 BUS ERROR AND HALT OPERATION

In a bus architecture that requires a handshake from an external device, such as the asynchronous bus used in the M68000 Family, the handshake may not always occur. A bus error input is provided to terminate a bus cycle in error when the expected signal is not asserted. Different systems and different devices within the same system require different maximum-response times. External circuitry can be provided to assert the bus error signal after the appropriate delay following the assertion of address strobe.

3.4.1 Bus Error Operation

A bus error is recognized when $\overline{\text{BERR}}$ is asserted, $\overline{\text{HALT}}$ is negated, and $\overline{\text{DTACK}}$ is not asserted before $\overline{\text{BERR}}$ (or not at all).

When the bus error condition is recognized, the current bus cycle is terminated in S7 ($\overline{\text{DTACK}}$ and $\overline{\text{BERR}}$ together) or S9 ($\overline{\text{BERR}}$ alone) for a read cycle, a write cycle, or the read portion of a read-modify-write cycle. For the write portion of a read-modify-write cycle, the current bus cycle is terminated in S19 ($\overline{\text{DTACK}}$ and $\overline{\text{BERR}}$ together) or S21

(BERR alone). As long as BERR remains asserted, the data bus is in the high-impedance state. Figure 3-24 shows the timing for the normal bus error.

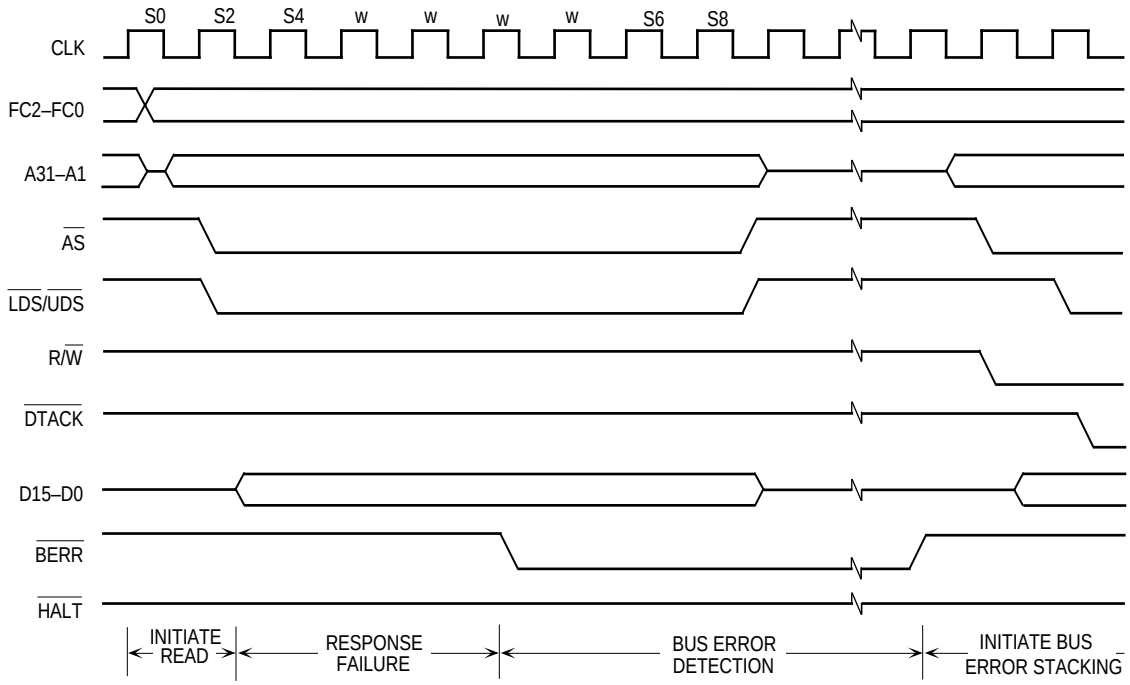


Figure 3-24. Bus Error Timing Diagram

After the aborted bus cycle is terminated and BERR is negated, the processor enters exception processing for the bus error exception. During the exception processing sequence, the following information is placed on the supervisor stack:

1. Status register
2. Program counter (two words, which may be up to five words past the instruction being executed)
3. Error information

The first two items are identical to the information stacked by any other exception. The EC000 core stacks bus error information to help determine and to correct the error.

After the processor has placed the required information on the stack, the bus error exception vector is read from vector table entry 2 (offset $08) and placed in the program counter. The processor resumes execution at the address in the vector, which is the first instruction in the bus error handler routine.

3.4.2 Retrying the Bus Cycle

The assertion of the bus error signal during a bus cycle in which HALT is also asserted by an external device initiates a retry operation. Figure 3-25 is a timing diagram of the retry operation.

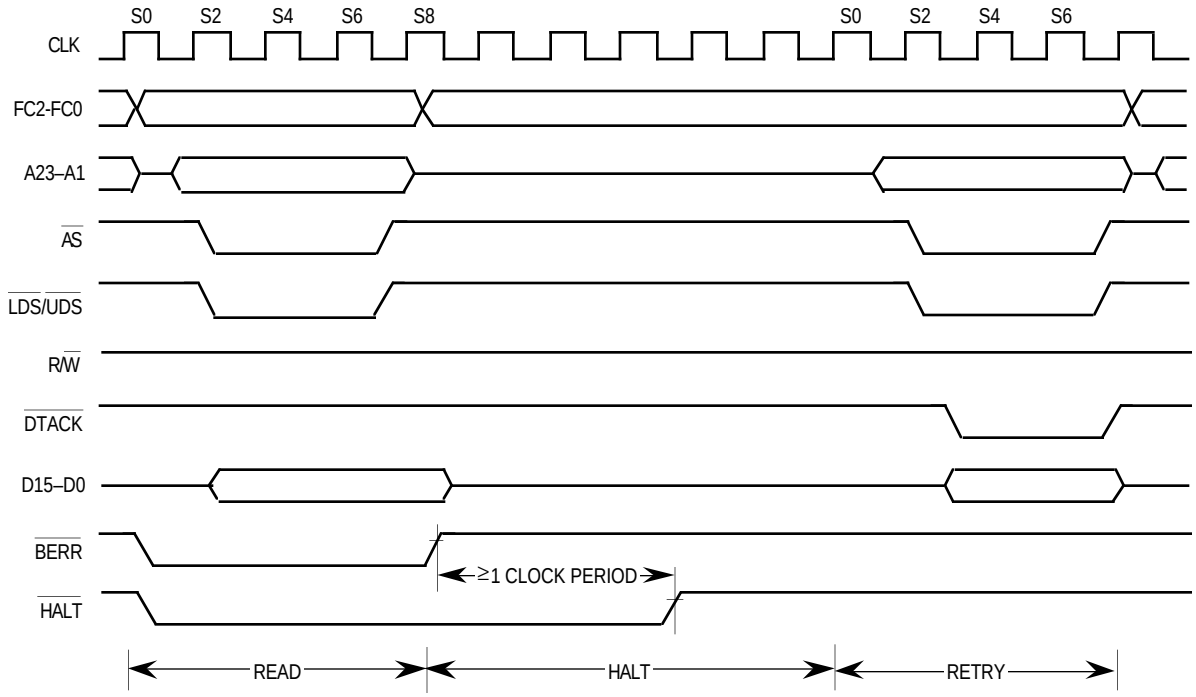


Figure 3-25. Retry Bus Cycle Timing Diagram

The processor terminates the bus cycle, and remains in this state until HALT is negated. Then the processor retries the preceding cycle using the same function codes, address, and data (for a write operation). BERR should be negated at least one clock cycle before HALT is negated.

NOTE

To guarantee that the entire read-modify-write cycle runs correctly and that the write portion of the operation is performed without negating the address strobe, the processor does not retry a read-modify-write cycle. When BERR occurs during a read-modify-write operation, a bus error operation is performed whether or not HALT is asserted.

3.4.3 Halt Operation

HALT performs a halt/run/single-step operation. When HALT is asserted by an external device, the processor halts and remains halted as long as the signal remains asserted, as shown in Figure 3-26.

While the processor is halted, bus arbitration is performed as usual. Should a bus error occur while HALT is asserted, the processor performs the retry operation previously described.

NOTE

If a RESET instruction is executed while HALT is asserted, the CPU will be reset.

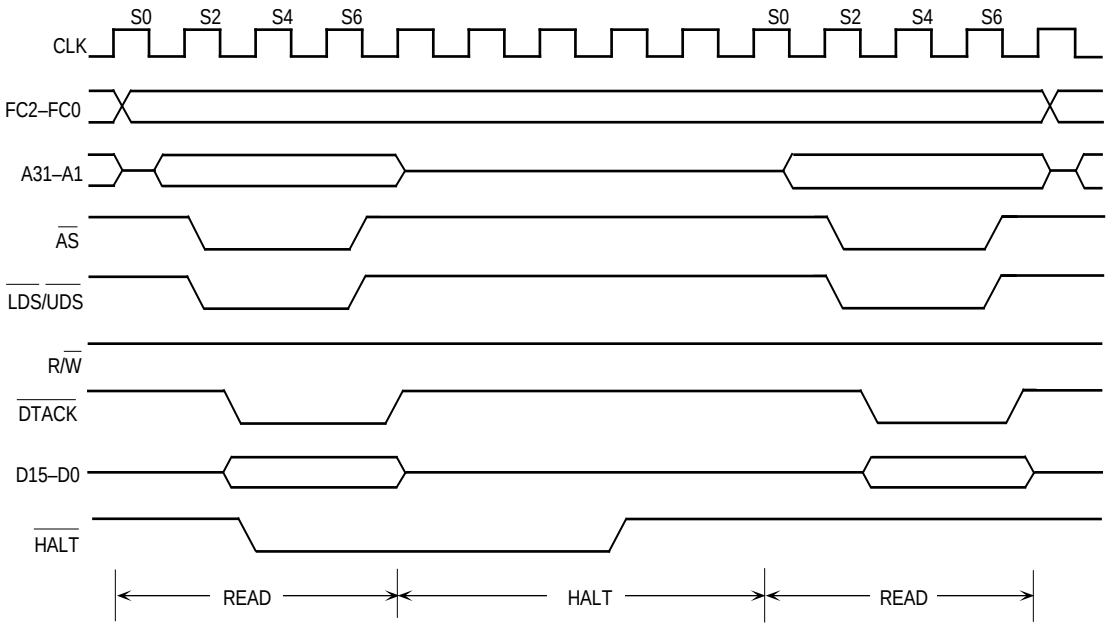


Figure 3-26. Halt Operation Timing Diagram

The single-step mode is derived from correctly timed transitions of $\overline{\text{HALT}}$. $\overline{\text{HALT}}$ is negated to allow the processor to begin a bus cycle, then asserted to enter the halt mode when the cycle completes. The single-step mode proceeds through a program one bus cycle at a time for debugging purposes. The halt operation and the hardware trace capability allow tracing of either bus cycles or instructions one at a time. These capabilities and a software debugging package provide total debugging flexibility.

3.4.4 Double Bus Fault

When a bus error exception occurs, the processor begins exception processing by stacking information on the supervisor stack. If another bus error occurs during exception processing (i.e., before execution of another instruction begins) the processor halts and asserts $\overline{\text{HALT}}$. This is called a double bus fault. Only an external reset operation can restart a processor halted due to a double bus fault.

A retry operation does not initiate exception processing; a bus error during a retry operation does not cause a double bus fault. The processor can continue to retry a bus cycle indefinitely if external hardware requests.

A double bus fault occurs during a reset operation when a bus error occurs while the processor is reading the vector table (before the first instruction is executed). The reset operation is described in the following paragraph.

3.5 RESET OPERATION

$\overline{\text{RESET}}$ is asserted externally for the initial processor reset. Subsequently, the signal can be asserted either externally or internally (executing a RESET instruction).

After the processor is reset, it reads the reset vector table entry (address $00000) and loads the contents into the supervisor stack pointer (SSP). Next, the processor loads the contents of address $00004 (vector table entry 1) into the program counter. Then the processor initializes the interrupt level in the status register to a value of seven. No other register is affected by the reset sequence. Figure 3-27 shows the timing of the reset operation.

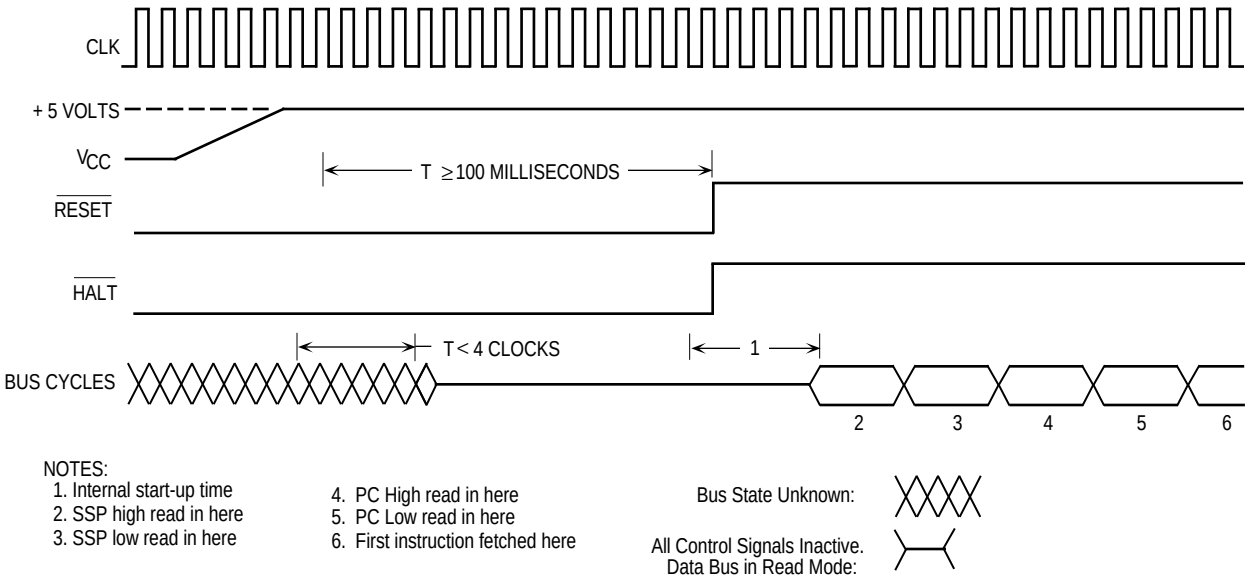


Figure 3-27. Reset Operation Timing Diagram

The active-low $\overline{\text{RESET}}$ signal is asserted by the EC000 core when a RESET instruction is executed. This signal should reset all external devices (the EC000 core itself is not affected). The processor drives $\overline{\text{RESET}}$ for 124 clock periods. The $\overline{\text{RESET}}$ signal is asserted by an external source to reset the EC000 core. $\overline{\text{RESET}}$ by itself will reset the EC000 core unless the processor is executing a RESET instruction. To guarantee a reset of the core, $\overline{\text{RESET}}$ must be asserted for at least 132 clocks (i.e., longer than the maximum duration of the RESET instruction), or $\overline{\text{RESET}}$ and $\overline{\text{HALT}}$ must be asserted together for at least 10 clocks.

3.6 THE RELATIONSHIP OF $\overline{\text{DTACK}}$, $\overline{\text{BERR}}$, AND $\overline{\text{HALT}}$

To properly control termination of a bus cycle for a retry or a bus error condition, $\overline{\text{DTACK}}$, $\overline{\text{BERR}}$, and $\overline{\text{HALT}}$ should be asserted and negated on the rising edge of the processor clock. This relationship assures that when two signals are asserted simultaneously, the required setup time (specification #47, **AC Electrical Specifications—Read and Write Cycles**) for both of them is met during the same bus state. External circuitry should be designed to incorporate this precaution. A related specification, #48, can be ignored when $\overline{\text{DTACK}}$, $\overline{\text{BERR}}$, and $\overline{\text{HALT}}$ are asserted and negated on the rising edge of the processor clock.

The possible bus cycle terminations can be summarized as follows (case numbers refer to Table 3-1).

Normal Termination: $\overline{DTACK}$ is asserted. $\overline{BERR}$ and $\overline{HALT}$ remain negated (case 1).

Halt Termination: $\overline{HALT}$ is asserted coincident with or preceding $\overline{DTACK}$, and $\overline{BERR}$ remains negated (case 2).

Bus Error Termination: $\overline{BERR}$ is asserted in lieu of, coincident with, or preceding $\overline{DTACK}$ (case 3).

Retry Termination: $\overline{HALT}$ and $\overline{BERR}$ asserted in lieu of, coincident with, or before $\overline{DTACK}$ (case 5).

Table 3-1 shows the details of the resulting bus cycle terminations for various combinations of signal sequences.

Table 3-1. $\overline{DTACK}$, $\overline{BERR}$, and $\overline{HALT}$ Assertion Results

Case No.	Control Signal	Asserted on Rising Edge of State		EC000 Core Results
		N	N+2	
1	$\overline{DTACK}$ $\overline{BERR}$ $\overline{HALT}$	A NA NA	S NA X	Normal cycle terminate and continue.
2	$\overline{DTACK}$ $\overline{BERR}$ $\overline{HALT}$	A NA A/S	S NA S	Normal cycle terminate and halt. Continue when $\overline{HALT}$ negated.
3	$\overline{DTACK}$ $\overline{BERR}$ $\overline{HALT}$	X A NA	X S NA	Terminate and take bus error trap.
4	$\overline{DTACK}$ $\overline{BERR}$ $\overline{HALT}$	A NA NA	S A NA	Normal cycle terminate and continue.
5	$\overline{DTACK}$ $\overline{BERR}$ $\overline{HALT}$	X A A/S	X S S	Terminate and retry when $\overline{HALT}$ removed.
6	$\overline{DTACK}$ $\overline{BERR}$ $\overline{HALT}$	A NA NA	S A A	Normal cycle terminate and continue.

LEGEND:

N — The number of the current even bus state (e.g., S4, S6, etc.)
A — Signal asserted in this bus state
NA — Signal not asserted in this bus state
X — Don't care
S — Signal asserted in preceding bus state and remains asserted in this state

NOTE: All operations are subject to relevant setup and hold times.

The negation of $\overline{BERR}$ and $\overline{HALT}$ under several conditions is shown in Table 3-2. ($\overline{DTACK}$ is assumed to be negated normally in all cases; for reliable operation, both $\overline{DTACK}$ and $\overline{BERR}$ should be negated when address strobe is negated).

EXAMPLE A:

A system uses a watchdog timer to terminate accesses to unused address space. The timer asserts $\overline{BERR}$ after timeout (case 3).

EXAMPLE B:

A system uses error detection on random-access memory (RAM) contents. The system designer may:

1. Delay $\overline{DTACK}$ until the data is verified. If data is invalid, return $\overline{BERR}$ and $\overline{HALT}$ simultaneously to retry the error cycle (case 5).
2. Delay $\overline{DTACK}$ until the data is verified. If data is invalid, return $\overline{BERR}$ at the same time as $\overline{DTACK}$ (case 3).

Table 3-2. $\overline{BERR}$ and $\overline{HALT}$ Negation Results

Conditions of Termination in Table 4-4	Control Signal	Negated on Rising Edge of State			Results—Next Cycle
		N		N+2	
Bus Error	$\overline{BERR}$ $\overline{HALT}$	• •	or or	• •	Takes bus error trap.
Rerun	$\overline{BERR}$ $\overline{HALT}$	• •	or	•	Illegal sequence; usually traps to vector number 0.
Rerun	$\overline{BERR}$ $\overline{HALT}$	•		 •	Reruns the bus cycle.
Normal	$\overline{BERR}$ $\overline{HALT}$	• •	 or	 •	May lengthen next cycle.
Normal	$\overline{BERR}$ $\overline{HALT}$	 •	 or	• none	If next cycle is started, it will be terminated as a bus error.

• = Signal is negated in this bus state.

3.7 ASYNCHRONOUS OPERATION

To achieve clock frequency independence at a system level, the bus can be operated in an asynchronous manner. Asynchronous bus operation uses the bus handshake signals to control the transfer of data. The handshake signals are $\overline{AS}$, $\overline{UDS}$, $\overline{LDS}$, $\overline{DTACK}$, $\overline{BERR}$, and $\overline{HALT}$. $\overline{AS}$ indicates the start of the bus cycle, and $\overline{UDS}$ and $\overline{LDS}$ signal valid data for a write cycle. After placing the requested data on the data bus (read cycle) or latching the data (write cycle), the slave device (memory or peripheral) asserts $\overline{DTACK}$ to terminate the bus cycle. If no device responds or if the access is invalid, external control logic asserts $\overline{BERR}$, or $\overline{BERR}$ and $\overline{HALT}$, to abort or retry the cycle. Figure 3-28 shows the use of the bus handshake signals in a fully asynchronous read cycle. Figure 3-29 shows a fully asynchronous write cycle.

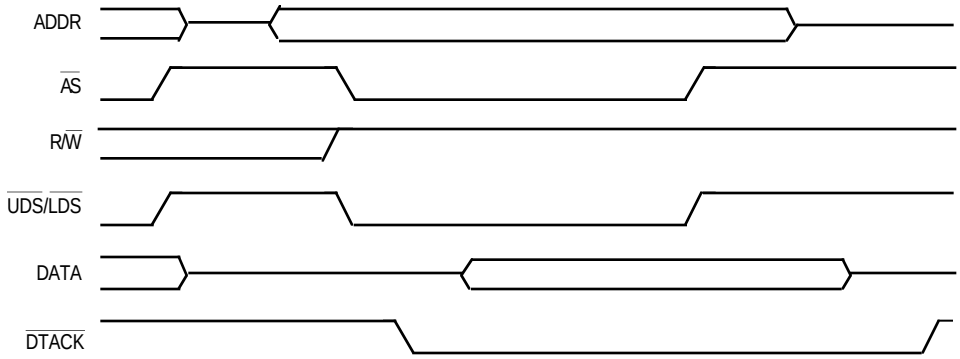


Figure 3-28 Fully Asynchronous Read Cycle

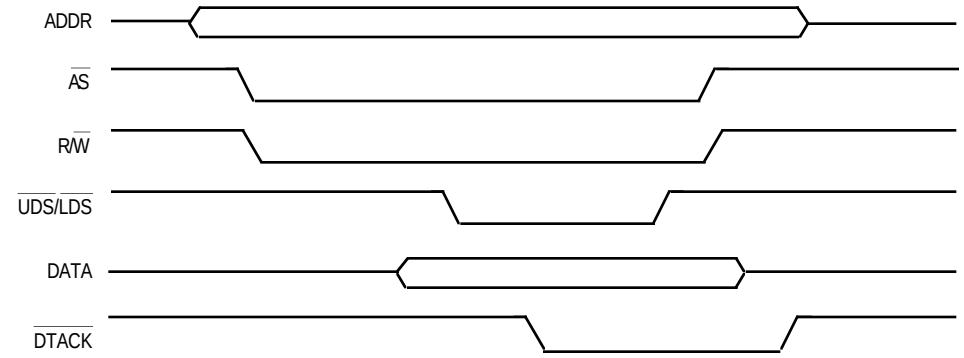


Figure 3-29. Fully Asynchronous Write Cycle

In the asynchronous mode, the accessed device operates independently of the frequency and phase of the system clock. For example, the MC68681 dual universal asynchronous receiver/transmitter (DUART) does not require any clock-related information from the bus master during a bus transfer. Asynchronous devices are designed to operate correctly with processors at any clock frequency when relevant timing requirements are observed.

A device can use a clock at the same frequency as the system clock (e.g., 8, 10, or 12.5 MHz), but without a defined phase relationship to the system clock. This mode of operation is pseudo-asynchronous; it increases performance by observing timing parameters related to the system clock frequency without being completely synchronous with that clock. A memory array designed to operate with a particular frequency processor but not driven by the processor clock is a common example of a pseudo-asynchronous device.

The designer of a fully asynchronous system can make no assumptions about address setup time, which could be used to improve performance. With the system clock frequency known, the slave device can be designed to decode the address bus before recognizing an address strobe. Parameter #11 (refer to **AC Electrical Specifications—Read and Write Cycles**) specifies the minimum time before address strobe during which the address is valid.

In a pseudo-asynchronous system, timing specifications allow DTACK to be asserted for a read cycle before the data from a slave device is valid. The length of time that DTACK may precede data is specified as parameter #31. This parameter must be met to ensure the validity of the data latched into the processor. No maximum time is specified from the assertion of AS to the assertion of DTACK. During this unlimited time, the processor inserts wait cycles in one-clock-period increments until DTACK is recognized. Figure 3-30 shows the important timing parameters for a pseudo-asynchronous read cycle.

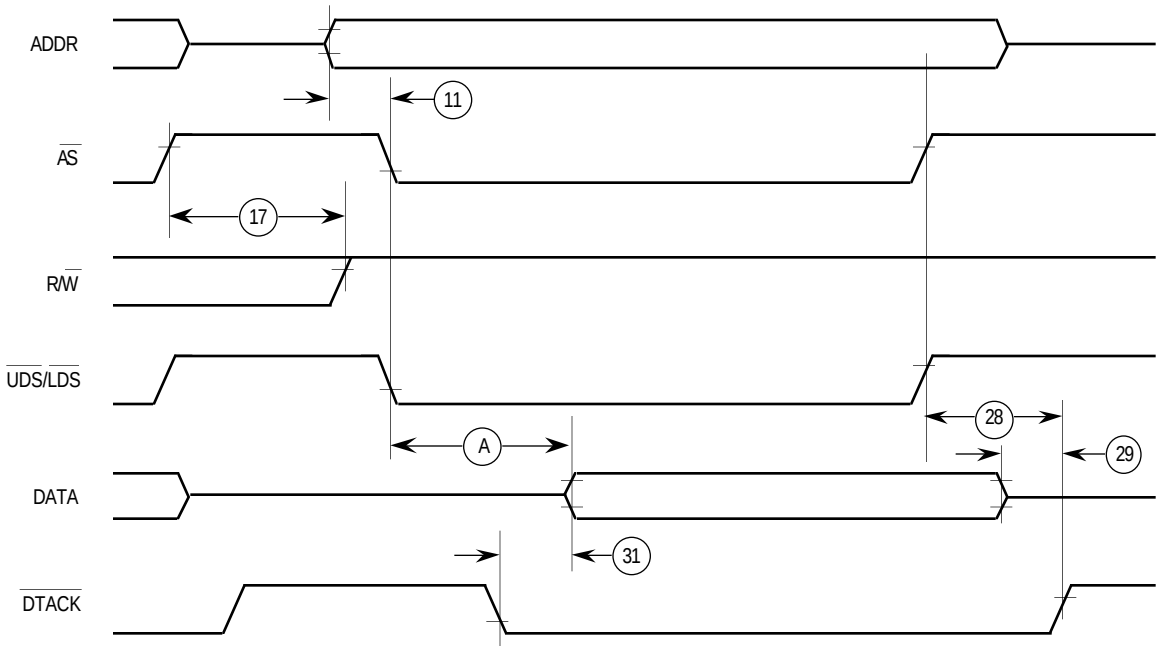


Figure 3-30. Pseudo-Asynchronous Read Cycle

During a write cycle, after the processor asserts AS but before driving the data bus, the processor drives R/W low. Parameter #55 specifies the minimum time between the transition of R/W and the driving of the data bus, which is effectively the maximum turnoff time for any device driving the data bus.

After the processor places valid data on the bus, it asserts the data strobe signal(s). A data setup time, similar to the address setup time previously discussed, can be used to improve performance. Parameter #26 is the minimum time a slave device can accept valid data before recognizing a data strobe. The slave device asserts DTACK after it accepts the data. Parameter #25 is the minimum time after negation of the strobes during which the valid data remains on the address bus. Parameter #28 is the maximum time between the negation of the strobes by the processor and the negation of DTACK by the slave device. If DTACK remains asserted past the time specified by parameter #28, the processor may recognize it as being asserted early in the next bus cycle and may terminate that cycle prematurely. Figure 3-31 shows the important timing specifications for a pseudo-asynchronous write cycle.

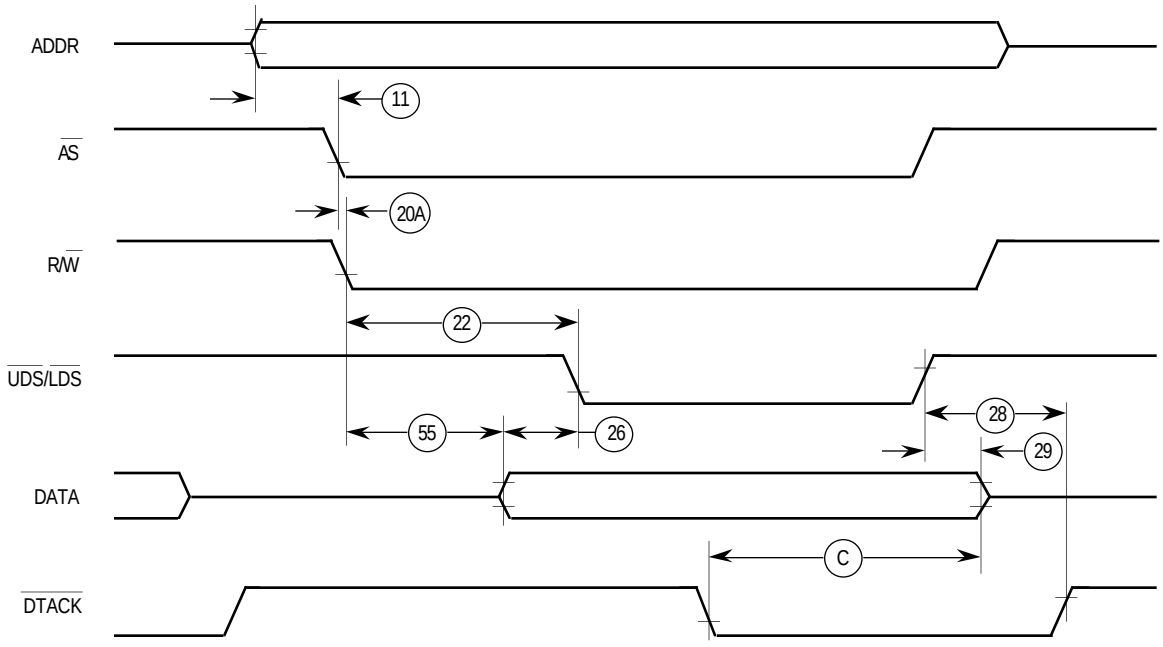


Figure 3-31. Pseudo-Asynchronous Write Cycle

3.8 SYNCHRONOUS OPERATION

In some systems, external devices use the system clock to generate $\overline{\text{DTACK}}$ and other asynchronous input signals. This synchronous operation provides a closely coupled design with maximum performance, appropriate for frequently accessed parts of the system. For example, memory can operate in the synchronous mode, but peripheral devices operate asynchronously. For a synchronous device, the designer uses explicit timing information shown in **AC Electrical Specifications—Read and Write Cycles**. These specifications define the state of all bus signals relative to a specific state of the processor clock.

The standard M68000 bus cycle consists of four clock periods (eight bus cycle states) and, optionally, an integral number of clock cycles inserted as wait states. Wait states are inserted as required to allow sufficient response time for the external device. The following state-by-state description of the bus cycle differs from those descriptions in **3.1.1 Read Cycle** and **3.1.2 Write Cycle** by including information about the important timing parameters that apply in the bus cycle states.

STATE 0 The bus cycle starts in S0, during which the clock is high. At the rising edge of S0, the function code for the access is driven externally. Parameter #6A defines the delay from this rising edge until the function codes are valid. Also, the R/$\overline{\text{W}}$ signal is driven high; parameter #18 defines the delay from the same rising edge to the transition of R/$\overline{\text{W}}$. The minimum value for parameter #18 applies to a read cycle preceded by a write cycle; this value is the maximum hold time for a low on R/$\overline{\text{W}}$ beyond the initiation of the read cycle.

STATE 1 Entering S1, a low period of the clock, the address of the accessed device is driven externally with an assertion delay defined by parameter #6.

STATE 2 On the rising edge of S2, a high period of the clock, $\overline{AS}$ is asserted. During a read cycle, $\overline{UDS}$ and/or $\overline{LDS}$ is also asserted at this time. Parameter #9 defines the assertion delay for these signals. For a write cycle, the R/$\overline{W}$ signal is driven low with a delay defined by parameter #20.

STATE 3 On the falling edge of the clock entering S3, the data bus is driven out of the high-impedance state with the data being written to the accessed device (in a write cycle). Parameter #23 specifies the data assertion delay. In a read cycle, no signal is altered in S3.

STATE 4 Entering the high clock period of S4, $\overline{UDS}/\overline{LDS}$ is asserted (during a write cycle) on the rising edge of the clock. As in S2 for a read cycle, parameter #9 defines the assertion delay from the rising edge of S4 for $\overline{UDS}/\overline{LDS}$. In a read cycle, no signal is altered by the processor during S4.

Until the falling edge of the clock at the end of S4 (beginning of S5), no response from any external device except $\overline{RESET}$ is acknowledged by the processor. If either $\overline{DTACK}$ or $\overline{BERR}$ is asserted before the falling edge of S4 and satisfies the input setup time defined by parameter #47, the processor enters S5 and the bus cycle continues. If either $\overline{DTACK}$ or $\overline{BERR}$ is asserted but without meeting the setup time defined by parameter #47, the processor may recognize the signal and continue the bus cycle; the result is unpredictable. If neither $\overline{DTACK}$ nor $\overline{BERR}$ is asserted before the next rise of clock, the bus cycle remains in S4, and wait states (complete clock cycles) are inserted until one of the bus cycle terminations is met.

STATE 5 S5 is a low period of the clock, during which the processor does not alter any signal.

STATE 6 S6 is a high period of the clock, during which data for a read operation is set up relative to the falling edge (entering S7). Parameter #27 defines the minimum period by which the data must precede the falling edge. For a write operation, the processor changes no signal during S6.

STATE 7 On the falling edge of the clock entering S7, the processor latches data and negates $\overline{AS}$ and $\overline{UDS}/\overline{LDS}$ during a read cycle. The hold time for these strobes from this falling edge is specified by parameter #12. The hold time for data relative to the negation of $\overline{AS}$ and $\overline{UDS}/\overline{LDS}$ is specified by parameter #29. For a write cycle, only $\overline{AS}$ and $\overline{UDS}/\overline{LDS}$, are negated; timing parameter #12 also applies.

On the rising edge of the clock, at the end of S7 (which may be the start of S0 for the next bus cycle), the processor places the address bus in the high-impedance state. During a write cycle, the processor also places the data bus in the high-impedance state and drives R/$\overline{W}$ high. External logic circuitry should respond to the negation of the $\overline{AS}$ and $\overline{UDS}$/$\overline{LDS}$ by negating $\overline{DTACK}$ and/or $\overline{BERR}$. Parameter #28 is the hold time for $\overline{DTACK}$, and parameter #30 is the hold time for $\overline{BERR}$.

Figure 3-32 shows a synchronous read cycle and the important timing parameters that apply. The timing for a synchronous read cycle, including relevant timing parameters, is shown in Figure 3-33.

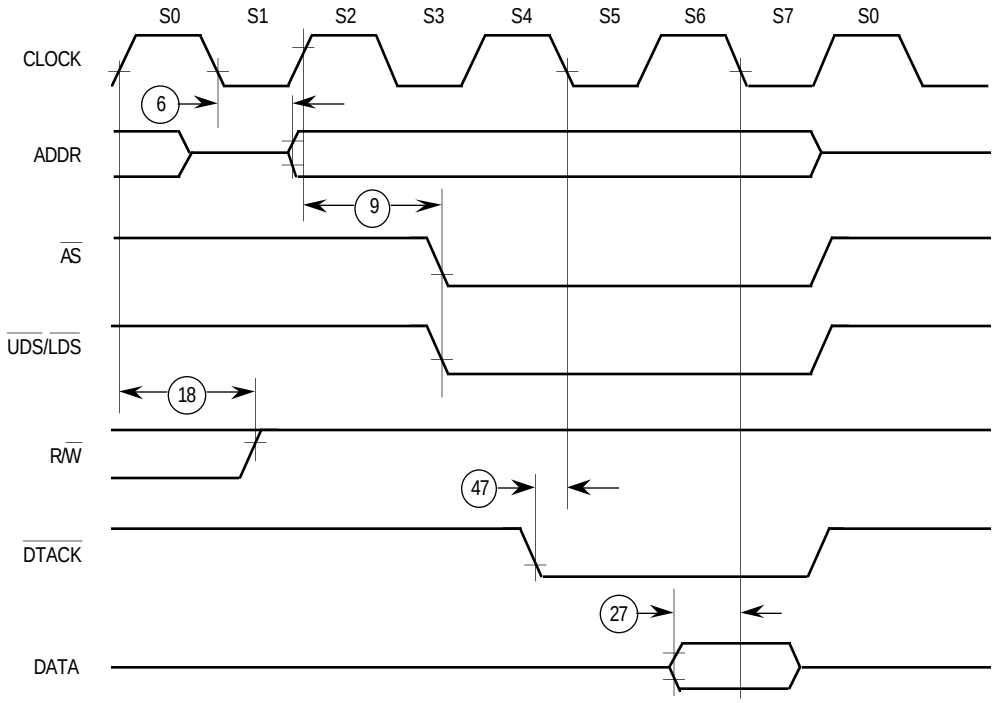


Figure 3-32. Synchronous Read Cycle

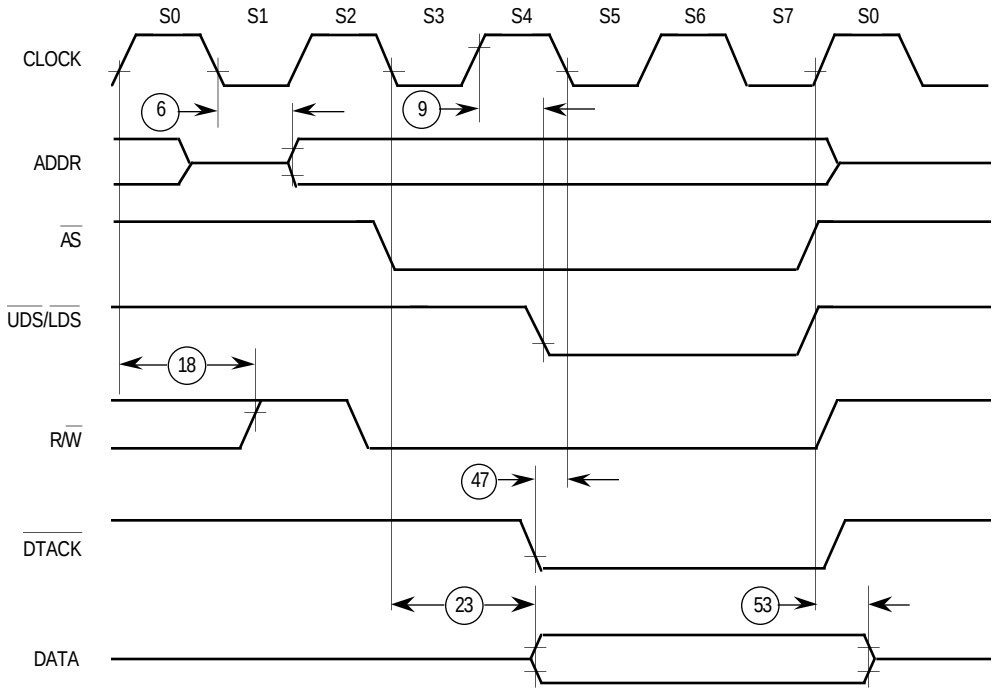


Figure 3-33. Synchronous Write Cycle

A key consideration when designing in a synchronous environment is the timing for the assertion of DTACK and BERR by an external device. To properly use external inputs, the processor must synchronize these signals to the internal clock. The processor must sample the external signal, which has no defined phase relationship to the CPU clock, which may be changing at sampling time, and must determine whether to consider the signal high or low during the succeeding clock period. Successful synchronization requires that the internal machine receives a valid logic level, whether the input is high, low, or in transition.

Parameter #47 of **AC Electrical Specifications—Read and Write Cycles** is the asynchronous input setup time. Signals that meet parameter #47 are guaranteed to be recognized at the next falling edge of the system clock. However, signals that do not meet parameter #47 are not guaranteed to be recognized. In addition, if DTACK is recognized on a falling edge, valid data is latched into the processor (during a read cycle) on the next falling edge, provided the data meets the setup time required (parameter #27). When parameter #27 has been met, parameter #31 may be ignored. If DTACK is asserted with the required setup time before the falling edge of S4, no wait states are incurred, and the bus cycle runs at its maximum speed of four clock periods.

SECTION 4
EC000 CORE PROCESSOR

The EC000 core has a 16-bit data bus and 32-bit address bus while the full architecture provides for 32-bit address and data register operations.

4.1 FEATURES

The following resources are available to the EC000 core:

- 8 32-Bit Address Registers
- 8 32-Bit Data Registers
- 4-Gbyte Direct Addressing Range
- 56 Powerful Instructions
- Operations on Five Main Data Types
- Memory-Mapped Input/Output (I/O)
- 14 Addressing Modes

4.2 PROCESSING STATES

The processor is always in one of three states: normal processing, exception processing, or halted. It is in the normal processing state when executing instructions, fetching instructions and operands, and storing instruction results.

Exception processing is the transition from program processing to system, interrupt, and exception handling. Exception processing includes fetching the exception vector, stacking operations, and refilling the instruction pipe after an exception. The processor enters exception processing when an exceptional internal condition arises such as tracing an instruction, an instruction results in a trap, or executing specific instructions. External conditions, such as interrupts and access errors, also cause exceptions. Exception processing ends when the first instruction of the exception handler begins to execute.

The processor halts when it receives an access error or generates an address error while in the exception processing state. For example, if during exception processing of one access error another access error occurs, the processor is unable to complete the transition to normal processing and cannot save the internal state of the machine. The processor assumes that the system is not operational and halts. Only an external reset can restart a halted processor. Note that when the processor executes a STOP instruction, it is in a special type of normal processing state, one without bus cycles. The processor stops, but it does not halt.

