MOTOROLA

Microprocessors and Memory
Technologies Group

MC68302

Integrated Multiprotocol Processor
User's Manual

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

© 1995 Motorola, Inc. All Rights Reserved

PREFACE

The complete documentation package for the MC68302 consists of the M68000PM/AD, *MC68000 Family Programmer's Reference Manual,* MC68302UM/AD, *MC68302 Integrated Multiprotocol Processor User's Manual,* and the MC68302/D, *MC68302 Integrated Multiprotocol Processor Product Brief*.

The *MC68302 Integrated Multiprotocol Processor User's Manual* describes the programming, capabilities, registers, and operation of the MC68302; the *MC68000 Family Programmer's Reference Manual* provides instruction details for the MC68302; and the *MC68302 Low Power Integrated Multiprotocol Processor Product Brief* provides a brief description of the MC68302 capabilities.

This user's manual is organized as follows:

Section 1	General Description
Section 2	MC68000/MC68008 Core
Section 3	System Integration Block (SIB)
Section 4	Communications Processor (CP)
Section 5	Signal Description
Section 6	Electrical Characteristics
Section 7	Mechanical Data And Ordering Information
Appendix B	Development Tools and Support
Appendix C	RISC Microcode from RAM
Appendix D	MC68302 Applications
Appendix E	SCC Programming Reference
Appendix F	Design Checklist

ELECTRONIC SUPPORT:

The Technical Support BBS, known as AESOP (Application Engineering Support Through On-Line Productivity), can be reach by modem or the internet. AESOP provides commonly asked application questons, latest device errata, device specs, software code, and many other useful support functions.

Modem: Call 1-800-843-3451 (outside US or Canada 512-891-3650) on a modem that runs at 14,400 bps or slower. Set your software to N/8/1/F emulating a vt100.

Internet: This access is provided by telneting to pirs.aus.sps.mot.com [129.38.233.1] or through the World Wide Web at http://pirs.aus.sps.mot.com.

— Sales Offices —

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

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 Lauderdal	(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

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, Inc.	(505) 298-7177
UTAH, Salt Lake City	
Utah Component Sales, Inc.	(801) 561-5099
WASHINGTON, Spokane	
Doug Kenley	(509) 924-2322
ARGENTINA, Buenos Aires	
Argonics, S.A.	(541) 343-1787

HYBRID COMPONENTS RESELLERS

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

TABLE OF CONTENTS

Paragraph Number	Title	Page Number
	Section 1 General Description	
1.1	Block Diagram	1-1
1.2	Features	1-3
1.3	MC68302 System Architecture	1-4
1.4	NMSI Communications-Oriented Environment	1-5
1.5	Basic Rate ISDN or Digital Voice/Data Terminal	1-6
	Section 2 MC68000/MC68008 Core	
2.1	Programming Model	2-1
2.2	Instruction Set Summary	2-3
2.3	Address Spaces	2-6
2.4	Exception Processing	2-8
2.4.1	Exception Vectors	2-8
2.4.2	Exception Stacking Order	2-9
2.5	Interrupt Processing	2-11
2.6	M68000 Signal Differences	2-11
2.7	MC68302 IMP Configuration Control	2-12
2.8	MC68302 Memory Map	2-14
2.9	Event Registers	2-19
	Section 3 System Integration Block (SIB)	
3.1	DMA Control	3-2
3.1.1	Key Features	3-2
3.1.2	IDMA Registers (Independent DMA Controller)	3-3
3.1.2.1	Channel Mode Register (CMR)	3-4
3.1.2.2	Source Address Pointer Register (SAPR)	3-6
3.1.2.3	Destination Address Pointer Register (DAPR)	3-6
3.1.2.4	Function Code Register (FCR)	3-7
3.1.2.5	Byte Count Register (BCR)	3-7
3.1.2.6	Channel Status Register (CSR)	3-7
3.1.3	Interface Signals	3-8
3.1.3.1	$\overline{\text{DREQ}}$ and $\overline{\text{DACK}}$	3-8
3.1.3.2	$\overline{\text{DONE}}$	3-8
3.1.4	IDMA Operational Description	3-9
3.1.4.1	Channel Initialization	3-9
3.1.4.2	Data Transfer	3-9

Paragraph Number	Title	Page Number
3.1.4.3	Address Sequencing	3-10
3.1.4.4	Transfer Request Generation	3-11
3.1.4.5	Block Transfer Termination	3-12
3.1.5	IDMA Programming	3-13
3.1.6	DMA Bus Arbitration	3-14
3.1.7	Bus Exceptions	3-14
3.1.7.1	Reset	3-15
3.1.7.2	Bus Error	3-15
3.1.7.3	Halt	3-15
3.1.7.4	Relinquish and Retry	3-15
3.2	Interrupt Controller	3-15
3.2.1	Overview	3-16
3.2.1.1	IMP Interrupt Processing Overview	3-16
3.2.1.2	Interrupt Controller Overview	3-17
3.2.2	Interrupt Priorities	3-18
3.2.2.1	INRQ and EXRQ Priority Levels	3-18
3.2.2.2	INRQ Interrupt Source Priorities	3-19
3.2.2.3	Nested Interrupts	3-19
3.2.3	Masking Interrupt Sources and Events	3-20
3.2.4	Interrupt Vector	3-21
3.2.5	Interrupt Controller Programming Model	3-24
3.2.5.1	Global Interrupt Mode Register (GIMR)	3-24
3.2.5.2	Interrupt Pending Register (IPR)	3-26
3.2.5.3	Interrupt Mask Register (IMR)	3-27
3.2.5.4	Interrupt In-Service Register (ISR)	3-28
3.2.6	Interrupt Handler Examples	3-28
3.3	Parallel I/O Ports	3-29
3.3.1	Port A	3-29
3.3.2	Port B	3-31
3.3.2.1	PB7–PB0	3-31
3.3.2.2	PB11–PB8	3-32
3.3.3	I/O Port Registers	3-32
3.4	Dual-Port RAM	3-33
3.5	Timers	3-35
3.5.1	Timer Key Features	3-36
3.5.2	General Purpose Timer Units	3-37
3.5.2.1	Timer Mode Register (TMR1, TMR2)	3-37
3.5.2.2	Timer Reference Registers (TRR1, TRR2)	3-38
3.5.2.3	Timer Capture Registers (TCR1, TCR2)	3-39
3.5.2.4	Timer Counter (TCN1, TCN2)	3-39
3.5.2.5	Timer Event Registers (TER1, TER2)	3-39
3.5.2.6	General Purpose Timer Example	3-40
3.5.2.6.1	Timer Example 1	3-40
3.5.2.6.2	Timer Example 2	3-40
3.5.3	Timer 3 - Software Watchdog Timer	3-41

Paragraph Number	Title	Page Number
3.5.3.1	Software Watchdog Timer Operation	3-41
3.5.3.2	Software Watchdog Reference Register (WRR)	3-41
3.5.3.3	Software Watchdog Counter (WCN)	3-42
3.6	External Chip-Select Signals and Wait-State Logic	3-42
3.6.1	Chip-Select Logic Key Features	3-45
3.6.2	Chip-Select Registers	3-45
3.6.2.1	Base Register (BR3–BR0)	3-45
3.6.2.2	Option Registers (OR3–OR0)	3-47
3.6.3	Chip Select Example	3-48
3.7	On-Chip Clock Generator	3-49
3.8	System Control	3-50
3.8.1	System Control Register (SCR)	3-50
3.8.2	System Status Bits	3-51
3.8.3	System Control Bits	3-52
3.8.4	Disable CPU Logic (M68000)	3-54
3.8.5	Bus Arbitration Logic	3-56
3.8.5.1	Internal Bus Arbitration	3-56
3.8.5.2	External Bus Arbitration	3-58
3.8.6	Hardware Watchdog	3-59
3.8.7	Reducing Power Consumption	3-60
3.8.7.1	Power-Saving Tips	3-60
3.8.7.2	Low-Power (Standby) Modes	3-60
3.8.7.2.1	Low-Power Mode	3-61
3.8.7.2.2	Lowest Power Mode	3-62
3.8.7.2.3	Lowest Power Mode with External Clock	3-62
3.9	Clock Control Register	3-64
3.9.1	Freeze Control	3-65
3.10	Dynamic Ram Refresh Controller	3-66
3.10.1	Hardware Setup	3-66
3.10.2	DRAM Refresh Controller Bus Timing	3-67
3.10.3	Refresh Request Calculations	3-67
3.10.4	Initialization	3-68
3.10.5	DRAM Refresh Memory Map	3-68
3.10.6	Programming Example	3-69

Section 4
Communications Processor (CP)

4.1	Main Controller	4-1
4.2	SDMA Channels	4-3
4.3	Command Set	4-5
4.3.1	Command Execution Latency	4-7
4.4	Serial Channels Physical Interface	4-7
4.4.1	IDL Interface	4-11
4.4.2	GCI Interface	4-14
4.4.3	PCM Highway Mode	4-16
4.4.4	Nonmultiplexed Serial Interface (NMSI)	4-19

Paragraph Number	Title	Page Number
4.4.5	Serial Interface Registers	4-19
4.4.5.1	Serial Interface Mode Register (SIMODE)	4-19
4.4.5.2	Serial Interface Mask Register (SIMASK)	4-22
4.5	Serial Communication Controllers (SCCs)	4-22
4.5.1	SCC Features	4-24
4.5.2	SCC Configuration Register (SCON)	4-24
4.5.2.1	Asynchronous Baud Rate Generator Examples	4-26
4.5.2.2	Synchronous Baud Rate Generator Examples	4-27
4.5.3	SCC Mode Register (SCM)	4-27
4.5.4	SCC Data Synchronization Register (DSR)	4-31
4.5.5	Buffer Descriptors Table	4-32
4.5.6	SCC Parameter RAM Memory Map	4-34
4.5.6.1	Data Buffer Function Code Register (TFCR, RFCR)	4-35
4.5.6.2	Maximum Receive Buffer Length Register (MRBLR)	4-36
4.5.6.3	Receiver Buffer Descriptor Number (RBD#)	4-36
4.5.6.4	Transmit Buffer Descriptor Number (TBD#)	4-36
4.5.6.5	Other General Parameters	4-37
4.5.7	SCC Initialization	4-37
4.5.8	Interrupt Mechanism	4-38
4.5.8.1	SCC Event Register (SCCE)	4-38
4.5.8.2	SCC Mask Register (SCCM)	4-39
4.5.8.3	SCC Status Register (SCCs)	4-39
4.5.8.4	Bus Error on SDMA Access	4-40
4.5.9	SCC Transparent Mode	4-41
4.5.10	Disabling the SCCs	4-42
4.5.11	UART Controller	4-43
4.5.11.1	Normal Asynchronous Mode	4-45
4.5.11.2	Asynchronous DDCMP MODE	4-46
4.5.11.3	UART Memory Map	4-46
4.5.11.4	UART Programming Model	4-48
4.5.11.5	UART Command Set	4-49
4.5.11.6	UART Address Recognition	4-50
4.5.11.7	UART Control Characters and Flow Control	4-51
4.5.11.8	Send Break	4-53
4.5.11.9	Send Preamble (IDLE)	4-53
4.5.11.10	Wakeup Timer	4-53
4.5.11.11	UART Error-Handling Procedure	4-54
4.5.11.12	Fractional Stop Bits	4-55
4.5.11.13	UART Mode Register.	4-56
4.5.11.14	UART Receive Buffer Descriptor (Rx BD)	4-57
4.5.11.15	UART Transmit Buffer Descriptor (Tx BD)	4-61
4.5.11.16	UART Event Register	4-63
4.5.11.17	UART MASK Register	4-65
4.5.11.18	S-Records Programming Example	4-65
4.5.12	HDLC Controller	4-66

Paragraph Number	Title	Page Number
4.5.12.1	HDLC Channel Frame Transmission Processing	4-68
4.5.12.2	HDLC Channel Frame Reception Processing	4-68
4.5.12.3	HDLC Memory Map	4-69
4.5.12.4	HDLC Programming Model	4-69
4.5.12.5	HDLC Command Set	4-70
4.5.12.6	HDLC Address Recognition	4-71
4.5.12.7	HDLC Maximum Frame Length Register (MFLR)	4-71
4.5.12.8	HDLC Error-Handling Procedure	4-72
4.5.12.9	HDLC Mode Register	4-73
4.5.12.10	HDLC Receive Buffer Descriptor (Rx BD)	4-75
4.5.12.11	HDLC Transmit Buffer Descriptor (Tx BD)	4-78
4.5.12.12	HDLC Event Register	4-80
4.5.12.13	HDLC Mask Register	4-82
4.5.13	BISYNC Controller	4-82
4.5.13.1	Bisync Channel frame Transmission Processing	4-84
4.5.13.2	Bisync Channel Frame Reception Processing	4-85
4.5.13.3	Bisync Memory Map	4-85
4.5.13.4	BISYNC Command Set	4-86
4.5.13.5	BISYNC Control Character Recognition	4-87
4.5.13.6	BSYNC-BISYNC SYNC Register	4-89
4.5.13.7	BDLE-BISYNC DLE Register	4-89
4.5.13.8	BISYNC Error-Handling Procedure	4-90
4.5.13.9	BISYNC Mode Register	4-91
4.5.13.10	BISYNC Receive Buffer Descriptor (Rx BD)	4-93
4.5.13.11	BISYNC Transmit Buffer Descriptor (Tx BD)	4-95
4.5.13.12	BISYNC Event Register	4-97
4.5.13.13	BISYNC Mask Register	4-98
4.5.13.14	Programming the BISYNC Controllers	4-99
4.5.14	DDCMP Controller	4-100
4.5.14.1	DDCMP Channel Frame Transmission Processing	4-101
4.5.14.2	DDCMP Channel Frame Reception Processing.	4-102
4.5.14.3	DDCMP Memory Map	4-103
4.5.14.4	DDCMP Programming Model	4-104
4.5.14.5	DDCMP Command Set.	4-104
4.5.14.6	DDCMP Control Character Recognition	4-105
4.5.14.7	DDCMP Address Recognition.	4-106
4.5.14.8	DDCMP Error-Handling Procedure	4-106
4.5.14.9	DDCMP Mode Register	4-108
4.5.14.10	DDCMP Receive Buffer Descriptor (Rx BD)	4-109
4.5.14.11	DDCMP Transmit Buffer Descriptor (Tx BD)	4-112
4.5.14.12	DDCMP Event Register	4-114
4.5.14.13	DDCMP Mask Register	4-115
4.5.15	V.110 Controller	4-115
4.5.15.1	Bit Rate Adaption of Synchronous Data Signaling Rates up to 19.2 kbps	4-116

Paragraph Number	Title	Page Number
4.5.15.2	Rate Adaption of 48- and 56-kbps User Rates to 64 kbps	4-116
4.5.15.3	Adaption for Asynchronous Rates up to 19.2 kbps	4-117
4.5.15.4	V.110 Controller Overview.	4-117
4.5.15.5	V.110 Programming Model	4-118
4.5.15.6	Error-Handling Procedure	4-118
4.5.15.7	V.110 Receive Buffer Descriptor (Rx BD)	4-118
4.5.15.8	V.110 Transmit Buffer Descriptor (Tx BD)	4-120
4.5.15.9	V.110 Event Register	4-121
4.5.15.10	V.110 Mask Register	4-122
4.5.16	Transparent Controller	4-122
4.5.16.1	Transparent Channel Buffer Transmission Processing	4-123
4.5.16.2	Transparent Channel Buffer Reception Processing	4-124
4.5.16.3	Transparent Memory Map	4-125
4.5.16.4	Transparent Commands	4-126
4.5.16.5	Transparent Synchronization	4-126
4.5.16.6	Transparent Error-Handling Procedure	4-128
4.5.16.7	Transparent Mode Register	4-129
4.5.16.8	Transparent Receive Buffer Descriptor (RxBD)	4-130
4.5.16.9	Transparent Transmit Buffer Descriptor (Tx BD)	4-131
4.5.16.10	Transparent Event Register	4-133
4.5.16.11	Transparent Mask Register	4-134
4.6	Serial Communication Port (SCP)	4-134
4.6.1	SCP Programming Model	4-136
4.6.2	SCP Transmit/Receive Buffer Descriptor	4-137
4.6.3	SCP Transmit/Receive Processing	4-137
4.7	Serial Management Controllers (SMCs)	4-138
4.7.1	Overview	4-138
4.7.1.1	Using IDL with the SMCs	4-138
4.7.1.2	Using GCI with the SMCs	4-138
4.7.2	SMC Programming Model	4-139
4.7.3	SMC Commands	4-140
4.7.4	SMC Memory Structure and Buffers Descriptors	4-140
4.7.4.1	SMC1 Receive Buffer Descriptor	4-141
4.7.4.2	SMC1 Transmit Buffer Descriptor	4-142
4.7.4.3	SMC2 Receive Buffer Descriptor	4-142
4.7.4.4	SMC2 Transmit Buffer Descriptor	4-143
4.7.5	SMC Interrupt Requests	4-143

Section 5
Signal Description

5.1	Functional Groups	5-1
5.2	Power Pins	5-2
5.3	Clocks	5-4
5.4	System Control	5-5
5.5	Address Bus Pins (A23–A1)	5-7

Paragraph Number	Title	Page Number
5.6	Data Bus Pins (D15—D0)	5-7
5.7	Bus Control Pins	5-8
5.8	Bus Arbitration Pins	5-10
5.9	Interrupt Control Pins	5-11
5.10	MC68302 Bus Interface Signal Summary	5-12
5.11	Physical Layer Serial Interface Pins	5-13
5.12	Typical Serial Interface Pin Configurations	5-14
5.13	NMSI1 or ISDN Interface Pins	5-14
5.14	NMSI2 Port or Port a Pins	5-17
5.15	NMSI3 Port or Port A Pins or SCP Pins	5-18
5.16	IDMA or Port A Pins	5-19
5.17	IACK or PIO Port B Pins	5-20
5.18	Timer Pins	5-20
5.19	Parallel I/O Pins with Interrupt Capability	5-22
5.20	Chip-Select Pins	5-22
5.21	No-Connect Pins	5-23
5.22	When to Use Pullup Resistors	5-23

Section 6
Electrical Characteristics

6.1	Maximum Ratings	6-1
6.2	Thermal Characteristics	6-1
6.3	Power Considerations	6-2
6.4	Power Dissipation	6-3
6.5	DC Electrical Characteristics	6-4
6.6	DC Electrical Characteristics—NMSI1 in IDL Mode	6-5
6.7	AC Electrical Specifications—Clock Timing	6-5
6.8	AC Electrical Specifications—IMP Bus Master Cycles	6-6
6.9	AC Electrical Specifications—DMA	6-13
6.10	AC Electrical Specifications—External Master Internal Asynchronous Read/Write Cycles	6-16
6.11	AC Electrical Specifications—External Master Internal Synchronous Read/Write Cycles	6-19
6.12	AC Electrical Specifications—Internal Master Internal Read/Write Cycles	6-23
6.13	AC Electrical Specifications—Chip-Select Timing Internal Master	6-24
6.14	AC Electrical Specifications—Chip-Select Timing External Master	6-25
6.15	AC Electrical Specifications—Parallel I/O	6-26
6.16	AC Electrical Specifications—Interrupts	6-27
6.17	AC Electrical Specifications—Timers	6-28
6.18	AC Electrical Specifications—Serial Communications Port	6-29
6.19	AC Electrical Specifications—IDL Timing	6-30
6.20	AC Electrical Specifications—GCI Timing	6-32
6.21	AC Electrical Specifications—PCM Timing	6-34
6.22	AC Electrical Specifications—NMSI Timing	6-36

Paragraph Number	Title	Page Number
	Section 7	
	Mechanical Data and Ordering Information	
7.1	Pin Assignments	7-1
7.1.1	Pin Grid Array (PGA)	7-1
7.1.2	Plastic Surface Mount (PQFP)	7-2
7.1.3	Thin Surface Mount (TQFP)	7-3
7.2	Package Dimensions	7-4
7.2.1	Pin Grid Array (PGA)	7-4
7.2.2	Plastic Surface Mount (PQFP)	7-5
7.2.3	Thin Surface Mount (TQFP)	7-6
7.3	Ordering Information	7-7
	Appendix A	
	SCC Performance	
	APpendix B	
	Development Tools and Support	
B.1	Motorola Software Overview	B-1
B.2	Motorola Software Modules	B-1
B.3	Third-Party Software Support	B-6
B.4	In-Circuit Emulation Support	B-6
B.5	302 Family ADS System	B-6
	Appendix C	
	RISC Microcode from RAM	
C.1	SS7 Protocol Support	C-2
C.2	Centronics Transmission Controller	C-2
C.3	Centronics Reception Controller	C-3
C.4	Profibus Controller	C-3
C.5	Autobaud Support Package	C-3
C.6	Microcode from RAM Initialization Sequence	C-4
	Appendix D	
	MC68302 Applications	
D.1	Minimum System Configuration	D-1
D.1.1	System Configuration	D-1
D.1.2	Reset Circuit	D-3
D.1.3	Memory Interface	D-4
D.1.4	Memory Circuit	D-4
D.1.5	Memory Timing Analysis	D-4
D.2	Switching the External ROM and RAM Using the MC68302	D-5
D.2.1	Conditions at Reset	D-5
D.2.2	First Things First	D-5
D.2.3	Switching Process	D-6
D.3	MC68302 Buffer Processing and Interrupt Handling	D-7
D.3.1	Buffer Descriptors Definition	D-7

Paragraph Number	Title	Page Number
D.3.2	MC68302 Buffer Processing	D-8
D.3.3	New Pointers	D-9
D.3.4	Initial Conditions	D-10
D.3.5	Transmit Algorithm	D-10
D.3.6	Interrupt Routine	D-10
D.3.7	Final Comments	D-11
D.3.8	HDLC Code Listing	D-11
D.4	Configuring A Uart on the MC68302	D-17
D.4.1	Purpose of the Code	D-17
D.4.2	Organization of Buffers	D-18
D.4.3	Assumptions about the System	D-19
D.4.4	UART Features Not Discussed	D-19
D.4.5	UART Code Listing	D-19
D.5	Independent DMA in the MC68302	D-23
D.5.1	IDMA Overview	D-23
D.5.2	IDMA Software Initialization	D-24
D.5.3	IDMA Bus Arbitration Signals	D-24
D.5.4	Triggering External IDMA Transfers	D-24
D.5.5	Performing Internally Generated IDMA Transfers	D-24
D.5.6	External Cycles Examples	D-26
D.5.7	Internal Interrupt Sequence	D-29
D.5.8	Final Notes	D-30
D.6	MC68302 Multiprotocol Controller Tied to IDL Bus Forms and ISDN Voice/data Terminal	D-30
D.6.1	M68000 Core	D-31
D.6.2	Communications Processor	D-31
D.6.3	System Integration Block	D-31
D.6.4	IDL Bus	D-31
D.6.5	IDL Bus Specification	D-32
D.6.6	IMP/IDL Interconnection	D-33
D.6.7	Serial Interface Configuration	D-35
D.6.8	SCC Configuration	D-36
D.6.9	Parallel I/O Port A Configuration	D-37
D.6.10	SCP Bus	D-37
D.6.11	SCP Configuration	D-38
D.6.12	SCP Data Transactions	D-38
D.6.13	Additional IMP To S/T Chip Connections	D-39
D.6.14	Initialization of the MC145475	D-40
D.6.15	MC145554 CODEC Filter	D-41
D.7	Interfacing a Master MC68302 to One or More Slave MC68302s	D-41
D.7.1	Synchronous vs. Asynchronous Accesses	D-43
D.7.2	Clocking	D-43
D.7.3	Programming the Base Address Registers (BARs)	D-43
D.7.4	Dealing with Interrupts	D-44
D.7.5	Arbitration	D-44

Paragraph Number	Title	Page Number
D.7.6	Final Notes	D-45
D.8	Using the MC68302 Transparent Mode	D-45
D.8.1	Transparent Mode Definition	D-45
D.8.2	Applications for Transparent Mode	D-46
D.8.3	Physical Interface to Accompany Transparent Mode	D-47
D.8.4	General Transparent Mode Behavior	D-50
D.8.5	Transparent Mode with the NMSI Physical Interface	D-52
D.8.6	Other NMSI Modes	D-56
D.8.6.1	BISYNC Mode.	D-56
D.8.6.2	Transync Mode.	D-58
D.8.7	Gating Clocks in NMSI Mode	D-58
D.8.8	Using Transparent Mode with PCM Highway Mode	D-60
D.8.9	PCM Mode Final Thoughts	D-64
D.8.10	Using Transparent Mode with IDL and GCI	D-64
D.8.11	Initializing Transparent Mode	D-65
D.8.12	Special Uses of Transparent Mode	D-67
D.8.12.1	5- OR 6-Bit UART.	D-67
D.8.12.2	Synchronous UART.	D-67
D.8.13	SCP as a Transparent Mode Alternative	D-68
D.8.14	Transparent Mode Summary	D-68
D.9	An Appletalk® Node with the MC68302 and MC68195	D-69
D.9.1	Overview of the Board	D-70
D.9.2	Important Side Notes	D-70

Appendix E
SCC Programming Reference

E.1	HDLC Programming Reference Section	E-1
E.1.1	HDLC Programming Model	E-1
E.1.1.1	COmmunications PRocessor (CP) Registers.	E-3
E.1.1.1.1	Command Register CR).	E-3
E.1.1.1.2	Serial Interface Mode Register (SIMODE).	E-4
E.1.1.1.3	Serial Interface Mask Register (SIMASK).	E-5
E.1.1.2	Per SCC Registers.	E-6
E.1.1.2.1	Serial Configuration Register (SCON).	E-6
E.1.1.2.2	SCC Mode Register (SCM).	E-6
E.1.1.2.3	SCC Data Synchronization Register (DSR)	E-8
E.1.1.2.4	HDLC Event Register (SCCE)	E-8
E.1.1.2.5	HDLC Mask Register (SCCM)	E-9
E.1.1.2.6	HDLC Status Register (SCCS)	E-9
E.1.1.3	General and HDLC Protocol-specific Parameter RAM	E-9
E.1.1.3.1	RFCR/TFCR—Rx Function Code/Tx Function Code.	E-9
E.1.1.3.2	MRBLR—Maximum Rx Buffer Length.	E-10
E.1.1.3.3	CRC Mask_L and CRC Mask_H.	E-10
E.1.1.3.4	DISFC—Discard Frame Counter.	E-10
E.1.1.3.5	CRCEC—CRC Error Counter.	E-10

Paragraph Number	Title	Page Number
E.1.1.3.6	ABTSC—Abort Sequence Counter.	E-10
E.1.1.3.7	NMARC—Nonmatching Address Received Counter.	E-10
E.1.1.3.8	RETRC—Frame Retransmission Counter.	E-10
E.1.1.3.9	MFLR—Maximum Frame Length Register	E-10
E.1.1.3.10	HMASK—HDLC Frame Address Mask	E-10
E.1.1.3.11	HADDR1, HADDR2, HADDR3, and HADDR4-HDLC Frame Address	E-10
E.1.1.4	Receive Buffer Descriptors	E-10
E.1.1.4.1	Receive BD Control/Status Word.	E-11
E.1.1.4.2	Receive Buffer Data Length.	E-12
E.1.1.4.3	Receive Buffer Pointer.	E-12
E.1.1.5	Transmit Buffer Descriptors	E-12
E.1.1.5.1	Transmit BD Control/Status Word.	E-12
E.1.1.5.2	Transmit Buffer Data Length.	E-13
E.1.1.5.3	Transmit Buffer Pointer.	E-13
E.1.2	Programming the SCC for HDLC	E-13
E.1.2.1	CP Initialization	E-13
E.1.2.2	General and HDLC Protocol-Specific RAM Initialization	E-13
E.1.2.3	SCC Initialization	E-14
E.1.2.4	SCC Operation	E-14
E.1.2.5	SCC Interrupt Handling	E-14
E.2	UART Programming Reference Section	E-15
E.2.1	UART Programming Model	E-15
E.2.1.1	Communications Processor (CP) Registers.	E-17
E.2.1.1.1	Command Register (CR).	E-17
E.2.1.1.2	Serial lnterface Mode Register (SlMODE)	E-18
E.2.1.1.3	Serial Interface Mask Register (SIMASK).	E-19
E.2.1.2	Per SCC Registers.	E-19
E.2.1.2.1	Serial Configuration Register (SCON)	E-19
E.2.1.2.2	SCC Mode Register (SCM).	E-20
E.2.1.2.3	SCC Data Synchronization Register (DSR).	E-22
E.2.1.2.4	UART Event Register (SCCE)	E-22
E.2.1.2.5	UART Mask Register (SCCM)	E-23
E.2.1.2.6	UART Status Register (SCCS)	E-23
E.2.1.3	General and UART Protocol-specific Parameter RAM	E-23
E.2.1.3.1	RFCR/TFCR—Rx Function Code/Tx Function Code.	E-24
E.2.1.3.2	MRBLR—Maximum Rx Buffer Length	E-24
E.2.1.3.3	MAX_IDL—Maximum IDLE Characters.	E-24
E.2.1.3.4	BRKCR—Break Count Register.	E-24
E.2.1.3.5	PAREC—Receive Parity Error Counter.	E-24
E.2.1.3.6	FRMEC—Receive Framing Error Counter.	E-24
E.2.1.3.7	NOSEC—Receive Noise Counter.	E-24
E.2.1.3.8	BRKEC—Receive Break Condition Counter.	E-24
E.2.1.3.9	UADDR1 and UADDR2.	E-24
E.2.1.4	Receive Buffer Descriptors	E-25
E.2.1.4.1	Receive BD Control/Status Word.	E-26

Paragraph Number	Title	Page Number
E.2.1.4.2	Receive Buffer Data Length.	E-27
E.2.1.4.3	Receive Buffer Pointer.	E-27
E.2.1.5	Transmit Buffer Descriptors.	E-27
E.2.1.5.1	Transmit BD Control/Status Word.	E-27
E.2.1.5.2	Transmit Buffer Data Length	E-28
E.2.2	Programming the SCC for UART	E-28
E.2.2.1	Initialization	E-29
E.2.2.2	General and UART Protocol-Specific RAM Initialization	E-29
E.2.2.3	SCC Initialization	E-29
E.2.2.4	SCC Operation	E-29
E.2.2.5	SCC Interrupt Handling	E-30
E.3	Transparent Programming Reference Section	E-30
E.3.1	Transparent Programming Model	E-30
E.3.1.1	Communications Processor (CP) Registers	E-32
E.3.1.1.1	Command Register (CR).	E-32
E.3.1.1.2	Serial Interface Mode Register (SIMODE).	E-33
E.3.1.1.3	Serial Interface Mask Register (SIMASK).	E-34
E.3.1.2	PER SCC Registers.	E-35
E.3.1.2.1	Serial Configuration Register (SCON)	E-35
E.3.1.2.2	SCC Mode Register (SCM)	E-35
E.3.1.2.3	SCC Data Synchronization Register (DSR)	E-36
E.3.1.2.4	Transparent Event Register (SCCE)	E-36
E.3.1.2.5	Transparent Mask Register (SCCM)	E-37
E.3.1.2.6	Transparent Status Register (SCCS)	E-38
E.3.1.3	General and Transparent Protocol-Specific Parameter RAM	E-38
E.3.1.3.1	RFCR/TFCR—Rx Function Code/Tx Function Code	E-38
E.3.1.3.2	MRBLR—Maximum Rx Buffer Length	E-38
E.3.1.4	Receive Buffer Descriptors.	E-38
E.3.1.4.1	Receive BD Control/Status Word.	E-38
E.3.1.4.2	Receive Buffer Data Length	E-39
E.3.1.4.3	Receive Buffer Pointer	E-39
E.3.1.5	Transmit Buffer Descriptors.	E-39
E.3.1.5.1	Transmit BD Control/Status Word	E-39
E.3.1.5.2	Transmit Buffer Data Length	E-40
E.3.1.5.3	Transmit Buffer Pointer	E-40
E.3.2	Programming the SCC for Transparent	E-40
E.3.2.1	CP Initialization	E-40
E.3.2.2	General and Transparent Protocol-Specific RAM Initialization	E-41
E.3.2.3	SCC Initialization	E-41
E.3.2.4	SCC Operation	E-41
E.3.2.5	SCC Interrupt Handling	E-41

Appendix F
Design Checklist

LIST OF FIGURES

Figure Number	Title	Page Number
	Section 1 General Description	
Figure 1-1.	MC68302 Block Diagram	1-2
Figure 1-2.	General-Purpose Microprocessor System Design	1-4
Figure 1-3.	MC68302 System Design	1-5
Figure 1-4.	NMSI Communications-Oriented Board Design	1-7
Figure 1-5.	Basic Rate IDL Voice/Data Terminal in ISDN	1-8
	Section 2 MC68000/MC68008 Core	
Figure 2-1.	M68000 Programming Model	2-2
Figure 2-2.	M68000 Status Register	2-3
Figure 2-3.	M68000 Bus/Address Error Exception Stack Frame	2-10
Figure 2-4.	M68000 Short-Form Exception Stack Frame	2-10
Figure 2-5.	MC68302 IMP Configuration Control	2-12
	Section 3 System Integration Block (SIB)	
Figure 3-1.	IDMA Controller Block Diagram	3-3
Figure 3-2.	Interrupt Controller Block Diagram	3-16
Figure 3-3.	Interrupt Request Logic Diagram for SCCs	3-21
Figure 3-4.	SCC1 Vector Calculation Example	3-23
Figure 3-5.	Parallel I/O Block Diagram for PA0	3-30
Figure 3-6.	Parallel I/O Port Registers	3-33
Figure 3-7.	RAM Block Diagram	3-35
Figure 3-8.	Timer Block Diagram	3-36
Figure 3-9.	Chip-Select Block Diagram	3-44
Figure 3-10.	Using an External Crystal	3-49
Figure 3-11.	System Control Register	3-50
Figure 3-12.	IMP Bus Arbiter	3-57
Figure 3-13.	DRAM Control Block Diagram	3-67
	Section 4 Communications Processor (CP)	
Figure 4-1.	Simplified CP Architecture	4-2
Figure 4-2.	Three Serial Data Flow Paths	4-4
Figure 4-3.	NMSI Physical Interface	4-8
Figure 4-4.	Multiplexed Mode on SCC1 Opens Additional Configuration Possibilities	4-9

Figure Number	Title	Page Number
Figure 4-5.	Serial Channels Physical Interface Block Diagram	4-10
Figure 4-6.	IDL Bus Signals	4-11
Figure 4-7.	IDL Terminal Adaptor	4-12
Figure 4-8.	GCI Bus Signals	4-15
Figure 4-9.	Two PCM Sync Methods	4-18
Figure 4-10.	PCM Channel Assignment on a T1/CEPT Line	4-19
Figure 4-11.	SCC Block Diagram	4-24
Figure 4-12.	SCC Baud Rate Generator	4-26
Figure 4-13.	Output Delays from RTS Low, Synchronous Protocol	4-29
Figure 4-14.	Output Delays from CTS Low, Synchronous Protocol	4-29
Figure 4-15.	Memory Structure	4-32
Figure 4-16.	SCC Buffer Descriptor Format	4-33
Figure 4-17.	UART Frame Format	4-43
Figure 4-18.	Two Configurations of UART Multidrop Operation	4-50
Figure 4-19.	UART Control Characters Table	4-51
Figure 4-20.	UART Receive Buffer Descriptor	4-58
Figure 4-21.	UART Rx BD Example	4-59
Figure 4-22.	UART Transmit Buffer Descriptor	4-61
Figure 4-23.	UART Interrupt Events Example	4-64
Figure 4-24.	Typical HDLC Frame	4-66
Figure 4-25.	HDLC Address Recognition Examples	4-71
Figure 4-26.	HDLC Receive Buffer Descriptor	4-75
Figure 4-27.	HDLC Receive BD Example	4-76
Figure 4-28.	HDLC Transmit Buffer Descriptor	4-78
Figure 4-29.	HDLC Interrupt Events Example	4-81
Figure 4-30.	Typical BISYNC Frames	4-83
Figure 4-31.	BISYNC Control Characters Table	4-88
Figure 4-32.	BISYNC Receive Buffer Descriptor	4-93
Figure 4-33.	BISYNC Transmit Buffer Descriptor	4-95
Figure 4-34.	Typical DDCMP Frames	4-100
Figure 4-35.	DDCMP Transmission/Reception Summary	4-102
Figure 4-36.	DDCMP Receive Buffer Descriptor	4-109
Figure 4-37.	DDCMP Transmit Buffer Descriptor	4-112
Figure 4-38.	Two-Step Synchronous Bit Rate Adaption	4-116
Figure 4-39.	Three-Step Asynchronous Bit Rate Adaption	4-117
Figure 4-40.	V.110 Receive Buffer Descriptor	4-119
Figure 4-41.	V.110 Transmit Buffer Descriptor	4-120
Figure 4-42.	Transparent Receive Buffer Descriptor	4-130
Figure 4-43.	Transparent Transmit Buffer Descriptor	4-131
Figure 4-44.	SCP Timing	4-135
Figure 4-45.	SCP vs. SCC Pin Multiplexing	4-137

Section 5
Signal Description

Figure 5-1.	Functional Signal Groups	5-3

Figure Number	Title	Page Number
Figure 5-2.	Clock Pins	5-4
Figure 5-3.	System Control Pins	5-5
Figure 5-4.	Address Bus Pins	5-7
Figure 5-5.	Data Bus Pins	5-7
Figure 5-6.	Bus Control Pins	5-8
Figure 5-7.	External Address/Data Buffer	5-9
Figure 5-8.	Bus Arbitration Pins	5-10
Figure 5-9.	Interrupt Control Pins	5-11
Figure 5-10.	NMSI1 or ISDN Interface Pins	5-14
Figure 5-11.	NMSI2 Port or Port A Pins	5-17
Figure 5-12.	NMSI3 Port or Port A Pins or SCP Pins	5-18
Figure 5-13.	IDMA or Port A Pins	5-19
Figure 5-14.	IACK or PIO Port B Pins	5-20
Figure 5-15.	Timer Pins	5-21
Figure 5-16.	Port B Parallel I/O Pins with Interrupt	5-22
Figure 5-17.	Chip-Select Pins	5-22

Section 6
Electrical Characteristics

Figure 6-1.	Clock Timing Diagram	6-5
Figure 6-2.	Read Cycle Timing Diagram	6-9
Figure 6-3.	Write Cycle Timing Diagram	6-10
Figure 6-4.	Read-Modify-Write Cycle Timing Diagram	6-11
Figure 6-5.	Bus Arbitration Timing Diagram	6-12
Figure 6-6.	DMA Timing Diagram (IDMA)	6-14
Figure 6-7.	DMA Timing Diagram (SDMA)	6-15
Figure 6-8.	External Master Internal Asynchronous Read Cycle Timing Diagram	6-17
Figure 6-9.	External Master Internal Asynchronous Write Cycle Timing Diagram	6-18
Figure 6-10.	External Master Internal Synchronous Read Cycle Timing Diagram	6-20
Figure 6-11.	External Master Internal Synchronous Read Cycle Timing Diagram (One Wait State)	6-21
Figure 6-12.	External Master Internal Synchronous Write Cycle Timing Diagram	6-22
Figure 6-13.	Internal Master Internal Read/Write Cycle Timing Diagram	6-23
Figure 6-14.	Internal Master Chip-Select Timing Diagram	6-25
Figure 6-15.	External Master Chip-Select Timing Diagram	6-26
Figure 6-16.	Parallel I/O Data-In/Data-Out Timing Diagram	6-27
Figure 6-17.	Interrupts Timing Diagram	6-27
Figure 6-18.	Timers Timing Diagram	6-28
Figure 6-19.	Serial Communication Port Timing Diagram	6-29
Figure 6-20.	IDL Timing Diagram	6-31
Figure 6-21.	GCI Timing Diagram	6-33
Figure 6-22.	PCM Timing Diagram (SYNC Envelopes Data)	6-35
Figure 6-23.	PCM Timing Diagram (SYNC Prior to 8-Bit Data)	6-35
Figure 6-24.	NMSI Timing Diagram	6-37

Figure Number | **Title** | **Page Number**

Appendix B
Development Tools and Support

Figure B-1. Software Overview B-3
Figure B-2. MC68302FADS B-8

Appendix C
RISC Microcode from RAM

Figure C-1. CP Architecture Running RAM Microcode C-1

Appendix D
MC68302 Applications

Figure D-1. MC68302 Minimum System Configuration (Sheet 1 of 2) D-2
Figure D-2. MC68302 Minimum System Configuration (Sheet 2 of 2) D-3
Figure D-3. Transmit and Receive BD Structure D-7
Figure D-4. Transmit and Receive BD Tables D-8
Figure D-5. Pointer during Execution D-9
Figure D-6. Transmit and Receive BD Tables and Buffers D-18
Figure D-7. Typical IDMA External Cycles (Normal Operation) D-27
Figure D-8. Typical IDMA External Cycles Showing Block Transfer Termination D-28
Figure D-9. Typical IDMA Source to Word Destination IDMA Cycles D-28
Figure D-10. Burst Mode Cycles D-29
Figure D-11. ISDN Voice/Data Terminal D-30
Figure D-12. IDL Bus Boundaries D-32
Figure D-13. IDL Frame Structure D-33
Figure D-14. IDL Bus to Other Slaves D-34
Figure D-15. Serial Interface Configuration D-35
Figure D-16. SCP Bus Interconnection D-38
Figure D-17. Discrete Signal Interconnection D-40
Figure D-18. CODEC/IDL Electrical Connection D-41
Figure D-19. Typical Slave Mode Example D-42
Figure D-20. Dual Master-Slave System D-46
Figure D-21. NMSI Pin Definitions D-48
Figure D-22. Multiplexed Modes Example D-49
Figure D-23. Simplest Transmit Case in NMSI D-53
Figure D-24. Simplest Receive Case in NMSI D-53
Figure D-25. Using CTS In the NMSI Transmit Case D-54
Figure D-26. Using CD (Sync) In the NMSI Transmit Case D-55
Figure D-27. Using CD (Sync) in the NMSI Receive Case D-56
Figure D-28. External Loopback with RTS Connected to CD D-56
Figure D-29. Routing Channels in PCM Envelope Mode D-61
Figure D-30. PCM Transmission Timing Technique D-63
Figure D-31. SCP Timing D-69
Figure D-32. Local Talk Adaptor Board D-71

LIST OF TABLES

Table Number	Title	Page Number

Section 2
MC68000/MC68008 Core

Table 2-1. M68000 Data Addressing Modes 2-4
Table 2-2. M68000 Instruction Set Summary 2-5
Table 2-3. M68000 Instruction Type Variations 2-6
Table 2-4. M68000 Address Spaces 2-7
Table 2-5. M68000 Exception Vector Assignment 2-8
Table 2-6. System Configuration Register 2-14
Table 2-7. System RAM 2-14
Table 2-8. Parameter RAM 2-15
Table 2-9. Internal Registers 2-17

Section 3
System Integration Block (SIB)

Table 3-1. SAPR and DAPR Incrementing Rules 3-10
Table 3-2. IDMA Bus Cycles 3-11
Table 3-3. EXRQ and INRQ Prioritization 3-19
Table 3-4. INRQ Prioritization within Interrupt Level 4 3-19
Table 3-5. Encoding the Interrupt Vector 3-23
Table 3-6. Port A Pin Functions 3-31
Table 3-7. Port B Pin Functions 3-32
Table 3-8. DTACK Field Encoding 3-47
Table 3-9. SCR Register Bits 3-51
Table 3-10. Bus Arbitration Priority Table 3-58
Table 3-11. DRAM Refresh Memory Map Table 3-68

Section 4
Communications Processor (CP)

Table 4-1. The Five Possible SCC Combinations 4-9
Table 4-2. PCM Highway Mode Pin Functions 4-17
Table 4-3. PCM Channel Selection 4-17
Table 4-4. Typical Bit Rates of Asynchronous Communication 4-27
Table 4-5. Transmit Data Delay (TCLK Periods) 4-28
Table 4-6. SCC Parameter RAM Memory Map 4-35
Table 4-7. UART Specific Parameter RAM 4-46
Table 4-8. HDLC-Specific Parameter RAM 4-69
Table 4-9. BISYNC Specific Parameter RAM 4-86
Table 4-10. DDCMP Specific Parameter RAM 4-104
Table 4-11. Transparent-Specific Parameter RAM 4-125

Table Number | **Title** | **Page Number**

Section 5
Signal Description

Table 5-1. Signal Definitions 5-1
Table 5-2. Bus Signal Summary—Core and External Master 5-12
Table 5-3. Bus Signal Summary—IDMA and SDMA 5-13
Table 5-4. Serial Interface Pin Functions 5-13
Table 5-5. Typical ISDN Configurations 5-14
Table 5-6. Typical Generic Configurations 5-14
Table 5-7. Mode Pin Functions 5-15
Table 5-8. PCM Mode Signals 5-16
Table 5-9. Baud Rate Generator Outputs 5-18

Appendix D
MC68302 Applications

Table D-1. IDMA Registers D-23
Table D-2. Channel Mode Register Bits D-25
Table D-3. Channel Status Register Bits D-28
Table D-4. PCM Highway Pin Names and Functions D-60
Table D-5 PCM Highway Channel Selection with LlSY0 and L1SY1 ..... D-60

Appendix E
SCC Programming Reference

Table E-1 (a) HDLC Programming Mode Receive and Transmit Buffer Descriptors for SCCx E-2
Table E-1 (b). HDLC Programming Model (Continued) General Parameter and HDLC Protocol-Specific RAM for SCCx E-2
Table E-1 (c) SCCx Register Set E-3
Table E-1 (d). General Registers (Only One Set) E-3
Table E-2 (a) UART Programming Model Receive and Transmit Buffer Descriptors for SCCx E-15
Table E-2 (b) UART Programming Model (Continued) General Parameter and UART Protocol-Specific RAM for SCCx E-16
Table E-2 (c) SCCx Register Set E-16
Table E-2 (d). General Registers (Only One Set) E-16
Table E-3 (a). Transparent Programming Model Receive and Transmit Buffer Descriptors for SCCx E-31
Table E-3 (b). Transparent Programming Model (Continued) General Parameter and Transparent Protocol-Specific RAM for SCCx E-31
Table E-3 (c). SCCx Register Set E-32
Table E-3 (d). General Registers (Only One Set) E-32

SECTION 1
GENERAL DESCRIPTION

The MC68302 integrated multiprotocol processor (IMP) is a very large-scale integration (VLSI) device incorporating the main building blocks needed for the design of a wide variety of controllers. The device is especially suitable to applications in the communications industry. The IMP is the first device to offer the benefits of a closely coupled, industry-standard M68000 microprocessor core and a flexible communications architecture. The IMP may be configured to support a number of popular industry interfaces, including those for the Integrated Services Digital Network (ISDN) basic rate and terminal adaptor applications. Concurrent operation of different protocols is easily achieved through a combination of architectural and programmable features. Data concentrators, line cards, modems, bridges, and gateways are examples of suitable applications for this device.

The IMP is a high-density complementary metal-oxide semiconductor (HCMOS) device consisting of an M68000 microprocessor core, a system integration block (SIB), and a communications processor (CP).

1.1 BLOCK DIAGRAM

The block diagram is shown in Figure 1-1.

By integrating the microprocessor core with the serial ports (in the CP) and the system peripherals (in the SIB), the IMP is capable of handling complex tasks such as all ISDN basic rate (2B + D) access tasks. For example, the IMP architecture and the serial communications controller (SCC) ports can support the interface of an S/T transceiver chip and the lower part (bit handling) ISO/OSI layer-2 functions. Other layer-2 functions and the higher protocol layers would then be implemented by software executed by the M68000 core.

Using the flexible memory-based buffer structure of the IMP, terminal adaptor applications also can be supported by transforming and sharing data buffer information between the three SCC ports and the serial communications port (SCP). Each SCC channel is available for HDLC/SDLC[1], UART, BISYNC, DDCMP[2], V.110, or transparent operation. The IMP provides a number of choices for various rate adaptive techniques and can be used for functions such as a terminal controller, multiplexer, or concentrator.

---

1. SDLC is a trademark of International Business Machines.

2. DDCMP is a trademark of Digital Equipment Corporation.

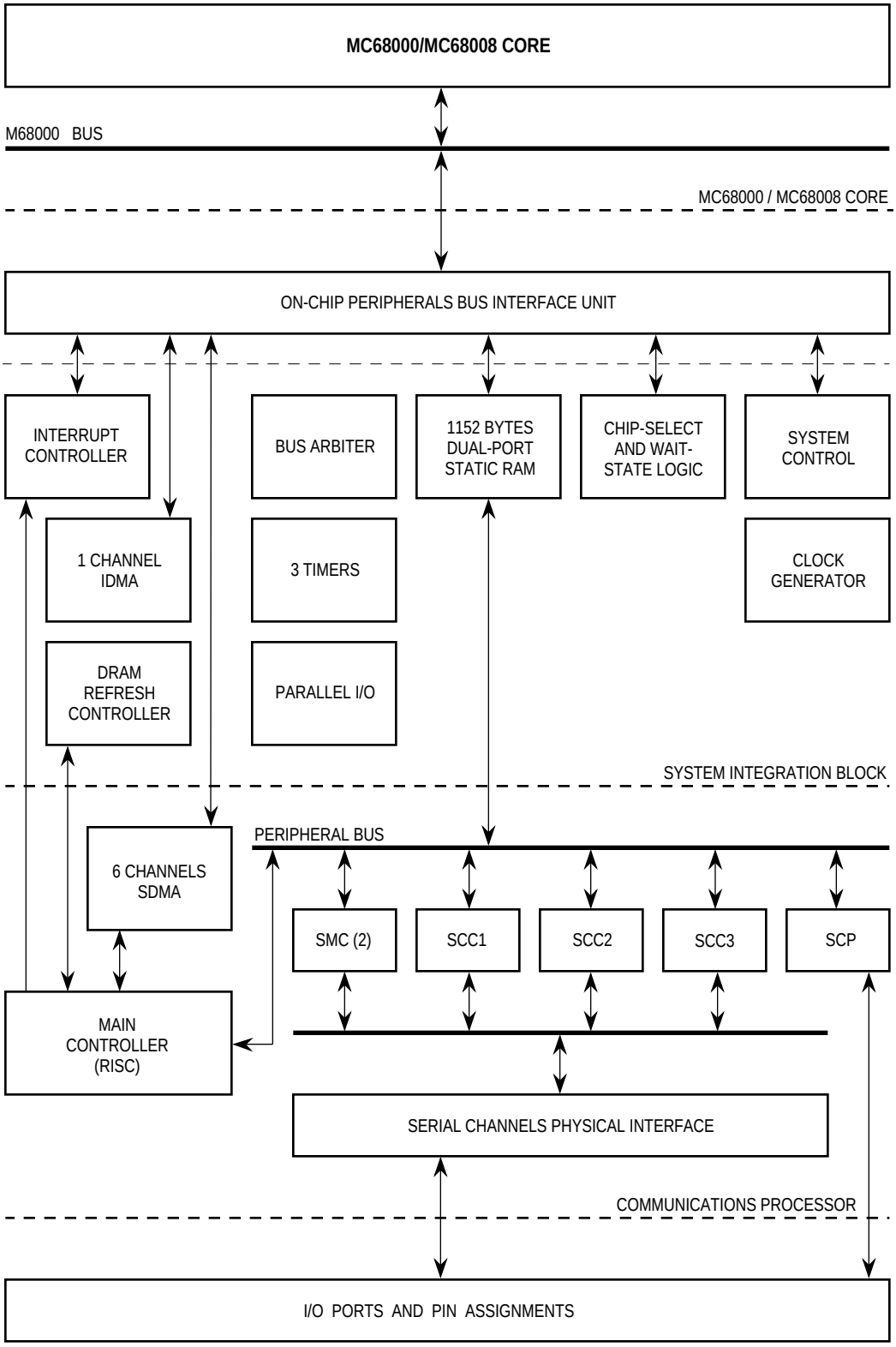


Figure 1-1. MC68302 Block Diagram

The MC68302 can also be used in applications such as board-level industrial controllers performing real-time control applications with a local control bus and an X.25 packet network connection. Such a system provides the real-time response to a demanding peripheral while permitting remote monitoring and communication through an X.25 packet network.

1.2 FEATURES

The features of the IMP are as follows:

- On-Chip HCMOS MC68000/MC68008 Core Supporting a 16- or 8-Bit M68000 Family-System
- IB Including:
  - —Independent Direct Memory Access (IDMA) Controller with Three Handshake Signals: $\overline{\text{DREQ}}$, $\overline{\text{DACK}}$, and $\overline{\text{DONE}}$.
  - —Interrupt Controller with Two Modes of Operation
  - —Parallel Input/Output (I/O) Ports, Some with Interrupt Capability
  - —On-Chip 1152-Byte Dual-Port RAM
  - —Three Timers Including a Watchdog Timer
  - —Four Programmable Chip-Select Lines with Wait-State Generator Logic
  - —Programmable Address Mapping of the Dual-Port RAM and IMP Registers
  - —On-Chip Clock Generator with Output Signal
  - —System Control:
    - Bus Arbitration Logic with Low-Interrupt Latency Support
    - System Status and Control Logic
    - Disable CPU Logic (M68000)
    - Hardware Watchdog
    - Low-Power (Standby) Modes
    - Freeze Control for Debugging
    - DRAM Refresh Controller
- CP Including:
  - —Main Controller (RISC Processor)
  - —Three Independent Full-Duplex Serial Communications Controllers (SCCs)
  - —Supporting Various Protocols:
    - High-Level/Synchronous Data Link Control (HDLC/SDLC)
    - Universal Asynchronous Receiver Transmitter (UART)
    - Binary Synchronous Communication (BISYNC)
    - Synchronous/Asynchronous Digital Data Communications Message Protocol (DDCMP)
    - Transparent Modes
    - V.110 Rate Adaption
  - —Six Serial DMA Channels for the Three SCCs
  - —Flexible Physical Interface Accessible by SCCs Including:
    - Motorola Interchip Digital Link (IDL)
    - General Circuit Interface (GCI, also known as IOM[3]-2)
    - Pulse Code Modulation (PCM) Highway Interface

3. IOM is a trademark of Siemens AG

Nonmultiplexed Serial Interface (NMSI) Implementing Standard Modem Signals
—SCP for Synchronous Communication
—Two Serial Management Controllers (SMCs) To Support IDL and GCI Auxiliary Channels

1.3 MC68302 SYSTEM ARCHITECTURE

Most general-purpose microprocessor-based systems use an architecture that interfaces all peripheral devices directly onto a single microprocessor bus (see Figure 1-2).

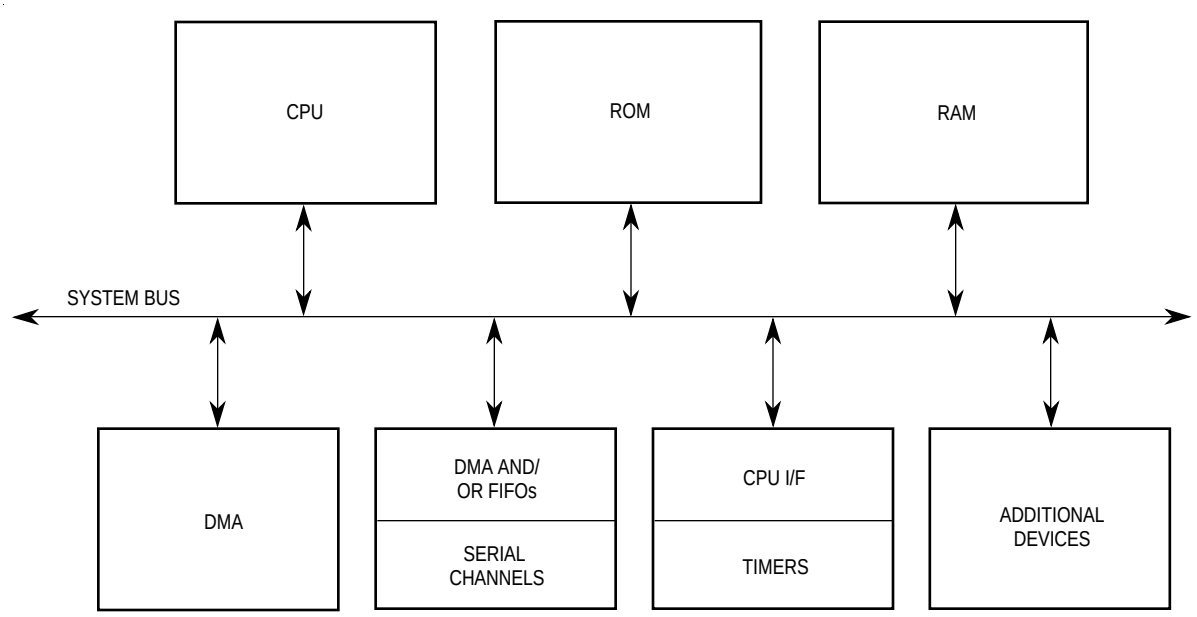


Figure 1-2. General-Purpose Microprocessor System Design

The MC68302 microprocessor architecture is shown in Figure 1-3. In this architecture, the peripheral devices are isolated from the system bus through a dual-port memory. Various parameters and counters and all memory buffer descriptor tables reside in the dual-port RAM. The receive and transmit data buffers may be located in the on-chip RAM or in the off-chip system RAM. Six DMA channels are dedicated to the six serial ports (receive and transmit for each of the three SCC channels). If data for an SCC channel is programmed to be located in the external RAM, the CP will program the corresponding DMA channel for the required accesses, bypassing the dual-port RAM. If data resides in the dual-port RAM, then the CP accesses the RAM with one clock cycle and no arbitration delays.

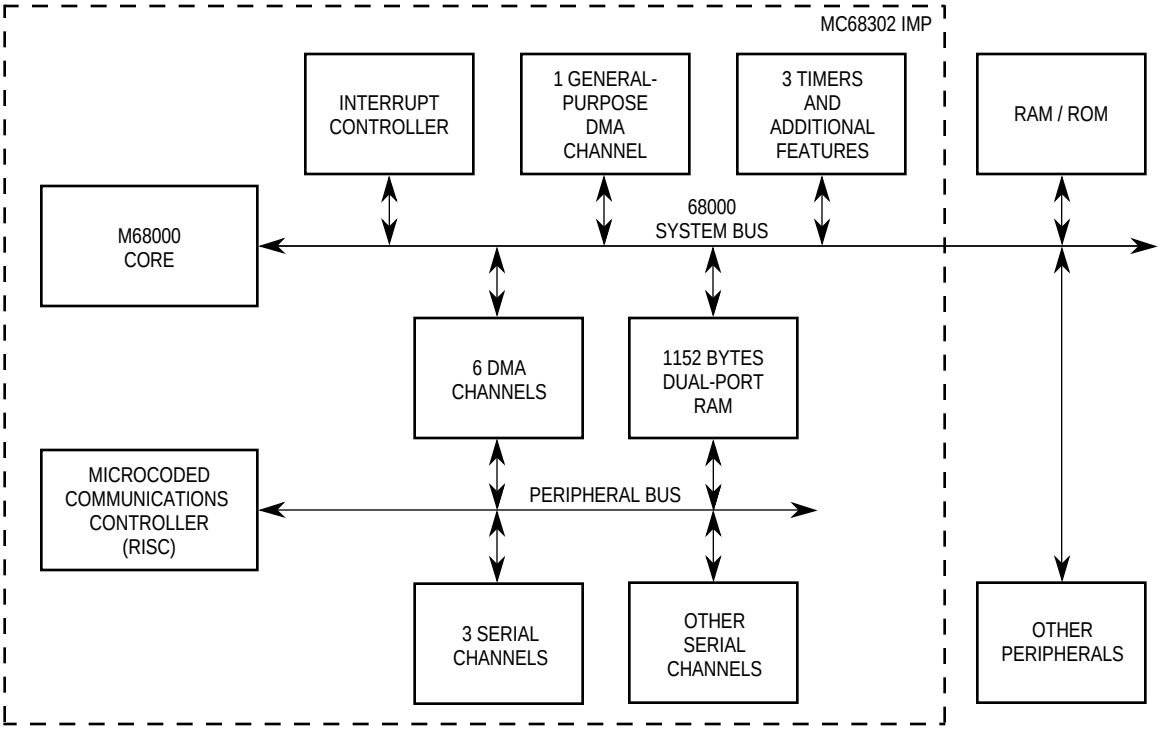


Figure 1-3. MC68302 System Design

The use of a unique arbitration scheme and synchronous transfers between the microprocessor and dual-port RAM gives zero wait-state operation to the M68000 microprocessor core. The dual-port RAM can be accessed by the CP main controller (RISC) once every clock cycle for either read or write operations. When the M68000 core accesses the dual-port RAM, each access is pipelined along with the CP accesses so that data is read or written without conflict. The net effect is the loss of a single memory access by the CP main controller per M68000 core access.

The buffer memory structure of the MC68302 can be configured to closely match I/O channel requirements by careful selection of buffer size and buffer linking. The interrupt structure is also programmable so that the on-chip M68000 processor can be off-loaded from the peripheral bit-handling functions to perform higher layer application software or protocol processing.

1.4 NMSI COMMUNICATIONS-ORIENTED ENVIRONMENT

When the interface to equipment or proprietary networks requires the use of standard control and data signals, the MC68302 can be programmed into the nonmultiplexed serial interface (NMSI) mode. This mode, which is available for one, two, or all three SCC ports, can be selected while the other ports use one of the multiplexed interface modes (IDL, GCI, or PCM highway).

In the example shown in Figure 1-4, one SCC channel connects through the NMSI mode to a commercial packet data network. This connection might be used for remote status monitoring or for maintenance functions for a system. Another SCC is used to connect to a local asynchronous terminal. The other SCC channel is used as a local synchronous channel, which could connect to another computer or subsystem. The SCP channel could then be used for local interconnection of interface chips or peripherals to the MC68302-based system.