4.3 PROGRAMMING MODEL

The EC000 core executes instructions in one of two modes—user mode or supervisor mode. The user mode provides the execution environment for the majority of application programs. The supervisor mode, which allows some additional instructions and privileges, is used by the operating system and other system software.

To provide upward compatibility of code written for a specific implementation of the EC000 core, the user programmer's model, illustrated in Figure 4-1, is common to all implementations. In the user programmer's model, the EC000 core offers 16, 32-bit, general-purpose registers (D0–D7, A0–A7), a 32-bit program counter, and an 8-bit condition code register. The first eight registers (D0–D7) are used as data registers for byte (8-bit), word (16-bit), and long-word (32-bit) operations. The second set of seven registers (A0–A6) and the user stack pointer (USP) can be used as software stack pointers and base address registers. In addition, the address registers can be used for word and long-word operations. All of the 16 registers can be used as index registers. The supervisor programmer's model consists of supplementary registers used in the supervisor mode.

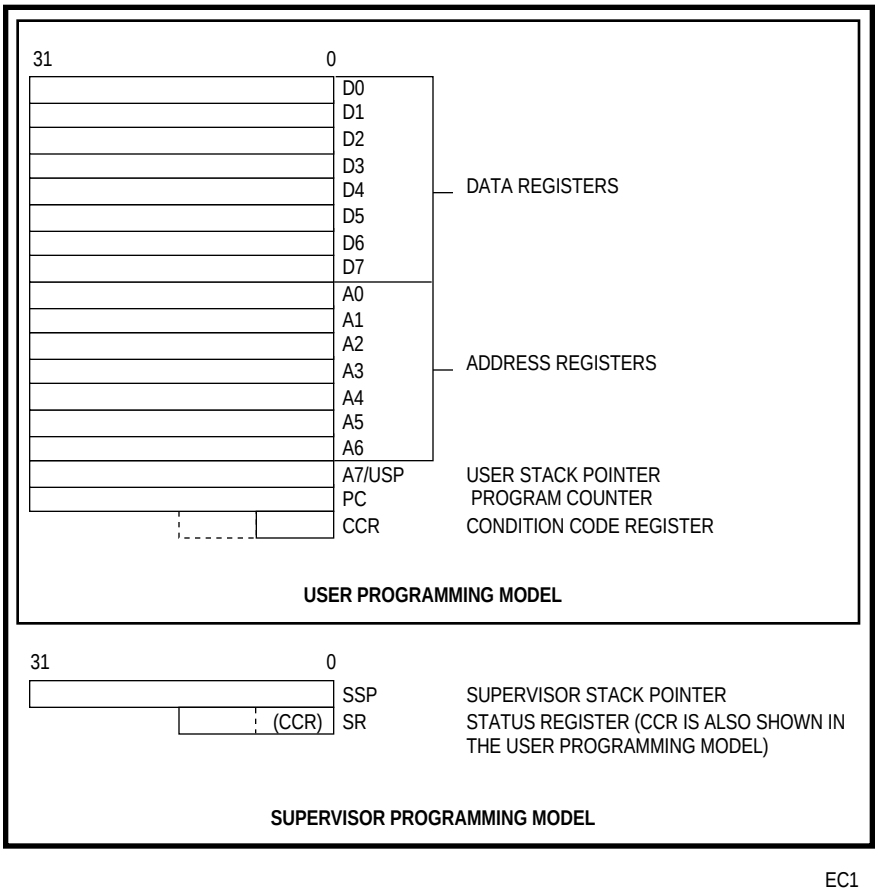


Figure 4-1. Programmer's Model

The status register, illustrated in Figure 4-2, contains the interrupt mask (eight levels available) and the following condition codes: overflow (V), zero (Z), negative (N), carry (C), and extend (X). Additional status bits indicate that the processor is in the trace (T) mode and/or in the supervisor (S) state.

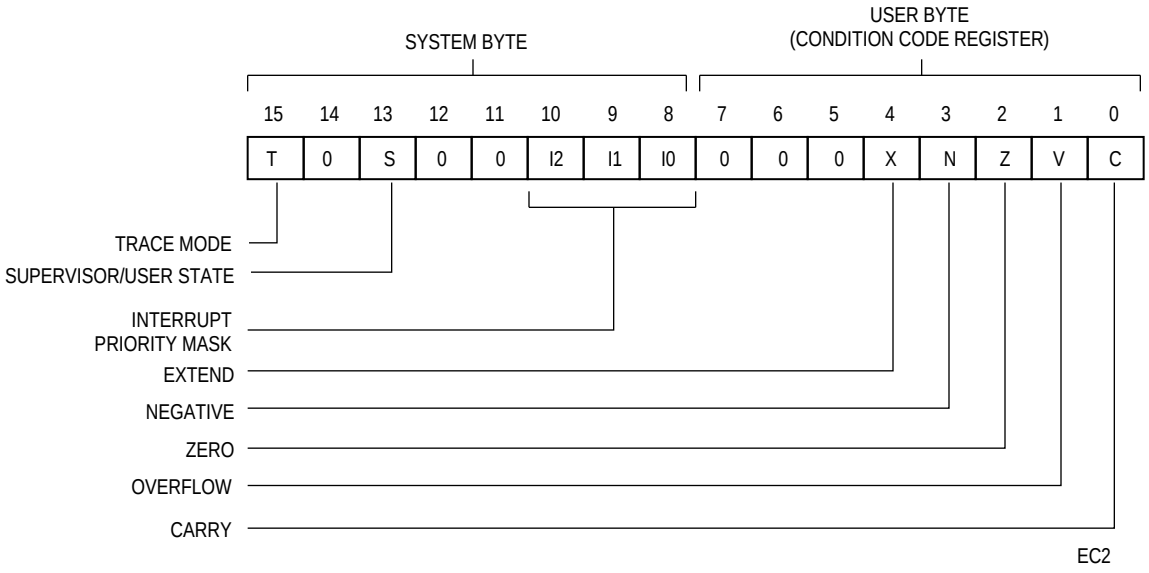


Figure 4-2. Status Register

4.3.1 Data Format Summary

The processor supports the basic data formats of the M68000 family. The instruction set supports operations on other data formats such as memory addresses.

The operand data formats supported by the integer unit (IU) are the standard twos-complement data formats defined in the M68000 family architecture. Registers, memory, or instructions themselves can contain IU operands. The operand size for each instruction is either explicitly encoded in the instruction or implicitly defined by the instruction operation. Table 4-1 lists the data formats for the processor. Refer to M68000PM/AD, *M68000 Family Programmer's Reference Manual,* for details on data format organization in registers and memory.

Table 4-1. Processor Data Formats

Operand Data Format	Size	Notes
Bit	1 Bit	—
Binary-Coded Decimal (BCD)	8 Bits	Packed: 2 Digits/Byte; Unpacked: 1 Digit/Byte
Byte Integer	8 Bits	—
Word Integer	16 Bits	—
Long-Word Integer	32 Bits	—

4.3.2 Addressing Capabilities Summary

The EC000 core supports the basic addressing modes of the M68000 family. The register indirect addressing modes support postincrement, predecrement, offset, and indexing, which are particularly useful for handling data structures common to sophisticated applications and high-level languages. The program counter indirect mode also has indexing and offset capabilities. This addressing mode is typically required to support position-independent software. Besides these addressing modes, the processor provides index sizing and scaling features.

An instruction's addressing mode can specify the value of an operand, a register containing the operand, or how to derive the effective address of an operand in memory. Each addressing mode has an assembler syntax. Some instructions imply the addressing mode for an operand. These instructions include the appropriate fields for operands that use only one addressing mode. Table 4-2 lists a summary of the effective addressing modes for the processor. Refer to M68000PM/AD, *M68000 Family Programmer's Reference Manual,* for details on instruction format and addressing modes.

Table 4-2. Effective Addressing Modes

Addressing Modes	Syntax
Register Direct Addressing	
Data Register Direct	EA=Dn
Address Register Direct	EA=An
Absolute Data Addressing	
Absolute Short	EA=(Next Word)
Absolute Long	EA=(Next Two Words)
Program Counter Relative Addressing	
Relative with Offset	EA=(PC)+d_{16}
Relative with Index and Offset	EA=(PC)+d_8
Register Indirect Addressing	
Register Indirect	EA=(An)
Postincrement Register Indirect	EA=(An), An ¨ An+N
Predecrement Register Indirect	An ¨ An–N, EA=(An)
Register Indirect with Offset	EA=(An)+d_{16}
Indexed Register Indirect with Offset	EA=(An)+(Xn)+d_8
Immediate Data Addressing	
Immediate	DATA=Next Word(s)
Quick Immediate	Inherent Data
Implied Addressing	
Implied Register	EA=SR, USP, SSP, PC, VBR, SFC, DFC

4.3.3 Notation Conventions

Table 4-3 lists the notation conventions used in this manual unless otherwise specified.

Table 4-3. Notation Conventions

Single and Double Operand Operations	
+	Arithmetic addition or postincrement indicator.
–	Arithmetic subtraction or predecrement indicator.
×	Arithmetic multiplication.
÷	Arithmetic division or conjunction symbol.
~	Invert; operand is logically complemented.
Λ	Logical AND
V	Logical OR
⊕	Logical exclusive OR
➧	Source operand is moved to destination operand.
➧➧	Two operands are exchanged.
<op>	Any double-operand operation.
<operand>tested	Operand is compared to zero and the condition codes are set appropriately.
sign-extended	All bits of the upper portion are made equal to the high-order bit of the lower portion.
Other Operations	
TRAP	Equivalent to Format ÷ Offset Word ➧ (SSP); SSP – 2 ➧ SSP; PC ➧ (SSP); SSP – 4 ➧ SSP; SR ➧ (SSP); SSP – 2 ➧ SSP; (Vector) ➧ PC
STOP	Enter the stopped state, waiting for interrupts.
<operand>$_{10}$	The operand is BCD; operations are performed in decimal.
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.
Register Specification	
An	Any Address Register n (example: A3 is address register 3)
Ax, Ay	Source and destination address registers, respectively.
$\overline{BR}$	Base Register—An, PC, or suppressed.
Dc	Data register D7–D0, used during compare.
Dh, Dl	Data registers high- or low-order 32 bits of product.
Dn	Any Data Register n (example: D5 is data register 5)
Dr, Dq	Data register’s remainder or quotient of divide.
Du	Data register D7–D0, used during update.
Dx, Dy	Source and destination data registers, respectively.
Rn	Any Address or Data Register
Rx, Ry	Any source and destination registers, respectively.
Xn	Index Register—An, Dn, or suppressed.

Table 4-3. Notation Conventions (Continued)

Data Format And Type	
<fmt>	Operand Data Format: Byte (B), Word (W), Long (L), or Packed (P).
B, W, L	Specifies a signed integer data type (twos complement) of byte, word, or long word.
k	A twos complement signed integer (–64 to +17) specifying a number's format to be stored in the packed decimal format.
Subfields and Qualifiers	
#<xxx> or #<data>	Immediate data following the instruction word(s).
( )	Identifies an indirect address in a register.
[ ]	Identifies an indirect address in memory.
bd	Base Displacement
d_n	Displacement Value, n Bits Wide (example: d_{16} is a 16-bit displacement).
LSB	Least Significant Bit
LSW	Least Significant Word
MSB	Most Significant Bit
MSW	Most Significant Word
od	Outer Displacement
SCALE	A scale factor (1, 2, 4, or 8, for no-word, word, long-word, or quad-word scaling, respectively).
SIZE	The index register's size (W for word, L for long word).
{offset:width}	Bit field selection.
Register Names	
CCR	Condition Code Register (lower byte of status register)
PC	Program Counter
SR	Status Register

Table 4-3. Notation Conventions (Concluded)

Register Codes	
*	General Case.
C	Carry Bit in CCR
cc	Condition Codes from CCR
FC	Function Code
N	Negative Bit in CCR
U	Undefined, Reserved for Motorola Use.
V	Overflow Bit in CCR
X	Extend Bit in CCR
Z	Zero Bit in CCR
—	Not Affected or Applicable.
Stack Pointers	
SP	Active Stack Pointer
SSP	Supervisor Stack Pointer
USP	User Stack Pointer
Miscellaneous	
<ea>	Effective Address
<label>	Assemble Program Label
<list>	List of registers, for example D3–D0.
LB	Lower Bound
m	Bit m of an Operand
m–n	Bits m through n of Operand
UB	Upper Bound

4.4 EC000 CORE INSTRUCTION SET OVERVIEW

Design of the instruction set gives special emphasis to support of structured, high-level languages and to ease of assembly language programming. Each instruction, with a few exceptions, operates on bytes, words, and long words, and most instructions can use any of the 14 addressing modes. Over 1000 useful instructions are provided by combining instruction types, data types, and addressing modes. These instructions include signed and unsigned multiply and divide, "quick" arithmetic operations, BCD arithmetic, and expanded operations (through traps). Additionally, the highly symmetric, proprietary microcoded structure of the instruction set provides a sound, flexible base for the future.

The EC000 core instruction set is listed in Table 4-4. For detailed information on the EC000 core instruction set, refer to M68000PM/AD, *M68000 Programmer's Reference Manual*.

Table 4-4. EC000 Core Instruction Set Summary

Opcode	Operation	Syntax
ABCD	BCD Source + BCD Destination + X ➧ Destination	ABCD Dy,Dx ABCD –(Ay),–(Ax)
ADD	Source + Destination ➧ Destination	ADD <ea>,Dn ADD Dn,<ea>
ADDA	Source + Destination ➧ Destination	ADDA <ea>,An
ADDI	Immediate Data + Destination ➧ Destination	ADDI #<data>,<ea>
ADDQ	Immediate Data + Destination ➧ Destination	ADDQ #<data>,<ea>
ADDX	Source + Destination + X ➧ Destination	ADDX Dy,Dx ADDX –(Ay),–(Ax)
AND	Source Λ Destination ➧ Destination	AND <ea>,Dn AND Dn,<ea>
ANDI	Immediate Data Λ Destination ➧ Destination	ANDI #<data>,<ea>
ANDI to CCR	Source Λ CCR ➧ CCR	ANDI #<data>,CCR
ANDI to SR	If supervisor state then Source Λ SR ➧ SR else TRAP	ANDI #<data>,SR
ASL, ASR	Destination Shifted by count ➧ Destination	ASd Dx,Dy[1] ASd #<data>,Dy[1] ASd <ea>[1]
Bcc	If condition true then PC + d_n ➧ PC	Bcc <label>
BCHG	~(bit number of Destination) ➧ Z; ~(bit number of Destination) ➧ (bit number) of Destination	BCHG Dn,<ea> BCHG #<data>,<ea>
BCLR	~(bit number of Destination) ➧ Z; 0 ➧ bit number of Destination	BCLR Dn,<ea> BCLR #<data>,<ea>
BRA	PC + d_n ➧ PC	BRA <label>
BSET	~(bit number of Destination) ➧ Z; 1 ➧ bit number of Destination	BSET Dn,<ea> BSET #<data>,<ea>
BSR	SP – 4 ➧ SP; PC ➧ (SP); PC + d_n ➧ PC	BSR <label>
BTST	–(bit number of Destination) ➧ Z;	BTST Dn,<ea> BTST #<data>,<ea>
CHK	If Dn < 0 or Dn > Source then TRAP	CHK <ea>,Dn
CLR	0 ➧ Destination	CLR <ea>
CMP	Destination – Source ➧ cc	CMP <ea>,Dn
CMPA	Destination – Source	CMPA <ea>,An
CMPI	Destination – Immediate Data	CMPI #<data>,<ea>

Table 4-4. EC000 Core Instruction Set Summary (Continued)

Opcode	Operation	Syntax
CMPM	Destination – Source ➧ cc	CMPM (Ay)+,(Ax)+
DBcc	If condition false then (Dn–1 ➧ Dn; If Dn ≠ –1 then PC + d_n ➧ PC)	DBcc Dn,<label>
DIVS	Destination ÷ Source ➧ Destination	DIVS.W <ea>,Dn 32 ÷ 16 ➧ 16r:16q DIVS.L <ea>,Dq 32 ÷ 32 ➧ 32q DIVS.L <ea>,Dr:Dq 64 ÷ 32 ➧ 32r:32q
DIVU	Destination ÷ Source ➧ Destination	DIVU.W <ea>,Dn 32 ÷ 16 ➧ 16r:16q DIVU.L <ea>,Dq 32 ÷ 32 ➧ 32q DIVU.L <ea>,Dr:Dq 64 ÷ 32 ➧ 32r:32q
EOR	Source ⊕ Destination ➧ Destination	EOR Dn,<ea>
EORI	Immediate Data ⊕ Destination ➧ Destination	EORI #<data>,<ea>
EORI to CCR	Source ⊕ CCR ➧ CCR	EORI #<data>,CCR
EORI to SR	If supervisor state then Source ⊕ SR ➧ SR else TRAP	EORI #<data>,SR
EXG	Rx ➧ ➧ Ry	EXG Dx,Dy EXG Ax,Ay EXG Dx,Ay EXG Ay,Dx
EXT	Destination Sign – Extended ➧ Destination	EXT.W Dn extend byte to word EXT.L Dn extend word to long word
JMP	Destination Address ➧ PC	JMP <ea>
JSR	SP – 4 ➧ SP; PC ➧ (SP) Destination Address ➧ PC	JSR <ea>
LEA	<ea> ➧ An	LEA <ea>,An
LINK	SP – 4 ➧ SP; An ➧ (SP) SP ➧ An, SP+d ➧ SP	LINK An,d_n
LSL, LSR	Destination Shifted by count ➧ Destination	LSd Dx,Dy[1] LSd #<data>,Dy[1] LSd <ea>[1]
MOVE	Source ➧ Destination	MOVE <ea>,<ea>
MOVE from SR	If supervisor state then SR ➧ Destination else TRAP	MOVE SR,<ea>
MOVE to CCR	Source ➧ CCR	MOVE <ea>,CCR
MOVE to SR	If supervisor state then Source ➧ SR else TRAP	MOVE <ea>,SR
MOVE USP	If supervisor state then USP ➧ An or An ➧ USP else TRAP	MOVE USP,An MOVE An,USP

Table 4-4. EC000 Core Instruction Set Summary (Continued)

Opcode	Operation	Syntax
MOVEA	Source → Destination	MOVEA <ea>,An
MOVEM	Registers → Destination Source → Registers	MOVEM <list>,<ea>[2] MOVEM <ea>,<list>[2]
MOVEP	Source → Destination	MOVEP Dx,(d_n,Ay) MOVEP (d_n,Ay),Dx
MOVEQ	Immediate Data → Destination	MOVEQ #<data>,Dn
MULS	Source × Destination → Destination	MULS.W <ea>,Dn 16 × 16 → 32 MULS.L <ea>,Dl 32 × 32 → 32 MULS.L <ea>,Dh–Dl 32 × 32 → 64
MULU	Source × Destination → Destination	MULU.W <ea>,Dn 16 × 16 → 32 MULU.L <ea>,Dl 32 × 32 → 32 MULU.L <ea>,Dh–Dl 32 × 32 → 64
NBCD	0 – (Destination$_{10}$) – X → Destination	NBCD <ea>
NEG	0 – (Destination) → Destination	NEG <ea>
NEGX	0 – (Destination) – X → Destination	NEGX <ea>
NOP	None	NOP
NOT	~ Destination → Destination	NOT <ea>
OR	Source V Destination → Destination	OR <ea>,Dn OR Dn,<ea>
ORI	Immediate Data V Destination → Destination	ORI #<data>,<ea>
ORI to CCR	Source V CCR → CCR	ORI #<data>,CCR
ORI to SR	If supervisor state then Source V SR → SR else TRAP	ORI #<data>,SR
PEA	SP – 4 → SP; <ea> → (SP)	PEA <ea>
RESET	If supervisor state then Assert $\overline{\text{RSTO}}$ Line else TRAP	RESET
ROL, ROR	Destination Rotated by count → Destination	ROd Rx,Dy[1] ROd #<data>,Dy[1]
ROXL, ROXR	Destination Rotated with X by count → Destination	ROXd Dx,Dy[1] ROXd #<data>,Dy[1] ROXd <ea>[1]
RTE	If supervisor state then (SP) → SR; SP + 2 → SP; (SP) → PC; SP + 4 → SP; restore state and deallocate stack according to (SP) else TRAP	RTE
RTR	(SP) → CCR; SP + 2 → SP; (SP) → PC; SP + 4 → SP	RTR
RTS	(SP) → PC; SP + 4 → SP	RTS

Table 4-4. EC000 Core Instruction Set Summary (Concluded)

Opcode	Operation	Syntax
SBCD	Destination$_{10}$ – Source$_{10}$ – X ➧ Destination	SBCD Dx,Dy SBCD –(Ax),–(Ay)
Scc	If condition true then 1s ➧ Destination else 0s ➧ Destination	Scc <ea>
STOP	If supervisor state then Immediate Data ➧ SR; STOP else TRAP	STOP #<data>
SUB	Destination – Source ➧ Destination	SUB <ea>,Dn SUB Dn,<ea>
SUBA	Destination – Source ➧ Destination	SUBA <ea>,An
SUBI	Destination – Immediate Data ➧ Destination	SUBI #<data>,<ea>
SUBQ	Destination – Immediate Data ➧ Destination	SUBQ #<data>,<ea>
SUBX	Destination – Source – X ➧ Destination	SUBX Dx,Dy SUBX –(Ax),–(Ay)
SWAP	Register 31–16 ⟷ Register 15–0	SWAP Dn
TAS	Destination Tested ➧ Condition Codes; 1 ➧ bit 7 of Destination	TAS <ea>
TRAP	SSP – 2 ➧ SSP; Format ÷ Offset ➧ (SSP); SSP – 4 ➧ SSP; PC ➧ (SSP); SSP – 2 ➧ SSP; SR ➧ (SSP); Vector Address ➧ PC	TRAP #<vector>
TRAPV	If V then TRAP	TRAPV
TST	Destination Tested ➧ Condition Codes	TST <ea>
UNLK	An ➧ SP; (SP) ➧ An; SP + 4 ➧ SP	UNLK An

NOTES:
1. d is direction, left or right.
2. List refers to register.

4.5 EXCEPTION PROCESSING

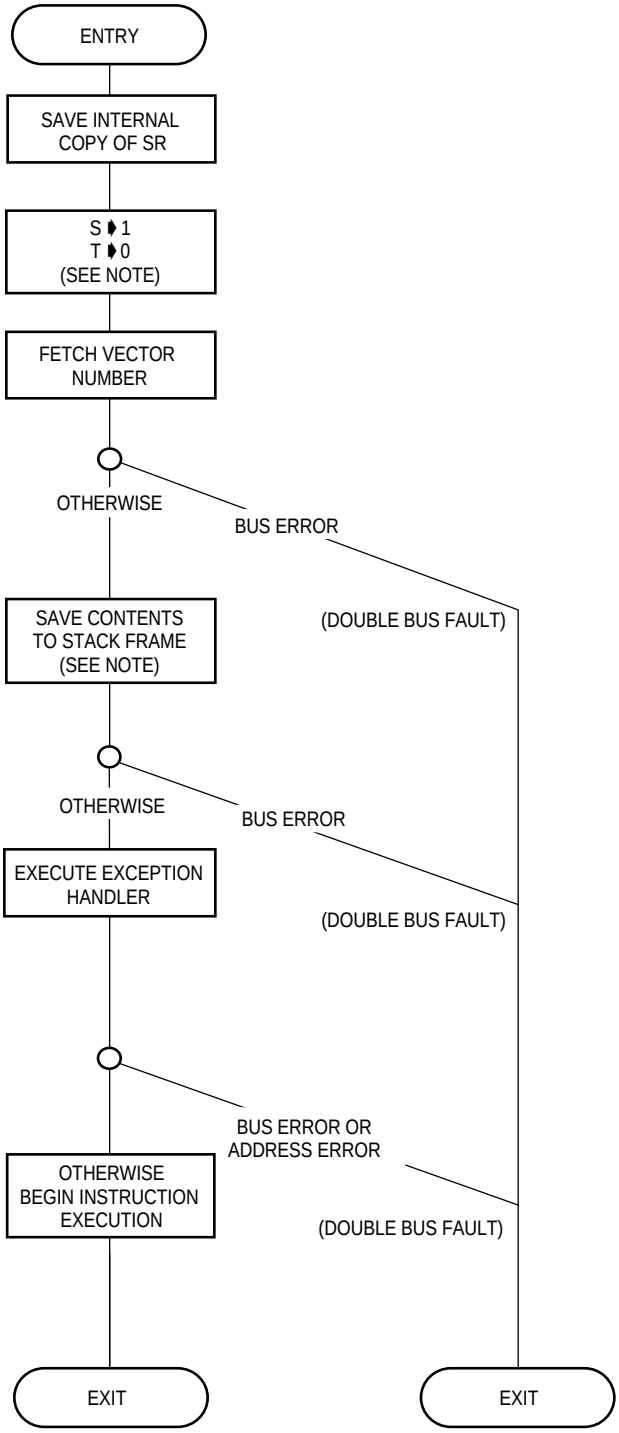
This section describes the processing for each type of exception, exception priorities, the return from an exception, and bus fault recovery. This section also describes the formats of the exception stack frames.

Exception processing is the activity performed by the processor in preparing to execute a special routine for any condition that causes an exception. In particular, exception processing does not include the execution of the routine itself. Exception processing is the transition from the normal processing of a program to the processing required for any special internal or external condition that preempts normal processing. External conditions that cause exceptions are interrupts from external devices, bus errors, and resets. Internal conditions that cause exceptions are instructions, address errors, and tracing. For example, the TRAP, TRAPV, CHK, RTE, and DIV instructions can generate exceptions as part of their normal execution. In addition, illegal instructions and privilege violations cause exceptions. Exception processing uses an exception vector table and an exception stack frame.

Exception processing occurs in four functional steps. However, all individual bus cycles associated with exception processing (vector acquisition, stacking, etc.) are not guaranteed to occur in the order in which they are described in this section. Figure 4-3 illustrates a general flowchart for the steps taken by the processor during exception processing.

During the first step, the processor makes an internal copy of the status register (SR). Then the processor changes to the supervisor mode by setting the S-bit and inhibits tracing of the exception handler by clearing the trace enable (T) bit in the SR. For the reset and interrupt exceptions, the processor also updates the interrupt priority mask in the SR.

During the second step, the processor determines the vector number for the exception. For interrupts, the processor performs an interrupt acknowledge bus cycle to obtain the vector number. For all other exceptions, internal logic provides the vector number. This vector number is used in the last step to calculate the address of the exception vector. Throughout this section, vector numbers are given in decimal notation.



NOTE: These blocks vary for reset and interrupt exceptions.

EC28

Figure 4-3. General Exception Processing Flowchart

The third step is to save the current processor contents for all exceptions other than reset exception, which does not stack information. The processor creates an exception stack frame on the active supervisor stack and fills it with information appropriate for the type of exception. Other information can also be stacked, depending on which exception is being processed and the state of the processor prior to the exception. Figure 4-4 illustrates the general form of the exception stack frame.

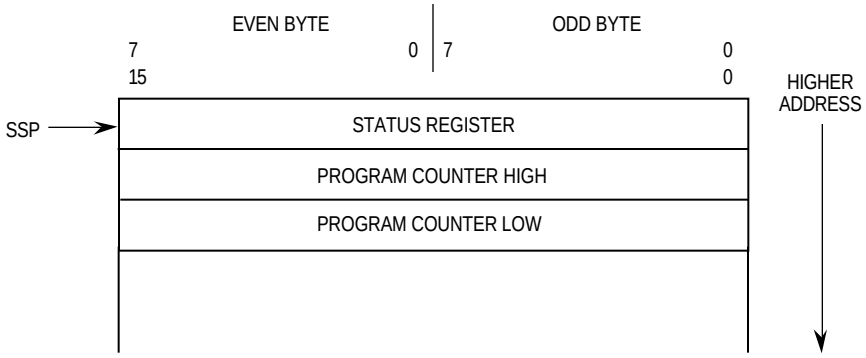


Figure 4-4. General Form of Exception Stack Frame

The last step initiates execution of the exception handler. The new program counter value is fetched from the exception vector. The processor then resumes instruction execution. The instruction at the address in the exception vector is fetched, and normal instruction decoding and execution is started.

4.5.1 Exception Vectors

An exception vector is a memory location from which the processor fetches the address of a routine to handle an exception. Each exception type requires a handler routine and a unique vector. All exception vectors are two words in length (see Figure 4-5), except for the reset vector, which is four words long. All exception vectors reside in the supervisor data space, except for the reset vector, which is in the supervisor program space. A vector number is an 8-bit number that is multiplied by four to obtain the offset of an exception vector. Vector numbers are generated internally or externally, depending on the cause of the exception. For interrupts, during the interrupt acknowledge bus cycle, a peripheral provides an 8-bit vector number (see Figure 4-6) to the processor on data bus lines D7–D0.

The processor forms the vector offset by left-shifting the vector number two bit positions and zero-filling the upper-order bits to obtain a 32-bit long-word vector offset. In the EC000 core this offset is used as the absolute address to obtain the exception vector itself, which is illustrated in Figure 4-6.

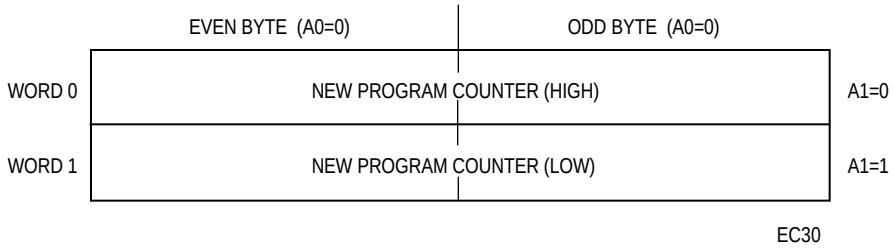


Figure 4-5. Exception Vector Format

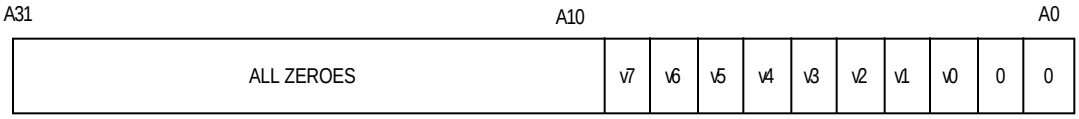


Figure 4-6. Address Translated from 8-Bit Vector Number

The actual address on the address bus is truncated to the number of address bits available on the bus of the particular implementation of the M68000 architecture. In the EC000 core, this is 24 address bits. The memory map for exception vectors is shown in Table 4-5.

The vector table is 512 words long (1024 bytes), starting at address 0 (decimal) and proceeding through address 1023 (decimal). The vector table provides 255 unique vectors, some of which are reserved for trap and other system function vectors. Of the 255, 192 are reserved for user interrupt vectors. However, the first 64 entries are not protected, so user interrupt vectors may overlap at the discretion of the systems designer.

Table 4-5. Exception Vector Assignments

Vector Number(s)	Vector Offset (Hex)	Space[6]	Assignment
0	000	SP	Reset Initial Interrupt Stack Pointer[2]
1	004	SP	Reset Initial Program Counter[2]
2	008	SD	Bus Error
3	00C	SD	Address Error
4	010	SD	Illegal Instruction
5	014	SD	Integer Divide by Zero
6	018	SD	CHK Instruction
7	01C	SD	TRAPV Instruction
8	020	SD	Privilege Violation
9	024	SD	Trace
10	028	SD	Line 1010 Emulator (Unimplemented A-Line Opcode)
11	02C	SD	Line 1111 Emulator (Unimplemented F-Line Opcode)
12[1]	030	—	(Unassigned, Reserved)
13[1]	034	—	(Unassigned, Reserved)
14	038	SD	Format Error[5]
15	03C	SD	Uninitialized Interrupt Vector
16–23[1]	040–05C	—	(Unassigned, Reserved)
24	060	SD	Spurious Interrupt[3]
25	064	SD	Level 1 Interrupt Autovector
26	068	SD	Level 2 Interrupt Autovector
27	06C	SD	Level 3 Interrupt Autovector
28	070	SD	Level 4 Interrupt Autovector
29	074	SD	Level 5 Interrupt Autovector
30	078	SD	Level 6 Interrupt Autovector
31	07C	SD	Level 7 Interrupt Autovector
32–47	080–0BC	SD	TRAP #0–15 Instruction Vectors[4]
48–63[1]	0C0–0FC	—	(Unassigned, Reserved)
64–255	100–3FC	SD	User Defined Vectors

NOTES:
1. Vector numbers 12, 13, 16–23, and 48–63 are reserved for future enhancements by Motorola. No user peripheral devices should be assigned these numbers.
2. Reset vector (0) requires four words, unlike the other vectors which only require two words, and is located in the supervisor program space.
3. The spurious interrupt vector is taken when there is a bus error indication during interrupt processing.
4. TRAP #n uses vector number 32+ n.
5. Reserved.
6. SP denotes supervisor program space, and SD denotes supervisor data space.

4.6 PROCESSING OF SPECIFIC EXCEPTIONS

The exceptions are classified according to their sources, and each type is processed differently. The following paragraphs describe in detail the types of exceptions and the processing of each type.

4.6.1 Reset Exception

The reset exception corresponds to the highest exception level. The processing of the reset exception is performed for system initiation and recovery from catastrophic failure. Any processing in progress at the time of the reset is aborted and cannot be recovered. The processor is forced into the supervisor state, and the trace state is forced off. The interrupt priority mask is set at level 7. The vector number is internally generated to reference the reset exception vector at location 0 in the supervisor program space. Because no assumptions can be made about the validity of register contents, in particular the SSP, neither the program counter nor the status register are saved. The address in the first two words of the reset exception vector is fetched as the initial SSP, and the address in the last two words of the reset exception vector is fetched as the initial program counter. Finally, instruction execution is started at the address in the program counter. The initial program counter should point to the power-up/restart code.

The RESET instruction does not cause a reset exception; it asserts the $\overline{\text{RESET}}$ signal to reset external devices, which allows the software to reset the system to a known state and continue processing with the next instruction.

4.6.2 Interrupt Exceptions

Seven levels of interrupt priorities are provided, numbered from 1–7. Level 7 has the highest priority. Devices can be chained externally within interrupt priority levels, allowing an unlimited number of peripheral devices to interrupt the processor. The status register contains a 3-bit mask indicating the current interrupt priority, and interrupts are inhibited for all priority levels less than or equal to the current priority. Priority level 7 is a special case. Level 7 interrupts cannot be inhibited by the interrupt priority mask, thus providing a non-maskable interrupt capability. An interrupt is generated each time the interrupt request level changes from some lower level to level 7. A level 7 interrupt may still be caused by the level comparison if the request level is a 7 and the processor priority is set to a lower level by an instruction.

An interrupt request is made to the processor by encoding the interrupt request level on the $\overline{\text{IPL2}}$–$\overline{\text{IPL0}}$; a zero indicates no interrupt request. Interrupt requests arriving at the processor do not force immediate exception processing, but the requests are made pending. Pending interrupts are detected between instruction executions. If the priority of the pending interrupt is lower than or equal to the current processor priority, execution continues with the next instruction, and the interrupt exception processing is postponed until the priority of the pending interrupt becomes greater than the current processor priority.

If the priority of the pending interrupt is greater than the current processor priority, the exception processing sequence is started. A copy of the status register is saved; the privilege mode is set to supervisor mode; tracing is suppressed; and the processor priority level is set to the level of the interrupt being acknowledged. The processor fetches the vector number from the interrupting device by executing an interrupt acknowledge cycle, which displays the level number of the interrupt being acknowledged on the address bus. If external logic requests an automatic vector, the processor internally generates a vector number corresponding to the interrupt level number. If external logic indicates a bus error,

the interrupt is considered spurious, and the generated vector number references the spurious interrupt vector. The processor then proceeds with the usual exception processing. The saved value of the program counter is the address of the instruction that would have been executed had the interrupt not been taken. The appropriate interrupt vector is fetched and loaded into the program counter, and normal instruction execution commences in the interrupt handling routine.

4.6.3 Uninitialized Interrupt Exception

An interrupting device provides a EC000 core interrupt vector number and asserts data transfer acknowledge ($\overline{\text{DTACK}}$) or bus error ($\overline{\text{BERR}}$) during an interrupt acknowledge cycle by the EC000 core. If the vector register has not been initialized, the responding M68000 family peripheral provides vector number 15, the uninitialized interrupt vector. This response conforms to a uniform way to recover from a programming error.

4.6.4 Spurious Interrupt Exception

During the interrupt acknowledge cycle, if no device responds by asserting $\overline{\text{DTACK}}$, $\overline{\text{BERR}}$ should be asserted to terminate the vector acquisition. The processor separates the processing of this error from bus error by forming a short format exception stack and fetching the spurious interrupt vector instead of the bus error vector. The processor then proceeds with the usual exception processing.

4.6.5 Instruction Traps

Traps are exceptions caused by instructions; they occur when a processor recognizes an abnormal condition during instruction execution or when an instruction is executed that normally traps during execution.

Exception processing for traps is straightforward. The status register is copied; the supervisor mode is entered; and tracing is turned off. The vector number is internally generated; for the TRAP instruction, part of the vector number comes from the instruction itself. The program counter, and the copy of the status register are saved on the supervisor stack. The saved value of the program counter is the address of the instruction following the instruction that generated the trap. Finally, instruction execution commences at the address in the exception vector.

Some instructions are used specifically to generate traps. The TRAP instruction always forces an exception and is useful for implementing system calls for user programs. The TRAPV and CHK instructions force an exception if the user program detects a run-time error, which may be an arithmetic overflow or a subscript out of bounds. A signed divide (DIVS) or unsigned divide (DIVU) instruction forces an exception if a division operation is attempted with a divisor of zero.

4.6.6 Illegal and Unimplemented Instructions

Illegal instruction is the term used to refer to any of the word bit patterns that do not match the bit pattern of the first word of a legal processor instruction. If such an instruction is fetched, an illegal instruction exception occurs. Motorola reserves the right to define

instructions using the opcodes of any of the illegal instructions. Three bit patterns always force an illegal instruction trap on all M68000 family-compatible microprocessors. The patterns are: $4AFA, $4AFB, and $4AFC. Two of the patterns, $4AFA and $4AFB, are reserved for Motorola system products. The third pattern, $4AFC, is reserved for customer use (as the take illegal instruction trap (ILLEGAL) instruction).

Word patterns with bits 15–12 equaling 1010 or 1111 are distinguished as unimplemented instructions, and separate exception vectors are assigned to these patterns to permit efficient emulation. These separate vectors allow the operating system to emulate unimplemented instructions in software.

Exception processing for illegal instructions is similar to that for traps. After the instruction is fetched and decoding is attempted, the processor determines that execution of an illegal instruction is being attempted and starts exception processing. The exception stack frame is then pushed on the supervisor stack, and the illegal instruction vector is fetched.

4.6.7 Privilege Violations

To provide system security, various instructions are privileged. An attempt to execute one of the privileged instructions while in the user mode causes an exception. The privileged instructions are as follows:

AND Immediate to SR
EOR Immediate to SR
MOVE to SR
MOVE from SR
MOVEC
MOVES
MOVE USP
OR Immediate to SR
RESET
RTE
STOP

Exception processing for privilege violations is nearly identical to that for illegal instructions. After the instruction is fetched and decoded and the processor determines that a privilege violation is being attempted, the processor starts exception processing. The status register is copied; the supervisor mode is entered; and tracing is turned off. The vector number is generated to reference the privilege violation vector, and the current program counter and the copy of the status register are saved on the supervisor stack. The saved value of the program counter is the address of the first word of the instruction causing the privilege violation. Finally, instruction execution commences at the address in the privilege violation exception vector.

4.6.8 Tracing

To aid in program development, the EC000 core includes a facility to allow tracing following each instruction. When tracing is enabled, an exception is forced after each instruction is executed. Thus, a debugging program can monitor the execution of the program under test.

The trace facility is controlled by the T-bit in the supervisor portion of the status register. If the T-bit is cleared (off), tracing is disabled and instruction execution proceeds from instruction to instruction as normal. If the T-bit is set (on) at the beginning of the execution of an instruction, a trace exception is generated after the instruction is completed. If the

instruction is not executed because an interrupt is taken or because the instruction is illegal or privileged, the trace exception does not occur. The trace exception also does not occur if the instruction is aborted by a reset, bus error, or address error exception. If the instruction is executed and an interrupt is pending on completion, the trace exception is processed before the interrupt exception. During the execution of the instruction, if an exception is forced by that instruction, the exception processing for the instruction exception occurs before that of the trace exception.

As an extreme illustration of these rules, consider the arrival of an interrupt during the execution of a TRAP instruction while tracing is enabled. First, the trap exception is processed, then the trace exception, and finally the interrupt exception. Instruction execution resumes in the interrupt handler routine.

After the execution of the instruction is complete and before the start of the next instruction, exception processing for a trace begins. A copy is made of the status register. The transition to supervisor mode is made, and the T-bit of the status register is turned off, disabling further tracing. The vector number is generated to reference the trace exception vector, and the current program counter and the copy of the status register are saved on the supervisor stack. The saved value of the program counter is the address of the next instruction. Instruction execution commences at the address contained in the trace exception vector.

4.6.9 Bus Error

When a bus error exception occurs, the current bus cycle is aborted. The current processor activity, whether instruction or exception processing, is terminated, and the processor immediately begins exception processing.