1.5 BASIC RATE ISDN OR DIGITAL VOICE/DATA TERMINAL

A basic rate ISDN (2B + D) or digital voice/data terminal can be made from a chip set based on the MC68302. Refer to Figure 1-5 for an example of a basic rate ISDN voice/data terminal. In this terminal, the CP can directly support the 2B + D channels and perform either V.110 or V.120 rate adaption. The physical layer serial interface is connected to the local interconnection bus (IDL in Figure 1-5, but the GCI and PCM buses can also be supported). The system then supports one of the B channels for voice (connected directly to the physical bus). The D channel consists of one SCC port; the other B channel is used for data transfer through a second SCC port. The data can be routed to a terminal (RS-232 type) via the third SCC port in the NMSI mode. The SCP functions as a control channel for the IDL bus in this case.

Some ISDN physical layer devices support the signaling and framing functions of the D channel. In these cases, the D channel can connect through the microprocessor interface to the physical layer device, and the extra SCC port can then be used for a second B channel to transfer data.

The benefit of a local interconnection bus (see Figure 1-5) versus a microprocessor bus is a lower pin count. It is also easy to maintain this low pin-count interface between several different interface chips, such as the MC145554 PCM codec/filter monocircuit and the MC145474 S/T transceiver.

The MC68302 combines the M68000 architecture with a number of peripherals for integrated applications in communications control. The M68000 core manages the CP through the on-chip, dual-port RAM and internal registers. The base address of the dual-port RAM and internal registers is selected through the base address register. Other peripherals are also accessed and controlled through internal registers: the IDMA controller, the three timers, I/O ports, and the interrupt controller.

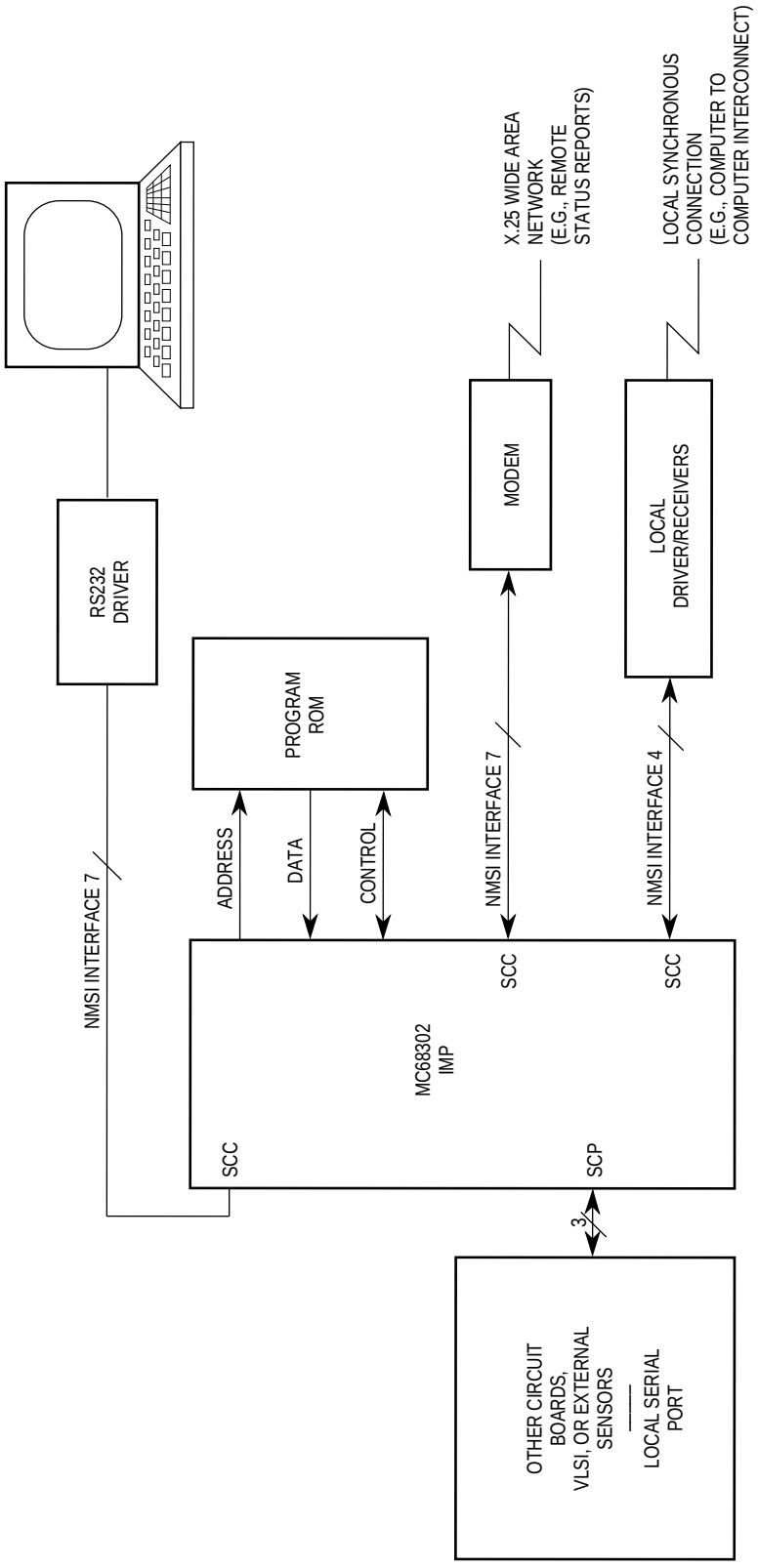


Figure 1-4. NMSI Communications-Oriented Board Design

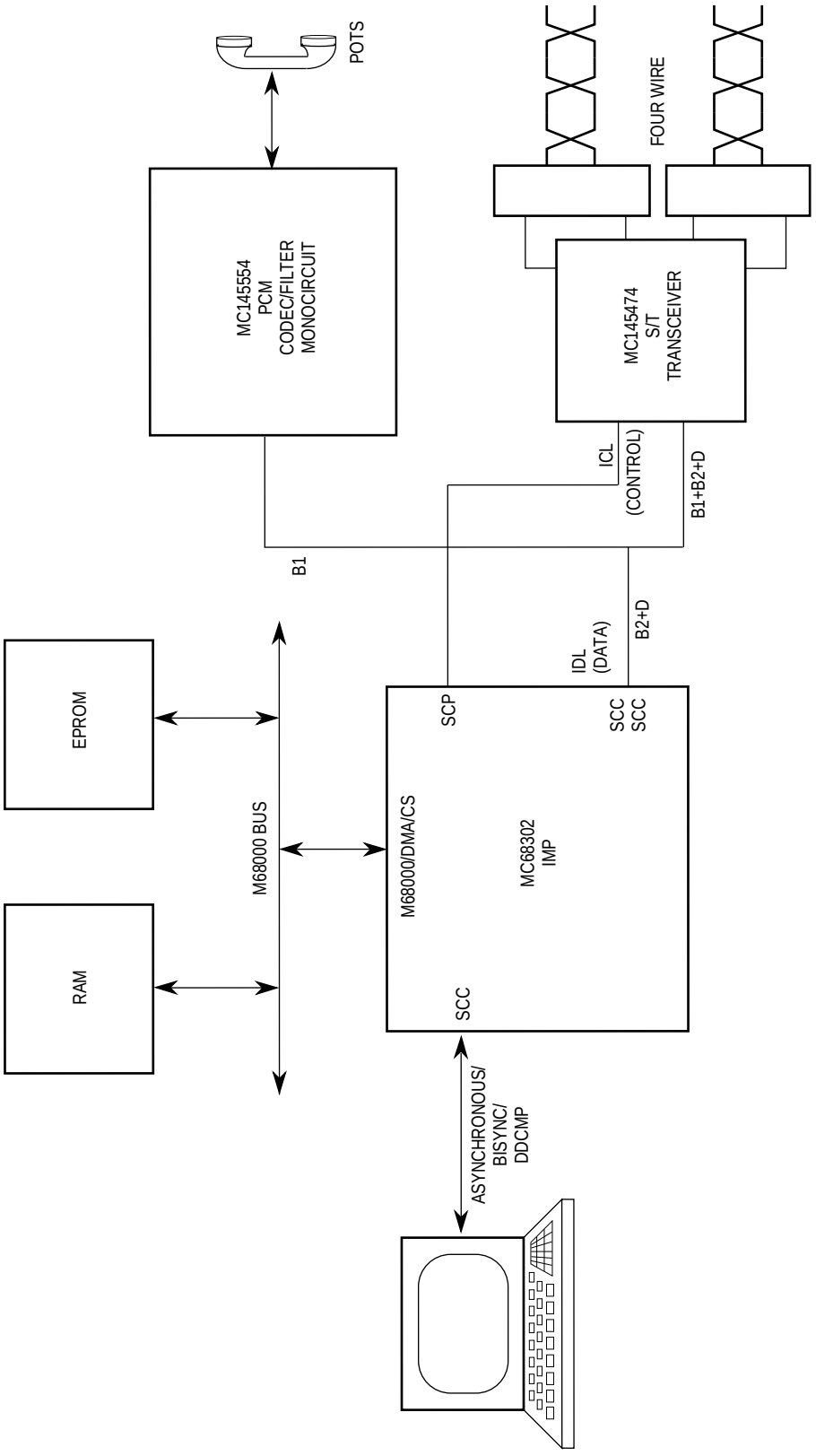


Figure 1-5. Basic Rate IDL Voice/Data Terminal in ISDN

SECTION 2
MC68000/MC68008 CORE

The MC68302 integrates a high-speed M68000 processor with multiple communications peripherals. The provision of direct memory access (DMA) control and link layer management with the serial ports allows high throughput of data for communications-intensive applications, such as basic rate Integrated Services Digital Network (ISDN).

The MC68302 can operate either in the full MC68000 mode with a 16-bit data bus or in the MC68008 mode with an 8-bit data bus by tying the bus width (BUSW) pin low. $\overline{\text{UDS}}$/A0 functions as A0 and $\overline{\text{LDS}}$/$\overline{\text{DS}}$ functions as $\overline{\text{DS}}$ in the MC68008 mode.

NOTE

The BUSW pin is static and is not intended to be used for dynamic bus sizing. If the state of BUSW is changed during operation of the MC68302, erratic operation may occur.

Refer to the MC68000UM/AD, *M68000 8-/16-/32-Bit Microprocessors User's Manual*, for complete details of the on-chip microprocessor. Throughout this manual, references may use the notation M68000, meaning all devices belonging to this family of microprocessors, or the notation MC68000, MC68008, meaning the specific microprocessor products.

2.1 PROGRAMMING MODEL

The M68000 microprocessor executes instructions in one of two modes: user or supervisor. The user mode provides the execution environment for most of the application programs. The supervisor mode, which allows some additional instructions and privileges, is intended for use by the operating system and other system software.

Shown in Figure 2-1, the M68000 core programming model offers 16, 32-bit, general-purpose registers (D7–D0, A7–A0), a 32-bit program counter (PC), and an 8-bit condition code register (CCR) when running in user space. The first eight registers (D7–D0) are used as data registers for byte (8-bit), word (16-bit), and long-word (32-bit) operations. The second set of seven registers (A6–A0) and the stack pointer (USP in user space) may be used as software stack pointers and base address registers. In addition, the address registers may be used for word and long-word operations. All 16 registers may be used as index registers.

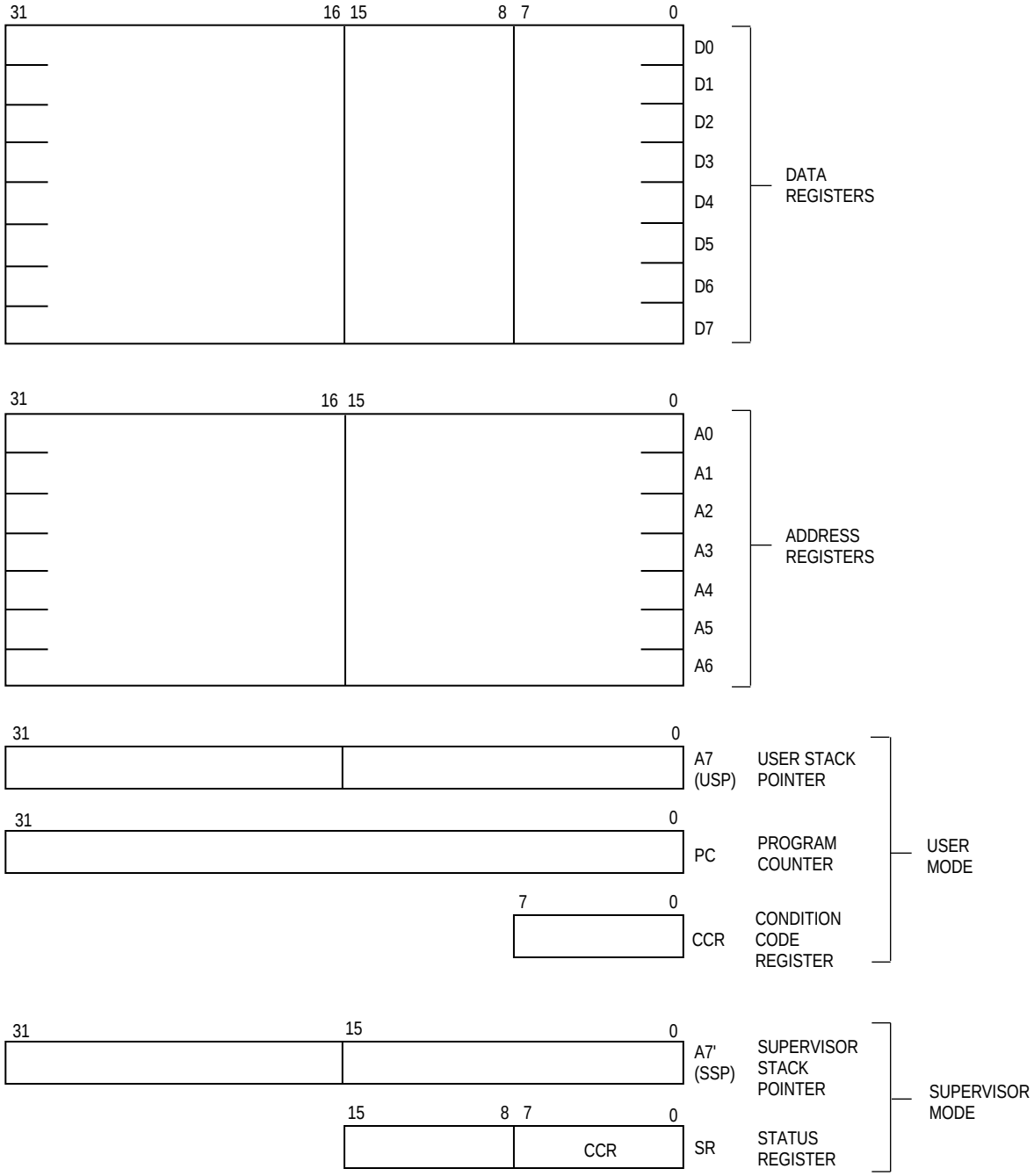


Figure 2-1. M68000 Programming Model

The supervisor's programming model includes supplementary registers, including the supervisor stack pointer (SSP) and the status register (SR) as shown in Figure 2-2. The SR contains the interrupt mask (eight levels available) as well as the following condition codes: overflow (V), zero (Z), negative (N), carry (C), and extend (X). Additional status bits indicate that the processor is in trace (T) mode and/or in a supervisor (S) state.

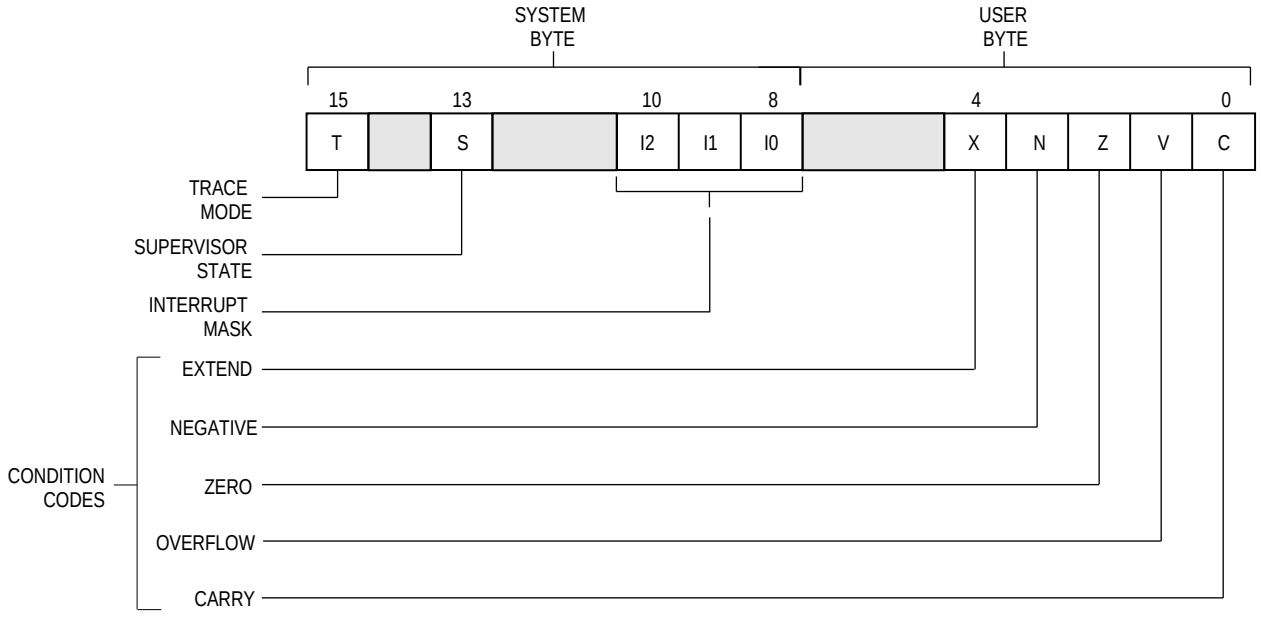


Figure 2-2. M68000 Status Register

2.2 INSTRUCTION SET SUMMARY

The five data types supported by the M68000 on the MC68302 are bits, binary-coded decimal (BCD) digits (4 bits), bytes (8 bits), words (16 bits), and long words (32 bits).

In addition, operations on other data types, such as memory addresses, status word data, etc., are provided for in the instruction set. Shown in Table 2-1, the 14 flexible addressing modes include six basic types:

- Register Direct
- Register Indirect
- Absolute
- Immediate
- Program Counter Relative
- Implied

The capability to perform postincrementing, predecrementing, offsetting, and indexing is included in the register indirect addressing modes. Program counter relative modes can also be modified via indexing and offsetting.

The M68000 instruction set is shown in Table 2-2.

Some basic instructions also have variations as shown in Table 2-3.

Special emphasis has been placed on the instruction set to simplify programming and to support structured high-level languages. With a few exceptions, each instruction operates

on bytes, words, or long words, and most instructions can use any of the 14 addressing modes.

Combining instruction types, data types, and addressing modes provides over 1000 useful instructions. These instructions include signed and unsigned multiply and divide, quick arithmetic operations, BCD arithmetic, and expanded operations (through traps).

Table 2-1. M68000 Data Addressing Modes

Mode	Generation
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) + Xn + d8
Register Indirect Addressing	
Register Indirect	EA = (An)
Postincrement Register Indirect	EA = (An), An ⇐ An + N
Predecrement Register Indirect	EA = ← 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

NOTES:

EA	=	Effective Address
An	=	Address Register
Dn	=	Data Register
Xn	=	Address or Data Register Used as an Index Register
SR	=	Status Register
PC	=	Program Counter
( )	=	Contents of
d_8	=	8-Bit Offset (Displacement)
d16	=	16-Bit Offset (Displacement)
N	=	1 for byte, 2 for word, and 4 for long word. If An is the stack pointer and the operand size is byte, N = 2 to keep the stack pointer on a word boundary.
←	=	Replaces

Table 2-2. M68000 Instruction Set Summary

Mnemonic	Description
ABCD	Add Decimal with Extend
ADD	Add
AND	Logical AND
ASL	Arithmetic Shift Left
ASR	Arithmetic Shift Right
Bcc	Branch Conditionally
BCHG	Bit Test and Change
BCLR	Bit Test and Clear
BRA	Branch Always
BSET	Bit Test and Set
BSR	Branch to Subroutine
BTST	Bit Test
CHK	Check Register Against Bounds
CLR	Clear Operand
CMP	Compare
DBcc	Decrement and Branch Conditionally
DIVS	Signed Divide
DIVU	Unsigned Divide
EOR	Exclusive OR
EXG	Exchange Registers
EXT	Sign Extend
JMP	Jump
JSR	Jump to Subroutine
LEA	Load Effective Address
LINK	Link Stack
LSL	Logical Shift Left
LSR	Logical Shift Right
MOVE	Move Source to Destination
MULS	Signed Multiply
MULU	Unsigned Multiply
NBCD	Negate Decimal with Extend
NEG	Negate
NOP	No Operation
NOT	Ones Complement
OR	Logical OR
PEA	Push Effective Address
RESET	Reset External Devices
ROL	Rotate Left without Extend
ROR	Rotate Right without Extend
ROXL	Rotate Left with Extend
ROXR	Rotate Right with Extend
RTE	Return from Exception
RTR	Return and Restore
RTS	Return from Subroutine
SBCD	Subtract Decimal with Extend
Scc	Set Conditionally
STOP	Stop
SUB	Subtract
SWAP	Swap Data Register Halves
TAS	Test and Set Operand
TRAP	Trap
TRAPV	Trap on Overflow
TST	Test
UNLK	Unlink

Table 2-3. M68000 Instruction Type Variations

Instruction Type	Variation	Description
ADD	ADD ADDA ADDQ ADDI ADDX	Add Add Address Add Quick Add Immediate Add with Extend
AND	AND ANDI ANDI to CCR ANDI to SR	Logical AND And Immediate And Immediate to Condition Codes And Immediate to Status Registers
CMP	CMP CMPA CMPM CMPI	Compare Compare Addresses Compare Memory Compare Immediate
EOR	EOR EORI EORI to CCR EORI to SR	Exclusive OR Exclusive OR Immediate Exclusive OR Immediate to Condition Codes Exclusive OR Immediate to Status Register
MOVE	MOVE MOVEA MOVEM MOVEP MOVEQ MOVE from SR MOVE to SR MOVE to CCR MOVE USP	Move Source to Destination Move Address Move Multiple Register Move Peripheral Data Move Quick Move from Status Register Move to Status Register Move to Condition Codes Move User Stack Pointer
NEG	NEG NEGX	Negate Negate with Extend
OR	OR ORI ORI to CCR ORI to SR	Logical OR OR Immediate OR Immediate to Condition Codes OR Immediate to Status Register
SUB	SUB SUBA SUBI SUBQ SUBX	Subtract Subtract Address Subtract Immediate Subtract Quick Subtract with Extend

2.3 ADDRESS SPACES

The M68000 microprocessor operates in one of two privilege states: user or supervisor. The privilege state determines which operations are legal, which operations are used by the external memory management device to control and translate accesses, and which operations are used to choose between the SSP and the USP in instruction references. The M68000 address spaces are shown in Table 2-4.

In the M68000 Family, the address spaces are indicated by function code pins. On the M68000, three function code pins are output from the device on every bus cycle of every executed instruction. This provides the purpose of each bus cycle to external logic.

Other bus masters besides the M68000 may also output function codes during their bus cycles. On the MC68302, this capability is provided for each potential internal bus master (i.e., the IDMA, SDMA, and DRAM refresh units). Also on the MC68302, provision is made for the decoding of function codes that are output from external bus masters (e.g., in the chip-select generation logic).

In computer design, function code information can be used to protect certain portions of the address map from unauthorized access or even to extend the addressable range beyond the M68000 16-Mbyte address limit. However, in controller applications, function codes are used most often as a debugging aid. Furthermore, in many controller applications, the M68000 stays continuously in the supervisor state.

Table 2-4. M68000 Address Spaces

Function Code Output			
FC2	**FC1**	**FC0**	**Reference Class**
1	0	0	(Unassigned)
0	0	1	User Data
0	1	0	User Program
0	1	1	(Unassigned)
1	0	0	(Unassigned)
1	0	1	Supervisor Data
1	1	0	Supervisor Program
1	1	1	CPU Space*

* This is the function code output for the M68000 interrupt acknowledge cycle.

All exception processing occurs in the supervisor state, regardless of the state of the S bit when the exception occurs. The bus cycles generated during exception processing are classified as supervisor references. All stacking operations during exception processing use the SSP.

The user state is the lower state of privilege. For instruction execution, the user state is determined by the S bit of the SR; if the S bit is negated (low), the processor is executing instructions in the user state. Most instructions execute identically in either user state or supervisor state. However, instructions having important system effects are privileged. User programs are not permitted to execute the STOP instruction or the RESET instruction. To ensure that a user program cannot enter the supervisor state except in a controlled manner, the instructions which modify the entire SR are privileged. To aid in debugging programs to be used in operating systems, the move-to-user-stack-pointer (MOVE to USP) and move-from-user-stack-pointer (MOVE from USP) instructions are also privileged.

The supervisor state is the highest state of privilege. For instruction execution, the supervisor state is determined by the S bit of the SR; if the S bit is asserted (high), the processor is in the supervisor state. The bus cycles generated by instructions executed in the supervisor state are classified as supervisor references. While the processor is in the supervisor privilege state, those instructions using either the system stack pointer implicitly or address register seven explicitly access the SSP.

Once the processor is in the user state and executing instructions, only exception processing can change the privilege state. During exception processing, the current state of the S bit in the SR is saved and the S bit is asserted, putting the processor in the supervisor state. Therefore, when instruction execution resumes at the address specified to process the exception, the processor is in the supervisor privilege state. The transition from the supervisor to user state can be accomplished by any of four instructions: return from exception (RTE), move to status register (MOVE to SR), AND immediate to status register (ANDI to SR), and exclusive OR immediate to status register (EORI to SR).

2.4 EXCEPTION PROCESSING

The processing of an exception occurs in four steps, with variations for different exception causes. During the first step, a temporary copy of the SR is made, and the SR is set for exception processing. During the second step, the exception vector is determined; during the third step, the current processor context is saved. During the fourth step, a new context is obtained, and the processor switches to instruction processing.

2.4.1 Exception Vectors

Exception vectors are memory locations from which the processor fetches the address of a routine to handle that exception. All exception vectors are two words long except for the reset vector, which is four words. All exception vectors lie 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 which, when multiplied by four, gives the offset of the exception vector. Vector numbers are generated internally or externally, depending on the cause of the exception. In the case of interrupts, during the interrupt acknowledge bus cycle, a peripheral may provide an 8-bit vector number to the processor on data bus lines D7–D0. Alternatively, the peripheral may assert autovector ($\overline{\text{AVEC}}$) instead of data transfer acknowledge ($\overline{\text{DTACK}}$) to request an autovector for that priority level of interrupt. The exception vector assignments for the M68000 processor are shown in Table 2-5.

Table 2-5. M68000 Exception Vector Assignment

Vector Number	Decimal	Address Hex	Space	Assignment
0	0	000	SP	Reset: Initial SSP[2]
1	4	004	SP	Reset: Initial PC[2]
2	8	008	SD	Bus Error
3	12	00C	SD	Address Error
4	16	010	SD	Illegal Instruction
5	20	014	SD	Zero Divide
6	24	018	SD	CHK Instruction
7	28	01C	SD	TRAPV Instruction
8	32	020	SD	Privilege Violation
9	36	024	SD	Trace
10	40	028	SD	Line 1010 Emulator

Table 2-5. M68000 Exception Vector Assignment

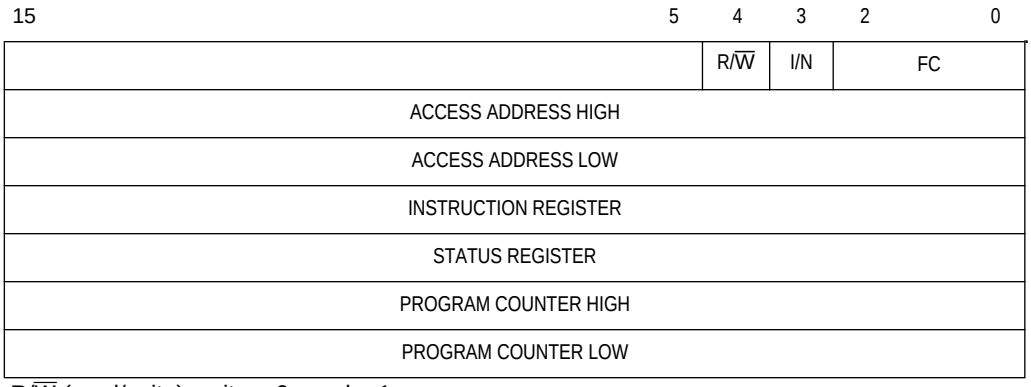
11	44	02C	SD	Line 1111 Emulator
12[1]	48	030	SD	(Unassigned, Reserved)
13[1]	52	034	SD	(Unassigned, Reserved)
14[1]	56	038	SD	(Unassigned, Reserved)
15	60	03C	SD	Uninitialized Interrupt Vector
16–23[1]	64	040	SD	(Unassigned, Reserved)
	92	05C	SD	
24	96	060	SD	Spurious Interrupt[3]
25	100	064	SD	Level 1 Interrupt Autovector
26	104	068	SD	Level 2 Interrupt Autovector
27	108	06C	SD	Level 3 Interrupt Autovector
28	112	070	SD	Level 4 Interrupt Autovector
29	116	074	SD	Level 5 Interrupt Autovector
30	120	078	SD	Level 6 Interrupt Autovector
31	124	07C	SD	Level 7 Interrupt Autovector
32–47	128	080	SD	TRAP Instruction Vectors[4]
	188	0BC	SD	
48–63[1]	192	0C0	SD	(Unassigned, Reserved)
	255	0FC	SD	
64–255	256	100	SD	User Interrupt Vectors
	1020	3FC	SD	

NOTES:

1. Vector numbers 12–14, 16–23, and 48–63 are reserved for future enhancements by Motorola (with vectors 60–63 being used by the M68302 (see 2.7 MC68302 IMP Configuration and Control)). No user peripheral devices should be assigned these numbers.
2. Unlike the other vectors which only require two words, reset vector (0) requires four 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.

2.4.2 Exception Stacking Order

Exception processing saves the most volatile portion of the current processor context on top of the supervisor stack. This context is organized in a format called the exception stack frame. The amount and type of information saved on the stack is determined by the type of exception. The reset exception causes the M68000 to halt current execution and to read a new SSP and PC as shown in Table 2-5. A bus error or address error causes the M68000 to store the information shown in Figure 2-3. The interrupts, traps, illegal instructions, and trace stack frames are shown in Figure 2-4.



R/$\overline{W}$ (read/write): write = 0, read = 1
I/N (instruction not): instruction = 0, not =1
FC: Function Code

Figure 2-3. M68000 Bus/Address Error Exception Stack Frame

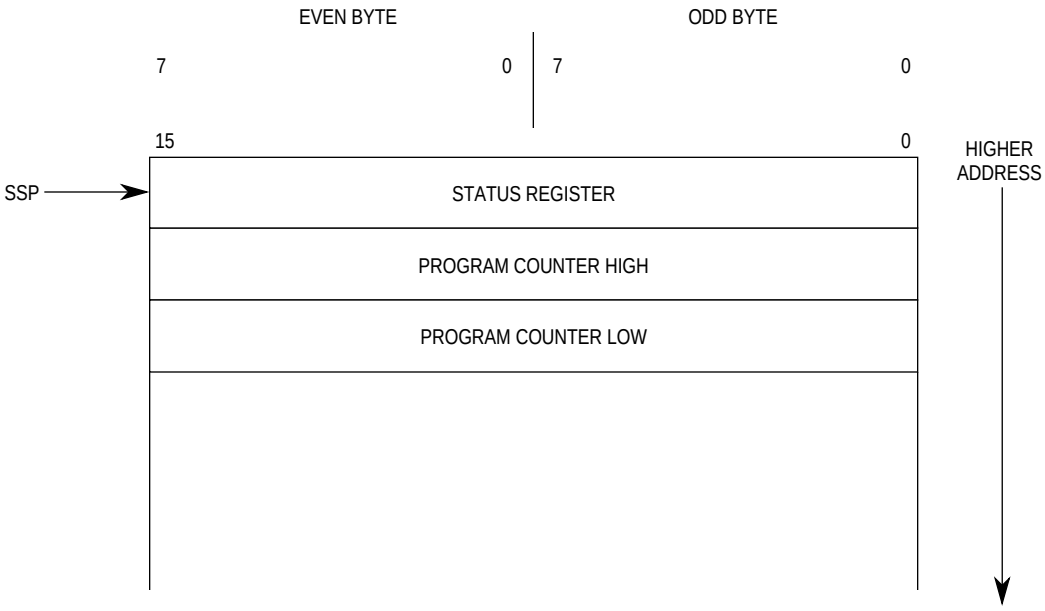


Figure 2-4. M68000 Short-Form Exception Stack Frame

NOTE

The MC68302 uses the exact same exception stack frames as the MC68000.

For exception processing times and instruction execution times, refer to MC68000UM/AD, *8-/16-/32-Bit Microprocessor User's Manual*.

2.5 INTERRUPT PROCESSING

Seven interrupt levels are provided by the M68000 core. If the IMP's interrupt controller is placed in the normal mode, six levels are available to the user. If the interrupt controller is in the dedicated mode, three levels are available to the user. In either mode, level 4 is reserved for the on-chip peripherals. Devices may be chained externally within one of the available priority levels, allowing an unlimited number of external peripheral devices to interrupt the processor. The SR contains a 3-bit mask indicating the current processor priority level. Interrupts are inhibited for all priority levels less than or equal to the current processor priority (see Figure 2-2).

An interrupt request is made to the processor by encoding the request on the interrupt request lines (normal mode) or by asserting the appropriate request line (dedicated mode). Rather than forcing immediate exception processing, interrupt requests arriving at the processor are made pending to be 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.

If the priority of the pending interrupt is greater than the current processor priority, the exception processing sequence is started. A copy of the SR is saved, the privilege state is set to supervisor state, 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, classifying the reference as an interrupt acknowledge on the address bus. If external logic requests automatic vectoring (via the $\overline{\text{AVEC}}$ pin), the processor internally generates a vector number determined by 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 number.

2.6 M68000 SIGNAL DIFFERENCES

The MC68302 core supports one additional signal not visible on the standard M68000: $\overline{\text{RMC}}$. Asserted externally on read-modify-write cycles, the $\overline{\text{RMC}}$ signal is typically used as a bus lock to ensure integrity of instructions using the locked read-modify-write operation of the test and set (TAS) instruction. The $\overline{\text{RMC}}$ signal from the M68000 core is applied to the MC68302 arbiter and can be programmed to prevent the arbiter from issuing bus grants until the completion of an MC68000-core-initiated read-modify-write cycle.

The MC68302 can be programmed to use the $\overline{\text{RMC}}$ signal to negate address strobe ($\overline{\text{AS}}$) at the end of the read portion of the cycle and assert $\overline{\text{AS}}$ at the beginning of the write portion of the cycle (See 3.8.3 System Control Bits).

Two M6800 signals are omitted from the MC68302: valid memory address ($\overline{\text{VMA}}$) and enable (E). The valid peripheral address ($\overline{\text{VPA}}$) signal is retained, but is only used on the MC68302 as $\overline{\text{AVEC}}$ to direct the core to use an autovector during interrupt acknowledge cycles.

2.7 MC68302 IMP CONFIGURATION CONTROL

Four reserved entries in the external M68000 exception vector table (see Table 2-5) are used as addresses for internal system configuration registers. These entries are at locations $0F0, $0F4, $0F8, and $0FC. The first entry is the on-chip peripheral base address register (BAR) entry; the second is the on-chip system control register (SCR) entry; the least significant word of the third entry is the clock control register (CKCR), and fourth entry is reserved for future use.

The BAR entry contains the BAR described in this section. The SCR entry contains the SCR described in 3.8.1 System Control Register (SCR). The CKCR entry contains the CKCR register described in 3.9 Clock Control Register.

Figure 2-5 shows all the MC68302 IMP on-chip addressable locations and how they are mapped into system memory.

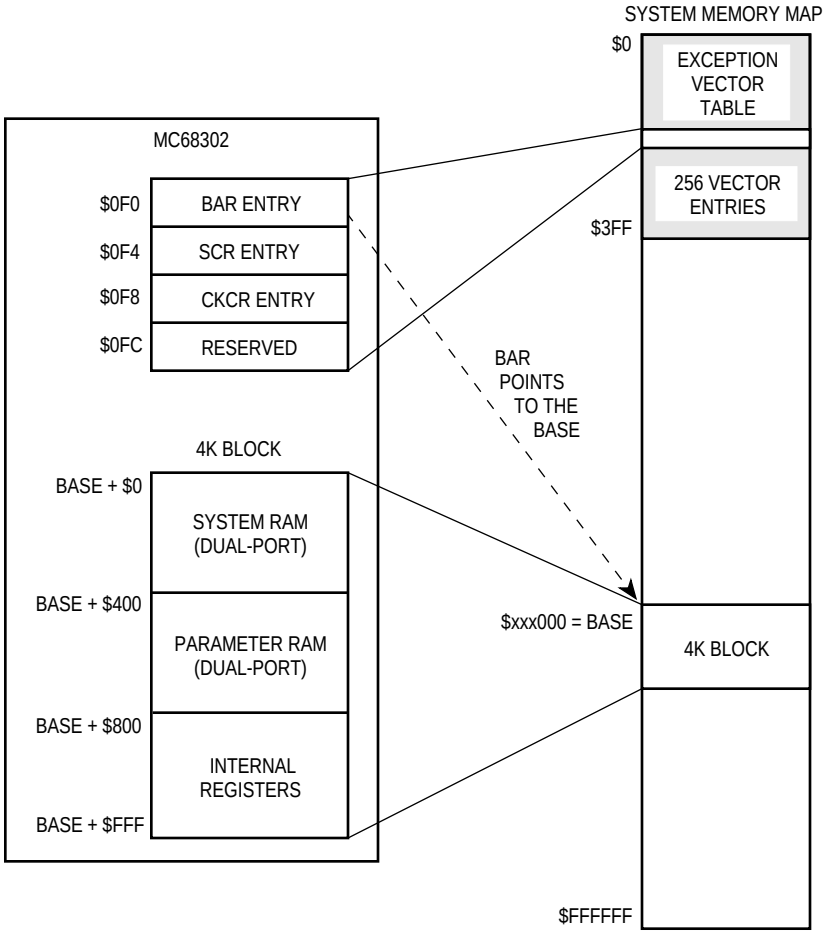


Figure 2-5. MC68302 IMP Configuration Control

The on-chip peripherals, including those peripherals in both the CP and SIB, require a 4K-byte block of address space. This 4K-byte block location is determined by writing the intended base address to the BAR in supervisor data space (FC = 5). The address of the BAR en-

try is $0F0; however, the actual BAR is a 16-bit value within the BAR entry and is located at $0F2.

After a total system reset, the on-chip peripheral base address is undefined, and it is not possible to access the on-chip peripherals at any address until BAR is written. The BAR and the SCR can always be accessed at their fixed addresses.

NOTE

The BAR, SCR and CKCR registers are internally reset only when a total system reset occurs by the simultaneous assertion of RESET and HALT. The chip-select (CS) lines are not asserted on accesses to these locations. Thus, it is very helpful to use CS lines to select external ROM/RAM that overlaps the BAR and SCR register locations, since this prevents potential bus contention. (The internal access (IAC) signal may also be used to prevent bus contention.)

NOTE

In 8-bit system bus operation, IMP accesses are not possible until the low byte of the BAR is written. Since the MOVE.W instruction writes the high byte followed by the low byte, this instruction guarantees the entire word is written.

Do not assign other devices on the system bus an address that falls within the address range of the peripherals defined by the BAR. If this happens, $\overline{\text{BERR}}$ is generated (if the address decode conflict enable (ADCE) bit is set) and the address decode conflict (ADC) bit in the SCR is set.

The BAR is a 16-bit, memory-mapped, read-write register consisting of the high address bits, the compare function code bit, and the function code bits. Upon a total system reset, its value may be read as $BFFF, but its value is not valid until written by the user. The address of this register is fixed at $0F2 in supervisor data space. BAR cannot be accessed in user data space.

15 13	12	11											0
FC2–FC0	CFC	BASE ADDRESS											
		23	22	21	20	19	18	17	16	15	14	13	12

Bits 15–13—FC2–FC0

The FC2–FC0 field is contained in bits 15–13 of the BAR. These bits are used to set the address space of 4K-byte block of on-chip peripherals. The address compare logic uses these bits, dependent upon the CFC bit, to cause an address match within its address space.

NOTE

Do not assign this field to the M68000 core interrupt acknowledge space (FC2–FC0 = 7).

CFC—Compare Function Code

0 = The FC bits in the BAR are ignored. Accesses to the IMP 4K-byte block occur without comparing the FC bits.
1 = The FC bits in the BAR are compared. The address space compare logic uses the FC bits to detect address matches.

Bits 11–0—Base Address

The high address field is contained in bit 11–0 of the BAR. These bits are used to set the starting address of the dual-port RAM. The address compare logic uses only the most significant bits to cause an address match within its block size.

2.8 MC68302 MEMORY MAP

The following tables show the additional registers added to the M68000 to make up the MC68302. All of the registers are memory-mapped. Four entries in the M68000 exception vectors table (located in low RAM) are reserved for addresses of system configuration registers (see Table 2-6) that reside on-chip. These registers have fixed addresses of $0F0–$0FF. All other on-chip peripherals occupy a 4K-byte relocatable address space. When an on-chip register or peripheral is accessed, the internal access (IAC) pin is asserted.

Table 2-6. System Configuration Register

Address	Name	Width	Description	Reset Value
$0F0	RES	16	Reserved	
$0F2*	BAR	16	Base Address Register	BFFF
$0F4*	SCR	32	System Control Register	0000 0F00
$0F8	RES	16	Reserved	
$0FA	CKCR	16	Clock Control Register	0000
$0FC	RES	32	Reserved	

*Reset only upon a total system reset.

The internal 1176-byte dual-port RAM has 576 bytes of system RAM (see Table 2-7) and 576 bytes of parameter RAM (see Table 2-8).

Table 2-7. System RAM

Address	Width	Block	Description
Base + 000 • • • Base + 23F	576 Bytes	RAM	User Data Memory
Base +240 • • • Base + 3FF			Reserved (Not Implemented)

The parameter RAM contains the buffer descriptors for each of the three SCC channels, the SCP, and the two SMC channels. The memory structures of the three SCC channels are

identical. When any SCC, SCP, or SMC channel buffer descriptors or parameters are not used, their parameter RAM area can be used for additional memory. For detailed information about the use of the buffer descriptors and protocol parameters in a specific protocol, see 4.5 Serial Communication Controllers (SCCs). Base + 67E contains the MC68302 revision number. Revision A parts (mask 1B14M) correspond to the value $0001. Revision B parts (mask 2B14M and 3B14M which are described in this manual) correspond to the value $0002. Revision C and D parts have revision number $0003.

Table 2-8. Parameter RAM

Address	Width	Block	Description
Base + 400	4 Word	SCC1	Rx BD 0
Base + 408	4 Word	SCC1	Rx BD 1
Base + 410	4 Word	SCC1	Rx BD 2
Base + 418	4 Word	SCC1	Rx BD 3
Base + 420	4 Word	SCC1	Rx BD 4
Base + 428	4 Word	SCC1	Rx BD 5
Base + 430	4 Word	SCC1	Rx BD 6
Base + 438	4 Word	SCC1	Rx BD 7
Base + 440	4 Word	SCC1	Tx BD 0
Base + 448	4 Word	SCC1	Tx BD 1
Base + 450	4 Word	SCC1	Tx BD 2
Base + 458	4 Word	SCC1	Tx BD 3
Base + 460	4 Word	SCC1	Tx BD 4
Base + 468	4 Word	SCC1	Tx BD 5
Base + 470	4 Word	SCC1	Tx BD 6
Base + 478	4 Word	SCC1	Tx BD 7
Base + 480 • • • Base + 4BF		SCC1 SCC1	Specific Protocol Parameters
Base + 4C0 • • • Base + 4FF			Reserved (Not Implemented)
Base + 500	4 Word	SCC2	Rx BD 0
Base + 508	4 Word	SCC2	Rx BD 1
Base + 510	4 Word	SCC2	Rx BD 2
Base + 518	4 Word	SCC2	Rx BD 3
Base + 520	4 Word	SCC2	Rx BD 4
Base + 528	4 Word	SCC2	Rx BD 5
Base + 530	4 Word	SCC2	Rx BD 6
Base + 538	4 Word	SCC2	Rx BD 7
Base + 540	4 Word	SCC2	Tx BD 0
Base + 548	4 Word	SCC2	Tx BD 1
Base + 550	4 Word	SCC2	Tx BD 2
Base + 558	4 Word	SCC2	Tx BD 3
Base + 560	4 Word	SCC2	Tx BD 4
Base + 568	4 Word	SCC2	Tx BD 5
Base + 570	4 Word	SCC2	Tx BD 6/DRAM Refresh
Base + 578	4 Word	SCC2	Tx BD 7/DRAM Refresh

Table 2-8. Parameter RAM

Base + 580 • • • Base + 5BF		SCC2 SCC2	Specific Protocol Parameters
Base + 5C0 • • • Base + 5FF			Reserved (Not Implemented)
Base + 600 Base + 608 Base + 610 Base + 618 Base + 620 Base + 628 Base + 630 Base + 638 Base + 640 Base + 648 Base + 650 Base + 658 Base + 660 Base + 666 Base + 668 Base + 66A Base + 66C Base + 66E # Base +67A Base +67C Base +67E #	4 Word 4 Word 4 Word 4 Word 4 Word 4 Word 4 Word 4 Word 4 Word 4 Word 4 Word 4 Word 3 Word Word Word Word Word 6 Word Word Word Word	SCC3 SCC3 SCC3 SCC3 SCC3 SCC3 SCC3 SCC3 SCC3 SCC3 SCC3 SCC3 SMC SCC1 SCC1 SCC2 SCC2 SMC1–SMC2 SCP SCC1–SCC3 CP	RxBD0 RxBD1 RxBD2 RxBD3 RxBD4 RxBD5 RxBD6 RxBD7 TxBD0 TxBD1 TxBD2 TxBD3 ## Reserved RxBD TxBD RxBD TxBD Internal Use Rx/TxBD BERR Channel Number MC68302 Revision Number
Base + 680 • • • Base + 6BF		SCC3 SCC3	Specific Protocol Parameters
Base + 6C0 • • • Base + 7FF			Reserved (Not Implemented)

Modified by the CP after a CP or system reset.

Tx BD 4, 5, 6, and 7 are not initially available to SCC3. (See 4.5.5 Buffer Descriptors Table for information on how they may be regained.)

In addition to the internal dual-port RAM, a number of internal registers support the functions of the various M68000 core peripherals. The internal registers (see Table 2-9) are memory-mapped registers offset from the BAR point1616er and are located on the internal M68000 bus.

NOTE

All undefined and reserved bits within registers and parameter RAM values written by the user in a given application should be written with zero to allow for future enhancements to the device.

Table 2-9. Internal Registers

Address	Name	Width	Block	Description	Reset Value
Base + 800	RES	16	IDMA	Reserved	
Base + 802	CMR	16	IDMA	Channel Mode Register	0000
Base + 804	SAPR	32	IDMA	Source Address Pointer	XXXX XXXX
Base + 808	DAPR	32	IDMA	Destination Address Pointer	XXXX XXXX
Base + 80C	BCR	16	IDMA	Byte Count Register	XXXX
! Base + 80E	CSR	8	IDMA	Channel Status Register	00
Base + 80F	RES	8	IDMA	Reserved	
Base + 810	FCR	8	IDMA	Function Code Register	XX
Base + 811	RES	8	IDMA	Reserved	
Base + 812 #	GIMR	16	Int Cont	Global Interrupt Mode Register	0000
! Base + 814	IPR	16	Int Cont	Interrupt Pending Register	0000
Base + 816	IMR	16	Int Cont	Interrupt Mask Register	0000
! Base + 818	ISR	16	Int Cont	In-Service Register	0000
Base + 81A	RES	16	Int Cont	Reserved	
Base + 81C	RES	16	Int Cont	Reserved	
Base + 81E #	PACNT	16	PIO	Port A Control Register	0000
Base + 820 #	PADDR	16	PIO	Port A Data Direction Register	0000
Base + 822 #	PADAT	16	PIO	Port A Data Register	XXXX ##
Base + 824 #	PBCNT	16	PIO	Port B Control Register	0080
Base + 826 #	PBDDR	16	PIO	Port B Data Direction Register	0000
Base + 828 #	PBDAT	16	PIO	Port B Data Register Reserved	XXXX ##
Base + 82A	RES	16	PIO		
Base + 82C	RES	16	CS	Reserved	
Base + 82E	RES	16	CS	Reserved	
Base + 830 #	BR0	16	CS0	Base Register 0	C001
Base + 832 #	OR0	16	CS0	Option Register 0	DFFD
Base + 834 #	BR1	16	CS1	Base Register 1	C000
Base + 836 #	OR1	16	CS1	Option Register 1	DFFD
Base + 838 #	BR2	16	CS2	Base Register 2	C000
Base + 83A #	OR2	16	CS2	Option Register 2	DFFD
Base + 83C #	BR3	16	CS3	Base Register 3	C000
Base + 83E #	OR3	16	CS3	Option Register 3	DFFD

Table 2-9. Internal Registers

Base + 840	TMR1	16	Timer	Timer Unit 1 Mode Register	0000
Base + 842	TRR1	16	Timer	Timer Unit 1 Reference Register	FFFF
Base + 844	TCR1	16	Timer	Timer Unit 1 Capture Register	0000
Base + 846	TCN1	16	Timer	Timer Unit 1 Counter	0000
Base + 848	RES	8	Timer	Reserved	
! Base + 849	TER1	8	Timer	Timer Unit 1 Event Register	00
Base + 84A	WRR	16	WD	Watchdog Reference Register	FFFF
Base + 84C	WCN	16	WD	Watchdog Counter	0000
Base + 84E	RES	16	Timer	Reserved	
Base + 850	TMR2	16	Timer	Timer Unit 2 Mode Register	0000
Base + 852	TRR2	16	Timer	Timer Unit 2 Reference Register	FFFF
Base + 854	TCR2	16	Timer	Timer Unit 2 Capture Register	0000
Base + 856	TCN2	16	Timer	Timer Unit 2 Counter	0000
Base + 858	RES	8	Timer	Reserved	
! Base + 859	TER2	8	Timer	Timer Unit 2 Event Register	00
Base + 85A	RES	16	Timer	Reserved	
Base + 85C	RES	16	Timer	Reserved	
Base + 85E	RES	16	Timer	Reserved	
Base + 860	CR	8	CP	Command Register	00
Base + 861 • • • Base + 87F				Reserved (Not Implemented)	
Base + 880	RES	16	SCC1	Reserved	
Base + 882	SCON1	16	SCC1	SCC1 Configuration Register	0004
Base + 884	SCM1	16	SCC1	SCC1 Mode Register	0000
Base + 886	DSR1	16	SCC1	SCC1 Data Sync. Register	7E7E
! Base + 888	SCCE1	8	SCC1	SCC1 Event Register	00
Base + 889	RES	8	SCC1	Reserved	
Base + 88A	SCCM1	8	SCC1	SCC1 Mask Register	00
Base + 88B	RES	8	SCC1	Reserved	
Base + 88C	SCCS1	8	SCC1	SCC1 Status Register	00
Base + 88D	RES	8	SCC1	Reserved	
Base + 88E	RES	16	SCC1	Reserved	
Base + 890	RES	16	SCC2	Reserved	
Base + 892	SCON2	16	SCC2	SCC2 Configuration Register	0004
Base + 894	SCM2	16	SCC2	SCC2 Mode Register	0000
Base + 896	DSR2	16	SCC2	SCC2 Data Sync. Register	7E7E
! Base + 898	SCCE2	8	SCC2	SCC2 Event Register	00
Base + 899	RES	8	SCC2	Reserved	
Base + 89A	SCCM2	8	SCC2	SCC2 Mask Register	00
Base + 89B	RES	8	SCC2	Reserved	
Base + 89C	SCCS2	8	SCC2	SCC2 Status Register	00
Base + 89D	RES	8	SCC2	Reserved	
Base + 89E	RES	16	SCC2		

Table 2-9. Internal Registers

Base + 8A0	RES	16	SCC3	Reserved	
Base + 8A2	SCON3	16	SCC3	SCC3 Configuration Register	0004
Base + 8A4	SCM3	16	SCC3	SCC3 Mode Register	0000
Base + 8A6	DSR3	16	SCC3	SCC3 Data Sync. Register	7E7E
! Base + 8A8	SCCE3	8	SCC3	SCC3 Event Register	00
Base + 8A9	RES	8	SCC3	Reserved	
Base + 8AA	SCCM3	8	SCC3	SCC3 Mask Register	00
Base + 8AB	RES	8	SCC3	Reserved	
Base + 8AC	SCCS3	8	SCC3	SCC3 Status Register	00
Base + 8AD	RES	8	SCC3	Reserved	
Base + 8AE	RES	16	SCC3	Reserved	
Base + 8B0	SPMODE	16	SCM	SCP, SMC Mode and Clock Control Register	0000
Base + 8B2 #	SIMASK	16	SI	Serial Interface Mask Register	FFFF
Base + 8B4 #	SIMODE	16	SI	Serial Interface Mode Register	0000
Base + 8B6 • • • Base + FFF				Reserved (Not Implemented)	

\# Reset only upon a total system reset ($\overline{\text{RESET}}$ and $\overline{\text{HALT}}$ assert together), but not on the execution of an M68000 RESET instruction. See the RESET pin description for details.

The output latches are undefined at total system reset.

! Event register with special properties (see 2.9 Event Registers).

2.9 EVENT REGISTERS

The IMP contains a few special registers designed to report events to the user. They are the channel status register (CSR) in the independent DMA, the interrupt pending register (IPR) and interrupt in-service register (ISR) in the interrupt controller, the timer event register 1 (TER1) in timer 1, the TER2 in timer 2, serial communication controller event register 1 (SCCE1) in SCC1, SCCE2 in SCC2, and SCCE3 in SCC3. Events in these register are always reported by a bit being set.

During the normal course of operation, the user software will clear these events after recognizing them. To clear a bit in one of these registers, the user software must WRITE A ONE TO THAT BIT. Writing a zero has no effect on the register. Thus, in normal operation, the hardware only *sets* bits in these registers; whereas, the software only *clears* them.

This technique prevents software from inadvertently losing the indication from an event bit that is “set” by the hardware between the software read and the software write of this register.

All these registers are cleared after a total system reset ($\overline{\text{RESET}}$ and $\overline{\text{HALT}}$ asserted together) and after the M68000 RESET instruction. Also some of the blocks (IDMA, timer1, timer2, and communication processor) have a reset (RST) bit located in a register in that block. This RST bit will reset that entire block, including any event registers contained therein.

Examples:

1. To clear bit 0 of SCCE1, execute "MOVE.B #$01,SCCE1"

2. To clear bits 0 and 1 of SCC1, execute "MOVE.B #$03,SCCE1”
3. To clear all bits in SCCE1, execute "MOVE.B #$ff,SCCE1"

where SCCE1 is equated to the actual address of SCCE1.

NOTE

DO NOT use read-modify-write instructions to clear bits in an event register, or ALL bits in that register will inadvertently be cleared. Read-modify-write instructions include BSET, BCLR, AND, OR, etc. These instructions read the contents of a location, perform an operation, and write the result back, leaving the rest of the bits unchanged. Thus, if a bit is a one when read, it will be written back with a one, clearing that bit. For example, the instruction “BSET.B #0,SCCE1” will actually clear ALL bits in SCCE1, not just bit 0.

SECTION 3
SYSTEM INTEGRATION BLOCK (SIB)

The MC68302 contains an extensive SIB that simplifies the job of both the hardware and software designer. It integrates the M68000 core with the most common peripherals used in an M68000-based system. The independent direct memory access (IDMA) controller relieves the hardware designer of the extra effort and board logic needed to connect an external DMA controller. The interrupt controller can be used in a dedicated mode to generate interrupt acknowledge signals without external logic. Also, the chip-select signals and their associated wait-state logic eliminate the need to generate chip-select signals externally. The three timers simplify control and improve reliability. These and other features in the SIB conserve board space and cost while decreasing development time.

The SIB includes the following functions:

- IDMA Controller with Three Handshake Signals: $\overline{\text{DREQ}}$, $\overline{\text{DACK}}$, and $\overline{\text{DONE}}$
- Interrupt Controller with Two Modes of Operation
- Parallel Input/Output (I/O) Ports, Some with Interrupt Capability
- On-Chip 1152-Byte Dual-Port RAM
- Three Timers Including a Software Watchdog Timer
- Four Programmable Chip-Select Lines with Wait-State Generator Logic
- On-Chip Clock Generator with Output Signal
- System Control

  —System Status and Control Logic
  —Disable CPU Logic (M68000)
  —Bus Arbitration Logic with Low-Interrupt Latency Support
  —Hardware Watchdog for Monitoring Bus Activity
  —Low-Power (Standby) Modes
  —Freeze Control for Debugging
- Clock Control

  —Adjustable CLKO Drive
  —Three-state RCLK1 and TCLK1
  —Disable BRG1
- DRAM Refresh Controller

3.1 DMA CONTROL

The IMP includes seven on-chip DMA channels, six serial DMA (SDMA) channels for the three serial communications controllers (SCCs) and one IDMA. The SDMA channels are discussed in 4.2 SDMA Channels. The IDMA is discussed in the following paragraphs.

3.1.1 Key Features

The IDMA (Independent DMA Controller) has the following key features:

- Two Address Pointers and One Counter
- Support of Memory-to-Memory, Peripheral-to-Memory, and Memory-to-Peripheral Data Transfers
- Three I/O Lines, $\overline{\text{DREQ}}$, $\overline{\text{DACK}}$, and $\overline{\text{DONE}}$, for Externally Requested Data Transfers
- Asynchronous M68000 Bus Structure with 24-Bit Address and 8-Bit or 16-Bit Data Bus
- Support for Data Blocks Located at Even or Odd Addresses
- Packing and Unpacking of Operands
- Fast Transfer Rates: Up to 4M bytes/second at 16.0 MHz with No Wait States
- Full Support of All M68000 Bus Exceptions: Halt, Bus Error, Reset, and Retry
- Flexible Request Generation:

  —Internal, Maximum Rate (One Burst)
  —Internal, Limited Rate (Limited Burst Bandwidth)
  —External, Burst ($\overline{\text{DREQ}}$ Level Sensitive)
  —External, Cycle Steal ($\overline{\text{DREQ}}$ Edge Sensitive)

The one general-purpose IDMA controller can operate in different modes of data transfer as programmed by the user. The IDMA is capable of transferring data between any combination of memory and I/O. In addition, data may be transferred in either byte or word quantities, and the source and destination addresses may be either odd or even. Note that the chip select and wait state generation logic on the MC68302 may be used with the IDMA, if desired.

Every IDMA cycle requires between two and four bus cycles, depending on the address boundary and transfer size. Each bus cycle is a standard M68000-type read or write cycle. If both the source and destination addresses are even, the IDMA fetches one word of data and immediately deposits it. If either the source or destination address begins on an odd boundary, the transfer is handled differently. For example, if the source address starts on an odd boundary and the destination address is even, the IDMA reads one byte from the source, then reads the second byte from the source, and finally stores the word in a single access. If the source is even and the destination odd, then the IDMA will read one word from the source and store it in two consecutive cycles. If both the source and destination are odd, the IDMA performs two read byte cycles followed by two write byte cycles until the transfer is complete.

If the IMP frequency is 16.0 MHz and zero wait state memory is used, then the maximum transfer rate is 4M byte/sec. This assumes that the operand size is 16-bits, the source and destination addresses are even, and the bus width is selected to be 16-bits.

The maximum transfer rate is calculated from the fact that 16 bits are moved every 8 clocks. The calculation is as follows:

$$\frac{\text{16 bits x 16M clocks/sec}}{\text{(2 bus cycles) x (4 clocks/bus cycle)}} = \frac{\text{2 bytes x 16M clocks/sec}}{\text{8 clocks}} = \frac{\text{4M bytes}}{\text{sec}}$$

The IDMA controller block diagram is shown in Figure 3-1.