Exception processing for a bus error follows the usual sequence of steps. The status register is copied, the supervisor mode is entered, and tracing is turned off. The vector number is generated to refer to the bus error vector. Since the processor is fetching the instruction or an operand when the error occurs, the context of the processor is more detailed. To save more of this context, additional information is saved on the supervisor stack. The program counter and the copy of the status register are saved. The value saved for the program counter is advanced 2–10 bytes beyond the address of the first word of the instruction that made the reference causing the bus error. If the bus error occurred during the fetch of the next instruction, the saved program counter has a value in the vicinity of the current instruction, even if the current instruction is a branch, a jump, or a return instruction. In addition to the usual information, the processor saves its internal copy of the first word of the instruction being processed and the address being accessed by the aborted bus cycle. Specific information about the access is also saved: type of access (read or write), processor activity (processing an instruction), and function code outputs when the bus error occurred. The processor is processing an instruction if it is in the normal state or processing a group 2 exception; the processor is not processing an instruction if it is processing a group 0 or a group 1 exception. Figure 4-7 illustrates how this information is organized on the supervisor stack. If a bus error occurs during the last step of exception processing, while either reading the exception vector or fetching the instruction, the value of the program counter is the address of the exception vector. Although this information is not generally sufficient to effect full recovery from the bus

error, it does allow software diagnosis. Finally, the processor commences instruction processing at the address in the vector. It is the responsibility of the error handler routine to clean up the stack and determine where to continue execution.

If a bus error occurs during the exception processing for a bus error, an address error, or a reset, the processor halts and all processing ceases. This halt simplifies the detection of a catastrophic system failure, since the processor removes itself from the system to protect memory contents from erroneous accesses. Only an external reset operation can restart a halted processor.

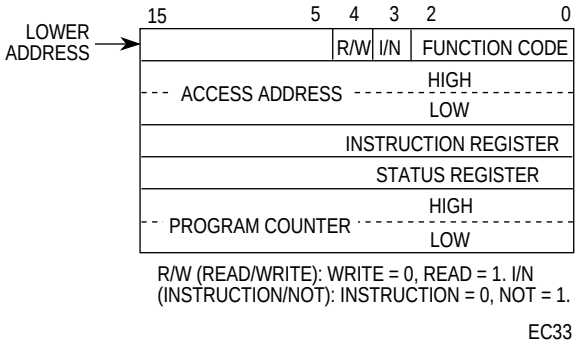


Figure 4-7. Supervisor Stack Order for Bus or Address Error Exception

4.6.10 Address Error

An address error exception occurs when the processor attempts to access a word or long-word operand or an instruction at an odd address. An address error is similar to an internally generated bus error. The bus cycle is aborted, and the processor ceases current processing and begins exception processing. The exception processing sequence is the same as that for a bus error, including the information to be stacked, except that the vector number refers to the address error vector. Likewise, if an address error occurs during the exception processing for a bus error, address error, or reset, the processor is halted.

4.6.11 Multiple Exceptions

When multiple exceptions occur simultaneously, they are processed according to a fixed priority. Table 4-6 lists the exceptions, grouped by characteristics, with group 0 as the highest priority. Within group 0, reset has highest priority, followed by address error and then bus error. Within group 1, trace has priority over external interrupts, which in turn takes priority over illegal instruction and privilege violation. Since only one instruction can be executed at a time, no priority relationship applies within group 2.

Table 4-6. Exception Grouping and Priority

Group	Exception	Processing
0	Reset, Address Error, and Bus Error	Exception processing begins within two clock cycles.
1	Trace, Interrupt, Illegal, and Privilege	Exception processing begins before the next instruction.
2	TRAP, TRAPV, CHK, and DIV	Exception processing is started by normal instruction execution.

The priority relationship between two exceptions determines which is taken, or taken first, if the conditions for both arise simultaneously. Therefore, if a bus error occurs during a TRAP instruction, the bus error takes precedence, and the TRAP instruction processing is aborted. In another example, if an interrupt request occurs during the execution of an instruction while the T-bit in the status register (SR) is asserted, the trace exception has priority and is processed first. Before instruction execution resumes, however, the interrupt exception is also processed, and instruction processing finally commences in the interrupt handler routine. As a general rule, the lower the priority of an exception, the sooner the handler routine for that exception executes. This rule does not apply to the reset exception; its handler is executed first even though it has the highest priority, because the reset operation clears all other exceptions.

SECTION 5
SYSTEM OPERATION

This section contains detailed descriptions and programming information for the system functions and registers outside the EC000 core in the MC68306.

NOTE

None of the MC68306 internal resources are accessible by an external bus master. The following address map and operation descriptions apply only to accesses by the internal EC000 core.

The effect of the RESET instruction and external assertion of the hardware $\overline{\text{RESET}}$ signal on MC68306 components is:

	External $\overline{\text{RESET}}$	**RESET Instruction**
EC000 Core	Yes	No
Serial Module	Yes	Yes
MC68306 Registers	See Individual Descriptions	No

5.1 MC68306 ADDRESS SPACE

The full 32-bit address capability of the MC68306 (corresponding to a 4-Gbyte address space) is decoded internally. A small portion of this address space is devoted to internal resources such as the serial module, configuration registers, and parallel ports. Table 5-1 is a memory map of the MC68306.

Table 5-1. MC68306 Memory Map

FC	A(31–0)	D(15–8) (EVEN ADDRESS)	D(7–0) (ODD ADDRESS)
5	FFFFFFFE/F	SYSTEM	TIMER VECTOR
5	FFFFFFFC/D	REFRESH RATE	BUS TIMEOUT PERIOD
5	FFFFFFFA	INTERRUPT CONTROL REGISTER	
5	FFFFFFF8	INTERRUPT STATUS REGISTER	
5	FFFFFFF6	RESERVED[2]	
5	FFFFFFF4/5	PORT A PIN ASSIGNMENT	PORT B PIN ASSIGNMENT
5	FFFFFFF2/3	PORT A DATA DIRECTION	PORT B DATA DIRECTION
5	FFFFFFF0/1	PORT A DATA	PORT B DATA
5	FFFFFFEF–FFFFFFE8	RESERVED[2]	
5	FFFFFFE6 FFFFFFE4	DRAM BANK 1 CONFIGURATION (LOW) DRAM BANK 1 CONFIGURATION (HIGH)	
5	FFFFFFE2 FFFFFFE0	DRAM BANK 0 CONFIGURATION (LOW) DRAM BANK 0 CONFIGURATION (HIGH)	
5	FFFFFFDE FFFFFFDC	CHIP SELECT 7 CONFIGURATION (LOW) CHIP SELECT 7 CONFIGURATION (HIGH)	
5	FFFFFFDA FFFFFFD8	CHIP SELECT 6 CONFIGURATION (LOW) CHIP SELECT 6 CONFIGURATION (HIGH)	
5	FFFFFFD6 FFFFFFD4	CHIP SELECT 5 CONFIGURATION (LOW) CHIP SELECT 5 CONFIGURATION (HIGH)	
5	FFFFFFD2 FFFFFFD0	CHIP SELECT 4 CONFIGURATION (LOW) CHIP SELECT 4 CONFIGURATION (HIGH)	
5	FFFFFFCE FFFFFFCC	CHIP SELECT 3 CONFIGURATION (LOW) CHIP SELECT 3 CONFIGURATION (HIGH)	
5	FFFFFFCA FFFFFFC8	CHIP SELECT 2 CONFIGURATION (LOW) CHIP SELECT 2 CONFIGURATION (HIGH)	
5	FFFFFFC6 FFFFFFC4	CHIP SELECT 1 CONFIGURATION (LOW) CHIP SELECT 1 CONFIGURATION (HIGH)	
5	FFFFFFC2 FFFFFFC0	CHIP SELECT 0 CONFIGURATION (LOW) CHIP SELECT 0 CONFIGURATION (HIGH)	
5	FFFFFFBF–FFFFF800	RESERVED[3]	
5	FFFFF7FF–FFFFF7E0	RESERVED[2]	SERIAL MODULE
5	FFFFF7DF–FFFFF000	RESERVED[4]	
1, 2, 6	FFFFFFFF–FFFFF000	AVAILABLE FOR CHIP SELECT/DRAM	
1, 2, 5, 6	FFFFEFFF–00000000	AVAILABLE FOR CHIP SELECT/DRAM	
7	–	INTERRUPT ACKNOWLEDGE: VECTOR SUPPLIED ON D7–D0	

1. A(31–24) are copied from A23 (sign-extended) in 16 Mbyte emulation mode.
2. Write data ignored, read data indeterminate.
3. Duplicate of FFFFFFC0–FFFFFFFF.
4. Duplicate of FFFF7FE0–FFFF7FFF

5.2 REGISTER DESCRIPTION

The following paragraphs describe the registers in the MC68306. The address of the register is listed above the register. The numbers in the first row are the bit positions of each bit in the register. The second row is the bit mnemonic. The reset value for each bit is listed beneath the bit mnemonic. Where the reset value is U, the value is indeterminate after power-up, and not affected by reset.

5.2.1 System Register

The system register controls system functions. The value of the AMODE bit after reset is the value of the AMODE pin latched at reset.

FFFFFFFE

15	14	13	12	11	10	9	8
BTERR	BTEN	0	AMODE	0	DUIPL2	DUIPL1	DUIPL0
RESET: 0	0	0	AMODE	0	1	0	0

SUPERVISOR ONLY

BTERR—Bus Timeout Error

This bit is read-only, and is cleared when read. Writes to this bit are ignored.

0 = No bus timeout bus error.
1 = Bus timeout bus error occurred.

BTEN—Bus Timeout Enable

This bit is used to enable the bus timeout timer.

0 = Bus timeout timer is disabled.
1 = Bus timeout timer is enabled.

AMODE—Address Mode

This bit selects the function of the multiplexed address and chip select pins. The address mode pin is latched at the end of reset, so the value must be valid and stable at this time. Writes to this bit are ignored.

0 = Address lines selected.
1 = Chip select lines selected.

DUIPL2–0—DUART Interrupt Priority Level

This bit selects the interrupt priority level for the serial module.

000 = Reserved
001 = Interrupt priority level 1
010 = Interrupt priority level 2
011 = Interrupt priority level 3
100 = Interrupt priority level 4
101 = Interrupt priority level 5
110 = Interrupt priority level 6
111 = Interrupt priority level 7

5.2.2 Timer Vector Register

FFFFFFFF

7	6	5	4	3	2	1	0
TVEC 7	TVEC 6	TVEC 5	TVEC 4	TVEC 3	TVEC 2	TVEC 1	TVEC 0
RESET: 0	0	0	0	1	1	1	1

SUPERVISOR ONLY

TVEC7–0—Timer Vector

The value set in this field supplies the vector for the DUART timer interrupt handler.

5.2.3 Bus Timeout Period Register

FFFFFFFC

7	6	5	4	3	2	1	0
BT7	BT6	BT5	BT4	BT3	BT2	BT1	BT0
RESET: U	U	U	U	U	U	U	U

SUPERVISOR ONLY

A programmable-period timer can generate a bus error to terminate any bus cycle after 16 to 4096 wait states, programmable in 16-wait state increments. The bus timeout timer is enabled by the BTEN bit in the system register.

The bus timeout timer restarts between the read and write portions of a TAS indivisible cycle.

BT7–0—Bus Timeout Period

The value set in this field supplies the bus timeout timer period. The bus timeout timer period can be calculated from the equation:

Period = (16 × (register value +1)) × EXTAL

Where:

EXTAL is the crystal period in nanoseconds and period is in nanoseconds.

5.2.4 Interrupt Registers

Up to seven prioritized external interrupts can be supported by programming the following registers. More interrupt sources can be supported by external daisy-chaining. The interrupt inputs are internally synchronized. Edge-triggered interrupts are not supported.

Each interrupt can be either active-high or active-low. The active level is self-programmed during reset, with no software intervention. Every interrupt must be at its inactive level at the end of reset. Each interrupt can be enabled or disabled by programming the corresponding bit in the interrupt control register.

Each interrupt can be auto-vectored, by programming the interrupt control register. Auto-vectored interrupt acknowledge cycles are zero wait states. If no active interrupt is present at the level being acknowledged, the MC68306 automatically generates a spurious interrupt vector, which is a zero wait state. Interrupt input synchronization is frozen during an interrupt acknowledge cycle, so the acknowledge can safely be used to automatically negate the interrupt.

5.2.4.1 INTERRUPT CONTROL REGISTER

FFFFFFFA/B

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IENT	IEN7	IEN6	IEN5	IEN4	IEN3	IEN2	IEN1	—	AVEC 7	AVEC6	AVEC5	AVEC4	AVEC3	AVEC 2	AVEC 1
RESET: 0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

SUPERVISOR ONLY

IENT—Timer Interrupt Enable

This bit enables the DUART timer interrupt.

0 = Interrupts disabled.
1 = Interrupts enabled.

IEN7–1—Interrupt Enable 7 through 1

These bits enable interrupt 7, 6, 5, 4, 3, 2, and 1.

0 = Interrupt disabled.
1 = Interrupt enabled.

AVEC7–1—Autovector Enable 7 through 1

These bits enable autovectoring for interrupts 7, 6, 5, 4, 3, 2, and 1.

0 = No autovector.

1 = Autovector.

5.2.4.2 INTERRUPT STATUS REGISTER. An enabled, active interrupt appears as a one in the interrupt status register, regardless of the active voltage level programmed at reset. This register is read-only, writes to this register are ignored.

FFFFFFF8/9

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IRQT	IRQ7	IRQ6	IRQ5	IRQ4	IRQ3	IRQ2	IRQ1	IRQD	IX7	IX6	IX5	IX4	IX3	IX2	IX1
RESET: 0	0	0	0	0	0	0	0	0	IRQ7	PB7/IRQ6	PB6/IRQ5	IRQ4	PB5/IRQ3	PB4/IRQ2	IRQ1

SUPERVISOR ONLY

IRQT—DUART Timer Interrupt State

0 = No interrupt.
1 = Interrupt asserted.

IRQ7–1—Interrupt Request 7 through 1

These bits indicate interrupt status for the external interrupts 7, 6, 5, 4, 3, 2, and 1.

0 = No interrupt.
1 = Interrupt asserted.

IRQD—DUART Interrupt State

This bit indicates the DUART interrupt state.

0 = No DUART interrupt.
1 = DUART interrupt asserted

IX7–1—Reset (inactive) level of external interrupts 7 through 1.

0 = Active high interrupt pin.
1 = Active low interrupt pin.

5.2.5 I/O Port Registers

The following paragraphs describe the registers controlling the parallel ports. All port pins are reset to input by a system reset, so pullup or pulldown resistors should be added externally as needed. To enable a port A bit as an output, write a one to the appropriate bit position of the port direction register. If a bit is programmed as an output, the data written to the port data register appears in true form at the pin. The data read back from the port pins register is the same level as appears at the pin. The data read from the port data register is the last value written to the register, regardless of the level at the pin. The port data register is not affected by any reset, so it should be initialized before enabling any bits as outputs.

Port B pins can be individually programmed as either IRQ, $\overline{\text{IACK}}$ or parallel port signals. To use any of the port B pins PB7–PB4 as interrupt request signals (IRQ6, IRQ5, IRQ3, IRQ2) be sure the bit is programmed as an input. Interrupt enables are provided for each interrupt level.

To use any of the port B pins PB3–PB0 as $\overline{\text{IACK6}}$, $\overline{\text{IACK5}}$, $\overline{\text{IACK3}}$, or $\overline{\text{IACK2}}$, program the port data bit and the autovector bit to zero.

To use any of the port B pins PB3–PB0 as port inputs, ensure that the autovector bit is one.

Open-drain or open-source operation can be emulated by programming the appropriate fixed data (e.g. 0 = open-drain) and toggling the direction control. PB7–PB4 pins can be programmed as outputs even when enabled as interrupt inputs, allowing inputs to be tested or emulated if the interrupt is open-drain or open-source. The active interrupt level is the inverse of the IX register bit.

5.2.5.1 PORT PINS REGISTER

FFFFFFF4/5

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
RESET: PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0

SUPERVISOR ONLY

The port pin register bits are the data at the port pins, regardless of pin direction. The port pins register is read-only, writes are ignored.

5.2.5.2 PORT DIRECTION REGISTER

FFFFFFF2/3

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PADIR7	PADIR6	PADIR5	PADIR4	PADIR3	PADIR2	PADIR1	PADIR0	PBDIR7	PBDIR6	PBDIR5	PBDIR4	PBDIR3	PBDIR2	PBDIR1	PBDIR0
RESET: 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SUPERVISOR ONLY

The port direction register bits determine the direction of data flow at the port pins.

PADIR7–0—Port A Direction Register Bit 7–0

This bit determines the direction of data flow at port A pins 7 through 0.

0 = Input.
1 = Output.

PBDIR7–0—Port B Direction Register Bit 7–0

This bit determines the direction of data flow at port B pins 7–0.

0 = Input.
1 = Output.

5.2.5.3 PORT DATA REGISTER. The port data register bits return the value as written, regardless of the direction register and pin state. For pins configured as outputs, the corresponding value in the port data register is driven externally.

FFFFFFF0/1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PAD7	PAD6	PAD5	PAD4	PAD3	PAD2	PAD1	PAD0	PBD7	PBD6	PBD5	PBD4	PBD3	PBD2	PBD1	PBD0
RESET: U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

SUPERVISOR ONLY

PAD7–0—Port A Data Bit 7 through 0.

PBD7–0—Port B Data Bit 7 through 0.

5.2.6 Chip Selects

The chip-select outputs are all active-low decodes of the high fifteen internal address bits (A31–A17), the three function code bits, and the read/write cycle type. The active duration of any chip select is the period of the address strobe low and either a data strobe or read/write low. Thus there are separate chip select pulses for the read and write portions of a read-modify-write cycle.

The four mask bits (CSM3–CSM0) are decoded to an n-of-15 mask, where n is the binary value of CSM3–CSM0. On every bus cycle, the n most significant address bits of the range A31–A17 are compared, and the remaining less significant bits are ignored.

The fifteen address bits are first masked by each chip select mask, then compared with each chip select base address (CSA31–CSA17). All CSAx bits not used in the comparison must be zero. The function code is matched with the CSFC qualifiers, and the cycle type is matched with the CSR/CSW qualifiers. If all three qualifiers are successful for any chip select, the cycle is a hit.

If the cycle hits multiple chip selects, the lowest numbered chip select has priority. All chip selects have priority over DRAM. After reset, $\overline{CS0}$ responds to the entire 4 Gbyte address space, except for the range dedicated to internal resources, i.e., $\overline{CS0}$ responds to 00000000–FFFFEFFF. The other chip selects are not affected by any reset, and must be explicitly programmed. This applies to all chip selects, whether used or not.

NOTE

Unused chip selects must be disabled to prevent interference with other chip selects, DRAM, or externally decoded resources.

There are three ways to disable a chip select, corresponding to the three match conditions:

1. All CSFCx bits are zero
2. Both CSW/CSR are zero
3. Any unused CSAx bit is one.

Chip selects 7 through 4 are still matched even if running in address mode (AMODE = 0). They can be disabled, or they can be used to provide automatic $\overline{\text{DTACK}}$ timing for externally decoded resources.

If more decodes are necessary than are supplied on the MC68306, one of the existing chip selects should be used (Figure 5-1) to enable the external decoding, since some signals used to qualify the chip selects are not available externally.

The registers listed below allow the base address, range, and cycle duration of each chip select to be independently programmed. The chip select configuration registers do not support byte writes. The registers can be written as either 16-bit or 32-bit, but 32-bit accesses are preferred. Any write access affects all 16 bits of the high half or low half register. Only chip select 0 is affected by reset.

5.2.6.1 CHIP SELECT CONFIGURATION REGISTERS (HIGH HALF)

FFFFFFC0 (CS0)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSA31	CSA30	CSA29	CSA28	CSA27	CSA26	CSA25	CSA24	CSA23	CSA22	CSA21	CSA20	CSA19	CSA18	CSA17	CSW
RESET: 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

SUPERVISOR ONLY

FFFFFFDC (CS7), FFFFFFD8 (CS6), FFFFFFD4 (CS5), FFFFFFD0 (CS4), FFFFFFCC (CS3), FFFFFFC8 (CS2), FFFFFFC4 (CS1)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSA31	CSA30	CSA29	CSA28	CSA27	CSA26	CSA25	CSA24	CSA23	CSA22	CSA21	CSA20	CSA19	CSA18	CSA17	CSW
RESET: U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

SUPERVISOR ONLY

CSA31–CSA17—Chip Select Address

This bit field selects the base address for each chip select.

CSW—Chip Select Write

This bit determines whether write cycles are permitted to chip select space. If read and write cycles are both inhibited, chip select is inhibited.

0 = Write cycles are inhibited to chip select space
1 = Write cycles are permitted to chip select space

5.2.6.2 CHIP SELECT CONFIGURATION REGISTERS (LOW HALF)

FFFFFFC2 (CS0)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSR	CSFC 6	CSFC 5	—	—	CSFC 2	CSFC 1	—	CSM3	CSM2	CSM1	CSM0	CSDT3	CSDT2	CSDT 1	CSDT 0
RESET: 0	1	1	1	1	1	1	1	0	0	0	0	1	1	1	0

SUPERVISOR ONLY

FFFFFFDE (CS7), FFFFFFDA (CS6), FFFFFFD6 (CS5), FFFFFFD2 (CS4), FFFFFFCE (CS3), FFFFFFCA (CS2), FFFFFFC6 (CS1)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSR	CSFC 6	CSFC 5	—	—	CSFC 2	CSFC 1	—	CSM3	CSM2	CSM1	CSM0	CSDT3	CSDT2	CSDT 1	CSDT 0
RESET: U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

SUPERVISOR ONLY

CSR—Chip Select Read

This bit determines whether read cycles are permitted to chip select space. If read and write cycles are both inhibited, chip select is inhibited.

0 = Read cycles are inhibited to chip select space
1 = Read cycles are permitted to chip select space

CSFC 6, 5, 2, 1—Chip Select Function Code Enable

This bit determines which function code accesses are permitted to chip select space. If all function code cycles are inhibited, chip select is inhibited.

0 = Function code ‘n’ cycles are inhibited to chip select space
1 = Function code ‘n’ cycles are permitted to chip select space

CSM3–0—Chip Select Address Match

This field determines which chip select bits must match address bits for chip select to occur. CSA bits not included in match must be set to zero, or else this chip select is inhibited.

0000 = A31–A17 ignored in chip select address match
0001 = A31 must match CSA31; A30–A17 ignored in chip select address match
0010 = A31–A30 must match CSA31–CSA30; A29–A17 ignored in chip select address match

.....

1111 = A31–A17 must match CSA31–CSA17 in chip select address match

Table 5-2 shows the entire range of address bits that must match for a chip select to occur.

Table 5-2. Chip Select Match Bits

	A31	A30	A29	A28	A27	A26	A25	A24	A23	A22	A21	A20	A19	A18	A17
0000	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0001	•	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0010	•	•	x	x	x	x	x	x	x	x	x	x	x	x	x
0011	•	•	•	x	x	x	x	x	x	x	x	x	x	x	x
0100	•	•	•	•	x	x	x	x	x	x	x	x	x	x	x
0101	•	•	•	•	•	x	x	x	x	x	x	x	x	x	x
0110	•	•	•	•	•	•	x	x	x	x	x	x	x	x	x
0111	•	•	•	•	•	•	•	x	x	x	x	x	x	x	x
1000	•	•	•	•	•	•	•	•	x	x	x	x	x	x	x
1001	•	•	•	•	•	•	•	•	•	x	x	x	x	x	x
1010	•	•	•	•	•	•	•	•	•	•	x	x	x	x	x
1011	•	•	•	•	•	•	•	•	•	•	•	x	x	x	x
1100	•	•	•	•	•	•	•	•	•	•	•	•	x	x	x
1101	•	•	•	•	•	•	•	•	•	•	•	•	•	x	x
1110	•	•	•	•	•	•	•	•	•	•	•	•	•	•	x
1111	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•

x = Address bit is a don't care, CSA bit must be 0 to allow match.

• = Address bit must match CSA bit for chip select to occur.

CSDT3–0—Chip Select $\overline{\text{DTACK}}$ Wait State Selection

This field determines whether automatic $\overline{\text{DTACK}}$ is returned, and how many wait states are inserted if automatic $\overline{\text{DTACK}}$ is enabled. When automatic $\overline{\text{DTACK}}$ is selected, the write portion of a TAS indivisible cycle is the same length as a normal write cycle to the same location. Any external $\overline{\text{DTACK}}$ generation circuit must recognize that $\overline{\text{AS}}$ remains asserted throughout a read-write indivisible cycle, if it supports TAS.

0000 = Automatic $\overline{\text{DTACK}}$, 0 wait states
0001 = Automatic $\overline{\text{DTACK}}$, 1 wait state
0010 = Automatic $\overline{\text{DTACK}}$, 2 wait states
0011 = Automatic $\overline{\text{DTACK}}$, 3 wait states
0100 = Automatic $\overline{\text{DTACK}}$, 4 wait states
0101 = Automatic $\overline{\text{DTACK}}$, 5 wait states
0110 = Automatic $\overline{\text{DTACK}}$, 6 wait states
0111 = Automatic $\overline{\text{DTACK}}$, 7 wait states
1000 = Automatic $\overline{\text{DTACK}}$, 8 wait states
1001 = Automatic $\overline{\text{DTACK}}$, 9 wait states
1010 = Automatic $\overline{\text{DTACK}}$, 10 wait states
1011 = Automatic $\overline{\text{DTACK}}$, 11 wait states
1100 = Automatic $\overline{\text{DTACK}}$, 12 wait states
1101 = Automatic $\overline{\text{DTACK}}$, 13 wait states
1110 = Automatic $\overline{\text{DTACK}}$, 14 wait states
1111 = No automatic $\overline{\text{DTACK}}$, external $\overline{\text{DTACK}}$ required

Figure 5-1 shows a method of expanding the number of chip selects in case more are required for the application.

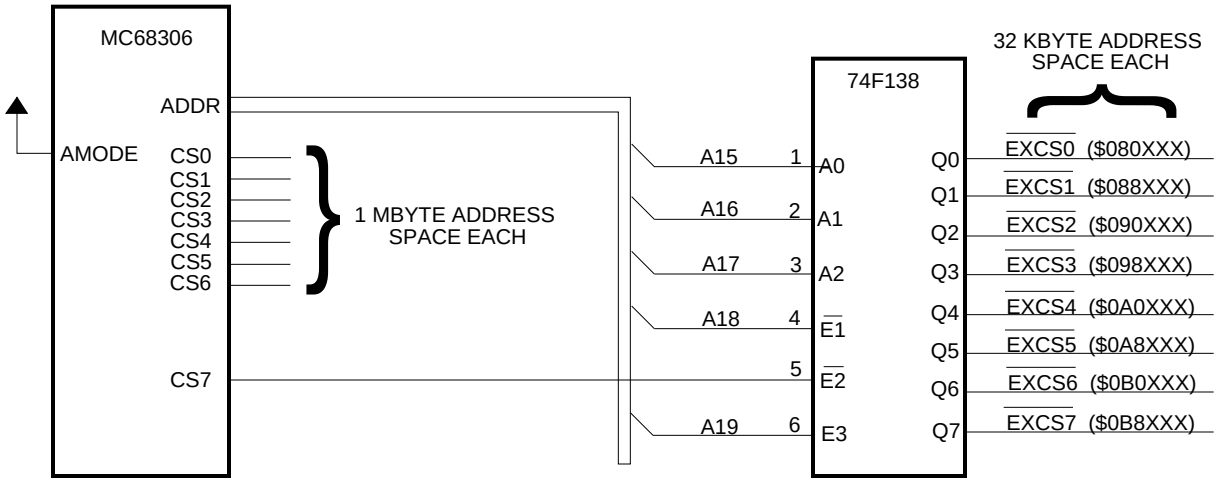


Figure 5-1. Chip Select Expansion

5.2.7 DRAM Control Registers

The DRAM address space decode mechanism is identical to the chip select mechanism. Bank 0 has priority over bank 1, but all chip selects have priority over DRAM. The MC68306 DRAM controller provides CAS-before-RAS refresh only. The refresh timer is a programmable period counter that generates a refresh request every 16 to 4096 EXTAL periods, programmable in 16 EXTAL period increments. Programming the refresh rate is described in paragraph 5.2.7.1. When a refresh is pending, a refresh cycle is inserted at the earliest availability of the RAS/CAS signals. Both banks and both bytes are refreshed together.

The refresh timer is not affected by any reset, and refresh cycles will appear under reset. The refresh timer is initialized by a write to the refresh rate register. When this register is written, the first refresh occurs immediately, so the refresh rate should be programmed after the DRAM configuration register DRDT bit. After power-up, the refresh rate register value is random. If power consumption is critical, the refresh rate should be set as soon as possible. In a system with soft-reset recovery, the hard/soft reset decision could take a long time. A safe algorithm is to read the register first; if it contains the correct value, do nothing. This will not disturb the timer, and the reset recovery can proceed at leisure.

Refresh stops only when the MC68306 is arbitrated off the bus. If the internal EC000 $\overline{BG}$ signal is asserted while a refresh cycle is in progress, the external $\overline{BG}$ signal is delayed until the refresh is complete. However, no refresh will occur during another master's tenure of the bus if the $\overline{BG}$ or $\overline{BGACK}$ signals are recognized before a refresh cycle starts. The task of DRAM refresh must be assumed by any other bus master. The refresh timer is not suspended while the bus is arbitrated away, so a refresh cycle is likely when the

68306 regains bus ownership. Only one refresh cycle occurs after bus ownership is regained, regardless of the time the bus was granted away.

The DRAM controller provides RAS/CAS timing, 15 multiplexed address bits, and refresh timing. All DRAM accesses are either zero or one wait state cycles, unless delayed by a refresh. Zero wait state operation supports DRAMs up to 80 ns RAS access, and one wait state cycles supports DRAMs up to 120 ns RAS access (at 16.67 MHz). External DTACK is not allowed on DRAM accesses. A refresh can add up to three extra wait-states to zero wait-state accesses or 4 extra wait-states to one-wait state accesses. Read-modify-write cycles to DRAM use page mode, and the write portion is always zero wait-state, regardless of the DRDT bit setting.

The organization of external DRAM is one or two banks, by two bytes. $\overline{\text{CAS0}}$ controls the high byte (D15–D8) and $\overline{\text{CAS1}}$ controls the low byte (D7–D0).

The minimum bank size is 128 Kbytes (64K $\times$ 2 bytes), because of the address multiplexer, shown in Table 5-3.

Table 5-3. DRAM Address Multiplexer

Value Of:	At RAS When DRSZ2–0 Is:								At CAS:
	111	110	101	100	011	010	001	000	
DRAMA14	A30	A29	A28	A27	A26	A25	A24	A23	A15
DRAMA13	A29	A28	A27	A26	A25	A24	A23	A22	A14
DRAMA12	A28	A27	A26	A25	A24	A23	A22	A21	A13
DRAMA11	A27	A26	A25	A24	A23	A22	A21	A20	A12
DRAMA10	A26	A25	A24	A23	A22	A21	A20	A19	A11
DRAMA9	A25	A24	A23	A22	A21	A20	A19	A18	A10
DRAMA8	A24	A23	A22	A21	A20	A19	A18	A17	A9
DRAMA7	A23	A22	A21	A20	A19	A18	A17	A16	A8
DRAMA6	A22	A21	A20	A19	A18	A17	A16	A15	A7
DRAMA5	A21	A20	A19	A18	A17	A16	A15	A14	A6
DRAMA4	A20	A19	A18	A17	A16	A15	A14	A13	A5
DRAMA3	A19	A18	A17	A16	A15	A14	A13	A12	A4
DRAMA2	A18	A17	A16	A15	A14	A13	A12	A11	A3
DRAMA1	A17	A16	A15	A14	A13	A12	A11	A10	A2
DRAMA0	A16	A15	A14	A13	A12	A11	A10	A9	A1

Because the DRAM address multiplexer provides contiguous address bits to the full 15-bit DRAMA bus width during RAS, more banks can be supported by externally decoding bits beyond the RAS address width of the DRAMs. If this is done, the DRAMA, $\overline{\text{CAS}}$, and $\overline{\text{DRAMW}}$ signals should be buffered. This will almost certainly require the wait state. Also, DRAMs with more row address pins than column address pins are supported.

5.2.7.1 DRAM REFRESH REGISTER. The refresh timer is a programmable period counter that generates a refresh request every 16 to 4096 EXTAL periods, programmable in 16 EXTAL period increments.

FFFFFFFC

15	14	13	12	11	10	9	8
RR7	RR6	RR5	RR4	RR3	RR2	RR1	RR0
RESET: U	U	U	U	U	U	U	U

SUPERVISOR ONLY

RR7–0—Refresh Rate Period

The value set in this field supplies the refresh rate for the DRAM controller. The refresh rate can be calculated from the equation:

Period = (16 × (register value +1)) × EXTAL

Where:

EXTAL is the crystal period in nanoseconds and period is in nanoseconds.

5.2.7.2 DRAM BANK CONFIGURATION REGISTER (HIGH HALF). The DRAM configuration registers are not affected by any reset, and must be explicitly programmed. This applies to both banks, whether used or not. Unused banks must be disabled to prevent interference with other address decodes.

FFFFFFE4/5 (DR1), FFFFFFE0/1 (DR0)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DRA31	DRA30	DRA29	DRA28	DRA27	DRA26	DRA25	DRA24	DRA23	DRA22	DRA21	DRA20	DRA19	DRA18	DRA17	DRW
RESET: U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

SUPERVISOR ONLY

DRA31–DRA17—DRAM Bank Address

This bit field selects the base address for DRAM bank.

DRW—DRAM Write

This bit determines whether write cycles are permitted to DRAM bank space. If read and write cycles are both inhibited, the DRAM bank is inhibited.

0 = Write cycles are inhibited to DRAM bank space
1 = Write cycles are permitted to DRAM bank space

NOTE

Never perform a TAS instruction to DRAM if the DRAM is configured as write-only.

5.2.7.3 DRAM BANK CONFIGURATION REGISTER (LOW HALF)

FFFFFFE6/7 (DR1), FFFFFFE2/3 (DR0)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DRR	DRFC 6	DRFC 5	—	—	DRFC 2	DRFC 1	—	DRM3	DRM2	DRM1	DRM0	DRSZ2	DRSZ1	DRSZ 0	DRDT
RESET: U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

SUPERVISOR ONLY

DRR—DRAM Read

This bit determines whether read cycles are permitted to DRAM bank space. If read and write cycles are both inhibited, DRAM bank is inhibited.

0 = Read cycles are inhibited to DRAM bank space
1 = Read cycles are permitted to DRAM bank space

DRFC6, 5, 2, 1—DRAM Bank Function Code 6, 5, 2, 1 Enable

This bit determines which function code accesses are permitted to DRAM bank space. If all function code cycles are inhibited, the DRAM bank is inhibited.

0 = Function code n cycles are inhibited to DRAM bank space
1 = Function code n cycles are permitted to DRAM bank space

DRM3–0—DRAM Bank Address Match

This field determines which DRAM bank address bits must match address bits for DRAM bank to occur. DRA bits not included in match must be set to zero, or else DRAM bank is inhibited.

0000 = A31–A17 ignored in DRAM bank address match
0001 = A31 must match DRA31; A30–A17 ignored in DRAM bank address match
0010 = A31–A30 must match DRA31–DRA30; A29–A17 ignored in DRAM bank address match
.....
1111 = A31–A17 must match DRA31–DRA17 in DRAM bank address match

Table 5-4 shows the entire range of address bits that must match for a DRAM bank to occur.

Table 5-4. DRAM Bank Match Bits

	A31	A30	A29	A28	A27	A26	A25	A24	A23	A22	A21	A20	A19	A18	A17
0000	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0001	•	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0010	•	•	x	x	x	x	x	x	x	x	x	x	x	x	x
0011	•	•	•	x	x	x	x	x	x	x	x	x	x	x	x
0100	•	•	•	•	x	x	x	x	x	x	x	x	x	x	x
0101	•	•	•	•	•	x	x	x	x	x	x	x	x	x	x
0110	•	•	•	•	•	•	x	x	x	x	x	x	x	x	x
0111	•	•	•	•	•	•	•	x	x	x	x	x	x	x	x
1000	•	•	•	•	•	•	•	•	x	x	x	x	x	x	x
1001	•	•	•	•	•	•	•	•	•	x	x	x	x	x	x
1010	•	•	•	•	•	•	•	•	•	•	x	x	x	x	x
1011	•	•	•	•	•	•	•	•	•	•	•	x	x	x	x
1100	•	•	•	•	•	•	•	•	•	•	•	•	x	x	x
1101	•	•	•	•	•	•	•	•	•	•	•	•	•	x	x
1110	•	•	•	•	•	•	•	•	•	•	•	•	•	•	x
1111	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•

x = Address bit is a don't care, DRA bit must be 0 to allow match.

• = Address bit must match DRA bit for DRAM bank to occur.

DRSZ—DRAM Size

DRAM address multiplexer provides (8 + DRSZ2–0) CAS address bits.

NOTE

Both DRAM banks must be the same size and speed. The DRAM logic uses the DRSZ and DRDT values programmed in the bank 0 configuration register only. These bits in the bank 1 configuration register are ignored.

DRDT—DRAM Automatic $\overline{\text{DTACK}}$ Response

0 = Automatic $\overline{\text{DTACK}}$, 0 wait states
1 = Automatic $\overline{\text{DTACK}}$, 1 wait state

NOTE

The write portion of a TAS is always 0-wait, regardless of the state of DRDT.

5.2.8 Automatic $\overline{\text{DTACK}}$ Generation

All eight chip selects and both DRAM banks can be independently programmed for automatic $\overline{\text{DTACK}}$ generation. Chip select accesses can be programmed for 0 to 14 wait states or external $\overline{\text{DTACK}}$, supporting memories as slow as 960 ns (at 16.67 MHz) with no external logic. Programming the automatic $\overline{\text{DTACK}}$ for chip selects is described in paragraph 5.2.6.2. Programming the automatic $\overline{\text{DTACK}}$ for DRAM banks is described in paragraph 5.2.7.3.

For the chip select address spaces, if automatic $\overline{\text{DTACK}}$ is enabled, the write portion of a TAS (test and set) indivisible cycle is the same length as a normal write to the same location. Any external $\overline{\text{DTACK}}$ generation circuit must recognize that $\overline{\text{AS}}$ remains asserted throughout a read-write indivisible cycle if it supports TAS.

For the DRAM address spaces, the write portion of a TAS is always 0-wait, regardless of DRDT.

5.3 CRYSTAL OSCILLATOR

The oscillator circuit is designed for applications using a crystal or ceramic resonator operating from 1 MHz to 20 MHz. The bias resistor and small startup capacitors are integrated into the oscillator circuit, shown in Figure 5-2. Depending on the crystal and application, an additional external capacitor may be required (consult crystal vendor for specific information). The following equation can be used to calculate the size of external capacitance:

$$C_L = C_P + \frac{C_{IN} \text{ X } C_{OUT}}{C_{IN} + C_{OUT}}$$

Where:

C_P is the parasitic capacitance, which can be neglected in most cases.

C_{IN} is the total input capacitance, consisting of C_{ext} + C1.

C_{OUT} is the total output capacitance, consisting of C2 and external parasitic capacitances (e.g., board and package capacitances).

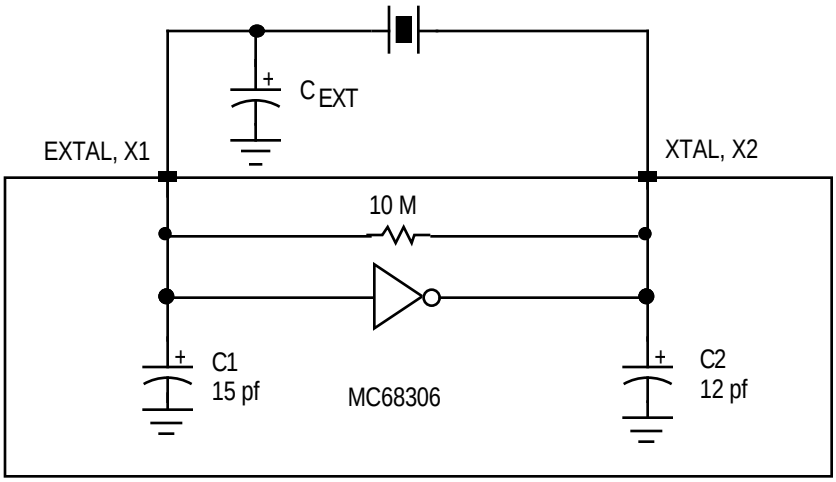


Figure 5-2. Oscillator Circuit Diagram

SECTION 6
SERIAL MODULE

The MC68306 serial module is a dual universal asynchronous/synchronous receiver/transmitter that interfaces directly to the CPU. The serial module, shown in Figure 6-1, consists of the following major functional areas:

- Two Independent Serial Communication Channels (A and B)
- Baud Rate Generator Logic
- Sixteen Bit Timer/Counter
- Internal Channel Control Logic
- Interrupt Control Logic

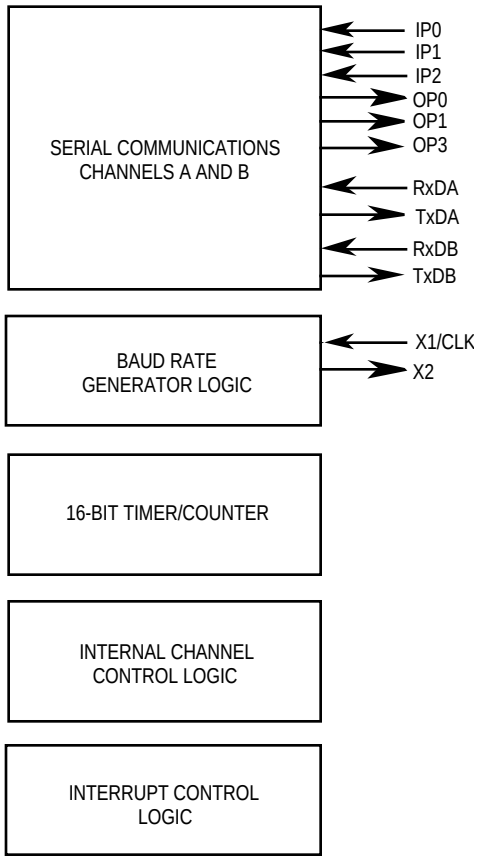


Figure 6-1. Simplified Block Diagram

6.1 MODULE OVERVIEW

Features of the serial module are as follows:

- Two, Independent, Full-Duplex Asynchronous/Synchronous Receiver/Transmitter Channels
- Quadruple-Buffered Receiver
- Double-Buffered Transmitter
- Independently Programmable Baud Rate for Each Receiver and Transmitter Selectable from:
  - —18 Fixed Rates: 50 to 38.4 kBaud
  - —Timer-Generated Baud Rate Up to 170 kbaud
- Programmable Data Format:
  - —Five to Eight Data Bits Plus Parity
  - —Odd, Even, No Parity, or Force Parity
  - —Nine-Sixteenths to Two Stop Bits Programmable in One-Sixteenth Bit Increments
- Programmable Channel Modes:
  - —Normal (Full Duplex)
  - —Automatic Echo
  - —Local Loopback
  - —Remote Loopback
- Automatic Wakeup Mode for Multidrop Applications
- Multi-Function Three-Bit Input Port
  - —Can Be Clock or Control Inputs
  - —Change of State Detection Available
- Multi-Function Three-Bit Output Port
  - —Individual Bit Set/Reset Capability
  - —Can Be Status or Interrupt Signal
- Multi-Function Sixteen-Bit Programmable Counter/Timer
- Eight Maskable Interrupt Conditions
- Timer/Counter Interrupt Can Be Independently Programmed
- 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

6.1.1 Serial Communication Channels A and B

Each communication channel provides a full-duplex asynchronous/synchronous receiver and transmitter using an operating frequency independently selected from a baud rate generator or an external clock input.

The transmitter accepts parallel data from the bus, converts it to a serial bit stream, inserts the appropriate start, stop, and optional parity bits, then outputs a composite serial data stream on the channel transmitter serial data output (TxDx). Refer to **6.3.2.1 Transmitter** for additional information.

The receiver accepts serial data on the channel receiver serial data input (RxDx), 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. Refer to **6.3.2.2 Receiver** for additional information.

6.1.2 Baud Rate Generator Logic

The crystal oscillator operates directly from a 3.6864-MHz crystal connected across the X1/CLK input and the X2 output or from an external clock of the same frequency connected to X1/CLK. The clock serves as the basic timing reference for the baud rate generator and other internal circuits.

The baud rate generator operates from the oscillator or external CMOS clock input and is capable of generating 18 commonly used data communication baud rates ranging from 50 to 38.4k by producing internal clock outputs at 16 times the actual baud rate. Refer to **6.2 Serial Module Signal Definitions** and **6.3.1 Baud Rate Generator** for additional information.

6.1.3 Timer/Counter

The timer/counter provides for an input which bypasses the 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. The external clock input allows the user to use the external input as the only clock source for the serial module if multiple baud rates are not required.

6.1.4 Interrupt Control Logic

Two interrupt request signals ($\overline{\text{IRQ}}$ and $\overline{\text{TIRQ}}$) are provided to notify the CPU of an interrupt condition. The $\overline{\text{IRQ}}$ output is the logical NOR of all (up to eight) unmasked interrupt status bits in the interrupt status register (DUISR). The $\overline{\text{TIRQ}}$ output is the inverted counter/timer ready interrupt status. $\overline{\text{TIRQ}}$ can be masked by the IENT bit of the interrupt control register external to the serial module.

The interrupt level of the serial module $\overline{\text{IRQ}}$ is programmed in the system register external to the serial module. When an interrupt at this level is acknowledged, the serial module is serviced before the external IRQ7 of the same level.

The $\overline{\text{TIRQ}}$ interrupt (if enabled) is fixed at level seven. When a level seven interrupt is acknowledged, the $\overline{\text{TIRQ}}$ interrupt is serviced before the external IRQ7. If the serial module $\overline{\text{IRQ}}$ is also programmed at level seven, the $\overline{\text{TIRQ}}$ interrupt is serviced first, then the serial module $\overline{\text{IRQ}}$, then the external IRQ7 last.

If the $\overline{\text{TIRQ}}$ interrupt is enabled, the serial module counter/timer ready interrupt in the DUISR should be masked, and the $\overline{\text{IRQ}}$ service routine should not service the counter/timer ready condition.

6.1.5 Comparison of Serial Module to MC68681

The serial module is code compatible with the MC68681 with some modifications, but OP2, OP4–7, and IP3–5 are not pinned out. A new interrrupt output ($\overline{\text{TIRQ}}$) is available.

6.2 SERIAL MODULE SIGNAL DEFINITIONS

The following paragraphs contain a brief description of the serial module signals. Figure 6-2 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.

6.2.1 Crystal Input or External Clock (X1/CLK)

This input is one of two connections to a crystal or a single connection to an external clock. A crystal or an external clock signal, at 3.6864 MHz, must be supplied when using the baud rate generator. If a crystal is used, a capacitor of approximately 10 pF should be connected from this signal to ground. If this input is not used, it must be connected to V_{CC} or GND.

6.2.2 Crystal Output (X2)

This output is the additional connection to a crystal. If a crystal is used, a capacitor of approximately 5 pF should be connected from this signal to ground. If an external CMOS-level clock is used on X1/CLK, the X2 output must be left open.

6.2.3 Channel A Transmitter Serial Data Output (TxDA)

This signal is the transmitter serial data output for channel A. The output 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.

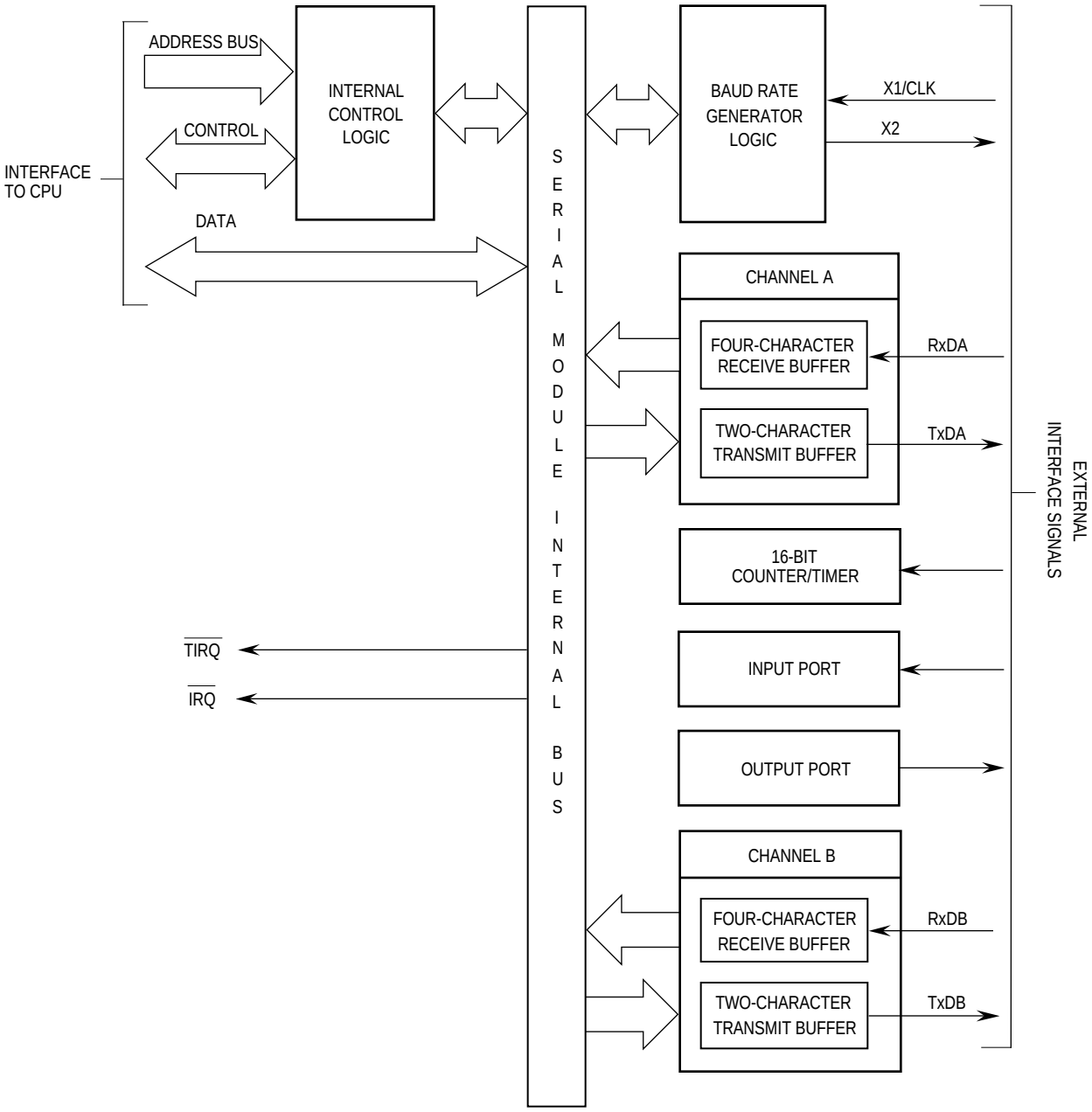


Figure 6-2. External and Internal Interface Signals

6.2.4 Channel A Receiver Serial Data Input (RxDA)

This signal is the receiver serial data input for channel A. Data received on this signal is sampled on the rising edge of the clock source, with the least significant bit received first.

6.2.5 Channel B Transmitter Serial Data Output (TxDB)

This signal is the transmitter serial data output for channel B. The output 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 at the falling edge of the clock source, with the least significant bit transmitted first.

6.2.6 Channel B Receiver Serial Data Input (RxDB)

This signal is the receiver serial data input for channel B. Data on this signal is sampled on the rising edge of the clock source, with the least significant bit received first.

6.2.7 Channel A Request-To-Send ($\overline{\text{RTSA}}$/$\overline{\text{OP0}}$)

This active-low output signal is programmable as the channel A request-to-send or as a dedicated parallel output.

6.2.7.1 $\overline{\text{RTSA}}$. When used for this function, this signal can be programmed to be automatically negated and asserted by either the receiver or transmitter. When connected to the clear-to-send ($\overline{\text{CTSx}}$) input of a transmitter, this signal can be used to control serial data flow.

6.2.7.2 OP0. When used for this function, this output is controlled by bit 0 in the output port data register (DUOP).

6.2.8 Channel B Request-To-Send ($\overline{\text{RTSB}}$/$\overline{\text{OP1}}$)

This active-low output signal is programmable as the channel B request-to-send or as a dedicated parallel output.

6.2.8.1 $\overline{\text{RTSB}}$. When used for this function, this signal can be programmed to be automatically negated and asserted by either the receiver or transmitter. When connected to the $\overline{\text{CTSx}}$ input of a transmitter, this signal can be used to control serial data flow.

6.2.8.2 OP1. When used for this function, this output is controlled by bit 1 in the DUOP register.

6.2.9 Channel A Clear-To-Send ($\overline{\text{CTSA}}$/IP0)

This active-low input is programmable as the channel A clear-to-send or as a dedicated parallel input. It can generate an interrupt on change-of-state.

6.2.9.1 $\overline{\text{CTSA}}$. When used for this function, this signal is the channel A clear-to-send input.

6.2.9.2 IP0. When used for this function, this signal is a general-purpose input.

6.2.10 Channel B Clear-To-Send ($\overline{\text{CTSB}}$/ IP1)

This active-low input is programmable as the channel B clear-to-send or as a dedicated parallel input. It can generate an interrupt on change-of-state.

6.2.10.1 $\overline{\text{CTSB}}$. When used for this function, this signal is the channel B clear-to-send input.

6.2.10.2 IP1. When used for this function, this signal is a general-purpose input.

6.3 OPERATION

The following paragraphs describe the operation of the baud rate generator, transmitter and receiver, and other functional operating modes of the serial module.

6.3.1 Baud Rate Generator

The baud rate generator consists of a crystal oscillator, baud rate generator, and clock selectors (see Figure 6-3). The crystal oscillator operates directly from a 3.6864-MHz crystal or from an external clock of the same frequency. Baud rates are selected by programming the clock-select register (DUCSR) for each channel.

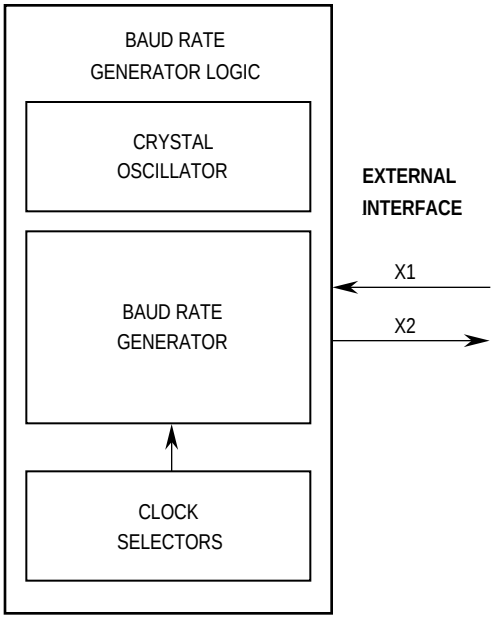
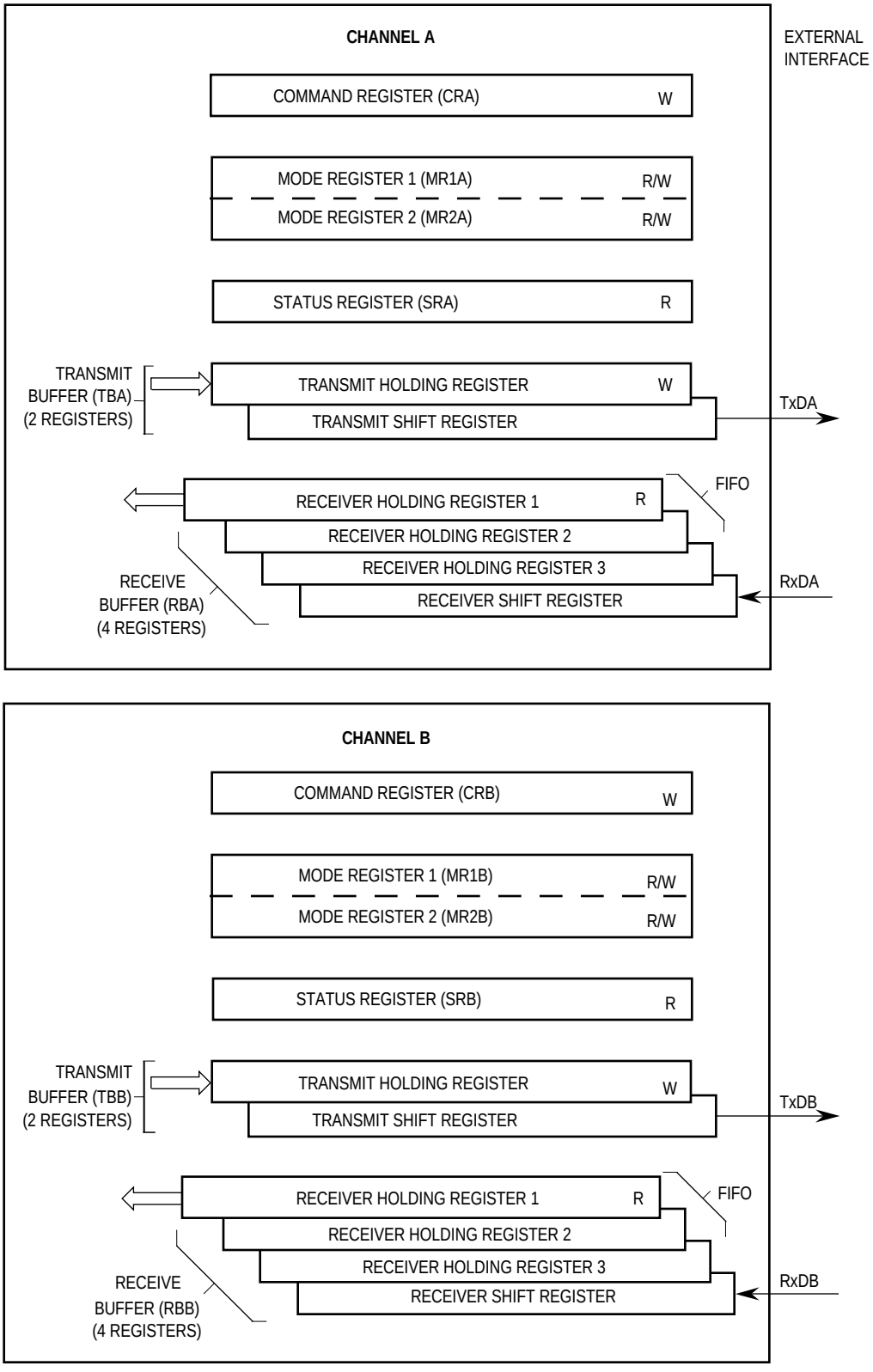


Figure 6-3. Baud Rate Generator Block Diagram

6.3.2 Transmitter and Receiver Operating Modes

The functional block diagram of the transmitter and receiver, including command and operating registers, is shown in Figure 6-4. The paragraphs that follow contain descriptions for both these functions in reference to this diagram. For detailed register information, refer to **6.4 Register Description and Programming**.

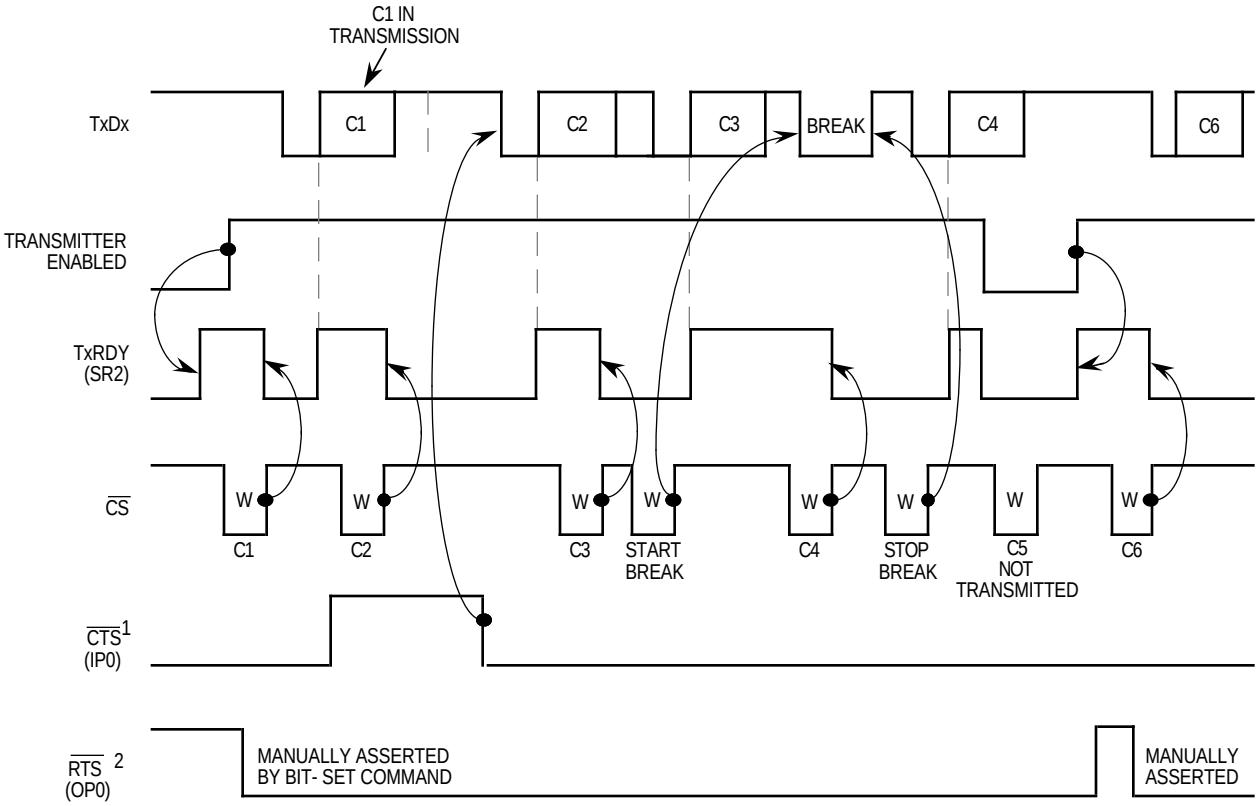


NOTE:
R/W = READ/WRITE
R = READ
W = WRITE

Figure 6-4. Transmitter and Receiver Functional Diagram

6.3.2.1 TRANSMITTER. The transmitters are enabled through their respective command registers (DUCR) located within the serial module. The serial module signals the CPU when it is ready to accept a character by setting the transmitter-ready bit (TxRDY) in the channel's status register (DUSR). Functional timing information for the transmitter is shown in Figure 6-5.

The transmitter converts parallel data from the CPU to a serial bit stream on TxDx. 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.



NOTES:
1. TIMING SHOWN FOR MR2(4) = 1
2. TIMING SHOWN FOR MR2(5) = 1
3. C_N = TRANSMIT CHARACTER
4. W = WRITE

Figure 6-5. Transmitter Timing Diagram

Following transmission of the stop bits, if a new character is not available in the transmitter holding register, the TxDx output remains high ('mark' condition), and the transmitter empty bit (TxEMP) in the DUSR is set. Transmission resumes and the TxEMP bit is cleared when the CPU loads a new character into the transmitter buffer (DUTB). 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 **6.4.1.5 Command Register (DUCR)**). The transmitter is re-enabled through the DUCR to resume operation after a disable or software reset.

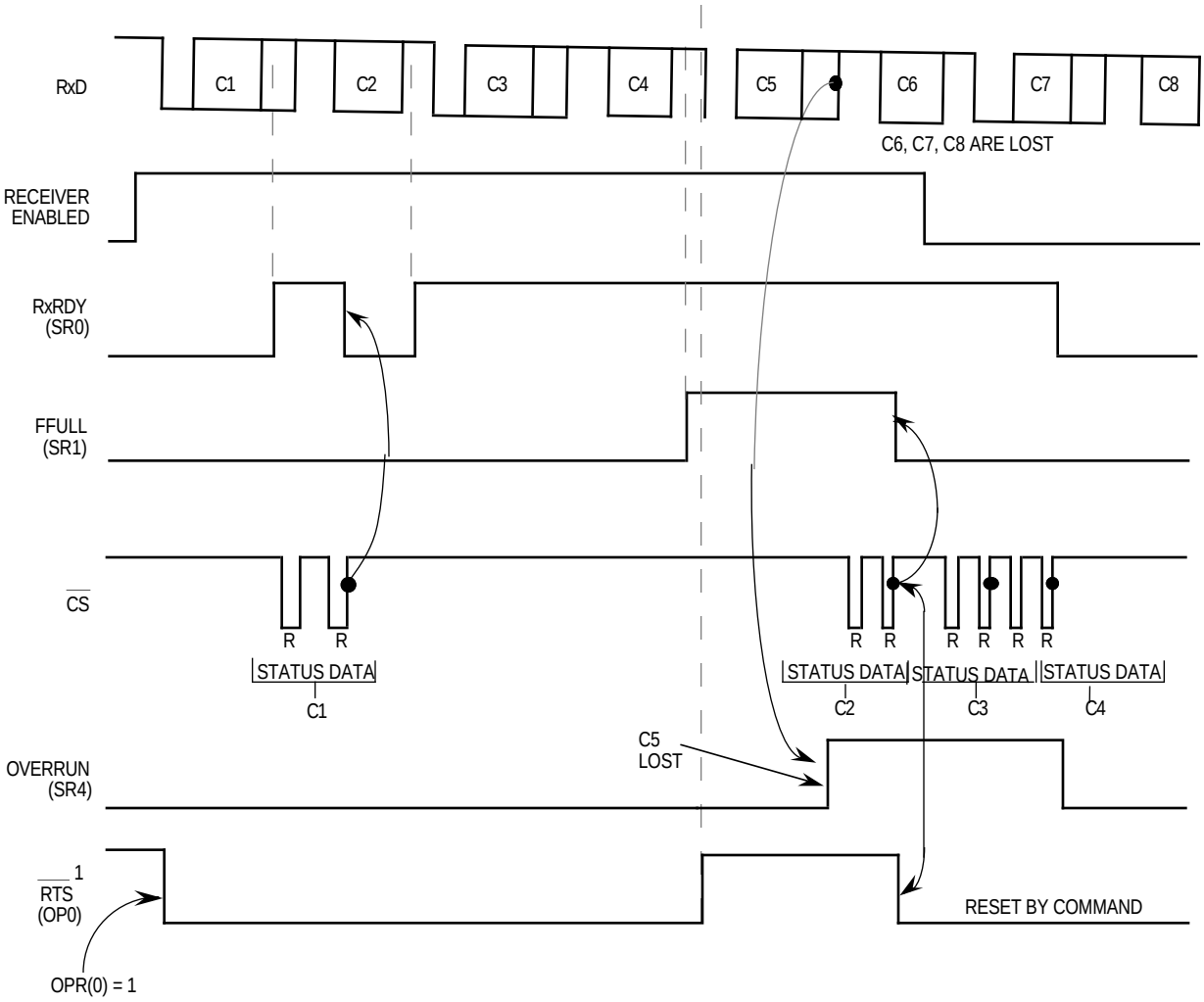
If clear-to-send operation is enabled, $\overline{\text{CTSx}}$ (IP0 for channel A, IP1 for channel B) must be asserted for the character to be transmitted. If $\overline{\text{CTSx}}$ is negated in the middle of a transmission, the character in the shift register is transmitted, and TxDx remains in the 'mark' state until $\overline{\text{CTSx}}$ is asserted again. If the transmitter is forced to send a continuous low condition by issuing a send break command, the state of $\overline{\text{CTSx}}$ is ignored by the transmitter.

The transmitter can be programmed to automatically negate request-to-send ($\overline{\text{RTSx}}$: OP0 for channel A, OP1 for channel B) outputs upon completion of a message transmission. If the transmitter is programmed to operate in this mode, $\overline{\text{RTSx}}$ must be manually asserted before a message is transmitted. In applications in which the transmitter is disabled after transmission is complete and $\overline{\text{RTSx}}$ is appropriately programmed, $\overline{\text{RTSx}}$ is negated one bit time after the character in the shift register is completely transmitted. The transmitter must be manually re-enabled by reasserting $\overline{\text{RTSx}}$ before the next message is to be sent.

6.3.2.2 RECEIVER. The receivers are enabled through their respective DUCRs located within the serial module. Functional timing information for the receiver is shown in Figure 6-6. The receiver looks for a high-to-low (mark-to-space) transition of the start bit on RxDx. When a transition is detected, the state of RxDx 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 RxDx is sampled high, the start bit is invalid, and the search for the valid start bit begins again. If RxDx 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 RxDx 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 appropriate DUSR 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 nonzero character is received without a stop bit (framing error) and RxDx 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 appropriate DUSR at the received character boundary and are valid only when the RxRDY bit in the DUSR is set.

If a break condition is detected (RxDx is low for the entire character including the stop bit), a character of all zeros is loaded into the receiver holding register, and the RB and RxRDY bits in the DUSR are set. The RxDx signal must return to a high condition for at least one-half bit time before a search for the next start bit begins.



NOTES:
1. Timing shown for MR1(7) = 1
2. Timing shown for OPCR(4) = 1 and MR1(6) = 0
3. R = Read
4. C_N = Received Character

Figure 6-6. Receiver Timing Diagram

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 DUSR. 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 DUSR.

6.3.2.3 FIFO STACK. The FIFO stack is used in each channel'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 RxDx (refer to Figure 6-4). 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, FE, and RB, are appended to each data character in the FIFO; OE is not appended. By programming the ERR bit in the channel's mode register (DUMR1), status is provided in character or block modes.

The RxRDY bit in the DUSR 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 DUSR 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 DUSR 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 DUSR 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 DUSR does not affect the FIFO. The FIFO is 'popped' only when the receive buffer is read. The DUSR 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 bit in the DUSR is set when the receiver detects the start bit of the new overrunning character.

To support control flow capability, the receiver can be programmed to automatically negate and assert $\overline{\text{RTSx}}$. When in this mode, $\overline{\text{RTSx}}$ 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, $\overline{\text{RTSx}}$ is asserted by the receiver. Using this mode of operation, overrun errors are prevented by connecting the $\overline{\text{RTSx}}$ to the $\overline{\text{CTSx}}$ input of the transmitting device.

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.

6.3.3 Looping Modes

Each serial module channel can be configured to operate in various looping modes as shown in Figure 6-7. These modes are useful for local and remote system diagnostic functions. The modes are described in the following paragraphs with further information available in **6.4 Register Description and Programming**.

The channel'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.

6.3.3.1 AUTOMATIC ECHO MODE. In this mode, the channel 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 TxDx. The receiver must be enabled, but the transmitter need not be enabled.

Since the transmitter is not active, the DUSR TxEMP and TxRDY bits 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.

6.3.3.2 LOCAL LOOPBACK MODE. In this mode, TxDx is internally connected to RxDx. This mode is useful for testing the operation of a local serial 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 RxDx input data is ignored, the TxDx is held marking, and the receiver is clocked by the transmitter clock. The transmitter must be enabled, but the receiver need not be enabled.

6.3.3.3 REMOTE LOOPBACK MODE. In this mode, the channel automatically transmits received data on the TxDx 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.

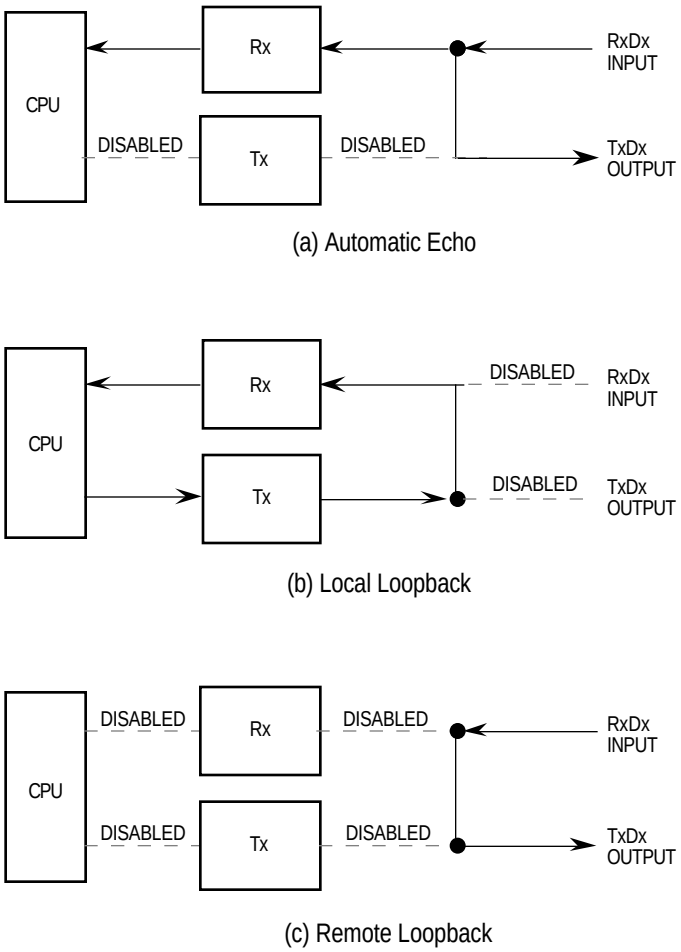


Figure 6-7. Looping Modes Functional Diagram

6.3.4 Multidrop Mode

A channel 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 6-8. The mode is selected by setting bits 3 and 4 in mode register 1 (DUMR1). 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 DUSR 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.

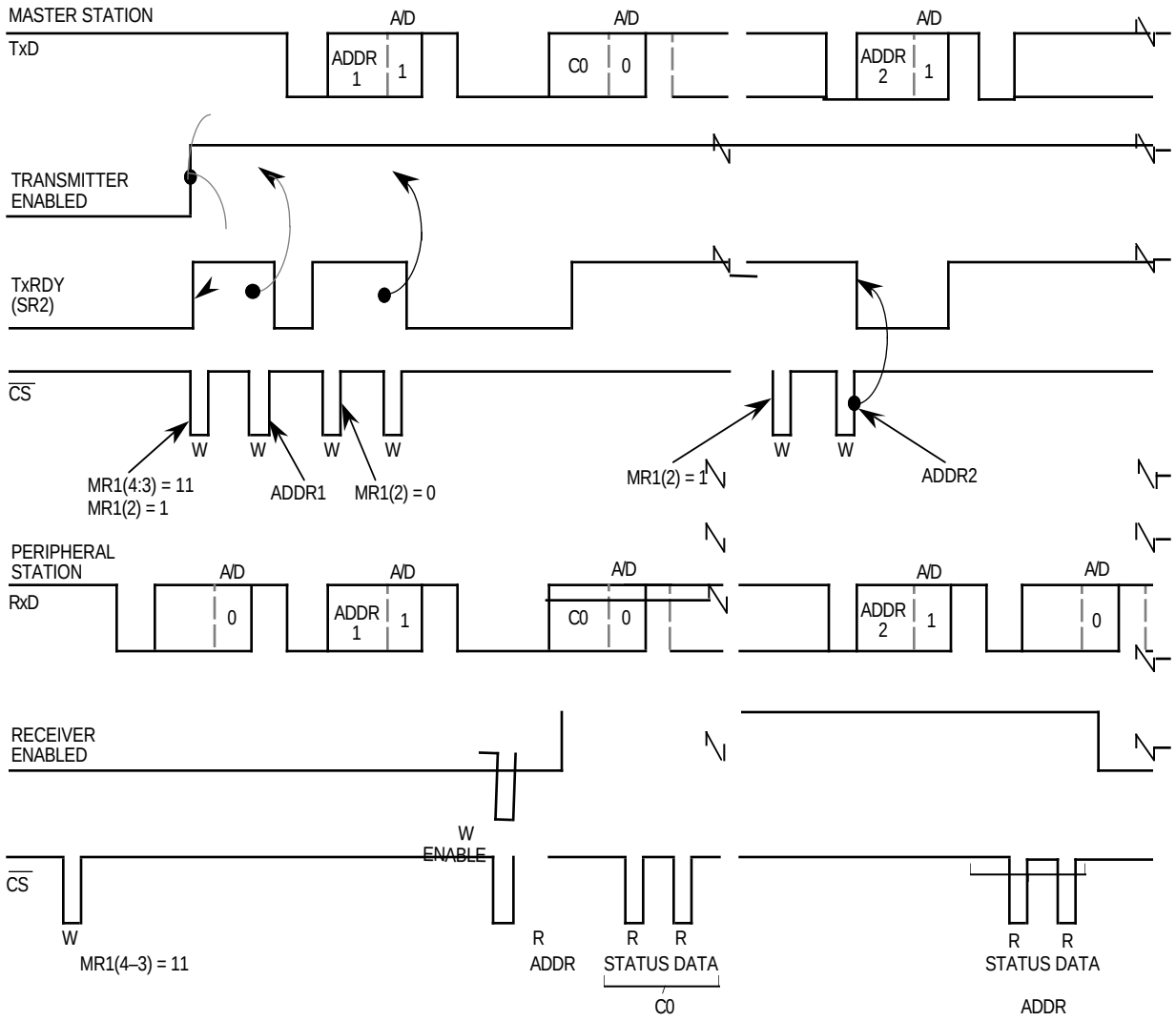


Figure 6-8. Multidrop Mode Timing Diagram

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 the DUMR1. The DUMR1 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 (DUSR 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.

6.3.5 Counter/Timer

The 16-bit counter/timer can operate in a counter mode or a timer mode. In either mode, the counter/timer clock source can be programmed to come from several sources and the counter/timer output can be programmed to appear on output port pin OP3 (inverted). The preload value stored in the concatenation of the counter/timer upper register (DUCTUR) and the counter/timer lower register (DUCTLR) can be from $0002 through $FFFF and this value can be changed at any time. In the counter mode, the counter/timer can be started and stopped by the CPU. Thus, this mode allows the counter/timer to be used as a system stopwatch, a real-time single interrupt generator, or a device watchdog. In the timer mode, the counter/timer runs continuously and cannot be started or stopped by the CPU. Instead, the CPU only resets the counter/timer. Thus, this mode allows the counter/timer to be used as a programmable clock source for channels A and B, periodic interrupt generator, or a variable duty cycle square-wave generator. Upon power-up and after reset, the counter/timer operates in counter mode.

6.3.5.1 COUNTER MODE. In the counter mode, the counter/timer counts down from the preload value using the programmed counter clock source. The counter clock source can be the X1/CLK pin, the channel A transmitter clock, the channel B transmitter clock, or an external clock on the input port pin IP2. The CPU can start and stop the counter and can read the count value (DUCUR:DUCLR). When a read at the start counter command address is performed, the counter initializes itself with the preload value and begins a countdown sequence. Upon reaching $0000 (terminal count), the counter sets the counter/timer-output and the counter/timer ready bit in the interrupt status register (DUISR[3]), rolls over from $0000 to $FFFF, and continues counting. The counter can be programmed to generate an interrupt request for this condition on the $\overline{\text{IRQ}}$ or $\overline{\text{TIRQ}}$ output. If the preload value is changed by the CPU, the counter will not recognize the new value until it receives the next start counter command (and must reinitialize itself). When a read at the stop counter command address is performed, the counter stops the countdown sequence and clears the C/T output and DUISR[3]. The count value should only be read while the counter is stopped. This is because only one of the count registers (either DUCUR or DUCLR) can be read at a time and if the counter is running, a decrement of DUCLR that requires a borrow from the DUCUR could take place between the two reads.

6.3.5.2 TIMER MODE. In the timer mode, the counter/timer generates a square-wave output derived from the programmed timer input (clock source). The timer clock source is X1/CLK or an external input on input port pin IP2, divided by one or sixteen. The square wave generated by the timer has a period of twice the preload value times the period of the clock source, is available as a clock source for both communications channels, and

can be programmed to appear on output pin OP3 (inverted). The timer runs continuously and cannot be stopped by the CPU. Because the timer cannot be stopped, the count value (DUCUR/DUCLR) should not be read. When a read at the start counter command address is performed, the timer terminates the current countdown sequence, clears its output, reinitializes itself with the preload value, and begins a new countdown sequence. Upon reaching $0000 (terminal count), the timer inverts its output, reinitializes itself with the preload value, and repeats the countdown sequence. If the timer output toggled from one to zero, the CTR/TMR_RDY bit (DUISR[3]) is also set. The timer can be programmed to generate an interrupt request for this condition on the $\overline{\text{IRQ}}$ or $\overline{\text{TIRQ}}$ output. If the preload value is changed by the CPU, the timer will not recognize the new value until it reaches the next terminal count (and must reinitialize itself). This feature is very useful when generating variable duty cycle square waves. When a read at the stop counter command address is performed, the timer clears DUISR bit 3 but does not stop. Because in timer mode the counter/timer runs continuously, it should be completely configured (preload value loaded and start counter command issued) before programming the timer output to appear on OP3.

6.3.6 Bus Operation

This section describes the operation of the bus during read, write, and interrupt acknowledge cycles to the serial module. All serial module registers must be accessed as bytes.

6.3.6.1 READ CYCLES. The serial module is accessed by the CPU with a variable number of wait states, depending on the relative phase of the CPU and serial module clocks. The maximum number of wait states for a 16.67 MHz CPU clock and 3.6864 MHz serial module clock is six. The serial module responds to reads with byte data on D7–D0. Reserved registers return logic zero during reads.

6.3.6.2 WRITE CYCLES. The serial module is accessed by the CPU with a variable number of wait states, up to six. The serial 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.

6.3.6.3 INTERRUPT ACKNOWLEDGE CYCLES. The serial 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 (DUIVR) must be initialized. If the DUIVR is not initialized, a spurious interrupt exception will be taken if interrupts are generated.

6.4 REGISTER DESCRIPTION AND PROGRAMMING

This section contains a detailed description of each register and its specific function as well as flowcharts of basic serial module programming.

6.4.1 Register Description

The operation of the serial module is controlled by writing control bytes into the appropriate registers. A list of serial module registers and their associated addresses is

shown in Figure 6-9. The mode, status, command, and clock-select registers are duplicated for each channel to provide independent operation and control.

NOTE

All serial module registers are only accessible as bytes. The contents of the mode registers (DUMR1 and DUMR2), clock-select register (DUCSR), and the auxiliary control register (DUACR) 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.