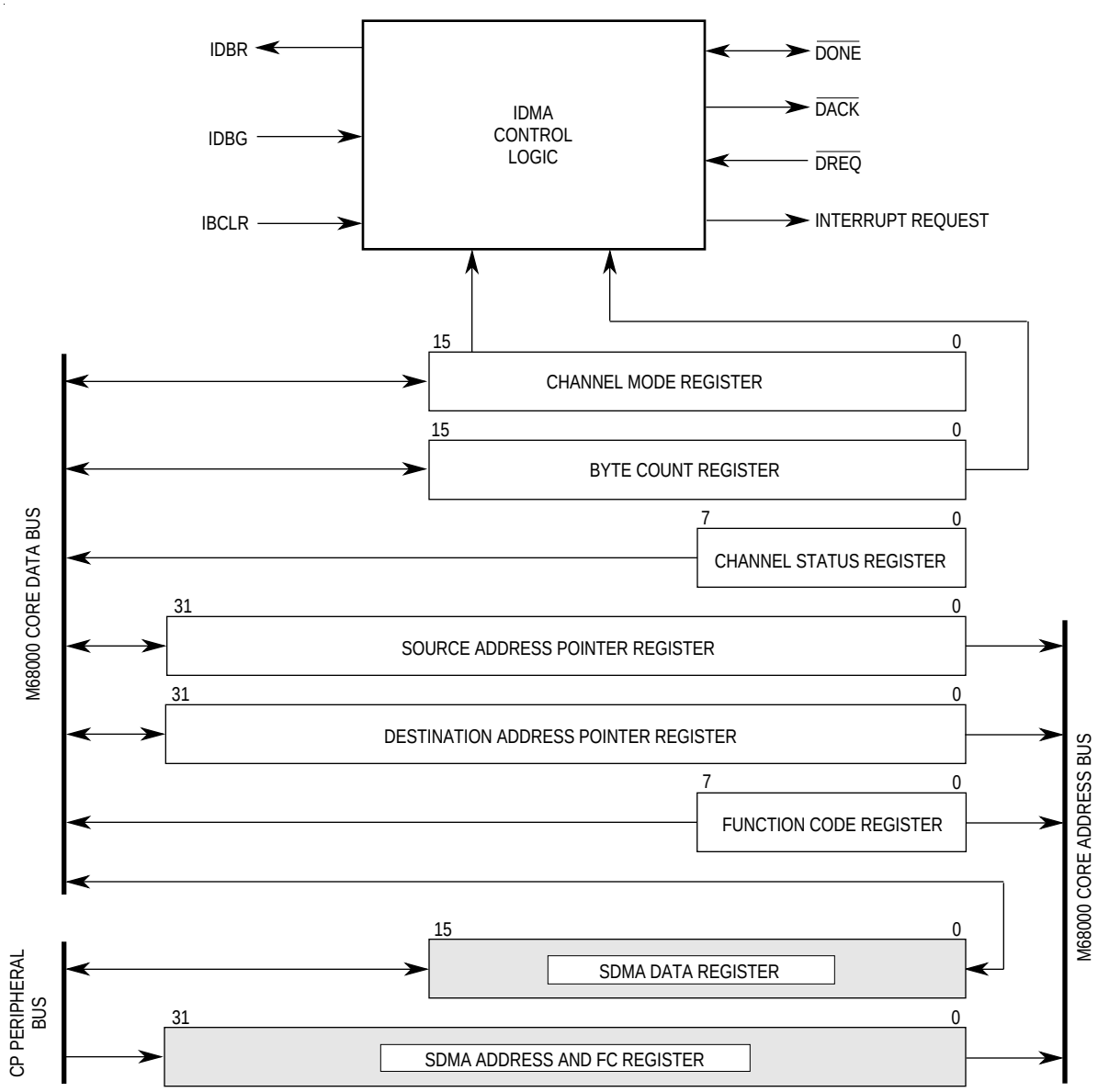


Figure 3-1. IDMA Controller Block Diagram

3.1.2 IDMA Registers (Independent DMA Controller)

The IDMA has six registers that define its specific operation. These registers include a 32-bit source address pointer register (SAPR), a 32-bit destination address pointer register

(DAPR), an 8-bit function code register (FCR), a 16-bit byte count register (BCR), a 16-bit channel mode register (CMR), and an 8-bit channel status register (CSR). These registers provide the addresses, transfer count, and configuration information necessary to set up a transfer. They also provide a means of controlling the IDMA and monitoring its status. All registers can be modified by the M68000 core. The IDMA also includes another 16-bit register, the data holding register (DHR), which is not accessible to the M68000 core and is used by the IDMA for temporary data storage.

3.1.2.1 Channel Mode Register (CMR)

The CMR, a 16-bit register, is reset to $0000.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	ECO	INTN	INTE	REQG		SAPI	DAPI	SSIZE		DSIZE		BT		RST	STR

Bit 15—Reserved for future use.

ECO—External Control Option

0 = If the request generation is programmed to be external in the REQG bits, the control signals ($\overline{\text{DACK}}$ and $\overline{\text{DONE}}$) are used in the source (read) portion of the transfer since the peripheral is the source.

1 = If the request generation is programmed to be external in the REQG bits, the control signals ($\overline{\text{DACK}}$ and $\overline{\text{DONE}}$) are used in the destination (write) portion of the transfer since the peripheral is the destination.

INTN—Interrupt Normal

0 = When the channel has completed an operand transfer without error conditions as indicated by $\overline{\text{DONE}}$, the channel does not generate an interrupt request to the IMP interrupt controller. The DONE bit remains set in the CSR.

1 = When the channel has completed an operand transfer without error conditions as indicated by $\overline{\text{DONE}}$, the channel generates an interrupt request to the IMP interrupt controller and sets DONE in the CSR.

NOTE

An interrupt will only be generated if the IDMA bit is set in the interrupt mask register (IMR).

INTE—Interrupt Error

0 = If a bus error occurs during an operand transfer either on the source read (BES) or the destination write (BED), the channel does not generate an interrupt to the IMP interrupt controller. The appropriate bit remains set in the CSR.

1 = If a bus error occurs during an operand transfer either on BES or BED, the channel generates an interrupt to the IMP interrupt controller and sets the appropriate bit (BES or BED) in the CSR.

NOTE

An interrupt will only be generated if the IDMA bit is set in the IMR.

REQG—Request Generation

The following decode shows the definitions for the REQG bits:

00 = Internal request at limited rate (limited burst bandwidth) set by burst transfer (BT) bits
01 = Internal request at maximum rate (one burst)
10 = External request burst transfer mode ($\overline{\text{DREQ}}$ level sensitive)
11 = External request cycle steal ($\overline{\text{DREQ}}$ edge sensitive)

SAPI—Source Address Pointer (SAP) Increment

0 = SAP is not incremented after each transfer.
1 = SAP is incremented by one or two after each transfer, according to the source size (SSIZE) bits and the starting address.

DAPI—Destination Address Pointer (DAP) Increment

0 = DAP is not incremented after each transfer.
1 = DAP is incremented by one or two after each transfer, according to the destination size (DSIZE) bits and the starting address.

SSIZE—Source Size

The following decode shows the definitions for the SSIZE bits.

00 = Reserved
01 = Byte
10 = Word
11 = Reserved

DSIZE—Destination Size

The following decode shows the definitions for the DSIZE bits.

00 = Reserved
01 = Byte
10 = Word
11 = Reserved

BT—Burst Transfer

The BT bits control the maximum percentage of the M68000 bus that the IDMA can use during each 1024 clock cycle period following the enabling of the IDMA. The IDMA runs for a consecutive number of cycles up to its burst transfer percentage if bus clear ($\overline{\text{BCLR}}$) is not asserted and the BCR is greater than zero. The following decode shows these percentages.

00 = IDMA gets up to 75% of the bus bandwidth.
01 = IDMA gets up to 50% of the bus bandwidth.
10 = IDMA gets up to 25% of the bus bandwidth.
11 = IDMA gets up to 12.5% of the bus bandwidth.

NOTE

These percentages are valid only when using internal limited request generation (REQG = 00).

RST—Software Reset

This bit will reset the IDMA to the same state as an external reset. The IDMA clears RST when the reset is complete.

0 = Normal operation
1 = The channel aborts any external pending or running bus cycles and terminates channel operation. Setting RST clears all bits in the CSR and CMR.

STR—Start Operation

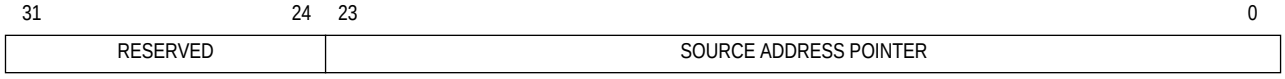
This bit starts the IDMA transfer if the REQG bits are programmed for an internal request. (The IDMA begins requesting the M68000 bus one clock after STR is set.) If the REQG bits are programmed for an external request, this bit must be set before the IDMA will recognize the first request on the $\overline{\text{DREQ}}$ input.

0 = Stop channel; clearing this bit will cause the IDMA to stop transferring data at the end of the current operand transfer. The IDMA internal state is not altered.
1 = Start channel; setting this bit will allow the IDMA to start (or continue if previously stopped) transferring data.

NOTE

STR is cleared automatically when the transfer is complete.

3.1.2.2 Source Address Pointer Register (SAPR)



The SAPR is a 32-bit register.

The SAPR contains 24 (A23–A0) address bits of the source operand used by the IDMA to access memory or memory-mapped peripheral controller registers. During the IDMA read cycle, the address on the master address bus is driven from this register. The SAPR may be programmed by the SAPI bit to be incremented or remain constant after each operand transfer.

The register is incremented using unsigned arithmetic and will roll over if an overflow occurs. For example, if a register contains $00FFFFFF and is incremented by one, it will roll over to $00000000. This register can be incremented by one or two, depending on the SSIZE bit and the starting address in this register.

3.1.2.3 Destination Address Pointer Register (DAPR)

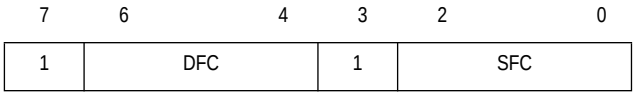
The DAPR is a 32-bit register.

The DAPR contains 24 (A23–A0) address bits of the destination operand used by the IDMA to access memory or memory-mapped peripheral controller registers. During the IDMA write cycle, the address on the master address bus is driven from this register. The DAPR may be programmed by the DAPI bit to be incremented or remain constant after each operand transfer.

The register is incremented using unsigned arithmetic and will roll over if overflow occurs. For example, if a register contains $00FFFFFF and is incremented by one, it will roll over to $00000000. This register can be incremented by one or two depending on the DSIZE bit and the starting address.

3.1.2.4 Function Code Register (FCR)

The FCR is an 8-bit register.



The SFC and the DFC bits define the source and destination function code values that are output by the IDMA and the appropriate address registers during an IDMA bus cycle. The address space on the function code lines may be used by an external memory management unit (MMU) or other memory-protection device to translate the IDMA logical addresses to proper physical addresses. The function code value programmed into the FCR is placed on pins FC2–FC0 during a bus cycle to further qualify the address bus value.

NOTE

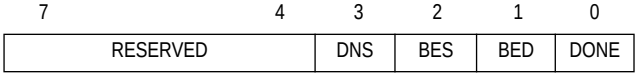
This register is undefined following power-on reset. The user should always initialize it and should not use the function code value “111” in this register.

3.1.2.5 Byte Count Register (BCR)

This 16-bit register specifies the amount of data to be transferred by the IDMA; up to 64K bytes (BCR = 0) is permitted. This register is decremented once for each byte transferred successfully. BCR may be even or odd as desired. DMA activity will terminate as soon as this register reaches zero. Thus, an odd number of bytes may be transferred in a 16-bit operand scenario.

3.1.2.6 Channel Status Register (CSR)

The CSR is an 8-bit register used to report events recognized by the IDMA controller. On recognition of an event, the IDMA sets its corresponding bit in the CSR (regardless of the INTE and INTN bits in the CMR). The CSR is a memory-mapped register which may be read at any time. A bit is cleared by writing a one and is left unchanged by writing a zero. More than one bit may be cleared at a time, and the register is cleared at reset.



Bits 7–4—These bits are reserved for future use.

DNS—Done Not Synchronized

This bit is set if operand packing is performed between 16-bit memory and an 8-bit peripheral and the $\overline{\text{DONE}}$ signal is asserted as an input to the IDMA (i.e., by the peripheral) during the first access of the 8-bit peripheral. In such a case, the IDMA will still attempt to finish the second access of the 8-bit peripheral even though $\overline{\text{DONE}}$ has been asserted (the access could be blocked with external logic); however, the DNS bit will be set to signify this condition. DNS will not be set if the transfer is terminated by an odd byte count, since, in this case, the exact number of requested bytes will be transferred by the IDMA.

BES—Bus Error Source

This bit indicates that the IDMA channel terminated with an error returned during the read cycle. The channel terminates the IDMA operation without setting DONE. BES is cleared by writing a one or by setting RST in the CMR. Writing a zero has no effect on BES.

BED—Bus Error Destination

This bit indicates that the IDMA channel terminated with an error during the write cycle. The channel terminates the IDMA operation without setting DONE. BED is cleared by writing a one or by setting RST in the CMR. Writing a zero has no effect on BED.

DONE—Normal Channel Transfer Done

This bit indicates that the IDMA channel has terminated normally. Normal channel termination is defined as 1) having decremented the BCR to zero with no errors occurring during any IDMA transfer bus cycle or 2) by the external peripheral asserting $\overline{\text{DONE}}$ with no errors occurring during any IDMA transfer bus cycle. DONE will not be set if the channel terminates due to an error. DONE is cleared by writing a one or by a software RST in the CMR. Writing a zero has no effect on this bit.

3.1.3 Interface Signals

The IDMA channel has three dedicated control signals: DMA request ($\overline{\text{DREQ}}$), DMA acknowledge ($\overline{\text{DACK}}$), and end of IDMA transfer ($\overline{\text{DONE}}$). The IDMA's use of the bus arbitration signals is described in 3.1.6 DMA Bus Arbitration. The peripheral used with these signals may be either a source or a destination of the transfers.

3.1.3.1 $\overline{\text{DREQ}}$ and $\overline{\text{DACK}}$

These are handshake signals between the peripheral requiring service and the IMP. When the peripheral requires IDMA service, it asserts $\overline{\text{DREQ}}$, and the IMP begins the IDMA process. When the IDMA service is in progress, $\overline{\text{DACK}}$ is asserted during accesses to the device. These signals are not used when the IDMA is programmed to internal request modes.

3.1.3.2 $\overline{\text{DONE}}$

This bidirectional signal is used to indicate the last IDMA transfer. With internal request modes, the IDMA activates $\overline{\text{DONE}}$ as an output during the last IDMA bus cycle. If DONE is externally asserted during internal request modes, the IDMA transfer is terminated. With external request modes, $\overline{\text{DONE}}$ may be used as an input to the IDMA controller indicating that the device being serviced requires no more transfers and that the transmission is to be terminated. $\overline{\text{DONE}}$ is an output if the transfer count is exhausted.

3.1.4 IDMA Operational Description

Every IDMA operation involves the following steps: IDMA channel initialization, data transfer, and block termination. In the initialization phase, the M68000 core (or external processor) loads the registers with control information, address pointers and transfer count, and then starts the channel. In the transfer phase, the IDMA accepts requests for operand transfers and provides addressing and bus control for the transfers. The termination phase occurs when the operation is complete and the IDMA interrupts the M68000 core, if interrupts are enabled.

3.1.4.1 Channel Initialization

To start a block transfer operation, the M68000 core must initialize IDMA registers with information describing the data block, device type, request generation method, and other special control options. See 3.1.2 IDMA Registers (Independent DMA Controller) and 3.1.5 IDMA Programming for further details.

3.1.4.2 Data Transfer

The IDMA supports dual address transfers only. Thus, each operand transfer consists of a source operand read and a destination operand write. The source operand is read from the address contained in the SAPR into the DHR. When the source and destination operand sizes differ, the operand read may take up to two bus cycles to complete. The operand is then written to the address contained in the DAPR. Again, this transfer may be up to two bus cycles long. In this manner, various combinations of peripheral, memory, and operand sizes may be used.

NOTE

When the SAPR and DAPR are programmed not to increment and the bus width is 16 bits, the SAPR and DAPR addresses must be even.

Source Operand Read

During this cycle, the SAPR drives the address bus, the FCR drives the source function codes, and the CMR drives the size control. The data is read from memory or the peripheral and placed temporarily into the data holding register (DHR) when the bus cycle is terminated with $\overline{\text{DTACK}}$. When the complete operand has been read, the SAPR is incremented by one or two, depending on the address and size information. See 3.1.2.2 Source Address Pointer Register (SAPR) for more details.

Destination Operand Write

During this cycle, the data in DHR is written to the device or memory selected by the address from the DAPR, using the destination function codes from the FCR and the size from the CMR. The same options exist for operand size and alignment as for the source operand read. When the complete operand is written, the DAPR is incremented by one or two, and the BCR is decremented by the number of bytes transferred. See 3.1.2.3 Destination Address Pointer Register (DAPR) and 3.1.2.5 Byte Count Register (BCR) for more details.

3.1.4.3 Address Sequencing

The manner in which the DAPR and SAPR are incremented during a transfer depends on the programming of the SAPI and DAPI bits, the source and destination sizes (DSIZE and SSIZE), and the system data bus width.

The IDMA will run at least two, and up to four, bus cycles to transfer each operand. With an 8-bit bus width, SSIZE and DSIZE are ignored, and each operand transfer requires two cycles. With a 16-bit bus width, the number of bus cycles required to transfer each operand is determined by DSIZE and SSIZE, whether the source and destination addresses are odd or even, and whether the BCR equals one. When SSIZE and DSIZE both select either a byte or word, there will be no operand packing, and the operand transfer will take two bus cycles. One exception occurs when DSIZE and SSIZE are words and the address is odd. In this case, there will be two (one byte each) memory cycles for each read or write at an odd address. When both the source and destination addresses are odd, four bus cycles are required to transfer each operand. When SSIZE and DSIZE are not equal, the IDMA will perform operand packing. If SSIZE is one byte, two read cycles are required to fetch the operand. If DSIZE is one byte, two write cycles are required to store the operand.

When SAPI and/or DAPI are programmed to increment either SAPR or DAPR, the amount (one or two) by which the address pointer increments depends upon DSIZE, SSIZE, and the bus width.

When operating in a 16-bit bus environment with an 8-bit peripheral, the peripheral may be placed on one-half of the bus (consecutive even or odd addresses only). In this case, SSIZE (or DSIZE) must be set to 16 bit, and the IDMA will perform data packing. As a result, the peripheral's addresses must be incremented twice after each peripheral bus cycle, which results in adding four to the address for each data transfer (two cycles per transfer). This is consistent with the M68000 MOVEP instruction. If the 8-bit peripheral is to be arranged with consecutive addresses, both SSIZE and DSIZE must be 8 bit.

Refer to Table 3-1 to see how the SAPR and DAPR will be incremented in all combinations.

Table 3-1. SAPR and DAPR Incrementing Rules

Bus Width	Source Size	Destination Size	SAPR Increment	DAPR Increment	Transfer Description
8 Bit	X	X	+1	+1	Read Byte—Write Byte Packing Is Not Possible
16 Bit	Byte	Byte	+1	+1	Read Byte—Write Byte Packing Is Not Desired
16 Bit	Byte	Word	+4	+2	Read Byte, Read Byte —Write Word Operand Packing
16 Bit	Word	Byte	+2	+4	Read Word—Write Byte, Write Byte Operand Unpacking
16 Bit	Word	Word	+2	+2	Read Word—Write Word

Refer toTable 3-2 for more details on the IDMA bus cycles.

Table 3-2. IDMA Bus Cycles

Source Size	Source Address	Destination Size	Destination Address	Cycles 8-bit Bus	Cycles 16-bit Bus
8	X	8	X	RW	RW
8	X	16	Even	RWRW*	RRW
8	X	16	Odd	RWRW*	RRWW
16	Even	8	X	RWRW*	RWW
16	Odd	8	X	RWRW*	RRWW
16	Even	16	Even	RWRW*	RW
16	Even	16	Odd	RWRW*	RWW
16	Odd	16	Even	RWRW*	RRW
16	Odd	16	Odd	RWRW*	RRWW

* - Considered as 2 operands.

3.1.4.4 Transfer Request Generation

IDMA transfers may be initiated by either internally or externally generated requests. Internally generated requests can be initiated by setting STR in the CMR. Externally generated transfers are those requested by an external device using $\overline{\text{DREQ}}$ in conjunction with the activation of STR.

Internal Maximum Rate

The first method of internal request generation is a nonstop transfer until the transfer count is exhausted. If this method is chosen, the IDMA will arbitrate for the bus and begin transferring data after STR is set and the IDMA becomes the bus master. If no exception occurs, all operands in the data block will be transferred in sequential bus cycles with the IDMA using 100 percent of the available bus bandwidth (unless an external bus master requests the bus or the M68000 core has an unmasked pending interrupt request and BCLM = 1). See 3.1.6 DMA Bus Arbitration for more details.

Internal Limited Rate

To guarantee that the IDMA will not use all the available system bus bandwidth during a transfer, internal requests can be limited to the amount of bus bandwidth allocated to the IDMA. Programming the REQG bits to “internal limited rate” and the BT bits to limit the percentage of bandwidth achieves this result. As soon as STR is set, the IDMA module arbitrates for the bus and begins to transfer data when it becomes bus master. If no exception occurs, transfers will continue uninterrupted, but the IDMA will not exceed the percentage of bus bandwidth programmed into the control register (12.5%, 25%, 50%, or 75%). This percentage is calculated over each ensuing 1024 internal clock cycle period.

For example, if 12.5% is chosen, the IDMA will attempt to use the bus for the first 128 clocks of each 1024 clock cycle period. However, because of other bus masters, the IDMA may not be able to take its 128 clock allotment in a single burst.

External Burst Mode

For external devices requiring very high data transfer rates, the external burst mode allows the IDMA to use all the bus bandwidth to service the device. In the burst mode, the

$\overline{\text{DREQ}}$ input to the IDMA is level-sensitive and is sampled at certain points to determine when a valid request is asserted by the device. The device requests service by asserting $\overline{\text{DREQ}}$ and leaving it asserted. In response, the IDMA arbitrates for the system bus and begins to perform an operand transfer. During each access to the device, the IDMA will assert $\overline{\text{DACK}}$ to indicate to the device that a request is being serviced. If $\overline{\text{DREQ}}$ remains asserted when the IDMA completes the peripheral cycle (the cycle during which $\overline{\text{DACK}}$ is asserted by the IDMA) one setup time (see specification. 80) before the S5 falling edge (i.e., before or with $\overline{\text{DTACK}}$), then a valid request for another operand transfer is recognized, and the IDMA will service the next request immediately. If $\overline{\text{DREQ}}$ is negated one setup time (see specification 80) before the S5 falling edge, a new request will not be recognized, and the IDMA will relinquish the bus.

NOTE:

If 8 to 16 bit packing occurs, then the $\overline{\text{DREQ}}$ is sampled during the last 8-bit cycle.

External Cycle Steal

For external devices that generate a pulsed signal for each operand to be transferred, the external cycle steal mode uses $\overline{\text{DREQ}}$ as a falling edge-sensitive input. The IDMA will respond to cycle-steal requests in the same manner as for all other requests. However, if subsequent $\overline{\text{DREQ}}$ pulses are generated before $\overline{\text{DACK}}$ is asserted in response to each request, they will be ignored. If $\overline{\text{DREQ}}$ is asserted after the IDMA asserts $\overline{\text{DACK}}$ for the previous request but one setup time (see specification 80) before the S5 falling edge, then the new request will be serviced before the bus is relinquished. If a new request has not been generated by one setup time (see specification 80) before the S5 falling edge, the bus will be released to the next bus master.

3.1.4.5 Block Transfer Termination

The user may stop the channel by clearing STR. Additionally, the channel operation can be terminated for any of the following reasons: transfer count exhausted, external device termination, or error termination. This is independent of how requests are generated to the IDMA.

Transfer Count Exhausted

When the channel begins an operand transfer, if the current value of the BCR is one or two (according to the operand size in the CMR), $\overline{\text{DONE}}$ is asserted during the last bus cycle to the device to indicate that the channel operation will be terminated when the current operand transfer has successfully completed. In the memory to memory case, $\overline{\text{DONE}}$ is asserted during the last access to memory (source or destination) as defined by the ECO bit. When the operand transfer has completed and the BCR has been decremented to zero, the channel operation is terminated, STR is cleared, and an interrupt is generated if INTN is set. The SAPR and/or DAPR are also incremented in the normal fashion.

NOTE

If the channel is started with BCR value set to zero, the channel will transfer 64K bytes.

External Device Termination

If desired, a transfer may be terminated by the device even before the BCR is decremented to zero. If $\overline{\text{DONE}}$ is asserted one setup time prior to the S5 falling edge (i.e., before or with $\overline{\text{DTACK}}$) during a device access, then the channel operation will be terminated following the operand transfer (see the DNS bit in the CSR). STR is cleared, and an interrupt is generated if INTN is set. The BCR is also decremented, and the SAPR and/or DAPR are incremented in the normal fashion. The use of DONE is not limited to external request generation only; it may also be used to externally terminate an internally generated IDMA transfer sequence.

Error Termination

When a fatal error occurs during an IDMA bus cycle, a bus error is used to abort the cycle and terminate the channel operation. STR is cleared, either BED or BES is set, and an error interrupt is generated if INTE is set.

3.1.5 IDMA Programming

Once the channel has been initialized with all parameters required for a transfer operation, it is started by setting the start operation (STR) bit in the CMR. After the channel has been started, any register that describes the current operation may be read but not modified (SAPR/DAPR, FCR, or BCR).

Once STR has been set, the channel is active and either accepts operand transfer requests in external mode or generates requests automatically in internal mode. When the first valid external request is recognized, the IDMA arbitrates for the bus. The $\overline{\text{DREQ}}$ input is ignored until STR is set.

STR is cleared automatically when the BCR reaches zero and the channel transfer is either terminated by $\overline{\text{DONE}}$ or the IDMA cycle is terminated by a bus error.

Channel transfer operation may be suspended at any time by clearing STR. In response, any operand transfer in progress will be completed, and the bus will be released. No further bus cycles will be started while STR remains negated. During this time, the M68000 core may access IDMA internal registers to determine channel status or to alter operation. When STR is set again, if a transfer request is pending, the IDMA will arbitrate for the bus and continue normal operation.

Interrupt handling for the IDMA is configured globally through the interrupt pending register (IPR), the IMR, and the interrupt in-service register (ISR). Within the CMR in the IDMA, two bits are used to either mask or enable the presence of an interrupt reported in the CSR of the IDMA. One bit is used for masking normal termination; the other bit is used for masking error termination. When these interrupt mask bits in the CMR (INTN and INTE) are cleared and the IDMA status changes, status bits are set in the CSR but not in the IPR. When either INTN or INTE is set and the corresponding event occurs, the appropriate bit is set in the IPR, and, if this bit is not masked, the interrupt controller will interrupt the M68000 core.

3.1.6 DMA Bus Arbitration

The IDMA controller uses the M68000 bus arbitration protocol to request bus mastership before entering the DMA mode of operation. The six SDMA channels have priority over the IDMA and can transfer data between any two IDMA bus cycles with $\overline{\text{BGACK}}$ remaining continuously low. Once the processor has initialized and started a DMA channel, an operand transfer request is made pending by either an external device or by using an internal request.

When the IDMA channel has an operand transfer request pending and $\overline{\text{BCLR}}$ is not asserted, the IDMA will request bus mastership from the internal bus arbiter using the internal signal IDBR (see Figure 3-12). The arbiter will assert the internal M68000 core bus request ($\overline{\text{CBR}}$) signal and will monitor the core bus grant ($\overline{\text{CBG}}$) and external $\overline{\text{BR}}$ to determine when it may grant the IDMA mastership. The IDMA will monitor the address strobe ($\overline{\text{AS}}$), $\overline{\text{HALT}}$, bus error ($\overline{\text{BERR}}$), and bus grant acknowledge ($\overline{\text{BGACK}}$) signals. These signals must be negated to indicate that the previous bus cycle has completed and the previous bus master has released the bus. When these conditions are met, the IDMA only asserts $\overline{\text{BGACK}}$ to indicate that it has taken control of the bus. When all operand transfers have occurred, the IDMA will release control of the bus by negating $\overline{\text{BGACK}}$.

Internally generated IDMA requests are affected by a mechanism supported to reduce the M68000 core interrupt latency and external bus master arbitration latency (see 3.8.5 Bus Arbitration Logic). The IDMA is forced to relinquish the bus when an external bus master requests the bus ($\overline{\text{BR}}$ is asserted) or when the M68000 core has an unmasked pending interrupt request. In these cases, the on-chip arbiter sends an internal bus-clear signal to the IDMA. In response, any operand transfer in progress will be fully completed (up to four bus cycles depending on the configuration), and bus ownership will be released.

When the IDMA regains the bus, it will continue transferring where it left off. If the core caused the bus to be relinquished, no further IDMA bus cycles will be started until IPA in the SCR is cleared. If the cause was an external request, no further IDMA bus cycles will be started while $\overline{\text{BR}}$ remains asserted. When $\overline{\text{BR}}$ is externally negated, if a transfer request is pending and IPA is cleared, the IDMA will arbitrate for the bus and continue normal operation.

3.1.7 Bus Exceptions

In any computer system, the possibility always exists that an error will occur during a bus cycle due to a hardware failure, random noise, or an improper access. When an asynchronous bus structure, such as that supported by the M68000 is used, it is easy to make provisions allowing a bus master to detect and respond to errors during a bus cycle. The IDMA recognizes the same bus exceptions as the M68000 core: reset, bus error, halt, and retry.

NOTE

These exceptions also apply to the SDMA channels except that the bus error reporting method is different. See 4.5.8.4 Bus Error on SDMA Access for further details.

3.1.7.1 Reset

Upon an external chip reset, the IDMA channel immediately aborts the channel operation, returns to the idle state, and clears CSR and CMR (including the STR bit). If a bus cycle is in progress when reset is detected, the cycle is terminated, the control and address/data pins are three-stated, and bus ownership is released. The IDMA can also be reset by RST in the CMR.

3.1.7.2 Bus Error

When a fatal error occurs during a bus cycle, a bus error exception is used to abort the cycle and systematically terminate that channel's operation. The IDMA terminates the current bus cycle, signals an error in the CSR, and generates a maskable interrupt. The IDMA clears STR and waits for a restart of the channel and the negation of $\overline{\text{BERR}}$ before starting any new bus cycles.

NOTE

Any data that was previously read from the source into the DHR will be lost.

3.1.7.3 Halt

IDMA transfer operation may be suspended at any time by asserting $\overline{\text{HALT}}$ to the IDMA. In response, any bus cycle in progress is completed (after $\overline{\text{DTACK}}$ is asserted), and bus ownership is released. No further bus cycles will be started while $\overline{\text{HALT}}$ remains asserted. When the IDMA is in the middle of an operand transfer when halted and $\overline{\text{HALT}}$ is subsequently negated, and if a new transfer request is pending, then IDMA will arbitrate for the bus and continue normal operation.

3.1.7.4 Relinquish and Retry

When $\overline{\text{HALT}}$ and $\overline{\text{BERR}}$ are asserted during a bus cycle, the IDMA terminates the bus cycle, releases the bus, and suspends any further operation until these signals are negated. When $\overline{\text{HALT}}$ and $\overline{\text{BERR}}$ are negated, the IDMA will arbitrate for the bus, re-execute the interrupted bus cycle, and continue normal operation.

3.2 INTERRUPT CONTROLLER

The IMP interrupt controller accepts and prioritizes both internal and external interrupt requests and generates a vector number during the CPU interrupt acknowledge cycle. Interrupt nesting is also provided so that an interrupt service routine of a lower priority interrupt may be suspended by a higher priority interrupt request. The interrupt controller block diagram is shown in Figure 3-2.

The on-chip interrupt controller has the following features:

- Two Operational Modes: Normal and Dedicated
- Eighteen Prioritized Interrupt Sources (Internal and External)
- A Fully Nested Interrupt Environment
- Unique Vector Number for Each Internal/External Source Generated
- Three Interrupt Request and Interrupt Acknowledge Pairs

- $\overline{\text{DTACK}}$ Generation When Vectors Supplied Internally

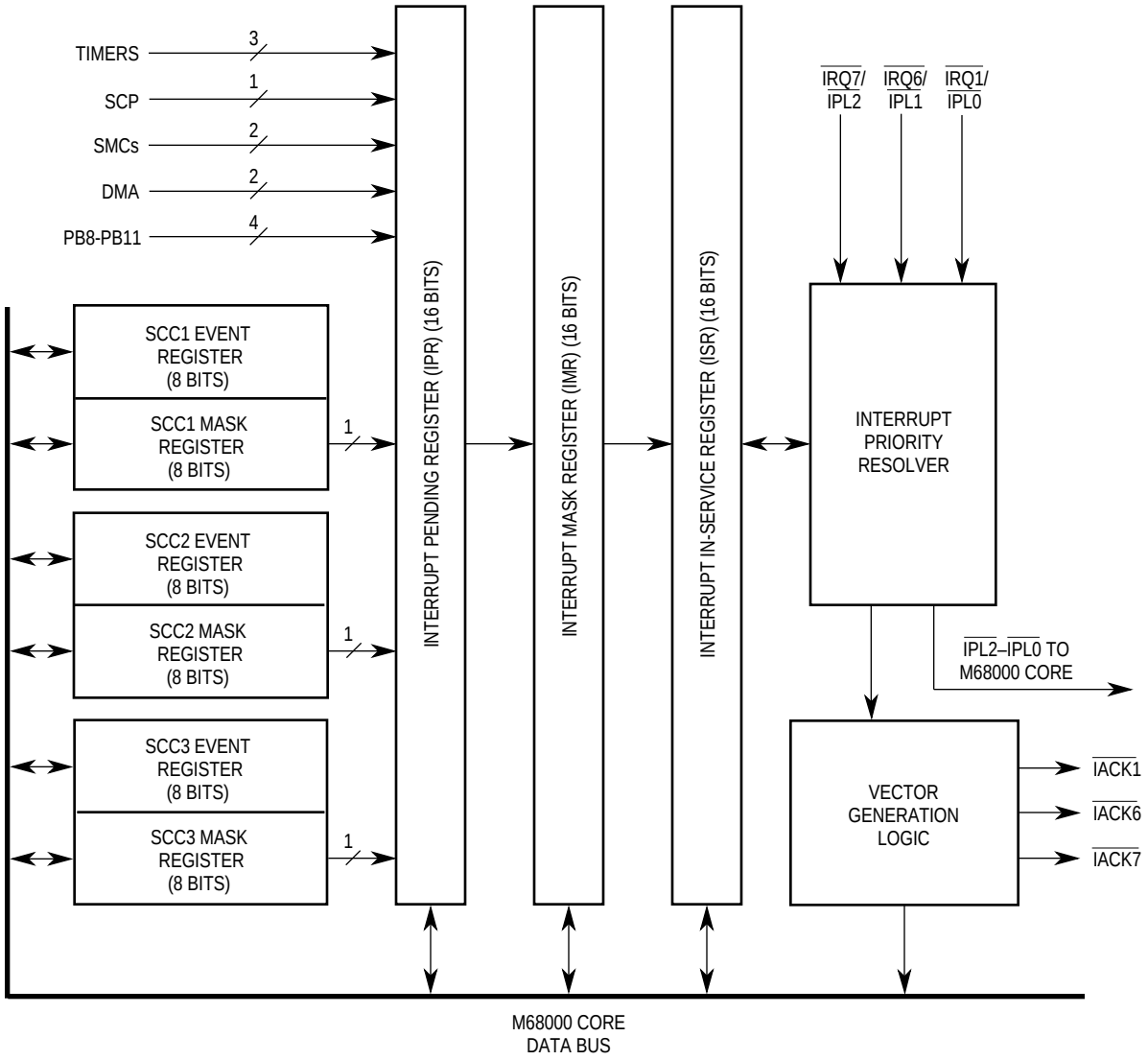


Figure 3-2. Interrupt Controller Block Diagram

3.2.1 Overview

An overview of IMP interrupt processing is presented in the following paragraphs.

3.2.1.1 IMP Interrupt Processing Overview

Interrupt processing on the IMP involves four steps. A typical sequence of these four steps is as follows:

1. The interrupt controller on the IMP collects interrupt events from on and off-chip peripherals, prioritizes them, and presents the highest priority request to the M68000 core.
2. The M68000 responds to the interrupt request by executing an interrupt acknowledge

bus cycle after the completion of the current instruction.

3. The interrupt controller recognizes the interrupt acknowledge cycle and places the interrupt vector for that interrupt request onto the M68000 bus.
4. The M68000 reads the vector, reads the address of the interrupt handler in the exception vector table, and then begins execution at that address.

Steps 2 and 4 are the responsibility of the M68000 core on the IMP; whereas, steps 1 and 3 are the responsibility of the interrupt controller on the IMP.

The M68000 core is not modified on the IMP; thus, steps 2 and 4 operate exactly as they would on the MC68000. In step 2, the M68000 status register (SR) is available to mask interrupts globally or to determine which priority levels can currently generate interrupts (see 2.5 Interrupt Processing for more details). Also in step 2, the interrupt acknowledge cycle is executed.

The interrupt acknowledge cycle carries out a standard M68000 bus read cycle, except that FC2–FC0 are encoded as 111, A3–A1 are encoded with the interrupt priority level (1–7, with 7 (i.e., 111) being the highest), and A19–A16 are driven high. $\overline{\text{UDS}}$ and $\overline{\text{LDS}}$ are both driven low.

In step 4, the M68000 reads the vector number, multiplies it by 4 to get the vector address, fetches a 4-byte program address from that vector address (seeTable 2-5), and then jumps to that 4-byte address. That 4-byte address is the location of the first instruction in the interrupt handler.

Steps 1 and 3 are the responsibility of the interrupt controller on the IMP. In steps 1 and 3, a number of configuration options are available. For instance, in step 1, there are two modes for handling external interrupts: normal and dedicated. In step 3, there are several different ways of generating vectors. These and other interrupt controller options are introduced in the following paragraphs.

3.2.1.2 Interrupt Controller Overview

The interrupt controller receives interrupts from internal sources such as the timers, the IDMA controller, the serial communication controllers, and the parallel I/O pins (port B pins 11–8). These interrupts are called internal requests (INRQ). The interrupt controller allows for masking each INRQ interrupt source. When multiple events within a peripheral can cause the INRQ interrupt, each event is also maskable in a register in that peripheral.

In addition to the INRQ interrupts, the interrupt controller can also receive external requests (EXRQ). EXRQ interrupts are input to the IMP according to normal or dedicated mode. In the normal mode, EXRQ interrupts are encoded on the $\overline{\text{IPL2}}$–$\overline{\text{IPL0}}$ lines. In the dedicated mode, EXRQ interrupts are presented directly as $\overline{\text{IRQ7}}$, $\overline{\text{IRQ6}}$, and $\overline{\text{IRQ1}}$.

Normal Mode

In this mode, the three external interrupt request pins are configured as $\overline{\text{IPL2}}$–$\overline{\text{IPL0}}$ as in the original MC68000. Up to seven levels of interrupt priority may be encoded. Level 4 is reserved for IMP INRQ interrupts and may not be generated by an external device.

Dedicated Mode

In this mode, the three interrupt request pins are configured as $\overline{IRQ7}$, $\overline{IRQ6}$, and $\overline{IRQ1}$ to provide dedicated request lines for three external sources at priority levels 1, 6, and 7. Each of these lines may be programmed to be edge-triggered or level-sensitive. In addition to level 4, which is reserved for INRQ interrupts, interrupt priority levels 2, 3, and 5 must not be assigned to external devices in this mode.

All INRQ and EXRQ sources are prioritized within the interrupt controller. The M68000 supports seven priority levels. Level 7, the highest priority level, is nonmaskable. EXRQ sources are given their own separate priority level. Priority level 4 is reserved exclusively for the INRQ sources with all the INRQ sources being further prioritized within this level. If more than one INRQ or EXRQ interrupt request is pending, the interrupt controller presents the highest priority interrupt to the M68000 core through an internal, hidden set of $\overline{IPL2}$–$\overline{IPL0}$ lines.

When the M68000 core executes the interrupt acknowledge cycle, a vector must be provided. If an INRQ source generated the interrupt, the interrupt controller always provides the vector. If an EXRQ source generated the interrupt, three options are available to generate the vector.

First, in most cases the interrupt controller can be configured to provide the vector to the M68000 core. This is usually the preferred solution.

Second, the external peripheral can generate the vector. To assist this process, the interrupt controller can provide up to three interrupt acknowledge outputs ($\overline{IACK7}$, $\overline{IACK6}$, and $\overline{IACK1}$).

Third, the external peripheral can assert the autovector ($\overline{AVEC}$) pin to cause the M68000 to use an autovector. The autovector method maps each interrupt level to a fixed vector location in the exception vector table, regardless of how many interrupt sources exist at that level.

To improve interrupt latency timing, a fast interrupt latency technique is supported in the IMP. On recognition of an interrupt, the IMP can assert the bus clear ($\overline{BCLR}$) signal externally, which can be used to force other bus masters off the bus. This involves the IPA and BCLM bits in the system control register (see 3.8 System Control).

3.2.2 Interrupt Priorities

INRQ and EXRQ interrupts are assigned to an interrupt priority level. INRQ interrupts are also assigned relative priorities within their given interrupt priority level. A fully nested interrupt environment is provided so that a higher priority interrupt is serviced before a lower priority interrupt.

3.2.2.1 INRQ and EXRQ Priority Levels

Seven levels of interrupt priority may be implemented in IMP system designs, with level 7 having the highest priority. INRQ interrupts are assigned to level 4 (fixed). EXRQ interrupts are assigned by the user to any of the remaining six priority levels in normal mode. In dedicated mode, EXRQ interrupts may be assigned to priority levels 7, 6, and 1.

Table 3-3 indicates the interrupt levels available in both normal and dedicated modes. This table also shows the $\overline{IPL2}$–$\overline{IPL0}$ encoding that should be provided by external logic for each EXRQ interrupt level in normal mode. For the dedicated mode, this table shows the IMP input pins ($\overline{IRQ7}$, $\overline{IRQ6}$, and $\overline{IRQ1}$) that should be asserted by an external device according to the desired interrupt priority level.

Table 3-3. EXRQ and INRQ Prioritization

Priority Level	Normal Mode $\overline{IPL2}$–$\overline{IPL0}$	Dedicated Mode $\overline{IRQ7}$, $\overline{IRQ6}$, $\overline{IRQ1}$	Interrupt Source
7 (Highest)	000	$\overline{IRQ7}$	EXRQ
6	001	$\overline{IRQ6}$	EXRQ
5	010	*	EXRQ
4	*	*	INRQ
3	100	*	EXRQ
2	101	*	EXRQ
1 (Lowest)	110	$\overline{IRQ1}$	EXRQ

* Priority level not available to an external device in this mode.

3.2.2.2 INRQ Interrupt Source Priorities

Although all INRQ interrupts are presented at level 4, the interrupt controller further organizes interrupt servicing of the 15 INRQ interrupts according to the priorities illustrated in Table 3-4. The interrupt from the port B pin 11 (PB11) has the highest priority, and the interrupt from the port B pin 8 (PB8) has the lowest priority. A single interrupt priority within level 4 is associated with each table entry. The IDMA entry is associated with the general-purpose DMA channel only, and not with the SDMA channels that service the SCCs. Those interrupts are reported through each individual SCC channel or, in the case of a bus error, through the SDMA channels bus error entry.

Table 3-4. INRQ Prioritization within Interrupt Level 4

Priority Level	Interrupt Source Description	Multiple Interrupt Events
Highest	General-Purpose Interrupt 3 (PB11)	No
	General-Purpose Interrupt 2 (PB10)	No
	SCC1	Yes
	SDMA Channels Bus Error	No
	IDMA Channel	Yes
	SCC2	Yes
	Timer 1	Yes
	SCC3	Yes
	General-Purpose Interrupt 1 (PB9)	No
	Timer 2	Yes
	SCP	No
	Timer 3	No
	SMC1	No
	SMC2	No
	General-Purpose Interrupt 0 (PB8)	No
Lowest	Error	—

3.2.2.3 Nested Interrupts

The following rules apply to nested interrupts:

1. The interrupt controller responds to all EXRQ and INRQ interrupts based upon their assigned priority level. The highest priority interrupt request is presented to the M68000 core for servicing. After the vector number corresponding to this interrupt is passed to the core during an interrupt acknowledge cycle, an INRQ interrupt request is cleared in IPR. (EXRQ requests must be cleared externally.) The remaining interrupt

requests, if any, are then assessed by priority so that another interrupt request may be presented to the core.

2. The 3-bit mask in the M68000 core status register (SR) ensures that a subsequent interrupt request at a higher interrupt priority level will suspend handling of a lower priority interrupt. The 3-bit mask indicates the current M68000 priority. Interrupts are inhibited for all priority levels less than or equal to the current M68000 priority. Priority level 7 cannot be inhibited by the mask; it is a nonmaskable interrupt level.
3. The interrupt controller allows a higher priority INRQ interrupt to be presented to the M68000 core before the servicing of a lower priority INRQ interrupt is completed. This is achieved using the interrupt in-service register (ISR). Each bit in the ISR corresponds to an INRQ interrupt source.
4. During an interrupt acknowledge cycle for an INRQ interrupt, the in–service bit is set by the interrupt controller for that interrupt source. When this bit is set, any subsequent INRQ interrupt requests at this priority level or lower are disabled until servicing of the current interrupt is completed and the in-service bit is cleared by the user. Pending interrupts for these sources are still set by the corresponding interrupt pending bit.
5. Thus, in the interrupt service routine for the INRQ interrupt, the user can lower the M68000 core mask to level 3 in the status register to allow higher priority level 4 (INRQ) interrupts to generate an interrupt request. This capability provides nesting of INRQ interrupt requests for sources within level 4. This capability is similar to the way the M68000 core interrupt mask provides nesting of interrupt requests for the seven interrupt priority levels.

3.2.3 Masking Interrupt Sources and Events

The user may mask EXRQ and INRQ interrupts to prevent an interrupt request to the M68000 core. EXRQ interrupt masking is handled external to the IMP—e.g., by programming a mask register within an external device. INRQ interrupt masking is accomplished by programming the IMR. Each bit in the IMR corresponds to one of 15 INRQ interrupt sources.

When a masked INRQ interrupt source has a pending interrupt request, the corresponding bit is set in the IPR, even though the interrupt is not generated to the core. By masking all interrupt sources using the IMR, the user may implement a polling interrupt servicing scheme for INRQ interrupts, as discussed in 3.2.5.2 Interrupt Pending Register (IPR).

When an INRQ interrupt source from an on-chip peripheral has multiple interrupt events, the user can individually mask these events by programming that peripheral's mask register (see Figure 3-3). Table 3-4 indicates the interrupt sources that have multiple interrupt events. In this case, when a masked event occurs, an interrupt request is not generated for the associated interrupt source, and the corresponding bit in the IPR is not set. If the corresponding bit in the IPR is already set, then masking the event in the peripheral mask register causes the IPR bit to be cleared. To determine the cause of a pending interrupt when an interrupt source has multiple interrupt events, the user interrupt service routine must read the event register within that on-chip peripheral. By clearing all unmasked bits in the event register, the IPR bit is also cleared.

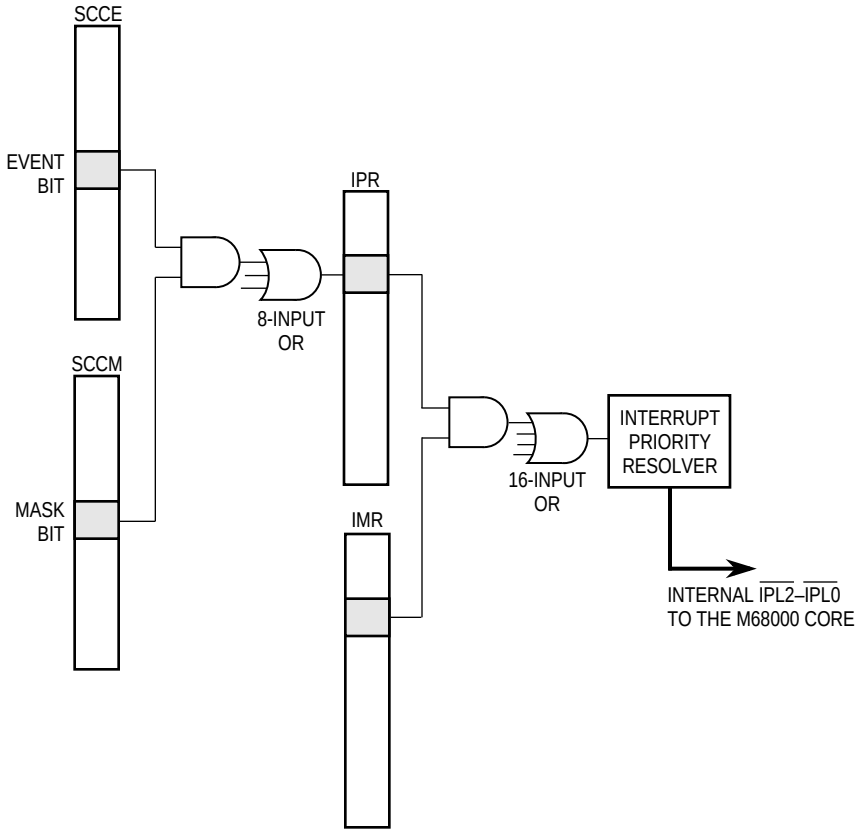


Figure 3-3. Interrupt Request Logic Diagram for SCCs

3.2.4 Interrupt Vector

Pending EXRQ interrupts and unmasked INRQ interrupts are presented to the M68000 core in order of priority. The M68000 core responds to an interrupt request by initiating an interrupt acknowledge cycle to receive a vector number, which allows the core to locate the interrupt's service routine.

If an INRQ source generated the interrupt, the interrupt controller always provides the vector corresponding to the highest priority, unmasked, pending interrupt. If an EXRQ source generated the interrupt, three options are available to generate the vector.

Option 1. By programming the GIMR, the user can enable the interrupt controller to provide the vector for any combination of EXRQ interrupt levels 1, 6, and 7. This is available regardless of whether normal or dedicated mode is selected. Whenever a vector is provided by the interrupt controller, $\overline{\text{DTACK}}$ is also provided by the interrupt controller during that interrupt acknowledge cycle. $\overline{\text{DTACK}}$ is an output from the IMP in this case.

The IMP can generate vectors for up to seven external peripherals by connecting the external request lines to $\overline{\text{IRQ7}}$, $\overline{\text{IRQ6}}$, $\overline{\text{IRQ1}}$, PB11, PB10, PB9, and PB8. PB11, PB10, PB9, and PB8 are prioritized within level 4.

Option 2. The external peripheral can generate the vector. In this case the external device must decode the interrupt acknowledge cycle, put out the 8-bit vector, and generate $\overline{\text{DTACK}}$. The decoding of the interrupt acknowledge cycle can be provided by the $\overline{\text{IACK7}}$, $\overline{\text{IACK6}}$, and $\overline{\text{IACK1}}$ signals (enabled in the PBCNT register) if either normal or dedicated mode is chosen. These signals eliminate the need for external logic to perform the decoding of the A19–A16, A3–A1, and FC2–FC0 pins externally to detect the interrupt acknowledge cycle. If the $\overline{\text{IACK}}$ signals are not needed, they can be regained as general purpose parallel I/O pins. The external device must generate $\overline{\text{DTACK}}$ in this mode, and $\overline{\text{DTACK}}$ is an input to the IMP.

Option 3. The external peripheral can assert the $\overline{\text{AVEC}}$ pin to cause the M68000 to use an autovector. In this case, $\overline{\text{DTACK}}$ should not be asserted by the external device. $\overline{\text{AVEC}}$ is recognized by the M68000 core on the falling edge of S4 and should meet the asynchronous setup time to the falling edge of S4. The $\overline{\text{IACK}}$ signals can be used to help generate the $\overline{\text{AVEC}}$ signal for priority levels 1, 6, and 7, if needed.

NOTE

If $\overline{\text{AVEC}}$ is asserted during an interrupt acknowledge cycle, an autovector is taken, regardless of the vector on the bus. $\overline{\text{AVEC}}$ should not be asserted during level 4 interrupt acknowledge cycles.

When the IMP generates the vector, the following procedure is used. The three most significant bits of the interrupt vector number are programmed by the user in the GIMR. These three bits are concatenated with five bits generated by the interrupt controller to provide an 8-bit vector number to the core. The interrupt controller's encoding of the five low-order bits of the interrupt vector is shown in Table 3-5. An example vector calculation is shown in Figure 3-4. When the core initiates an interrupt acknowledge cycle for level 4 and there is no internal interrupt pending, the interrupt controller encodes the error code 00000 onto the five low-order bits of the interrupt vector.

Table 3-5. Encoding the Interrupt Vector

Priority Level	5-Bit Vector	Interrupt Source
7 (Highest)	10111	External Device
6	10110	External Device
5	None	External Device
4	01111	General-Purpose Interrupt 3 (PB11)
4	01110	General-Purpose Interrupt 2 (PB10)
4	01101	SCC1
4	01100	SDMA Channels Bus Error
4	01011	IDMA Channel
4	01010	SCC2
4	01001	Timer 1
4	01000	SCC3
4	00111	General-Purpose Interrupt 1 (PB9)
4	00110	Timer 2
4	00101	SCP
4	00100	Timer 3
4	00011	SMC1
4	00010	SMC2
4	00001	General-Purpose Interrupt 0 (PB8)
4	00000	Error
3	None	External Device
2	None	External Device
1 (Lowest)	10001	External Device

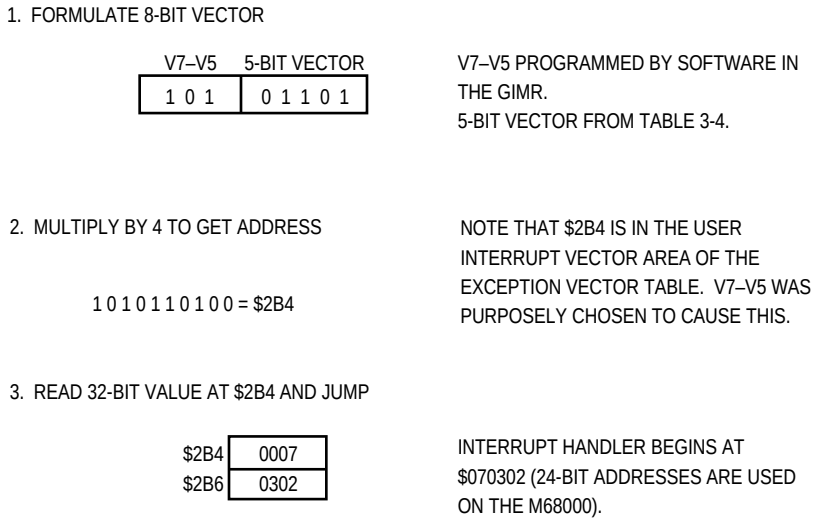


Figure 3-4. SCC1 Vector Calculation Example

3.2.5 Interrupt Controller Programming Model

The user communicates with the interrupt controller using four registers. The global interrupt mode register (GIMR) defines the interrupt controller's operational mode. The interrupt pending register (IPR) indicates which INRQ interrupt sources require interrupt service. The interrupt mask register (IMR) allows the user to prevent any of the INRQ interrupt sources from generating an interrupt request. The interrupt in-service register (ISR) provides a capability for nesting INRQ interrupt requests.

3.2.5.1 Global Interrupt Mode Register (GIMR)

The user normally writes the GIMR soon after a total system reset. The GIMR is initially $0000 and is reset only upon a total system reset. If bits V7–V5 of the GIMR are not written to specify an interrupt vector prior to the first interrupt condition, the interrupt controller will pass the vector $0F (the uninitialized interrupt vector), regardless of the interrupt source.

15	14	13	12	11	10	9	8	7 5	4 0
MOD	IV7	IV6	IV1	—	ET7	ET6	ET1	V7–V5	RESERVED

MOD—Mode

0 = Normal operational mode. Interrupt request lines are configured as $\overline{IPL2}$–$\overline{IPL0}$.
1 = Dedicated operational mode. Interrupt request lines are configured as $\overline{IRQ7}$, $\overline{IRQ6}$, and $\overline{IRQ1}$.

IV7—Level 7 Interrupt Vector

This bit is valid in both normal and dedicated modes.

0 = Internal vector. The interrupt controller will provide the vector number for a level 7 interrupt during the interrupt acknowledge cycle.
1 = External vector. The interrupt controller will not provide the vector number for a level 7 interrupt.

IV6—Level 6 Interrupt Vector

This bit is valid in both normal and dedicated modes.

0 = Internal vector. The interrupt controller will provide the vector number for a level 6 interrupt during the interrupt acknowledge cycle.
1 = External vector. The interrupt controller will not provide the vector number for a level 6 interrupt.

IV1—Level 1 Interrupt Vector

This bit is valid in both normal and dedicated modes.

0 = Internal vector. The interrupt controller will provide the vector number for a level 1 interrupt acknowledge cycle.
1 = External vector. The interrupt controller will not provide the vector number for a level 1 interrupt.

ET7—$\overline{\text{IRQ7}}$ Edge-/Level-Triggered

This bit is valid only in the dedicated mode.

0 = Level-triggered. An interrupt is made pending when $\overline{\text{IRQ7}}$ is low.

NOTE

The M68000 always treats level 7 as an edge-sensitive interrupt. Normally, users should not select the level-triggered option. The level-triggered option is useful when it is desired to make the negation of $\overline{\text{IRQ7}}$ cause the IOUT2–IOUT0 pins to cease driving a level 7 interrupt request when the MC68302 is used in the disable CPU mode. This situation is as follows:

For a slave-mode MC68302, when it is triggered by $\overline{\text{IRQ1}}$, $\overline{\text{IRQ6}}$, or $\overline{\text{IRQ7}}$ to generate an interrupt, its interrupt controller will output the interrupt request on pins IOUT2–IOUT0 to another processor (MC68302, MC68020, etc.) For cases when the slave MC68302 does not generate a level 4 vector (i.e., the VGE bit is cleared), one must set the ET1, ET6, and ET7 bits to level-triggered and then negate the $\overline{\text{IRQ1}}$, $\overline{\text{IRQ6}}$, and $\overline{\text{IRQ7}}$ lines externally in the interrupt handler code. If the ET1, ET6, and ET7 bits are set to edge-triggered and the VGE bit is clear, the IOUT2–IOUT0 pins will never be cleared.

1 = Edge-triggered. An interrupt is made pending when $\overline{\text{IRQ7}}$ changes from one to zero (falling edge).

ET6—IRQ6 Edge-/Level-Triggered

This bit is valid only in the dedicated mode.

0 = Level-triggered. An interrupt is made pending when $\overline{\text{IRQ6}}$ is low.

NOTE

While in disable CPU mode during the host processor interrupt acknowledge cycle for $\overline{\text{IRQ6}}$, if $\overline{\text{IRQ6}}$ is not continuously asserted, the interrupt controller will still provide the vector number (and $\overline{\text{DTACK}}$) according to the IV6 bit. The $\overline{\text{IACK6}}$ falling edge may be used externally to negate $\overline{\text{IRQ6}}$.

1 = Edge-triggered. An interrupt is made pending when $\overline{\text{IRQ6}}$ changes from one to zero (falling edge).

ET1—$\overline{\text{IRQ1}}$ Edge-/Level-Triggered

This bit is valid only in the dedicated mode.

0 = Level-triggered. An interrupt is made pending when $\overline{\text{IRQ1}}$ is low.

NOTE

While in disable CPU mode, during the host processor interrupt acknowledge cycle for $\overline{IRQ1}$, if $\overline{IRQ1}$ is not continuously asserted, the interrupt controller will still provide the vector number (and $\overline{DTACK}$) according to the IV1 bit. The $\overline{IACK6}$ falling edge can be used externally to negate $\overline{IRQ1}$.

1 = Edge-triggered. An interrupt is made pending when $\overline{IRQ1}$ changes from one to zero (falling edge).

V7–V5—Interrupt Vector Bits 7–5

These three bits are concatenated with five bits provided by the interrupt controller, which indicate the specific interrupt source, to form an 8-bit interrupt vector number. If these bits are not written, the vector $0F is provided.

Note:

These three bits should be greater than or equal to ‘010’ in order to put the interrupt vector in the area of the exception vector table for user vectors.

Bits 11 and 4–0—Reserved for future use.

3.2.5.2 Interrupt Pending Register (IPR)

Each bit in the 16-bit IPR corresponds to an INRQ interrupt source. When an INRQ interrupt is received, the interrupt controller sets the corresponding bit in the IPR.

In a vectored interrupt environment, the interrupt controller clears the IPR bit when the vector number corresponding to the INRQ interrupt source is passed to the M68000 core during an interrupt acknowledge cycle, unless an event register exists for that INRQ interrupt. In a polled interrupt scheme, the user must periodically read the IPR. When a pending interrupt is handled, the user should clear the corresponding bit in the IPR by writing a one to that bit. (If an event register exists, the unmasked event register bits should be cleared instead, causing the IPR bit to be cleared.) Since the user can only clear bits in this register, the bits that are written as zeros will not be affected. The IPR is cleared at reset.

NOTE

The ERR bit is set if the user drives the $\overline{IPL2}$–$\overline{IPL0}$ lines to interrupt level 4 and no INRQ interrupt is pending.

15	14	13	12	11	10	9	8
PB11	PB10	SCC1	SDMA	IDMA	SCC2	TIMER1	SCC3

7	6	5	4	3	2	1	0
PB9	TIMER2	SCP	TIMER3	SMC1	SMC2	PB8	ERR

3.2.5.3 Interrupt Mask Register (IMR)

Each bit in the 16-bit IMR corresponds to an INRQ interrupt source. The user masks an interrupt source by clearing the corresponding bit in the IMR. When a masked INRQ interrupt occurs, the corresponding bit in the IPR is set, but the IMR bit prevents the interrupt request from reaching the M68000 core. If an INRQ source is requesting interrupt service when the user clears the IMR bit, the request to the core will cease, but the IPR bit remains set. If the IMR bit is then set later by the user, the pending interrupt request will once again request interrupt service and will be processed by the core according to its assigned priority. The IMR, which can be read by the user at any time, is cleared by reset.

It is not possible to mask the ERR INRQ source in the IMR. Bit 0 of the IMR is undefined.

NOTE

If a bit in the IMR is masked at the same time that the interrupt at level 4 is causing the M68000 core to begin the interrupt acknowledge cycle, then the interrupt is not processed, and one of two possible cases will occur. First, if other unmasked interrupts are pending at level 4, then the interrupt controller will acknowledge the interrupt with a vector from the next highest priority unmasked interrupt source. Second, if no other interrupts are pending at level 4, then the interrupt controller will acknowledge the interrupt with the error vector (00000 binary).

To avoid handling the error vector, the user can raise the interrupt mask in the M68000 core status register (SR) to 4 before masking the interrupt source and then lower the level back to its original value. Also, if the interrupt source has multiple events (e.g., SCC1), then the interrupts for that peripheral can be masked within the peripheral mask register.

NOTE

To clear bits that were set by multiple interrupt events, the user should clear all the unmasked events in the corresponding on-chip peripheral's event register.

15	14	13	12	11	10	9	8
PB11	PB10	SCC1	SDMA	IDMA	SCC2	TIMER1	SCC3

7	6	5	4	3	2	1	0
PB9	TIMER2	SCP	TIMER3	SMC1	SMC2	PB8	—

3.2.5.4 Interrupt In-Service Register (ISR).

Each bit in the 16-bit ISR corresponds to an INRQ interrupt source. In a vectored interrupt environment, the interrupt controller sets the ISR bit when the vector number corresponding to the INRQ interrupt source is passed to the core during an interrupt acknowledge cycle. The user's interrupt service routine should clear this bit during the servicing of the interrupt. (If an event register exists for this peripheral, its bits should also be cleared by the user program.) To clear a bit in the ISR, the user writes a one to that bit. The user can only clear bits in this register, and bits that are written with zeros will not be affected. The ISR is cleared at reset.

This register may be read by the user to determine which INRQ interrupts are currently being processed. More than one bit in the ISR may be a one if the capability is used to allow higher priority level 4 interrupts to interrupt lower priority level 4 interrupts. See 3.2.2.3 Nested Interrupts for more details.

The user can control the extent to which level 4 interrupts may interrupt other level 4 interrupts by selectively clearing the ISR. A new INRQ interrupt will be processed if it has a higher priority than the highest priority INRQ interrupt having its ISR bit set. Thus, if an INRQ interrupt routine lowers the 3-bit mask in the M68000 core to level 3 and also clears its ISR bit at the beginning of the interrupt routine, then a lower priority INRQ interrupt can interrupt it as long as the lower priority is higher than any other ISR bits that are set.

If the INRQ error vector is taken, no bit in the ISR is set. Bit 0 of the ISR is always zero.

15	14	13	12	11	10	9	8
PB11	PB10	SCC1	SDMA	IDMA	SCC2	TIMER1	SCC3

7	6	5	4	3	2	1	0
PB9	TIMER2	SCP	TIMER3	SMC1	SMC2	PB8	0

3.2.6 Interrupt Handler Examples

The following examples illustrate proper interrupt handling on the IMP. Nesting of level 4 interrupts (a technique described earlier) is not implemented in the following examples.

Example 1—Timer 3 (Software Watchdog Timer) Interrupt Handler

1. Vector to interrupt handler.
2. (Handle Event)
3. Clear the TIMER3 bit in the ISR.
4. Execute RTE instruction.

Example 2— SCC1 Interrupt Handler

1. Vector to interrupt handler.

2. Immediately read the SCC1 event (SCCE1) register into a temporary location.
3. Decide which events in the SCCE1 will be handled in this handler and clear those bits in the SCCE1 as soon as possible.

(Handle events in the SCC1 Rx or Tx BD tables.)

At the end:

4. Clear the SCC1 bit in the ISR.
5. Execute RTE instruction. If any unmasked bits in SCCE1 remain at this time (either uncleared by the software or set by the IMP during the execution of this handler), this interrupt source will be made pending again immediately following the RTE instruction.

In example 1, the hardware clears the TIMER3 bit in the IPR during the interrupt acknowledge cycle. This is an example of a handler for an interrupt source without multiple events. In example 2, the IPR bit remains set as long as one or more unmasked event bits remain the in the SCCE1 register. This is an example of a handler for an interrupt source with multiple events.

Note that, in both cases, it is not necessary to clear the IPR bit; however, in both cases, it is necessary to clear the ISR bit to allow future interrupts from this source.

3.3 PARALLEL I/O PORTS

The IMP supports two general-purpose I/O ports, port A and port B, whose pins can be general-purpose I/O pins or dedicated peripheral interface pins. Some port B pins are always maintained as four general-purpose I/O pins, each with interrupt capability.

3.3.1 Port A

Each of the 16 port A pins are independently configured as a general-purpose I/O pin if the corresponding port A control register (PACNT) bit is cleared. Port A pins are configured as dedicated on-chip peripheral pins if the corresponding PACNT bit is set. An example block diagram of PA0 is given in Figure 3-5

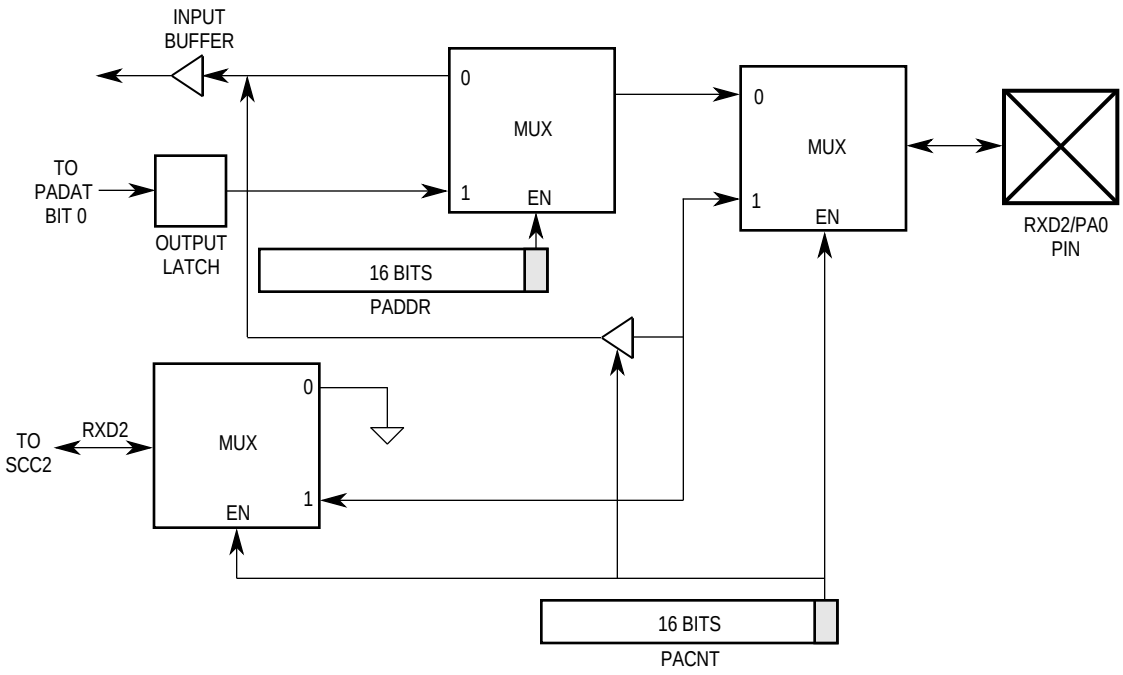


Figure 3-5. Parallel I/O Block Diagram for PA0

When acting as a general-purpose I/O pin, the signal direction for that pin is determined by the corresponding control bit in the port A data direction register (PADDR). The port I/O pin is configured as an input if the corresponding PADDR bit is cleared; it is configured as an output if the corresponding PADDR bit is set. All PACNT bits and PADDR bits are cleared on total system reset, configuring all port A pins as general-purpose input pins. (Note that these port pins do not have internal pullup resistors).

If a port A pin is selected as a general-purpose I/O pin, it may be accessed through the port A data register (PADAT). Data written to the PADAT is stored in an output latch. If a port A pin is configured as an output, the output latch data is gated onto the port pin. In this case, when the PADAT is read, the contents of the output latch associated with the output port pin are read. If a port A pin is configured as an input, data written to PADAT is still stored in the output latch but is prevented from reaching the port pin. In this case, when PADAT is read, the state of the port pin is read.

If a port A pin is selected as a dedicated on-chip peripheral pin, the corresponding bit in the PADDR is ignored, and the direction of the pin is determined by the operating mode of the on-chip peripheral. In this case, the PADAT contains the current state of the peripheral's input pin or output driver.

Certain pins may be selected as general-purpose I/O pins, even when other pins related to the same on-chip peripheral are used as dedicated pins. For example, a system that configures SCC2 to operate in a nonmultiplexed mode without the modem control lines and external clocks (RCLK2, TCLK2, $\overline{\text{CD2}}$, $\overline{\text{CTS2}}$, and $\overline{\text{RTS2}}$) may dedicate the data lines (RXD2 and TXD2) to SCC2 and configure the others as general-purpose I/O pins. What the peripheral

now receives as its input, given that some of its pins have been reassigned, is shown in Table 3-6. If an input pin to a channel (for example $\overline{CD2}$ or $\overline{CTS2}$) is used as a general-purpose I/O pin, then the input to the peripheral is automatically connected internally to V_{DD} or GND, based on the pin's function. This does not affect the operation of the port pins in their general-purpose I/O function.

NOTE

If the $\overline{DREQ}$/PA13 pin is selected to be PA13, then $\overline{DREQ}$ is tied low. If the IDMA is programmed for external requests, then it always recognizes an external request, and the entire block will be transferred in one burst.

Table 3-6. Port A Pin Functions

PACNT Bit = 1 Pin Function	PACNT Bit = 0 Pin Function	Input to SCC2/SCC3/IDMA
RXD2	PAO	GND
TXD2	PA1	—
RCLK2	PA2	GND
TCLK2	PA3	RCLK2 #
$\overline{CTS2}$	PA4	GND
$\overline{RTS2}$	PA5	—
$\overline{CD2}$	PA6	GND
SDS2/BRG2	PA7	—
RXD3	PA8	GND
TXD3	PA9	—
RCLK3	PA10	GND
TCLK3	PA11	RCLK3 #
BRG3	PA12	—
$\overline{DREQ}$	PA13	GND
$\overline{DACK}$	PA14	—
$\overline{DONE}$	PA15	V_{DD}

Allows a single external clock source on the RCLK pin to clock both the SCC receiver and transmitter.

3.3.2 Port B

Port B has 12 pins. PB7–PB0 may be configured as general-purpose I/O pins or as dedicated peripheral interface pins; whereas, PB11–PB8 are always maintained as four general-purpose pins, each with interrupt capability.

3.3.2.1 PB7–PB0

Each port B pin may be configured as a general-purpose I/O pin or as a dedicated peripheral interface pin. PB7–PB0 functions exactly like PA15–PA0, except that PB7–PB0 is controlled by the port B control register (PBCNT), the port B data direction register (PBDDR), and the port B data register (PBDAT), and PB7 is configured as an open-drain output ($\overline{WDOG}$) upon total system reset.

Table 3-7 shows the dedicated function of each pin. The third column shows the input to the peripheral when the pin is used as a general-purpose I/O pin.

Table 3-7. Port B Pin Functions

PBCNT Bit = 1 Pin Function	PBCNT Bit = 0 Pin Function	Input to Interrupt Control and Timers
IACK7	PB0	—
IACK6	PB1	—
IACK1	PB2	—
TIN1	PB3	GND
TOUT1	PB4	—
TIN2	PB5	GND
TOUT2	PB6	—
WDOG	PB7	—

3.3.2.2 PB11–PB8

PB11–PB8 are four general-purpose I/O pins continuously available as general-purpose I/O pins and, therefore, are not referenced in the PBCNT. PB8 operates like PB11–PB9 except that it can also be used as the DRAM refresh controller request pin, as selected in the system control register (SCR).

The direction of each pin is determined by the corresponding bit in the PBDDR. The port pin is configured as an input if the corresponding PBDDR bit is cleared; it is configured as an output if the corresponding PBDDR bit is set. PBDDR11–PBDDR8 are cleared on total system reset, configuring all PB11–PB8 pins as general-purpose input pins. (Note that the port pins do not have internal pullup resistors). The GIMR is also cleared on total system reset so that if any PB11–PB8 pin is left floating it will not cause a spurious interrupt.

The PB11–PB8 pins are accessed through the PBDAT. Data written to PBDAT11–PBDAT8 is stored in an output latch. If the port pin is configured as an output, the output latch data is gated onto the port pin. In this case, when PBDAT11–PBDAT8 is read, the contents of the output latch associated with the output port pin are read. If a port B pin is configured as an input, data written to PBDAT is still stored in the output latch but is prevented from reaching the port pin. In this case, when PBDAT is read, the state of the port pin is read.

When a PB11–PB8 pin is configured as an input, a high-to-low change will cause an interrupt request signal to be sent to the IMP interrupt controller. Each of the four interrupt requests is associated with a fixed internal interrupt priority level within level 4. (The priority at which each bit requests an interrupt is detailed in Table 3-4.) Each request can be masked independently in the IMP interrupt controller by clearing the appropriate bit in the IMR (PB11–PB8). The input signals to PB11–PB8 must meet specifications 190 and 191 shown in Table 6.16 of the AC Electrical Specifications.

3.3.3 I/O Port Registers

The I/O port consists of three memory-mapped read-write 16-bit registers for port A and three memory-mapped read-write 16-bit registers for port B. Refer to Figure 3-6 for the I/O port registers. The reserved bits are read as zeros.

Port A Control Register(PACNT

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

0 = I/O 1 = Peripheral

Port A Data Direction Register(PADDR)

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

0 = Input 1 = Output

Port A Data Register(PADAT)

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

Port B Control Register(PBCNT)

15	8	7	6	5	4	3	2	1	0
RESERVED		CB	CB	CB	CB	CB	CB	CB	CB

0 = I/O 1 = Peripheral

Port B Data Direction Register(PBDDR)

15	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED		DB	DB	DB	DB	DB	DB	DB	DB	DB	DB	DB	DB

0 = Input 1 = Output

Port B Data Register(PBDAT)

15	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED		PB	PB	PB	PB	PB	PB	PB	PB	PB	PB	PB	PB

Figure 3-6. Parallel I/O Port Registers

3.4 DUAL-PORT RAM

The CP has 1152 bytes of static RAM configured as a dual-port memory. The dual-port RAM can be accessed by the CP main controller or by one of three bus masters: the M68000 core, the IDMA, or an external master. The M68000 core and the IDMA access the RAM synchro-

nously with no wait states. The external master requests the M68000 bus using the $\overline{BR}$ pin and is granted bus ownership. The external master must then access the RAM synchronously with respect to the IMP system clock with zero or one wait state, or asynchronously as determined by the EMWS and SAM bits in the system control register. Except for several locations initialized by the CP, the dual-port RAM is undefined at power-on reset but is not modified by successive resets. The RAM is divided into two parts: parameter RAM and system RAM.

The 576-byte parameter RAM area includes pointers, counters, and registers used with the serial ports. This area is accessed by the CP during communications processing. Any individual locations not required in a given application may be used as general-purpose RAM.

The 576-byte system RAM is a general-purpose RAM, which may be used as M68000 data and/or program RAM or CP microcode RAM. As data RAM, it can include serial port data buffers or can be used for other purposes such as a no-wait-state cache for the M68000 core. As CP microcode RAM, it is used exclusively to store microcode for the CP main controller, allowing the development of special protocols or protocol enhancements, under special arrangement with Motorola. Appendix C discusses available offerings.

The RAM block diagram is shown in Figure 3-7. The M68000 core, the IDMA, and the external master access the RAM through the IMP bus interface unit (BIU) using the M68000 bus. When an access is made, the BIU generates a wait signal to the CP main controller to prevent simultaneous access of the RAM. The CP main controller waits for one cycle to allow the RAM to service the M68000 bus cycle and then regenerates its RAM cycle. This mechanism allows the RAM to be accessed synchronously by the M68000 core, IDMA, or external master without wait states. Thus, during the four-clock M68000 memory cycle, three internal accesses by the CP main controller may occur. The BIU also provides the $\overline{DTACK}$ signal output when the RAM and on-chip registers are accessed by any M68000 bus master.

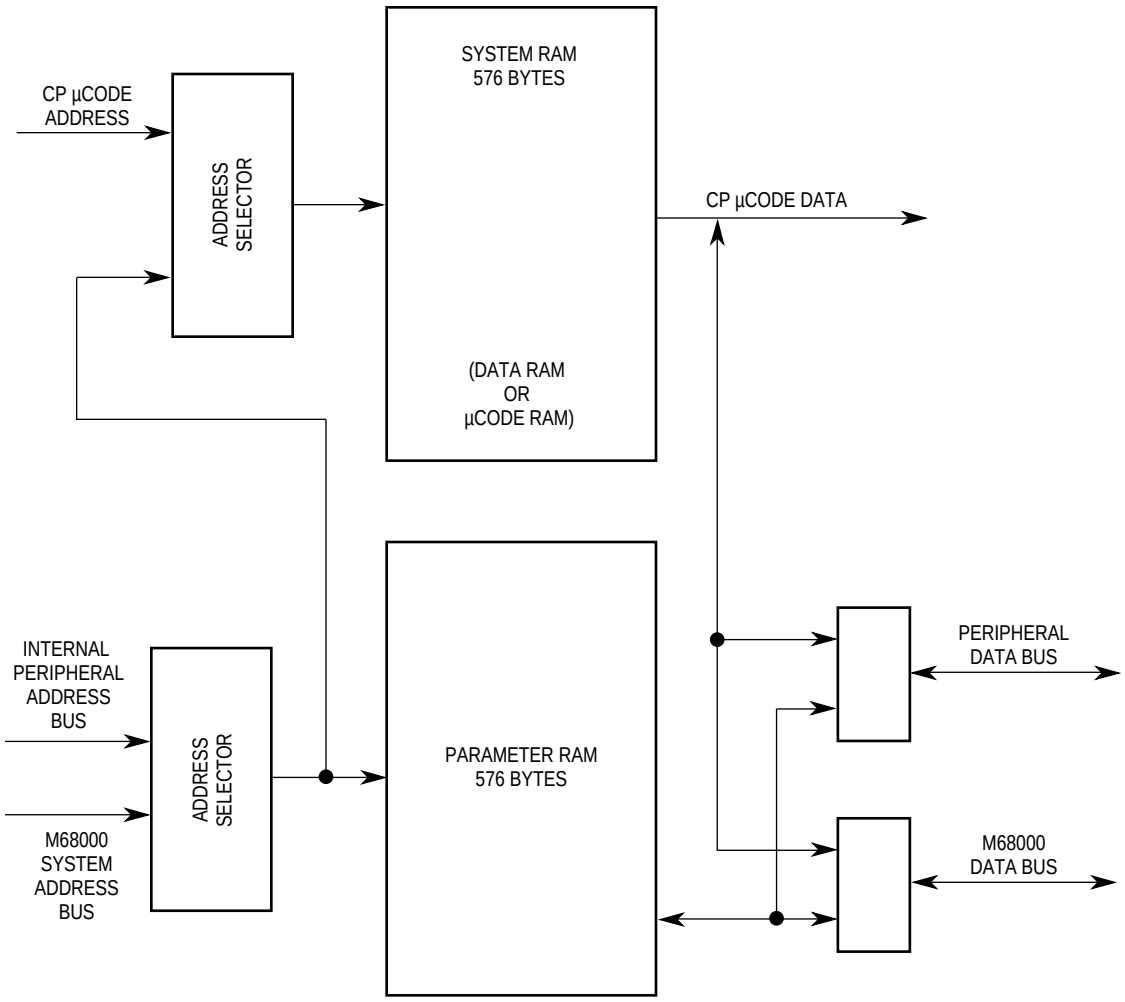


Figure 3-7. RAM Block Diagram

3.5 TIMERS

The MC68302 includes three timer units: two identical general-purpose timers and a software watchdog timer.

Each general-purpose timer consists of a timer mode register (TMR), a timer capture register (TCR), a timer counter (TCN), a timer reference register (TRR), and a timer event register (TER). The TMR contains the prescaler value programmed by the user. The software watchdog timer, which has a watchdog reference register (WRR) and a watchdog counter (WCN), uses a fixed prescaler value. The timer block diagram is shown in Figure 3-8.

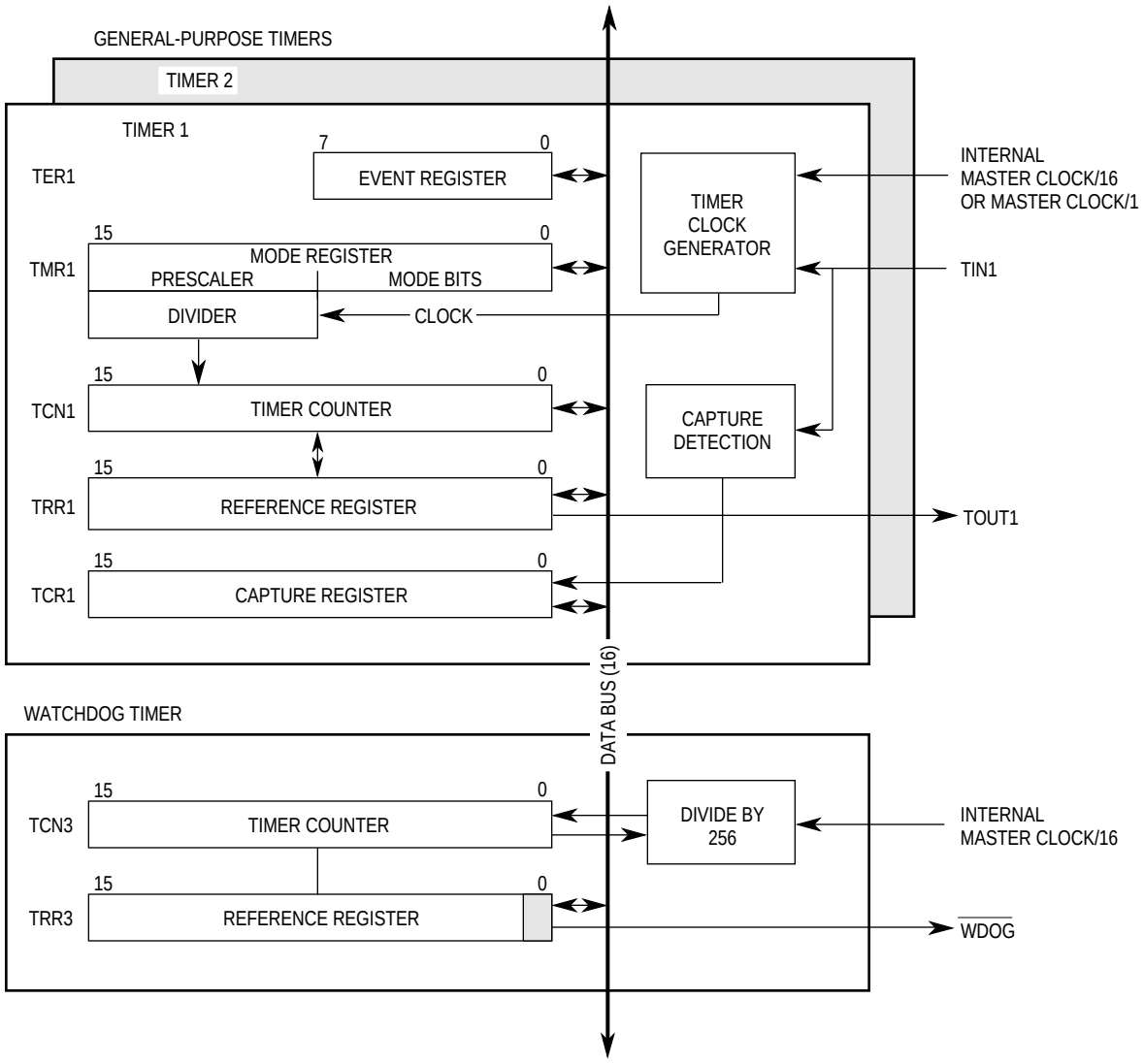


Figure 3-8. Timer Block Diagram

3.5.1 Timer Key Features

The two identical general-purpose timer units have the following features:

- Maximum Period of 16 Seconds (at 16.67 MHz)
- 60-ns Resolution (at 16.67 MHz)
- Programmable Sources for the Clock Input
- Input Capture Capability
- Output Compare with Programmable Mode for the Output Pin
- Two Timers Cascadable to Form a 32-Bit Timer
- Free Run and Restart Modes

The watchdog timer has the following features:

- A 16-Bit Counter and Reference Register
- Maximum Period of 16.78 Seconds (at 16 MHz)
- 0.5 ms Resolution (at 16 MHz)
- Output Signal (WDOG)
- Interrupt Capability

3.5.2 General Purpose Timer Units

The clock input to the prescaler may be selected from the main clock (divided by 1 or by 16) or from the corresponding timer input (TIN) pin. TIN is internally synchronized to the internal clock. The clock input source is selected by the ICLK bits of the corresponding TMR. The prescaler is programmed to divide the clock input by values from 1 to 256. The output of the prescaler is used as an input to the 16-bit counter.

The resolution of the timer is one clock cycle (60 ns at 16.67 MHz). The maximum period (when the reference value is all ones) is 268,435,456 cycles (16.78 seconds at 16.00 MHz).

Each timer may be configured to count until a reference is reached and then either resets to zero on the next clock or continues to run. The free run/restart (FRR) bit of the corresponding TMR selects each mode. Upon reaching the reference value, the corresponding TER bit is set, and an interrupt is issued if the output reference interrupt enable (ORI) bit in TMR is set.

Each timer may output a signal on the timer output ($\overline{\text{TOUT1}}$ or $\overline{\text{TOUT2}}$) pin when the reference value is reached, as selected by the output mode (OM) bit of the corresponding TMR. This signal can be an active-low pulse for one clock cycle or a toggle of the current output. The output can also be used as an input to the other timer, resulting in a 32-bit timer.

Each timer has a 16-bit TCR, which is used to latch the value of the counter when a defined transition (of TIN1 or TIN2) is sensed by the corresponding input capture edge detector. The type of transition triggering the capture is selected by the capture edge and enable interrupt (CE) bits in the corresponding TMR. Upon a capture or reference event, the corresponding TER bit is set, and a maskable interrupt is issued.

The timer registers may be modified at any time by the user.

3.5.2.1 Timer Mode Register (TMR1, TMR2)

TMR1 and TMR2 are identical 16-bit registers. TMR1 and TMR2, which are memory-mapped read-write registers to the user, are cleared by reset.

15 – 8	7 – 6	5	4	3	2 – 1	0
PRESCALER VALUE (PS)	CE	OM	ORI	FRR	ICLK	RST

RST—Reset Timer

This bit performs a software reset of the timer identical to that of an external reset.

0 = Reset timer (software reset), includes clearing the TMR, TRR, and TCN.
1 = Enable timer

ICLK—Input Clock Source for the Timer

00 = Stop count
01 = Master clock
10 = Master clock divided by 16. Note that this clock source is not synchronized to the timer; thus, successive timeouts may vary slightly in length.
11 = Corresponding TIN pin, TIN1 or TIN2 (falling edge)

FRR—Free Run/Restart

0 = Free run—timer count continues to increment after the reference value is reached.
1 = Restart—timer count is reset immediately after the reference value is reached.

ORI—Output Reference Interrupt Enable

0 = Disable interrupt for reference reached (does not affect interrupt on capture function)
1 = Enable interrupt upon reaching the reference value

OM—Output Mode

0 = Active-low pulse for one CLKO clock cycle (60 ns at 16.67 MHz)
1 = Toggle output

NOTE

After reset, the $\overline{\text{TOUT}}$ signal begins in a high state, but is not available externally until the PBCNT register is configured for this function.

CE—Capture Edge and Enable Interrupt

00 = Capture function is disabled
01 = Capture on rising edge only and enable interrupt on capture event
10 = Capture on falling edge only and enable interrupt on capture event
11 = Capture on any edge and enable interrupt on capture event

PS—Prescaler Value

The prescaler is programmed to divide the clock input by values from 1 to 256. The value 00000000 divides the clock by 1; the value 11111111 divides the clock by 256. The resolution of the timer varies directly with the size of the prescaler. In order to make smaller adjustments to the timer as needed, the prescaler should be as small as possible (see 3.5.2.6 General Purpose Timer Example).

3.5.2.2 Timer Reference Registers (TRR1, TRR2)

Each TRR is a 16-bit register containing the reference value for the timeout. TRR1 and TRR2 are memory-mapped read-write registers.

When working in the MC68008 mode (BUSW is low), writing the high byte of TRR1 and TRR2 will disable the timer's compare logic until the low byte is written.

TRR1 and TRR2 are set to all ones by reset. The reference value is not “reached” until TCN increments to equal TRR.

3.5.2.3 Timer Capture Registers (TCR1, TCR2)

Each TCR is a 16-bit register used to latch the value of the counter during a capture operation when an edge occurs on the respective TIN1 or TIN2 pin. TCR1 and TCR2 appear as memory-mapped read-only registers to the user.

When working in the MC68008 mode (BUSW is low), reading the high byte of TCR1 and TCR2 will disable the timer's capture logic until the low byte is read.

TCR1 and TCR2 are cleared at reset.

3.5.2.4 Timer Counter (TCN1, TCN2)

TCN1 and TCN2 are 16-bit up-counters. Each is memory-mapped and can be read and written by the user. A read cycle to TCN1 and TCN2 yields the current value of the timer and does not affect the counting operation.

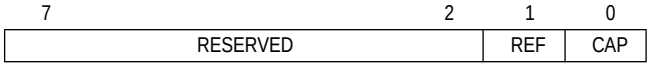
When working in the MC68008 mode (BUSW is low), reading the high byte of TCN1 and TCN2 will latch the low byte into a temporary register; a subsequent read cycle on the low byte yields the value of the temporary register.

A write cycle to TCN1 and TCN2 causes both the counter register and the corresponding prescaler to be reset to zero. In MC68008 mode (BUSW is low), a write cycle to either the high or low byte of the TCN will reset the counter register and the corresponding prescaler to zero.

3.5.2.5 Timer Event Registers (TER1, TER2)

Each TER is an 8-bit register used to report events recognized by any of the timers. On recognition of an event, the timer will set the appropriate bit in the TER, regardless of the corresponding interrupt enable bits (ORI and CE) in the TMR. TER1 and TER2, which appear to the user as memory-mapped registers, may be read at any time.

A bit is cleared by writing a one to that bit (writing a zero does not affect a bit's value). More than one bit may be cleared at a time. Both bits must be cleared before the timer will negate the INRQ to the interrupt controller. This register is cleared at reset.



CAP—Capture Event

The counter value has been latched into the TCR. The CE bits in the TMR are used to enable the interrupt request caused by this event.

REF—Output Reference Event

The counter has reached the TRR value. The ORI bit in the TMR is used to enable the interrupt request caused by this event.

Bits 7–2—Reserved for future use.

3.5.2.6 General Purpose Timer Example

This section gives two examples on how to program the general purpose timers.

3.5.2.6.1 Timer Example 1

Generate an interrupt every 10 mS using the 20 MHz system clock.

1. Take the desired interrupt period and divide by the timer clock period to get an initial count value to calculate prescaler.

$$\frac{Tout}{Tin} = \frac{10ms}{\frac{1}{20MHz}} = Count = 200,000$$

2. To calculate the value for the clock divider, divide the count by 65536 (2^{16}).

$$\frac{Count}{65536} = Divider = 3.05176$$

3. The divider must be rounded up to the next integer value. A clock divider of 4 then changes the input timer period to Tin*4. A new count is calculated based on the new timer period, and this value will be written to the TRR. The prescaler in the TMR is equal to the clock divider minus 1 (or 4-1 = 3).

$$\frac{Tout}{Tin(Divider)} = \frac{10ms}{50ns(4)} = 50,000$$

4. Program the TRR to $C350 (= 50000 decimal).
5. Program the TMR to $031B (prescaler = 3, ORI =1 to enable interrupt, FRR = 1 to re-start counter after reference is reached, ICLK = 01 to use the master clock, and RST = 1 to enabled the timer).

Fine adjustments can be made to the timer by varying the TRR up or down.

3.5.2.6.2 Timer Example 2

Generate a 100 Hz square wave using the 20 MHz system clock. As in Timer Example 1, the period is 10 mS, so we can use the same Prescaler and Reference values. When OM is set, the TOUT pin only toggles when the reference value is reached. Therefore the reference value must be divided by two in order to generate two edges every 100 mS.

1. Program the Port B control register to change the port pin from a general purpose input pin to TOUT.
2. Program the TRR to $61A8 (= 50000/2).
3. Program the TMR to $321B (prescaler = 3, OM =1 to toggle TOUT, FRR = 1 to restart

counter after reference is reached, ICLK = 01 to use the master clock, and RST = 1 to enabled the timer).

Fine adjustments can be made to the timer by varying the TRR up or down.

3.5.3 Timer 3 - Software Watchdog Timer

A watchdog timer is used to protect against system failures by providing a means to escape from unexpected input conditions, external events, or programming errors. Timer 3 may be used for this purpose. Once started, the watchdog timer must be cleared by software on a regular basis so that it never reaches its timeout value. Upon reaching the timeout value, the assumption may be made that a system failure has occurred, and steps can be taken to recover or reset the system.

3.5.3.1 Software Watchdog Timer Operation

The watchdog timer counts from zero to a maximum of 32767 (16.67 seconds at 16.00 MHz) with a resolution or step size of 8192 clock periods (0.5 ms at 16.00 MHz). This timer uses a 16-bit counter with an 8-bit prescaler value.

The watchdog timer uses the main clock divided by 16 as the input to the prescaler. The prescaler circuitry divides the clock input by a fixed value of 256. The output of this prescaler circuitry is connected to the input of the 16-bit counter. Since the least significant bit of the WCN is not used in the comparison with the WRR reference value, the effective value of the prescaler is 512.

The timer counts until the reference value is reached and then starts a new time count immediately. Upon reaching the reference value, the counter asserts the $\overline{\text{WDOG}}$ output for a period of 16 master clock (CLKO) cycles, and issues an interrupt to the interrupt controller. The value of the timer can be read any time.

To use the software watchdog function directly with the M68000 core, the timer 3 open-drain output pin ($\overline{\text{WDOG}}$) can be connected externally to the $\overline{\text{IPL2}}$–$\overline{\text{IPL0}}$ pins to generate a level 7 interrupt (normal mode), to $\overline{\text{IRQ7}}$ (dedicated mode), or to the $\overline{\text{RESET}}$ and $\overline{\text{HALT}}$ pin. After a total system reset, the $\overline{\text{WDOG}}$ pin function is enabled on pin PB7. The timer 3 counter is automatically enabled after reset.

The software watchdog timer has an 8-bit prescaler that is not accessible to the user, a read-only 16-bit counter, and a reference register (WRR).

3.5.3.2 Software Watchdog Reference Register (WRR)

WRR is a 16-bit register containing the reference value for the timeout. The EN bit of the register enables the timer. WRR appears as a memory-mapped read-write register to the user.

When operating in the MC68008 mode (BUSW is low), writing to the high byte of WRR will disable the timer compare logic until the low byte is written.

Reset initializes the register to $FFFF, enabling the watchdog timer and setting it to the maximum timeout period. This causes a timeout to occur if there is an error in the boot program.

15	1	0
REFERENCE VALUE		EN

3.5.3.3 Software Watchdog Counter (WCN)

WCN, a 16-bit up-counter, appears as a memory-mapped register and may be read at any time. Clearing EN in WRR causes the counter to be reset and disables the count operation.

A read cycle to WCN causes the current value of the timer to be read. When working in MC68008 mode (BUSW is low), reading the high byte of WCN will latch the low byte into a temporary register. When reading the low byte, the temporary register value is read. Reading the timer does not affect the counting operation.

A write cycle to WCN causes the counter and prescaler to be reset. In the MC68008 mode (BUSW is low), a write cycle to either the high or low byte resets the counter and the prescaler. A write cycle should be executed on a regular basis so that the watchdog timer is never allowed to reach the reference value during normal program operation.

3.6 EXTERNAL CHIP-SELECT SIGNALS AND WAIT-STATE LOGIC

The MC68302 provides a set of four programmable chip-select signals. Each chip-select signal has an identical internal structure. For each memory area, the user may also define an internally generated cycle termination signal ($\overline{\text{DTACK}}$). This feature eliminates board space that would be necessary for cycle termination logic.

The four chip-select signals allow four different classes of memory to be used: e.g., high-speed static RAM, slower dynamic RAM, EPROM, and nonvolatile RAM. If more than four chip selects are required, additional chip selects may be decoded externally, as on the MC68000.

The chip-select block diagram is shown in Figure 3-9.

The chip-select logic is active for memory cycles generated by internal bus masters (M68000 core, IDMA, SDMA, DRAM refresh) or external bus masters. These signals are driven externally on the falling edge of $\overline{\text{AS}}$ and are valid shortly after $\overline{\text{AS}}$ goes low.

For each chip select, the user programs the block size by choosing the starting address in the base register and the length in the option register. The starting address must be on a block boundary. Thus, an 8K block size must begin on an 8K address boundary, and a 64K block size must begin on a 64K address boundary, etc.

For a given chip-select block, the user may also choose 1) whether the chip-select block should be considered as read-only, write-only, or read/write, 2) whether the chip-select block should be active on only one particular function code signal combination or for all function codes, and 3) whether a $\overline{\text{DTACK}}$ should be automatically generated for this chip-select block, and after how many wait states.

$\overline{\text{DTACK}}$ generation occurs under the same constraints as the chip-select signal—if the chip-select signal does not activate, then neither will the $\overline{\text{DTACK}}$ signal.

Chip select 0 has the special property of being enabled upon system reset to the address range from 0 to 8K bytes. This property allows chip select 0 to function as the “boot ROM” select on system start-up. $\overline{\text{DTACK}}$ is initially enabled for six wait states on this chip select.

External masters may use the chip-select logic on the IMP during an external master access to external memory/peripherals. In this case, the external master chip-select timing diagram (see Figure 6-15) must be used. Since the chip-select logic is slightly slower when using external masters, an optional provision can be made to add an additional wait state to an external access by an external master. See the EMWS bit in the SCR for more details (3.8.3 System Control Bits).

A priority structure exists within the chip-select block. For a given address, the priority is as follows:

1. Access to any IMP internal address (BAR, dual-port RAM, etc.)
   No chip select asserted.
2. Chip Select 0
3. Chip Select 1
4. Chip Select 2
5. Chip Select 3

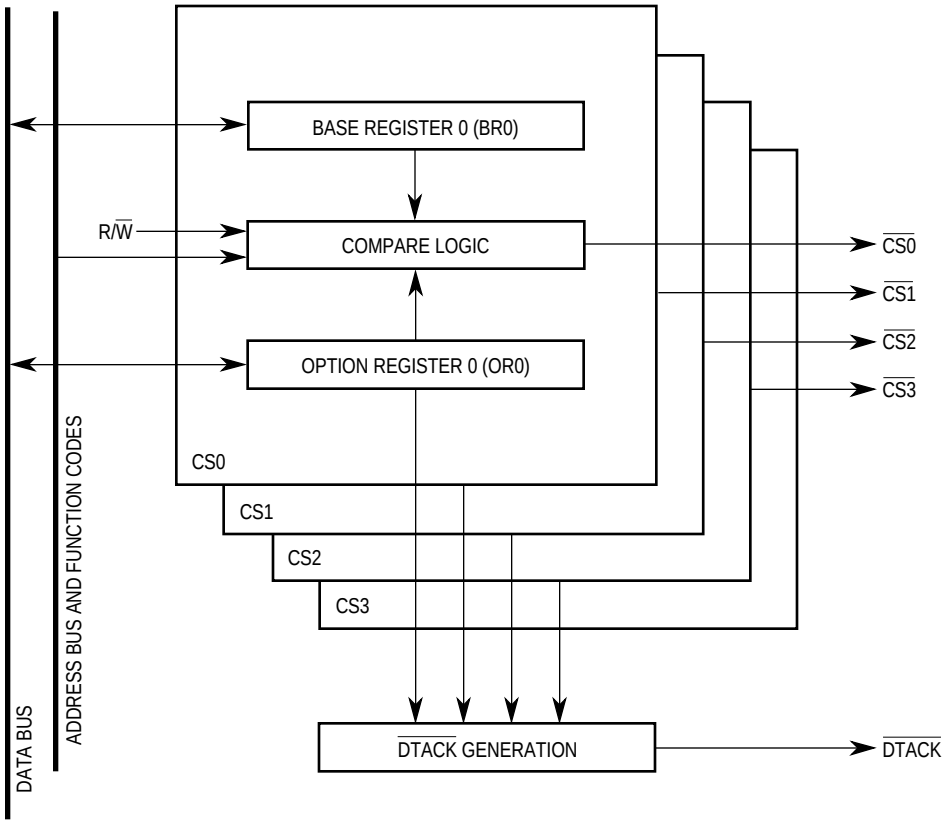


Figure 3-9. Chip-Select Block Diagram

The user should not normally program more than one chip-select line to the same area. When this occurs, the address compare logic will set address decode conflict (ADC) in the system control register (SCR) and generate $\overline{\text{BERR}}$ if address decode conflict enable (ADCE) is set. Only one chip-select line will be driven because of internal line priorities. $\overline{\text{CS0}}$ has the highest priority, and $\overline{\text{CS3}}$ the lowest. $\overline{\text{BERR}}$ will not be asserted on write accesses to the chip-select registers.

If one chip select is programmed to be read-only and another chip select is programmed to be write-only, then there will be no overlap conflict between these two chip selects, and the ADC bit will not be set.

When a bus master attempts to write to a read-only location, the chip-select logic will set write protect violation (WPV) in the SCR and generate $\overline{\text{BERR}}$ if write protect violation enable (WPVE) is set. The $\overline{\text{CS}}$ line will not be asserted.

NOTE

The chip-select logic is reset only on total system reset (assertion of $\overline{\text{RESET}}$ and $\overline{\text{HALT}}$). Accesses to the internal RAM and registers, including the system configuration registers (BAR and

SCR), will not activate the chip-select lines. Thus, it is convenient to use one of the chip-select lines to select external ROM/RAM that overlaps these register addresses, since, in this way, bus contention is completely avoided during a read access to these addresses. If, in a given application, it is not possible to use the chip-select lines for this purpose, the IAC signal may be used externally to prevent bus contention.

NOTE

The chip-select logic does not allow an address match during interrupt acknowledge cycles.

A special case occurs when the locked read-modify-write test and set (TAS) instruction is executed in combination with the chip selects. The assertion of wait states on the write portion of the cycle will only occur if the RMCST bit in the SCR is set. Refer to 3.8.3 System Control Bits for more details.

3.6.1 Chip-Select Logic Key Features

Key features of the chip-select logic are as follows:

- Four Programmable Chip-Select Lines
- Various Block Sizes: 8K, 16K, 32K, 64K, 128K, 256K, 512K, 1M, 2M, 4M, 8M, and 16M Bytes
- Read-Only, Write-Only, or Read-Write Select
- Internal $\overline{\text{DTACK}}$ Generation with Wait-State Options
- Default Line ($\overline{\text{CS0}}$) to Select an 8K-Boot ROM Containing the Reset Vector and Initial Program

3.6.2 Chip-Select Registers

Each of the four chip-select units has two registers that define its specific operation. These registers are a 16-bit base register (BR) and a 16-bit option register (OR) (e.g., BR0 and OR0). These registers may be modified by the M68000 core. The BR should normally be programmed after the OR since the BR contains the chip-select enable bit.

3.6.2.1 Base Register (BR3–BR0)

These 16-bit registers consist of a base address field, a read-write bit, and a function code field.

FC2–FC0 —Function Code Field

This field is contained in bits 15–13 of each BR. These bits are used to set the address space function code. The address compare logic uses these bits to determine whether an

address match exists within its address space and, therefore, whether to assert the chip-select line.

111 = Not supported; reserved. Chip select will not assert if this value is chosen.
110 = Value may be used.
•
•
•
000 = Value may be used.

After system reset, the FC field in BR3–BR0 defaults to supervisor program space (FC = 110) to select a ROM device containing the reset vector. Because of the priority mechanism and the EN bit, only the $\overline{CS0}$ line is active after a system reset.

NOTE

The FC bits can be masked and ignored by the chip-select logic using CFC in the OR.

Bits 12–2—Base Address

These bits are used to set the starting address of a particular address space. The address compare logic uses only A23–A13 to cause an address match within its block size. The base address should be located on a block boundary. For example, if the block size is 64k bytes, then the base address should be a multiple of 64k.

After system reset, the base address defaults to zero to select a ROM device on which the reset vector resides. All base address values default to zero on system reset, but, because of the priority mechanism, only $\overline{CS0}$ will be active.

NOTE

All address bits can be masked and ignored by the chip-select logic through the base address mask in the OR.

RW—Read/Write

0 = The chip-select line is asserted for read operations only.
1 = The chip-select line is asserted for write operations only.

After system reset, this bit defaults to zero (read-only operation).

NOTE

This bit can be masked and ignored by the read-write compare logic, as determined by MRW in the OR. The line is then asserted for both read and write cycles.

On write protect violation cycles (RW = 0 and MRW = 1), $\overline{BERR}$ will be generated if WPVE is set, and WPV will be set.

If the write protect mechanism is used by an external master, the R/$\overline{W}$ low to $\overline{AS}$ asserted timing should be 16 ns minimum.

EN—Enable

0 = The chip-select line is disabled.
1 = The chip-select line is enabled.

After system reset, only $\overline{CS0}$ is enabled; $\overline{CS3}$–$\overline{CS1}$ are disabled. In disable CPU mode, $\overline{CS3}$–$\overline{CS0}$ are disabled at system reset. The chip select does not require disabling before changing its parameters.

3.6.2.2 Option Registers (OR3–OR0)

These four 16-bit registers consist of a base address mask field, a read/write mask bit, a compare function code bit, and a $\overline{DTACK}$ generation field.

Bits 15–12—DTACK Field

These bits are used to determine whether $\overline{DTACK}$ is generated internally with a programmable number of wait states or externally by the peripheral. With internal $\overline{DTACK}$ generation, zero to six wait states can be automatically inserted before the $\overline{DTACK}$ pin is asserted as an output (see Port A Control Register (PACNT)).

Table 3-8. DTACK Field Encoding

Bits			Description
15	14	13	
0	0	0	No Wait State
0	0	1	1 Wait State
0	1	0	2 Wait States
0	1	1	3 Wait States
1	0	0	4 Wait States
1	0	1	5 Wait States
1	1	0	6 Wait States
1	1	1	External DTACK

When all the bits in this field are set to one, $\overline{DTACK}$ must be generated externally, and the IMP or external bus master waits for $\overline{DTACK}$ (input) to terminate its bus cycle. After system reset, the bits of the DTACK field default to six wait states.

The DTACK generator uses the IMP internal clock to generate the programmable number of wait states. For asynchronous external bus masters, the programmable number of wait states is counted directly from the internal clock. When no wait state is programmed (DTACK = 000), the DTACK generator will generate $\overline{DTACK}$ asynchronously.

The $\overline{CS}$ lines are asserted slightly earlier for internal IMP master memory cycles than for an external master using the $\overline{CS}$ lines. Set external master wait state (EMWS) in the SCR whenever these timing differences require an extra memory wait state for external masters.

NOTE

Do not assert $\overline{\text{DTACK}}$ externally when it is programmed to be generated internally.

Bits 12–2—Base Address Mask

These bits are used to set the block size of a particular chip-select line. The address compare logic uses only the address bits that are not masked (i.e., mask bit set to one) to detect an address match within its block size.

0 = The address bit in the corresponding BR is masked; the address compare logic does not use this address bit. The corresponding external address line value is a don't care in the comparison.
1 = The address bit in the corresponding BR is not masked; the address compare logic uses this address bit.

For example, for a 64K-byte block, this field should be M13, M14, M15 = 0 with the rest of the base address mask bits (M23–M16) equal to one.

After system reset, the bits of the base address mask field default to ones (selecting the smallest block size of 8K) to allow $\overline{\text{CS0}}$ to select the ROM device containing the reset vector.

MRW—Mask Read/Write

0 = The RW bit in the BR is masked. The chip select is asserted for both read and write operations.
1 = The RW bit in the BR is not masked. The chip select is asserted for read-only or write-only operations as programmed by the corresponding RW bit in BR3–BR0.

After system reset, this bit defaults to zero.

CFC—Compare Function Code

0 = The FC bits in the BR are ignored. The chip select is asserted without comparing the FC bits. If the application requires the user to recognize several address spaces (e.g., user space without distinguishing between data and program space), FC bits must be decoded externally.
1 = The FC bits on the BR are compared. The address space compare logic uses the FC bits to assert the $\overline{\text{CS}}$ line.

After system reset, this bit defaults to one.

NOTE

Even when CFC = 0, if the function code lines are internally or externally generated as “111”, the chip select will not be asserted.

3.6.3 Chip Select Example

Set up chip select 2 to assert for a 1 Megabyte block of external RAM beginning at $200000 with 1 wait state. Note that the address must be on a block boundary (i.e. the starting address of a 1 Megabyte block could not be $210000).

1. Calculate what the mask should be. For a 1 Megabyte block, the address lines A0 through A19 are used to address bytes within the block, so they need to be masked out.
2. Write $3E00 to OR2 (DTACK=1 for 1 wait state, M23-M20 = 1 to use these bits in the comparison, M19-M13 = 0 to mask these address bits, MRW = 0 to enable the chip select for both read and write, and CFC = 0 to mask off function code comparison).
3. Write $0401 to BR2 (FC2-FC0 = 0 don't care, A23-A13 = base address, RW = 0 don't care, and EN =1 to enable the chip select.

NOTE

The mask bits in the OR are used to mask the individual address bits, so in the previous example, if bit 12 (M23) was changed to a zero, then CS2 would assert for a 1 Megabyte block beginning at $200000 and a 1 Megabyte block at $A00000.

3.7 ON-CHIP CLOCK GENERATOR

The IMP has an on-chip clock generator that supplies clocks to both the internal M68000 core and peripherals and to an external pin. The clock circuitry uses three dedicated pins: EXTAL, XTAL, and CLKO.

The external clock/crystal (EXTAL) input provides two clock generation options. EXTAL may be used to interface the internal generator to an external crystal (see Figure 3-10). Typical circuit parameters are C1 = C2 = 25 pF and R = 700 kΩ using a parallel resonant crystal. Typical crystal parameters are Co < 10 pF and Rx = 50 Ω. The equivalent load capacitance (C_L) of this circuit is 20 pF, calculated as (C1 + C_{in})/2, where C1 = C2 = 25 pF and C_{in} = 15 pF maximum on the EXTAL pin.

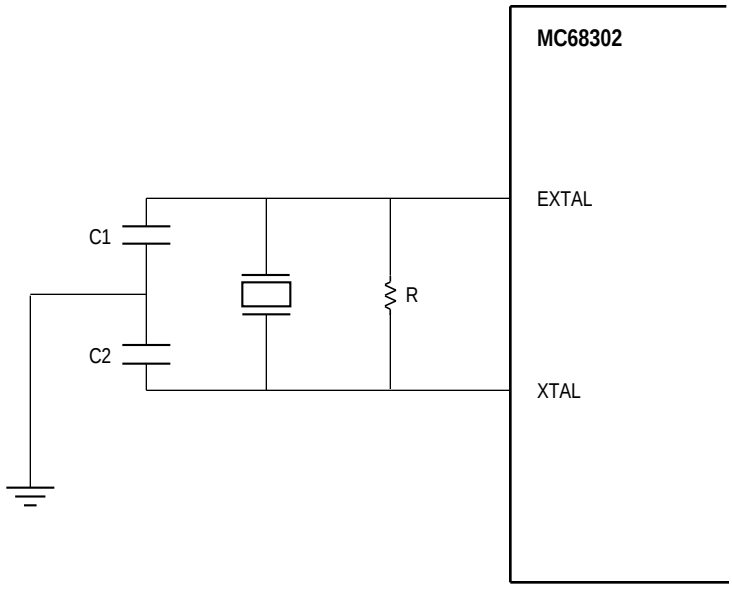


Figure 3-10. Using an External Crystal

EXTAL can also accept a CMOS-level clock input. The crystal output (XTAL) connects the internal crystal generator output to an external crystal. If an external clock is used, XTAL should be left unconnected. The CLKO pin, which drives the high-speed system clock, may be used to synchronize other peripherals to the IMP system clock.

3.8 SYSTEM CONTROL

The IMP system control consists of a System Control Register (SCR) that configures the following functions:

- System Status and Control Logic
- $\overline{AS}$ Control During Read-Modify-Write-Cycles
- Disable CPU (M68000) Logic
- Bus Arbitration Logic with Low-Interrupt Latency Support
- Hardware Watchdog
- Low-Power (Standby) Modes
- Freeze Control

3.8.1 System Control Register (SCR)

The SCR is a 32-bit register that consists of system status and control bits, a bus arbiter control bit, hardware watchdog control bits, low-power control bits, and freeze select bits. Refer to Figure 3-11, Table 3-7, and to the following paragraphs for a description of each bit in this register. The SCR is a memory-mapped read-write register. The address of this register is fixed at $0F4 in supervisor data space (FC = 5).

31	30	29	28	27	26	25	24
0	0	0	0	IPA	HWT	WPV	ADC

23	22	21	20	19	18	17	16
0	ERRE	VGE	WPVE	RMCST	EMWS	ADCE	BCLM

15	14	13	12	11	10	8
FRZW	FRZ2	FRZ1	SAM	HWDEN	HWDCN2–HWDCN0	

7	6	5	4	0
LPREC	LPP16	LPEN	LOW-POWER CLOCK DIVIDER	

Figure 3-11. System Control Register

Table 3-9. SCR Register Bits

Bit	Name	Section(s)
IPA	Interrupt Priority Active	3.8.2
HWT	Hardware Watchdog Timeout	3.8.2, 3.8.6
WPV	Write Protect Violation	3.8.2
ADC	Address Decode Conflict	3.8.2
ERRE	External RISC Request Enable	3.9
VGE	Vector Generation Enable	3.8.4
WPVE	Write Protect Violation Enable	3.8.3
RMCST	Read-Modify-Write Cycle Special Treatment	3.8.3
EMWS	External Master Wait State	3.8.3, 3.8.4
ADCE	Address Decode Conflict Enable	3.8.3
BCLM	Bus Clear Mask	3.8.2, 3.8.3, 3.8.5
FRZW	Freeze Watchdog Timer Enable	3.8.8
FRZ1	Freeze Timer 1 Enable	3.8.8
FRZ2	Freeze Timer 2 Enable	3.8.8
SAM	Synchronous Access Mode	3.8.3, 3.8.4
HWDEN	Hardware Watchdog Enable	3.8.6
HWDCN	Hardware Watchdog Count	3.8.6
LPREC	Low-Power Recovery	3.8.7
LPP16	Low-Power Clock Prescale Divide by 16	3.8.7
LPEN	Low-Power Enable	3.8.7
LPCD	Low-Power Clock Divider Selects	3.8.7

3.8.2 System Status Bits

The eight most significant bits of the SCR are used to report events recognized by the system control logic. On recognition of an event, this logic sets the corresponding bit in the SCR. The bits may be read at any time. A bit is reset by one and is left unchanged by zero. More than one bit may be reset at a time.

After system reset (simultaneous assertion of $\overline{\text{RESET}}$ and $\overline{\text{HALT}}$), these bits are cleared.

IPA—Interrupt Priority Active

This bit is set when the M68000 core has an unmasked interrupt request. When bus clear mask (BCLM) is set, $\overline{\text{BCLR}}$ and the internal bus clear to the IDMA are asserted.

NOTE

If BCLM is set, an interrupt handler will normally clear IPA at the end of the interrupt routine to allow an alternate bus master to regain the bus; however, if BCLM is cleared, no additional action need be taken in the interrupt handler.

In the case of nested interrupts, the user may wish to clear the IPA bit only at the end of the original lower priority interrupt routine to keep $\overline{\text{BCLR}}$ asserted until it completes. To guarantee that

this happens and that other pending interrupts at the same original priority level also execute with $\overline{BCLR}$ continuously asserted, the following technique may be used. Using a parallel I/O line connected to the IRQ1 line, the original priority level interrupt toggles this I/O line just before it executes the RTE instruction, causing a request for a level 1 interrupt. Since this is the lowest interrupt level, this routine will not be executed until all other pending interrupt routines have executed. Then in the level 1 interrupt routine, the IPA bit in the SCR is cleared.

HWT—Hardware Watchdog Timeout

This bit is set when the hardware watchdog (see 3.8.6 Hardware Watchdog) reaches the end of its time interval; $\overline{BERR}$ is generated following the watchdog timeout, even if this bit is already set.

WPV—Write Protect Violation

This bit is set when a bus master attempts to write to a location that has RW set to zero (read only) in its associated base register (BR3–BR0). Provided WPVE (bit 20) is set, $\overline{BERR}$ will be asserted on the bus cycle that sets this bit. If WPV and WPVE are both set when a write protect violation occurs, $\overline{BERR}$ will still be generated.

ADC—Address Decode Conflict

This bit is set when a conflict has occurred in the chip-select logic because two or more chip-select lines attempt assertion in the same bus cycle. This conflict may be caused by a programming error in which the user-allocated memory areas for each chip select overlap each other. Provided ADCE (bit 17) is set, the occurrence of ADC will cause $\overline{BERR}$ to be asserted. If this bit is already set when another address decode conflict occurs, $\overline{BERR}$ will still be generated. The chip-select logic will protect the IMP from issuing two simultaneous chip selects by employing a priority system.

NOTE

Regardless of the state of the chip-select programming, this bit will not be set and $\overline{BERR}$ will not be asserted for an address decode conflict occurring during access to a system configuration register. This is provided to guarantee access to the system configuration registers (BAR and SCR) during initialization.

3.8.3 System Control Bits

The system control logic uses six control bits in the SCR.

WPVE—Write Protect Violation Enable

0 = $\overline{BERR}$ is not asserted when a write protect violation occurs.
1 = $\overline{BERR}$ is asserted when a write protect violation occurs.

After system reset, this bit defaults to zero.

NOTE

WPV will be set, regardless of the value of WPVE.

RMCST—RMC Cycle Special Treatment

0 = The locked read-modify-write cycles of the TAS instruction will be identical to the M68000 ($\overline{AS}$ and $\overline{CS}$ will be asserted during the entire cycle). The arbiter will issue $\overline{BG}$, regardless of the M68000 core $\overline{RMC}$. If an IMP chip select is used, the $\overline{DTACK}$ generator will insert wait states on the read cycle only.

1 = The MC68302 uses $\overline{RMC}$ to negate $\overline{AS}$ and $\overline{CS}$ at the end of the read portion of the RMC cycle and reasserts $\overline{AS}$ and $\overline{CS}$ at the beginning of the write portion. $\overline{BG}$ will not be asserted until the end of the write portion. If an IMP chip select is used, the $\overline{DTACK}$ generator will insert wait states on both the read and write portion of the cycles.

The assertion of the $\overline{RMC}$ by the M68000 core is seen by the arbiter and will prevent the arbiter from issuing bus grants until the completion of M68000-initiated locked read-modify-write activity. After system reset, this bit defaults to zero.

EMWS—External Master Wait State (EMWS);

When EMWS is set and an external master is using the chip-select logic for $\overline{DTACK}$ generation or is synchronously reading from the internal peripherals (SAM = 1), one additional wait state will be inserted in external master cycle to external memory and peripherals and also in every cycle from the external master to MC68302 internal memory and peripherals. When EMWS is cleared, all synchronous internal accesses will be with zero wait states, and the chip-select logic will generate $\overline{DTACK}$ after the exact programmed number of wait states. The chip-select lines are asserted slightly earlier for internal master memory cycles than for an external master. EMWS should be set whenever these timing differences will necessitate an additional wait state for external masters. After system reset, this bit defaults to zero.

ADCE—Address Decode Conflict Enable

0 = $\overline{BERR}$ is not asserted by a conflict in the chip-select logic when two or more chip-select lines are programmed to overlap the same area.

1 = $\overline{BERR}$ is asserted by a conflict in the chip-select logic when two or more chip-select lines are programmed to overlap the same area.

After system reset, this bit defaults to zero.

NOTE

ADC will be set, regardless of the value of ADCE.

BCLM—Bus Clear Mask

0 = The arbiter does not use the M68000 core internal IPEND signal to assert the internal and external bus clear signals.

1 = The arbiter uses the M68000 core internal IPEND signal to assert the internal and external bus clear signals.

After system reset, this bit defaults to zero. If BCLM is set, then the typical maximum interrupt latency is about 78 clocks in a zero-wait-state system. This assumes a standard instruction mix, that the IDMA is just beginning a four-bus-cycle transfer when the interrupt becomes pending, and that an SDMA has an access pending (one bus cycle). Interrupt execution time is 44 clocks and includes the time to execute the interrupt acknowledge cycle, save the status register and PC value on the stack, and then vector to the first location of the interrupt service routine. Thus, the calculation is 78 = 14 (instruction completion) + 20 (DMAs) + 44 (interrupt execution).

SDMA operation is not affected by the BCLM bit. Note that the SDMA accesses only one byte/word of external memory at a time before giving up the bus and that accesses are relatively infrequent. External bus master operation may or may not be affected by the BCLM bit, depending on whether the $\overline{\text{BCLR}}$ signal is used to clear the external master off the bus.

Without using the BCLM bit, the maximum interrupt latency includes the maximum time that the IDMA or external bus master could use the bus in the worst case. Note that the IDMA can limit its bus usage if its requests are generated internally.

NOTE

The IPA status bit will be set, regardless of the BCLM value.

SAM—Synchronous Access Mode

This bit controls how external masters may access the MC68302 peripheral area. This bit is not relevant for applications that do not have external bus masters that access the MC68302. In applications such as disable CPU mode, in which the M68000 core is not operating, the user should note that SAM may be changed by an external master on the first access of the MC68302, but that first write access must be asynchronous with three wait states. (If $\overline{\text{DTACK}}$ is used to terminate bus cycles, this change need not influence hardware.)

0 = Asynchronous accesses. All accesses to the MC68302 internal RAM and registers (including BAR and SCR) by an external master are asynchronous to the MC68302 clock. Read and write accesses are with three wait states, and $\overline{\text{DTACK}}$ is asserted by the MC68302 assuming three wait-state accesses. This is the default value.

1 = Synchronous accesses. All accesses to the MC68302 internal RAM and registers (including BAR and SCR) must be synchronous to the MC68302 clock. Synchronous read accesses may occur with one wait state if EMWS is also set to one.

3.8.4 Disable CPU Logic (M68000)

The MC68302 can be configured to operate solely as a peripheral to an external processor. In this mode, the on-chip M68000 CPU should be disabled by strapping DISCPU high during system reset ($\overline{\text{RESET}}$ and $\overline{\text{HALT}}$ asserted simultaneously). The internal accesses to the MC68302 peripherals and memory may be asynchronous or synchronous. During synchronous reads, one wait state may be used if required (EMWS bit set). The following pins change their functionality in this mode:

1. $\overline{\text{BR}}$ will be an output from the IDMA and SDMA to the external M68000 bus, rather than being an input to the MC68302.