Address	Register Read (R/W = 1)	Register Write (R/W = 0)
FFFFF7E1	MODE REGISTER A (DUMR1A, DUMR2A)	MODE REGISTER A (DUMR1A, DUMR2A)
FFFFF7E3	STATUS REGISTER A (DUSRA)	CLOCK-SELECT REGISTER A (DUCSRA)
FFFFF7E5	DO NOT ACCESS[1]	COMMAND REGISTER A (DUCRA)
FFFFF7E7	RECEIVER BUFFER A (DURBA)	TRANSMITTER BUFFER A (DUTBA)
FFFFF7E9	INPUT PORT CHANGE REGISTER (DUIPCR)	AUXILIARY CONTROL REGISTER (DUACR)
FFFFF7EB	INTERRUPT STATUS REGISTER (DUISR)	INTERRUPT MASK REGISTER (DUIMR)
FFFFF7ED	COUNTER MODE:CURRENT MSB OF COUNTER	COUNTER/TIMER UPPER REGISTER
FFFFF7EF	COUNTER MODE:CURRENT LSB OF COUNTER	COUNTER/TIMER LOWER REGISTER
FFFFF7F1	MODE REGISTER B (DUMR1B, DUMR2B)	MODE REGISTER B (DUMR1B, DUMR2B)
FFFFF7F3	STATUS REGISTER B (DUSRB)	CLOCK-SELECT REGISTER B (DUCSRB)
FFFFF7F5	DO NOT ACCESS[1]	COMMAND REGISTER B (DUCRB)
FFFFF7F7	RECEIVER BUFFER B (DURBB)	TRANSMITTER BUFFER B (DUTBB)
FFFFF7F9	INTERRUPT VECTOR REGISTER (DUIVR)	INTERRUPT VECTOR REGISTER (DUIVR)
FFFFF7FB	INPUT PORT REGISTER (DUIP)	OUTPUT PORT CONFIGURATION REGISTER (DUOPCR)
FFFFF7FD	START COUNTER COMMAND[2]	OUTPUT PORT (DUOP)[2] BIT SET
FFFFF7FF	STOP COUNTER COMMAND[2]	OUTPUT PORT (DUOP)[2] BIT RESET

NOTES:
1. This address is used for factory testing and should not be read. Reading this location wiill result in undesired effects and possible incorrect transmission or reception of characters. Register contents may also be changed.
2. Address-triggered commands

Figure 6-9. Serial Module Programming Model

6.4.1.1 MODE REGISTER 1 (DUMR1). DUMR1 controls some of the serial module configuration. This register can be read or written at any time. It is accessed when the

channel A mode register pointer points to DUMR1. The pointer is set to DUMR1 by $\overline{\text{RESET}}$ or by a set pointer command, using control register A. After reading or writing DUMR1A, the pointer points to DUMR2A.

DUMR1A, DUMR1B

7	6	5	4	3	2	1	0
RxRTS	RxIRQ	ERR	PM1	PM0	PT	B/C1	B/C0
RESET:							
0	0	0	0	0	0	0	0

Read/Write

RxRTS—Receiver Request-to-Send Control

1 = Upon receipt of a valid start bit, $\overline{\text{RTSx}}$ is negated if the channel's FIFO is full. $\overline{\text{RTSx}}$ is reasserted when the FIFO has an empty position available.
0 = The receiver has no effect on $\overline{\text{RTSx}}$.

This feature can be used for flow control to prevent overrun in the receiver by using the $\overline{\text{RTSx}}$ output to control the $\overline{\text{CTSx}}$ input of the transmitting device. If both the receiver and transmitter are programmed for $\overline{\text{RTS}}$ control, $\overline{\text{RTS}}$ control will be disabled for both since this configuration is incorrect. See **6.4.1.17 Mode Register 2** for information on programming the transmitter $\overline{\text{RTSx}}$ 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 DUSR for the channel.

1 = Block mode—The values in the channel DUSR 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 **6.4.1.5 Command Register (DUCR)** for more information on serial module commands.
0 = Character mode—The values in the channel DUSR 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 6-1). 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 address character transmission. Table 6-1 lists the parity mode and type or the multidrop mode for each combination of the parity mode and the parity type bits.

Table 6-1. 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 6-2 does not include start, parity, or stop bits.

Table 6-2. 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

6.4.1.2 MODE REGISTER 2 (DUMR2). DUMR2 controls some of the serial module configuration. It is accessed when the channel A mode register pointer points to DUMR2, which ocurs after any access to DUMR1. Accesses to DUMR2 do not change the pointer.

DUMR2A, DUMR2B

7	6	5	4	3	2	1	0
CM1	CM0	TxRTS	TxCTS	SB3	SB2	SB1	SB0
RESET:							
0	0	0	0	0	0	0	0

Read/Write

CM1–CM0—Channel Mode

These bits select a channel mode as listed in Table 6-3. See **6.3.3 Looping Modes** for more information on the individual modes.

Table 6-3. 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 $\overline{\text{RTSA}}$ or $\overline{\text{RTSB}}$ signals. The output is normally asserted by setting OP0 or OP1 and negated by clearing OP0 or OP1 (see **6.4.1.18 Output Port Control Register (DUOPCR)**).

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 $\overline{\text{RTSx}}$.

TxCTS—Transmitter Clear-to-Send

1 = Enables clear-to-send operation. The transmitter checks the state of the $\overline{\text{CTSx}}$ input each time it is ready to send a character. If $\overline{\text{CTSx}}$ is asserted, the character is transmitted. If $\overline{\text{CTSx}}$ is negated, the channel TxDx remains in the high state, and the transmission is delayed until $\overline{\text{CTSx}}$ is asserted. Changes in $\overline{\text{CTSx}}$ 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 $\overline{\text{CTSx}}$ 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 6-4. 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, DUMR2 bit 3 = 0 selects one stop bit, and DUMR2 bit 3 = 1 selects two stop bits for transmission.

Table 6-4. 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

6.4.1.3 STATUS REGISTER (DUSR). The DUSR indicates the status of the characters in the FIFO and the status of the channel transmitter and receiver.

DUSRA, DUSRB

7	6	5	4	3	2	1	0
RB	FE	PE	OE	TxEMP	TxRDY	FFULL	RxRDY
RESET:							
0	0	0	0	0	0	0	0

Read Only

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 the channel RxDx 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 DUMR1, 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 DUCR.

0 = No overrun has occurred.

TxEMP—Transmitter Empty

1 = The channel 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 DUCR.

TxRDY—Transmitter Ready

This bit is duplicated in the DUISR; bit 0 for channel A and bit 4 for channel B.

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 in channel B and is waiting in the receiver buffer FIFO.

0 = The FIFO is not full, but may contain up to two unread characters.

RxRDY—Receiver Ready

1 = One or more characters has been received in channel B 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.

6.4.1.4 CLOCK-SELECT REGISTER (DUCSR). The DUCSR selects the baud rate clock for the channel receiver and transmitter.

DUCSRA, DUCSRB

7	6	5	4	3	2	1	0
RCS3	RCS2	RCS1	RCS0	TCS3	TCS2	TCS1	TCS0
RESET:							
0	0	0	0	0	0	0	0

Write Only

RCS3–RCS0—Receiver Clock Select

These bits select the baud rate clock for the channel receiver from a set of baud rates listed in Table 6-5. The baud rate set selected depends upon the auxiliary control register (DUACR) bit 7. Set 1 is selected if DUACR bit 7 = 0, and set 2 is selected if DUACR bit 7 = 1. The receiver clock is always 16 times the baud rate shown in this list, except when the clock select bits = 1111.

Table 6-5. RCSx Control Bits

RCS3	RCS2	RCS1	RCS0	Set 1	Set 2
0	0	0	0	50	75
0	0	0	1	110	110
0	0	1	0	134.5	134.5
0	0	1	1	200	150
0	1	0	0	300	300
0	1	0	1	600	600
0	1	1	0	1200	1200
0	1	1	1	1050	2000
1	0	0	0	2400	2400
1	0	0	1	4800	4800
1	0	1	0	7200	1800
1	0	1	1	9600	9600
1	1	0	0	38.4k	19.2k
1	1	0	1	TIMER	TIMER
1	1	1	0	–	–
1	1	1	1	–	–

TCS3–TCS0—Transmitter Clock Select

These bits select the baud rate clock for the channel transmitter from a set of baud rates listed in Table 6-6 The baud rate set selected depends upon DUACR bit 7. Set 1 is selected if DUACR bit 7 = 0, and set 2 is selected if DUACR bit 7 = 1. The transmitter clock is always 16 times the baud rate shown in this list, except when the clock select bits = 1111.

Table 6-6. TCSx Control Bits

TCS3	TCS2	TCS1	TCS0	Set 1	Set 2
0	0	0	0	50	75
0	0	0	1	110	110
0	0	1	0	134.5	134.5
0	0	1	1	200	150
0	1	0	0	300	300
0	1	0	1	600	600
0	1	1	0	1200	1200
0	1	1	1	1050	2000
1	0	0	0	2400	2400
1	0	0	1	4800	4800
1	0	1	0	7200	1800
1	0	1	1	9600	9600
1	1	0	0	38.4k	19.2k
1	1	0	1	TIMER	TIMER
1	1	1	0	–	–
1	1	1	1	–	–

6.4.1.5 COMMAND REGISTER (DUCR). The DUCR is used to supply commands to the channel. Multiple commands can be specified in a single write to the DUCR if the commands are not conflicting—e.g., reset transmitter and enable transmitter commands cannot be specified in a single command.

DUCRA, DUCRB

7	6	5	4	3	2	1	0
–	MISC2	MISC1	MISC0	TC1	TC0	RC1	RC0
RESET:							
0	0	0	0	0	0	0	0

Write Only

MISC3–MISC0—Miscellaneous Commands

These bits select a single command as listed in Table 6-7.

Table 6-7. MISCx 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 DUMR1.

Reset Receiver—The reset receiver command resets the channel receiver. The receiver is immediately disabled, the FFULL and RxRDY bits in the DUSR 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 channel transmitter. The transmitter is immediately disabled, and the TxEMP and TxRDY bits in the DUSR are cleared. All other registers are unaltered. This command should be used in lieu of the transmitter disable command whenever the transmitter configuration is changed because it places the transmitter in a known state.

Reset Error Status—The reset error status command clears the channel's RB, FE, PE, and OE bits (in the DUSR). 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 DUISR.

Start Break—The start break command forces the channel's TxDx 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 $\overline{\text{CTSx}}$ input is ignored for this command.

Stop Break—The stop break command causes the channel's TxDx 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 6-8.

Table 6-8. TCx 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 DUSR 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 DUSR. 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 6-9.

Table 6-9. 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—The receiver enable command enables operation of the channel's receiver. If the serial module is not in multidrop mode, 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 serial 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.

6.4.1.6 RECEIVER BUFFER (DURB). The receiver buffer contains three receiver holding registers and a serial shift register. The channel's RxDx 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 6-4).

DURBA, DURBB

7	6	5	4	3	2	1	0
RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
RESET:							
0	0	0	0	0	0	0	0

Read Only

RB7–RB0—These bits contain the character in the receiver buffer.

6.4.1.7 TRANSMITTER BUFFER (DUTB). The transmitter buffer consists of two registers, the transmitter holding register and the transmitter shift register (see Figure 6-4). The holding register accepts characters from the bus master if the TxRDY bit in the channel's DUSR 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 channel's DUSR. Writes to the transmitter buffer when the channel's DUSR TxRDY bit is clear and when the transmitter is disabled have no effect on the transmitter buffer.

DUTBA, DUTBB

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

TB7–TB0—These bits contain the character in the transmitter buffer.

6.4.1.8 INPUT PORT CHANGE REGISTER (DUIPCR). The DUIPCR shows the current state and the change-of-state for the IP0, IP1, and IP2 pins.

DUIPCR

7	6	5	4	3	2	1	0
0	COS2	COS1	COS0	1	IP2	IP1	IP0
RESET:							
0	0	0	0	1	IP2	IP1	IP0

Read Only

Bits 7, 6, 3, 2—Reserved

COS2, COS1, COS0—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 DUACR 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 DUIPCR. A read of the DUIPCR also clears the DUISR COS bit.

IP2, IP1, IP0—Current State

Starting two serial clock periods after reset, the IPx bits reflect the state of the IPx pins. If a $\overline{\text{CTS}}$x pin is detected as asserted at that time, the associated COSx bit will be set, which will initiate an interrupt if the corresponding IECx bit of the DUACR register is enabled.

1 = The current state of the respective IPx input is logic one (negated, if used as $\overline{\text{CTS}}$).

0 = The current state of the respective IPx input is logic zero (asserted, if used as $\overline{\text{CTS}}$).

6.4.1.9 AUXILIARY CONTROL REGISTER (DUACR). The DUACR selects which baud rate is used and controls the handshake of the transmitter/receiver.

DUACR

7	6	5	4	3	2	1	0
BRG	CTMS2	CTMS1	CTMS0	–	IEC2	IEC1	IEC0
RESET: 0	0	0	0	0	0	0	0

Write Only

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 **6.4.1.4 Clock-Select Register (DUCSR)** for more information on the baud rates.

CTMS2–0— Counter/Timer Mode and Source Select

Table 6-10 lists the counter/timer mode and source select bit fields.

Table 6-10. Counter/Timer Mode and Source Select Bits

MISC2	MISC1	MISC0	Mode Command	Clock Source Select Command
0	0	0	Counter	External–IP2
0	0	1	Counter	TxCA
0	1	0	Counter	TxCB
0	1	1	Counter	Crystal or External Clock
1	0	0	Timer	External–IP2
1	0	1	Timer	External–IP2 Divided by 16
1	1	0	Timer	Crystal or External Clock
1	1	1	Timer	External Clock Divided by 16

IEC2, IEC1, IEC0—Input Enable Control

1 = DUISR bit 7 will be set and an interrupt will be generated when the corresponding bit in the DUIPCR (COS2, COS1, or COS0) is set by an external transition on the IPx input (if bit 7 of the interrupt mask register (DUIMR) is set to enable interrupts).

0 = Setting the corresponding bit in the DUIPCR has no effect on DUISR bit 7.

6.4.1.10 INTERRUPT STATUS REGISTER (DUISR). The DUISR provides status for all potential interrupt sources. The contents of this register are masked by the DUIMR. If a flag in the DUISR is set and the corresponding bit in DUIMR is also set, the $\overline{IRQ}$ output is asserted. If the corresponding bit in the DUIMR is cleared, the state of the bit in the DUISR has no effect on the output.

NOTE

The IDUMR does not mask reading of the DUISR. True status is provided regardless of the contents of DUIMR. The contents of DUISR are cleared when the serial module is reset.

DUISR

7	6	5	4	3	2	1	0
COS	DBB	RxRDYB	TxRDYB	CTR/TMR_RDY	DBA	RxRDYA	TxRDYA
RESET: 0	0	0	0	1	0	0	0

Read Only

COS—Change-of-State

1 = A change-of-state has occurred at one of the IPx inputs and has been selected to cause an interrupt by programming bit 2, 1 and/or bit 0 of the DUACR.

0 = No selected COSx in the DUIPCR.

DBB—Delta Break B

1 = The channel B receiver has detected the beginning or end of a received break.
0 = No new break-change condition to report. Refer to **6.4.1.5 Command Register (DUCR)** for more information on the reset break-change interrupt command.

RxRDYB—Channel B Receiver Ready or FIFO Full

The function of this bit is programmed by DUMR1B bit 6. It is a duplicate of either the FFULL or RxRDY bit of DUSRB.

TxRDYB—Channel B Transmitter Ready

This bit is the duplication of the TxRDY bit in DUSRB.

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 TxRDYx=0 are not transmitted.

CTR/TMR_RDY—Counter/Timer Ready

1 = Counter/timer ready.
0 = Counter/timer not ready.

DBA—Delta Break A. See DBB.

RxRDYA—Channel A Receiver Ready or FIFO Full. See RxRDYB.

The function of this bit is programmed by DUMR1A bit 6.

TxRDYA—Channel A Transmitter Ready. See TxRDYB.

This bit is the duplication of the TxRDY bit in DUSRA.

6.4.1.11 INTERRUPT MASK REGISTER (DUIMR). The DUIMR selects the corresponding bits in the DUISR that cause an interrupt output ($\overline{IRQ}$). If one of the bits in the DUISR is set and the corresponding bit in the DUIMR is also set, the $\overline{IRQ}$ output is asserted. If the corresponding bit in the DUIMR is zero, the state of the bit in the DUISR has no effect on the $\overline{IRQ}$ output. The DUIMR does not mask the reading of the DUISR.

DUIMR

7	6	5	4	3	2	1	0
COS	DBB	FFULLB	TxRDYB	CTR/TMR_RDY	DBA	FFULLA	TxRDYA
RESET: 0	0	0	0	0	0	0	0

Write Only

COS—Change-of-State

1 = Enable interrupt
0 = Disable interrupt

DBB—Delta Break B

1 = Enable interrupt
0 = Disable interrupt

FFULLB—Channel B FIFO Full

1 = Enable interrupt
0 = Disable interrupt

TxRDYB, TxRDYA—Transmitter Ready

1 = Enable interrupt
0 = Disable interrupt

CTR/TMR_RDY—Counter/Timer Ready

1 = Enable interrupt
0 = Disable interrupt

DBA—Delta Break A

1 = Enable interrupt
0 = Disable interrupt

FFULLA—Channel A FIFO Full

1 = Enable interrupt
0 = Disable interrupt

6.4.1.12 COUNT REGISTER: CURRENT MSB OF COUNTER (DUCUR). This register holds the most-significant byte of the current value in the counter/timer. It should only be read when the counter/timer is in counter mode and the counter is stopped. See **6.3.5 Counter/Timer** for further information.

6.4.1.13 COUNT REGISTER: CURRENT LSB OF COUNTER (DUCLR). This register holds the least-significant byte of the current value in the counter/timer. It should only be read when the counter/timer is in counter mode and the counter is stopped. See **6.3.5 Counter/Timer** for further information.

6.4.1.14 COUNTER/TIMER UPPER PRELOAD REGISTER (DUCTUR). This register holds the eight most-significant bits of the preload value to be used by the conter/timer in either the count or timer mode. The minimum value that can be loaded on the concatenation of DUCTUR with DUCTLR is 0002 (hex). This register is write only and cannot be read by the CPU.

6.4.1.15 COUNTER/TIMER LOWER PRELOAD REGISTER (DUCTLR). This register holds the eight least-significant bits of the preload value to be used by the conter/timer in either the count or timer mode. The minimum value that can be loaded on the concatenation of DUCTUR with DUCTLR is 0002 (hex). This register is write only and cannot be read by the CPU.

6.4.1.16 INTERRUPT VECTOR REGISTER (DUIVR). The DUIVR contains the 8-bit vector number of the $\overline{IRQ}$ interrupt.

DUIVR

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

IVR7–IVR0—Interrupt Vector Bits

Each module that generates interrupts can have an interrupt vector field. 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 DUIVR is reset to $0F, which indicates an uninitialized interrupt condition. See **Section 4 EC000 Core Processor** for more information.

6.4.1.17 INPUT PORT REGISTER. The DUIP register shows the current state of the IPx inputs.

DUIP

7	6	5	4	3	2	1	0
1	1	IP5	IP4	IP3	IP2	IP1	IP0
1	1	1	1	1	IP2	IP1	IP0

Read Only

IP5, IP4, IP3, IP2, IP1, IP0—Current State

1 = The current state of the respective IP input is logic one.
0 = The current state of the respective IP input is logic zero.

The information contained in these bits is latched and reflects the state of the input pins at the time that the DUIP is read.

NOTE

These bits have the same function and value of the DUIPCR bits 1 and 0.

IP5, IP4, and IP3 are not pinned out on the MC68306, and are internally set to logic one.

6.4.1.18 OUTPUT PORT CONTROL REGISTER (DUOPCR). The DUOPCR configures six bits of the 8-bit parallel DUOP for general-purpose use or for auxiliary functions serving the communication channels.

DUOPCR

7	6	5	4	3	2	1	0
OP7 / $\overline{\text{TxRDYB}}$	OP6 / $\overline{\text{TxRDYA}}$	OP5 / $\overline{\text{RxRDYB}}$	OP4 / $\overline{\text{RxRDYA}}$	OP3		OP2	
RESET:							
0	0	0	0	0	0	0	0

Write Only

NOTE

OP bits 7, 6, 5, 4, and 2 are not pinned out on the MC68306; thus changing bits 7, 6, 5, 4, 1, and 0 of this register has no effect.

OPCR3–OPCR2—Output Port 3 Function Select

00 = OPR bit 3
01 = Counter/timer output
10 = TxCB (1X)
11 = RxCB (1X)

NOTE

OP3 is open-drain in this mode, and an external pullup is required.

OPCR1–OPCR0—Output Port 2 Function Select

00 = OPR bit 2
01 = TxCA (16X)
10 = TxCA (1X)
11 = RxCA (1X)

6.4.1.19 OUTPUT PORT DATA REGISTER (DUOP). The bits in the DUOP register are set by performing a bit set command (writing to $FFFFF7FD) and are cleared by performing a bit reset command (writing to offset $FFFFF7FF).

Bit Set

DUOP

7	6	5	4	3	2	1	0
$\overline{OP7}$	$\overline{OP6}$	$\overline{OP5}$	$\overline{OP4}$	$\overline{OP3}$	$\overline{OP2}$	$\overline{OP1}$	$\overline{OP0}$
RESET:							
0	0	0	0	0	0	0	0

Write Only

NOTE

The output port bits are inverted at the pins.

OP bits 7, 6, 5, 4, and 2 are not pinned out on the MC68306; thus, changing these bits has no effect.

$\overline{OP3}$, $\overline{OP1}$, $\overline{OP0}$ —Output Port Parallel Outputs

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.

Bit Reset

DUOP

7	6	5	4	3	2	1	0
$\overline{OP7}$	$\overline{OP6}$	$\overline{OP5}$	$\overline{OP4}$	$\overline{OP3}$	$\overline{OP2}$	$\overline{OP1}$	$\overline{OP0}$
RESET:							
0	0	0	0	0	0	0	0

Write Only

$\overline{OP3}$, $\overline{OP1}$, $\overline{OP0}$ —Output Port Parallel Outputs

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.

6.4.1.20 Start Counter Command Register. A read at this address starts the counter/timer. The read data has no meaning.

6.4.1.21 Stop Counter Command Register. A read at this address stops the counter and clears the counter output (visible on OP3) if counter mode is selected. A read at this address also clears DUISR CTR/TMR_RDY bit in counter or timer mode. The read data has no meaning.

6.4.2 Programming

The basic interface software flowchart required for operation of the serial module is shown in Figure 6-10. The routines are divided into three categories:

- Serial Module Initialization
- I/O Driver
- Interrupt Handling

6.4.2.1 SERIAL MODULE INITIALIZATION. The serial module initialization routines consist of SINIT and CHCHK. SINIT is called at system initialization time to check channel A and channel B 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 channels. If SINIT finds no errors in either channel A or channel B, the respective receivers and transmitters are enabled. The CHCHK routine performs the actual channel checks as called from the SINIT routine. When called, SINIT places the specified channel in the local loopback mode and checks for the following errors:

- Transmitter Never Ready
- Receiver Never Ready
- Parity Error
- Incorrect Character Received

6.4.2.2 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 channel receiver. OUTCH is used to send a character to the channel transmitter.

6.4.2.3 INTERRUPT HANDLING. The interrupt handling routine consists of SIRQ, which is executed after the serial module generates an interrupt caused by a channel 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.

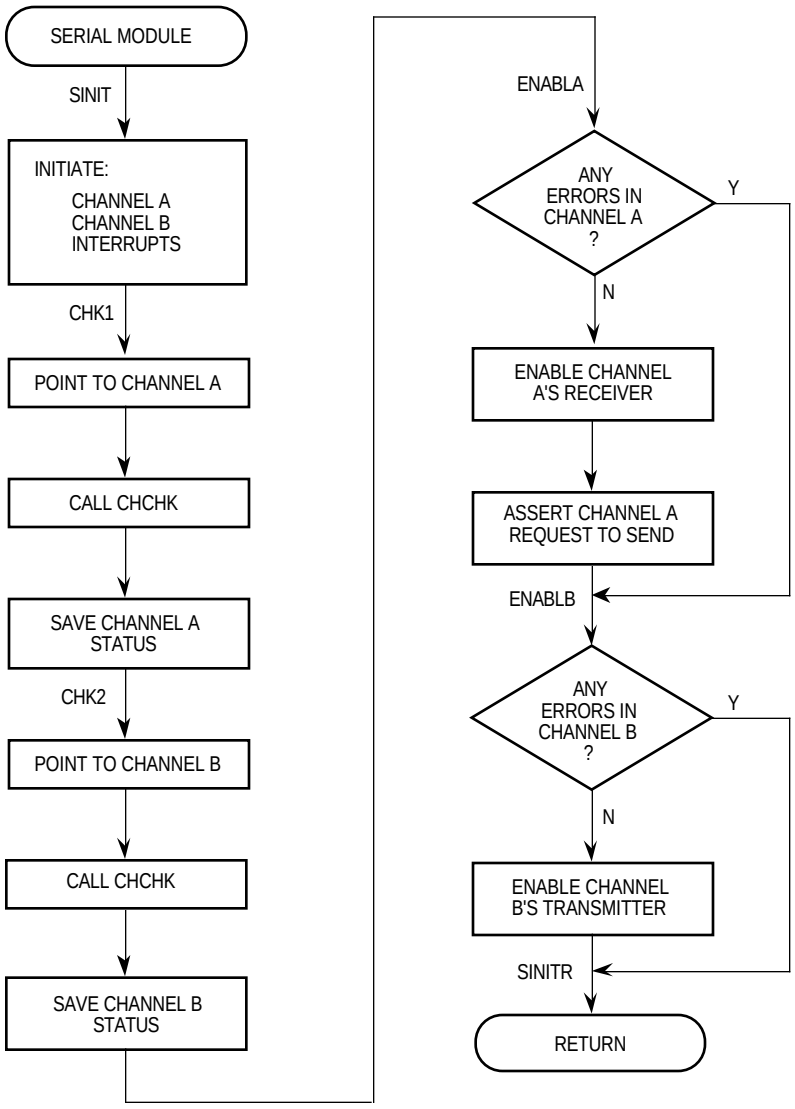


Figure 6-10. Serial Module Programming Flowchart (1 of 5)

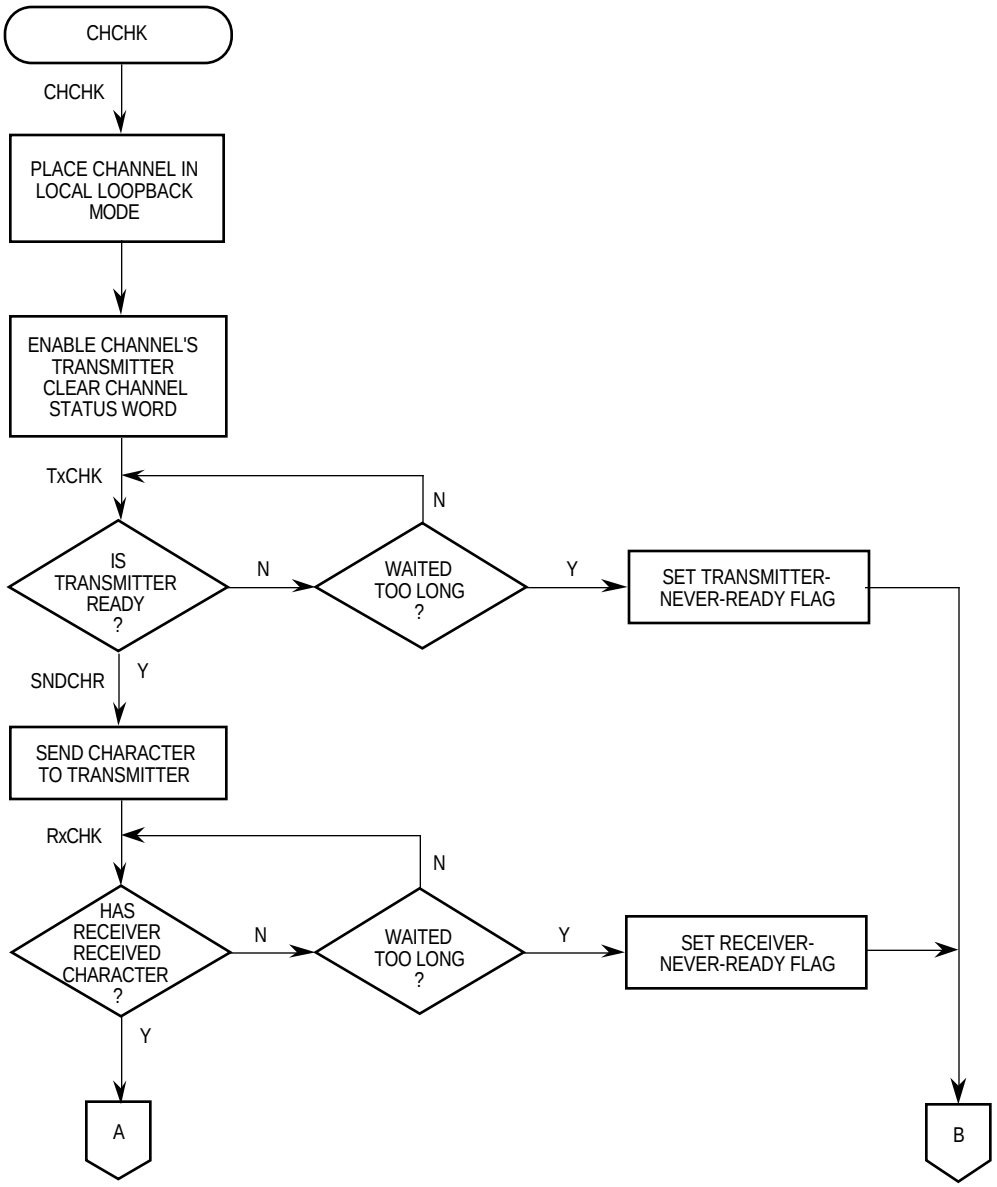


Figure 6-10. Serial Module Programming Flowchart (2 of 5)

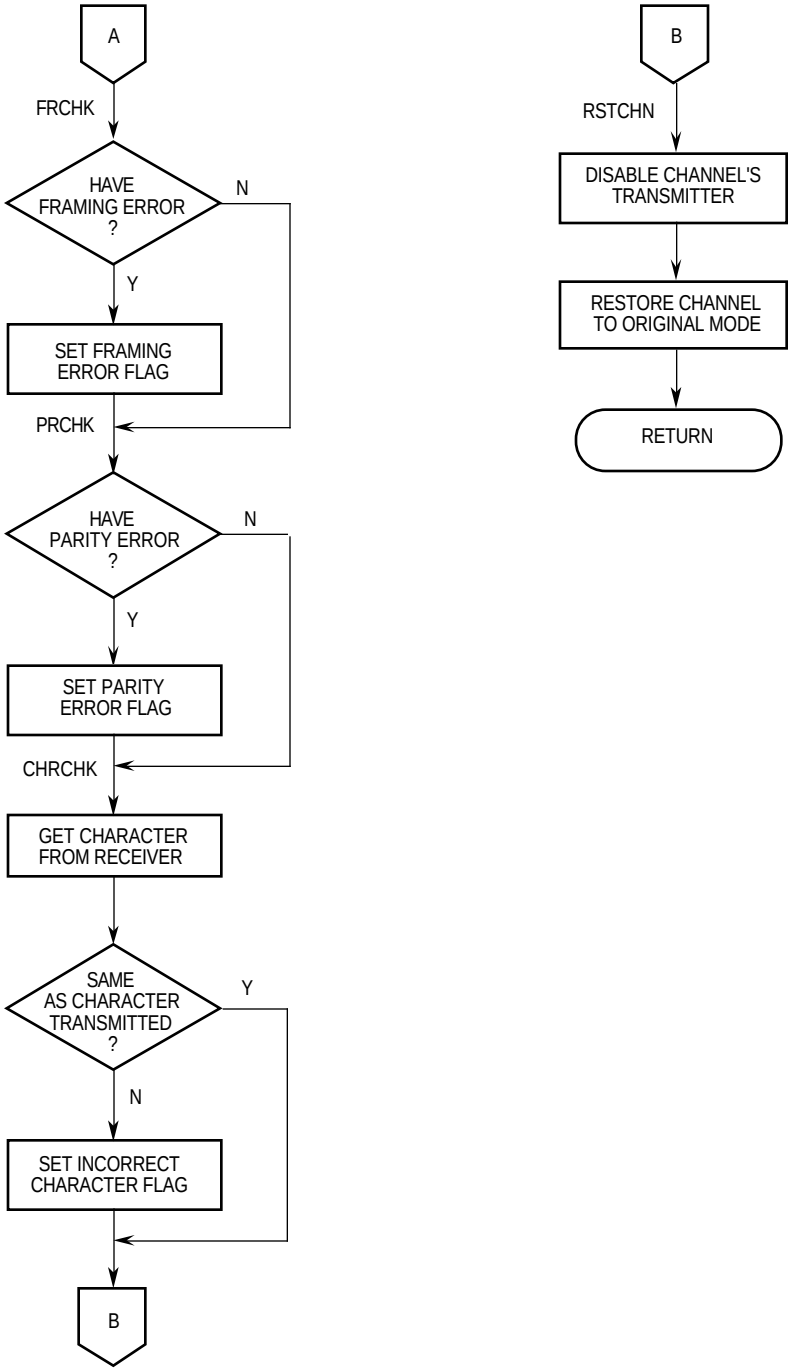


Figure 6-10. Serial Module Programming Flowchart (3 of 5)

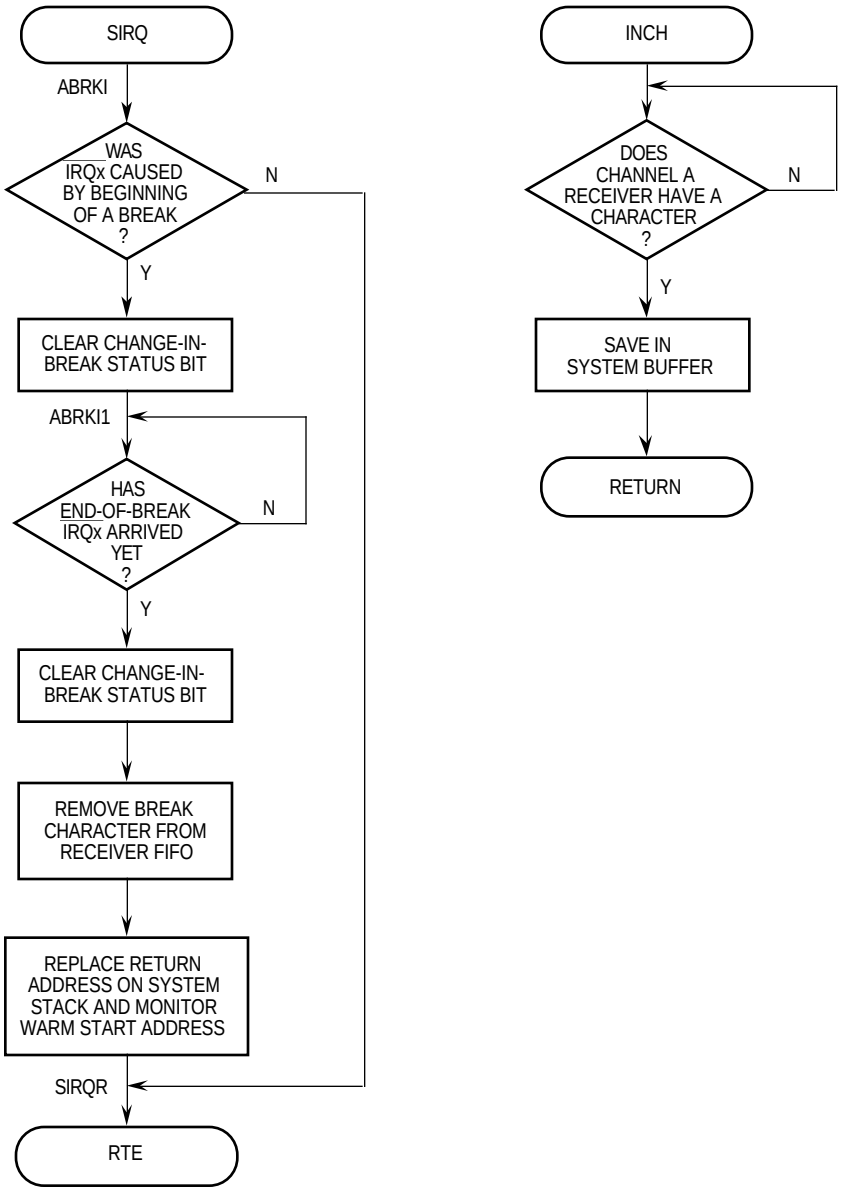


Figure 6-10. Serial Module Programming Flowchart (4 of 5)

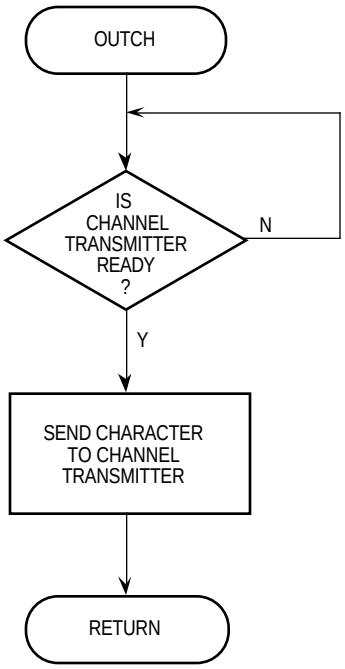


Figure 6-10. Serial Module Programming Flowchart (5 of 5)

6.5 SERIAL MODULE INITIALIZATION SEQUENCE

If the serial capability of the MC68306 is being used, the following steps are required to properly initialize the serial module.

NOTE

The serial module registers can be accessed by word or byte operations, but only the data byte D7–D0 is valid.

Command Register (DUCR)

- Reset the receiver and transmitter for each channel.

The following steps program both channels:

Interrupt Vector Register (DUIVR)

- Program the vector number for a serial module interrupt.

Interrupt Mask Register (DUIMR)

- Enable the desired interrupt sources.

Auxiliary Control Register (DUACR)

- Select baud rate set (BRG bit).
- Initialize the input enable control (IEC bits).
- Select counter/timer mode and clock source if necessary.

Output Port Control Register (DUOPCR)

- Select the function of the output port pins.

The following steps are channel specific:

Clock Select Register (DUCSR)

- Select the receiver and transmitter clock.

Mode Register 1 (DUMR1)

- If desired, 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 (DUMR2)

- Select the mode of channel operation (CMx bits).
- If desired, program operation of transmitter ready-to-send (TxRTS bit).

- If desired, program operation of clear-to-send (TxCTS bit).
- Select stop-bit length (SBx bits).

Command Register (DUCR)

- Enable the receiver and transmitter.

SECTION 7
IEEE 1149.1 TEST ACCESS PORT

The MC68306 includes dedicated user-accessible test logic that is fully compatible with the *IEEE 1149.1 Standard Test Access Port and Boundary Scan Architecture*. Problems associated with testing high-density circuit boards have led to development of this standard under the sponsorship of the Test Technology Committee of IEEE and the Joint Test Action Group (JTAG). The MC68306 implementation supports circuit-board test strategies based on this standard.

The test logic includes a test access port (TAP) consisting of five dedicated signal pins, a 16-state controller, an instruction register, and four test data registers. A boundary scan register links all device signal pins into a single shift register. The test logic, implemented using static logic design, is independent of the device system logic. The MC68306 implementation provides the following capabilities:

a. Perform boundary scan operations to test circuit-board electrical continuity
b. Sample the MC68306 system pins during operation and transparently shift out the result in the boundary scan register
c. Bypass the MC68306 for a given circuit-board test by effectively reducing the boundary scan register to a single bit
d. Disable the output drive to pins during circuit-board testing
e. Drive output pins to stable levels

NOTE

Certain precautions must be observed to ensure that the IEEE 1149.1 test logic does not interfere with non-test operation. See **7.6 Non-IEEE 1149.1 Operation** for details.

7.1 OVERVIEW

NOTE

This description is not intended to be used without the supporting IEEE 1149.1 document.

The discussion includes those items required by the standard and provides additional information specific to the MC68306 implementation. For internal details and applications of the standard, refer to the IEEE 1149.1 document.

An overview of the MC68306 implementation of IEEE 1149.1 is shown in Figure 7-1. The MC68306 implementation includes a 16-state controller, a 3-bit instruction register, and four test registers (a 1-bit bypass register, a 124-bit boundary scan register, a 3-bit module mode register, and a 32-bit ID register). This implementation includes a dedicated TAP consisting of the following signals:

TRST — active low JTAG logic reset (with pullup).
TCK — test clock input to synchronize the test logic (with pulldown).
TMS — test mode select input (with an internal pullup resistor) that is sampled on the rising edge of TCK to sequence the TAP controller's state machine.
TDI — test data input (with an internal pullup resistor) that is sampled on the rising edge of TCK.
TDO — three-state test data output that is actively driven in the shift-IR and shift-DR controller states. TDO changes on the falling edge of TCK.

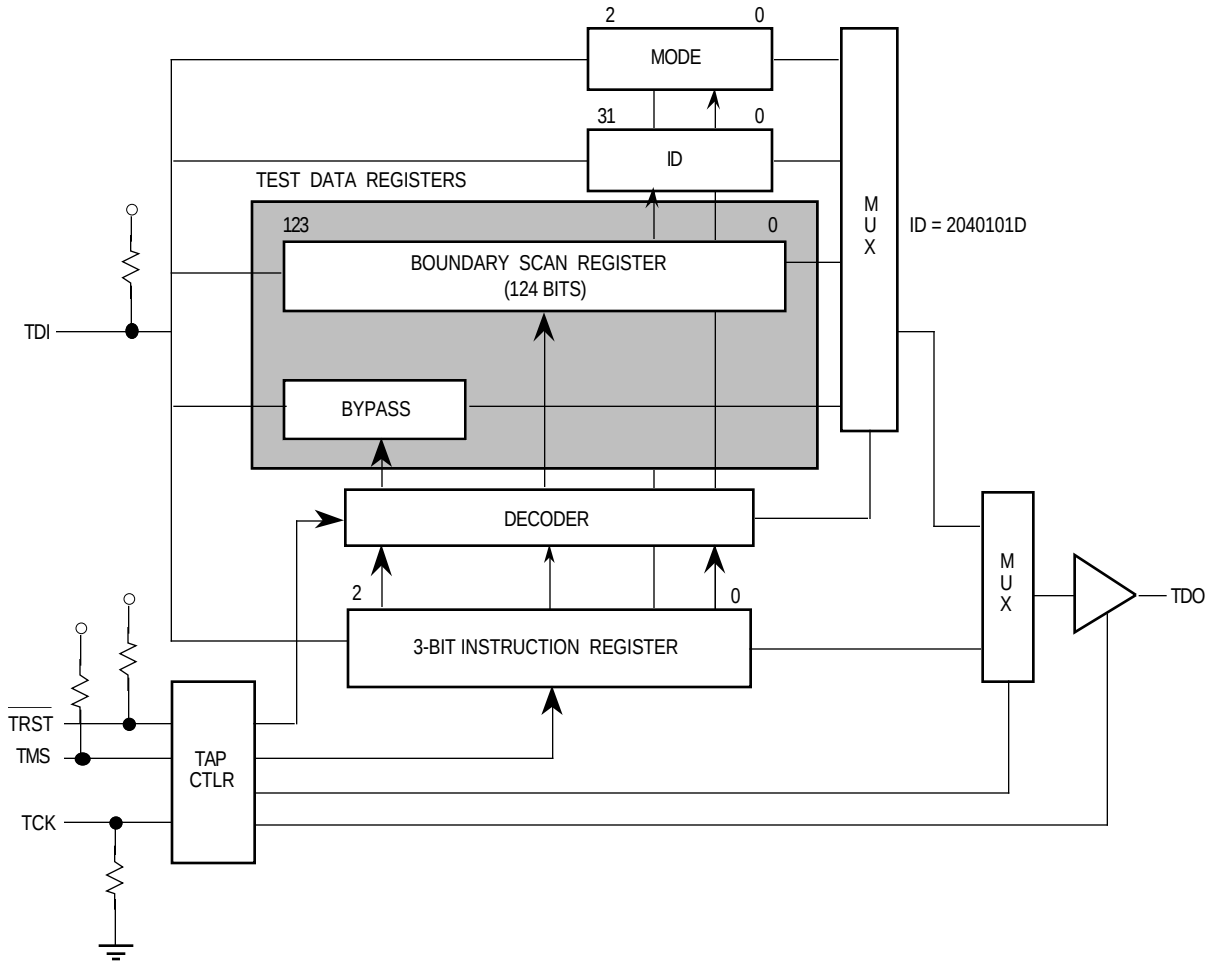


Figure 7-1. Test Access Port Block Diagram

7.2 TAP CONTROLLER

The TAP controller is responsible for interpreting the sequence of logical values on the TMS signal. It is a synchronous state machine that controls the operation of the JTAG logic. The state machine is shown in Figure 7-2; the value shown adjacent to each arc represents the value of the TMS signal sampled on the rising edge of the TCK signal. For a description of the TAP controller states, please refer to the IEEE 1149.1 document.

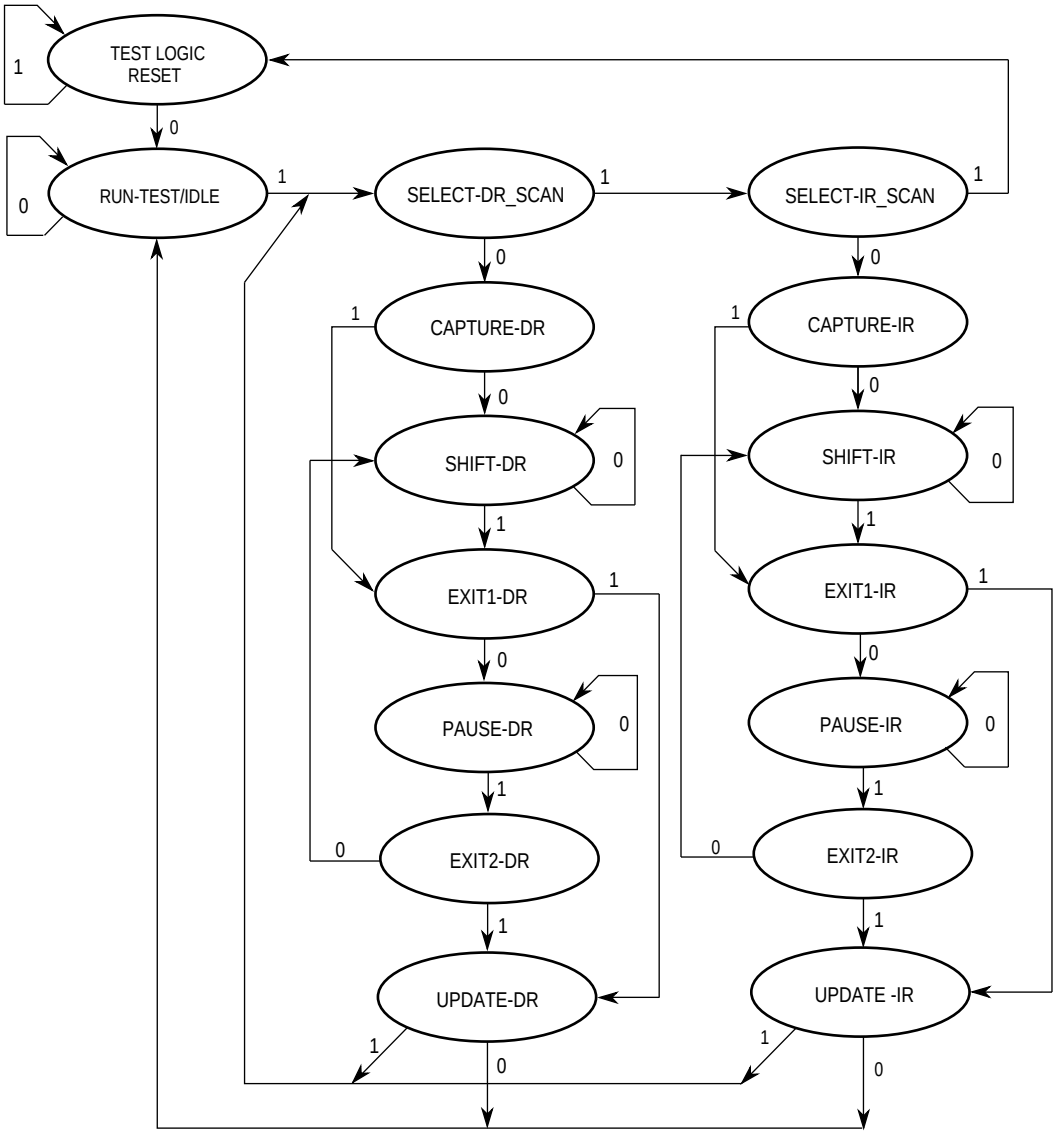


Figure 7-2. TAP Controller State Machine

7.3 BOUNDARY SCAN REGISTER

The MC68306 IEEE 1149.1 implementation has a 124-bit boundary scan register. This register contains bits for all device signal and clock pins and associated control signals.

The XTAL and X2 pins are associated with analog signals and are not included in the boundary scan register.

All MC68306 bidirectional pins, except the open-drain I/O pins ($\overline{\text{HALT}}$, $\overline{\text{DTACK}}$, $\overline{\text{BERR}}$, and $\overline{\text{RESET}}$), have a single register bit for pin data and an associated control bit in the boundary scan register. All open drain I/O pins have a single register bit for pin data and no associated control bit. To ensure proper operation, the open-drain pins require external pullups. Twenty-four control bits in the boundary scan register define the output enable signal for associated groups of bidirectional and three-state pins. The control bits and their bit positions are listed in Table 7-1.

Table 7-1. Boundary Scan Control Bits

Name	Bit Number	Name	Bit Number	Name	Bit Number
$\overline{\text{OPOE3}}$	8	$\overline{\text{PPOE15}}$	26	$\overline{\text{PPOE7}}$	42
$\overline{\text{PPOE8}}$	12	$\overline{\text{PPOE0}}$	28	DOE	58
$\overline{\text{PPOE9}}$	14	$\overline{\text{PPOE1}}$	30	HiZ	67
$\overline{\text{PPOE10}}$	16	$\overline{\text{PPOE2}}$	32	$\overline{\text{DRAMWOE}}$	86
$\overline{\text{PPOE11}}$	18	$\overline{\text{PPOE3}}$	34	$\overline{\text{DRAMOE}}$	88
$\overline{\text{PPOE12}}$	20	$\overline{\text{PPOE4}}$	36	CPMOE	97
$\overline{\text{PPOE13}}$	22	$\overline{\text{PPOE5}}$	38	$\overline{\text{CSOE}}$	100
$\overline{\text{PPOE14}}$	24	$\overline{\text{PPOE6}}$	40	$\overline{\text{AOE}}$	118

Boundary scan bit definitions are shown in Table 7-2. The first column in Table 7-2 defines the bit's ordinal position in the boundary scan register. The shift register bit nearest TDO (i.e., first to be shifted out) is defined as bit 0; the last bit to be shifted out is bit 123.

The second column references one of the five MC68306 cell types depicted in Figures 7-3–7-7, which describe the cell structure for each type.

The third column lists the pin name for all pin-related bits or defines the name of bidirectional control register bits.

The last column indicates the associated boundary scan register control bit.

Bidirectional pins include a single scan bit for data (IO.Cell) as depicted in Figure 7-7. These bits are controlled by one of the two bits shown in Figures 7-5 and 7-6. The value of the control bit determines whether the bidirectional pin is an input or an output. One or more bidirectional data bits can be serially connected to a control bit as shown in Figure 7-8. Note that, when sampling the bidirectional data bits, the bit data can be interpreted only after examining the IO control bit to determine pin direction.

Table 7-2. Boundary Scan Bit Definitions

Bit Num	Cell Type	Signal	Control
0	O.Cell	OP1	HiZ
1	O.Cell	OP0	HiZ
2	I.Cell	IP1	
3	I.Cell	IP0	
4	O.Cell	TXDB	HiZ
5	I.Cell	RXDB	
6	O.Cell	TXDA	HiZ
7	I.Cell	RXDA	
8	En.Cell	$\overline{\text{OPOE3}}$	
9	O.Cell	OP3	$\overline{\text{OPOE3}}$
10	I.Cell	IP2	
11	I.Cell	X1	
12	En.Cell	$\overline{\text{PPOE8}}$	
13	IO.Cell	PA0	$\overline{\text{PPOE8}}$
14	En.Cell	$\overline{\text{PPOE9}}$	
15	IO.Cell	PA1	$\overline{\text{PPOE9}}$
16	En.Cell	$\overline{\text{PPOE10}}$	
17	IO.Cell	PA2	$\overline{\text{PPOE10}}$
18	En.Cell	$\overline{\text{PPOE11}}$	
19	IO.Cell	PA3	$\overline{\text{PPOE11}}$
20	En.Cell	$\overline{\text{PPOE12}}$	
21	IO.Cell	PA4	$\overline{\text{PPOE12}}$
22	En.Cell	$\overline{\text{PPOE13}}$	
23	IO.Cell	PA5	$\overline{\text{PPOE13}}$
24	En.Cell	$\overline{\text{PPOE14}}$	
25	IO.Cell	PA6	$\overline{\text{PPOE14}}$
26	En.Cell	$\overline{\text{PPOE15}}$	
27	IO.Cell	PA7	$\overline{\text{PPOE15}}$
28	En.Cell	$\overline{\text{PPOE0}}$	
29	IO.Cell	PB0/$\overline{\text{IACK2}}$	$\overline{\text{PPOE0}}$
30	En.Cell	$\overline{\text{PPOE1}}$	
31	IO.Cell	PB1/$\overline{\text{IACK3}}$	$\overline{\text{PPOE1}}$
32	En.Cell	$\overline{\text{PPOE2}}$	
33	IO.Cell	PB2/$\overline{\text{IACK5}}$	$\overline{\text{PPOE2}}$

Bit Num	Cell Type	Signal	Control
34	En.Cell	$\overline{\text{PPOE3}}$	
35	IO.Cell	PB3/$\overline{\text{IACK6}}$	$\overline{\text{PPOE3}}$
36	En.Cell	$\overline{\text{PPOE4}}$	
37	IO.Cell	PB4/IRQ2	$\overline{\text{PPOE4}}$
38	En.Cell	$\overline{\text{PPOE5}}$	
39	IO.Cell	PB5/IRQ3	$\overline{\text{PPOE5}}$
40	En.Cell	$\overline{\text{PPOE6}}$	
41	IO.Cell	PB6/IRQ5	$\overline{\text{PPOE6}}$
42	En.Cell	$\overline{\text{PPOE7}}$	
43	IO.Cell	PB7/IRQ6	$\overline{\text{PPOE7}}$
44	O.Cell	$\overline{\text{IACK1}}$	HiZ
45	O.Cell	$\overline{\text{IACK4}}$	HiZ
46	O.Cell	$\overline{\text{IACK7}}$	HiZ
47	I.Cell	IRQ1	
48	I.Cell	IRQ4	
49	I.Cell	IRQ7	
50	IO.Cell	D0	DOE
51	IO.Cell	D1	DOE
52	IO.Cell	D2	DOE
53	IO.Cell	D3	DOE
54	IO.Cell	D4	DOE
55	IO.Cell	D5	DOE
56	IO.Cell	D6	DOE
57	IO.Cell	D7	DOE
58	En.Cell	DOE	
59	IO.Cell	D8	DOE
60	IO.Cell	D9	DOE
61	IO.Cell	D10	DOE
62	IO.Cell	D11	DOE
63	IO.Cell	D12	DOE
64	IO.Cell	D13	DOE
65	IO.Cell	D14	DOE
66	IO.Cell	D15	DOE
67	En.Cell	HiZ	

Table 7-2. Boundary Scan Bit Definitions (Continued)

Bit Num	Cell Type	Signal	Control
68	IO.Cell	$\overline{\text{BERR}}$	$\overline{\text{BERR}}$
69	IO.Cell	$\overline{\text{DTACK}}$	$\overline{\text{DTACK}}$
70	IO.Cell	FC0	CPMOE
71	IO.Cell	FC1	CPMOE
72	IO.Cell	FC2	CPMOE
73	IOx0.Cell	$\overline{\text{RESET}}$	$\overline{\text{RESET}}$
74	IOx0.Cell	$\overline{\text{HALT}}$	$\overline{\text{HALT}}$
75	O.Cell	CLKOUT	HiZ
76	I.Cell	$\overline{\text{BR}}$	
77	O.Cell	$\overline{\text{BG}}$	HiZ
78	I.Cell	$\overline{\text{BGACK}}$	
79	IO.Cell	$\overline{\text{AS}}$	CPMOE
80	IO.Cell	R/$\overline{\text{W}}$	CPMOE
81	IO.Cell	$\overline{\text{UDS}}$	CPMOE
82	IO.Cell	$\overline{\text{LDS}}$	CPMOE
83	O.Cell	$\overline{\text{UW}}$	HiZ
84	O.Cell	$\overline{\text{LW}}$	HiZ
85	O.Cell	$\overline{\text{OE}}$	HiZ
86	En.Cell	$\overline{\text{DRAMWOE}}$	
87	O.Cell	$\overline{\text{DRAMW}}$	$\overline{\text{DRAMWOE}}$
88	En.Cell	$\overline{\text{DRAMOE}}$	
89	O.Cell	$\overline{\text{RAS1}}$	$\overline{\text{DRAMOE}}$
90	O.Cell	$\overline{\text{RAS0}}$	$\overline{\text{DRAMOE}}$
91	O.Cell	$\overline{\text{CAS1}}$	$\overline{\text{DRAMOE}}$
92	O.Cell	$\overline{\text{CAS0}}$	$\overline{\text{DRAMOE}}$
93	O.Cell	$\overline{\text{CS0}}$	$\overline{\text{CSOE}}$
94	O.Cell	$\overline{\text{CS1}}$	$\overline{\text{CSOE}}$
95	O.Cell	$\overline{\text{CS2}}$	$\overline{\text{CSOE}}$
96	O.Cell	$\overline{\text{CS3}}$	$\overline{\text{CSOE}}$
97	En.Cell	CPMOE	
98	O.Cell	A20/$\overline{\text{CS4}}$	$\overline{\text{CSOE}}$
99	O.Cell	A21/$\overline{\text{CS5}}$	$\overline{\text{CSOE}}$
100	En.Cell	$\overline{\text{CSOE}}$	
101	O.Cell	A22–$\overline{\text{CS6}}$	$\overline{\text{CSOE}}$
102	O.Cell	A23–$\overline{\text{CS7}}$	$\overline{\text{CSOE}}$
103	IO.Cell	A1/DRAMA0	CPMOE
104	IO.Cell	A2/DRAMA1	CPMOE
105	IO.Cell	A3/DRAMA2	CPMOE
106	IO.Cell	A4/DRAMA3	CPMOE
107	IO.Cell	A5/DRAMA4	CPMOE
108	IO.Cell	A6/DRAMA5	CPMOE
109	IO.Cell	A7/DRAMA6	CPMOE
110	IO.Cell	A8/DRAMA7	CPMOE
111	IO.Cell	A9/DRAMA8	CPMOE
112	IO.Cell	A10/DRAMA9	CPMOE
113	IO.Cell	A11/DRAMA10	CPMOE
114	IO.Cell	A12/DRAMA11	CPMOE
115	IO.Cell	A13/DRAMA12	CPMOE
116	IO.Cell	A14/DRAMA13	CPMOE
117	IO.Cell	A15/DRAMA14	CPMOE
118	En.Cell	$\overline{\text{AOE}}$	
119	O.Cell	A16	$\overline{\text{AOE}}$
120	O.Cell	A17	$\overline{\text{AOE}}$
121	O.Cell	A18	$\overline{\text{AOE}}$
122	O.Cell	A19	$\overline{\text{AOE}}$
123	I.Cell	AMODE	

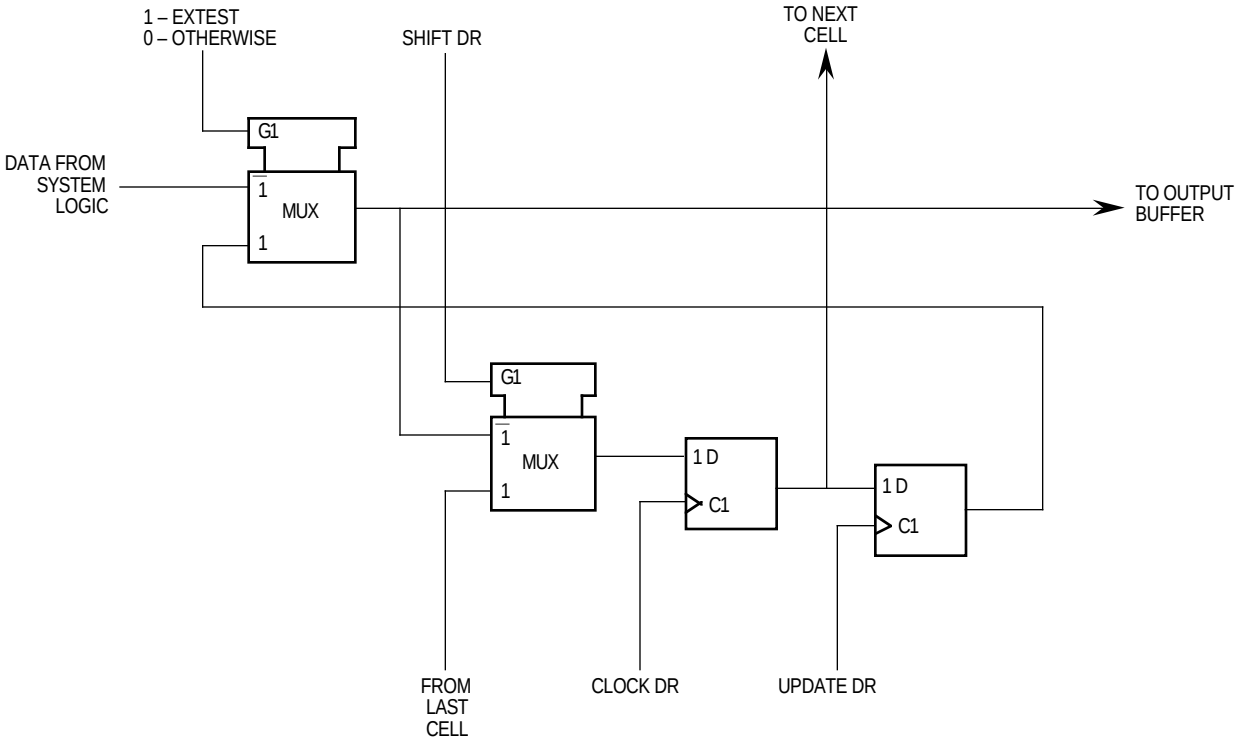


Figure 7-3. Output Cell (O.Cell)

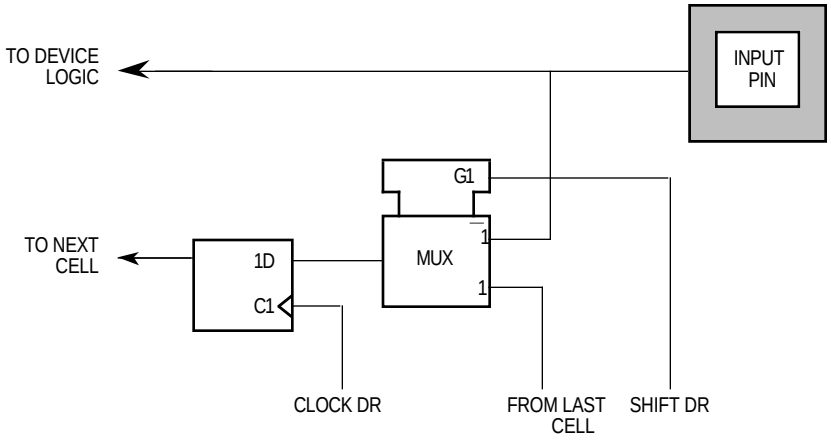


Figure 7-4. Input Cell (I.Cell)

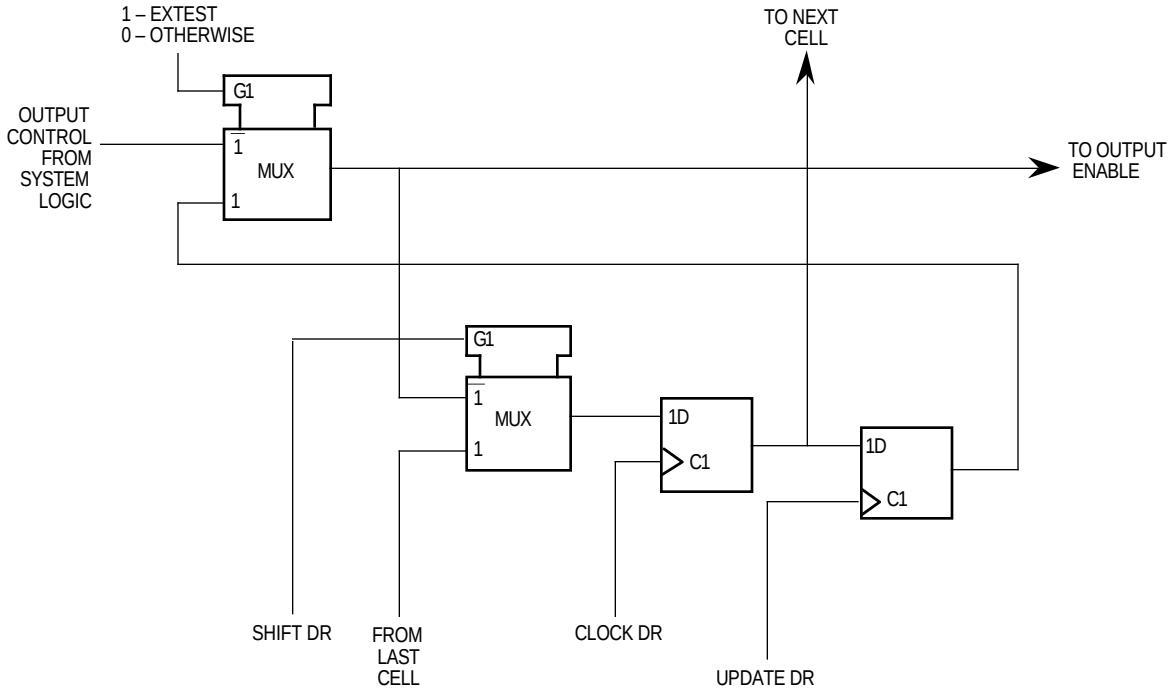


Figure 7-5. Output Control Cell (En.Cell)

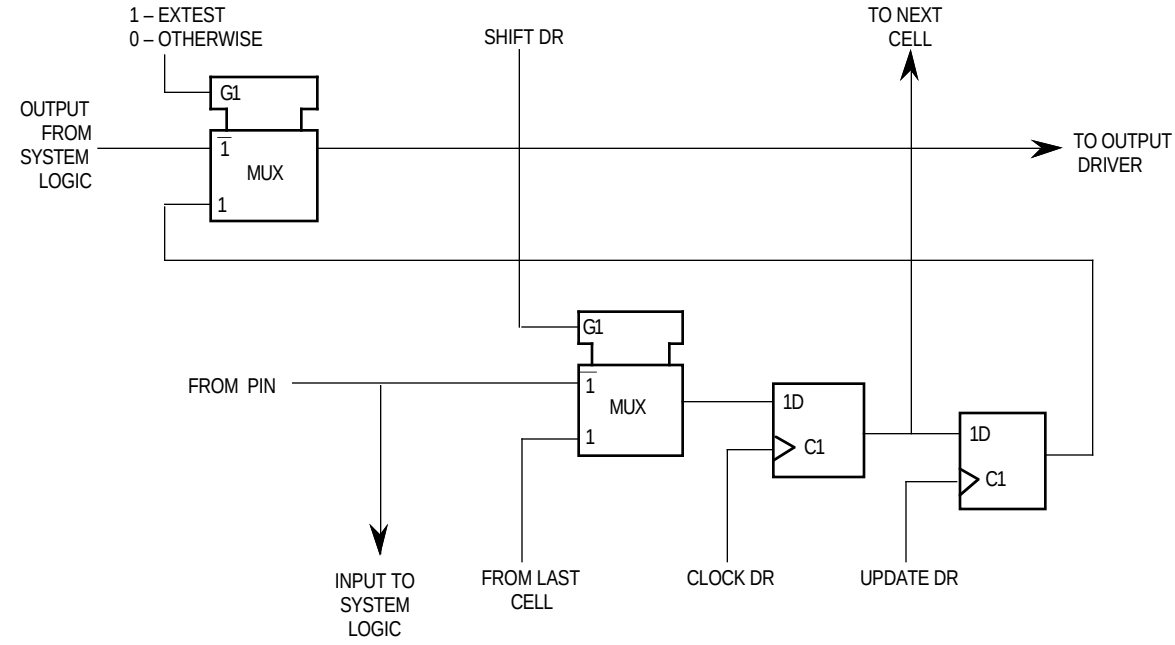


Figure 7-6. Bidirectional Cell (IO.Cell)

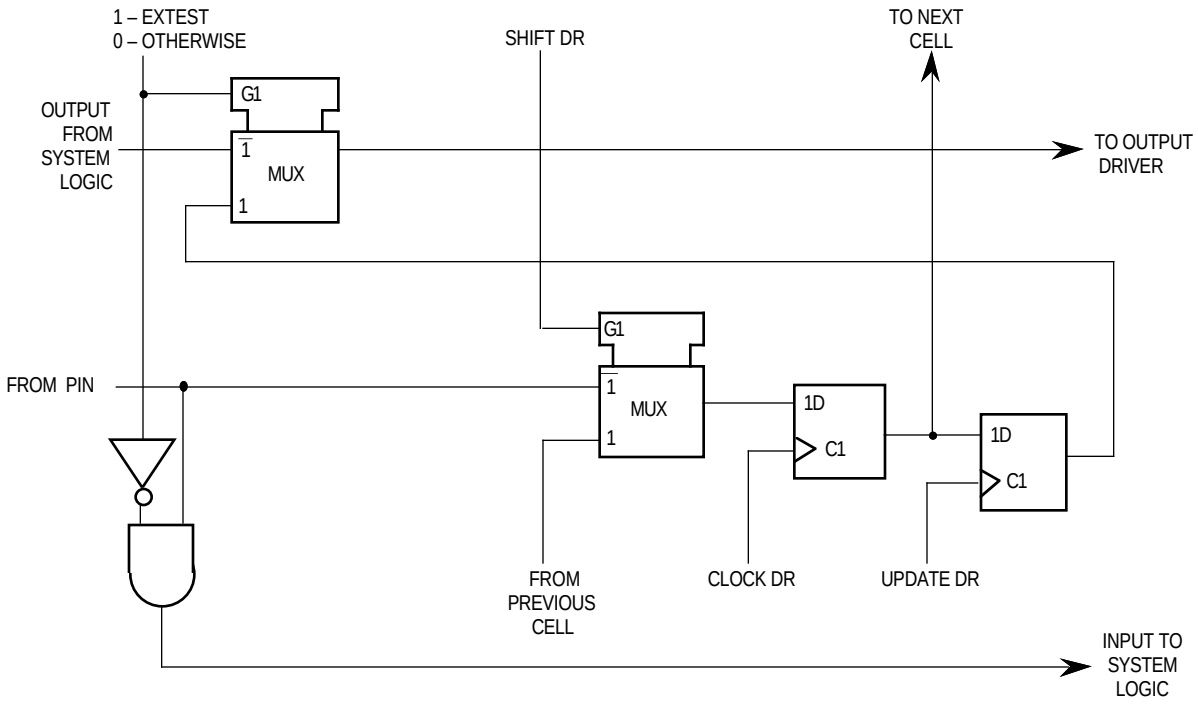
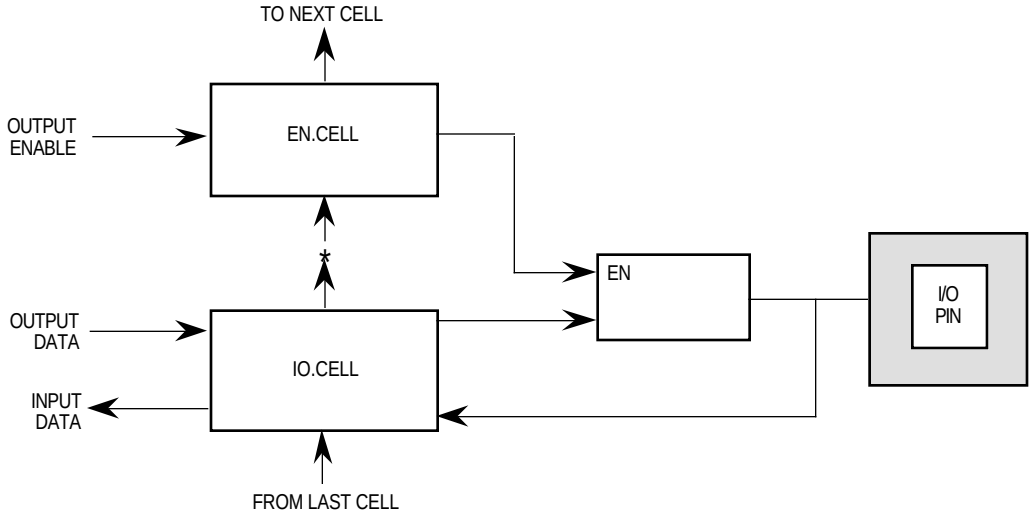


Figure 7-7. Bidirectional Cell (IOx0.Cell)



NOTE: More than one IO.Cell could be serially connected and controlled by a single En.Cell.

Figure 7-8. General Arrangement for Bidirectional Pins

7.4 INSTRUCTION REGISTER

The MC68306 IEEE 1149.1 implementation includes the three mandatory public instructions (EXTEST, SAMPLE/PRELOAD, and BYPASS), the optional public ID instruction, plus one additional public instruction (CLAMP) defined by IEEE 1149.1. The

MC68306 includes a 3-bit instruction register without parity, consisting of a shift register with three parallel outputs. Data is transferred from the shift register to the parallel outputs during the update-IR controller state. The three bits are used to decode the six unique instructions listed in Table 7-3.

The parallel output of the instruction register is reset to 001 in the test-logic-reset controller state. Note that this preset state is equivalent to the ID instruction.

Table 7-3. Instructions

Code			Instruction
B2	B1	B0	
0	0	0	EXTEST
0	0	1	ID
0	1	0	BYPASS
0	1	1	CLAMP
1	0	0	MODULE MODE
1	0	1	BYPASS
1	1	0	SAMPLE/PRELOAD
1	1	1	BYPASS

During the capture-IR controller state, the parallel inputs to the instruction shift register are loaded with the 3-bit binary value (001). The parallel outputs, however, remain unchanged by this action since an update-IR signal is required to modify them.

7.4.1 EXTEST (000)

The external test (EXTEST) instruction selects the 124-bit boundary scan register. EXTEST asserts internal reset for the MC68306 system logic to force a predictable benign internal state while performing external boundary scan operations.

By using the TAP, the register is capable of a) scanning user-defined values into the output buffers, b) capturing values presented to input pins, c) controlling the direction of bidirectional pins, and d) controlling the output drive of three-state output pins. For more details on the function and uses of EXTEST, please refer to the IEEE 1149.1 document.

7.4.2 SAMPLE/PRELOAD (110)

The SAMPLE/PRELOAD instruction selects the 124-bit boundary scan register and provides two separate functions. First, it provides a means to obtain a snapshot of system data and control signals. The snapshot occurs on the rising edge of TCK in the capture-DR controller state. The data can be observed by shifting it transparently through the boundary scan register.

NOTE

Since there is no internal synchronization between the IEEE 1149.1 clock (TCK) and the system clock (CLKOUT), the user must provide some form of external synchronization to achieve meaningful results.

The second function of SAMPLE/PRELOAD is to initialize the boundary scan register output bits prior to selection of EXTEST. This initialization ensures that known data will appear on the outputs when entering the EXTEST instruction.

7.4.3 BYPASS (010, 101, 111)

The BYPASS instruction selects the single-bit bypass register as shown in Figure 7-9. This creates a shift-register path from TDI to the bypass register and, finally, to TDO, circumventing the 124-bit boundary scan register. This instruction is used to enhance test efficiency when a component other than the MC68306 becomes the device under test.

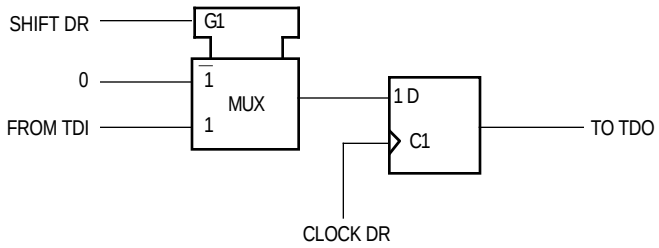


Figure 7-9. Bypass Register

When the bypass register is selected by the current instruction, the shift-register stage is set to a logic zero on the rising edge of TCK in the capture-DR controller state. Therefore, the first bit to be shifted out after selecting the bypass register will always be a logic zero.

7.4.4 CLAMP (011)

When the CLAMP instruction is invoked, the boundary scan multiplexer control signal EXTEST is asserted, and the BYPASS register is selected. CLAMP should be invoked after valid data has been shifted into the boundary scan register, e.g. by SAMPLE/PRELOAD. CLAMP allows static levels to be presented at the MC68306 output and bidirectional pins, like EXTEST, but without the shift latency of the boundary scan register from TDI to TDO.

7.5 MC68306 RESTRICTIONS

The control afforded by the output enable signals using the boundary scan register and the EXTEST instruction requires a compatible circuit-board test environment to avoid device-destructive configurations. The user must avoid situations in which the MC68306 output drivers are enabled into actively driven networks. Overdriving the TDO driver when it is active is not recommended.

Also, the MC68306 contains dynamic logic, so EXTAL must be driven by a free-running clock at all times.

7.6 NON-IEEE 1149.1 OPERATION

In non-IEEE 1149.1 operation, the IEEE 1149.1 test logic must be kept transparent to the system logic by forcing the TAP controller into the test-logic-reset state. This requires either:

1. An active (low) signal applied to $\overline{\text{TRST}}$.
2. A minimum of five consecutive TCK rising edges withh TMS high. TMS has an internal pullup, and may be left unconnected.

If TMS either remains unconnected or is connected to V_{CC}, then the TAP controller cannot leave the test-logic-reset state, regardless of the state of TCK or $\overline{\text{TRST}}$.

SECTION 8
ELECTRICAL CHARACTERISTICS

This section contains detailed information on power considerations, DC/AC electrical characteristics, and AC timing specifications of the MC68306. Refer to **Section 9 Ordering Information and Mechanical Data** for specific part numbers corresponding to voltage, frequency, and temperature ratings.

8.1 MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Supply Voltage [1, 2]	V_{CC}	–0.3 to + 7.0	V
Input Voltage [1, 2]	V_{in}	–0.3 to + 7.0	V
Operating Temperature Range	T_A	0 to 70	°C
Storage Temperature Range	T_{stg}	–55 to +150	°C

NOTES:

1. Permanent damage can occur if maximum ratings are exceeded. Exposure to voltages or currents in excess of recommended values affects device reliability. Device modules may not operate normally while being exposed to electrical extremes.
2. Although sections of the device contain circuitry to protectagainst damage from high static voltages or electrical fields, take normal precautions to avoid exposure to voltages higher than maximum-rated voltages.

This device contains protective circuitry against damage due to high static voltages or electrical fields; however, it is advised that normal precautions be taken to avoid application of any voltages higher than maximum-rated voltages to this high-impedance circuit. Reliability of operation is enhanced if unused inputs are tied to an appropriate logic voltage level (e.g., either GND or V_{CC}).

The following ratings define a range of conditions in which the device will operate without being damaged. However, sections of the device may not operate normally while being exposed to the electrical extremes.

8.2 THERMAL CHARACTERISTICS

Characteristic	Symbol	Value	Unit
Thermal Resistance—Junction to Case Plastic 132-Pin QFP Plastic 144-Pin Thin QFP	θ_{JC}	 20* 20*	°C/W
Thermal Resistance—Junction to Ambient Plastic 132-Pin QFP Plastic 144-Pin Thin QFP	θ_{JA}	 42* 42*	°C/W

* Estimated

8.3 POWER CONSIDERATIONS

The average chip-junction temperature, T_J, in °C can be obtained from:

$$T_J = T_A + (P_D \bullet \theta_{JA}) \qquad (1)$$

where:

T_A	=	Ambient Temperature, °C
θ_{JA}	=	Package Thermal Resistance, Junction-to-Ambient, °C/W
P_D	=	$P_{INT} + P_{I/O}$
P_{INT}	=	I_C C x V_{CC}, Watts—Chip Internal Power
$P_{I/O}$	=	Power Dissipation on Input and Output Pins—User Determined

For most applications, $P_{I/O} < P_{INT}$ and can be neglected.

An approximate relationship between P_D and T_J (if $P_{I/O}$ is neglected) is:

$$P_D = K \div (T_J + 273°C)$$

Solving Equations (1) and (2) for K gives:

$$K = P_D \bullet (T_A + 273°C) + \theta_{JA} \bullet P_D^2$$

where K is a constant pertaining to the particular part. K can be determined from equation (3) by measuring P_D (at thermal equilibrium) for a known T_A. Using this value of K, the values of P_D and T_J can be obtained by solving Equations (1) and (2) iteratively for any value of T_A.

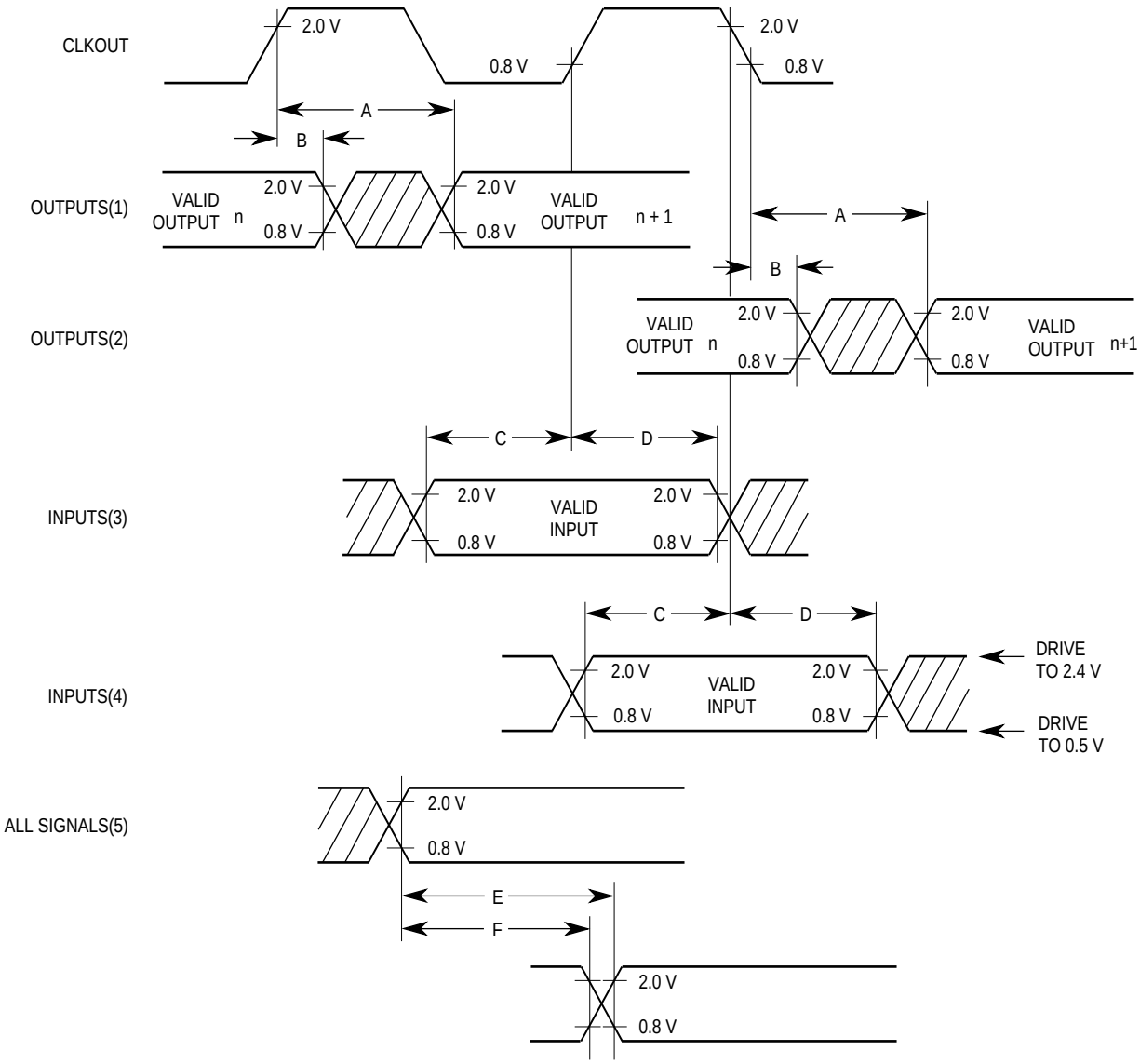
8.4 AC ELECTRICAL SPECIFICATION DEFINITIONS

The AC specifications presented consist of output delays, input setup and hold times, and signal skew times. All signals are specified relative to an appropriate edge of the clock and possibly to one or more other signals.