2. $\overline{BG}$ will be an input to the IDMA and SDMA from the external M68000 bus, rather than being an output from the MC68302. When BG is sampled as low by the MC68302, it waits for $\overline{AS}$, $\overline{BERR}$, $\overline{HALT}$, and $\overline{BGACK}$ to be negated, and then asserts $\overline{BGACK}$ and performs one or more bus cycles. See Section 6 for timing diagrams.
3. $\overline{BCLR}$ will be an input to the IDMA, but will remain an output from the SDMA.
4. The interrupt controller will output its interrupt request lines ($\overline{IPL0}$, $\overline{IPL1}$, $\overline{IPL2}$) normally sent to the M68000 core on pins $\overline{IOUT0}$, $\overline{IOUT1}$, and $\overline{IOUT2}$, respectively. $\overline{AVEC}$, $\overline{RMC}$, and $\overline{CS0}$, which share pins with $\overline{IOUT0}$, $\overline{IOUT1}$, and $\overline{IOUT2}$, respectively, are not available in this mode.

DISCPU should remain continuously high during disable CPU mode operation. Although the $\overline{CS0}$ pin is not available as an output from the device in disable CPU mode, it may be enabled to provide $\overline{DTACK}$ generation. In disable CPU mode, BR0 is initially $C000.

Accesses by an external master to the MC68302 RAM and registers may be asynchronous or synchronous to the MC68302 clock. (This feature is actually available regardless of disable CPU mode). See the SAM and EMWS bits in the SCR for details.

In disable CPU mode, the interrupt controller may be programmed to generate or not generate interrupt vectors during interrupt acknowledge cycles. When multiple MC68302 devices share a single M68000 bus, vector generation at level 4 should be prevented on all but one MC68302. When using disable CPU mode to implement an interface, such as between the MC68020 and a single MC68302, vector generation can be enabled. For this purpose, the VGE bit is defined.

VGE—Vector Generation Enable

0 = In disable CPU mode, the MC68302 will not output interrupt vectors during interrupt acknowledge cycles.
1 = In disable CPU mode, the MC68302 will output interrupt vectors for internal level 4 interrupts (and for levels 1, 6, and/or 7 as enabled in the interrupt controller) during interrupt acknowledge cycles.

NOTE

Do not use the function code value “111” during external accesses to the IMP, except during interrupt acknowledge cycles.

In disable CPU mode, the low-power modes will be entered immediately upon the setting of the LPEN bit in the SCR by an external master. In this case, low-power mode will continue until the LPEN bit is cleared. Users may wish to use a low-power mode in conjunction with disable CPU mode to save power consumed by the disabled M68000 core.

All MC68302 functionality not expressly mentioned in this section is retained in disable CPU mode and operates identically as before.

NOTE

Even without the use of the disable CPU logic, another processor can be granted access to the IMP on-chip peripherals by re-

questing the M68000 bus with $\overline{BR}$. See 3.8.6 Hardware Watchdog for further details.

3.8.5 Bus Arbitration Logic

Both internal and external bus arbitration are discussed in the following paragraphs.

3.8.5.1 Internal Bus Arbitration

The IMP bus arbiter supports three bus request sources in the following standard priority:

1. External bus master ($\overline{BR}$ pin)
2. SDMA for the SCCs (six channels)
3. IDMA (one channel)

When one of these sources desires the bus, the M68000 core will be forced off through an internal bus request signal ($\overline{CBR}$) from the bus arbiter to the M68000 core (see Figure 3-12). When the arbiter detects the assertion of the M68000 core bus grant ($\overline{CBG}$) signal, it asserts the requester's bus grant signal according to its priority. Thus, as seen externally, the SDMA and IDMA channels do not affect $\overline{BR}$ and $\overline{BG}$, but only $\overline{BGACK}$ (unless disable CPU mode is used).

The MC68302 provides several options for changing the preceding bus master priority list. The options are configured by setting the BCLM bit in the SCR and deciding whether or not the $\overline{BCLR}$ pin is used externally to cause external bus masters to relinquish the bus (see Table 3-10).

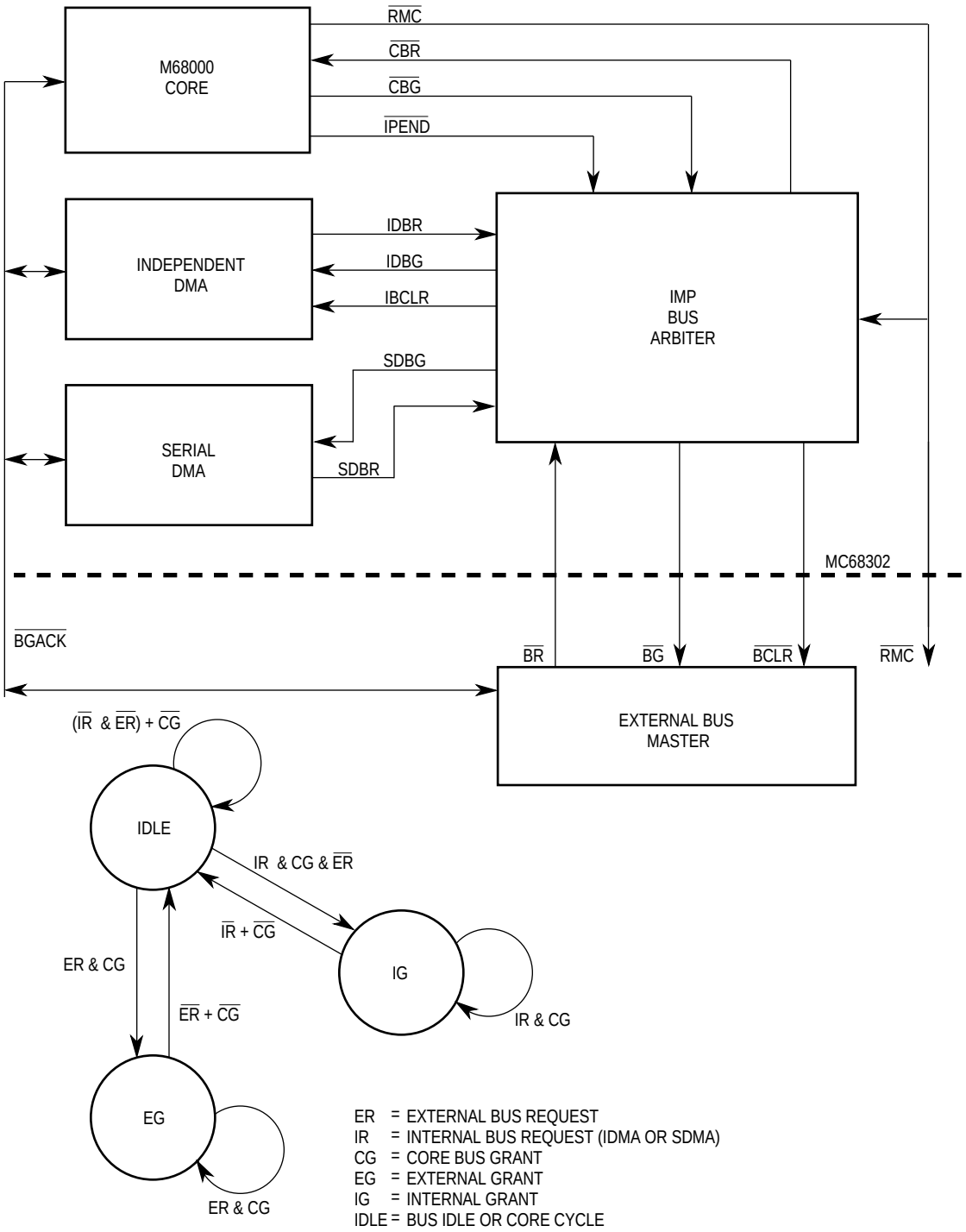


Figure 3-12. IMP Bus Arbiter

Table 3-10. Bus Arbitration Priority Table

$\overline{\text{BCLR}}$ Ignored BCLM = 0	$\overline{\text{BCLR}}$ Used BCLM = 0	$\overline{\text{BCLR}}$ Ignored BCLM = 1	$\overline{\text{BCLR}}$ Used BCLM = 1
$\overline{\text{BR}}$ Pin SDMA IDMA M68000 Interrupts M68000	SDMA IDMA $\overline{\text{BR}}$ Pin M68000 Interrupts M68000	$\overline{\text{BR}}$ Pin SDMA M68000 Interrupts IDMA[4] M68000	SDMA M68000 Interrupts IDMA[4] $\overline{\text{BR}}$ Pin M68000

NOTES:
1. The SDMA on a given IMP always has a higher priority than the IDMA on that IMP.
2. This table assumes the M68000 core is not in disable CPU mode. In disable CPU mode, the SDMA and IDMA make requests to the M68000 bus when they wish to become bus masters.
3. "BCLR Used" means that the BCLR pin is used externally to force the external bus master off the bus, even though its priority is still the highest in the system from the standpoint of the IMP bus arbiter.
4. The bus arbitration priority for IDMA in the two rightmost columns of the table applies only to the case when the IDMA request is internally generated; for the cases of external request, the bus arbitration priority of IDMA is right below that of SDMA.

The IMP bus arbiter also supports an M68000 core low-interrupt latency option. When the M68000 core processor has an unmasked interrupt request, it asserts an internal interrupt pending signal (IPEND). The bus arbiter uses this signal according to BCLM in the SCR to assert external ($\overline{\text{BCLR}}$) and internal bus-clear (IBCLR) signals. These bus-clear signals allow the M68000 core to eliminate long latencies potentially associated with an external bus master or the IDMA, respectively.

The external $\overline{\text{BCLR}}$ is asserted whenever 1) one of the SDMA channels requests the bus when the IDMA is not the bus master or 2) the M68000 core has an unmasked pending interrupt request, provided BCLM in the SCR is set. In this case, $\overline{\text{BCLR}}$ will be asserted until the interrupt priority active (IPA) bit in the SCR is cleared. To implement this feature, $\overline{\text{BCLR}}$ would be used to force external devices to release bus ownership.

IBCLR to the IDMA is asserted whenever 1) an external bus master requests the bus ($\overline{\text{BR}}$ asserted); 2) the M68000 core has an unmasked pending interrupt request, provided BCLM in the SCR is set and the IDMA request is internally generated, and in this case, BCLR will be asserted until IPA is cleared (Note that $\overline{\text{BCLR}}$ could be used to negate $\overline{\text{DREQ}}$ when the IDMA is in external request mode); 3) the M68000 CPU is disabled, and BCLR is asserted.

The IBCLR signal causes the IDMA to release bus ownership at the end of the current operand transfer. IBCLR is not routed to the SDMA channels since they always release bus ownership after one operand transfer.

RMC is issued by the M68000 core and can be used by the internal bus arbiter to delay issuance of BG during read-modify-write cycles. This is controlled by the RMCST bit in the SCR. Otherwise, the MC68000/MC68008 core may be forced off the bus after any bus cycle.

3.8.5.2 External Bus Arbitration

An external bus master may gain ownership of the M68000 bus by asserting the bus request ($\overline{\text{BR}}$) pin. After gaining ownership, it may access the IMP registers or RAM or any system memory address. Chip selects and system control functions, such as the hardware watchdog, continue to operate.

When an external master desires to gain ownership, the standard M68000 bus arbitration protocol should be used:

1. Issue $\overline{BR}$ (to the IMP on-chip bus arbiter).
2. Wait for $\overline{BG}$ (from the IMP on-chip bus arbiter).
3. When $\overline{BG}$ is asserted, wait for the negation of both $\overline{AS}$ and $\overline{BGACK}$.
4. Assert $\overline{BGACK}$ and begin external master bus cycles.
5. Negate $\overline{BR}$ (to the IMP on-chip bus arbiter), causing $\overline{BG}$ to be negated by the IMP on-chip bus arbiter.
6. Negate $\overline{BGACK}$ after the external master bus cycles have completed.

This protocol is also followed by the on-chip bus masters (IDMA, SDMA, and DRAM refresh) except that they request the bus internally from the on-chip bus arbiter.

In the disable CPU mode, the IMP makes requests for the bus rather than granting the bus. In such a system, the IMP functions as an external master, and the external processor (e.g., an MC68030) need not assert $\overline{BGACK}$ as it accesses the IMP's on-chip RAM and registers.

NOTE

When the IMP's BUSW pin is low causing the M68000 core to operate as an MC68008, the $\overline{BGACK}$ signal should still be used in bus arbitration control. On the original MC68008, the $\overline{BGACK}$ signal was not available externally, and therefore could not be used.

3.8.6 Hardware Watchdog

The hardware watchdog logic is used to assert $\overline{BERR}$ and set HWT when a bus cycle is not terminated by $\overline{DTACK}$ and after a programmable number of clock cycles has elapsed. The hardware watchdog logic has a 10-bit downcounter and a 4-bit prescaler. When enabled, the watchdog timer commences counting clock cycles as $\overline{AS}$ is asserted (for internal or external bus masters). The count is terminated normally by the negation of $\overline{AS}$; however, if the count reaches zero before $\overline{AS}$ is negated, $\overline{BERR}$ will be asserted until $\overline{AS}$ is negated. The hardware watchdog will operate with internal as well as external bus masters.

The hardware watchdog logic uses four bits in the SCR.

HWDEN—Hardware Watchdog Enable

0 = The hardware watchdog is disabled.
1 = The hardware watchdog is enabled.

After system reset, this bit defaults to one to enable the hardware watchdog.

HWDCN–HWDCN0—Hardware Watchdog Count 2–0

000 = $\overline{\text{BERR}}$ is asserted after 128 clock cycles (8 μs, 16-MHz clock)
001 = $\overline{\text{BERR}}$ is asserted after 256 clock cycles (16 μs, 16-MHz clock)
010 = $\overline{\text{BERR}}$ is asserted after 512 clock cycles (32 μs, 16-MHz clock)
011 = $\overline{\text{BERR}}$ is asserted after 1K clock cycles (64 μs, 16-MHz clock)
100 = $\overline{\text{BERR}}$ is asserted after 2K clock cycles (128 μs, 16-MHz clock)
101 = $\overline{\text{BERR}}$ is asserted after 4K clock cycles (256 μs, 16-MHz clock)
110 = $\overline{\text{BERR}}$ is asserted after 8K clock cycles (512 μs, 16-MHz clock)
111 = $\overline{\text{BERR}}$ is asserted after 16K clock cycles (1 ms, 16-MHz clock)

After system reset, these bits default to all ones; thus, $\overline{\text{BERR}}$ will be asserted after 1 ms for a 16-MHz system clock.

NOTE

Successive timeouts of the hardware watchdog may vary slightly in length. The counter resolution is 16 clock cycles.

3.8.7 Reducing Power Consumption

There are a number of ways to reduce power consumption on the IMP. They can be classified as general power-saving tips and low-power modes.

3.8.7.1 Power-Saving Tips

Without using any of the IMP low-power modes, power consumption may be reduced in the following ways.

1. The system clock frequency of the IMP may be reduced to the lower limit of its operating frequency range (e.g., 8 MHz) as specified in Section 6 Electrical Characteristics.
2. When not used, the SCCs should be disabled by clearing the ENT and ENR bits in the SCM registers.
3. If not needed, the SCC baud rate generators should be disabled or caused to clock at a low frequency. The baud rate generators are initialized to a very fast clock rate after reset, which can be reduced by programming the SCON register.
4. The two general-purpose timer prescalers should be set to the maximum divider value, and the timers should be disabled if not used.
5. Any unneeded peripheral output pins that are multiplexed with parallel I/O pins should be left configured as parallel I/O pins. The smaller the number of output transistors switching, the less current used.

3.8.7.2 Low-Power (Standby) Modes

The IMP also supports several types of low-power modes. The low-power modes on the IMP are used when no processing is required from the M68000 core and when it is desirable to reduce system power consumption to its minimum value. All low-power modes are entered by first setting the low-power enable (LPEN) bit, and then executing the M68000 STOP instruction.

At the end of the STOP instruction, a major change to the IMP occurs. The M68000 core immediately goes into a standby state in which it executes no instructions. In this state, the clock internally sent to the M68000 core is internally divided, saving power. The amount of this divide ratio is configured by the user. At the same time, however, the rest of the IMP continues to operate at the normal system frequency (i.e., the frequency on the EXTAL pin). All peripherals continue to operate normally during this time. Also, during low-power modes, all IMP external signals continue to function normally and at full speed.

In all modes, any of the 16 possible internal interrupt request (INRQ) sources can cause the IMP to leave a low-power mode. Masked interrupt sources will never cause a lower power mode to be exited. If it is desired to have an external signal cause the IMP to exit a low-power mode, it must be routed to one of the PB11–PB8 port pins with interrupt capability, so that an INRQ interrupt can be generated.

There are three low-power modes: low power, lowest power, and lowest power with external clock. Low-power mode allows execution to resume immediately upon recognition of the interrupt, with no loss of any IMP state information. Lowest power mode requires execution to resume with an internal M68000 reset. The M68000 state is lost, but the rest of the IMP peripheral states are completely retained. The lowest power mode with external clock offers the absolute minimum power consumption with the IMP, but requires external hardware.

3.8.7.2.1 Low-Power Mode

This mode is possible if the user-selected low-power frequency applied to the M68000 core remains above the operating limits specified in Section 6 Electrical Characteristics (e.g., 8 MHz). In this mode, LPREC is set to zero. Once an INRQ interrupt becomes pending, the system control block switches the M68000 core back to full frequency and power, and begins handling the interrupt in the usual manner. No M68000 core or peripheral status is lost in this mode. Note: The CLKO signal will remain at the same frequency that is on the EXTAL input.

The following list gives a step-by-step example of how to use the low-power mode. For this example, an interrupt from either timer 1 or timer 2 causes the IMP to exit the low-power mode. This example also assumes an initial operating frequency of 16.67 MHz for the IMP.

1. Set the lower byte of the SCR (location $F7) to $20. This sets the LPEN bit and sets the clock divider to a value of 2 (divide by 2).
2. Disable all interrupts except TIMER1 and TIMER2 in the IMR.
3. Turn off any unneeded peripherals, such as the SCCs, by clearing the ENR and ENT bits. Also turn off any unneeded baud rate generators by setting the EXTC bits in the SCON registers. This procedure can save as much as 4 mA per SCC at 16.67 MHz. (EXTC is cleared by default on power-on reset.)
4. Execute the STOP instruction. The low-power mode is now entered.
5. When a timer 1 or 2 interrupt occurs, the M68000 resumes execution with the timer 1 or 2 interrupt handler. After the RTE instruction, execution continues with the instruction following the STOP instruction in step 4 above. All IMP state information is retained.

3.8.7.2.2 Lowest Power Mode

In this mode, the processor frequency can be further reduced beyond the minimum system frequency limit (e.g., lower than the limit of 8 MHz). In this mode, the LPREC bit must be set to one by the user, causing the M68000 core to be reset as the lowest power mode is exited. The M68000 core is given an internal reset sequence for 16 to 32 clock cycles, and execution resumes with the fetching of the reset vector. The RESET pin does not externally assert during the internal reset sequence. The entire M68000 core status is lost in this mode (A0–A7, D0–D7, PC, SR, etc.); however, the IMP peripheral status is retained (this includes the dual-port RAM, internal registers, BAR, etc.).

The following list gives a step-by-step example of how to use the lowest power mode. For this example, an external wakeup signal is issued to the PB11 pin to exit the lowest power mode.

1. Set the lower byte of the SCR (location $F7) to $FF. This sets the LPREC bit, the LPEN bit, and sets the clock divider to its maximum value (divide by 1024).
2. Disable all interrupts except PB11 in the IMR.
3. Turn off any unneeded peripherals, such as the SCCs, by clearing the ENR and ENT bits. Also turn off any unneeded baud rate generators by setting the EXTC bits in the SCON registers. This procedure can save as much as 4 mA per SCC at 16.67 MHz. (EXTC is cleared by default after reset.)
4. Execute the STOP instruction. Lowest power mode is now entered.
5. A wakeup signal comes from the system to the PB11 pin.
6. The IMP then generates the PB11 interrupt and a reset is automatically generated to the M68000 core.
7. After the IMP is reset, software processing continues from the exception vector table reset vector address. The M68000 is reset, but the rest of the IMP retains its state.

3.8.7.2.3 Lowest Power Mode with External Clock

After the IMP is safely in the lowest power mode the EXTAL frequency can be externally reduced to a lower frequency. In this mode, the clock dividing should not be done in the LPCD bits, but rather externally on the EXTAL pin.

NOTE

The input to EXTAL must be greater than or equal to 25kHz.

The major difference in this mode, is that the entire IMP is now running at a lower clock rate. Any IMP on-chip peripherals and any bus cycles executed by one of the on-chip masters are thus slowed down.

NOTE

The use of external clocks with the SCCs allows the original serial rates to be maintained; however, before attempting this, the SCC performance data should be carefully reviewed (see Ap-

pendix A SCC Performance). Also, the minimum 1:2.5 serial to CLKO clock ratio must be maintained at all times.

The following list gives a step-by-step example of how to achieve the lowest possible power using an external clock. For this example, an external wakeup signal is issued to the PB11 pin to exit the lowest power mode.

1. Set the lower byte of the SCR (location $F7) to $A0. This sets the LPREC bit and the LPEN bits only.
2. Disable all interrupts except PB11 in the IMR.
3. Turn off any unneeded peripherals, such as the SCCs, by clearing the ENR and ENT bits. Also, turn off any unneeded baud rate generators by setting the EXTC bits in the SCON registers. This procedure can save as much as 4 mA per SCC at 16.67 MHz. (EXTC is cleared by default on after reset.)
4. Start off a timer now to toggle a $\overline{\text{TOUT}}$ pin in approximately 20 clocks. Do not wait for this to occur, but continue on to the next step.
5. Execute the STOP instruction. The IMP is now safely in the lowest power mode.
6. Use the toggled $\overline{\text{TOUT}}$ pin to switch the EXTAL clock rate to approximately 50 kHz. Ensure no glitches occur on the EXTAL signal which exceed the maximum clock frequency.
7. Power consumption is now the lowest.
8. A wakeup signal comes from the system.
9. The wakeup signal switches the clock frequency back to the 8–16.67-MHz range and pulls the PB11 pin low. These two events can happen simultaneously.
10. The IMP generates the PB11 interrupt, and a M68000 core reset is generated.
11. After the IMP is reset, software processing continues from the exception vector table reset vector address. The M68000 is reset, but the rest of the IMP retains its state.

The low-power logic uses eight bits in the SCR.

LPCD4–LPCD0—Low-power Clock Divider Selects

The low-power clock divider select bits (LPCD4—LPCD0) specify the divide ratio of the low-power clock divider equal to LPCD4—LPCD0 + 1. The system clock is divided by 2, then divided by the clock divider value (1 to 32). Thus, a divide ratio of 2 to 64 (LPCD4—LPCD0 0 to 31) can be selected. After a system reset, these bits default to zero.

LPEN—Low-power Enable

0 = The low-power modes are disabled.
1 = The low-power modes are enabled.

After a system reset, this bit defaults to zero to disable the low-power modes.

LPP16—Low-power Clock Prescale Divide by 16

0 = The low-power clock divider input clock is the main clock.
1 = The low-power clock divider input clock is the main clock divided by 16. Thus, a divide ratio of 32 to 1024 (LPCD4—LPCD0 0 to 31) can be selected.

After a system reset, this bit defaults to zero.

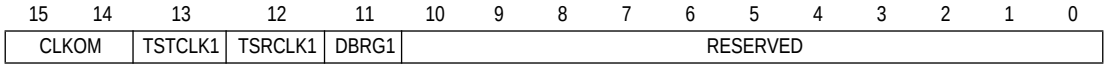
LPREC—Low-Power Recovery

0 = Nondestructive recovery from low power. The processor returns to full frequency and then proceeds using currently held status value. This is called low-power mode.
1 = Destructive recovery from low power. The processor returns to full frequency and then drives RESET for 16 clock cycles. This is called lowest power mode.

After a system reset, the bit defaults to zero.

3.9 CLOCK CONTROL REGISTER

The CKCR is a 16-bit register is a memory-mapped read-write register. The address of this register is fixed at $0FA in supervisor data space (FC = 5). This register controls the state of CLKO, RCLK1, TCLK1, and BRG1. This register is cleared at reset.



CLKOM—CLKO Mode

These bits may be written at any time to change the mode of the CLKO pin. Changes to CLKO are made while the CLKO signal is high. No spikes on CLKO will occur when the CLKOM bits are changed.

00 = The CLKO pin functions normally
01 = The CLKO pin output driver is two thirds its normal strength. Specification 5a at 16.67 MHz is 2 to 14 ns and at 20 MHz is 2 to 11 ns. The output drive derating factor for CLKO in this mode is not specified.
01 = The CLKO pin output driver is one third its normal strength. Specification 5a at 16.67 MHz is 2 to 20ns and at 20 MHz is 2 to 16ns. The output drive derating factor for CLKO in this mode is not specified.
11 = The CLKO pin output is disabled, but is driven high by an internal pullup. Significant power savings can be obtained by disabling CLKO. Typical power savings may range between 2 and 6 mA, depending on the CLKO loading. Disabling CLKO can also reduce noise and electromagnetic interference on the printed circuit board.

TSTCLK1—Three-state TCLK1

0 = Normal operation
1 = The TCLK1 pin is three-stated. This option may be used to prevent contention on the TCLK1 pin if an external clock is provided to the TCLK1 pin while the SCC1 baud rate generator is output on TCLK1. This option may also be chosen if it is required to run the SCC1 baud rate generator at high speed (for instance in a high speed UART application), but the TCLK1 output is not needed, and it is desired to

reduce power. An external pullup should be used if TCLK1 is not driven externally. TSTCLK1 may be toggled at any time, but the SCC1 transmitter should be disabled and re-enabled if any dynamic change is made on TSTCLK1 during the operation of the SCC1 transmitter.

TSRCLK1—Three-state RCLK1

0 = Normal operation

1 = The RCLK1 pin is three-stated. This option may be used to prevent contention on the RCLK1 pin if an external clock is provided to the RCLK1 pin while the SCC1 baud rate generator is output on RCLK1. This option may also be chosen if it is required to run the SCC1 baud rate generator at high speed (for instance in a high speed UART application), but the RCLK1 output is not needed, and it is desired to reduce power. An external pullup should be used if RCLK1 is not driven externally. TSRCLK1 may be toggled at any time, but the SCC1 receiver should be disabled and re-enabled if any dynamic change is made on TSRCLK1 during the operation of the SCC1 receiver.

DBRG1—Disable BRG1

0 = Normal operation

1= The BRG1 pin is disabled and is driven high. This option should be chosen if it is required to run the SCC1 baud rate generator at high speed, but the BRG1 output is not needed and it is desired to reduce power. Although DBRG1 may be modified at any time, the user should note that glitches on BRG1 are not prevented by the MC68302 when the state of DBRG1 is changed.

Bits 10 - 0—Reserved. Should be written with zeros.

3.9.1 Freeze Control

Used to freeze the activity of selected peripherals, $\overline{\text{FRZ}}$ is useful for system debugging purposes. When $\overline{\text{FRZ}}$ is asserted:

- The CP main controller freezes its activity on the next clock (CLKO) and will continue in a frozen state as long as $\overline{\text{FRZ}}$ remains asserted. No new interrupt requests and no memory accesses (internal or external) will occur, and the main controller will not access the serial channels.
- The IDMA completes any bus cycle that is in progress (after $\overline{\text{DTACK}}$ is asserted) and releases bus ownership. No further bus cycles will be started as long as $\overline{\text{FRZ}}$ remains asserted.
- Each timer can be programmed to freeze by setting the appropriate bit in the SCR. After a one-clock (CLKO) delay, the selected timers will freeze their activity (count, capture) as long as $\overline{\text{FRZ}}$ remains asserted.

NOTE

Regardless of whether or not the freeze logic is used, FRZ must be negated during system reset.

FRZ1—Freeze Timer 1 Enable

0 = Freeze timer 1 logic is disabled.
1 = Freeze timer 1 logic is enabled.

After system reset, this bit defaults to zero.

FRZ2—Freeze Timer 2 Enable

0 = Freeze timer 2 logic is disabled.
1 = Freeze timer 2 logic is enabled.

After system reset, this bit defaults to zero.

FRZW—Freeze Watchdog Timer Enable

0 = Freeze watchdog timer logic is disabled.
1 = Freeze watchdog timer logic is enabled.

After system reset, this bit defaults to zero.

No other MC68302 peripherals are directly affected by the freeze logic; however, consequential errors such as receiver overruns in the SCC FIFOs may occur due to the CP main controller being disabled. Note that use of the freeze logic does not clear any IPR bits that were already set.

3.10 DYNAMIC RAM REFRESH CONTROLLER

The communications processor (CP) main (RISC) controller may be configured to handle the dynamic RAM (DRAM) refresh task without any intervention from the M68000 core. Use of this feature requires a timer or SCC baud rate generator (either from the MC68302 or externally), the I/O pin PB8, and two transmit buffer descriptors from SCC2 (Tx BD6 and Tx BD7).

The DRAM refresh controller routine executes in 25 clock cycles. Assuming a refresh cycle every 15.625 μs, two wait state DRAMs, and a 16.67-MHz EXTAL frequency, this routine uses about 10 percent of the microcontroller bandwidth and 4 percent of the M68000 bus bandwidth. The refresh cycle will not be executed during a period that a bus exception (i.e., $\overline{\text{RESET}}$, $\overline{\text{HALT}}$, or $\overline{\text{BERR}}$) is active. The refresh cycle is a standard M68000-type read cycle (an SDMA byte read cycle). It does not generate row address strobe (RAS) and column address strobe (CAS) to the external DRAM. These functions require an external PAL. Use of the DRAM refresh controller will slightly reduce the maximum possible serial data rates of the SCCs.

3.10.1 Hardware Setup

An output of timer 1 or timer 2 (the $\overline{\text{TOUT}}$ pin) or one of the SCC's baud rate generator outputs (BRG3–BRG1) should be connected externally to PB8. A high-to-low transition on this edge causes a request to be generated to the main controller to perform one refresh cycle. The DRAM refresh request takes priority over all SCC channels and commands given to the CP command register.

A block diagram of an MC68302 DRAM system is shown in Figure 3-13. The MC68302 generates standard M68000 read and write cycles that must be converted to DRAM read and write cycles. The address buffers provide the multiplexing of the row and column addresses

to the DRAM bank. The PAL generates the RAS and CAS lines for the DRAM chips and controls the address multiplexing in the external address buffers. One of the MC68000 chip-select lines can be used as the DRAM bank enable signal, if desired.

The refresh operation is a byte read operation. Thus, $\overline{\text{UDS}}$ or $\overline{\text{LDS}}$ will be asserted from the MC68302, but not both. A refresh to an odd address will assert $\overline{\text{LDS}}$; whereas, a refresh to an even address will assert $\overline{\text{UDS}}$.

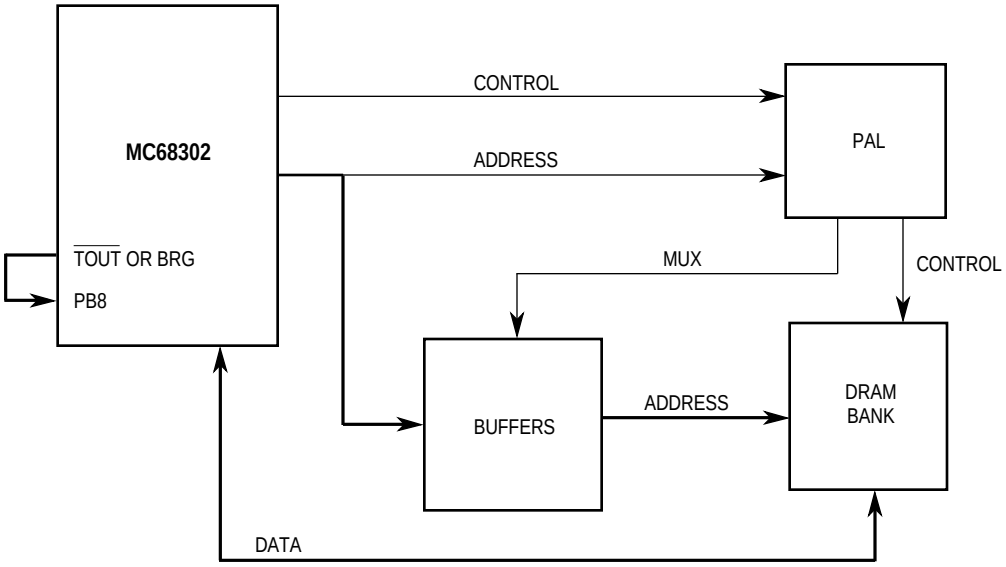


Figure 3-13. DRAM Control Block Diagram

3.10.2 DRAM Refresh Controller Bus Timing

The DRAM refresh controller bus cycles are actually SDMA byte read accesses (see 4.2 SDMA Channels for more details). All timings, signals, and arbitration characteristics of SDMA accesses apply to the DRAM refresh controller accesses. For example, DRAM refresh cycles activate the $\overline{\text{BCLR}}$ signal, just like the SDMA. Note that the function code bits may be used to distinguish DRAM refresh cycles from SDMA cycles, if desired.

A bus error on a DRAM refresh controller access causes the $\overline{\text{BERR}}$ channel number at offset BASE + $67C to be written with a $0001. This is also the value written if the SCC1 receive SDMA channel experiences a bus error; thus, these two sources cannot be distinguished upon a bus error. The DRAM refresh SDMA channel and SCC1 receive SDMA channel are separate and independent in all other respects.

3.10.3 Refresh Request Calculations

A typical 1-Mbyte DRAM needs one refresh cycle every 15.625 µs. The DRAM refresh controller is configured to execute one refresh cycle per request; thus, the PB8 pin should see a high-to-low transition every 15.625 µs. This is once every 260 cycles for a 16.67-MHz clock. Note that one refresh per request minimizes the speed loss on the SCC channels.

3.10.4 Initialization

The user should first initialize the refresh routine parameters in the SCC2 parameter RAM. These parameters are the DRAM low starting address, the DRAM high starting address, the DRAM address increment step (number of bytes in a row), the count (number of rows), and a temporary count. Then, mask the PB8 bit in the IMR (unless an interrupt is desired on each refresh request). Next, the timer or baud rate generator should be programmed to provide the desired refresh clock to the PB8 pin. Next, the ERRE bit in the SCR should be set. Then, upon every high-to-low transition of PB8, the refresh routine executes one refresh (read) cycle.

ERRE—External RISC Request Enable

0 = Normal operation.
1 = When this bit is set, a high-to-low transition on PB8 causes the CP to execute the DRAM refresh routine.

3.10.5 DRAM Refresh Memory Map

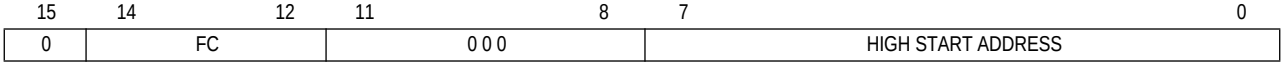
The DRAM refresh memory map replaces the SCC2 TxBD6 and TxBD7 structures in the parameter RAM. The wrap bit must therefore be set in SCC2 TxBD5 so that only six TxBDs are used for SCC2. These parameters should be written before the DRAM refresh controller receives its first request, but may be read at any time. They are undefined at reset.

Table 3-11. DRAM Refresh Memory Map Table

Address	Name	Width	Description
Base + 570 #	DRAM-High	Word	Dynamic RAM High Address and FC
Base + 572 #	DRAM-Low	Word	Dynamic RAM Low Address
Base + 574 #	INCREMENT	Word	Increment Step (number of bytes/row)
Base + 576 #	COUNT	Word	RAM Refresh Cycle Count (number of rows)
Base + 578	T-ptr-H	Word	Temporary Refresh High Address and FC
Base + 57A	T-ptr-L	Word	Temporary Refresh Low Address
Base + 57C #	T-count	Word	Temporary Refresh Cycles Count
Base + 57E	RESERVED	Word	Reserved

Initialized by the user (M68000 core).

DRAM_High—Dynamic RAM High Address and Function Codes



This 16-bit parameter contains the dynamic RAM address space function code output during the refresh cycle and the high eight bits of the dynamic RAM starting address. This parameter should be initialized by the user before activating the refresh routine.

NOTE

The FC bits should not be programmed to the value “111.”

DRAM_Low—Dynamic RAM Low Address

This 16-bit parameter contains the lower 16 bits of the dynamic RAM starting address. This parameter should be initialized by the user before activating the refresh routine.

INCREMENT—Increment Step

This 16-bit parameter contains the number of bytes in a row. The refresh routine will increment its pointer with this parameter value every refresh cycle. This parameter should be initialized by the user before activating the refresh routine.

COUNT—RAM Refresh Cycle Count

This 16-bit parameter contains the number of rows in the DRAM. The refresh routine will execute the COUNT number of refresh cycles before wrapping back to the RAM base address. This parameter should be initialized by the user before activating the refresh routine.

T_ptr_H and T_ptr_L—Temporary Pointer High and Low

These two 16--bit parameters contain the next refresh cycle address and function code to be used by the CP. They correspond to DRAM_High and DRAM_Low, respectively.

T_count—Temporary Count

This 16-bit parameter contains the number of refresh cycles that the DRAM refresh controller must still perform before it will wrap to the beginning of the DRAM. This parameter should be initialized to zero by the user before activating the refresh routine.

3.10.6 Programming Example

An example of programming the DRAM parameters is given for the Motorola MC514256 DRAM. This 1M-bit DRAM is organized in a 256K $\times$ 4 arrangement. There are 512 rows and 512 columns on this device. A bank of 4 of these DRAMs is assumed in this example, giving 512K-bytes of memory. If this bank is placed at location $000000 in the MC68302 address space, its range would then be $0 to $7FFFFF. Assuming a RAS-only refresh technique is used, acceptable parameters would be as follows:

DRAM_High = $0000
DRAM_Low = $0100
INCREMENT = $0002
COUNT = $0200
T_Count = $0000

The value of $0000 in DRAM_High results in the refresh access using a function code of 000. Usually, the function code is not required by the DRAM control logic but may assist in the identification of DRAM refresh accesses during debugging. The starting address is picked to be $000100 (instead of $000000) to avoid refreshing the BAR and SCR registers on the IMP. Depending on the PAL design, an increment value of $0002 can actually refresh a word at a time, even though the refresh access from the MC68302 is a byte read. The COUNT value is the number of word refreshes required in the entire DRAM bank.

SECTION 4
COMMUNICATIONS PROCESSOR (CP)

The CP includes the following modules:

- Main Controller (RISC Processor)
- Six Serial Direct Memory Access (SDMA) Channels
- A Command Set Register
- Serial Channels Physical Interface Including:

  —Motorola Interchip Digital Link (IDL)
  —General Circuit Interface (GCI), also known as IOM-2
  —Pulse Code Modulation (PCM) Highway Interface
  —Nonmultiplexed Serial Interface (NMSI) Implementing Standard
  —Modem Signals
- Three Independent Full Duplex Serial Communication Controllers (SCCs) Supporting the Following Protocols:

  —High-Level/Synchronous Data Link Control (HDLC/SDLC)
  —Universal Asynchronous Receiver Transmitter (UART)
  —Binary Synchronous Communication (BISYNC)
  —Synchronous/Asynchronous Digital Data Communications Message
  —Protocol (DDCMP)
  —Transparent Modes
  —V.110 Rate Adaption
- Serial Communication Port (SCP) for Synchronous Communication
- Two Serial Management Controllers (SMCs) to Support the IDL and GCI Management Channels

4.1 MAIN CONTROLLER

The CP main controller is a RISC processor that services the three SCCs, the SCP, and the SMCs. Its primary responsibilities are to work with the serial channels to implement the user-chosen protocol and to manage the SDMA channels that transfer data between the SCCs and memory. The CP main controller also executes commands issued by the M68000 core (or an external processor) and generates interrupts to the interrupt controller.

The operation of the main controller is transparent to the user, executing microcode located in a private internal ROM (see Figure 4-1). Commands may be explicitly written to the main controller by the M68000 core through the CP command register. Additionally, commands and status are exchanged between the main controller and the M68000 core through the

buffer descriptors of the serial channels. Also, a number of protocol-specific parameters are exchanged through several parameter RAM areas in the internal dual-port RAM.

The RISC controller uses the peripheral bus to communicate with all its peripherals. Each SCC has a separate transmit and receive FIFO. Depending on the protocol chosen, the transmit FIFO is either 3 bytes or 4 words, and the receive FIFO is either 3 bytes or 3 words. Each SCC is configured by parameters written to the dual-port RAM and by SCC hardware registers that are written by the M68000 core (or an external master). The SCC registers that configure each SCC are the SCON, DSR, and SCM. There are three sets of these registers, one for each SCC. The serial channels physical interface is configured by the M68000 core through the SIMODE and SIMASK registers.

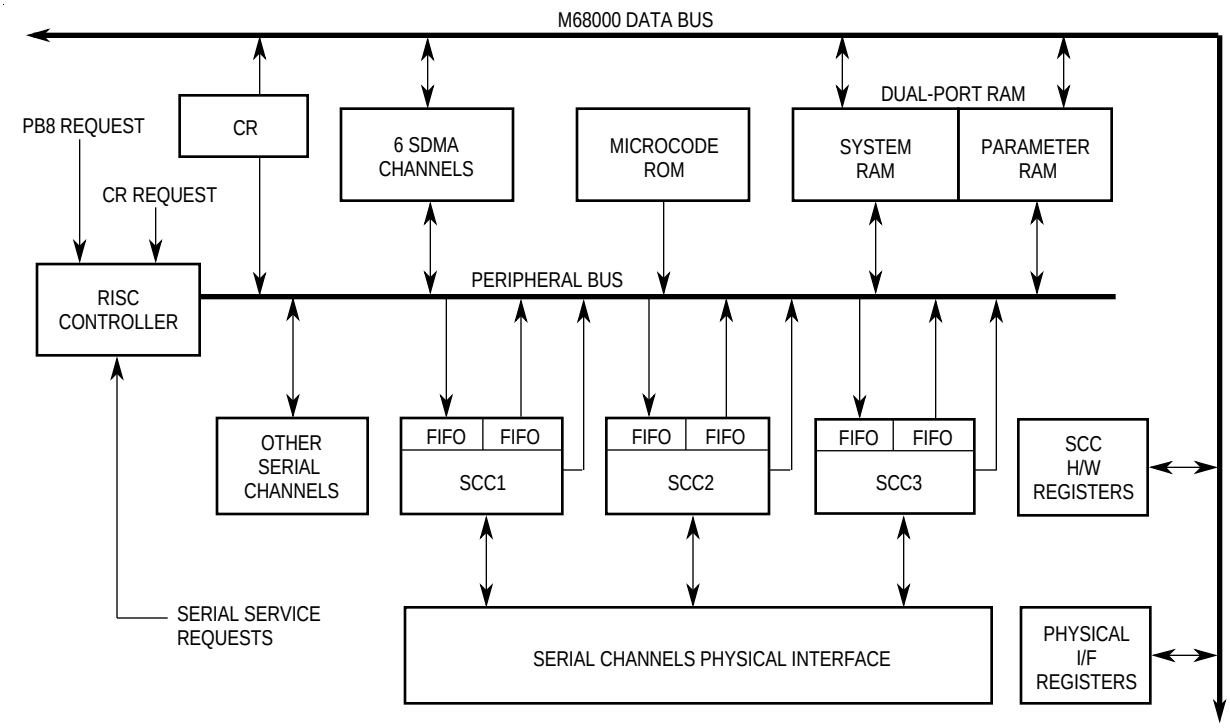


Figure 4-1. Simplified CP Architecture

Simultaneous access of the dual-port RAM by the main controller and the M68000 core (or external processor) is prevented. During a standard four-clock cycle access of the dual-port RAM by the M68000 core, three main controller accesses are permitted. The main controller is delayed one clock cycle, at most, in accessing the dual-port RAM.

The main controller has a priority scheduler that determines which microcode routine is called when more than one internal request is pending. Requests are serviced in the following priority:

1. CP or System Reset
2. SDMA Bus Error

3. DRAM Refresh Controller
4. Commands Issued to the Command Register
5. SCC1 Receive Channel
6. SCC1 Transmit Channel
7. SCC2 Receive Channel
8. SCC2 Transmit Channel
9. SCC3 Receive Channel
10. SCC3 Transmit Channel
11. SMC1 Receive Channel
12. SMC1 Transmit Channel
13. SMC2 Receive Channel
14. SMC2 Transmit Channel
15. SCP Receive Channel
16. SCP Transmit Channel

For details on the DRAM refresh controller, see 3.10 Dynamic Ram Refresh Controller.

4.2 SDMA CHANNELS

Six serial (SDMA) channels are associated with the three full-duplex SCCs. Each channel is permanently assigned to service the receive or transmit operation of one of the SCCs and is always available, regardless of the SCC protocol chosen.

The SDMA channels allow flexibility in managing the data flow. The user can, on a buffer-by-buffer basis, determine whether data should be transferred between the SCCs and external memory or between the SCCs and on-chip dual-port RAM. This choice is controlled in each SCC buffer descriptor. The SCC to external memory path bypasses the dual-port RAM by allowing the SDMA channel to arbitrate for the M68000 bus directly. The SCC to dual-port RAM path saves external memory and eliminates the need to arbitrate for the bus.

Figure 4-2 shows the paths of the data flow. Data from the SCCs may be routed directly to external RAM as shown in path 1. In path 2, data is sent over the peripheral bus to the internal dual-port RAM. The SMCs and SCP, shown in path 3, always route their data to the dual-port RAM since they only receive and transmit a byte at a time.

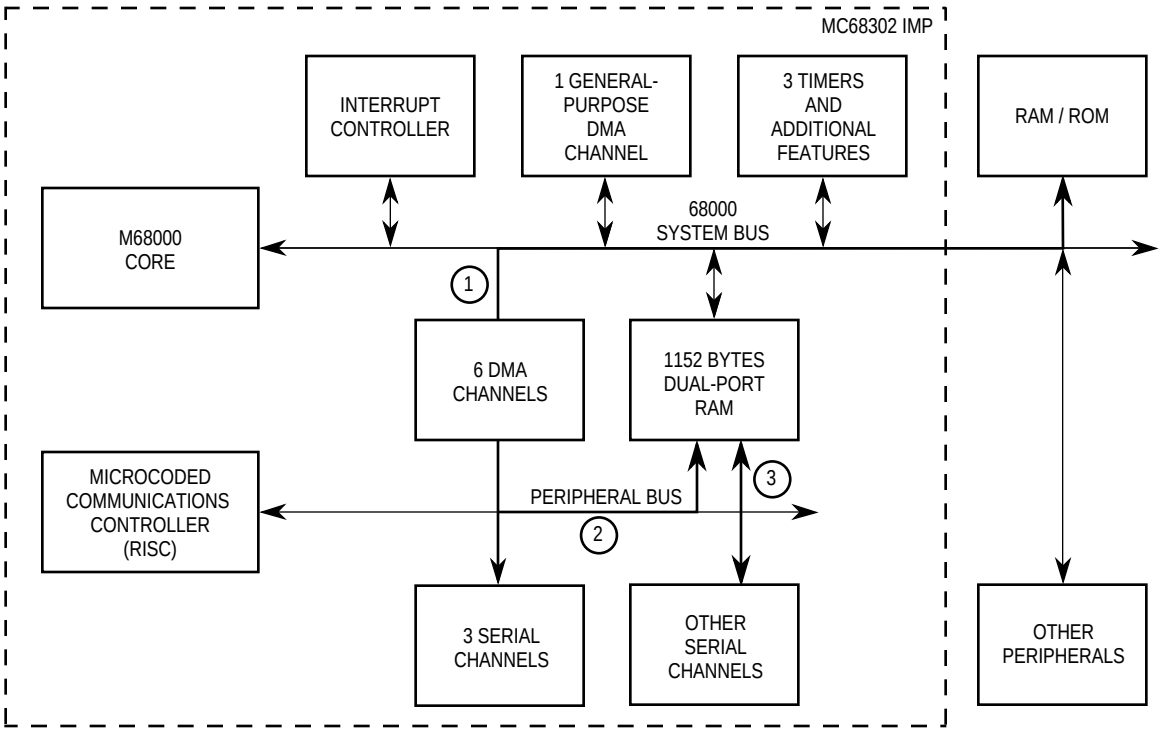


Figure 4-2. Three Serial Data Flow Paths

The SDMA channels implement bus-cycle-stealing data transfers controlled by microcode in the CP main controller. Having no user-accessible registers associated with them, the channels are effectively controlled by the choice of SCC configuration options.

When one SDMA channel needs to transfer data to or from external memory, it will request the M68000 bus with the internal signal SDBR, wait for SDBG, and then only assert the external signal $\overline{\text{BGACK}}$ (see 3.8.5 Bus Arbitration Logic). It remains the bus master for only one bus cycle. The six SDMA channels have priority over the IDMA controller. If the IDMA is bus master when an SDMA channel needs to transfer over the M68000 bus, the SDMA will steal a cycle from the IDMA with no arbitration overhead while $\overline{\text{BGACK}}$ remains continuously low and $\overline{\text{BCLR}}$ remains high. Each SDMA channel may be programmed with a separate function code, if desired. The SDMA channel will read 16 bits at a time. It will write 8 bits at a time except during the HDLC or transparent protocols where it writes 16 bits at a time. Each bus cycle is a standard M68000-type bus cycle. The chip select and wait state generation logic on the MC68302 may be used with the SDMA channels.

NOTE

When external buffer memory is used, the M68000 bus arbitration delay must be less than what would cause the SCC internal FIFOs to overrun or underrun. This aspect is discussed in more detail in 4.5 Serial Communication Controllers (SCCs) and in Appendix A SCC Performance.

The SDMA will assert the external $\overline{\text{BCLR}}$ pin when it requests the bus. $\overline{\text{BCLR}}$ can be used to clear an external bus master from the external bus, if desired. For instance, $\overline{\text{BCLR}}$ can be connected through logic to the external master's $\overline{\text{HALT}}$ signal, and then be negated externally when the external master's AS signal is negated. $\overline{\text{BCLR}}$ as seen from the MC68302 is negated by the SDMA during its access to memory.

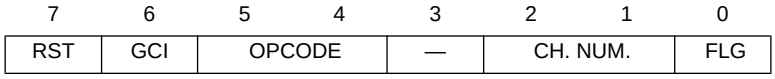
The SDMA keeps the M68000 bus for only one operand (8 or 16 bit) transfer before giving it up. If the SDMA begins a word operation on an 8-bit bus, it will complete that operation before giving up the bus, unless a bus exception such as reset, halt, retry, or bus error occurs. Reset suspends and resets all SDMA activity. Halt suspends activity after the current bus cycle. For information on a bus error during an SDMA access, see 4.5.8.4 Bus Error on SDMA Access.

SDMA operation occurs regardless of the value of the BCLM bit in the SCR, and thus is not affected by the low interrupt latency mechanism.

4.3 COMMAND SET

The M68000 core processor (or an external processor) issues commands to the CP by writing to the CP command register (CR). Only one CR exists on the MC68302. The M68000 core should set the least significant bit (FLG) of the command register when it issues commands. The CP clears FLG after completing the command to indicate to the M68000 core that it is ready for the next command. Subsequent commands to the CR may be given only after FLG is cleared. The software reset (issued with the RST bit) command may be given regardless of the state of FLG, but the M68000 core should still set FLG when setting RST.

The CR, an 8-bit, memory-mapped, read-write register, is cleared by reset.



RST—Software Reset Command

This bit is set by the M68000 core and cleared by the CP. This command is useful when the M68000 core wants to reset the registers and parameters for all channels (SCCs, SCP, SMCs). The main controller in the CP detects this command by hardware, clears the FLG bit within two clocks, and resets the entire CP in approximately 60 clocks. User initialization of the CP registers may begin as soon as the FLG bit is cleared. The CP reset resets the SCCs to the state following a hardware reset, but it does not affect the serial interface (the port A and B registers, the configuration of the SIMODE and SIMASK registers, and the SCON registers). Note that this operation does not clear IPR bits in the interrupt controller.

GCI-OPCODE—GCI Commands and Command Opcodes

0 = When the GCI bit is zero, the commands are as follows:

OPCODE—Command Opcode

These bits are set by the M68000 core to define the specific SCC command. The precise meaning of each command below depends on the protocol chosen.

00 = STOP TRANSMIT Command
01 = RESTART TRANSMIT Command
10 = ENTER HUNT MODE Command
11 = RESET RECEIVER BCS CALCULATION (used only in BISYNC mode)

The detailed command description for the UART protocol is presented in 4.5.11 UART Controller.

The detailed command description for the HDLC protocol is presented in 4.5.12 HDLC Controller.

The detailed command description for the BISYNC protocol is presented in 4.5.13 BI-SYNC Controller.

The detailed command description for the DDCMP protocol is presented in 4.5.14 DDC-MP Controller.

The detailed command description for the V.110 protocol is presented in 4.5.15 V.110 Controller.

The detailed command description for the transparent protocol is presented in 4.5.16 Transparent Controller.

1 = When GCI is set in conjunction with the opcode bits, the two GCI commands (ABORT REQUEST and TIMEOUT) are generated. The accompanying CH. NUM. should be 10, and FLG should be set.

OPCODE—Command Opcode (GCI Mode Only)

These bits are set by the M68000 core to define the specific GCI command. See 4.7 Serial Management Controllers (SMCs) for more details.

00 = TRANSMIT ABORT REQUEST; the GCI receiver sends an abort request on the E bit.
01 = TIMEOUT Command
10 = Reserved
11 = Reserved

Bit 3—Reserved bit; should be set to zero.

CH. NUM.—Channel Number

These bits are set by the M68000 core to define the specific SCC channel that the command is to operate upon.

00 = SCC1
01 = SCC2
10 = SCC3
11 = Reserved

FLG—Command Semaphore Flag

The bit is set by the M68000 core and cleared by the CP.

0 = The CP is ready to receive a new command.

1 = The CR contains a command that the CP is currently processing. The CP clears this bit at the end of command execution. Note that the execution of the STOP TRANSMIT or RESTART TRANSMIT commands may not affect the TXD pin until many clocks after the FLG bit is cleared by the CP due to the transmit FIFO latency.

4.3.1 Command Execution Latency

Commands are executed at a priority higher than the SCCs, but less than the priority of the DRAM refresh controller. The longest command, the ENTER HUNT MODE command, executes in 41 clocks. All other commands execute in less than 20 clocks. The maximum command latency is calculated as follows:

- Command execution time (41 or 20) +
- 25 clocks if DRAM refresh controller is used +

—205 clocks if any SCC is enabled with BISYNC; or
—165 clocks if any SCC is enabled with Transparent; or
—165 clocks if any SCC is enabled with HDLC; or
—150 clocks if any SCC is enabled with UART; or
—140 clocks if any SCC is enabled with DDCMP; else
—0

For example, if HDLC and UART modes are used on the SCCs with the DRAM refresh controller operating, the maximum command latency is 41 + 25 + 165 = 231 clocks = 13 μs at 16.67 MHz. The equations assume that the DRAM refresh cycle occurs once during the command latency. Note that commands are typically given only in special error-handling situations and that the typical latency is much less than the worst case value.

4.4 SERIAL CHANNELS PHYSICAL INTERFACE

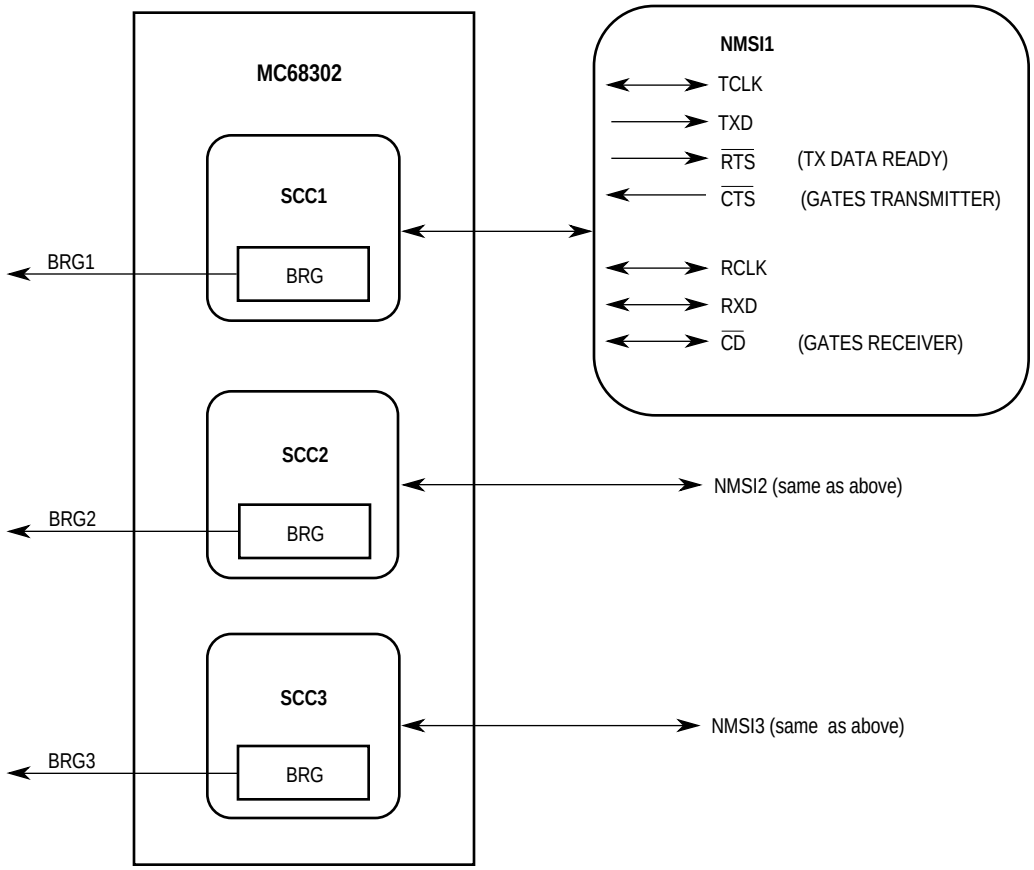
The serial channels physical interface joins the physical layer serial lines to the three SCCs and the two SMCs. (The separate three-wire SCP interface is described in 4.6 Serial Communication Port (SCP).)

The MC68302 supports four different external physical interfaces from the SCCs:

1. NMSI—Nonmultiplexed Serial Interface
2. PCM—Pulse Code Modulation Highway
3. IDL—Interchip Digital Link
4. GCI—General Circuit Interface

The most generic physical interface on the MC68302 is the nonmultiplexed serial interface (NMSI). The NMSI consists of seven of the basic modem (or RS-232) signals: TXD, TCLK, RXD, RCLK, $\overline{\text{RTS}}$, $\overline{\text{CTS}}$, and $\overline{\text{CD}}$. Each SCC can have its own set of NMSI signals as shown in Figure 4-3. In addition to the NMSI, the baud rate generator clocks may be output on separate BRG pins as shown. All NMSI2 pins are multiplexed with parallel I/O pins. The user may choose which NMSI2 pins are used by the SCC2 and which are used as parallel I/O. On NMSI3, the TXD, TCLK, RXD, and RCLK pins are multiplexed with parallel I/O lines;

$\overline{RTS}$, $\overline{CTS}$, and $\overline{CD}$ are multiplexed with the SCP. See 4.6 Serial Communication Port (SCP) for more details on the SCP.



NMSI — Nonmultiplexed serial interface (also called the modem I/F).

Figure 4-3. NMSI Physical Interface

The other three physical interfaces, PCM, IDL, and GCI here are called multiplexed interfaces since they allow data from one, two, or all three SCCs to be time multiplexed together on the same pins. Note that if a multiplexed mode is chosen, the first SCC to use that mode must be SCC1 since the three multiplexed modes share pins with SCC1. After choosing a multiplexed mode, the user may decide whether SCC2 and SCC3 should be part of the multiplexed interface or whether they should have their own set of NMSI pins (see Figure 4-4). If SCC2 or SCC3 is part of the multiplexed interface, all NMSI2 and NMSI3 pins may be used for other functions such as parallel I/O. If a multiplexed mode is chosen, the baud rate generator clock is output on the BRG or TCLK pin, depending on whether the NMSI mode or multiplexed mode, respectively, was chosen for that SCC.

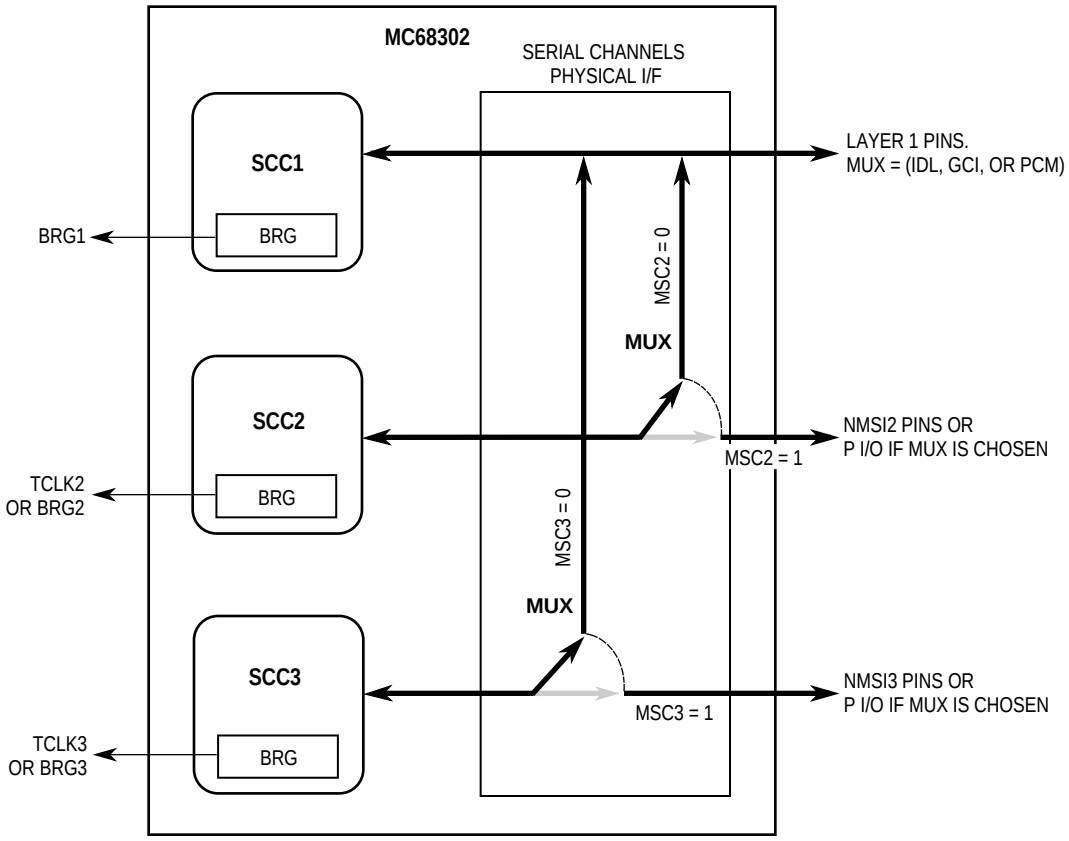


Figure 4-4. Multiplexed Mode on SCC1 Opens Additional Configuration Possibilities

There are five serial channel physical interface combinations for the three SCCs (see Table 4-1).