The measurement of the AC specifications is defined by the waveforms shown in Figure 8-1. To test the parameters guaranteed by Motorola, inputs must be driven to the voltage levels specified in that figure. Outputs are specified with minimum and/or maximum limits, as appropriate, and are measured as shown in Figure 8-1. Inputs are specified with minimum setup and hold times and are measured as shown. Finally, the measurement for signal-to-signal specifications is also shown.

NOTE

The testing levels used to verify conformance to the AC specifications do not affect the guaranteed DC operation of the device as specified in the DC electrical specifications.



NOTES:
1. This output timing is applicable to all parameters specified relative to the rising edge of the clock.
2. This output timing is applicable to all parameters specified relative to the falling edge of the clock.
3. This input timing is applicable to all parameters specified relative to the rising edge of the clock.
4. This input timing is applicable to all parameters specified relative to the falling edge of the clock.
5. This timing is applicable to all parameters specified relative to the assertion/negation of another signal.

LEGEND:
A. Maximum output delay specification.
B. Minimum output hold time.
C. Minimum input setup time specification.
D. Minimum input hold time specification.
E. Signal valid to signal valid specification (maximum or minimum).
F. Signal valid to signal invalid specification (maximum or minimum).

Figure 8-1. Drive Levels and Test Points for AC Specifications

8.5 DC ELECTRICAL SPECIFICATIONS

(The electrical specifications in this document are preliminary)

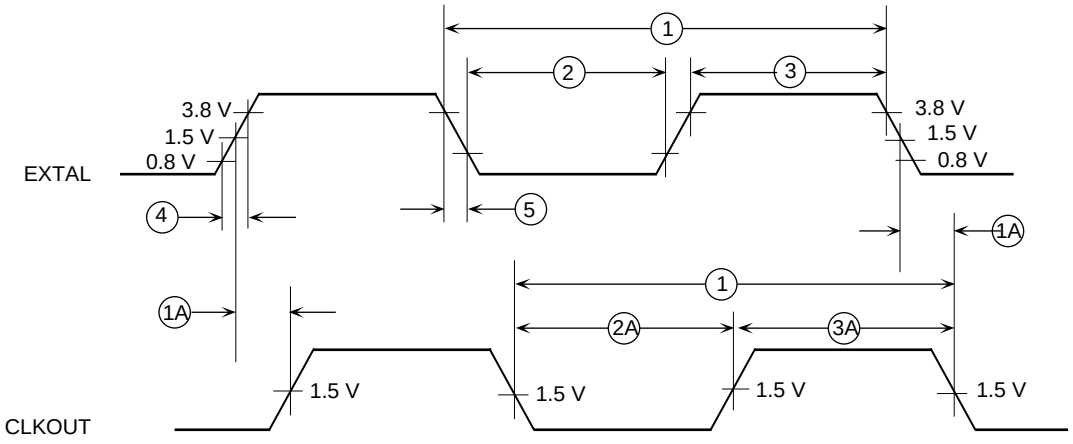
Characteristic	Symbol	Min	Max	Unit
Input High Voltage (except clock)	V_{IH}	2.0	V_{CC}	V
Input Low Voltage	V_{IL}	GND– 0.3	0.8	V
Clock Input High Voltage	V_{IHC}	0.7*(V_{CC})	V_{CC} +0.3	V
Input Leakage Current (All Input Only Pins)[1] V_{in} = V_{CC} or GND	I_{in}	–2.5	2.5	µA
Three-State (Off State) Input Current @ 2.4 V/0.4 V	I_{TSI}	—	20	µA
Output High Voltage (I_{OH} = Rated Maximum)	V_{OH}	V_{CC} –0.75	—	V
Output Low Voltage (I_{OL} = Rated Maximum)	V_{OL}	—	0.5	V
Current Drain[2] TA = 70°C, V_{CC} = 5.25 V, f = 16.67 MHz	I_D	—	100	mA
Power Dissipation f = 16.67 MHz	P_D	—	0.5	W
Input Capacitance[3]	C_{in}			pF
All Input-Only Pins		—	10	
All I/O Pins		—	20	
Load Capacitance[3]	C_L	—	100	pF

1. Not including internal pullup or pulldown
2. Currents listed are with no loading.
3. Capacitance is periodically sampled rather than 100% tested.

8.6 AC ELECTRICAL SPECIFICATIONS—CLOCK TIMING

(The electrical specifications in this document are preliminary; see Figure 8-2.)

Num.	Characteristic	Symbol	Min	Max	Unit
	Frequency of Operation	f	8.0	16.7	MHz
1	Cycle time	t_{cyc}	60	125	ns
	Crystal Oscillator Start-Up Time	t_{gidyup}	—	TBD	ms
2	EXTAL Pulse Width Low	t_{CW}	27	62.5	ns
3	EXTAL Pulse Width High	t_{CW}	27	62.5	ns
4,5	EXTAL Rise and Fall Times	t_{Cr}	—	5	ns
		t_{Cf}	—	5	
1A	External Clock to CLKOUT Skew (Typical)		8		ns
2A	CLKOUT Pulse Width Low		(2) –5	(2) +5	ns
3A	CLKOUT Pulse Width High		(3) –5	(3) +5	ns



NOTE: Timing measurements are referenced to and from a low voltage of 0.8 V and a high voltage of 3.8 V, unless otherwise noted. The voltage swing through this range should start outside and pass through the range such that the rise or fall will be linear between 0.8 V and 3.8 V.

Figure 8-2. Clock Output Timing

8.7 AC ELECTRICAL SPECIFICATIONS—READ AND WRITE CYCLES (The electrical specifications in this document are preliminary; see Figures 8-3 and 8-4)

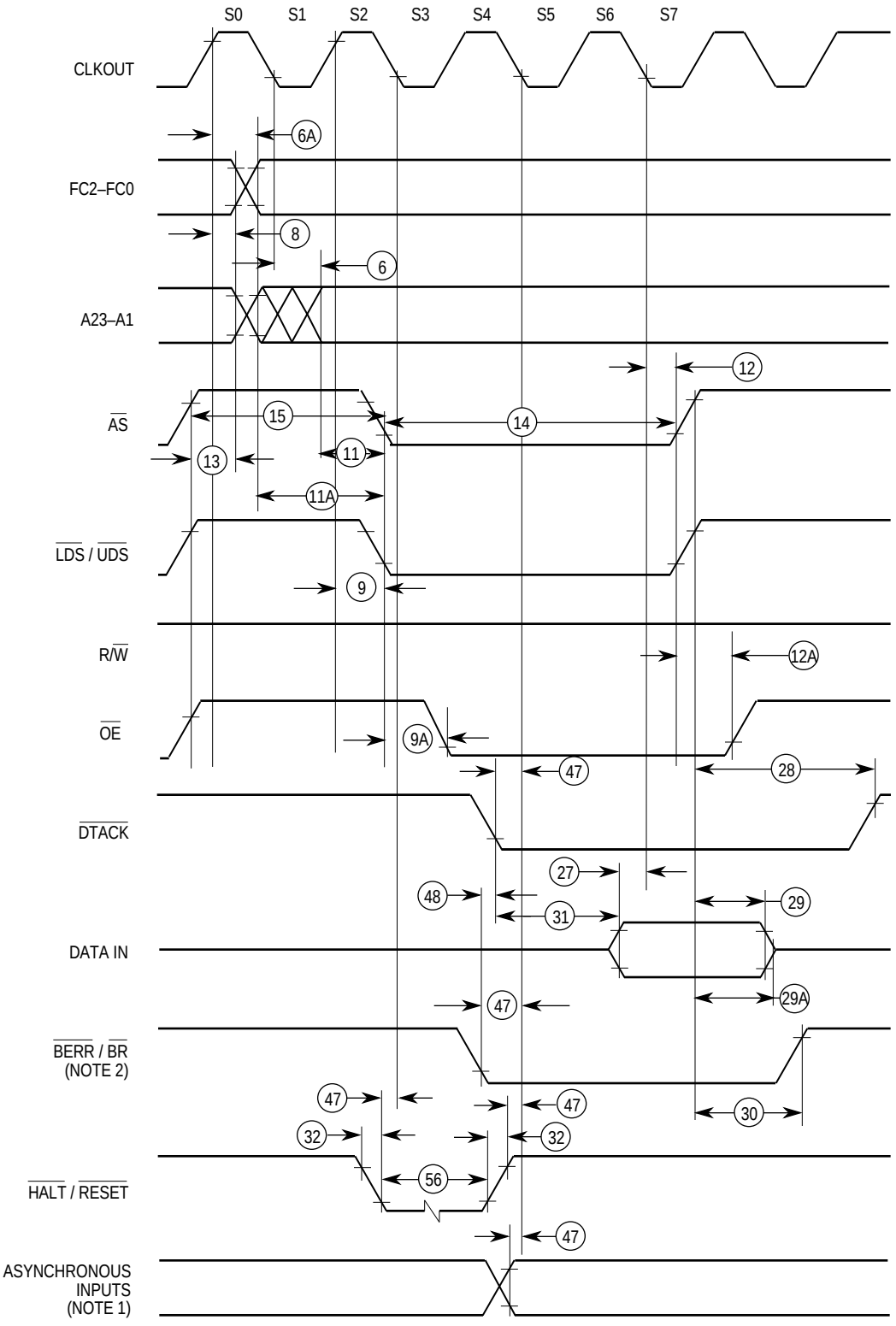
		16.67 MHz		
Num	**Characteristic**	**Min**	**Max**	**Unit**
6	CLKOUT Low to Address Valid (Row Address for DRAM Cycle)	—	30	ns
6A	CLKOUT High to FC Valid	—	30	ns
7	CLKOUT High to Data Bus High Impedance (Maximum)	—	50	ns
8	CLKOUT High to Address, FC Invalid (Minimum)	0	—	ns
9[1]	CLKOUT High to $\overline{AS}$, $\overline{LDS}$, $\overline{UDS}$ Asserted	3	30	ns
9A	$\overline{UDS}$, $\overline{LDS}$ Asserted to $\overline{OE}$, $\overline{UW}$, $\overline{LW}$ Asserted	0	15	ns
11[2]	Address Valid to $\overline{AS}$, $\overline{LDS}$, $\overline{UDS}$ Asserted (Read)/ $\overline{AS}$ Asserted (Write)	15	—	ns
11A[2]	FC Valid to $\overline{AS}$, $\overline{LDS}$, $\overline{UDS}$ Asserted (Read)/ $\overline{AS}$, Asserted (Write)	45	—	ns
12[1]	CLKOUT Low to $\overline{AS}$, $\overline{LDS}$, $\overline{UDS}$ Negated	3	30	ns
12A	$\overline{UDS}$, $\overline{LDS}$ Negated to $\overline{OE}$, $\overline{UW}$, $\overline{LW}$ Negated	0	15	ns
13[2]	$\overline{AS}$, $\overline{LDS}$, $\overline{UDS}$ Negated to Address, FC Invalid	15	—	ns
14[2]	$\overline{AS}$ (and $\overline{LDS}$, $\overline{UDS}$ Read) Width Asserted	120	—	ns
14A[2]	$\overline{LDS}$, $\overline{UDS}$, Width Asserted (Write)	50	—	ns
15[2]	$\overline{AS}$, $\overline{LDS}$, $\overline{UDS}$ Width Negated	60	—	ns
16	CLKOUT High to Control Bus High Impedance	—	50	ns
17[2]	$\overline{AS}$, $\overline{LDS}$, $\overline{UDS}$ Negated to R/$\overline{W}$ Invalid	15	—	ns
18[1]	CLKOUT High to R/$\overline{W}$ High (Read)	0	30	ns
20[1]	CLKOUT High to R/$\overline{W}$ Low (Write)	0	30	ns

8.7 AC ELECTRICAL SPECIFICATIONS—READ AND WRITE CYCLES (Continued)

		16.67 MHz		
Num	**Characteristic**	**Min**	**Max**	**Unit**
20A[6]	$\overline{AS}$ Asserted to R/$\overline{W}$ Low (Write)	—	10	ns
21[2]	Address Valid to R/$\overline{W}$ Low (Write)	0	—	ns
21A[2]	FC Valid to R/$\overline{W}$ Low (Write)	30	—	ns
22[2]	R/$\overline{W}$ Low to $\overline{LDS}$, $\overline{UDS}$ Asserted (Write)	30	—	ns
23	CLKOUT Low to Data-Out Valid (Write)	—	30	ns
25[2]	$\overline{AS}$, $\overline{LDS}$, $\overline{UDS}$ Negated to Data-Out Invalid (Write)	15	—	ns
26[2]	Data-Out Valid to $\overline{LDS}$, $\overline{UDS}$ Asserted (Write)	15	—	ns
27[5]	Data-In Valid to CLKOUT Low (Setup Time on Read)	5	—	ns
28[2]	$\overline{AS}$, $\overline{LDS}$, $\overline{UDS}$ Negated to $\overline{DTACK}$ Negated (Asynchronous Hold)	0	110	ns
29	$\overline{AS}$, $\overline{LDS}$, $\overline{UDS}$ Negated to Data-In Invalid (Hold Time on Read)	0	—	ns
29A	$\overline{AS}$, $\overline{LDS}$, $\overline{UDS}$ Negated to Data-In High Impedance	—	90	ns
30	$\overline{AS}$, $\overline{LDS}$, $\overline{UDS}$ Negated to $\overline{BERR}$ Negated	0	—	ns
31[2,5]	$\overline{DTACK}$ Asserted to Data-In Valid (Setup Time)	—	50	ns
32	$\overline{HALT}$ and $\overline{RESET}$ Input Transition Time	—	150	ns
47[5]	Asynchronous Input Setup Time	5	—	ns
48[3]	$\overline{BERR}$ Asserted to $\overline{DTACK}$ Asserted	10	—	ns
53	Data-Out Hold from CLKOUT High	0	—	ns
55	R/$\overline{W}$ Asserted to Data Bus Impedance Change	0	—	ns
56[4]	$\overline{HALT}$/$\overline{RESET}$ Pulse Width	10	—	Clks

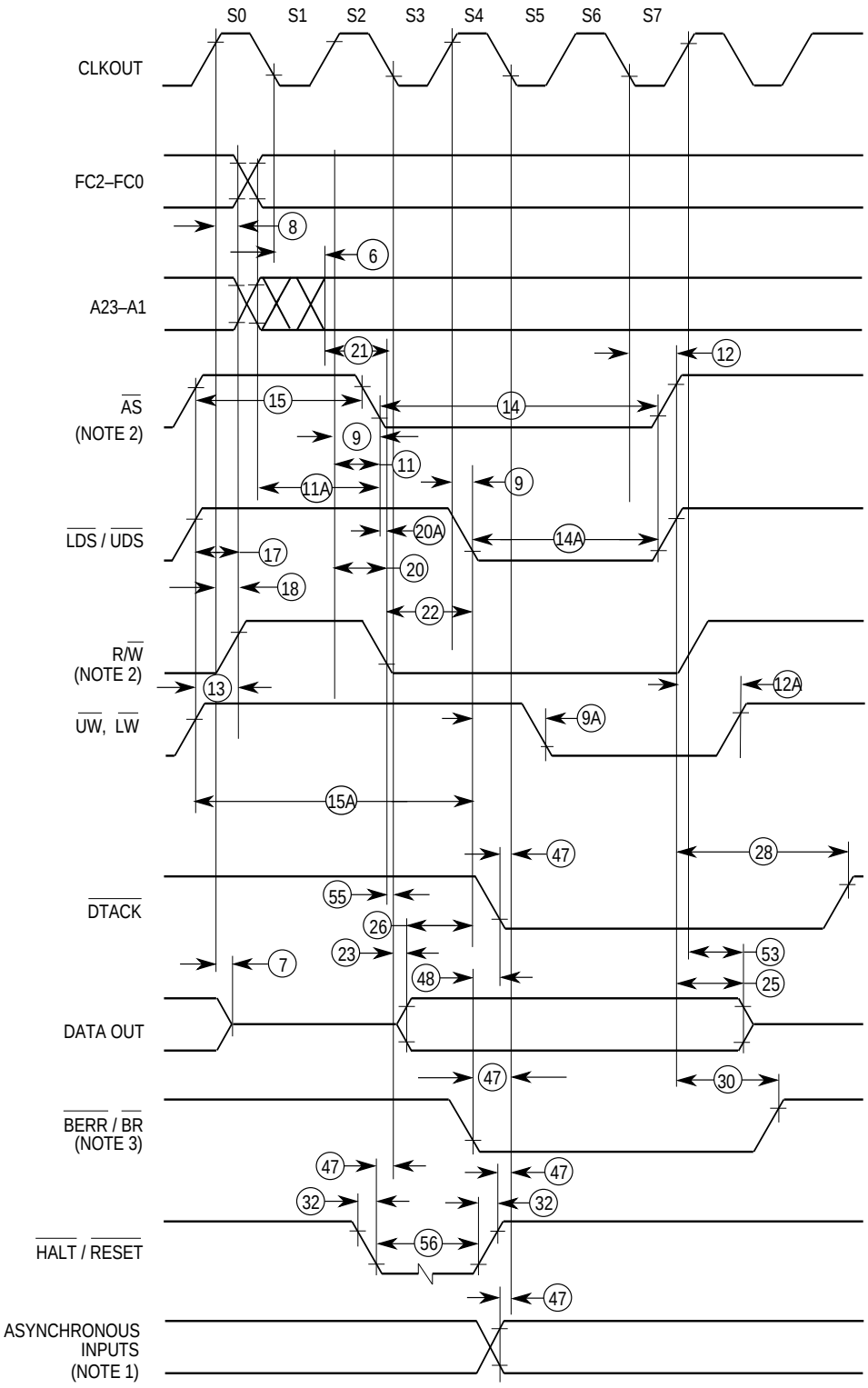
NOTES:

1. For a loading capacitance of less than or equal to 50 pF, subtract 5 ns from the value given in the maximum columns.
2. Actual value depends on clock period.
3. If #47 is satisfied for both $\overline{DTACK}$ and $\overline{BERR}$, #48 may be ignored. In the absence of $\overline{DTACK}$, $\overline{BERR}$ is an asynchronous input using the asynchronous input setup time (#47).
4. For power-up, the MC68306 must be held in the reset state for 100 ms to allow stabilization of on-chip circuitry. After the system is powered up, #56 refers to the minimum pulse width required to reset the controller.
5. If the asynchronous input setup time (#47) requirement is satisfied for $\overline{DTACK}$, the $\overline{DTACK}$ asserted to data setup time (#31) requirement can be ignored. The data must only satisfy the data-in to clock low setup time (#27) for the following clock cycle.
6. When $\overline{AS}$ and R/$\overline{W}$ are equally loaded (±20%), subtract 5 ns from the values given in these columns.
7. The processor will negate $\overline{BG}$ and begin driving the bus again if external arbitration logic negates $\overline{BR}$ before asserting $\overline{BGACK}$.
8. The minimum value must be met to guarantee proper operation. If the maximum value is exceeded, $\overline{BG}$ may be reasserted.
9. $\overline{AS}$ is always asserted, regardless of whether it is mapped to internal or external resources. If the designer wishes to decode more chip selects than are provided, use one of CS0–7 as the enable for the external decode.



NOTES:
1. Setup time (#47) for asynchronous inputs ($\overline{\text{HALT}}$, $\overline{\text{RESET}}$, $\overline{\text{BR}}$, $\overline{\text{BGACK}}$, $\overline{\text{DTACK}}$, $\overline{\text{BERR}}$, $\overline{\text{IRQx}}$) guarantees their recognition at the next falling edge of the clock.
2. $\overline{\text{BR}}$ need fall at this time only to ensure being recognized at the end of the bus cycle.

Figure 8-3. Read Cycle Timing Diagram



NOTES:

1. Setup time (#47) for asynchronous inputs ($\overline{\text{HALT}}$, $\overline{\text{RESET}}$, $\overline{\text{BR}}$, $\overline{\text{BGACK}}$, $\overline{\text{DTACK}}$, $\overline{\text{BERR}}$, $\overline{\text{IRQx}}$) guarantees their recognition at the next falling edge of the clock.
2. Because of loading variations, R/$\overline{\text{W}}$ may be valid after $\overline{\text{AS}}$ even though both are initiated by the rising edge of S2 (specification #20A).
3. $\overline{\text{BR}}$ need fall at this time only to ensure being recognized at the end of the bus cycle.

Figure 8-4. Write Cycle Timing Diagram

8.8 AC ELECTRICAL SPECIFICATIONS—CHIP SELECTS AND INTERRUPT ACKNOWLEDGE (The electrical specifications in this document are preliminary.)

		16.67 MHz		
Num	**Characteristic**	**Min**	**Max**	**Unit**
61	Address Valid to $\overline{CSx}$ Asserted (Read or Write)	15	—	ns
61A	FC Valid to $\overline{CSx}$ Asserted (Read or Write)	45	—	ns
62	$\overline{AS}$ **V** $\overline{DS}$ to $\overline{CSx}$	0	5	ns
63	$\overline{AS}$ **V** R/$\overline{W}$ to $\overline{CSx}$	0	5	ns
64	$\overline{CSx}$ Width Asserted	120	—	ns
65	$\overline{CSx}$ Negated to FC, Addess Invalid	15	—	ns
66	$\overline{CSx}$ Negated to R/$\overline{W}$ Invalid	15	—	ns
67	Data-Out Valid to $\overline{CSx}$ Negated (Write)	90	—	ns
68	$\overline{CSx}$ Negated to Data-Out Invalid (Write)	15	—	ns
69	$\overline{CSx}$ Negated to Data-In High Impedance	—	90	ns
70	CLKOUT High to $\overline{IACKx}$ Asserted	0	30	ns
70A	$\overline{LDS}$ High to $\overline{IACKx}$ Negated	0	10	ns

V = Boolean OR

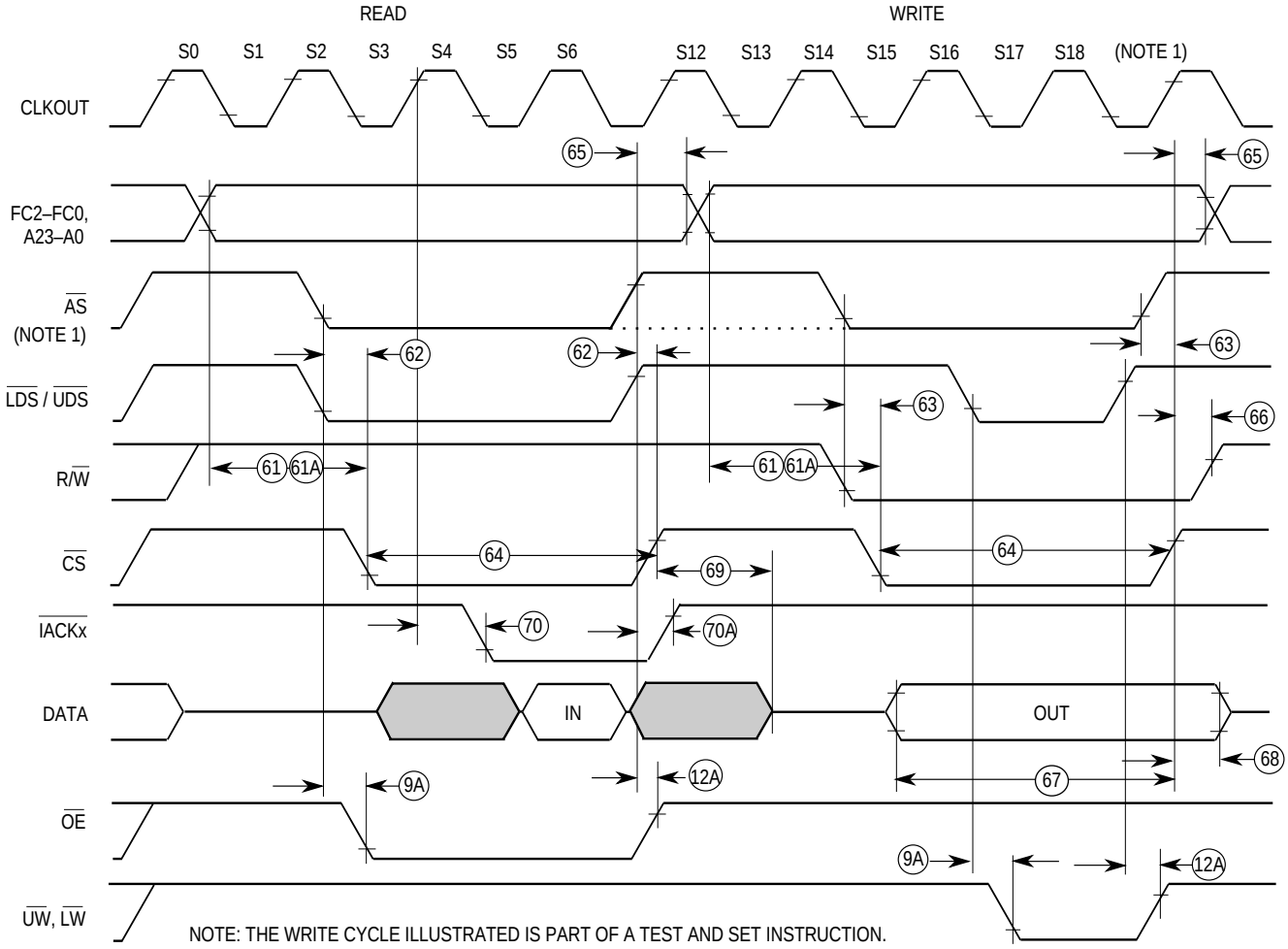


Figure 8-5. Chip Select and Interrupt Acknowledge Timing Diagram

8.9 AC ELECTRICAL SPECIFICATIONS—BUS ARBITRATION (The electrical specifications in this document are preliminary. See Figures 8-6–8-7)

		16.67 MHz		
Num	**Characteristic**	**Min**	**Max**	**Unit**
7	CLKOUT High to Address, Data Bus High Impedance (Maximum)	—	50	ns
16	CLKOUT High to Control Bus High Impedance	—	50	ns
33	CLKOUT High to $\overline{BG}$ Asserted	—	40	ns
34	CLKOUT High to $\overline{BG}$ Negated	—	40	ns
35	$\overline{BR}$ Asserted to $\overline{BG}$ Asserted	1.5	6.5	Clks
36	$\overline{BR}$ Negated to $\overline{BG}$ Negated	1.5	3.5	Clks
37	$\overline{BGACK}$ Asserted to $\overline{BG}$ Negated	1.5	3.5	Clks
37A[2]	$\overline{BGACK}$ Asserted to $\overline{BR}$ Negated	20 ns	1.5	Clks
38	$\overline{BG}$ Asserted to Control, Address, Data Bus High Impedance ($\overline{AS}$ Negated)	—	50	ns
39	$\overline{BG}$ Width Negated	1.5	—	Clks
47	Asynchronous Input Setup Time	10	—	ns
57	$\overline{BGACK}$ Negated to Bus Driven	1	—	Clks
58	$\overline{BR}$ Negated to Bus Driven	1	—	Clks

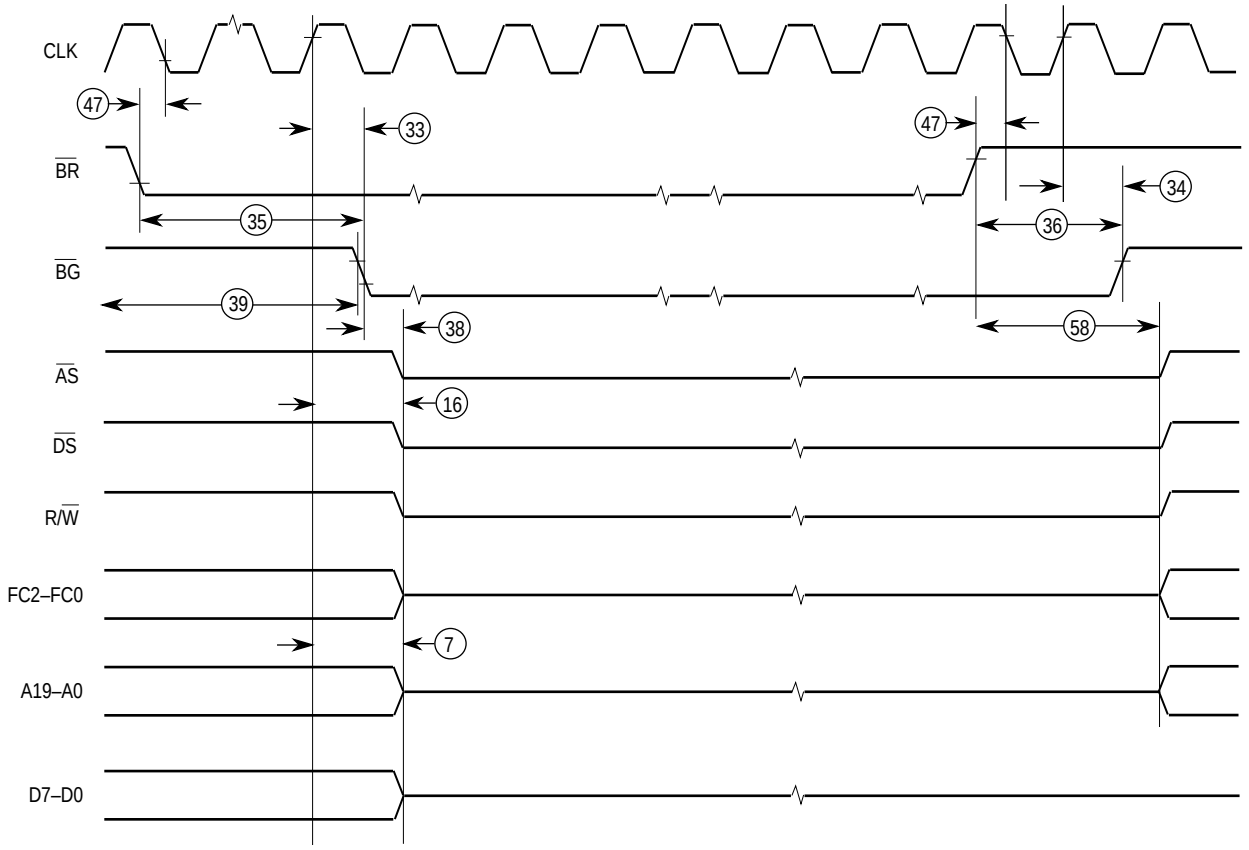
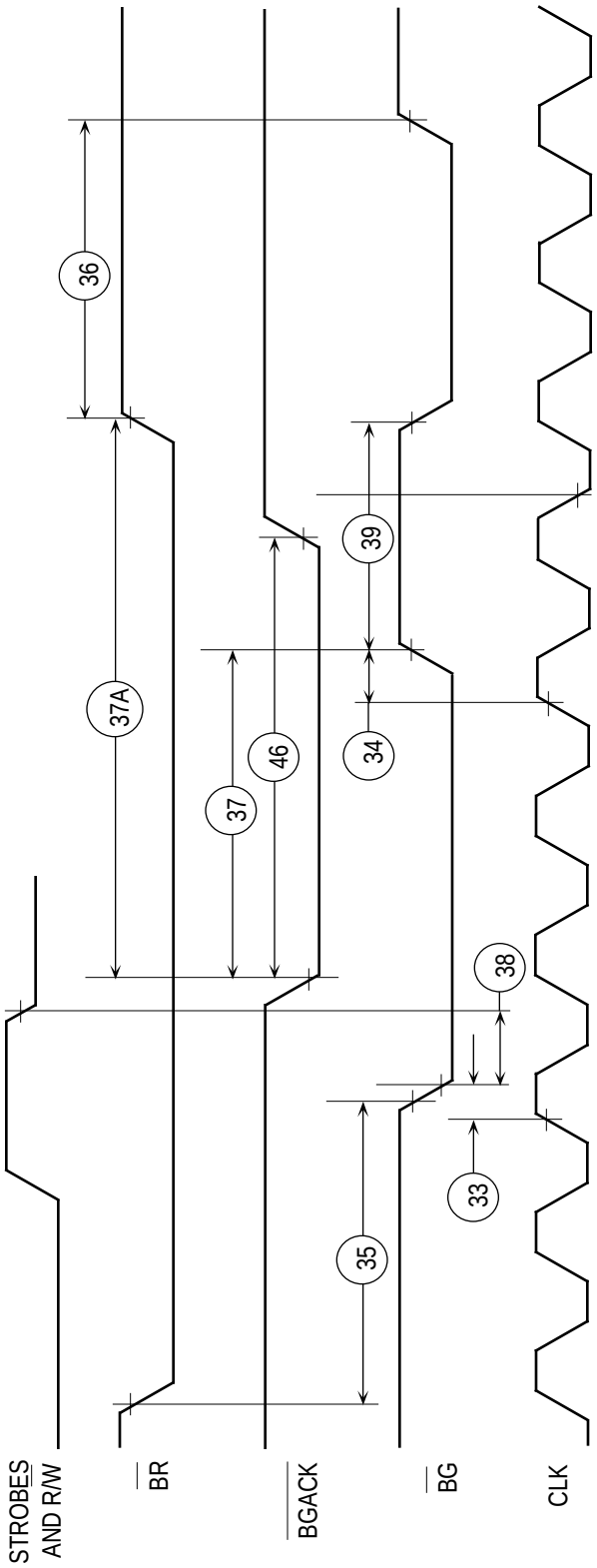


Figure 8-6. Bus Arbitration Timing Diagram



NOTE: Setup time to the clock (#47) for the asynchronous inputs $\overline{BERR}$, $\overline{BGACK}$, $\overline{BR}$, $\overline{DTACK}$, $\overline{HALT}$, $\overline{RESET}$, and $\overline{IRQx}$ guarantees their recognition at the next falling edge of the clock.

Figure 8-7. Bus Arbitration Timing Diagram

8.10 BUS OPERATION—DRAM ACCESSES AC TIMING SPECIFICATIONS (The electrical specifications in this document are preliminary. See Figures 8-8–8-11)

		16.67 MHz				
		0-Wait		1-Wait		
Num.	Characteristic	Min	Max	Min	Max	Unit
71	CLKOUT High to $\overline{RASx}$ Asserted (0 Wait State Operation)	0	30	–	–	ns
71A	$\overline{AS}$ Asserted to $\overline{RASx}$ Asserted (0 Wait State Operation)	0	10	–	–	ns
72	CLKOUT Low to $\overline{RASx}$ Asserted (1 Wait State Operation)	–	–	0	25	ns
73	CLKOUT Low to Row Address Valid	0	30	0	30	ns
74	Row Address Valid to $\overline{RASx}$ Asserted	15	–	30	–	ns
75	$\overline{RASx}$ Asserted to Row Address Invalid	20	–	40	–	ns
76	CLKOUT High to Row Address Invalid (0 Wait State Operation)	0	–	–	–	ns
77	CLKOUT Low to Row Address Invalid (1 Wait State Operation)	–	–	0	–	ns
78	$\overline{RASx}$ Width Asserted (Non-Page Mode)	120	180	150	210	ns
79	$\overline{RASx}$ Width Asserted (Page Mode)	480	540	510	570	ns
80	$\overline{RASx}$ Width Negated (Back to Back Cycles)	60	–	90	–	ns
81	$\overline{RASx}$ Asserted to $\overline{CASx}$ Asserted	45	–	60	–	ns
82	CLKOUT High to Column Address Valid (0 Wait State Operation)	0	30	–	–	ns
83	CLKOUT Low to Column Address Valid (1 Wait State Operation)	–	–	0	30	ns
84	CLKOUT Low to $\overline{CASx}$ Asserted (0 Wait State Operation)	0	20	–	–	ns
85	CLKOUT High to $\overline{CASx}$ Asserted (1 Wait State Operation)	–	–	0	20	ns
86	Column Address Valid to $\overline{CASx}$ Asserted	20	–	20	–	ns
87	$\overline{CASx}$ Asserted to Column Address Invalid	75	–	100	–	ns
88	$\overline{CASx}$ Width Asserted	60	90	90	120	ns
89	CLKOUT Low to $\overline{RASx}$ /$\overline{CASx}$ Negated	0	30	0	30	ns
89A	$\overline{AS}$ Negated to $\overline{RASx}$ /$\overline{CASx}$ Negated	0	10	0	10	ns
90	$\overline{CASx}$ Width Negated (Back to Back Cycles)	150	–	180	–	ns
91	$\overline{CASx}$ Width Negated (Page Mode[2])	240	300	240	300	ns
92	$\overline{UDS}$/$\overline{LDS}$ Asserted to $\overline{CASx}$ Asserted[1] (Page Mode[2])	0	10	0	10	ns
93	$\overline{DRAMW}$ Low to $\overline{CASx}$ Asserted (Write)	30	–	60	–	ns
94	Data Out Valid to $\overline{CASx}$ Asserted (Write)	15	–	45	–	ns
95	CLKOUT Low to $\overline{CASx}$ Asserted (Refresh Cycle)	0	20	0	20	ns
96	CLKOUT High to $\overline{CASx}$ Negated (Refresh Cycle)	0	20	0	20	ns
97	$\overline{CASx}$ Width Asserted (Refresh Cycle)	80	120	140	180	ns
98	$\overline{CASx}$ Asserted to $\overline{RASx}$ Asserted (Refresh Cycle)	20	60	20	60	ns
99	CLKOUT High to $\overline{RASx}$ Asserted (Refresh Cycle)	0	30	0	30	ns
100	CLKOUT Low to $\overline{RASx}$ Negated (Refresh Cycle)	0	25	0	25	ns
101	$\overline{RASx}$ Width Asserted (Refresh Cycle)	80	120	140	180	ns
102	$\overline{DRAMW}$ High to $\overline{RASx}$ Asserted (Refresh Cycle)	20	60	20	60	ns
103	$\overline{DRAMW}$ High Hold After $\overline{RASx}$ Asserted (Refresh Cycle)	20	–	20	–	ns

NOTES:
1. On write portion of TAS, $\overline{CAS}$ assertion is gated by $\overline{UDS}$/$\overline{LDS}$ (not CLKOUT as in all other operation).
2. Page mode is used on Read-Modify-Write (TAS instruction) cycles only.

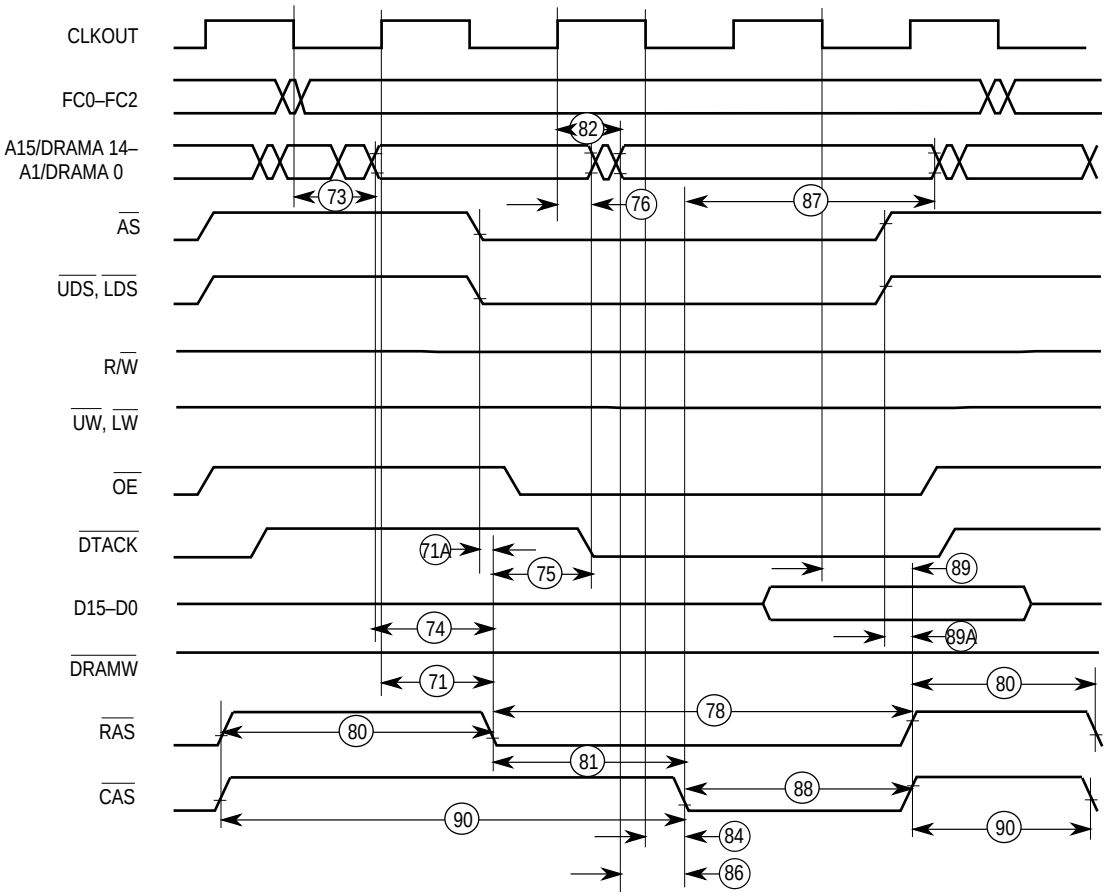


Figure 8-8. DRAM Timing – 0-Wait Read, No Refresh

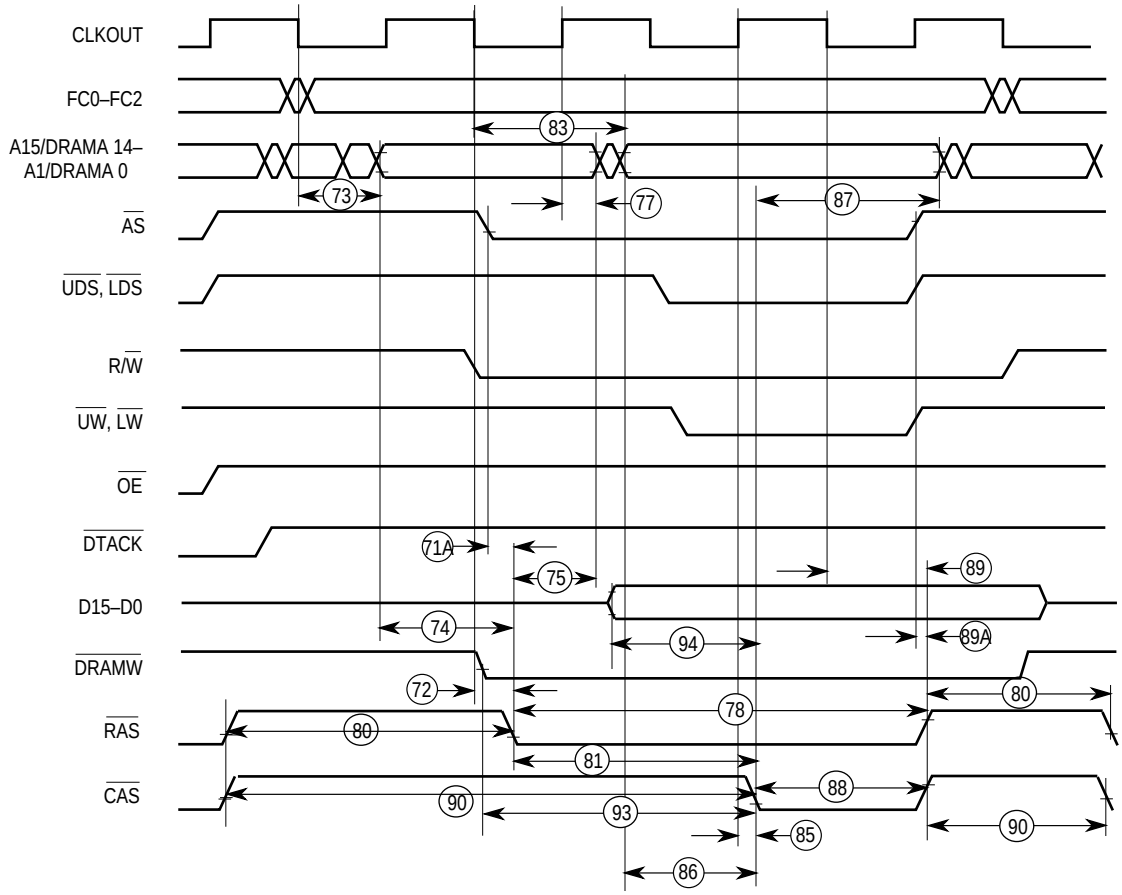


Figure 8-9. DRAM Timing – 1-Wait Write, No Refresh

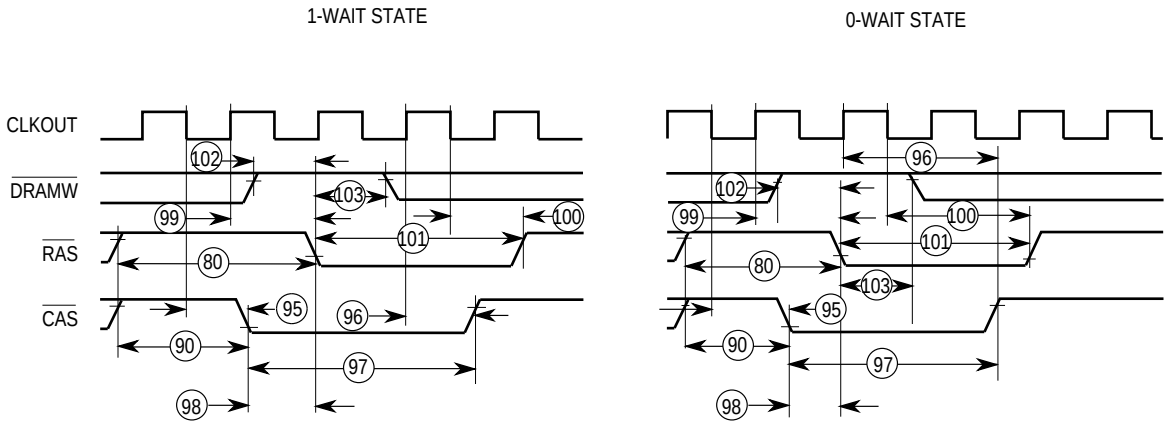
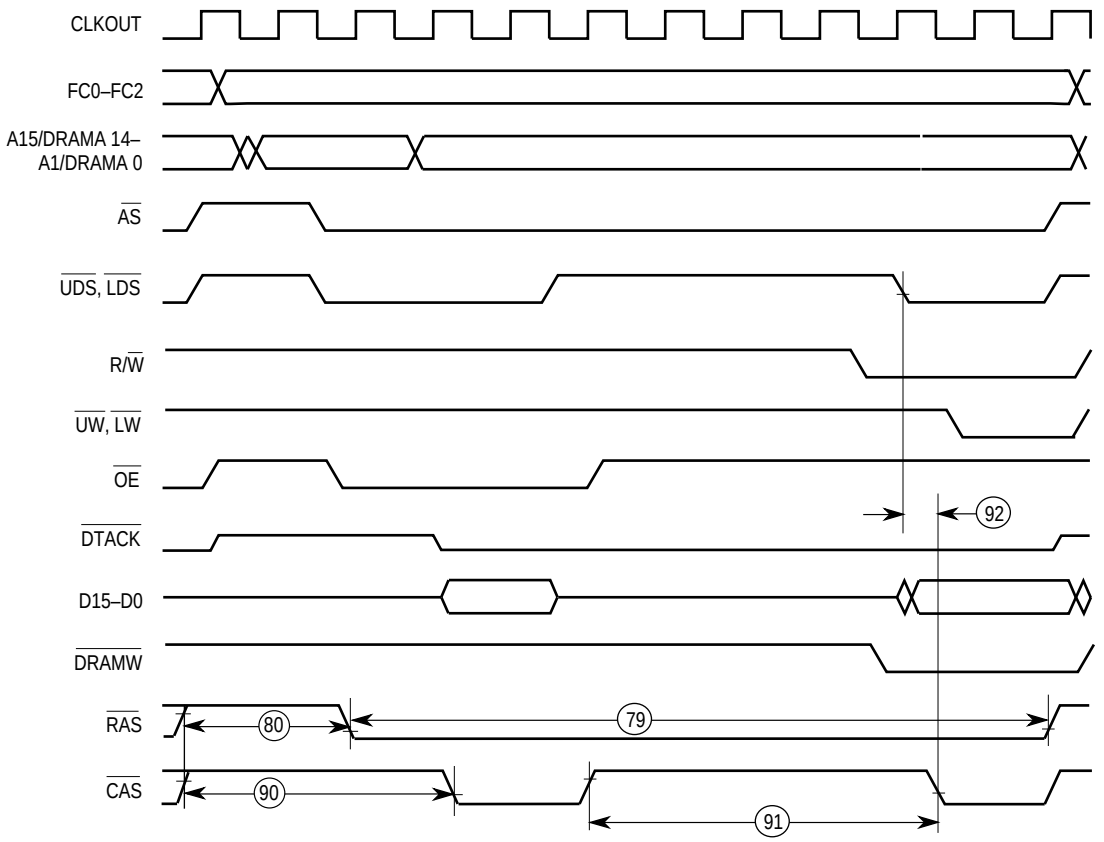


Figure 8-10. DRAM Timing – 0- and 1-Wait Refresh



* NOTE: TAS IS A BYTE-ONLY INSTRUCTION, THEREFORE ONLY ONE OF UW, LW AND ONLY ONE CAS WILL BE ASSERTED.

Figure 8-11. DRAM Timing – 1-Wait, Test and Set

8.11 SERIAL MODULE ELECTRICAL CHARACTERISTICS

(T_A = 0 °C to 70 °C, V_{CC} = 5.0 V ±5%, See Note 1)

Characteristic	Symbol	Min	Max	Unit
X1/CLK Input Leakage Current	I_{X1L}		2.5	µA
X1/CLK Frequency (see Note 2)	f_{CLK}	2.0	4.0	MHz
Counter/Timer Clock Frequency (IP2)	f_{CTC}	0	16.67	MHz

NOTES:

1. All voltage measurements are referenced to ground (GND). For testing, all input signals except X1/CLK swing between 0.4 V and 2.4 V with a maximum transition time of 20 ns. For X1/CLK, this swing is between 0.4 V and 4.4 V. All time measurements are referenced at input and output voltages of 0.8 V and 2.0 V as appropriate. Test conditions for outputs: C_L = 150 pF, R_L = 750 Ω to V_{CC}.
2. To use the standard baud rates selected by the clock-select register given in Tables 6-5 and 6-6, the X1/CLK frequency should be set to 3.6864 MHz or a 3.6864 MHz crystal should be connected across pins X1/CLK and X2.
3. IP5–2 for RxC, TxC are not supported in the MC68306.

8.12 SERIAL MODULE AC ELECTRICAL CHARACTERISTICS—CLOCK TIMING (See Figure 8-12.)

Characteristic	Symbol	Min	Max	Unit
Counter/Timer Clock High or Low Time	t_{CTC}	25	—	ns
Clock Rise Time	t_r	—	20	ns
Clock Fall Time	t_f	—	20	ns

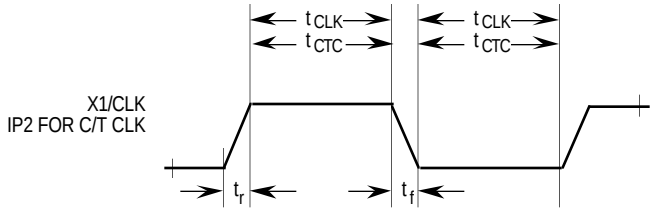


Figure 8-12. Clock Timing

8.13 AC ELECTRICAL CHARACTERISTICS—PORT TIMING

(See Figure 8-13 and Note.)

Characteristic	Symbol	Min	Max	Unit
Port Input Setup Time to $\overline{LDS}$ Asserted	t_{PS}	0	—	ns
Port Input Hold Time from $\overline{LDS}$ Negated	t_{PH}	0	—	ns
Port Output Valid from $\overline{LDS}$ Negated	t_{PD}	—	60	ns

NOTE: Test conditions for port outputs: C_L = 50 pF, R_L = 27 kΩ to V_{CC}.

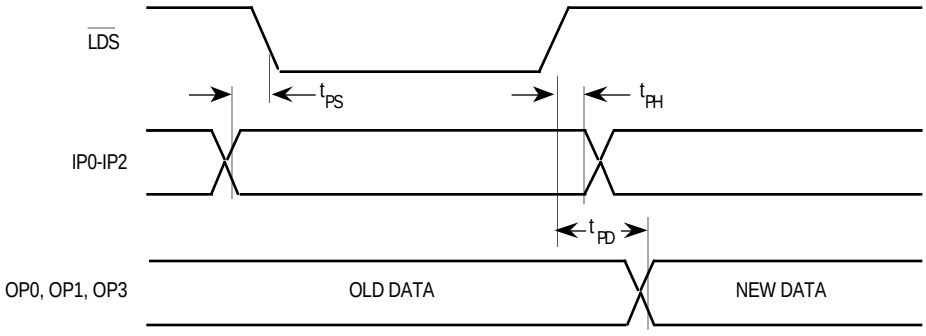
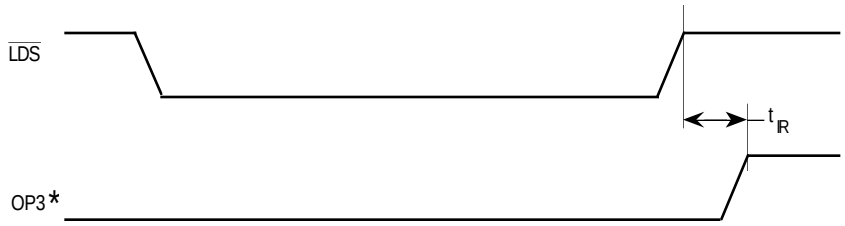


Figure 8-13. Port Timing

8.14 AC ELECTRICAL CHARACTERISTICS—INTERRUPT RESET TIMING (See Figure 8-14 and Note)

Characteristic	Symbol	Min	Max	Unit
OP3 High (When Used as Counter Interrupt) from $\overline{LDS}$ Negated After Stop Counter Command	t_{IR}	—	100	ns

NOTE: Test conditions for interrupt output: C_L = 50 pF, R_L = 2 kΩ to V_{CC}.



* When used as counter interrupt output.

Figure 8-14. Interrupt Reset Timing

8.15 AC ELECTRICAL CHARACTERISTICS—TRANSMITTER TIMING

(See Figure 8-15 and Note)

Characteristic	Symbol	Min	Max	Unit
TxD Output Valid from TxC Low	t_{TxD}	—	100	ns
CTS Input Setup to Tx Clock High *	t_{CS}	30	—	ns
CTS Input Hold from Tx Clock High *	t_{CH}	30	—	ns
RTS Output Valid from Tx Clock	t_{TRD}	—	100	ns

* CTS is an asynchronous input. This specification is only provided to guarantee CTS recognition on a particular Tx clock edge.

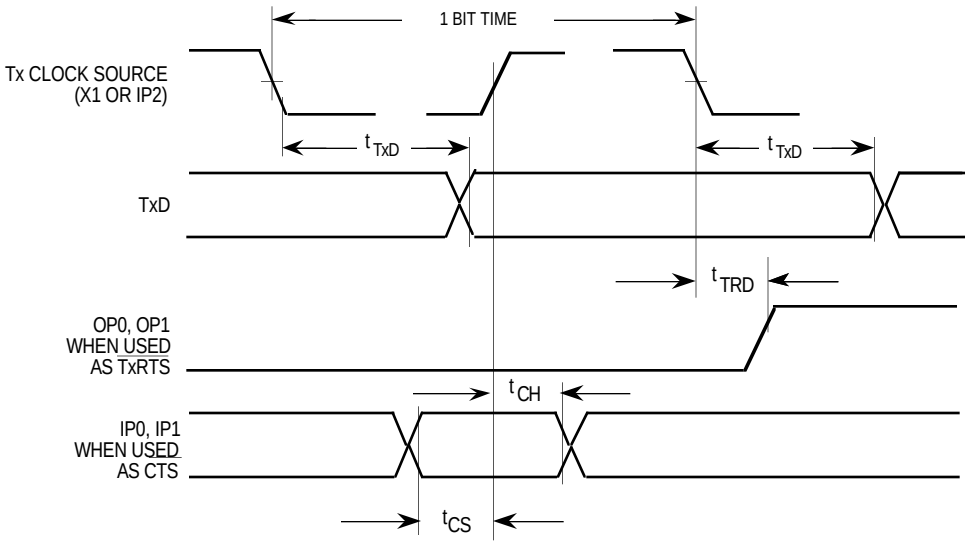


Figure 8-15. Transmit Timing

8.16 AC ELECTRICAL CHARACTERISTICS—RECEIVER TIMING

(See Figure 8-16 and Note)

Characteristic	Symbol	Min	Max	Unit
RxD Data Setup Time to RxC High	t_{RxS}	240	—	ns
RxD Data Hold Time from RxC High	t_{RxH}	200	—	ns
RTS Output Valid from Rx Clock	t_{RRD}	—	100	ns

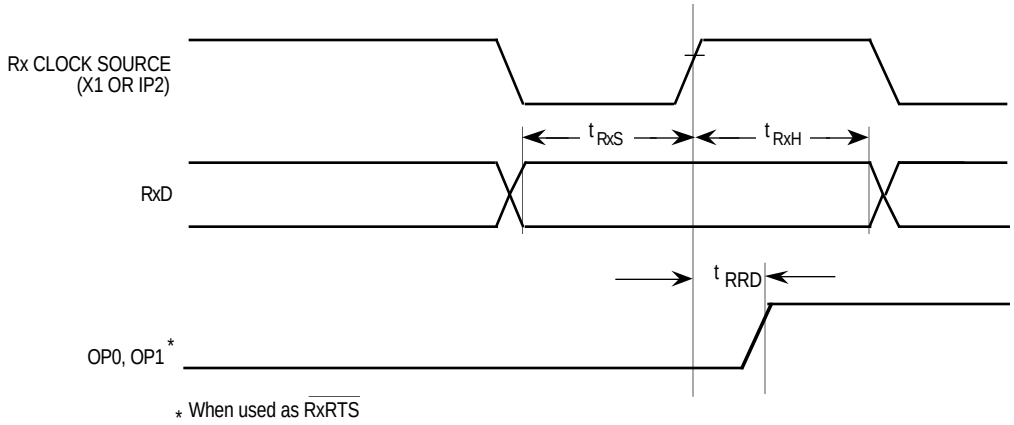


Figure 8-16. Receive Timing

8.17 IEEE 1149.1 ELECTRICAL CHARACTERISTICS

(The electrical specifications in this document are preliminary; see Figures 8-17–8-19.)

Num.	Characteristic	Min	Max	Unit
	TCK Frequency of Operation	0	10.0	MHz
1	TCK Cycle Time	100	—	ns
2	TCK Clock Pulse Width Measured at 1.5 V	45	—	ns
3	TCK Rise and Fall Times	0	5	ns
6	Boundary Scan Input Data Setup Time	15	—	ns
7	Boundary Scan Input Data Hold Time	15	—	ns
8	TCK Low to Output Data Valid	0	80	ns
9	TCK Low to Output High Impedance	0	80	ns
10	TMS, TDI Data Setup Time	15	—	ns
11	TMS, TDI Data Hold Time	15	—	ns
12	TCK Low to TDO Data Valid	0	30	ns
13	TCK Low to TDO High Impedance	0	30	ns
14	$\overline{\text{TRST}}$ Width Low	80	—	ns

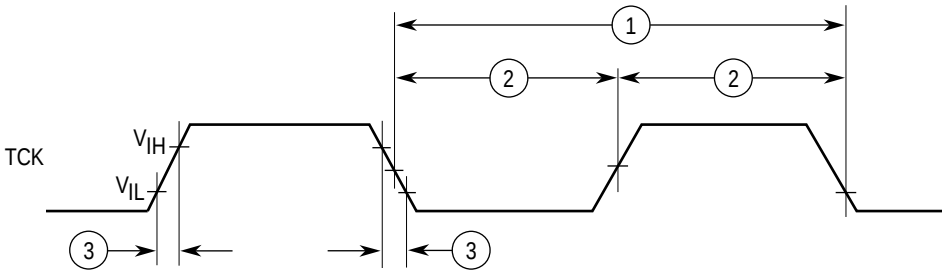


Figure 8-17. Test Clock Input Timing Diagram

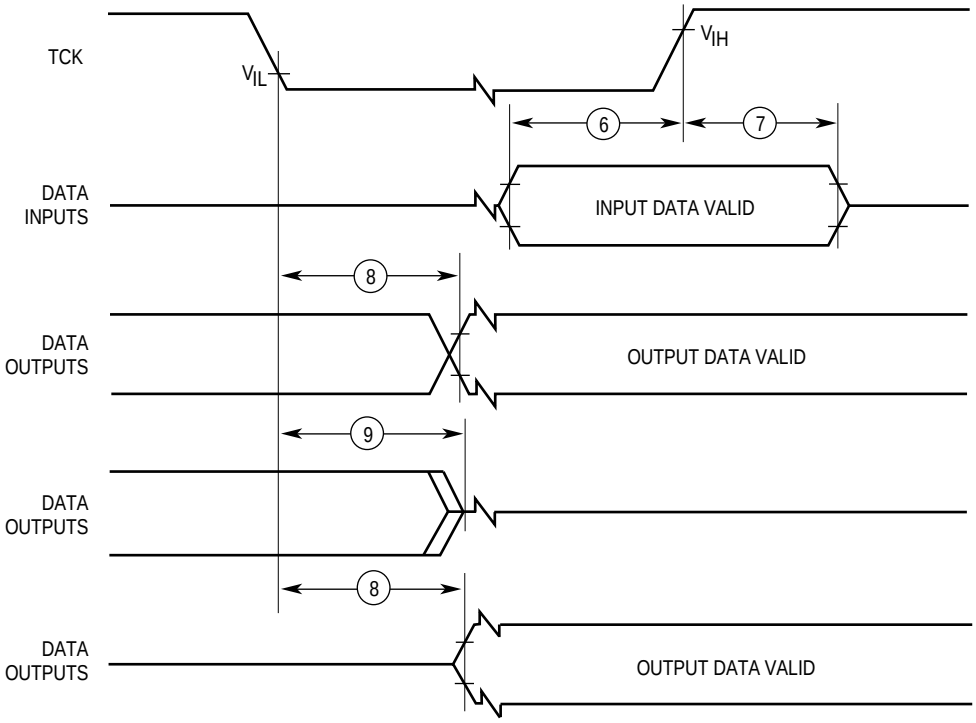


Figure 8-18. Boundary Scan Timing Diagram

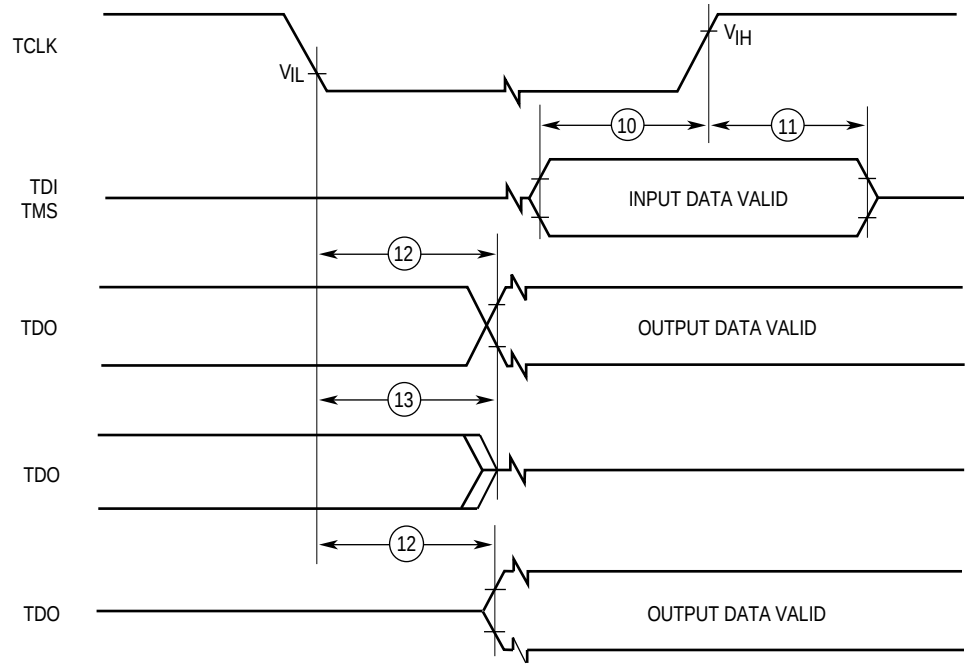


Figure 8-19. Test Access Port Timing Diagram

SECTION 9
ORDERING INFORMATION AND MECHANICAL DATA

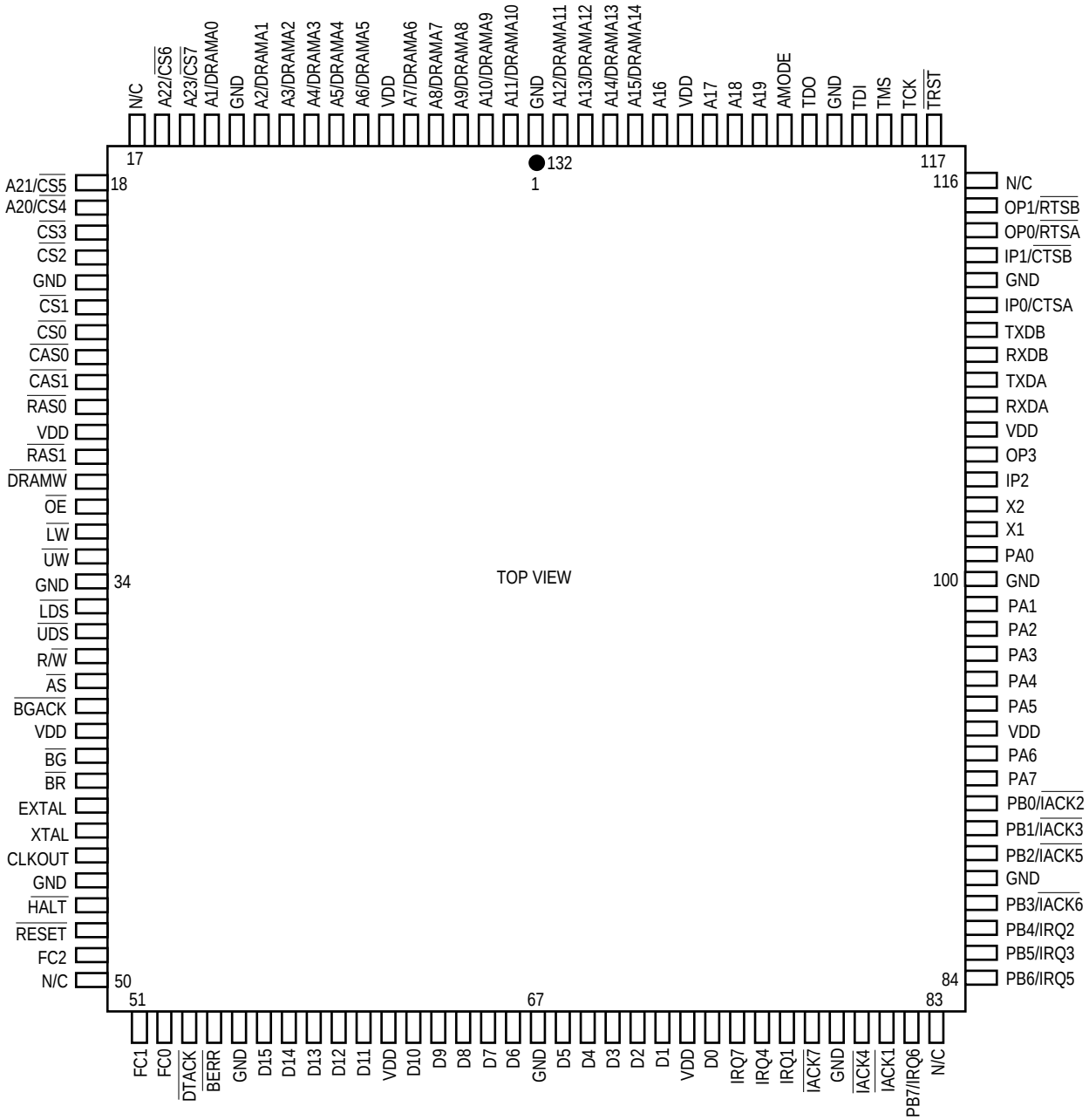
This section contains the ordering information, pin assignments, and package dimensions for the MC68306.

9.1 STANDARD ORDERING INFORMATION

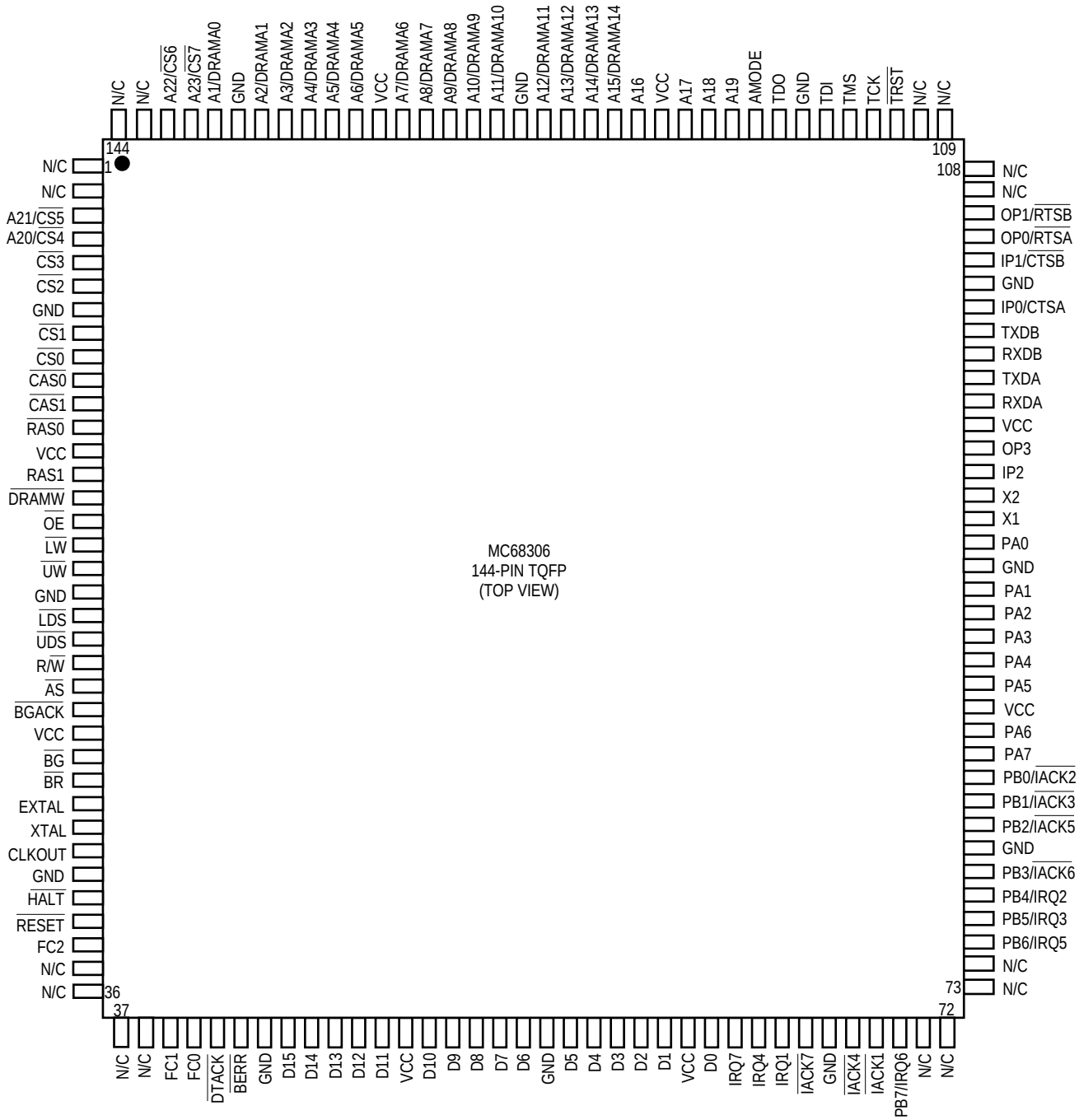
Package Type	Frequency (MHz)	Temperature	Order Number
132-Lead Plastic Quad Flat Pack (FC Suffix)	8–16.7	0°C to 70°C	MC68306FC16
144-Lead Thin Quad Flat Pack (PV Suffix)	8–16.7	0°C to 70°C	MC68306PV16

9.2 PIN ASSIGNMENTS

132-Lead Plastic Quad Flat Pack (PQFP)



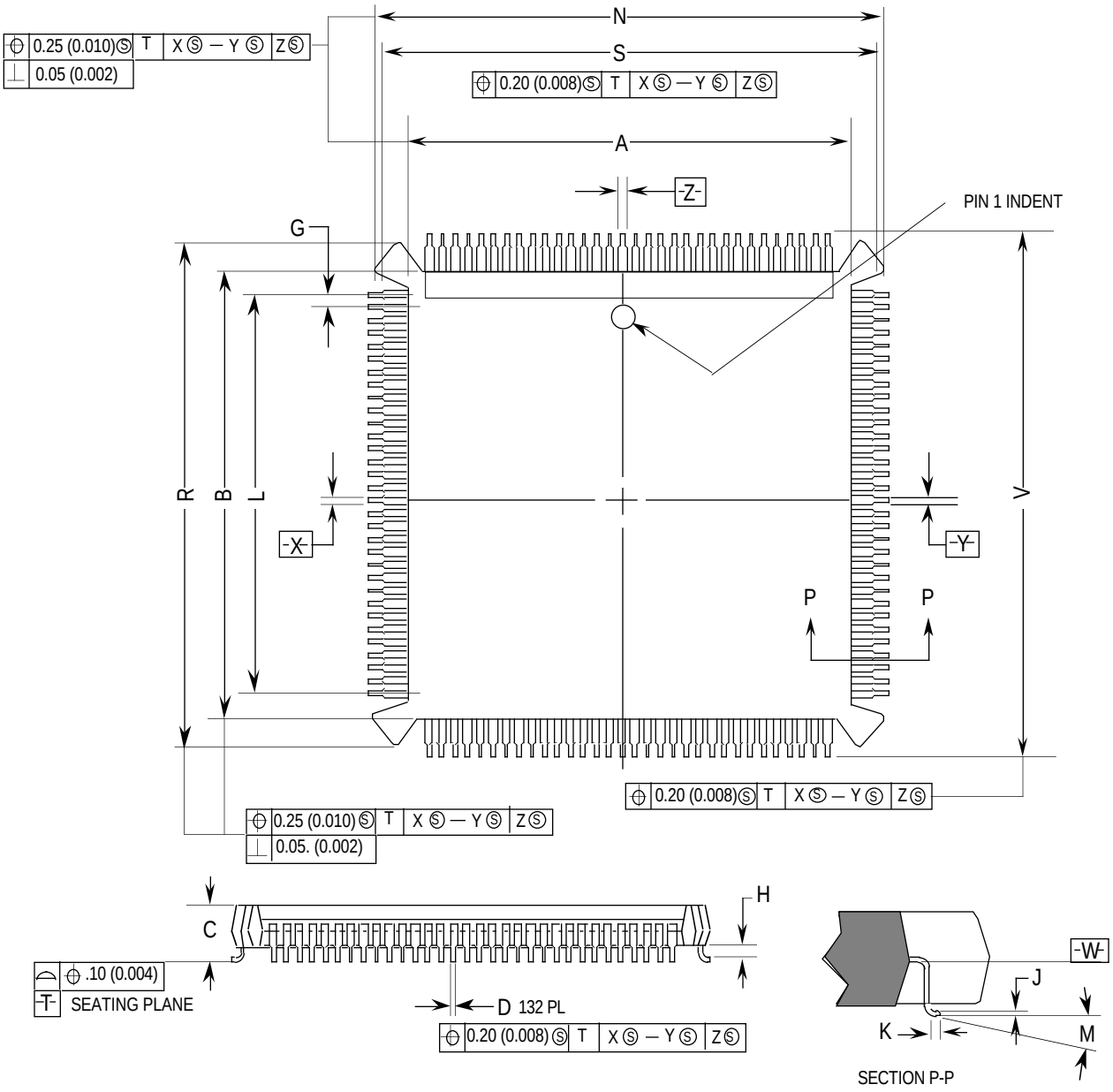
144-Lead Thin Quad Flat Pack (TQFP)



9.3 PACKAGE DIMENSIONS

132 Pin PQFP (FC Suffix)

CASE 831A-01



	MILLIMETERS		INCHES	
DIM	MIN	MAX	MIN	MAX
A	24.06	24.20	0.947	0.953
B	24.06	24.20	0.947	0.953
C	4.07	4.57	0.160	0.180
D	0.21	0.30	0.008	0.012
G	0.64 BSC		0.025 BSC	
H	0.51	1.01	0.020	0.040
J	0.16	0.20	0.006	0.008
K	0.51	0.76	0.020	0.030
L	20.32 REF		0.800 REF	
M	0°	8°	0°	8°
N	27.88	28.01	1.097	1.103
R	27.88	28.01	1.097	1.103
S	27.31	27.55	1.075	1.085
V	27.31	27.55	1.075	1.085

NOTES:

1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: INCH
3. DIMENSIONS A, B, N, AND R DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE MOLD PROTRUSION FOR DIMENSIONS A AND B IS 0.25 (0.010), FOR DIMENSIONS N AND R IS 0.18 (0.007).
4. DATUM PLANE **-W-** IS LOCATED AT THE UNDERSIDE OF LEADS WHERE LEADS EXIT PACKAGE BODY.
5. DATUMS **-X-** , **-Y-**, AND **-Z-** TO BE DETERMINED WHERE CENTER LEADS EXIT PACKAGE BODY AT DATUM **-W-** .
6. DIMENSIONS S AND V TO BE DETERMINED AT SEATING PLANE, DATUM **-T-**.
7. DIMENSIONS A, B, N AND R TO BE DETERMINED AT DATUM PLANE **-W-** .

144-Lead Thin Quad Flat Pack (PV Suffix)

INDEX

— A —

Access Error, 4-1
    Exception, 4-21
Addressing Modes, 4-4
    Index Sizing and Scaling, 4-4
    Indexing, 4-4
    Postincrement, Predecrement, Offset, and
        Program Counter Indirect, 4-4
    Register Indirect, 4-4
AS, 3-4, 3-7, 3-16
Asynchronous Bus Arbitration Signals, 3-17
Asynchronous Mode, 3-32
Autovector, 5-5
Automatic DTACK Generation, 5-16

— B —

Baud Rate Generator, 6-3, 6-4, 6-7
BERR, 3-4, 3-7, 3-10, 3-37
BG, 3-16
Boundary Scan Bit Definitions, 7-5
Boundary Scan, 7-1
    Bit Definitions, 7-5
Bus Arbitration, 3-12
Bus Error Exception, 3-28, 4-20
Bus Grant Signal, 3-16
Bus Timeout Period Register, 5-4
Byte Read Cycle Flowchart, 3-2

— C —

Chip Select Configuration Register, 5-9
Counter Mode, 6-16
Counter/Timer, 6-16
CTLR, 6-34
CTUR, 6-34

— D —

Data Formats, 4-3
Data Types
    Access Errors, 4-1
        M-bit, 4-14
    Denormalized Numbers, 4-3
    Infinities, 4-3
    NANs, 4-3
    Normalized Numbers, 4-3
    Zeros, 4-3
Double Bus Fault, 3-29
DRAM
    Configuration Register, 5-14
    Refresh Register, 5-13
DTACK, 3-4, 3-7, 3-10, 3-33, 3-37
DUACR, 6-30
DUCR, 6-26
DUCSR, 6-24
DUCUR, 6-33
DUIMR, 6-33
DUIP, 6-34
DUIPCR, 6-29
DUISR, 6-31
DUIVR, 6-34
DUMR1, 6-18
DUMR2, 6-20
DUOP, 6-35
DUOPCR, 6-35
DURBA, 6-29
DURBB, 6-29
DUSR, 6-22
DUTBA, 6-29
DUTBB, 6-29

— E —

Exception Handler, 4-14
Exceptions, 4-12
Exception Vector, 4-14
    Table, 4-12

— F —

FC2–FC0, 3-4, 3-7
FIFO Stack, 6-11

— H —

HALT, 3-28

— I —

I/O Driver Routines, 6-37
Initialization Routines, 6-36
Instructions
    STOP, 4-1
    TRAP, TRAPV, CHK, RTE, and DIV, 4-12
Interrupt, 4-1
    Acknowledge Bus Cycle, 4-12
    Control Register, 5-5
    Handling Routine, 6-37
    Priorities, 4-17
    Priority Mask, 4-12
    Request Signals, 6-3
    Request, 4-17
    Status Register, 5-6
    Priority Mask, 4-12

— J —

JTAG, 7-1

— L —

LDS, 3-7
Level 7 Interrupts, 4-17
Looping Modes, 6-13

— N —

Non-IEEE 1149.1 Operation, 7-12

— O —

Operand Size, 4-3
Oscillator Circuit, 5-16

— P —

Package Dimensions, 9-1
Pin Assignments, 9-1
Port Data Register, 5-8
Port Direction Register, 5-7
Port Pin Register, 5-7
Privileged Instructions, 4-19
Processing States
    Normal, Exception, Halted, 4-1

— R —

R/$\overline{W}$, 3-4, 3-7
Read Cycle, 3-1
Read-Modify-Write Cycle, 3-7
Reset Exception, 4-17
Retry Operation, 3-28

— S —

Serial Module Counter/Timer Interrupt, 6-4
Single Step, 3-28
Stack Frame, 4-14
Status Register, 4-12
System Register, 5-3

— T —

TAP, 7-1
TAS, 3-7
Three-Wire Bus Arbitration, 3-12
Timer Mode, 6-16
Timer Vector Register, 5-4
Timer/Counter, 6-3
Trace Exception, 4-19
Two-Wire Bus Arbitration, 3-12

— U —

$\overline{UDS}$, 3-7
$\overline{UDS}$/$\overline{LDS}$, 3-4, 3-10
Uninitialized Interrupt Vector, 4-18

— V —

Valid Start Bit, 6-10
Vector Number, 4-12

— W —

Word Read Cycle Flowchart, 3-2
Write Cycle, 3-4