Table 4-1. The Five Possible SCC Combinations

SCC	1	2	3	4	5
SCC1	NMSI	MUX	MUX	MUX	MUX
SCC2	NMSI	NMSI	MUX	NMSI	MUX
SCC3	NMSI	NMSI	NMSI	MUX	MUX

NOTE: *MUX* is defined as one of the following: IDL, GCI, or PCM highway.

The PCM highway interface is a flexible time-division multiplexed interface. It allows the MC68302 to connect to popular time-slot interfaces such as T1 and CEPT as well as user-defined time-slot interfaces.

The IDL and GCI (IOM-2) interfaces are used to connect to semiconductor devices that support the Integrated Services Digital Network (ISDN). IDL and GCI allow the MC68302 to communicate over any of the 2B + D ISDN basic rate channels.

When using the IDL or GCI buses, additional control functions in the frame structure are required. These functions are supported in the MC68302 through two SMC channels: SMC1 and SMC2. (For other matters relating to the SMCs, refer to 4.7 Serial Management Controllers (SMCs)).

The serial interface also supports two testing modes: echo and loopback. Echo mode provides a return signal from the physical interface by retransmitting the signal it has received. The physical interface echo mode differs from the individual SCC echo mode in that it can operate on the entire multiplexed signal rather than just on a particular SCC channel (which may further have particular bits masked). Loopback mode causes the physical interface to receive the same signal it is transmitting. The physical interface loopback mode checks more than the individual SCC loopback mode; it checks the physical interface and the internal channel routes.

Refer to Figure 4-5 for the serial channels physical interface block diagram.

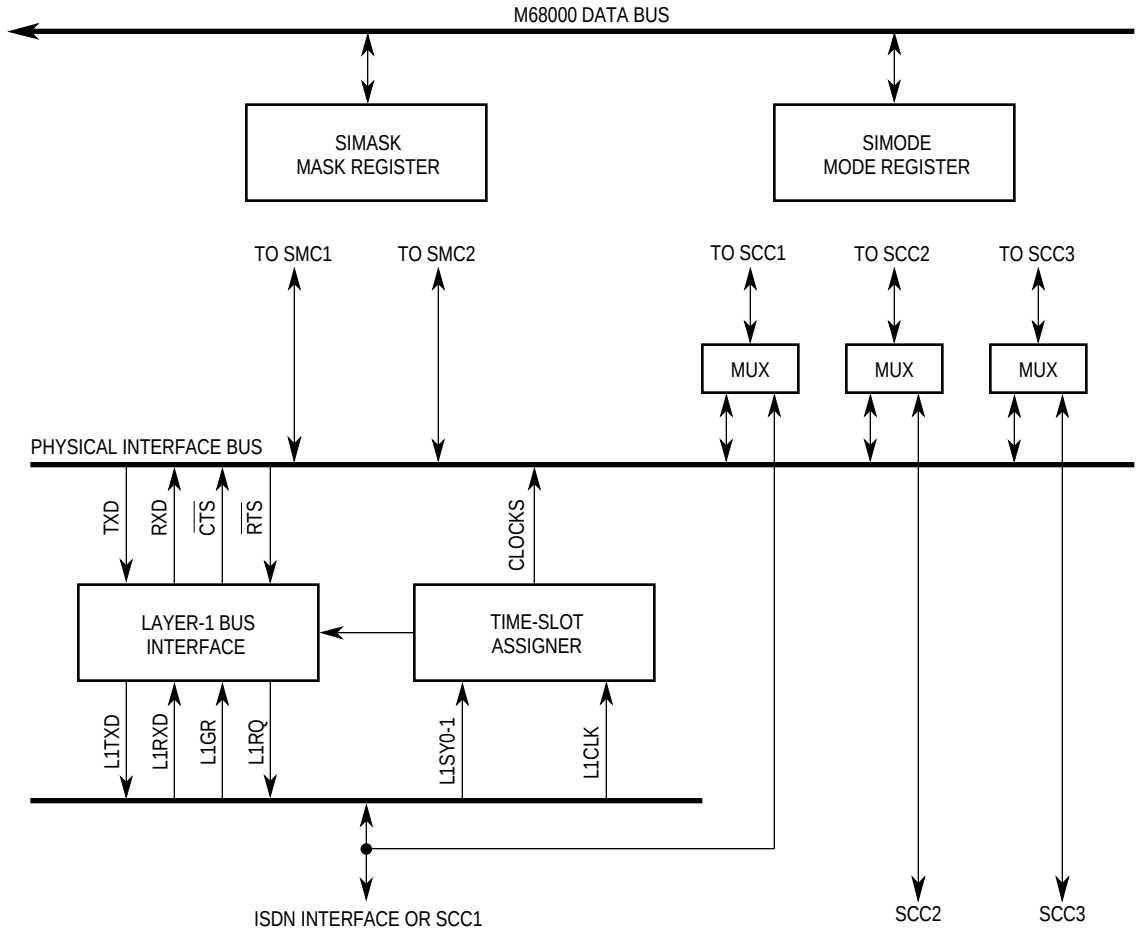


Figure 4-5. Serial Channels Physical Interface Block Diagram

4.4.1 IDL Interface

The IDL interface is a full-duplex ISDN interface used to interconnect a physical layer device (such as the Motorola ISDN S/T transceiver MC145474) to the integrated multiprotocol processor (IMP). Data on five channels (B1, B2, D, A, and M) is transferred in a 20-bit frame every 125 µs, providing 160-kbps full-duplex bandwidth. The IMP is an IDL slave device that is clocked by the IDL bus master (physical layer device). The IMP provides direct connections to the MC145474. Refer to Figure 4-6 for the IDL bus signals.

The IMP supports 10-bit IDL as shown in Figure 4-6; it does not support 8-bit IDL.

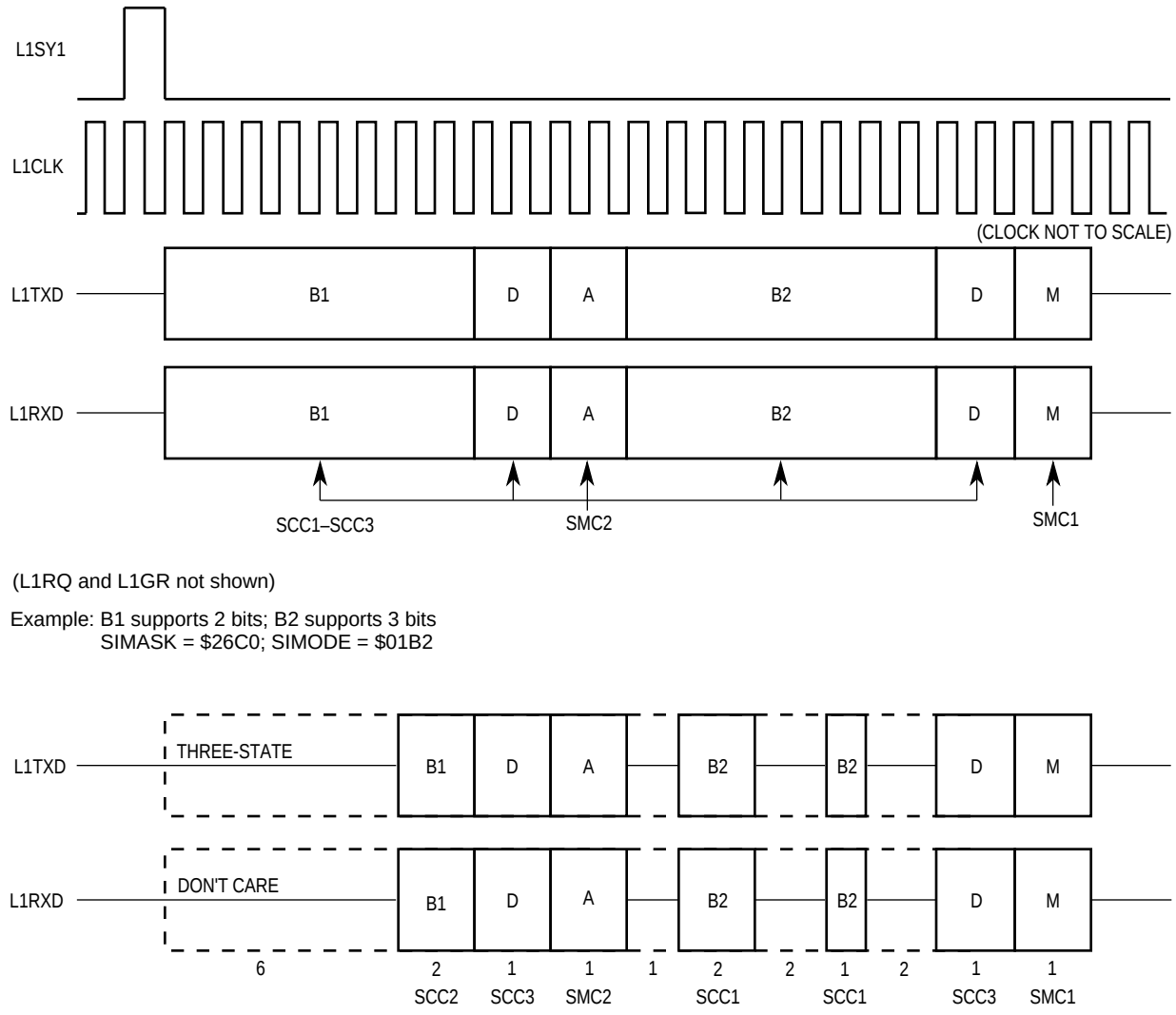


Figure 4-6. IDL Bus Signals

An application of the IDL interface is to build a basic rate ISDN terminal adaptor (see Figure 4-7). In such an application, the IDL interface is used to connect the 2B + D channels between the IMP, CODEC, and S/T transceiver. One of the IMP SCCs would be configured to HDLC mode to handle the D channel; another IMP SCC would be used to rate adapt the

terminal data stream over the B channel. That SCC would be configured for HDLC mode if V.120 rate adaption is required, or for V.110 mode if V.110 rate adaption is required. The second B channel could be routed to the CODEC as a digital voice channel, if desired. The SCP is used to send initialization commands and periodically check status from the S/T transceiver. The SCC connected to the terminal would be configured for UART, BISYNC, or DDCMP mode depending on the terminal protocol used.

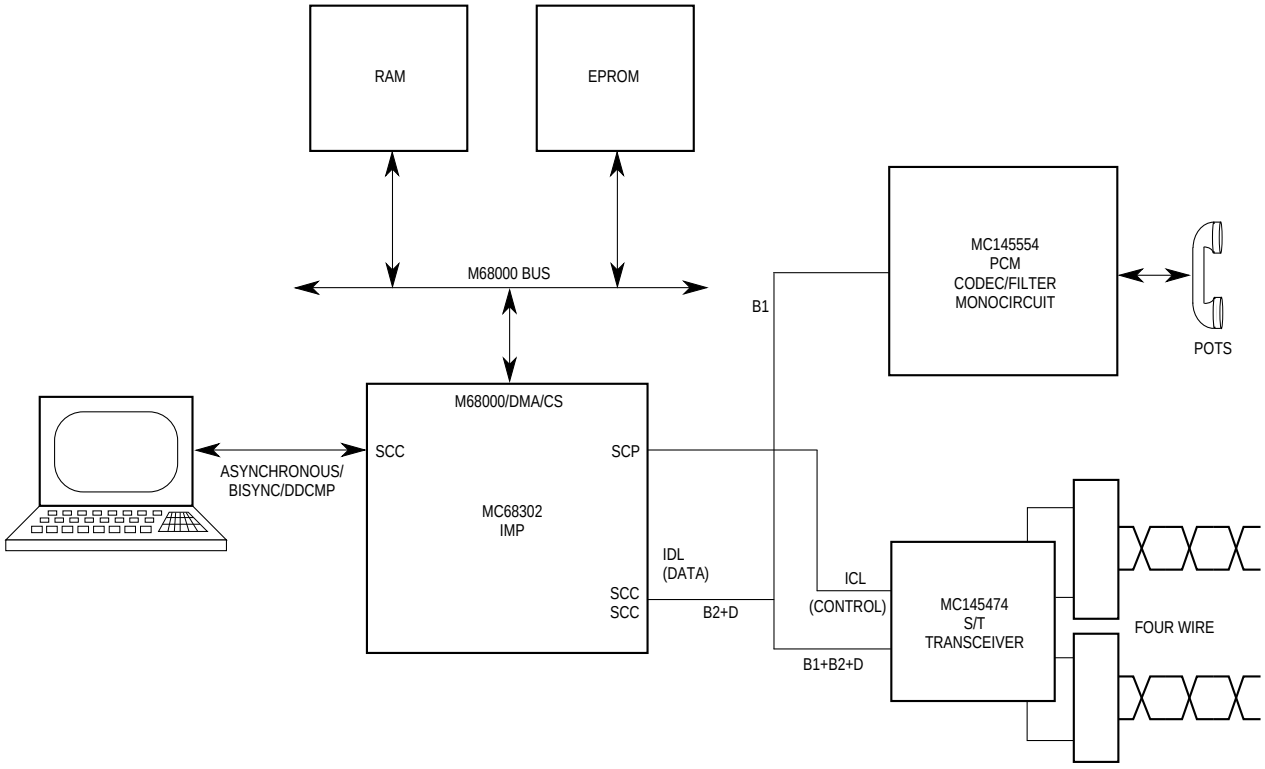


Figure 4-7. IDL Terminal Adaptor

The IMP has two output data strobe lines (SDS1 and SDS2) for selecting either or both the B1 and B2 channels. These signals are used for interfacing devices that do not support the IDL bus. These signals, configured by the SIMASK register, are active only for bits that are not masked. The IDL signals are as follows:

L1CLK	IDL clock; input to the IMP.
L1TXD	IDL transmit data; output from the IMP. Valid only for the bits that are supported by the IDL; three-stated otherwise.
L1RXD	IDL receive data; input to the IMP. Valid for the 20 bits of the IDL; ignored for other signals that may be present.
L1SY1	IDL SYNC signal; input to the IMP. This signal indicates that the 20 clock periods following the pulse designate the IDL frame.
L1RQ	Request permission to transmit on the D channel; output from the IMP.
L1GR	Grant permission to transmit on the D channel; input to the IMP.
SDS1	Serial data strobe 1
SDS2	Serial data strobe 2

NOTE

The IDL bus signals, L1TXD and L1RXD, require pull-up resistors in order to ensure proper operation with transceivers.

In addition to the 144-kbps ISDN 2B + D channels, IDL provides channels for maintenance and auxiliary bandwidth. The IDL bus has five channels:

B1	64-kbps Bearer Channel
B2	64-kbps Bearer Channel
D	16-kbps Signaling Channel
M	8-kbps Maintenance Channel (not required by IDL)
A	8-kbps Auxiliary Channel (not required by IDL)

The IMP supports all five channels of the IDL bus. The following table shows where each channel can be routed. The two B channels can be concatenated and routed to the same SCC channel.

IDL Channel	Serial Controllers
D	SCC1, SCC2, SCC3
B1	SCC1, SCC2, SCC3
B2	SCC1, SCC2, SCC3
M	SMC1
A	SMC2

The IMP supports the request-grant method for contention detection on the D channel. When the IMP has data to transmit on the D channel, it asserts L1RQ. The physical layer device monitors the physical layer bus for activity on the D channel and indicates that the channel is free by asserting L1GR. The IMP samples the L1GR signal when L1SY1 is asserted. If L1GR is high (active), the IMP transmits the first zero of the opening flag in the first bit of the D channel. If a collision is detected on the D channel, the physical layer device negates L1GR. The IMP then stops its transmission and retransmits the frame when L1GR is asserted again. This is handled automatically for the first two buffers of the frame.

The IDL interface supports the CCITT I.460 recommendation for data rate adaptation. The IDL interface can access each bit of the B channel as an 8-kbps channel. A serial interface mask register (SIMASK) for the B channels specifies which bits are supported by the IDL interface. The receiver will support only the bits enabled by SIMASK. The transmitter will transmit only the bits enabled by the mask register and will three-state L1TXD otherwise.

Refer to Figure 4-6 for an example of supporting two bits in the B1 channel and three bits in the B2 channel.

4.4.2 GCI Interface

The normal mode of the GCI (also known as ISDN-Oriented Modular rev 2.2 (IOM2)) ISDN bus is fully supported by the IMP. The IMP also supports channel 0 of the Special Circuit Interface T (SCIT) interface, and in channel 2 of SCIT, supports the D channel access control for S/T interface terminals, using the command/indication (C/I) field. The IMP does not support the Telecom IC (TIC) bus.

The GCI bus consists of four lines: two data lines, a clock, and a frame synchronization line. Usually an 8-kHz frame structure defines the various channels within the 256-kbps data rate as indicated in Figure 4-8. However, the interface can also be used in a multiplexed frame structure on which up to eight physical layer devices multiplex their GCI channels. L1SY1 must provide the channel SYNC. In this mode, the data rate would be 2048 kbps.

The GCI clock rate is twice the data rate. The clock rate for the IMP must not exceed the ratio of 1:2.5 serial clock to parallel clock. Thus, for a 16.67-MHz system clock, the serial clock rate must not exceed 6.67 MHz.

The IMP also supports another line for D-channel access control—the L1GR line. This signal is not part of the GCI interface definition and may be used in proprietary interfaces.

NOTE

When the L1GR line is not used, it should be pulled high. The IMP has two data strobe lines (SDS1 and SDS2) for selecting either or both of the B1 and B2 channels and the data rate clock (L1CLK). These signals are used for interfacing devices that do not support the GCI bus. They are configured with the SIMASK register and are active only for bits that are not masked.

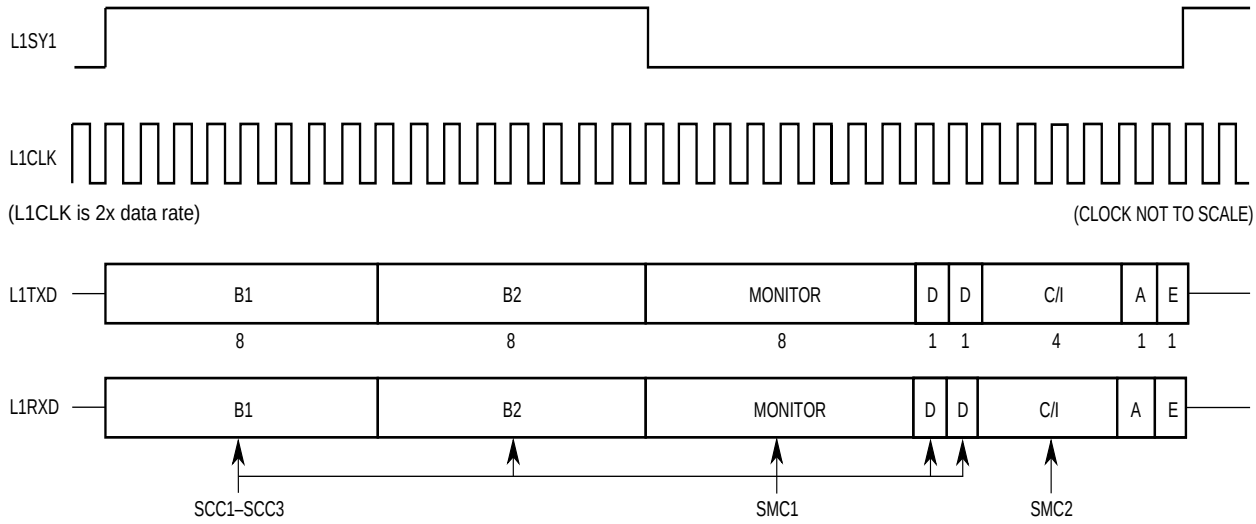


Figure 4-8. GCI Bus Signals

The GCI signals are as follows:

L1CLK	GCI clock; input to the IMP.
L1TXD	GCI transmit data; open drain output.
L1RXD	GCI receive data; input to the IMP.
L1SY1	GCI SYNC signal; input to the IMP.
L1GR	Grant permission to transmit on the D channel; input to the IMP.
SDS1	Serial data strobe 1; output from the IMP.
SDS2	Serial data strobe 2; output from the IMP.
GCIDCL	GCI interface data clock; output from the IMP.

NOTE

The GCI bus signals, L1TXD and L1RXD, require pull-up resistors in order to ensure proper operation with transceivers.

The GCI bus has five channels:

B1	64-kbps Bearer Channel (8 bits)
B2	64-kbps Bearer Channel (8 bits)
M	64-kbps Monitor Channel (8 bits)
D	16-kbps Signaling Channel (2 bits)
C/I, A, E	48-kbps Command/Indication Channel (6 bits)

In addition to the 144-kbps ISDN 2B + D channels, GCI provides two channels for maintenance and control functions.

The monitor channel is used to transfer data between layer-1 devices and the control unit (i.e., the M68000 core). The command/indication channel is used to control activation/deactivation procedures or for the switching of test loops by the control unit.

The IMP supports all five channels of the GCI channel 0. The following table shows where each channel can be routed. The two B channels can be concatenated and routed to the same SCC channel.

GCI Channel 0	Serial Controllers
D	SCC1, SCC2, SCC3
B1	SCC1, SCC2, SCC3
B2	SCC1, SCC2, SCC3
M	SMC1
C/I	SMC2

The GCI interface supports the CCITT I.460 recommendation for data rate adaptation. The GCI interface can access each bit of the B channel as an 8-kbps channel. The mask register (SIMASK) for the B channels specifies which bits are supported by the GCI interface. The receiver will receive only the bits that are enabled by SIMASK; the transmitter will transmit only the bits that are enabled by SIMASK and will not drive the L1TXD pin otherwise (L1TXD in GCI mode is an open-drain output).

The IMP supports contention detection on the D channel. When the IMP has data to transmit on the D channel, it checks bit 4 of the SCIT C/I channel 2. The physical layer device monitors the physical layer bus for activity on the D channel and indicates with this bit that the channel is free. If a collision is detected on the D channel, the physical layer device sets bit 4 of C/I channel 2 to logic high. The IMP then aborts its transmission and retransmits the frame when this bit is asserted again. This procedure is handled automatically for the first two buffers of a frame. The L1GR line may also be used for access to the S interface D channel. This signal is checked by the IMP, and the physical layer device should indicate that the S interface D channel is free by asserting L1GR.

In the deactivated state, the clock pulse is disabled, and the data line is a logic one. The layer-1 device activates the IMP by enabling the clock pulses and by an indication in the channel 0 C/I channel. The IMP will then report to the M68000 core by a maskable interrupt that a valid indication is in the SMC2 receive buffer descriptor.

When the M68000 core activates the line, it sets SETZ in the serial interface mode (SIMODE) register, causing the data output from L1TXD to become a logic zero. Code 0 (command timing TIM) will be transmitted on channel 0 C/I channel to the layer-1 device until the SETZ is reset. The physical layer device will resume transmitting the clock pulses and will give an indication in the channel 0 C/I channel. The M68000 core should reset SETZ to enable data output.

4.4.3 PCM Highway Mode

In PCM highway mode, one, two, or all three SCCs can be multiplexed together to support various time-division multiplexed interfaces. PCM highway supports the standard T1 and CEPT interfaces as well as user-defined interfaces. In this mode, the NMSI1 pins have new names and functions (see Table 4-2).

Table 4-2. PCM Highway Mode Pin Functions

Signal	Definition	Function
L1RXD	Receive Data	Input
L1TXD	Transmit Data	Output
L1CLK	Receive and Transmit Clock	Input
L1SY0	Sync Signal 0	Input
L1SY1	Sync Signal 1	Input
$\overline{\text{RTS1}}$, $\overline{\text{RTS2}}$, $\overline{\text{RTS3}}$	Three Request-to-Send Signals	Outputs

L1CLK is always an input to the MC68302 in PCM highway mode and is used as both a receive and transmit clock. Thus, data is transmitted and received simultaneously in PCM highway mode. (If receive data needs to be clocked into the MC68302 at a different time or speed than transmit data is being clocked out, then NMSI mode should be used instead of PCM highway.)

The two sync signals, L1SY0 and L1SY1, are also inputs to the MC68302. They select one of three PCM channels to which data is routed or select no channel (see Table 4-3).

Table 4-3. PCM Channel Selection

L1SY1	L1SY0	Selection
0	0	No Channel Selected
0	1	PCM Channel 1 Selected
1	0	PCM Channel 2 Selected
1	1	PCM Channel 3 Selected

A PCM channel is not an SCC channel. A PCM channel is an intermediate internal channel that can be routed to any SCC, as selected in the SIMODE register. This extra layer of indirection keeps the hardware (which must generate L1SY1 and L1SY0 signals externally) from having to be modified if a change in the SCC data routing is required.

The routing of each channel is determined in the SIMODE register by the DRB-DRA bits for channel 1, the B1RB–B1RA bits for channel 2, and the B2RB–B2RA bits for channel 3. Once the routing of a PCM channel is selected, data is transmitted from the selected SCC transmitter over the physical interface using the L1CLK pin. At the same time, data is received from the physical interface and routed to the selected SCC receiver. When no sync is asserted, the L1TXD pin is three-stated, and the L1RXD pin is ignored.

Two different methods exist for using the L1SY1–L1SY0 pins: one-clock-prior mode and envelope mode (see Figure 4-9). In one-clock-prior mode, the sync signals should go active for a single clock period prior to an 8-bit time slot. In envelope mode, the sync signals should go active on the first bit of the time slot and stay active the entire time slot. The envelope mode is more general, allowing a time slot to be from one to N bits long.

An example of the use of the L1SY1 and L1SY0 sync signals in the envelope mode is shown in Figure 4-10. The three PCM channels defined in the figure show some of the flexibility available in the PCM highway envelope mode. As shown, PCM channel time slots do not

have to be contiguous in the PCM highway, but rather can be separated by other time slots. Also, PCM channel time slots need not be an even multiple of eight bits in envelope mode. Although not shown in the figure, it is also possible to route multiple PCM channels to a single SCC, causing the SCC to process one higher speed data stream.

The PCM highway interface also supports the $\overline{\text{RTS}}$ signals. They will be asserted (just like in NMSI mode) when an SCC desires to transmit over the PCM highway and will stay asserted until the entire frame is transmitted (regardless of how many time slots that takes). The $\overline{\text{RTS}}$ signal that asserts corresponds to the SCC that desires to transmit. The $\overline{\text{RTS}}$ signals may be useful in debugging but are not required for proper PCM highway operation. If the $\overline{\text{RTS}}$ signals are not needed, they can be ignored or reassigned as parallel I/O.

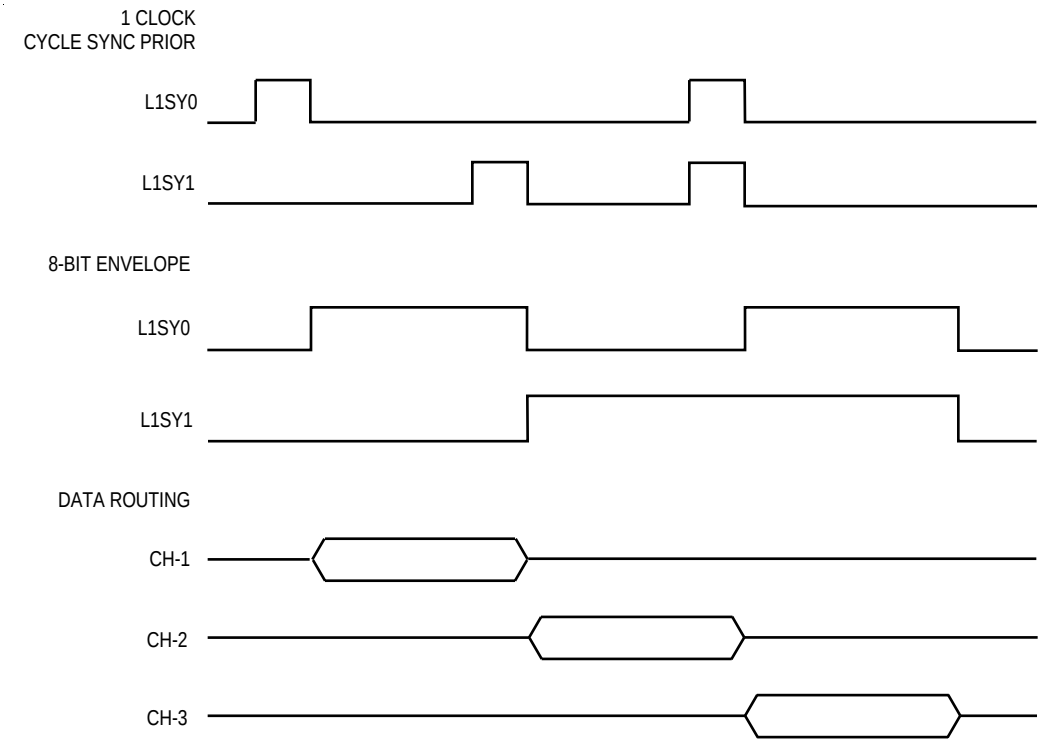
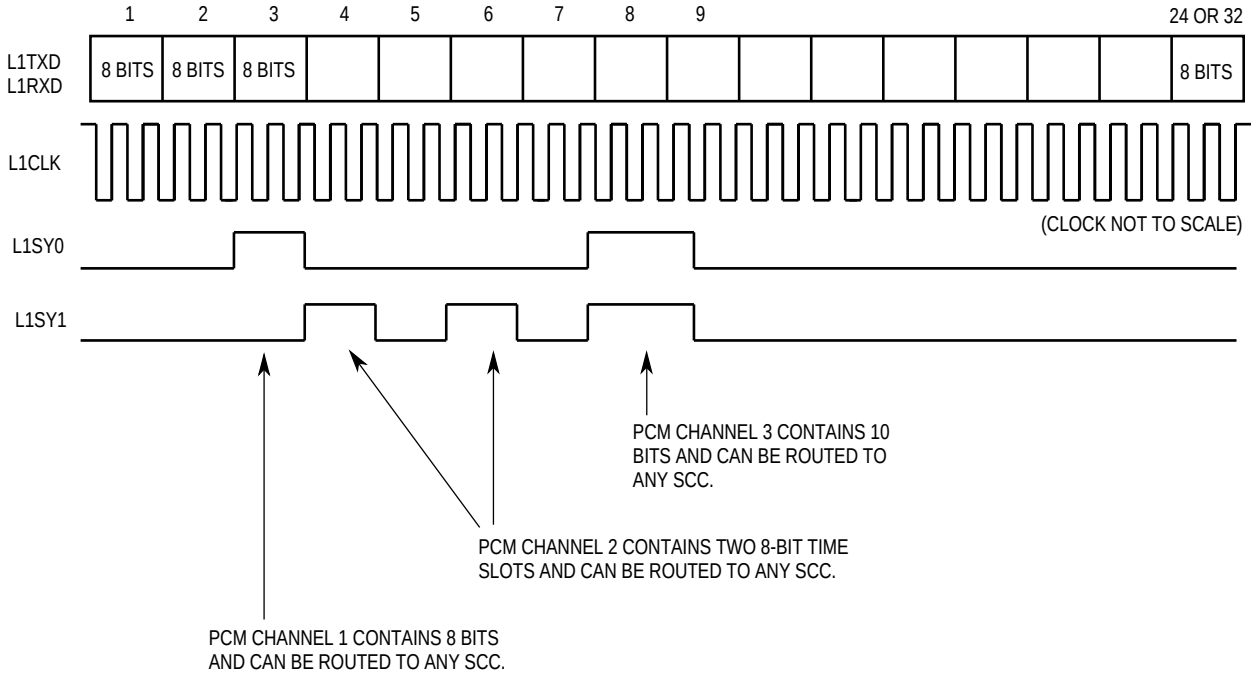


Figure 4-9. Two PCM Sync Methods



NOTE: Whenever the syncs are active, data from that SCC is transmitted and received using L1CLK edges.

Figure 4-10. PCM Channel Assignment on a T1/CEPT Line

4.4.4 Nonmultiplexed Serial Interface (NMSI)

The IMP supports the NMSI with modem signals. In this case, the serial interface connects the seven serial lines of the NMSI1/ISDN interface (RXD1, TXD1, RCLK1, TCLK1, CD1, CTS1, and RTS1) directly to the SCC1 controller. NMSI pins associated with SCC2 and SCC3 can be used as desired or left as general-purpose I/O port pins. See 3.3 Parallel I/O Ports for an example. $\overline{\text{RTS}}$ is an output of the transmitter, while $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ are inputs to the transmitter and receiver, respectively. See 4.5.3 SCC Mode Register (SCM) and 4.4 Serial Channels Physical Interface for additional information.

$\overline{\text{CTS}}$ and $\overline{\text{CD}}$ may be programmed to control transmission and reception automatically or to just generate interrupts.

4.4.5 Serial Interface Registers

There are two serial interface registers: SIMODE and SIMASK. The SIMODE register is a 16-bit register used to define the serial interface operation modes. The SIMASK register is a 16-bit register used to determine which bits are active in the B1 and B2 channels of ISDN.

4.4.5.1 Serial Interface Mode Register (SIMODE)

If the IDL or GCI mode is used, this register allows the user to support any or all of the ISDN channels independently. Any extra SCC channel can then be used for other purposes in

NMSI mode. The SIMODE register is a memory-mapped read-write register cleared by reset.

15	14	13	12	11	10	9	8
SETZ	SYNC/SCIT	SDIAG1	SDIAG0	SDC2	SDC1	B2RB	B2RA

7	6	5	4	3	2	1	0
B1RB	B1RA	DRB	DRA	MSC3	MSC2	MS1	MS0

SETZ—Set L1TXD to zero (valid only for the GCI interface)

0 = Normal operation
1 = L1TXD output set to a logic zero (used in GCI activation, refer to 4.4.2 GCI Interface)

SYNC/SCIT—SYNC Mode/SCIT Select Support

SYNC is valid only in PCM mode.

0 = One pulse wide prior to the 8-bit data
1 = N pulses wide and envelopes the N-bit data

The SCIT (Special Circuit Interface T) interface mode is valid only in GCI mode.

0 = SCIT support disabled
1 = SCIT D-channel collision enabled. Bit 4 of channel 2 C/I used by the IMP for receiving indication on the availability of the S interface D channel.

SDIAG1–SDIAG0—Serial Interface Diagnostic Mode (NMSI1 Pins Only)

00 = Normal operation
01 = Automatic echo
The channel automatically retransmits the received data on a bit-by-bit basis. The receiver operates normally, but the transmitter can only retransmit received data. In this mode, L1GR is ignored.
10 = Internal loopback
The transmitter output (L1TXD) is internally connected to the receiver input (L1RXD). The receiver and the transmitter operate normally. Transmitted data appears on the L1TXD pin, and any external data received on L1RXD pin is ignored. In this mode, L1RQ is asserted normally, and L1GR is ignored.
11 = Loopback control
In this mode, the transmitter output (TXD1/L1TXD) is internally connected to the receiver input (RXD1/L1RXD). The TXD1/L1TXD, TXD2, TXD3, $\overline{\text{RTS1}}$, $\overline{\text{RTS2}}$, and $\overline{\text{RTS3}}$ pins will be high, but L1TXD will be three-stated in IDL and PCM modes. This mode may be used to accomplish multiplex mode loopback testing without affecting the multiplexed layer 1 interface. It also prevents an SCC's individual loopback (configured in the SCM) from affecting the pins of its associated NMSI interface.

SDC2—Serial Data Strobe Control 2

0 = SDS2 signal is asserted during the B2 channel
1 = SDS1 signal is asserted during the B2 channel

SDC1—Serial Data Strobe Control 1

0 = SDS1 signal is asserted during the B1 channel
1 = SDS2 signal is asserted during the B1 channel

B2RB, B2RA—B2 Channel Route in IDL/GCI Mode or CH-3 Route in PCM Mode

00 = Channel not supported
01 = Route channel to SCC1
10 = Route channel to SCC2 (if MSC2 is cleared)
11 = Route channel to SCC3 (if MSC3 is cleared)

B1RB, B1RA—B1 Channel Route in IDL/GCI Mode or CH-2 Route in PCM Mode

00 = Channel not supported
01 = Route channel to SCC1
10 = Route channel to SCC2 (if MSC2 is cleared)
11 = Route channel to SCC3 (if MSC3 is cleared)

DRB, DRA—D-Channel Route in IDL/GCI Mode or CH-1 Route in PCM Mode

00 = Channel not supported
01 = Route channel to SCC1
10 = Route channel to SCC2 (if MSC2 is cleared)
11 = Route channel to SCC3 (if MSC3 is cleared)

MSC3—SCC3 Connection

0 = SCC3 is connected to the multiplexed serial interface (PCM, IDL, or GCI) chosen in MS1–MS0. NMSI3 pins are all available for other purposes.
1 = SCC3 is not connected to a multiplexed serial interface but is connected directly to the NMSI3 pins or SCP pins or is not used. The choice of general-purpose I/O port pins versus SCC3 functions is made in the port A control register. The choice of SCP pins versus SCC3 functions is made in the SPMODE register.

MSC2—SCC2 Connection

0 = SCC2 is connected to the multiplexed serial interface (PCM, IDL, or GCI) chosen in MS1–MS0. NMSI2 pins are all available for other purposes.
1 = SCC2 is not connected to a multiplexed serial interface but is either connected directly to the NMSI2 pins or not used. The choice of general-purpose I/O port pins versus SCC2 functions is made in the port A control register.

MS1—MS0—Mode Supported

00 = NMSI Mode
When working in NMSI mode, SCC1 is connected directly to the seven NMSI1 pins (RXD1, TXD1, RCLK1, TCLK1, $\overline{CD1}$, $\overline{CTS1}$, and $\overline{RTS1}$). SCC2 functions can be routed to port A as NMSI functions or configured instead as PA6–PA0. Four of the SCC3 functions can be routed to port A or retained as PA11–PA8. The other

three SCC3 functions ($\overline{CTS3}$, $\overline{RTS3}$, and $\overline{CD3}$) can be routed to replace the three SCP pins or else not used.
In NMSI mode, the MSC2 and MSC3 bits are ignored. The choice of general-purpose I/O port pins versus SCC2 and SCC3 functions is made in the port A control register. See 3.3 Parallel I/O Ports for an example and more information. The choice of SCP pins versus three SCC3 functions is made in the SPMODE register in the SCP. See 4.6 Serial Communication Port (SCP) for more details.

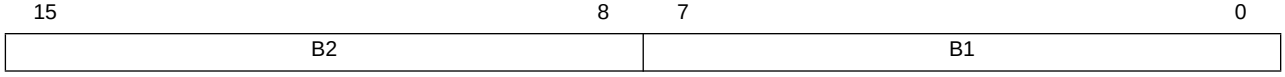
01 = PCM Mode
When working in PCM mode, each of the three multiplexed channels CH-1, CH-2, and CH-3 can be routed independently to each of the three SCCs. This connection is determined by the DRB, DRA, B1RB, B1RA, B2RB, and B2RA bits. SCC2 and SCC3 can be connected directly to their respective NMSI pins (if they are not needed for the PCM channels) as determined by the MSC3–MSC2 bits. In the NMSI case, the choice still exists for port/SCP functions versus SCC functions as described in case 00. The MSC3–MSC2 bits override the PCM routing for a specific SCC.

10 = IDL Mode
When working in IDL/GCI mode, each ISDN channel (D, B1, and B2) can be routed independently to each of the three SCCs. This connection is determined by the DRB, DRA, B1RB, B1RA, B2RB, and B2RA bits. SCC2 and SCC3 can be connected directly to their respective NMSI pins (if they are not needed for ISDN channels) determined by the MSC3–MSC2 bits. In the NMSI case, the choice still exists for port/SCP functions versus SCC functions as described in case 00. Note that the MSC3–MSC2 bits override the ISDN connection for a specific SCC.

11 = GCI Interface
Refer to the IDL mode description.

4.4.5.2 Serial Interface Mask Register (SIMASK)

The SIMASK register, a memory-mapped read-write register, is set to all ones by reset. SIMASK is used in IDL and GCI to determine which bits are active in the B1 and B2 channels. Any combination of bits may be chosen. A bit set to zero is not used by the IMP. A bit set to one signifies that the corresponding B channel bit is used for transmission and reception on the B channel. Note that the serial data strobes, SD1 and SD2, are asserted for the entire 8-bit time slot independent of the setting of the bits in the SIMASK register.



NOTE

Bit 0 of this register is the first bit transmitted or received on the IDL/GCI B1 channel.

4.5 SERIAL COMMUNICATION CONTROLLERS (SCCS)

The IMP contains three independent SCCs, each of which can implement different protocols. This configuration provides the user with options for controlling up to three independent full-duplex lines implementing bridges or gateway functions or multiplexing up to three SCCs onto the same physical layer interface to implement a 2B + D ISDN basic rate channel or

three channels of a PCM highway. Each protocol-type implementation uses identical buffer structures to simplify programming.

The following protocols are supported: HDLC/SDLC, BISYNC, synchronous and asynchronous DDCMP, UART, several transparent modes, and V.110 rate adaption support. Each protocol can be implemented with IDL, GCI, PCM, or NMSI physical layer interfaces (see 4.4 Serial Channels Physical Interface) and can be configured to operate in either echo or loopback mode. Echo mode provides a return signal from an SCC by retransmitting the received signal. Loopback mode is a local feedback connection allowing an SCC to receive the signal it is transmitting. (Echo and loopback mode for multiplexed interfaces are discussed in 4.4 Serial Channels Physical Interface).

The receive and transmit section of each SCC is supported with one of the six dedicated SDMA channels (see 4.2 SDMA Channels). These channels transfer data between the SCCs and either external RAM or on-chip dual-port RAM. This function is transparent to the user, being enabled and controlled according to the configuration of each SCC channel. Each SCC can be clocked by either an external source (with the clock pins RCLK or TCLK) or by an internal source through a baud rate generator for each SCC channel. The baud rate generator can derive its clock from the main IMP clock or from a separate input clock. The SCC transmitter and receiver sections are independent and may be clocked at different rates.

The SCCs exhibit two types of performance limitations. The first type is a hardware clocking limit, which is the same for each SCC. The SCC clocks must not exceed a ratio of 1:2.5 serial clock (RCLK or TCLK) to parallel clock (EXTAL). Thus, for a 16.67-MHz system clock frequency, the serial clock must not exceed 6.67 MHz. The second type concerns the system data rate. The SDMA channels and CP main controller must have enough time to service the SCCs, thus preventing FIFO underruns and overruns in the SCCs. This requirement depends on a number of factors discussed in more detail in Appendix A SCC Performance.

Each SCC supports the standard seven-line modem interface (also referred to as NMSI) with the signals RXD, TXD, RCLK, TCLK, $\overline{\text{RTS}}$, $\overline{\text{CTS}}$, and $\overline{\text{CD}}$. Other modem signals (such as DSR and DTR) may be supported through the parallel I/O pins. A block diagram of the SCC is depicted in Figure 4-11.

To provide extra modem serial output lines, the user must define I/O port A or B pins as outputs in the port A/B data direction register and write to the port A/B data register to cause the state of the pin to change. Extra serial input lines with interrupts may be supported by defining the port B pins as inputs in the port B data direction register. When a change in the state of the pin occurs, the interrupt handler may assert or negate the extra outputs to support the hand-shaking protocol. (See 3.3 Parallel I/O Ports for related details.)

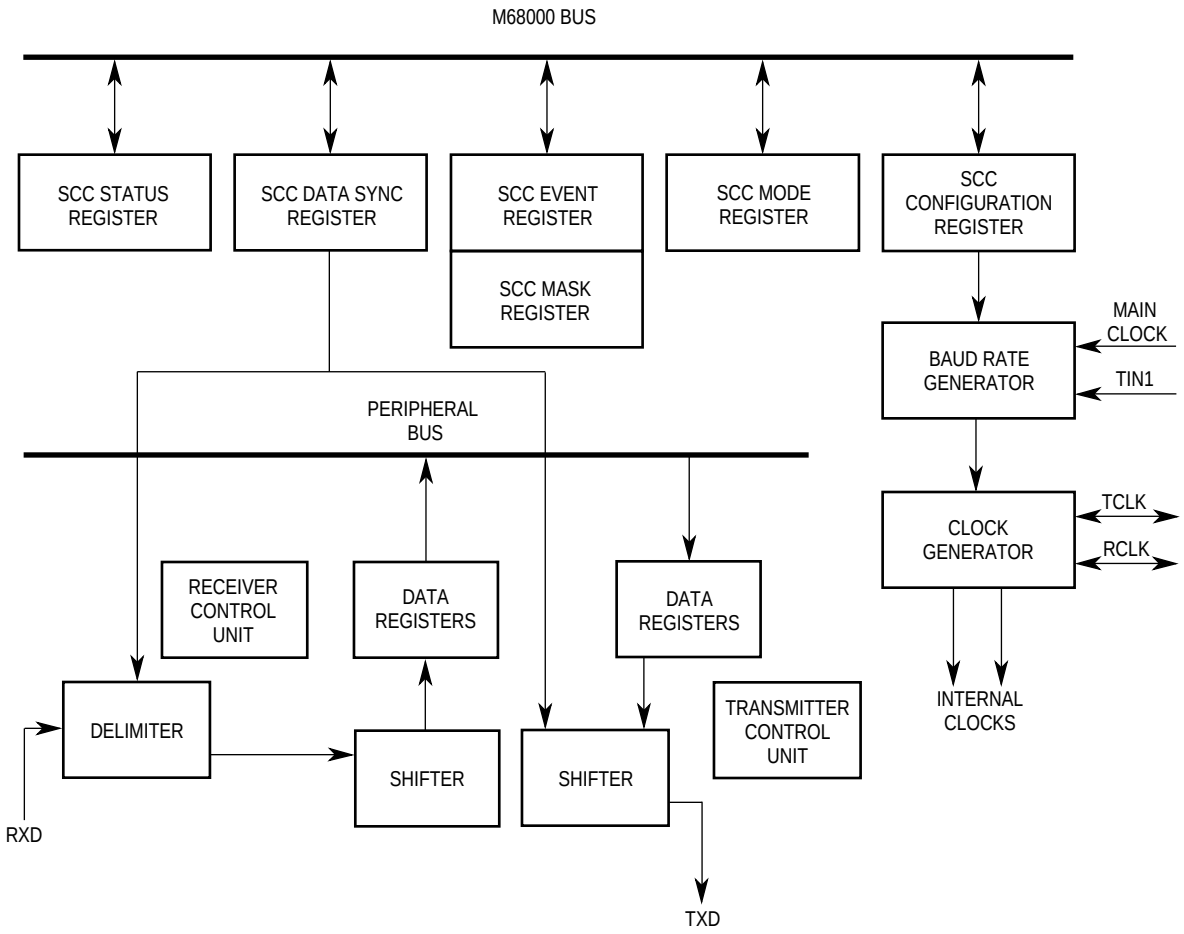


Figure 4-11. SCC Block Diagram

4.5.1 SCC Features

Each SCC channel has the following features:

- HDLC/SDLC, BISYNC, DDCMP, UART, Transparent, or V.110 Protocols
- Programmable Baud Rate Generator Driven by Main Clock or an External Clock
- Data Clocked by the Baud Rate Generator or Directly by an External Pin
- Supports Modem Signals (RXD, TXD, RCLK, TCLK, $\overline{\text{RTS}}$, $\overline{\text{CTS}}$, and $\overline{\text{CD}}$)
- Full-Duplex Operation
- Echo Mode
- Local Loopback Mode
- Baud Rate Generator Outputs Available Externally

4.5.2 SCC Configuration Register (SCON)

Each SCC controller has a configuration register that controls its operation and selects its clock source and baud rate. Figure 4-12 shows one of the three SCC baud rate generators.

Each SCON is a 16-bit, memory-mapped, read-write register. The SCONs are set to $0004 by reset, resulting in the baud rate generator output clock rate being set to the main clock rate divided by 3. The baud rate generator output clock is always available externally, as shown in Table 5-8.

NOTE

The BRG output are 180 degrees out of phase to the TCLK and RCLK signals used by the SCC and output on the RCLK and TCLK pins.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WOMS	EXTC	TCS	RCS	CD10	CD9	CD8	CD7	CD6	CD5	CD4	CD3	CD2	CD1	CD0	DIV4

WOMS—Wired-OR

When WOMS is set, the TXD driver is programmed to function as an open-drain output and may be externally wired together with other TXD pins in an appropriate bus configuration. In this case, an external pullup resistor is required. When WOMS is cleared, the TXD pin operates normally with an active internal pullup.

NOTE

This bit is valid only in NMSI mode.

EXTC—External Clock

The EXTC bit selects whether the baud rate generator input clock source is the internal main clock (EXTC = 0) or external clock (EXTC = 1). If EXTC = 1, the external clock is taken from the TIN1 pin. Note that the single TIN1 pin can be used to supply a clock for all three baud rate generators.

TCS—Transmit Clock Source

The TCS bit selects either the baud rate generator output (TCS = 0) or the TCLK pin (TCS = 1) for the transmitter clock. If TCS = 0, then the baud rate generator output is driven onto the TCLK pin. This bit should be programmed to one if a multiplexed mode is chosen for the SCC.

After system reset, SCC hardware causes TCLK to default to an input and stay an input until a zero is written to TCS. For SCC2 and SCC3, TCLK can be derived directly from the RCLK pin as shown in Table 3-6.

RCS—Receive Clock Source

The RCS bit selects either the baud rate generator output (RCS = 0) or the RCLK pin (RCS = 1) for the receiver clock. If RCS = 0, then the baud rate generator output is driven onto the RCLK pin. This bit should be programmed to one if a multiplexed mode is chosen for the SCC.

After system reset, SCC hardware causes the RCLK to default to an input and stay an input until a zero is written to RCS.

CD10–CD0—Clock Divider

The clock divider bits and the prescaler determine the baud rate generator output clock rate. CD10–CD0 are used to preset an 11-bit counter that is decremented at the prescaler output rate. The counter is not otherwise accessible to the user. When the counter reaches zero, it is reloaded with the clock divider bits. Thus, a value of $7FF in CD10–CD0 produces the minimum clock rate (divide by 2048); a value of $000 produces the maximum clock rate (divide by 1).

NOTE

Because of SCC clocking restrictions, the maximum baud rate that may be used to clock an SCC is divide by 3.

When dividing by an odd number, the counter ensures a 50% duty cycle by asserting the terminal count once on a clock high and next on a clock low. The terminal count signals the counter expiration and toggles the clock.

DIV4—SCC Clock Prescaler Divide by 4

The SCC clock prescaler bit selects a divide-by-1 (DIV4 = 0) or divide-by-4 (DIV4 = 1) prescaler for the clock divider input. The divide-by-4 option is useful in generating very slow baud rates.

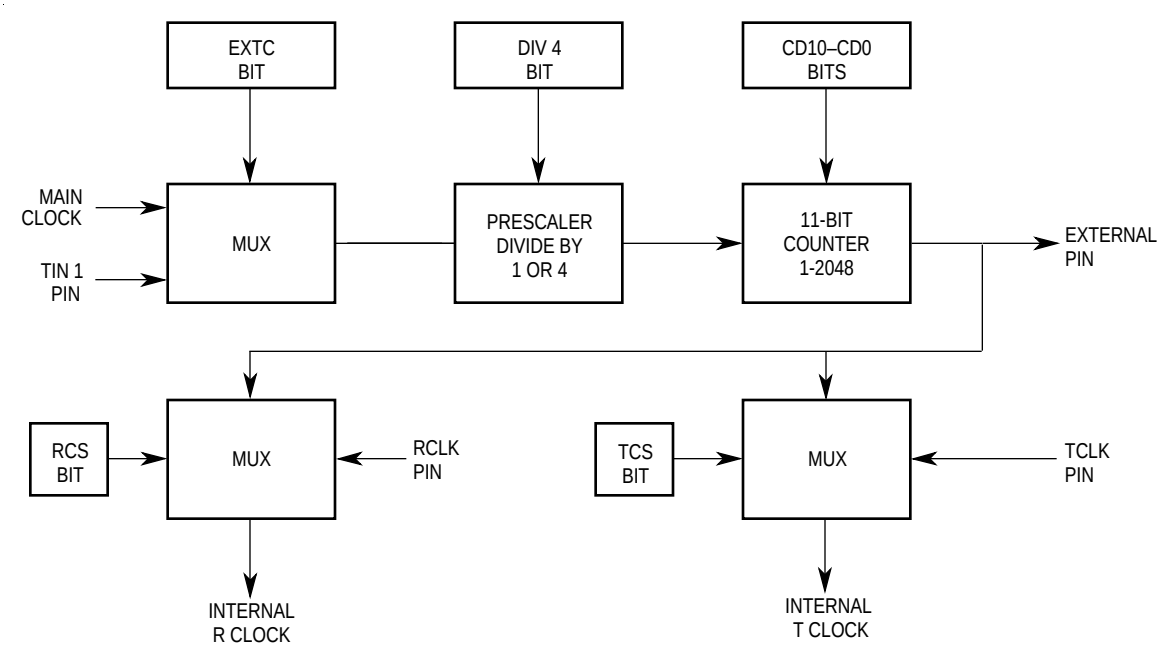


Figure 4-12. SCC Baud Rate Generator

4.5.2.1 Asynchronous Baud Rate Generator Examples

The UART circuitry always uses a clock that is 16x the baud rate. The ratio of the 16x UART clock to the system parallel clock must not exceed 1:2.5. For an internally supplied clock, an integer divider value must be used; therefore, the divider must be 3 or greater. Thus, using a clock divider value of 3 (programmed as 2 in the SCON) and a 16.67-MHz crystal gives a UART clock rate of 5.56 MHz and a baud rate of 347 kbaud. Assuming again a 16.67-MHz

crystal, an externally supplied UART clock on the TCLK or RCLK pins can be as high as 6.67 MHz, giving a maximum baud rate of 417 kbaud.

The baud rate using the baud rate generator is (System Clock or TIN1 clock)/(1 or 4)/(Clock Divider + 1)/16. The baud rate using the baud rate generator with an externally supplied clock to the TCLK or RCLK pins is always (TCLK or RCLK)/16.

Table 4-4 shows examples of typical bit rates of asynchronous communication and how to obtain them with the baud rate generator using an internally supplied clock.

Table 4-4. Typical Bit Rates of Asynchronous Communication

Baud Rates	15.36			16.0			16.667		
	DIV4	DIV	Actual Frequency	DIV4	DIV	Actual Frequency	DIV4	DIV	Actual Frequency
150	1	1599	150	1	1666	149.97	1	1735	150.01
300	1	799	300	1	832	300.12	1	867	300.02
600	0	1599	600	0	1666	599.88	0	1735	600.05
1200	0	799	1200	0	832	1200.48	0	867	1200.1
2400	0	399	2400	0	416	2398.08	0	433	2400.2
4800	0	199	4800	0	207	4807.69	0	216	4800.4
9600	0	99	9600	0	103	9615.34	0	108	9556.76
19200	0	49	19200	0	51	19230.8	0	53	19290.5
38400	0	24	38400	0	25	38461.53	0	26	38581.0

4.5.2.2 Synchronous Baud Rate Generator Examples

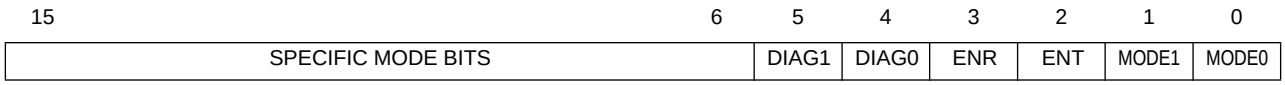
For synchronous communication (HDLC/SDLC, BISYNC, DDCMP, Transparent, and V.110), the internal clock is identical to the baud rate output. To obtain the desired rate, the user selects the appropriate system clock according to the following equation:

Baud rate = (System Clock or TIN1 Clock)/(Clock Divider + 1)/(1 or 4) according to the DIV4 bit

For example, to get the data rate of 64 kbps, the system clock can be 15.36 MHz, DIV4 = 0, and the Clock Divider = 239. Of course, a 64 kbps rate provided externally on the TCLK or RCLK pins could also be used.

4.5.3 SCC Mode Register (SCM)

Each SCC has a mode register. The functions of bits 5–0 are common to each protocol. The function of the specific mode bits varies according to the protocol selected by the MODE1–MODE0 bits. They are described in the relevant sections for each protocol type. Each SCM is a 16-bit, memory-mapped, read-write register. The SCMs are cleared by reset.



DIAG1–DIAG0—Diagnostic Mode

00 = Normal operation ($\overline{CTS}$, $\overline{CD}$ lines under automatic control)
In this mode, the CTS and CD lines are monitored by the SCC controller. The SCC controller uses these lines to automatically enable/disable reception and transmission.

If $\overline{RTS}$ is programmed to be asserted by the SCC, it will be asserted once buffered data is loaded into the transmit FIFO and a falling TCLK edge occurs. The following table shows the transmit data delays.

Table 4-5. Transmit Data Delay (TCLK Periods)

Protocol Type	From RTS Low	From CTS Low
Asynchronous Protocols (16x clock) Synchronous Protocols (1x clock)	0 1	48 3.5

NOTES:
1. $\overline{RTS}$ low values assume $\overline{CTS}$ is already asserted when $\overline{RTS}$ is asserted.
2. $\overline{CTS}$ low values assume $\overline{CTS}$ met the asynchronous setup time; otherwise, an additional clock may be added.

$\overline{RTS}$ is negated by the SCC one clock after the last bit in the frame. Figure 4-13 shows a diagram of synchronous mode timing from $\overline{RTS}$ low. Figure 4-14 shows a diagram of synchronous mode timing delays from $\overline{CTS}$ low.

The SCC samples $\overline{CTS}$ on the every rising edge of the TCLK. If $\overline{CTS}$ is negated when $\overline{RTS}$ is asserted, a CTS lost error occurs. If a synchronous protocol is used, the transmit data will be aborted after four additional bits are transmitted. If an asynchronous protocol is used, the transmit data will be aborted after three additional bits are transmitted. See the transmit error section of each protocol for further details and steps to be taken following a CTS lost error.

The SCC latches its first bit of valid receive data on the same clock edge (rising RCLK) that samples $\overline{CD}$ as low. The only exception is when the EXSYN bit is set in the SCC mode register for the BISYNC and Transparent protocols.

If $\overline{CD}$ is negated during frame reception, a CD lost error occurs and the SCC will quit receiving data within four additional bit times. At this point, any residue of bits less than 8 bits (or 16 bits in HDLC or transparent modes) will be discarded and not written to memory. Thus, the last bit written to memory will be within plus or minus four bit times from the point at which $\overline{CD}$ was negated.

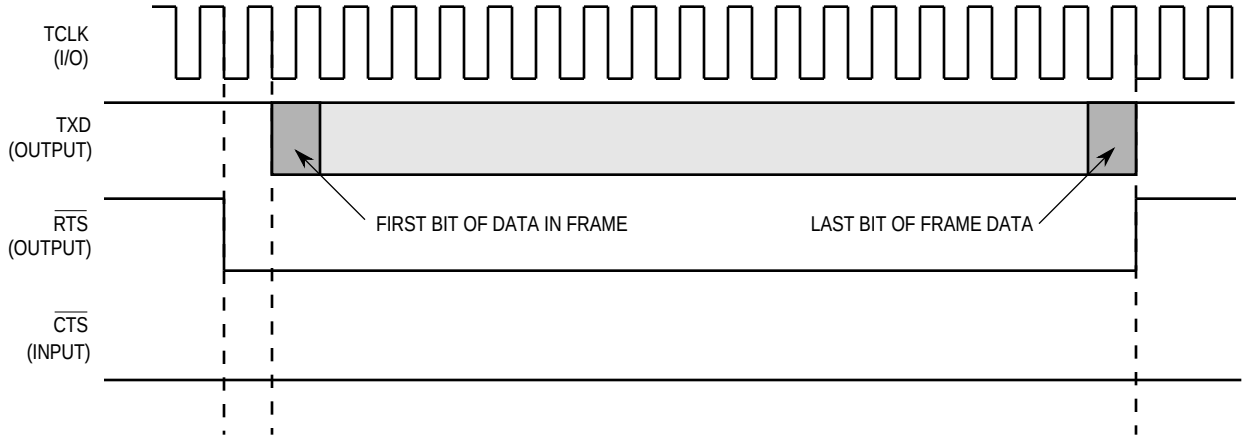
NOTE

The CTS lost error and CD lost error (with $\overline{CTS}$ and $\overline{CD}$ under automatic control) is not intended to implement a flow control method in the UART protocol. The software operation of the DIAG1–DIAG0 bits should be chosen if UART flow control is desired, with transmission being temporarily suspended by the FRZ bit in the UART event register. CTS lost and CD lost, as defined here, are intended to implement the aborting of transmission and reception as defined in many synchronous protocols.

01 = Loopback mode

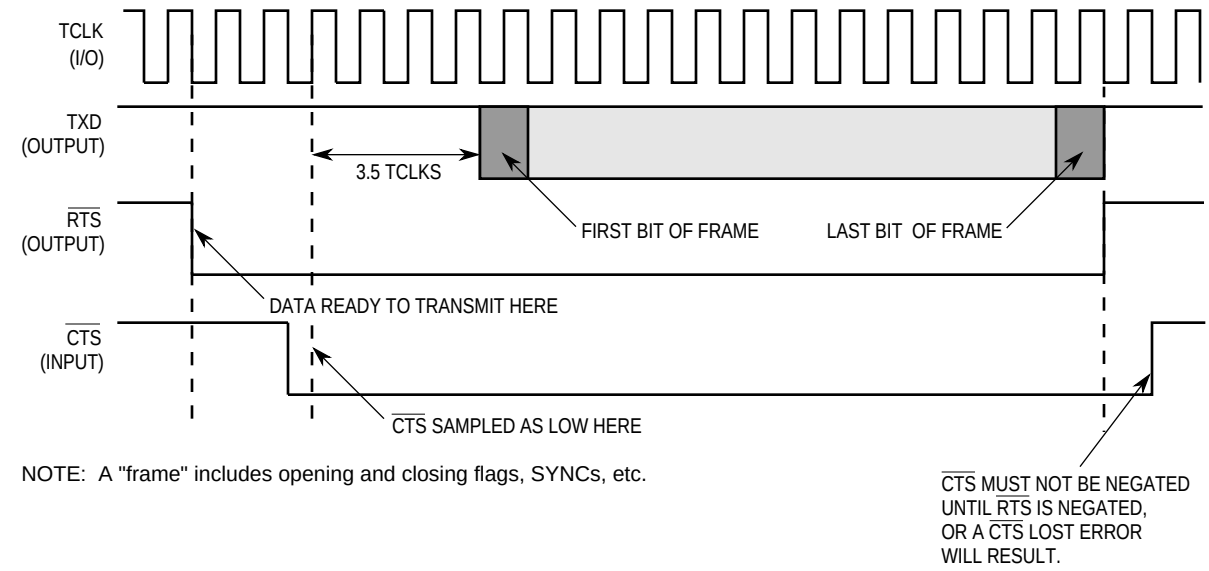
In this mode, the transmitter output is internally connected to the receiver input while the receiver and the transmitter operate normally. The value on the RXD pin is ignored. For the NMSI2 and NMSI3 pins, the TXD pin may be programmed to either show the transmitted data or not show the data by programming port A par-

allel I/O lines in the PACNT register. To cause the TXD and $\overline{RTS}$ pins to simply remain high in NMSI1, NMSI2, and NMSI3 modes, use this loopback mode in conjunction with setting the SDIAG1–SDIAG0 bits in the SIMODE register to loopback control.



NOTE: A "frame" includes opening and closing flags in HDLC and SYNCs in BISYNC and DDCMP.

Figure 4-13. Output Delays from RTS Low, Synchronous Protocol



NOTE: A "frame" includes opening and closing flags, SYNCs, etc.

Figure 4-14. Output Delays from $\overline{CTS}$ Low, Synchronous Protocol

If an internal loopback is desired when this SCC is configured to a multiplexed physical interface, then only the SDIAG1–SDIAG0 bits need be set.
When using loopback mode, the clock source for the transmitter and the receiver (as set in the TCS and RCS bits in the SCON register), must be the same. Thus,

for an internal clock, TCS and RCS may both be zero, or, for an external clock, they may both be one. The other two combinations are not allowed in this mode.

NOTE

If external loopback is desired (i.e., external to the MC68302), then the DIAG1–DIAG0 bits should be set for either normal or software operation, and an external connection should be made between the TXD and RXD pins. Clocks may be generated internally, externally, or an internally generated TCLK may be externally connected to RCLK. If software operation is used, the $\overline{\text{RTS}}$, $\overline{\text{CD}}$, and $\overline{\text{CTS}}$ pins need not be externally connected. If normal operation is used, the $\overline{\text{RTS}}$ pin may be externally connected to the $\overline{\text{CD}}$ pin, and the $\overline{\text{CTS}}$ pin may be grounded.

NOTE

Do not use this mode for loopback operation of IDL in the Serial Interface. Instead program the diag bits to Normal Operation, and (1) assert the L1GR pin externally from the S/T chip, or (2) configure the SDIAG1-0 bits in the SIMODE to Internal Loopback or Loopback Control.

10 = Automatic echo

In this mode, the channel automatically retransmits the received data on a bit-by-bit basis. The receiver operates normally, but the transmitter simply retransmits the received data. The $\overline{\text{CD}}$ pin must be asserted for the receiver to receive data, and the $\overline{\text{CTS}}$ line is ignored. The data is echoed out the TXD pin with a few nanosecond delay from RXD. No transmit clock is required, and the ENT bit in the SCC mode register does not have to be set.

NOTE

The echo function may also be accomplished in software by receiving buffers from an SCC, linking them to transmit buffer descriptors, and then transmitting them back out of that SCC.

11 = Software operation (CTS, CD lines under software control)

In this mode, the CTS and CD lines are just inputs to the SCC event (SCCE)and status (SCCS) registers. The SCC controller does not use these lines to enable/disable reception and transmission, but leaves low (i.e., active) in this mode. Transmission delays from RTS low are zero TCLKs (asynchronous protocols) or one TCLK (synchronous protocols).

NOTE

The MC68302 provides several tools for enabling and disabling transmission and/or reception. Choosing the right tool is application and situation dependent. For the receiver, the tools are 1) the empty bit in the receive buffer descriptor, 2) the ENR bit, and 3) the ENTER HUNT MODE command. For the transmitter, the

tools are 1) the ready bit in the transmit buffer descriptor, 2) the ENT bit, 3) the STOP TRANSMIT command, 4) the RESTART TRANSMIT command, and 5) the FRZ bit in the SCM (UART mode only).

ENR—Enable Receiver

When ENR is set, the receiver is enabled. When it is cleared, the receiver is disabled, and any data in the receive FIFO is lost. If ENR is cleared during data reception, the receiver aborts the current character. ENR may be set or cleared regardless of whether serial clocks are present. To restart reception, the ENTER HUNT MODE command should be issued before ENR is set again.

ENT—Enable Transmitter

When ENT is set, the transmitter is enabled; when ENT is cleared, the transmitter is disabled. If ENT is cleared, the transmitter will abort any data transmission, clear the transmit data FIFO and shift register, and force the TXD line high (idle). Data already in the transmit shift register will not be transmitted. ENT may be set or cleared regardless of whether serial clocks are present.

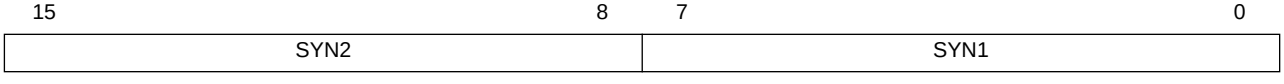
The STOP TRANSMIT command additionally aborts the current frame and would normally be given to the channel before clearing ENT. The command does not clear ENT automatically. In a similar manner, to restart transmission, the user should issue the RESTART TRANSMIT command and then set ENT. The command register is described in 4.3 Command Set. The specific actions taken with each command vary somewhat according to protocol and are discussed in each protocol section.

MODE1–MODE0—Channel Mode

00 = HDLC
01 = Asynchronous (UART and DDCMP)
10 = Synchronous DDCMP and V.110
11 = BISYNC and Promiscuous Transparent

4.5.4 SCC Data Synchronization Register (DSR)

Each DSR is a 16-bit, memory-mapped, read-write register. DSR specifies the pattern used in the frame synchronization procedure of the SCC in the synchronous protocols. In the UART protocol it is used to configure fractional stop bit transmission. After reset, the DSR defaults to $7E7E (two FLAGs); thus, no additional programming is necessary for the HDLC protocol. For BISYNC, DDCMP, and V.110, the contents of the DSR should be written before the channel is enabled. Note that for the DDCMP, SYN1 must equal SYN2 must equal DSYN1 for proper operation.



NOTE

The DSR register has no relationship to the RS-232 signal “data set ready,” which is also abbreviated DSR.

4.5.5 Buffer Descriptors Table

Data associated with each SCC channel is stored in buffers. Each buffer is referenced by a buffer descriptor (BD). BDs are located in each channel's BD table (located in dual-port RAM). There are two such tables for each SCC channel: one is used for data received from the serial line; the other is used to transmit data. The actual buffers may reside in either external memory or internal memory (dual-port RAM). For internal memory data buffers, the data buffer pointer is in the low-order data pointer word and is an offset from the device base address to any available area in the dual-port RAM. (Data buffers may reside in the parameter RAM of an SCC if it is not enabled).

The BD table allows the user to define up to eight buffers for the transmit channel and up to eight buffers for the receive channel (Figure 4-15). Each BD table forms a circular queue. The format of the BDs is the same for each SCC mode of operation (HDLC, UART, DDCMP, BISYNC, V.110, and transparent) and for both transmit or receive. Only the first field (containing status and control bits) differs for each protocol. The BD format is shown in Figure 4-16.

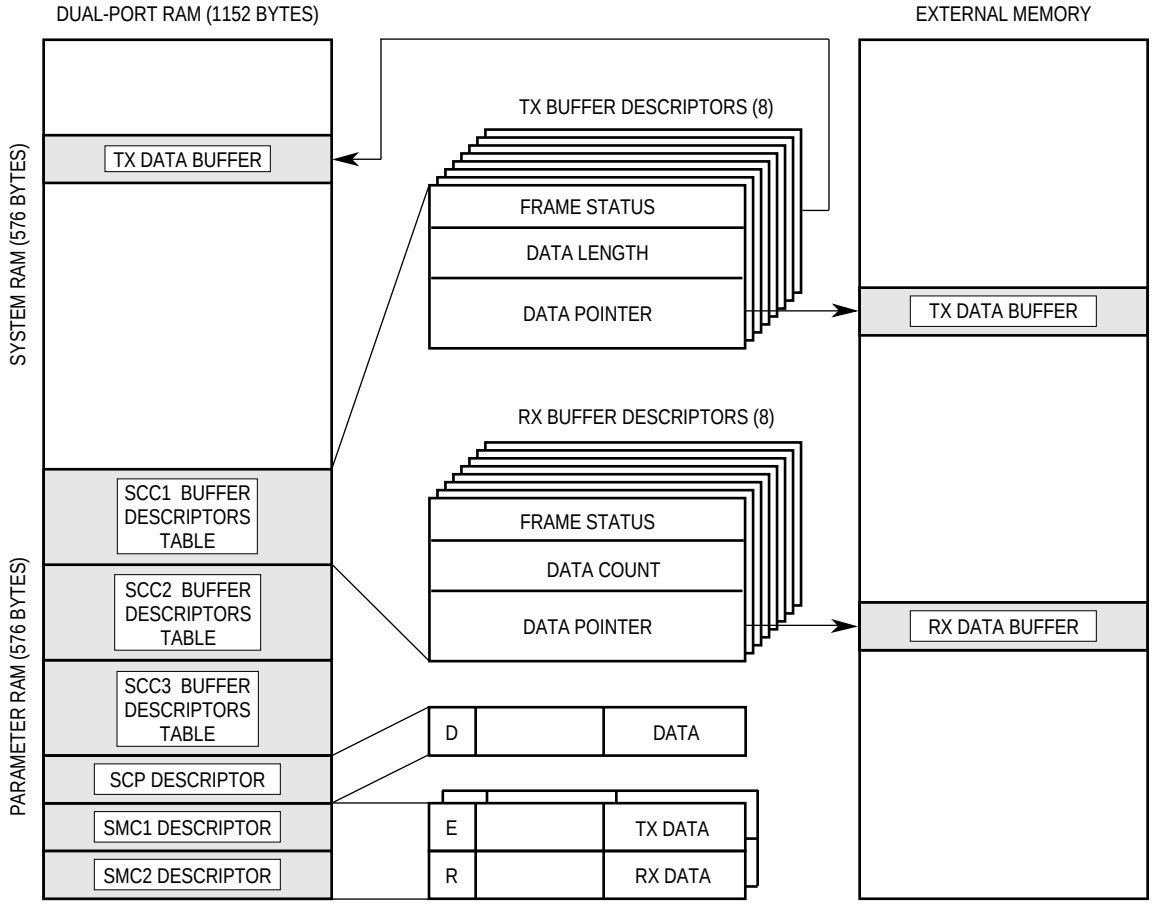


Figure 4-15. Memory Structure

	15	0
OFFSET + 0	STATUS AND CONTROL	
OFFSET + 2	DATA LENGTH	
OFFSET + 4	HIGH-ORDER DATA BUFFER POINTER (only lower 8 bits used, upper 8 bits must be 0)	
OFFSET + 6	LOW-ORDER DATA BUFFER POINTER	

Figure 4-16. SCC Buffer Descriptor Format

For frame-oriented protocols (HDLC, BISYNC, DDCMP, V.110), a frame may reside in as many buffers as are necessary (transmit or receive). Each buffer has a maximum length of 64K–1 bytes. The CP does not assume that all buffers of a single frame are currently linked to the BD table, but does assume that the unlinked buffers will be provided by the processor in time to be either transmitted or received. Failure to do so will result in a TXE error being reported by the CP.

For example, assume the first six buffers of the transmit BD table have been transmitted and await processing by the M68000 core (with all eight buffers used in the circular queue), and a three-buffer frame awaits transmission. The first two buffers may be linked to the remaining two entries in the table as long as the user links the final buffer into the first entry in the BD table before the IMP attempts its transmission. If the final buffer is not linked in time to the BD table by the time the CP attempts its transmission, the CP will report an underrun error.

Buffers allocated to an SCC channel may be located in either internal or external memory. Memory allocation occurs for each BD individually. If internal memory is selected, the CP uses only the lower 11 address bits (A10–A0) as an offset to the internal dual-port RAM. Accesses to the internal memory by the CP are one clock cycle long and occur without arbitration. If external memory is selected, the pointers to the data buffers are used by the CP as 24 bits of address.

Extra caution should be used if function codes are included in the decoding of the external buffer address (e.g., in the on-chip chip select logic). The function code of this SCC channel must be set before external buffers can be accessed; it can then be changed only when the user is sure that the CP is not currently accessing external buffers for that channel. There are six separate function code registers located in the parameter RAM for the three SCC channels: three for receive data buffers (RFCR) and three for transmit data buffers (TFCR).

NOTE

The RFCR and TFCR function codes should never be initialized to “111.”

The CP processes the transmit BDs in a straightforward fashion. Once the transmit side of an SCC is enabled, it starts with the first BD in that SCC's transmit BD table, periodically checking a bit to see if that BD is "ready". Once it is ready, it will process that BD, reading a word at a time from its associated buffer, doing certain required protocol processing on the data, and moving resultant data to the SCC transmit FIFO. When the first buffer has been processed, the CP moves on to the next BD, again waiting for that BD's “ready” bit to be set. Thus, the CP does no look-ahead BD processing, nor does it skip over BDs that are not ready. When the CP sees the “wrap” bit set in a BD, it goes back to the beginning of the BD

table, after processing of this BD is complete. After using a BD, the CP sets the “ready” bit to not-ready; thus, the CP will never use a BD twice until the BD has been confirmed by the M68000 core.

The CP uses the receive BDs in a similar fashion. Once the receive side of an SCC is enabled, it starts with the first BD in that SCC's receive BD table. Once data arrives from the serial line into the SCC, the CP performs certain required protocol processing on the data and moves the resultant data (either bytes or words at a time depending on the protocol) to the buffer pointed to by the first BD. Use of a BD is complete when there is no more room left in the buffer or when certain events occur, such as detection of an error or an end-of-frame. Whatever the reason, the buffer is then said to be “closed,” and additional data will be stored using the next BD. Whenever the CP needs to begin using a BD because new data is arriving, it will check the “empty” bit of that BD. If the current BD is not empty, it will report a” busy” error. However, it will not move from the current BD until it becomes empty. When the CP sees the “wrap” bit set in a BD, it goes back to the beginning of the BD table, after use of this BD is complete. After using a BD, the CP sets the “empty” bit to not-empty; thus, the CP will never use a BD twice until the BD has been “processed” by the M68000 core.

In general, each SCC has eight transmit BDs and eight receive BDs. However, it is possible in one special case to assign up to 16 receive BDs at the expense of all transmit BDs. Since the transmit BDs directly follow the receive BDs in the memory map for each SCC, if an SCC is configured exclusively for half-duplex reception, it is possible to have up to 16 receive BDs available for that SCC.

If the DRAM refresh unit is used, SCC2 has six transmit BDs rather than the normal eight. SCC3 normally only has four transmit BDs. However, it is actually possible to regain additional Tx BDs for SCC3 as follows. The Tx BD table may be extended by two BDs to six BDs if the SMCs are not used. Additionally, all eight Tx BDs for SCC3 may be used if the following is considered: 1) the SCP and SMCs must not be used; 2) various words within the last two BDs will be changed by the CP during the initialization routine following any reset; and 3) the BERR channel number value will be written into the last BD after any SDMA bus error (see 4.5.8.4 Bus Error on SDMA Access), but this is not a major concern since the CP must be reset after any SDMA bus error.

4.5.6 SCC Parameter RAM Memory Map

Each SCC maintains a section in the dual-port RAM called the parameter RAM. Each SCC parameter RAM area begins at offset $80 from each SCC base area ($400, $500, or $600) and continues through offset $BF. Refer to Table 2-8 for the placement of the three SCC parameter RAM areas. Part of each SCC parameter RAM (offset $80–$9A), which is identical for each protocol chosen, is shown in Table 4-6. Offsets $9C–$BF comprise the protocol-specific portion of the SCC parameter RAM and are discussed relative to the particular protocol chosen.

Table 4-6. SCC Parameter RAM Memory Map

Address	Name	Width	Description
SCC Base + 80 #	RFCR	Byte	Rx Function Code
SCC Base + 81 #	TFCR	Byte	Tx Function Code
SCC Base + 82 #	MRBLR	Word	Maximum Rx Buffer Length
SCC Base + 84 ##		Word	Rx Internal State
SCC Base + 86 ##		Byte	Reserved
SCC Base + 87 ##	RBD#	Byte	Rx Internal Buffer Number
SCC Base + 88		2 Words	Rx Internal Data Pointer
SCC Base + 8C		Word	Rx Internal Byte Count
SCC Base + 8E		Word	Rx Temp
SCC Base + 90 ##		Word	Tx Internal State
SCC Base + 92 ##		Byte	Reserved
SCC Base + 93 ##	TBD#	Byte	Tx Internal Buffer Number
SCC Base + 94		2 Words	Tx Internal Data Pointer
SCC Base + 98		Word	Tx Internal Byte Count
SCC Base + 9A		Word	Tx Temp
SCC Base + 9C SCC Base + BF			First Word of Protocol-Specific Area Last Word of Protocol-Specific Area

Initialized by the user (M68000 core).

Modified by the CP following a CP or system reset.

Certain parameter RAM values need to be initialized by the user before the SCC is enabled. Those values not so designated are initialized/written by the CP. Once initialized, most parameter RAM values will not need to be accessed in user software since most of the activity is centered around the transmit and receive buffer descriptors, not the parameter RAM. However, if the parameter RAM is accessed by the user, the following should be noted. The parameter RAM can be read at any time. The parameter RAM values related to the SCC transmitted can only be written 1) whenever the ENT bit in the SCM is zero or 2) after a STOP TRANSMIT command and before a RESTART TRANSMIT command. The parameter RAM values related to the SCC receiver can only be written 1) whenever the ENR bit in the SCM is zero or 2) if the receiver has previously been enabled, after the ENTER HUNT MODE command and before the ENR bit is set. See 4.5.10 Disabling the SCCs for a discussion of when the SCC registers may be changed.

The registers (see Table 4-6) that typically need to be accessed by the user are described in the following paragraphs.

4.5.6.1 Data Buffer Function Code Register (TFCR, RFCR)

This register defines the address space of the receive (RFCR) and transmit (TFCR) data buffers. These registers must be initialized if the SCC is used.

NOTE

The value of the function code register for any channel may be equal to that of any other, but do not initialize FC2–FC0 with the value “111” which causes a conflict with the interrupt acknowledge cycle to occur.

7	6	5	4	3	2	1	0
0	FC2	FC1	FC0	0	0	0	0

4.5.6.2 Maximum Receive Buffer Length Register (MRBLR)

Each SCC has one MRBLR that is used to define the receive buffer length for that SCC. The MRBLR defines the maximum number of bytes that the IMP will write to a receive buffer on that SCC before moving to the next buffer. The IMP may write fewer bytes to the buffer than MRBLR if a condition such as an error or end of frame occurs, but it will never write more bytes than the MRBLR value. Thus, buffers supplied by the user for use by the IMP should always be of size MRBLR (or greater) in length.

The transmit buffers for an SCC are not affected in any way by the value programmed into MRBLR. Transmit buffers may be individually chosen to have varying lengths, as needed. The number of bytes to be transmitted is chosen by programming the data length field in the Tx BD.

NOTE

MRBLR was not intended to be changed dynamically while an SCC is operating. However, if it is modified in a single bus cycle with one 16-bit move (NOT two 8-bit back-to-back bus cycles), then a dynamic change in receive buffer length can be successfully achieved, which occurs when the CP moves control to the next Rx BD in the table. Thus, a change to MRBLR will not have an immediate effect. To guarantee the exact Rx BD on which the change will occur, the user should change MRBLR only while the SCC receiver is disabled (see 4.5.6 SCC Parameter RAM Memory Map).

NOTE

The MRBLR value should be greater than zero in all modes. In the HDLC and transparent modes, the MRBLR should have an even value.

4.5.6.3 Receiver Buffer Descriptor Number (RBD#)

The RBD# for each SCC channel defines the next BD to which the receiver will move data when it is in the IDLE state or defines the current BD during frame processing. The RBD# is the BD offset from the SCC base in the Rx BD table. For Rx BD 0, RBD# = $00; for Rx BD 1, RBD# = $08, etc. Upon reset, the CP main controller sets this register to zero. The user can change this register only after the ENR bit is clear and after the ENTER HUNT MODE command has been issued. In most applications, this parameter will never need to be modified by the user.

4.5.6.4 Transmit Buffer Descriptor Number (TBD#)

The TBD# for each SCC channel defines the next BD from which the transmitter will move data when it is in the IDLE state or defines the current BD during frame transmission. The TBD# is the BD offset from the SCC base in the Tx BD table. For Tx BD 0, TBD# = $40; for Tx BD 1, TBD# = $48, etc. Upon reset, the CP main controller sets this register to $40. The user can change this register only after the STOP TRANSMIT command has been issued. In most applications, this parameter will never need to be modified by the user.

4.5.6.5 Other General Parameters

Additional parameters are listed in Table 4-2. These parameters do not need to be accessed by the user in normal operation, and are listed only because they may provide helpful information for experienced users and for debugging.

The Rx and Tx internal data pointers are updated by the SDMA channels to show the next address in the buffer to be accessed.

The Tx byte count is a down-count value that is initialized with the Tx BD data length and decremented with every byte read by the SDMA channels. The Rx byte count is a down-count value that is initialized with the MRBLR value and decremented with every byte written by the SDMA channels.

NOTE

The Rx byte count, Rx internal data pointer, and RBD# can be used to extract data out of a receive buffer before the buffer is completely full. However, the use of this technique is not recommended unless no other solution to the application requirement can be found! The IMP was specifically designed to eliminate the need for this technique by allowing a programmable receive buffer size for each SCC, by closing buffers immediately upon an error, and by closing a receive buffer after a user-programmable line idle period in the case of UART mode. Having considered these capabilities, a user desiring to extract data from a partially full data buffer should note the following cautions:

1. The Rx byte count and Rx internal data pointer may not be valid before the first byte has been written to the buffer or after the last byte has been written to the buffer.

2. The parameters, Rx byte count, Rx internal data pointer, and RBD#, are not updated simultaneously.

3. The RBD# and the empty bit of the Rx BD are not updated simultaneously.

The Rx internal state, Tx internal state, Rx temp, Tx temp, and reserved areas are for RISC use only.

4.5.7 SCC Initialization

The SCCs require a number of registers and parameters to be configured after a power-on reset. The following is a proper sequence for initializing the SCCs, regardless of the protocol used.

1. If SCC2 or SCC3 is used, write the parallel port A and B control registers (PACNT and PBCNT) to configure pins as parallel I/O lines or peripheral functions as needed (see 3.3 Parallel I/O Ports).
2. Write SIMODE to configure the serial channels physical interface for the three SCCs

(i.e., NMSI, PCM, GCI, IDL modes). If IDL or GCI is chosen in SIMODE, write SIMASK in the serial channels physical interface (see 4.4.5 Serial Interface Registers).

3. Write SCON (see 4.5.2 SCC Configuration Register (SCON)).
4. Write SCM (SCC Mode) but do not set the ENT or ENR bits yet (see 4.5.3 SCC Mode Register (SCM)).
5. Write DSR as required if a protocol other than HDLC is used (see specific protocol section).
6. Initialize the required values in the general-purpose parameter RAM (see 4.5.6 SCC Parameter RAM Memory Map).
7. Initialize the required values in the protocol-specific parameter RAM (see specific protocol section).
8. Clear out any current events in SCCE, if desired (see specific protocol section).
9. Write SCCM to enable the interrupts in SCCE that should reach the interrupt controller (see specific protocol section).
10. Write IMR in the interrupt controller to enable the SCC interrupt to the interrupt controller (see 3.2.5.3 Interrupt Mask Register (IMR)).
11. Set the ENR and/or ENT bits in SCM (see 4.4.3 PCM Highway Mode).

The buffer descriptors may have their ready/empty bits set at any time. Notice that the command register (CR) does not need to be accessed following power-on reset. An SCC should be disabled and re-enabled (see 4.5.10 Disabling the SCCs) after any dynamic change in its parallel I/O ports or serial channels physical interface configuration. A full reset using the RST bit in the CR is a comprehensive reset that may also be used.

4.5.8 Interrupt Mechanism

Interrupt handling for each of the SCC channels is configured on a global per-channel basis in the interrupt pending register (IPR), the interrupt mask register (IMR), and the interrupt in-service register (ISR). Within each of these registers, one bit is used to either mask or report the presence of a pending or in-service interrupt in an SCC channel. However, an SCC interrupt may be caused by a number of events. To allow interrupt handling for SCC-specific events, further registers are provided within the SCCs.

Up to eight events can cause the SCC to interrupt the processor. The events differ in accordance with the SCC protocol chosen. The events are handled independently for each channel by the SCC event register (SCCE) and the SCC mask register (SCCM). All unmasked event bits must be cleared in order for the corresponding IPR bit to be cleared. The interrupt handler typically reads the event register, and then immediately clears those bits that it will deal with during the interrupt handler.

4.5.8.1 SCC Event Register (SCCE)

This 8-bit register is used to report events recognized by any of the SCCs. On recognition of an event, the SCC will set its corresponding bit in the SCC event register (regardless of the corresponding mask bit in the SCC mask register). The SCC event register is a memory-mapped register that may be read at any time. A bit is cleared by writing a one (writing a zero does not affect a bit's value).

NOTE

Bit manipulation instructions such as BSET should not be used to clear bits in the event register because any bits that were set will be written back as ones (thus clearing all pending interrupts) as well as the desired bit.

More than one bit may be cleared at a time. This register is cleared at reset (total system reset, CP reset, or the M68000 RESET instruction).

4.5.8.2 SCC Mask Register (SCCM)

This 8-bit read-write register allows enabling or disabling interrupt generation by the CP for specific events in each SCC channel. An interrupt will only be generated if the SCC interrupts for this channel are enabled in the IMR in the interrupt controller.

If a bit in the SCC mask register is zero, the CP will not proceed with its usual interrupt handling whenever that event occurs. Any time a bit in the SCC mask register is set, a one in the corresponding bit in the SCC event register will cause the SCC event bit in the IPR to be set.

The bit locations in the SCC mask register are identical to those in the SCC event register. SCCM is cleared upon reset.

4.5.8.3 SCC Status Register (SCCs)

Each SCCS reflects line status for that SCC. It is used primarily in the NMSI physical interface mode to read the current status of the $\overline{\text{CTS}}$ pin, the $\overline{\text{CD}}$ pin, and idle status of the RXD pin. This 8-bit read-only register may be read at any time.

The $\overline{\text{CTS}}$ status indication in the SCCS is not valid until after the SCC transmitter is enabled (ENT bit is set). After this, the $\overline{\text{CTS}}$ indication will only be updated in the SCCS when any change in its condition is sampled by a rising edge of TCLK. The $\overline{\text{CD}}$ and ID status indications are not valid until the SCC receiver is enabled (ENR bit is set). After this, the CD and ID indications will only be updated in the SCCS when any change in their condition is sampled by a rising edge of RCLK.

NOTE

Since the RISC controller is involved in the update process, a slight delay between the external line condition change and the update of the SCCS is induced.

Beyond what the SCCS provides, in the BDs for each protocol, indications are given as to whether the status of these signals has changed during the reception/transmission of a given buffer. Furthermore, in the event registers (SCCE) for each protocol, a maskable interrupt bit is provided to allow the detection of any change in signal status.

NOTE

After power-on reset, when the SCC is enabled for the first time, the SCCE register will show that a change of status occurred, re-

gardless of what happens externally. This signifies that the corresponding SCCS bit is now valid.

7	3	2	1	0
RESERVED		ID	CD	CTS

Bits 7–3—Reserved for future use.

ID—Idle Status on the Receiver Line

This bit is meaningful only if the SCC is programmed to HDLC or UART mode. In HDLC mode, this bit is a one after 15 continuous ones are received on the line. This bit will be zero after a single zero occurs on the line. If flags, rather than idles, are received between frames, the ID bit will remain zero between frames.

In UART mode, this bit is a one after one idle character (9 to 13 bits) is received and is a zero after a single zero occurs on the line (e.g., a start bit).

If the DIAG1–DIAG0 bits in the SCM are programmed to normal mode, then the $\overline{CD}$ signal is an enable signal for ID status. In this case, if $\overline{CD}$ is not asserted, the ID bit will always be one, regardless of the activity on the line.

If the DIAG1–DIAG0 bits in the SCM are programmed to software operation mode, then the ID bit will always reflect line activity, regardless of the state of the $\overline{CD}$ pin.

The ID bit is valid in both the multiplexed and nonmultiplexed modes, once the ENR bit is set.

$\overline{CD}$—Carrier Detect Status on the Channel Pin

This bit has the same polarity as the external pin. In the multiplexed modes, it is always zero. $\overline{CD}$ is undefined until ENR is set.

$\overline{CTS}$ —Clear-to-Send Status on the Channel Pin.

This bit has the same polarity as the external pin. In the PCM highway mode, it is always zero. In the GCI and IDL mode, if the SCC is connected to the D channel, then this bit is valid; otherwise, it is always zero. $\overline{CTS}$ is undefined until the ENT bit is set.

When the $\overline{CTS}$ and $\overline{CD}$ lines are programmed to software control in the SCC mode register, these lines do not affect the SCC and can be used for other purposes such as a data set ready (DSR), a data terminal ready (DTR) line, or an interrupt source in the SCCE register according to the behavior just described.

4.5.8.4 Bus Error on SDMA Access

When a bus error occurs on an access by the SDMA channel, the CP generates a unique interrupt (see 3.2 Interrupt Controller). The interrupt service routine should read the bus error channel number from the parameter RAM at BASE + 67C as follows:

0—SCC1 Tx Channel

1—SCC1 Rx Channel or DRAM Refresh Cycle

2—SCC2 Tx Channel

3—SCC2 Rx Channel

4—SCC3 Tx Channel

5—SCC3 Rx Channel

Next, the pointer that caused the bus error can be determined by reading the Rx or Tx internal data pointer from the parameter's memory map of the particular SCC. Following this bus error, the CP must be reset with a hardware reset or the setting of the RST bit in the command register.

An SDMA retry cycle is not indicated with any status information or interrupts.

4.5.9 SCC Transparent Mode

The SCC supports several transparent receive and transmit modes, which vary according to the protocol being implemented. As used here, transparent means data transparency and is not the same as the totally transparent (promiscuous) mode described later in 4.5.16 Transparent Controller. The transparent modes for each protocol are as follows:

UART Mode

If the normal mode is selected (UM1–UM0 = 00 in the UART mode register) and the control characters table is empty, then the UART will transfer all received characters to memory except break and idle characters.

HDLC Mode

When the HDLC mask register = $0000, the HDLC controller will transfer all received frames to the receive buffers, including address, control, data, and CRC. The HDLC controller still performs the standard HDLC functions of zero insertion/deletion and flag recognition.

BISYNC Mode

The BISYNC mode may be used to transparently receive data that is delimited by SYNCs. To do this, the BISYNC control characters table should be empty, and SYNC/DLE stripping should be disabled. Thereafter, all data following the SYNCs will be received into the receive buffers.

There are two possible ways of recognizing SYNCs. Either the EXSYN bit in the BISYNC mode register is cleared and a SYNC is defined by the SYN1–SYN2 characters in DSR, or the EXSYN bit is set and a SYNC is determined by an external SYNC pulse.

When the BISYNC control characters table is empty, the SYNC and DLE stripping is disabled, and the BISYNC controller is in normal nontransparent mode (RTR = 0 in BISYNC mode register). In this case, the configuration is as follows:

- If EXSYN in the BISYNC mode register is cleared, then the BISYNC controller transfers all characters that follow the SYN1–SYN2 sequence to the receive buffers.
- If EXSYN in the BISYNC mode register is set, then the BISYNC controller transfers all characters that follow the external SYNC pulse to the receive buffers.

NOTE

The BISYNC controller can reverse the bit order in both modes.

Totally Transparent (Promiscuous) Mode

The MC68302 can both receive and transmit the entire serial bit stream transparently. See 4.5.16 Transparent Controller for details.

4.5.10 Disabling the SCCs

If an SCC transmitter or receiver is not needed for a period of time or a mode change is required, then it may be disabled and re-enabled later. In this case, a sequence of operations is followed.

For the SCC transmitter, the sequence is as follows:

STOP TRANSMIT Command

Wait for the FIFO to empty

Clear ENT

-
- (The SCC transmitter is now disabled)
-

RESTART TRANSMIT Command

Set ENT

For the SCC receiver, the sequence is as follows:

Clear ENR

-
- (The SCC receiver is now disabled)
-

ENTER HUNT MODE Command

Set ENR

This sequence assures that any buffers in use will be properly closed and that new data will be transferred to/from a new buffer.

While an SCC is disabled (and only while an SCC is disabled) the SCON and SCM registers may be modified. Thus, once disabled, changes such as the SCC protocol, diagnostic mode, or baud rate may be made. Such parameters cannot be modified “on-the-fly.” The DSR should also only be modified while an SCC is disabled, although an exception exists to this in the UART mode concerning the transmission of partial stop bits.

The TBD# and RBD# values in the parameter RAM are not reset by the disabling process; thus, the very next BDs will be used when the SCC is re-enabled. A full software reset of the entire CP including the three SCCs is accomplished in the command register (CR). To reset an SCC to its initial state, the RX internal state, the TX internal state, the TBD#, and the RBD# can be written to their values after reset. The user can read these values for each SCC after a reset.

The SCC should be disabled and re-enabled if any change is made to the SCC's parallel I/O or serial channels physical interface configuration. The SCC does not need to be disabled if only a change to a parameter RAM value is made. See 4.5.6 SCC Parameter RAM Memory Map for a discussion of when parameter RAM values may be modified.

To save power, the SCCs may simply be disabled. Clearing the enable transmitter (ENT) bit in the SCC mode register causes the SCC transmitter to consume the least possible power; clearing the ENR bit causes a similar action for the SCC receiver.

The above statement on saving power is independent of a decision to use the low-power modes (see 3.8 System Control). If a low-power mode is desired, the SCC may be disabled before entering the low-power mode, or it may be left enabled so that an SCC interrupt may bring the IMP out of the low-power mode. One common use of the low-power mode is to disable the transmitter but leave the receiver enabled (i.e., in the hunt mode) so that an arriving frame destined for this station will cause an interrupt, waking the IMP from its low-power mode.

The low-power mode affects the M68000 core, not the SCCs. Since the SCCs are usually clocked at a far lower rate than the M68000 core, significant power savings may still be achieved with the SCCs fully enabled and the M68000 core in the low-power mode.

4.5.11 UART Controller

Many applications need a simple method of communicating low-speed data between equipment. The universal asynchronous receiver transmitter (UART) protocol is the *de-facto* standard for such communications. The term asynchronous is used because it is not necessary to send clocking information with the data that is sent. UART links are typically 38400 baud or less in speed and are character oriented (i.e., the smallest unit of data that can be correctly received or transmitted is a character). Typical applications of asynchronous links are connections between terminals and computer equipment. Even in applications where synchronous communication is required, the UART is often used for a local debugging port to run board debugger software. The character format of the UART protocol is shown in Figure 4-17.

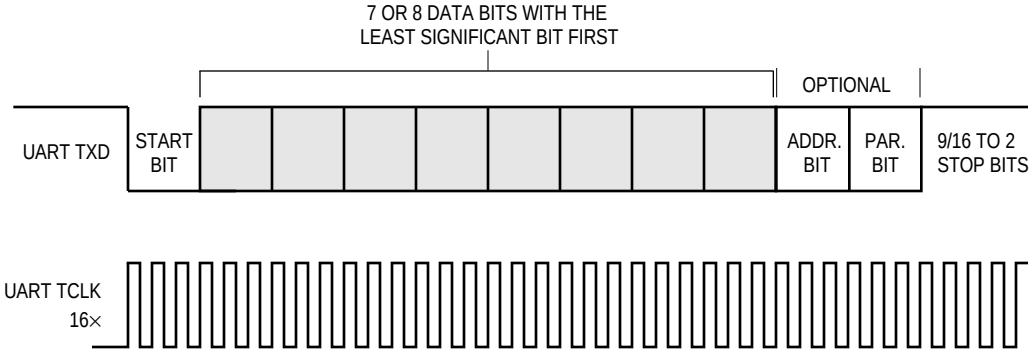


Figure 4-17. UART Frame Format

Since the transmitter and receiver work asynchronously, there is no need to connect transmit and receive clocks. Instead, the receiver over-samples the incoming data stream by a factor of 16 and uses some of these samples to determine the bit value. Traditionally, the middle three of the 16 samples are used. Two UARTs can communicate using a system like this if parameters such as the parity scheme and character length are the same for both transmitter and receiver.

When data is not transmitted in the UART protocol, a continuous stream of ones is transmitted. This is called the idle condition. Since the start bit is always a zero, the receiver can detect when real data is once again present on the line. The UART also specifies an all-zeros character, called a break, which is used to abort a character transfer sequence.

Many different protocols have been defined that use asynchronous characters, but the most popular of these is the RS-232 standard. RS-232 specifies standard baud rates, handshaking protocols, and mechanical/electrical details. Another popular standard using the same character format is RS-485, which defines a balanced line system allowing longer cables than RS-232 links. Synchronous protocols like HDLC or DDCMP are sometimes defined to run over asynchronous links. Other protocols like PROFIBUS (see Appendix C RISC Microcode from RAM) extend the UART protocol to include LAN-oriented features such as token passing.

All the standards provide handshaking signals, but some systems require just three physical lines: transmit data, receive data, and ground.

Many proprietary standards have been built around the asynchronous character frame, and some even implement a multidrop configuration. In multidrop systems, more than two stations may be present on a network, with each station having a specific address. Frames composed of many characters may be broadcast, with the first character acting as a destination address. To allow this procedure, the UART frame is extended by one bit to distinguish between an address character and the normal data characters.

By appropriately setting the SCC mode register, any of the SCC channels may be configured to function as a UART. The UART controller provides standard serial I/O using asynchronous character-oriented (start-stop) protocols. The UART may be used to communicate with other existing UART devices. Also, in conjunction with another SCC channel, it may be used in either ISDN terminal adaptor or X.25 packet assembly and disassembly (PAD) applications.

The UART provides a port for serial communication to other microprocessors, terminals, etc., either locally or through modems. It includes facilities for communication using standard asynchronous bit rates and protocols. The UART supports a multidrop mode for master/slave operation with wakeup capability on either an idle line or an address bit.

The UART uses a seven-pin interface in NMSI mode. It transmits data from memory (internal or external) to the TXD line and receives data from the RXD line into memory. The seven dedicated serial interface pins are transmit data (TXD), receive data (RXD), receive clock (RCLK), transmit clock (TCLK), carrier detect ($\overline{CD}$), clear to send ($\overline{CTS}$), and request to send

($\overline{\text{RTS}}$). Other modem lines such as data set ready (DSR) and data terminal ready (DTR) can be supported through the parallel I/O pins.

The UART consists of separate transmit and receive sections whose operations are asynchronous with the M68000 core and may be either synchronous or asynchronous with respect to each other. Each clock can be supplied either from the baud rate generator or from the external pins.

The UART key features are as follows:

- Flexible Message-Oriented Data Buffers
- Multidrop Operation
- Receiver Wakeup on IDLE Line or Address Mode
- Eight Control Character Comparison Registers
- Two Address Comparison Registers
- Four 16-Bit Error Counters
- Programmable Data Length (7 or 8 Bits)
- Programmable 1 or 2 Stop Bits with Fractional Stop Bits
- Even/Odd/Force/No Parity Generation
- Even/Odd/No Parity Check
- Frame Error, Noise Error, Break, and IDLE Detection
- Transmits Preamble and Break Sequences
- Freeze Transmission Option
- Maintenance of Four 16-Bit Error Counters
- Provides Asynchronous Link for DDCMP Use
- Flow Control Character Transmission Supported

4.5.11.1 Normal Asynchronous Mode

In the normal asynchronous mode, the receive shift register receives the incoming data on the RXD pin. The length and the format of the serial word in bits are defined by the control bits in the UART mode register. The order of reception is as follows:

Start Bit
Seven or Eight Data Bits with the Least Significant Bit First
Address/Data Bit (Optional)
Parity Bit (Optional)
Stop Bits

The receiver samples each bit of the incoming data three times around its center. The value of the bit is determined by the majority of those samples. If all the samples do not agree, a noise indication counter is incremented. When a complete character has been clocked in, the contents of the shift register are transferred to the UART receive data register. If there is an error in this character, then the appropriate error bits will be set by the IMP.

The UART may receive fractional stop bits. The next character's start bit may begin anytime after the 11th internal clock of the previous character's first stop bit (the UART uses a 16X clock).

The UART transmit shift register transmits the outgoing data on the TXD pin as shown in Figure 4-17. Data is clocked synchronously with the transmit clock, which may have either an internal or external source. The order of bit transmission is as stated for reception.

Only the data portion of the UART frame is actually stored in the data buffers. The start and stop bits are always generated and stripped by the UART controller. The parity bit may also be generated in the case of transmission, and checked during reception. Although parity is not stored in the data buffer, its value may be inferred by the reporting mechanism in the data buffer (i.e., characters with parity errors are identified). Similarly, the optional address bit is not stored in the transmit or receive data buffer, but is implied from the buffer descriptor itself. Parity is generated and checked for the address bit, when present.

4.5.11.2 Asynchronous DDCMP MODE

The IMP also allows the DDCMP protocol to be run over an asynchronous connection, using the UART. The description of this operation is contained in 4.5.14 DDCMP Controller. This operation uses the DDCMP buffer structures and the DDCMP-specific parameter RAM; however, the SCC mode register must be configured as a UART. The proper programming of the UART mode register to obtain asynchronous DDCMP is covered in 4.5.11.13 UART Mode Register.

4.5.11.3 UART Memory Map

When configured to operate in UART mode, the IMP overlays the structure (see Table 4-6) onto the protocol-specific area of that SCC's parameter RAM. Refer to 2.8 MC68302 Memory Map for the placement of the three SCC parameter RAM areas and to Table 4-5 for the other parameter RAM values.

Table 4-7. UART Specific Parameter RAM

Address	Name	Width	Description
SCC Base + 9C # SCC Base + 9E SCC Base + A0 #	MAX_IDL IDLC BRKCR	Word Word Word	Maximum IDLE Characters (Receive) Temporary Receive IDLE Counter Break Count Register (Transmit)
SCC Base + A2 # SCC Base + A4 # SCC Base + A6 # SCC Base + A8 #	PAREC FRMEC NOSEC BRKEC	Word Word Word Word	Receive Parity Error Counter Receive Framing Error Counter Receive Noise Counter Receive Break Condition Counter
SCC Base + AA # SCC Base + AC #	UADDR1 UADDR2	Word Word	UART ADDRESS Character 1 UART ADDRESS Character 2
SCC Base + AE SCC Base + B0 # SCC Base + B2 # SCC Base + B4 # SCC Base + B6 # SCC Base + B8 # SCC Base + BA # SCC Base + BC # SCC Base + BE #	RCCR CHARACTER1 CHARACTER2 CHARACTER3 CHARACTER4 CHARACTER5 CHARACTER6 CHARACTER7 CHARACTER8	Word Word Word Word Word Word Word Word Word	Receive Control Character Register CONTROL Character 1 CONTROL Character 2 CONTROL Character 3 CONTROL Character 4 CONTROL Character 5 CONTROL Character 6 CONTROL Character 7 CONTROL Character 8

Initialized by the user (M68000 core).

MAX_IDL

The UART controller watches the receive line, regardless of whether or not actual data is being received. If the line is idle, the UART controller counts how many idle characters have been received. An idle character is defined as 9 to 13 consecutive ones. For a given application, the number of bits in the idle character is calculated as follows:

1 + data length (either 7 or 8) + (1 if address bit used) + (1 if parity bit used) + number of stop bits (either 1 or 2)

MAX_IDL is programmed with a value from 1 (MAX_IDL = 1) to 65536 (MAX_IDL = 0).

Once a character of data is received on the line, the UART controller begins counting any idle characters received. If a MAX_IDL number of idle characters is received before the next data character is received, an idle timeout occurs, and the buffer is closed. This, in turn, can produce an interrupt request to the M68000 core to receive the data from the buffer. MAX_IDL then provides a convenient way to demarcate frames in the UART mode (see also 4.5.11.11 UART Error-Handling Procedure.

NOTE

Program MAX_IDL to $0001 for the minimum timeout value; program MAX_IDL to $0000 for the maximum timeout value.

IDLC

This value is used by the RISC to store the current idle counter value in the MAX_IDL timeout process. IDLC is a down counter. It does not need to be initialized or accessed by the user.

BRKCR

The UART controller will send a break character sequence whenever a STOP TRANSMIT command is given. The number of break characters sent by the UART controller is determined by the value in BRKCR. The length of one break character is 9 to 13 zeros depending on the configuration. The same equation applies for BRKCR as that used for MAX_IDL. See 4.5.11.8 Send Break for more details. BRKCR is programmed with a value from 0 (BRKCR = 0) to 65535 (BRKCR = 65535).

PAREC, FRMEC, NOSEC, BRKEC

These counters are initialized by the user. When the associated condition occurs, they will be incremented by the RISC controller. See 4.5.11.11 UART Error-Handling Procedure for more details.

UADDR1, UADDR2

In the multidrop mode, the UART controller can provide automatic address recognition of two addresses. In this case, the lower order byte of UADDR1 and UADDR2 are programmed by the user with the two desired addresses. See 4.5.11.6 UART Address Recognition for more details.

RCCR, CHARACTER

The UART controller can automatically recognize special characters and generate interrupts. It also allows a convenient method for inserting flow control characters into the transmit stream. See 4.5.11.7 UART Control Characters and Flow Control for more details.

If neither of these capabilities are desired, initialize CHARACTER1 to $8000 and CHARACTER8 to $0000 to disable both functions.

4.5.11.4 UART Programming Model

An SCC configured as a UART uses the same data structure as the other protocols. The UART data structure supports multibuffer operation. The UART may also be programmed to perform address comparison whereby messages not destined for a given programmable address are discarded. Also, the user can program the UART to accept or reject control characters. If a control character is rejected, an interrupt may be generated. The UART enables the user to transmit break and preamble sequences. Overrun, parity, noise, and framing errors are reported using the buffer descriptor (BD) table and/or error counters. An indication of the status of the line (idle) is reported through the status register, and a maskable interrupt is generated upon a status change.

In its simplest form, the UART can function in a character-oriented environment. Each character is transmitted with accompanying stop bits and parity (as configured by the user) and is received into separate one-byte buffers. Reception of each buffer may generate a maskable interrupt.

Many applications may want to take advantage of the message-oriented capabilities supported by the UART using linked buffers to receive or transmit data. In this case, data is handled in a message-oriented environment; users can work on entire messages rather than operating on a character-by-character basis. A message may span several linked buffers. For example, rather than being interrupted after the reception of each character, a terminal driver may want to wait until an end-of-line character has been typed by a user before handling the input data.

As another example, when transmitting ASCII files, the data may be transferred as messages ending on the end-of-line character. Each message could be both transmitted and received as a circular list of buffers without any intervention from the M68000 core. This technique achieves both ease in programming and significant savings in processor overhead.

On the receive side, the user may define up to eight control characters. Each control character may be configured to designate the end of a message (such as end of line) or to generate a maskable interrupt without being stored in the data buffer. This latter option is useful when flow-control characters such as XON or XOFF need to alert the M68000 core, yet do not belong to the message being received. Flow-control characters may also be transmitted at any time.

In the message-oriented environment, the data stream is divided into buffers. However, the physical format of each character (stop bits, parity, etc.) is not altered.

4.5.11.5 UART Command Set

These commands are issued to the command register described in 4.3 Command Set**.**

STOP TRANSMIT Command

After a hardware or software reset and the enabling of the channel by writing the SCC mode register, the channel is in the transmit enable mode and starts polling the first BD in the table approximately every eight transmit clocks.

The channel STOP TRANSMIT command disables the transmission of characters on the transmit channel. If this command is received by the UART controller during message transmission, transmission of that message is aborted. The UART completes transmission of any data already transferred to the UART FIFO (up to three characters) and then stops transmitting data. The TBD# is not advanced.

The UART transmitter will transmit a programmable number of break sequences and then start to transmit idles. The number of break sequences (which may be zero) should be written to the break count register (BRKCR) before this command is given to the UART controller.

The STOP TRANSMIT command must be issued before the SCC mode register is used to disable the transmitter if the transmitter will be reenabled at a later time.

RESTART TRANSMIT Command

The channel RESTART TRANSMIT command re-enables the transmission of characters on the transmit channel. This command is expected by the UART in three situations: after issuing a STOP TRANSMIT command, after issuing a STOP TRANSMIT and then disabling the channel using the SCC mode register, or after transmitter errors (CTS lost). The UART controller will resume transmission from the current transmitter BD number (TBD#) in the channel's Tx BD table.

If the transmitter is being re-enabled, the RESTART TRANSMIT command must be used and should be followed by the enabling of the transmitter in the SCC mode register.

ENTER HUNT MODE Command

After a hardware or software reset and the enabling of the channel by its SCC mode register, the channel is in the receive enable mode and will use the first BD in the table.

The ENTER HUNT MODE command is used to force the UART controller to abort reception of the current message, generate an RX interrupt (if enabled) as the buffer is closed, and enter the hunt mode. The UART controller will resume reception using the next BD once an address character or a single idle character is received. In multidrop hunt mode, the UART controller continually scans the input data stream for the address character. While not in multidrop mode, the UART controller will wait for a single IDLE character. In the UART mode, none of the data received in the FIFO is lost when ENTER HUNT MODE command is issued; however, this command does reset the receive FIFO in other protocols, e.g., HDLC.

If an enabled receiver has been disabled by clearing ENR in the SCC mode register, the ENTER HUNT MODE command must be given to the channel before setting ENR again. Reception will then begin with the next BD.

4.5.11.6 UART Address Recognition

In multidrop systems, more than two stations may be present on a network, with each having a specific address. Figure 4-18 shows two examples of such a configuration. Frames comprised of many characters may be broadcast, with the first character acting as a destination address. To achieve this, the UART frame is extended by one bit, called the address bit, to distinguish between an address character and the normal data characters. The UART can be configured to operate in a multidrop environment in which two modes are supported:

Automatic Multidrop Mode—The IMP automatically checks the incoming address character and accepts the data following it only if the address matches one of two 8-bit preset values. In this mode, UM1–UM0 = 11 in the UART mode register.

Nonautomatic Multidrop Mode—The IMP receives all characters. An address character is always written to a new buffer (it may be followed by data characters in the same buffer). In this mode, UM1–UM0 = 01 in the UART mode register.

Each UART controller has two 8-bit address registers (UADDR1 and UADDR2) for address recognition. In the automatic mode, the incoming address is checked against the lower order byte of the UART address registers. Upon an address match, the address match (M) bit in the BD is set/cleared to indicate which address character was matched. The data following it is written to the same data buffer.

NOTE

For 7-bit characters, the eighth bit (bit 7) in UADDR1 and UADDR2 should be zero.

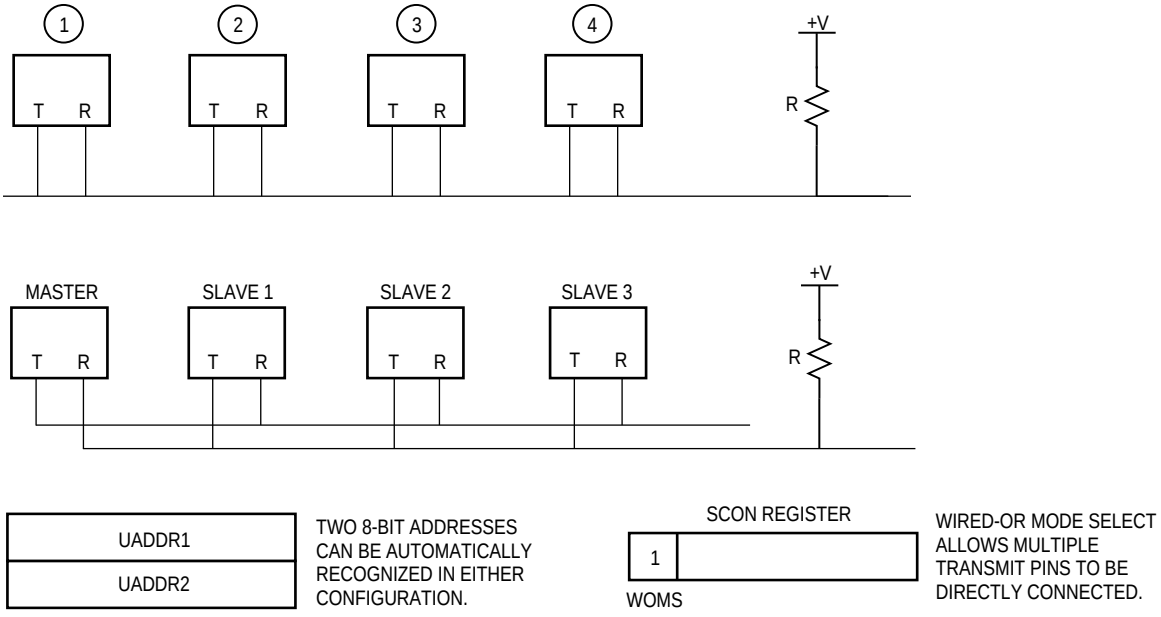


Figure 4-18. Two Configurations of UART Multidrop Operation

4.5.11.7 UART Control Characters and Flow Control

The UART has the capability to recognize special control characters. These characters may be used when the UART functions in a message-oriented environment. Up to eight control characters may be defined by the user in the control characters table. Each of these characters may be either stored (written to the receive buffer, after which the current buffer is closed and a new receive buffer taken) or rejected. If rejected, the character is written to the received control character register (RCCR) in internal RAM, and a maskable interrupt is generated. This method is useful for notifying the user of the arrival of control characters (e.g., XOFF) that are not part of the received messages.

The UART uses a table of 16-bit entries to support control-character recognition. Each entry consists of the control character, an end-of-table bit, and a reject character bit. The control characters table is shown in Figure 4-19.

NOTE

To disable all functions of the control characters and flow control table, initialize CHARACTER1 to $8000 and CHARACTER8 to $0000.

RCCR—Received Control Character Register

Upon a control character match for which the reject bit is set, the UART will write the control character into the RCCR and generate a maskable interrupt. The M68000 core must process the interrupt and read the RCCR before a second control character arrives. Failure to do so will result in the UART overwriting the first control character.

	15	14	13	12	11	10	9	8	7 … 0
OFFSET + 0									RCCR
OFFSET + 2	E	R							CHARACTER1
OFFSET + 4	E	R							CHARACTER2
OFFSET + 6	E	R							CHARACTER3
•									
•									
•									
OFFSET + 10	E	R	REA	I	CT	0	0	A	CHARACTER8

Figure 4-19. UART Control Characters Table

CHARACTER7–CHARACTER1—Control Character Value

These fields define control characters that should be compared to the incoming character. For 7-bit characters, the eighth bit (bit 7) should be zero.

E—End of Table

0 = This entry is valid. The lower eight bits will be checked against the incoming character.
1 = The entry is not valid. No valid entries lie beyond this entry.

NOTE

In tables with eight receive control characters, E is always zero.

R—Reject Character

0 = The character is not rejected but is written into the receive buffer. The buffer is then closed, and a new receive buffer is used if there is more data in the message. A maskable interrupt is generated in the RX bit of the UART event register.
1 = If this character is recognized, it will not be written to the receive buffer. Instead, it is written to the RCCR, and a maskable interrupt is generated in the CCR bit in the UART event register. The current buffer is not closed when a control character is received with R set.

Transmission of out-of-sequence characters is also supported and is normally used for the transmission of flow control characters such as XON or XOFF. This is performed using the last (eighth) entry in the UART control characters table. The UART will poll this character whenever the transmitter is enabled for UART operation: during freeze, during buffer transmission, and when no buffer is ready for transmission. The character is transmitted at a higher priority than the other characters in the transmit buffer (if any), but does not pre-empt characters already in the transmit FIFO.

CHARACTER8—Control Character Value

The eighth entry in the UART control characters table is defined as follows:

E—Empty

Must be one to use this entry as a flow control transmission character. To use this entry instead as a receive control characters entry, this E bit (and all other E bits in the table) should be zero.

R—Reject

Must be zero to use this entry as a flow control transmission character. For a receive control characters entry, it maintains its functionality as previously defined.

REA—Ready

This bit is set by the M68000 core when the character is ready for transmission and will remain one while the character is being transmitted. The CP clears this bit after transmission.

I—Interrupt

If set, the M68000 core will be interrupted when this character has been transmitted. (The TX bit will be set in the UART event register.)

CT—Clear-to-Send Lost

This status bit indicates that the $\overline{CTS}$ signal was negated during transmission of this character. If this occurs, the CTS bit in the UART event register will also be set.

NOTE

If the $\overline{CTS}$ signal was negated during transmission, and the CP transmits this character in the middle of buffer transmission, the $\overline{CTS}$ signal could actually have been negated either during this character's transmission (i.e., CHARACTER8) or during a buffer character's transmission. In this case, the CP sets the CT bit both here and in the Tx BD status word.

A—Address

When working in a multidrop configuration, the user should include the address bit in this position.

CHARACTER8—Flow Control Character Value

Any 7- or 8-bit character value may be transmitted. This value may be modified only while the REA bit is cleared. A 7-bit character should comprise bits 6–0.

4.5.11.8 Send Break

A break is an all-zeros character without stop bits—i.e., 9 to 13 continuous zeros. A break is sent by issuing the STOP TRANSMIT command. The UART completes transmission of any outstanding data in the FIFO and then sends 9 to 13 zeros (depending on the UM1–UM0, SL, PEN, and CL bits in the UART mode register). The UART transmits a programmable number of break characters according to the value of the break count register (BRKCR), and then reverts to idle or sends data if the RESTART TRANSMIT command was given before completion. Upon transmission of the entire set of break characters, the transmitter sends at least one high bit before transmitting any data to guarantee recognition of a valid start bit.

4.5.11.9 Send Preamble (IDLE)

A preamble sequence gives the programmer a convenient way of ensuring that the line goes idle before starting a new message. The preamble sequence length is 9 to 13 consecutive ones (depending on the UM1–UM0, SL, PEN, and CL bits in the UART mode register). If the preamble bit in a BD is set, the SCC will send a preamble sequence before transmitting that data buffer.

4.5.11.10 Wakeup Timer

By issuing the ENTER HUNT MODE command, the user can temporarily disable the UART receiver. It will remain inactive until an idle or address character is recognized (depending on the setting of UM1–UM0).

If the UART is still in the process of receiving a message that the user has already decided to discard, the message may be aborted by issuing the ENTER HUNT MODE command. The UART receiver will be re-enabled when the message is finished by detecting one idle

character of 9 to 13 consecutive ones (if UM1–UM0 = 00) or by the address bit of the next message (if UM0 = 1).

When the receiver is in sleep mode and a break sequence is received, the receiver will increment the BRKEC counter and generate the BRK interrupt (if enabled).

4.5.11.11 UART Error-Handling Procedure

The UART controller reports character reception and transmission error conditions through the channel BDs, the error counters, and the UART event register (SCCE). The modem interface lines can also be monitored directly by the SCC status register.

Transmission Error

Clear to Send Lost During Character Transmission. When this error occurs and the channel is not programmed to control this line with software, the channel terminates buffer transmission, closes the buffer, sets the CTS lost (CT) bit in the BD, and generates the TX interrupt (if enabled). The channel will resume transmission after the reception of the RESTART TRANSMIT command.

Reception Errors

1. Overrun Error. The UART controller maintains an internal three-byte FIFO for receiving data. The CP begins programming the SDMA channel (if the data buffer is in external memory) when the first byte is received into the FIFO. When a receiver FIFO overrun occurs, the channel writes the received character into the internal FIFO over the previously received character (the previous character and its status bits are lost). Then the channel writes the received character to the buffer, closes the buffer, sets overrun (OV) in the BD, and generates the RX interrupt (if enabled). In automatic multidrop mode, the receiver enters hunt mode immediately.
2. Carrier Detect Lost During Character Reception. When this error occurs and the channel is not programmed to control this line with software, the channel terminates character reception, closes the buffer, sets the carrier detect lost (CD) bit in the BD, and generates the RX interrupt (if enabled). This error's priority is the highest; the last character in the buffer is lost and other errors are not checked. In automatic multidrop mode, the receiver enters hunt mode immediately.
3. Framing Error. Framing error is reported by the UART controller when no stop bit is detected in a received data string. When this error occurs, the channel writes the received character to the buffer, closes the buffer, sets framing error (FR) in the BD, and generates the RX interrupt (if enabled). The channel also increments the framing error counter (FRMEC). When this error occurs, parity is not checked for this character. In automatic multidrop mode, the receiver enters hunt mode immediately.
4. Parity Error. When the parity error occurs, the channel writes the received character to the buffer, closes the buffer, sets parity error (PR) in the BD, and generates the RX interrupt (if enabled). The channel also increments the parity error counter (PAREC). In automatic multidrop mode, the receiver enters hunt mode immediately.
5. Noise Error. Noise error is detected by the UART controller when the three samples taken on every bit are not identical. When this error occurs, the channel writes the received character to the buffer and proceeds normally but increments the noise error

counter (NOSEC).

6. IDLE Sequence. Receive IDLE (preamble) is detected by the UART controller when a character with 9 to 13 consecutive ones (depending on the UM1–UM0, SL, PEN, and CL bits in the UART mode register) is received. When an IDLE sequence is received, the channel starts to count the number of IDLE sequences received. If it reaches the MAX_IDL value, the buffer is closed and an RX interrupt is generated (if enabled). The counter is reset every time a character is received.
7. BREAK Sequence. A BREAK sequence is detected by the UART receiver when a character with zero value and framing error is received. When a BREAK sequence is received, the channel will increment the BRKEC counter, close the buffer, set the BR bit (if a buffer was currently open), and generate a BRK interrupt (if enabled). Also, if the channel was in the middle of buffer processing, the buffer is closed and an RX is generated (if enabled). A long break sequence only increments the counter once.

Error Counters

The UART maintains four 16-bit (modulo–2**16) error counters for the receive portion of each UART controller. They can be initialized by the user when the channel is disabled. The counters are as follows:

PAREC—Parity Error Counter
FRMEC—Framing Error Counter
NOSEC—Noise Error Counter
BRKEC—BREAK Error Counter

4.5.11.12 Fractional Stop Bits

The UART transmitter can be programmed to transmit fractional stop bits. Three bits in the SCC data synchronization register (DSR) are used to program the length of the last stop bit transmitted. These DSR bits may be modified at any time. If two stop bits are transmitted, only the second one is affected. Idle characters are always transmitted as full-length characters. In UART mode, bits 14–12 in the DSR are now decoded as follows:

14–12 of DSR			
111	Last Transmit Stop Bit	16/16	(the default value after reset)
110	Last Transmit Stop Bit	15/16	
. . .			
001	Last Transmit Stop Bit	10/16	
000	Last Transmit Stop Bit	9/16	

The setting of the DSR in combination with the setting of the CL bit in the UART mode register causes the number of stop bits transmitted to be either 9/16 to 1 or 1-9/16 to 2 stop bits.

The UART receiver can always receive fractional stop bits. The next character's start bit may begin anytime after the 11th internal clock of the previous character's first stop bit (the UART uses a 16x clock).

4.5.11.13 UART Mode Register.

Each SCC mode register is a 16-bit, memory- mapped, read-write register that controls the SCC operation. The term UART mode register refers to the protocol-specific bits (15–6) of the SCC mode register when that SCC is configured as a UART. The read-write UART mode register is cleared by reset.

15	14	13	12	11	10	9	8	7	6	5 … 0
TPM1	TPM0	RPM	PEN	UM1	UM0	FRZ	CL	RTSM	SL	COMMON SCC MODE BITS

TPM1–TPM0—Transmitter Parity Mode

TMP1—TMP0 select the type of parity to be performed.

00 = Odd parity; always send an odd number of ones.
01 = Force low parity; always send a zero in the parity bit position.
10 = Even parity; always send an even number of ones.
11 = Force high parity; always send a one in the parity bit position.

RPM—Receiver Parity Mode

0 = Odd parity
1 = Even parity

When odd parity is selected, the receiver will count the number of ones in the data word. If the total number of ones is not an odd number, the parity bit is set to one to produce an odd number of ones. If the receiver counts an even number of ones, an error in transmission has occurred. Similarly, for even parity, an even number of ones must result from the calculation performed at both ends of the line.

PEN—Parity Enable

0 = No parity
1 = Parity is enabled for the transmitter and receiver as determined by the parity mode bits.

UM1–UM0—UART Mode 1–0

00 = Normal UART operation. Multidrop mode is disabled for point-to-point operation and an idle-line wakeup is selected. In the idle-line wakeup mode, the UART receiver is re-enabled by an idle string of 9 to 13 consecutive ones (depending on character length and parity mode).
01 = In the multidrop mode, an additional address/data bit is transmitted with each character. The multidrop asynchronous modes are compatible with the Motorola MC68681 DUART, the Motorola MC68HC11 SCI interface, and the Motorola DSP56000 SCI interface. UM0 is also used to select the wakeup mode before enabling the receiver or issuing the ENTER HUNT MODE command.
Multidrop mode is enabled and an address bit wakeup is selected. In the address bit wakeup mode, the UART receiver is re-enabled when the last data bit (the 8th or 9th) in a character is one. This configuration means that the received character is an address, which should be processed by all inactive processors. The IMP re-

ceives the address character and writes it to a new buffer. No address recognition is performed.

10 = The DDCMP protocol is implemented over the asynchronous channel.

11 = Multidrop mode is enabled as in the 01 case, and the IMP automatically checks the address of the incoming address character and either accepts or discards the data following the address.

FRZ—Freeze Transmission

This bit allows the user to halt the UART transmitter and to continue transmission from the next character in the buffer at a later time.

0 = Normal operation (or resume transmission after FRZ is set).

1 = The UART completes transmission of any data already transferred to the UART FIFO (up to three characters) and then stops transmitting data. The UART continues to receive normally.

CL—Character Length

0 = 7-bit character length. On receive, bit 7 in memory is written as zero. On transmit, bit 7 in memory is a don't care.

1 = 8-bit character length

RTSM—RTS Mode

0 = $\overline{\text{RTS}}$ is asserted whenever the transmitter is enabled and there are characters to transmit. $\overline{\text{RTS}}$ is negated after the last stop bit of a transmitted character when both the shift register and the transmit FIFO are empty. RTS is also negated at the end of a buffer to guarantee accurate reporting of the CTS bit in the BD.

1 = $\overline{\text{RTS}}$ is asserted whenever the transmitter is enabled (the ENT bit is set).

SL—Stop Length

This bit selects the number of the stop bits transmitted by the UART. The receiver is always enabled for one stop bit. Fractional stop bits are configured in the DSR (see 4.5.11.12 Fractional Stop Bits).

0 = One stop bit

1 = Two stop bits

COMMON SCC MODE BITS—see 4.5.3 SCC Mode Register (SCM) for a description of the DIAG1, DIAG0, ENR, ENT, MODE1, and MODE0 bits.

4.5.11.14 UART Receive Buffer Descriptor (Rx BD)

The CP reports information about each buffer of received data by its BDs. The Rx BD is shown in Figure 4-20. The CP closes the current buffer, generates a maskable interrupt, and starts to receive data in the next buffer due to any of the following events:

1. Reception of a user-defined control character (when reject (R) bit = 0)
2. Detection of an error during message processing
3. Detection of a full receive buffer
4. Reception of a programmable number of consecutive IDLE characters

5. Reception of an address character when working in multidrop mode

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET + 0	E	X	W	I	C	A	M	ID	—	—	BR	FR	PR	—	OV	CD
OFFSET + 2	DATA LENGTH															
OFFSET + 4	RX BUFFER POINTER (24-bits used, upper 8 bits must be 0)															
OFFSET +6																

Figure 4-20. UART Receive Buffer Descriptor

The first word of the Rx BD contains the control and status bits.

NOTE

In the nonautomatic multidrop mode (UM1–UM0 = 01), the address character will be written into the next buffer for comparison by the user software.

An example of the UART receive process is shown in Figure 4-21. This figure shows the resulting state of the Rx BDs after receipt of 10 characters, an idle period, and five characters—one with a framing error. The example assumes that MRBLR = 8 in the SCC parameter RAM.

E—Empty

0 = The data buffer associated with this BD has been filled with received data, or data reception has been aborted due to an error condition. The M68000 core is free to examine or write to any fields of the BD.

1 = The data buffer associated with the BD is empty. This bit is used to signify that the BD and its associated buffer are available to the CP. After it sets this bit, the M68000 core should not write to any fields of this BD when this bit is set. Note that the empty bit will remain set while the CP is currently filling the buffer with received data.

X—External Buffer

0 = The buffer associated with this BD is in internal dual-port RAM.

1 = The buffer associated with this BD is in external memory.

W—Wrap (Final BD in Table)

0 = This is not the last BD in the Rx BD table.

1 = This is the last BD in the Rx BD table. After this buffer has been used, the CP will receive incoming data into the first BD in the table, allowing the user to use fewer than eight BDs to conserve internal RAM.

NOTE

The user is required to set the wrap bit in one of the first eight BDs; otherwise, errant behavior may occur.

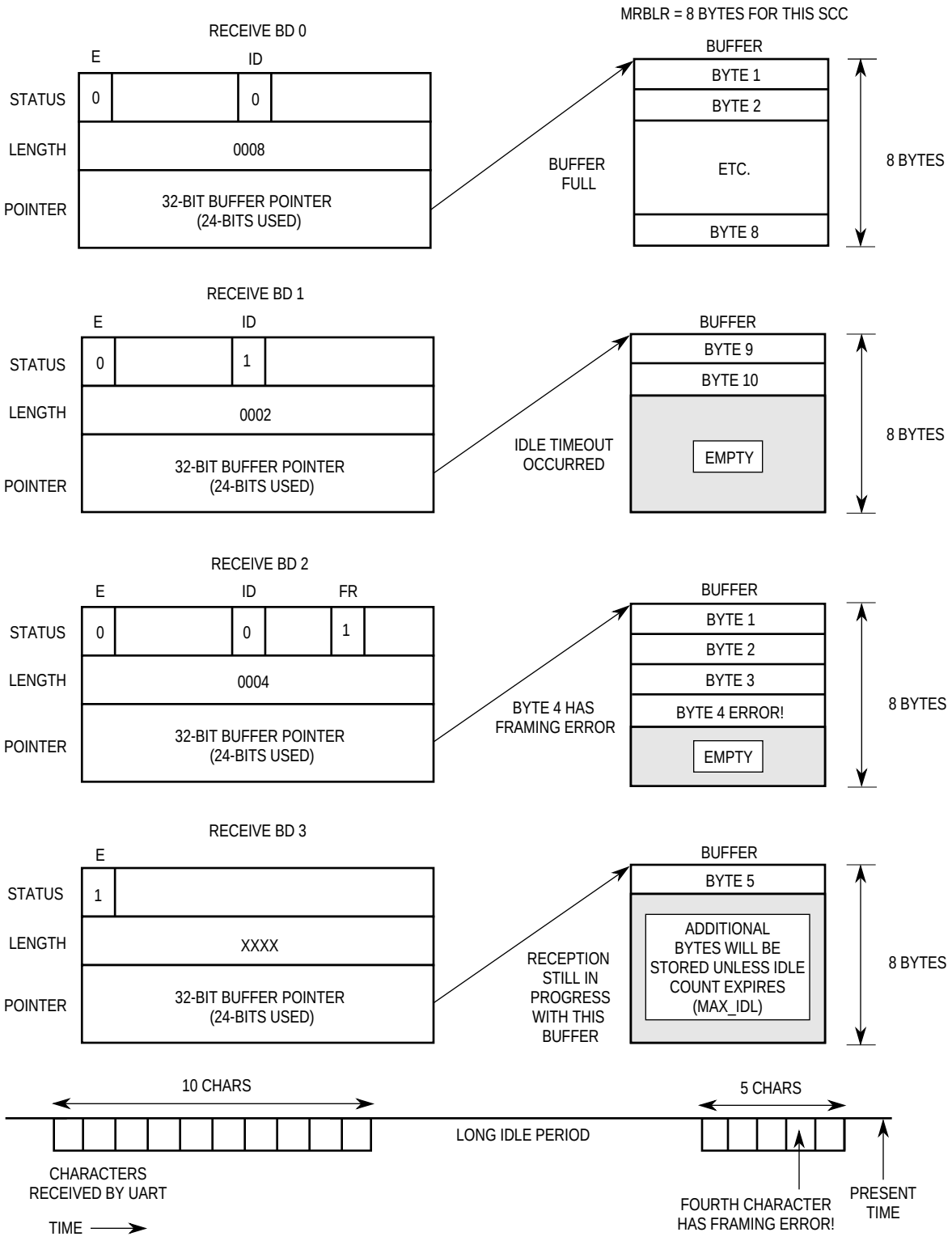


Figure 4-21. UART Rx BD Example

I—Interrupt

0 = No interrupt is generated after this buffer has been filled.
1 = The RX bit in the UART event register will be set when this buffer has been completely filled by the CP, indicating the need for the M68000 core to process the buffer. The RX bit can cause an interrupt.

The following bits contain status information written by the CP after it has finished receiving data in the associated data buffer.

C—Control Character

0 = This buffer does not contain a control character.
1 = This buffer contains a user-defined control character in the last byte location.

A—Address

0 = The buffer contains data only.
1 = When working in nonautomatic multidrop mode (UM1–UM0 = 01), this bit indicates that the first byte of this buffer contains an address byte. The address comparison should be implemented in software. In automatic multidrop mode, this bit indicates that the BD contains a message received immediately following an address recognized in UADDR1 or UADDR2. This address is not written into the receive buffer.

M—Address Match

This bit is meaningful only if the A bit (bit 10) is set and UM1–UM0 = 11 in the UART mode register. Following an address match, this bit defines which address character matched the user-defined address character, enabling the UART to receive the data.

0 = The address-matched user-defined UADDR2
1 = The address-matched user-defined UADDR1

ID—Buffer Closed on Reception of Idles

The buffer was closed due to the reception of the programmable number of consecutive IDLE sequences (defined in MAX_IDL).

Bits 7–6, 2—Reserved for future use.

BR—Break Received

A break sequence was received while receiving data into this buffer.

FR—Framing Error

A character with a framing error was received and is located in the last byte of this buffer. A framing error is detected by the UART controller when no stop bit is detected in the receive data string.

PR—Parity Error

A character with a parity error was received and is located in the last byte of this buffer.

OV—Overrun

A receiver overrun occurred during message reception.

CD—Carrier Detect Lost

The carrier detect signal was negated during message reception.

Data Length

Data length contains the number of octets written by the CP into this BD's data buffer. It is written by the CP once as the BD is closed.

NOTE

The actual amount of memory allocated for this buffer should be greater than or equal to the contents of maximum receive buffer length register (MRBLR).

Rx Buffer Pointer

The receive buffer pointer, which always points to the first location of the associated data buffer, may be even or odd. The buffer may reside in either internal or external memory.

NOTE

For correct operation of the function codes, the upper 8 bits of the pointer must be initialized to zero.

4.5.11.15 UART Transmit Buffer Descriptor (Tx BD)

Data is presented to the CP for transmission on an SCC channel by arranging it in buffers referenced by the channel's Tx BD table. The CP confirms transmission (or indicates error conditions) through the BDs to inform the M68000 core that the buffers have been serviced. The Tx BD shown in Figure 4-22.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET + 0	R	X	W	I	CR	A	P	—	—	—	—	—	—	—	—	CT
OFFSET + 2	DATA LENGTH															
OFFSET + 4 OFFSET +6	TX BUFFER POINTER (24-bits used, upper 8 bits must be 0)															

Figure 4-22. UART Transmit Buffer Descriptor

The first word of the Tx BD contains status and control bits. The following bits are prepared by the user before transmission and set by the CP after the buffer has been transmitted.

R—Ready

0 = This buffer is not currently ready for transmission. The user is free to manipulate the BD (or its associated buffer). The CP clears this bit after the buffer has been transmitted or after an error condition has been encountered.

1 = The data buffer, which has been prepared for transmission by the user, has not been transmitted or is currently transmitting. No fields of this BD may be written by the user once this bit is set.

X—External Buffer

0 = The buffer associated with this BD is in internal dual-port RAM.
1 = The buffer associated with this BD is in external memory.

W—Wrap (Final BD in Table)

0 = This is not the last BD in the Tx BD table.
1 = This is the last BD in the Tx BD table. After this buffer has been used, the CP will transmit data from the first BD in the table.

NOTE

The user is required to set the wrap bit in one of the first eight BDs; otherwise, errant behavior may occur.

I—Interrupt

0 = No interrupt is generated after this buffer has been serviced.
1 = The TX bit in the UART event register will be set when this buffer has been serviced by the CP, which can cause an interrupt.

Bits 8–1—Reserved for future use.

CR—Clear-to-Send Report

This bit allows a choice of no delay between buffers transmitted in UART mode, versus a more accurate CTS lost error reporting and two bits of idle between buffers.

0 = The buffer following this buffer will be transmitted with no delay (assuming it is ready), but the CT bit may not be set in the correct Tx BD, or may not be set at all in a CTS lost condition. The user is advised to monitor the CTS bit in the UART event register for an indication of CTS lost, in addition to the CT bits in the Tx BDs. The CTS bit will always be set properly.
1 = Normal CTS lost (CT bit) error reporting, and two bits of idle occur between back-to-back buffers.

If the DIAG1–DIAG0 bits in the SCM are set to software operation (rather than normal operation), then this bit only affects the delay between buffers, not the CTS reporting, and would normally be set to zero.

A—Address

This bit is valid only in multidrop mode (UM0 = 1).
0 = This buffer contains data only.
1 = Set by the M68000 core, this bit indicates that this buffer contains address character(s). All the buffer's data will be transmitted as address characters.

P—Preamble

0 = No preamble sequence is sent.
1 = The UART sends one preamble sequence (9 to 13 ones) before sending the data.

The following bits are written by the CP after it has finished transmitting the associated data buffer.

CT—CTS Lost

0 = The $\overline{\text{CTS}}$ signal remained active during transmission.
1 = The $\overline{\text{CTS}}$ signal was negated during transmission.

Data Length

The data length is the number of octets that the CP should transmit from this BD's data buffer. It is never modified by the CP. This value should be normally greater than zero. The data length may be equal to zero with the P bit set, and only a preamble will be sent.

Tx Buffer Pointer

The transmit buffer pointer, which always points to the first location of the associated data buffer, may be even or odd. The buffer may reside in either internal or external memory.

NOTE

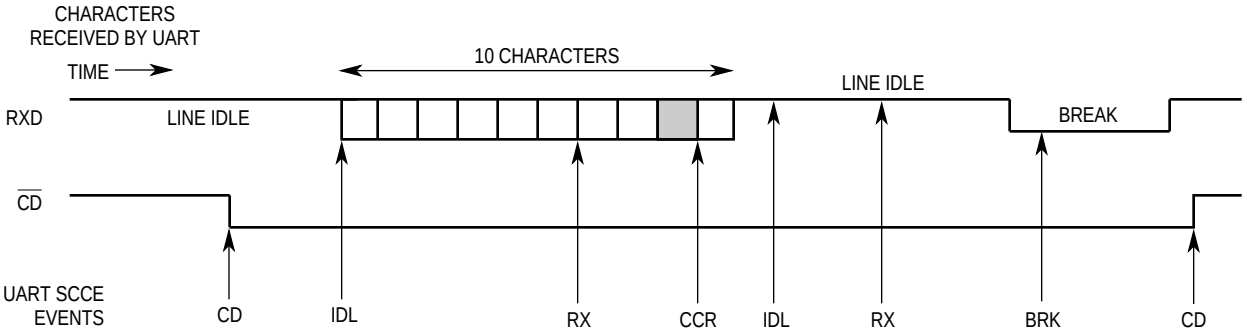
For correct operation of the function codes, the upper 8 bits of the pointer must be initialized to zero.

4.5.11.16 UART Event Register

The SCC event register (SCCE) is called the UART event register when the SCC is operating as a UART. It is an 8-bit register used to report events recognized by the UART channel and generate interrupts. On recognition of an event, the UART controller will set the corresponding bit in the UART event register. Interrupts generated by this register may be masked in the UART mask register.

The UART event register is a memory-mapped register that may be read at any time. A bit is cleared by writing a one (writing a zero does not affect a bit's value). More than one bit may be cleared at a time. All unmasked bits must be cleared before the CP will clear the internal interrupt request. This register is cleared at reset.

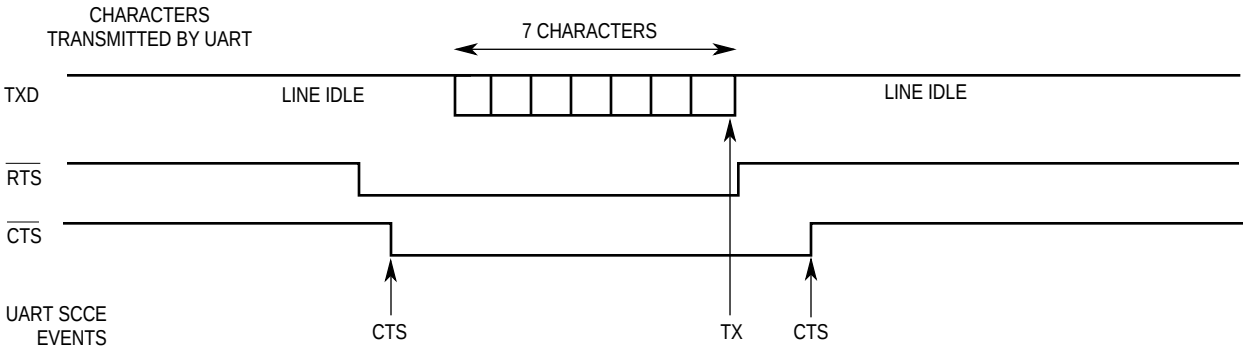
An example of the timing of various events in the UART event register is shown in Figure 4-23.



NOTES:
1. The first RX event assumes receive buffers are six bytes each.
2. The second IDL event occurs after 9 to 13 ones received in a row.
3. The second RX event position is programmable based on the max_IDL value.
4. The BRK event occurs after the first break character is received.

LEGEND:

Is a receive control character defined not to be stored in the receive buffer.



NOTE: TX event assumes all seven characters were put into a single buffer.

Figure 4-23. UART Interrupt Events Example

7	6	5	4	3	2	1	0
CTS	CD	IDL	BRK	CCR	BSY	TX	RX

CTS—Clear-To-End Status Changed

A change in the status of the $\overline{\text{CTS}}$ line was detected on the UART channel. The SCC status register may be read to determine the current status.

CD—Carrier Detect Status Changed

A change in the status of the $\overline{\text{CD}}$ line was detected on the UART channel. The SCC status register may be read to determine the current status.

IDL—IDLE Sequence Status Changed

A change in the status of the receive serial line was detected on the UART channel. The SCC status register may be read to determine the current status.

BRK—Break Character Received

A break character was received.

CCR—Control Character Received

A control character was received (with reject (R) character = 1) and stored in the receive control character register (RCCR).

BSY—Busy Condition

A character was received and discarded due to lack of buffers. The receiver automatically enters hunt mode immediately if in the multidrop mode. The latest that an Rx BD can be made empty (have its empty bit set) and still avoid the busy condition is the middle of the stop bit of the first character to be stored in that buffer.

TX—Tx Buffer

A buffer has been transmitted over the UART channel. If CR = 1 in the Tx BD, this bit is set no sooner than when the last data bit of the last character in the buffer begins to be transmitted. If CR = 0, this bit is set after the last character was written to the transmit FIFO.

RX—Rx Buffer

A buffer has been received over the UART channel. This event occurs no sooner than the middle of the first stop bit of the character that causes the buffer to be closed.

4.5.11.17 UART MASK Register

The SCC mask register (SCCM) is referred to as the UART mask register when the SCC is operating as a UART. It is an 8-bit read-write register with the same bit formats as the UART event register. If a bit in the UART mask register is a one, the corresponding interrupt in the event register will be enabled. If the bit is zero, the corresponding interrupt in the event register will be masked. This register is cleared upon reset.

4.5.11.18 S-Records Programming Example

In the following paragraphs, an example of a downloading application is given that utilizes an SCC channel as a UART controller. The application performs downloads and uploads of S records between a host computer and an intelligent peripheral through a serial asynchronous line.

The S records are strings of ASCII characters that begin with `S' and end in an end-of-line character. This characteristic will be used to impose a message structure on the communication between the devices. Note that each device may also transmit XON and XOFF characters for flow control, which do not form part of the program being uploaded or downloaded.

The UART mode register should be set as required, with the freeze (FRZ) bit cleared and the enable transmitter/receiver (ENT, ENR) bits set. Receive buffers should be linked to the receive buffer table with the interrupt (I) bit set. For simplicity, assume that the line is not multidrop (no addresses are transmitted) and that each S record will fit into a single data buffer.

Three characters should first be entered into the UART control character table:

1. End of Line—The empty (E) bit is cleared; the reject (R) bit is cleared. When an end-of-line character is received, the current buffer is closed (the next BD taken by the IMP) and made available to the M68000 core for processing. This buffer contains an entire S record, which the processor can now check and copy to memory or disk as required.
2. XOFF—E should be cleared and R should be set. Whenever the M68000 core receives a control character received interrupt and the receive control character register contains XOFF, the software should immediately stop transmitting to the other station by setting the FRZ bit in the UART mode register. This prevents data from being lost by the other station when it runs out of receive buffers.
3. XON—XON should be received after XOFF. E should be cleared and R should be set. The FRZ bit on the transmitter should now be cleared. The IMP automatically resumes transmission of the serial line at the point at which it was previously stopped. Like XOFF, the XON character is not stored in the receive buffer.

To receive the S records, the M68000 core must only wait for the RX interrupt, indicating the reception of a complete S-record buffer. Transmission requires assembling S records into data buffers and linking them to the transmit buffer table (transmission may be temporarily halted by reception of an XOFF character). This scheme minimizes the number of interrupts received by the M68000 core (one per S record) and relieves it from the task of continually scanning for control characters.

4.5.12 HDLC Controller

Layer 2 of the seven-layer OSI model is the data link layer. One of the most common layer 2 protocols is HDLC. Many other common layer 2 protocols are heavily based on HDLC, particularly its framing structure: namely, SDLC, SS#7, LAPB, and LAPD. The framing structure of HDLC is shown in Figure 4-24.

OPENING FLAG	ADDRESS	CONTROL	INFORMATION (OPTIONAL)	CRC	CLOSING FLAG
8 BITS	16 BITS	8 BITS	8N BITS	16 BITS	8 BITS

Figure 4-24. Typical HDLC Frame

HDLC uses a zero insertion/deletion process (commonly known as bit-stuffing) to ensure that the bit pattern of the delimiter flag does not occur in the fields between flags. The HDLC frame is synchronous and therefore relies on the physical layer to provide a method of clocking and synchronizing the transmitter and receiver.

Since the layer 2 frame can be transmitted over a point-to-point link, a broadcast network, or packet and circuit-switched systems, an address field is needed to carry the frame's destination address. The length of this field is commonly 0, 8, or 16 bits, depending on the data link layer protocol. For instance, SDLC and LAPB use an 8-bit address. SS#7 has no address field at all because it is always used in point-to-point signaling links. LAPD further divides its 16-bit address into different fields to specify various access points within one piece of equipment. It also defines a broadcast address. Some HDLC-type protocols also allow for extended addressing beyond 16-bits.

The 8- or 16-bit control field provides a flow control number and defines the frame type (control or data). The exact use and structure of this field depends upon the protocol using the frame.

Data is transmitted in the data field, which can vary in length depending upon the protocol using the frame. Layer 3 frames are carried in the data field.

Error control is implemented by appending a cyclic redundancy check (CRC) to the frame, which is 16-bits long in most protocols, but may be 32-bits long in some.

When the MODE1–MODE0 bits of an SCC mode register (SCM) select the HDLC mode, then that SCC functions as an HDLC controller. The HDLC controller handles the basic functions of the HDLC/SDLC protocol on either the D channel, a B channel, or from a multiplexed serial interface (IDL, GCI (IOM-2), or PCM highway). When the HDLC controller is used to support the B or D channel of the ISDN, the SCC outputs are internally connected to the physical layer serial interface.

NOTE

SDLC is fully supported, but the SDLC loop mode (ring configuration) is not supported.

When an SCC in HDLC mode is used with a nonmultiplexed modem interface, then the SCC outputs are connected directly to the external pins. In this case, the serial interface uses seven dedicated pins: transmit data (TXD), receive data (RXD), receive clock (RCLK), transmit clock (TCLK), carrier detect ($\overline{CD}$), clear to send ($\overline{CTS}$), and request to send ($\overline{RTS}$). Other modem signals may be supported through the parallel I/O pins.

The HDLC controller consists of separate transmit and receive sections whose operations are asynchronous with the M68000 core and may be either synchronous or asynchronous with respect to the other SCCs. Up to eight frames may be transmitted or received without M68000 core intervention. When the HDLC controller is connected to one of the multiplexed physical interface options (IDL, GCI, or PCM highway), the receive and transmit clocks are identical and are supplied externally by the physical layer. In non-ISDN applications, each clock can be supplied either from the baud rate generator or externally. The baud rate generator is discussed more fully in 4.5.2 SCC Configuration Register (SCON).

The HDLC controller key features are as follows:

- Flexible Data Buffers with Multiple Buffers per Frame Allowed
- Separate Interrupts for Frames and Buffers (Receive and Transmit)
- Four Address Comparison Registers with Mask
- Maintenance of Five 16-Bit Error Counters
- Flag/Abort/Idle Generation/Detection
- Zero Insertion/Deletion
- NRZ/NRZI Data Encoding
- 16-Bit or 32-Bit CRC-CCITT Generation/Checking

- Detection of Non-Octet Aligned Frames
- Detection of Frames That Are Too Long
- Programmable Flags (0–15) between Successive Frames
- Automatic Retransmission in Case of Collision

4.5.12.1 HDLC Channel Frame Transmission Processing

The HDLC transmitter is designed to work with almost no intervention from the M68000 core. When the M68000 core enables one of the transmitters, it will start transmitting flags or idles as programmed in the HDLC mode register. The HDLC controller will poll the first buffer descriptor (BD) in the transmit channel's BD table. When there is a frame to transmit, the HDLC controller will fetch the data from memory and start transmitting the frame (after first transmitting the user-specified minimum number of flags between frames). When the end of the current BD has been reached and the last buffer in the frame bit is set, the cyclic redundancy check (CRC), if selected, and the closing flag are appended.

Following the transmission of the closing flag, the HDLC controller writes the frame status bits into the BD and clears the ready bit. When the end of the current BD has been reached, and the last bit is not set (working in multibuffer mode), only the ready bit is cleared. In either mode, an interrupt is issued according to the interrupt bit in the BD. The HDLC controller will then proceed to the next BD in the table. In this way, the user may be interrupted after each buffer, after a specific buffer has been transmitted, or after each frame.

To rearrange the transmit queue before the IMP has completed transmission of all buffers, issue the STOP TRANSMIT command. This technique can be useful for transmitting expedited data before previously linked buffers or for error situations. When receiving the STOP TRANSMIT command, the HDLC controller will abort the current frame being transmitted and start transmitting idles or flags. When the HDLC controller is given the RESTART TRANSMIT command, it resumes transmission.

4.5.12.2 HDLC Channel Frame Reception Processing

The HDLC receiver is also designed to work with almost no intervention from the M68000 core. The HDLC receiver can perform address recognition, CRC checking, and maximum frame length checking. The received frame (all fields between the opening and closing flags) is made available to the user for performing any HDLC-based protocol.

When the M68000 core enables one of the receivers, the receiver waits for an opening flag character. When the receiver detects the first byte of the frame, the HDLC controller will compare the frame address against the user-programmable addresses. The user has four 16-bit address registers and an address mask available for address matching. The HDLC controller will compare the received address field to the user-defined values after masking with the address mask. The HDLC controller can also detect broadcast (all ones) addressed frames, if one address register is written with all ones.

If a match is detected, the HDLC controller will fetch the next BD and, if empty, will start to transfer the incoming frame to the BD's associated data buffer starting with the first address byte. When the data buffer has been filled, the HDLC controller clears the empty bit in the BD and generates an interrupt if the interrupt bit in the BD is set. If the incoming frame ex-

ceeds the length of the data buffer, the HDLC controller will fetch the next BD in the table and, if it is empty, will continue to transfer the rest of the frame to this BD's associated data buffer.

During this process, the HDLC controller will check for a frame that is too long. When the frame ends, the CRC field is checked against the recalculated value and is written to the data buffer starting with the first address byte. The data length written to the last BD in the HDLC frame is the length of the entire frame. This enables HDLC protocols that “lose” frames to correctly recognize the frame-too-long condition. The HDLC controller then sets the last buffer in frame bit, writes the frame status bits into the BD, and clears the empty bit. The HDLC controller next generates a maskable interrupt, indicating that a frame has been received and is in memory. The HDLC controller then waits for a new frame. Back-to-back frames may be received with only a single shared flag between frames. Also, flags that share a zero will be recognized as two consecutive flags.

4.5.12.3 HDLC Memory Map

When configured to operate in HDLC mode, the IMP overlays the structure shown in Table 4-7 onto the protocol-specific area of that SCC parameter RAM. Refer to 2.8 MC68302 Memory Map for the placement of the three SCC parameter RAM areas and to Table 4-2 for the other parameter RAM values.

Table 4-8. HDLC-Specific Parameter RAM

Address	Name	Width	Description
SCC Base + 9C SCC Base + 9E SCC Base + A0 # SCC Base + A2 # SCC Base + A4 SCC Base + A6	RCRC_L RCRC_H C_MASK_L C_MASK_H TCRC_L TCRC_H	Word Word Word Word Word Word	Temp Receive CRC Low Temp Receive CRC High Constant ($F0B8 16-Bit CRC, $DEBB 32-Bit CRC) Constant ($XXXX 16-Bit CRC, $20E3 32-Bit CRC) Temp Transmit CRC Low Temp Transmit CRC High
SCC Base + A8 # SCC Base + AA # SCC Base + AC # SCC Base + AE # SCC Base + B0 #	DISFC CRCEC ABTSC NMARC RETRC	Word Word Word Word Word	Discard Frame Counter CRC Error Counter Abort Sequence Counter Nonmatching Address Received Counter Frame Retransmission Counter
SCC Base + B2 # SCC Base + B4	MFLR MAX_cnt	Word Word	Max Frame Length Register Max_Length Counter
SCC Base + B6 # SCC Base + B8 # SCC Base + BA # SCC Base + BC # SCC Base + BE #	HMASK HADDR1 HADDR2 HADDR3 HADDR4	Word Word Word Word Word	User-Defined Frame Address Mask User-Defined Frame Address User-Defined Frame Address User-Defined Frame Address User-Defined Frame Address

Initialized by the user (M68000 core).

NOTE

An incorrect initialization of C_MASK may be used to “force” receive CRC errors for software testing purposes. The transmit CRC will not be affected.

4.5.12.4 HDLC Programming Model

The M68000 core configures each SCC to operate in one of four protocols by the MODE1–MODE0 bits in the SCC mode register (SCM). MODE1–MODE0 = 00 selects HDLC mode.

The HDLC controller uses the same data structure as the UART, BISYNC, and DDCMP controllers. This data structure supports multibuffer operation and address comparisons.

The receive errors (overrun, nonoctet aligned frame, $\overline{CD}$ lost, aborted frame, and CRC error) are reported through the receive BD. The transmit errors (underrun and $\overline{CTS}$ lost) are reported through the transmit BD. An indication about the status of the lines (idle, $\overline{CD}$, and $\overline{CTS}$) is reported through the SCC status register (SCCS), and a maskable interrupt is generated upon a status change in any one of those lines.

4.5.12.5 HDLC Command Set

The following commands are issued to the command register.

STOP TRANSMIT Command

After a hardware or software reset and the enabling of the channel in the SCC mode register, the channel is in the transmit enable mode and starts polling the first BD in the table approximately every eight or sixteen transmit clocks as determined by the FLG bit in the HDLC mode register.

The channel STOP TRANSMIT command disables the transmission of frames on the transmit channel. If this command is received by the HDLC controller during frame transmission, transmission of that frame is aborted after the contents of the FIFO are transmitted (up to four words). The TBD# is not advanced. No new BD is accessed, and no new frames are transmitted for this channel. The transmitter will transmit an abort sequence (if the command was given during frame transmission) and then begin to transmit flags or idles as indicated by the HDLC mode register. The abort sequence on transmit is a zero followed by seven ones (01111111).

This command is useful for performing frame retransmission. The M68000 core may issue the STOP TRANSMIT command, reorganize the transmit BD table, and issue the RESTART TRANSMIT command. The STOP TRANSMIT command may also be used in the X.25 protocol to send a reject frame or a link reset command.

The STOP TRANSMIT command must be issued before the SCC mode register is used to disable the transmitter if the transmitter is to be re-enabled at a later time.

RESTART TRANSMIT Command

The RESTART TRANSMIT command re-enables the transmission of characters on the transmit channel. This command is expected by the HDLC controller after a STOP TRANSMIT command, after a STOP TRANSMIT command and disabling the channel in its SCC mode register, or after transmitter error (underrun or $\overline{CTS}$ lost when no automatic frame retransmission is performed). The HDLC controller will resume transmission from the current transmitter BD (TBD#) in the channel's transmit BD table.

If the transmitter is being re-enabled, the RESTART TRANSMIT command must be used and should be followed by the enabling of the transmitter in the SCC mode register.

ENTER HUNT MODE Command

After a hardware or software reset and the enabling of the channel by its SCC mode register, the channel is in the receive enable mode and will use the first BD in the table.

The ENTER HUNT MODE command is generally used to force the HDLC receiver to abort reception of the current frame, generate an RXB interrupt (if enabled) as the buffer is closed, and enter the hunt mode. In the hunt mode, the HDLC controller continually scans the input data stream for the flag sequence. After receiving the command, the current receive buffer is closed, and the CRC is reset. Further frame reception will use the next BD.

If an enabled receiver has been disabled by clearing ENR in the SCC mode register), the ENTER HUNT MODE command must be given to the channel before setting ENR again. Subsequent frames will then be received, starting with the next BD.

4.5.12.6 HDLC Address Recognition

Each HDLC controller has five 16-bit registers for address recognition: one mask register and four address registers (HMASK, HADDR1, HADDR2, HADDR3, and HADDR4). The HDLC controller reads the frame's address from the HDLC receiver, checks it against the four address register values, and then masks the result with the user-defined HMASK. A one in HMASK represents a bit position for which address comparison should occur; a zero represents a masked bit position. Upon an address match, the address and the data following are written into the data buffers. When the addresses are not matched and the frame is error free, the nonmatching address received counter (NMARC) is incremented.

NOTE

For 8-bit addresses, mask out the eight high-order bits in the HMASK register.

Examples of 16- and 8-bit HDLC address recognition are shown in Figure 4-25.

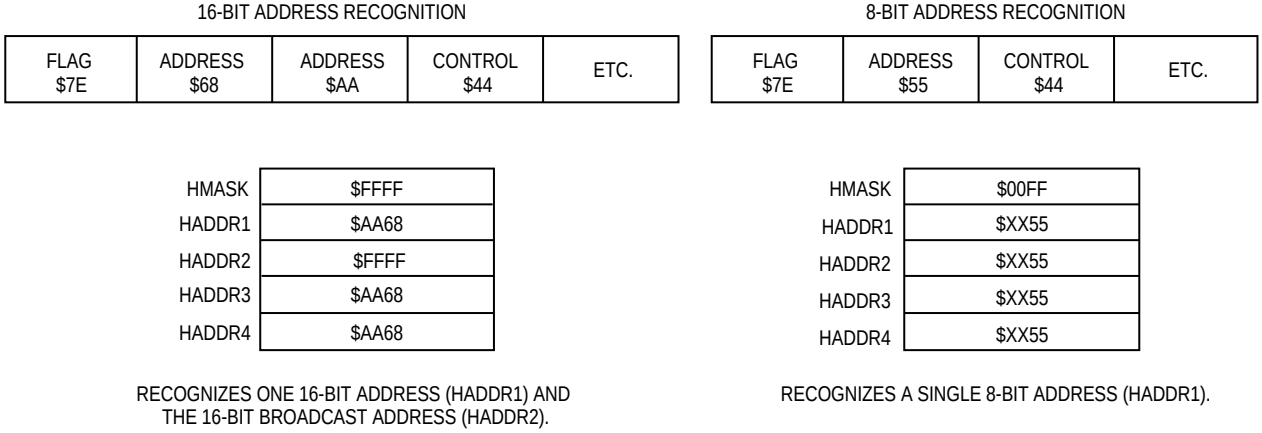


Figure 4-25. HDLC Address Recognition Examples

4.5.12.7 HDLC Maximum Frame Length Register (MFLR)

The HDLC controller checks the length of an incoming HDLC frame against the user-defined value given in this 16-bit register. If this limit is exceeded, the remainder of the incoming HDLC frame is discarded, and the LG (Rx frame too long) bit is set in the last BD belonging to that frame. The HDLC controller waits to the end of the frame and reports the frame status

and the frame length in the last BD. MFLR is defined as all the in-frame bytes between the opening flag and the closing flag (address, control, data, and CRC). MAX_CNT is a temporary downcounter used to track the frame length.

4.5.12.8 HDLC Error-Handling Procedure

The HDLC controller reports frame reception and transmission error conditions using the channel BDs, the error counters, and the HDLC event register. The modem interface lines can also be directly monitored in the SCC status register.

Transmission Errors:

1. Transmitter Underrun. When this error occurs, the channel terminates buffer transmission, closes the buffer, sets the underrun (UN) bit in the BD, and generates the TXE interrupt (if enabled). The channel will resume transmission after the reception of the RESTART TRANSMIT command. The transmit FIFO size is four words.
2. Clear-To-Send Lost (Collision) During Frame Transmission. When this error occurs and the channel is not programmed to control this line with software, the channel terminates buffer transmission, closes the buffer, sets the CTS lost (CT) bit in the BD, and generates the TXE interrupt (if enabled). The channel will resume transmission after the RESTART TRANSMIT command is given.

NOTE

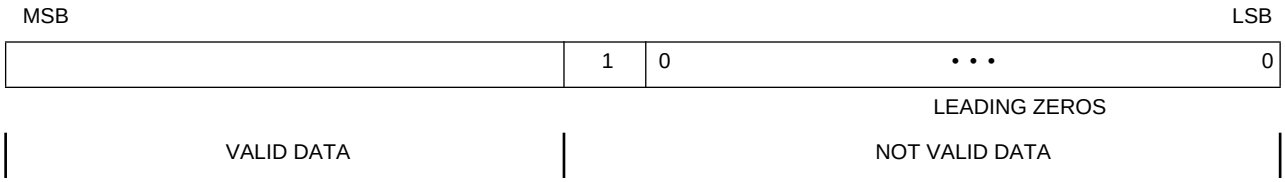
If this error occurs on the first or second buffer of the frame and the retransmit enable (RTE) bit in the HDLC mode register is set, the channel will retransmit the frame when the $\overline{\text{CTS}}$ line becomes active again. When using this feature, users should design transmit frames to fit within two buffers or less. When working in ISDN mode with D-channel collision possibility, to ensure the retransmission method functions properly, the first and second data buffers should contain more than 10 bytes of data if multiple buffers per frame are used. (Small frames consisting of a single buffer are not subject to this requirement). The channel will also increment the retransmission counter (RETRC).

Reception Errors:

1. Overrun Error. The HDLC controller maintains an internal three-word FIFO for receiving data. The CP begins programming the SDMA channel (if the data buffer is in external memory) and updating the CRC when the first word is received in the FIFO. When a receive FIFO overrun occurs, the channel writes the received data byte to the internal FIFO over the previously received byte. The previous data byte and the frame status are lost. Then the channel closes the buffer with the overrun (OV) bit in the BD set and generates the RXF interrupt (if enabled). The receiver then enters the hunt mode.
   Even if the overrun occurs during a frame whose address is not matched in the address recognition logic, a BD of length two will be opened to report the overrun, and the RXB interrupt will be generated (if enabled).
2. Carrier Detect Lost During Frame Reception. When this error occurs and the channel

is not programmed to control this line with software, the channel terminates frame reception, closes the buffer, sets the carrier detect lost (CD) bit in the BD, and generates the RXF interrupt (if enabled). This error has the highest priority. The rest of the frame is lost, and other errors are not checked in that frame. The receiver then enters the hunt mode.

3. Abort Sequence. An abort sequence is detected by the HDLC controller when seven or more consecutive ones are received while receiving a frame. When this error occurs, the channel closes the buffer by setting the Rx abort sequence (AB) bit in the BD and generates the RXF interrupt (if enabled). The channel also increments the abort sequence counter (ABTSC). The receiver then enters hunt mode immediately. The CRC and nonoctet error status conditions are not checked on aborted frames. The receiver then enters hunt mode.
4. Nonoctet Aligned Frame. When this error occurs, the channel writes the received data to the data buffer, closes the buffer, sets the Rx nonoctet aligned frame (NO) bit in the BD, and generates the RXF interrupt (if enabled). The CRC error status should be disregarded on nonoctet frames. After a nonoctet aligned frame is received, the receiver enters hunt mode. (An immediately following back-to-back frame will be received.) The nonoctet data may be derived from the last word in the data buffer as follows:



Consistent with other HDLC operation, the MSB is the first bit received in this word, and the low-order valid data bit is the last.

5. CRC Error. When this error occurs, the channel writes the received CRC to the data buffer, closes the buffer, sets the CR bit in the BD, and generates the RXF interrupt (if enabled). The channel also increments the CRC error counter (CRCEC). After receiving a frame with a CRC error, the received enters hunt mode. (An immediately following back-to-back frame will be received.) CRC checking cannot be disabled, but the CRC error may be ignored if checking is not required.

Error Counters

The CP maintains five 16-bit (modulo - 2**16) error counters for each HDLC controller. They can be initialized by the user when the channel is disabled. The counters are as follows:

DISFC—Discarded Frame Counter (error-free frames but no free buffers)
CRCEC—CRC Error Counter (includes frames not addressed to the user or frames received in the BSY condition, but does not include overrun errors)
ABTSC—Abort Sequence Counter
NMARC—Nonmatching Address Received Counter (error-free frames only)
RETRC—Frame Retransmission Counter (due to collision)

4.5.12.9 HDLC Mode Register

Each SCC mode register is a 16-bit, memory-mapped, read-write register that controls the SCC operation. The term HDLC mode register refers to the protocol-specific bits (15–6) of

the SCC mode register when that SCC is configured for HDLC. The read-write HDLC mode register is cleared by reset.

15	14	13	12	11	10	9	8	7	6	5 … 0
NOF3	NOF2	NOF1	NOF0	C32	FSE	—	RTE	FLG	ENC	COMMON SCC MODE BITS

NOF3–NOF0—Minimum Number of Flags between Frames or before Frames (0 to 15 Flags)

If NOF3–NOF0 = 0000, then no flags will be inserted between frames. Thus, the closing flag of one frame will be followed immediately by the opening flag of the next frame in the case of back-to-back frames.

C32—CRC16/CRC32

0 = 16-bit CCITT CRC ($X^{16} + X^{12} + X^5 + 1$)
1 = 32-bit CCITT CRC ($X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X^1 + 1$)

FSE—Flag Sharing Enable

0 = Normal operation
1 = If NOF3–NOF0 = 0000, then a single shared flag is transmitted between back- to-back frames. Other values of NOF3–NOF0 are decremented by one when FSE is set. This is useful in Signaling System #7 applications.

Bit 9—Reserved for future use.

RTE—Retransmit Enable

0 = No automatic retransmission will be performed.
1 = Automatic retransmit enabled

Automatic retransmission occurs if a $\overline{\text{CTS}}$ lost condition happens on the first or second buffer of the frame. See 4.5.12.8 HDLC Error-Handling Procedure.

FLG—Transmit Flags/Idles between Frames and Control the RTS Pin

0 = Send ones between frames; $\overline{\text{RTS}}$ is negated between frames. If NOF–NOF0 is greater than zero, $\overline{\text{RTS}}$ will be negated for a multiple of eight transmit clocks. The HDLC controller can transmit ones in both the NRZ and NRZI data encoding formats. The CP polls the Tx BD ready bit every 16 transmit clocks.
1 = Send flags between frames. $\overline{\text{RTS}}$ is always asserted. The CP polls the Tx BD ready bit every eight transmit clocks.

NOTE

This bit may be dynamically modified. If toggled from a one to a zero between frames, a maximum of two additional flags will be transmitted before the idle condition will begin. Toggling FLG will never result in partial flags being transmitted.

ENC—Data Encoding Format

0 = Non-return to zero (NRZ). A one is a high level; a zero is a low level.
1 = Non-return to zero inverted (NRZI). A one is represented by no change in the level; a zero is represented by a change in the level. The receiver decodes NRZI, but a clock must be supplied. The transmitter encodes NRZI. During an idle condition, with the FLG bit cleared, the line will be forced to a high state.

COMMON SCC MODE BITS—See 4.5.3 SCC Mode Register (SCM) for a description of the DIAG1, DIAG0, ENR, ENT, MODE1, and MODE0 bits.

4.5.12.10 HDLC Receive Buffer Descriptor (Rx BD)

The HDLC controller uses the Rx BD to report information about the received data for each buffer. The Rx BD is shown in Figure 4-26.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET + 0	E	X	W	I	L	F	—	—	—	—	LG	NO	AB	CR	OV	CD
OFFSET + 2	DATA LENGTH															
OFFSET + 4	RX BUFFER POINTER (24-bits used, upper 8 bits must be 0)															
OFFSET +6																

Figure 4-26. HDLC Receive Buffer Descriptor

An example of the HDLC receive process is shown in Figure 4-27. This shows the resulting state of the Rx BDs after receipt of a complete frame spanning two receive buffers and a second frame with an unexpected abort sequence. The example assumes that MRBLR = 8 in the SCC parameter RAM.

The first word of the Rx BD contains control and status bits. Bits 15–10 are written by the user before the buffer is linked to the Rx BD table, and bits 5–0 are set by the CP following frame reception. Bit 15 is set by the M68000 core when the buffer is available to the HDLC controller; it is cleared by the HDLC controller when the buffer is full.

E—Empty

0 = The data buffer associated with this BD has been filled with received data, or data reception has been aborted due to an error condition. The M68000 core is free to examine or write to any fields of the BD.
1 = The data buffer associated with the BD is empty. This bit signifies that the BD and its associated buffer are available to the HDLC controller. The M68000 core should not write to any fields of this BD after it sets this bit. The empty bit will remain set while the HDLC controller is currently filling the buffer with received data.

X—External Buffer

0 = The buffer associated with this BD is in internal dual-port RAM.
1 = The buffer associated with this BD is in external memory.

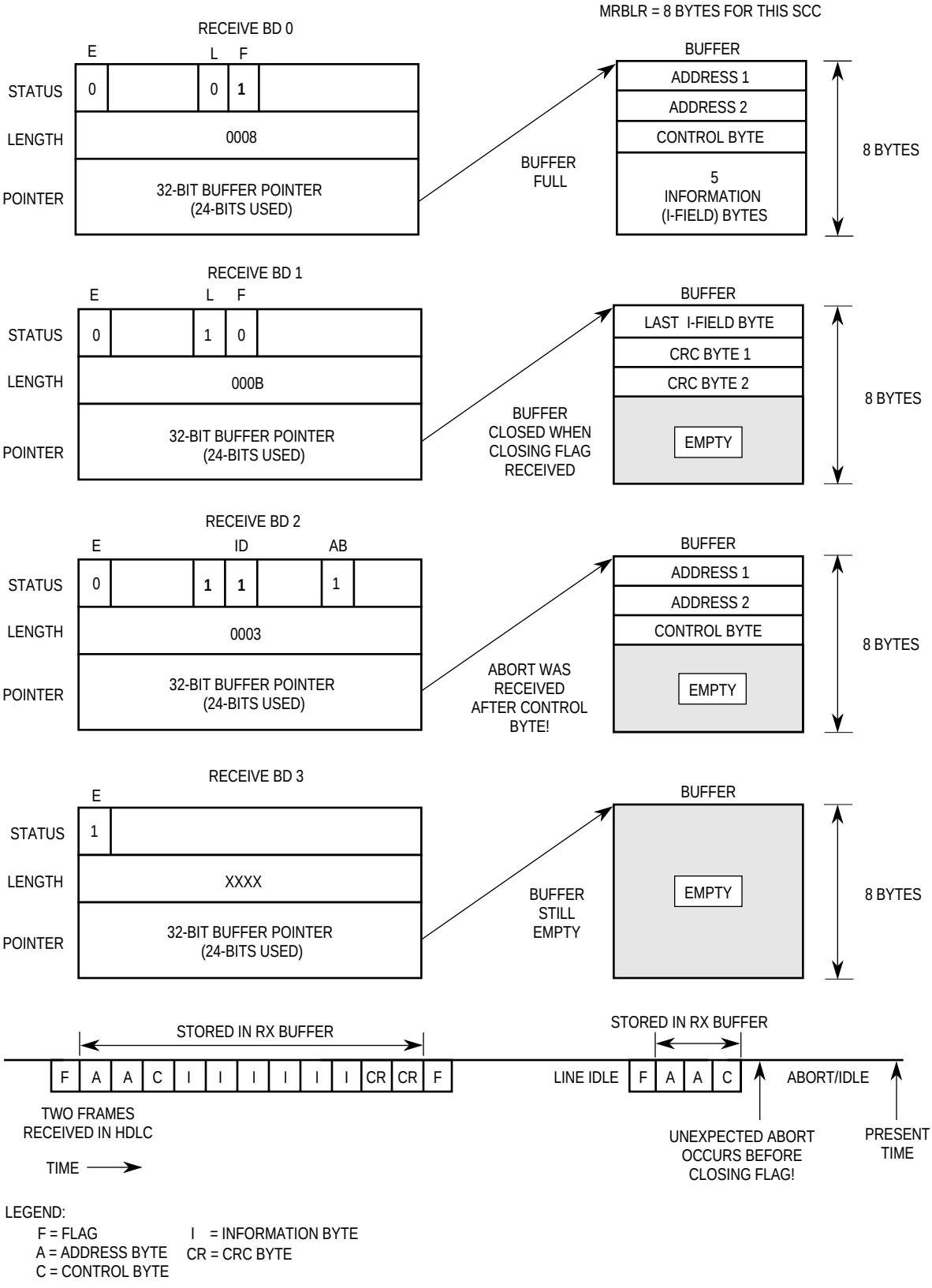


Figure 4-27. HDLC Receive BD Example

W—Wrap (Final BD in Table)

0 = This is not the last BD in the Rx BD table.
1 = This is the last BD in the Rx BD table. After this buffer has been used, the HDLC controller will receive incoming data into the first BD in the table.

NOTE

The user is required to set the wrap bit in one of the first eight BDs; otherwise, errant operation may occur.

I—Interrupt

0 = The RXB bit is not set after this buffer has been used, but RXF operation remains unaffected.
1 = The RXB or RXF bit in the HDLC event register will be set when this buffer has been used by the HDLC controller, which can cause an interrupt.

The following status bits are written by the HDLC controller after the received data has been placed into the associated data buffer.

L—Last in Frame

This bit is set by the HDLC controller when this buffer is the last in a frame. This implies the reception of a closing flag or reception of an error, in which case one or more of the CD, OV, AB, and LG bits are set. The HDLC controller will write the number of frame octets to the data length field.

0 = This buffer is not the last in a frame.
1 = This buffer is the last in a frame.

F—First in Frame

This bit is set by the HDLC controller when this buffer is the first in a frame.

0 = The buffer is not the first in a frame.
1 = The buffer is the first in a frame.

Bits 9–6—Reserved for future use.

LG—Rx Frame Length Violation.

A frame length greater than the maximum defined for this channel was recognized (only the maximum-allowed number of bytes (MFLR) is written to the data buffer). This event will not be reported until the Rx BD is closed and the RXF bit is set, after receipt of the closing flag. The actual number of bytes received between flags is written to the data length field of this BD.

NO—Rx Nonoctet Aligned Frame

A frame that contained a number of bits not exactly divisible by eight was received.

AB—Rx Abort Sequence

A minimum of seven consecutive ones was received during frame reception.

CR—Rx CRC Error

This frame contains a CRC error.

OV—Overrun

A receiver overrun occurred during frame reception.

CD—Carrier Detect Lost

The carrier detect signal was negated during frame reception. This bit is valid only when working in NMSI mode.

Data Length

The data length is the number of octets written to this BD's data buffer by the HDLC controller. It is written by the CP once as the BD is closed.

When this BD is the last BD in the frame (L = 1), the data length contains the total number of frame octets (including any previous linked receive data buffers and two or four bytes for the CRC) in the frame. This behavior is useful for determining the total number of octets received, even if MFLR was exceeded.

NOTE

The actual amount of memory allocated for this buffer should be even and greater than or equal to the contents of maximum receive buffer length register (MRBLR).

Rx Buffer Pointer

The receive buffer pointer, which always points to the first location of the associated data buffer, may reside in either internal or external memory.

NOTE

The Rx buffer pointer must be even, and the upper 8 bits must of the pointer must be zero for the function codes to operate correctly.

4.5.12.11 HDLC Transmit Buffer Descriptor (Tx BD)

Data is presented to the HDLC controller for transmission on an SCC channel by arranging it in buffers referenced by the channel's Tx BD table. The HDLC controller confirms transmission (or indicates error conditions) using the BDs to inform the M68000 core that the buffers have been serviced. The Tx BD is shown in Figure 4-28.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET + 0	R	X	W	I	L	TC	—	—	—	—	—	—	—	—	UN	CT
OFFSET + 2	DATA LENGTHTX BUFFER POINTER (24-bits used, upper 8 bits must be 0)															
OFFSET + 4																
OFFSET + 6																

Figure 4-28. HDLC Transmit Buffer Descriptor

The first word of the Tx BD contains status and control bits. Bits 15–10 are prepared by the user before transmission; bits 1–0 are set by the HDLC controller after the buffer has been transmitted. Bit 15 is set by the user when the buffer and BD have been prepared and is cleared by the HDLC controller after the frame has been transmitted.

R—Ready

0 = This buffer is not currently ready for transmission. The user is free to manipulate this BD (or its associated buffer). The HDLC controller clears this bit after the buffer has been fully transmitted or after an error condition has been encountered.
1 = The data buffer, which has been prepared for transmission by the user, has not yet transmitted. No fields of this BD may be written by the user once this bit is set.

X—External Buffer

0 = The buffer associated with this BD is in internal dual-port RAM.
1 = The buffer associated with this BD is in external memory.

W—Wrap (Final BD in Table)

0 = This is not the last BD in the Tx BD table.
1 = This is the last BD in the Tx BD table. After this buffer has been used, the HDLC controller will transmit data from the first BD in the table.

NOTE

The user is required to set the wrap bit in one of the first eight BDs; otherwise, errant behavior may occur.

I—Interrupt

0 = No interrupt is generated after this buffer has been serviced.
1 = Either TXB or TXE in the HDLC event register will be set when this buffer has been serviced by the HDLC controller, which can cause an interrupt.

L—Last

0 = This is not the last buffer in the frame.
1 = This is the last buffer in the current frame.

TC—Tx CRC

This bit is valid only when the last (L) bit is set.

0 = Transmit the closing flag after the last data byte. This setting can be used for testing purposes to send a “bad” CRC after the data.
1 = Transmit the CRC sequence after the last data byte.

Bits 9–2—Reserved for future use.

The following status bits are written by the HDLC controller after it has finished transmitting the associated data buffer.

UN—Underrun

The HDLC controller encountered a transmitter underrun condition while transmitting the associated data buffer.

CT—CTS Lost

CTS in NMSI mode or L1GR (layer-1 grant) in IDL/GCI mode was lost during frame transmission. If data from more than one buffer is currently in the FIFO when this error occurs, this bit will be set in the Tx BD that is currently open.

Data Length

The data length is the number of octets the HDLC controller should transmit from this BD's data buffer. It is never modified by the CP. The value of this field should be greater than zero.

Tx Buffer Pointer

The transmit buffer pointer, which contains the address of the associated data buffer, may be even or odd. The buffer may reside in either internal or external memory.

NOTE

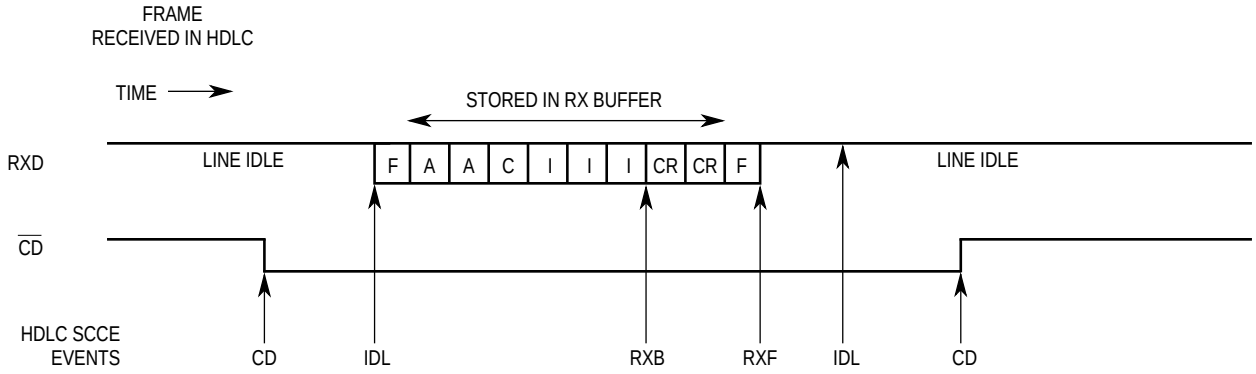
For correct operation of the function codes, the upper 8 bits of the pointer must be initialized to zero.

4.5.12.12 HDLC Event Register

The SCC event register (SCCE) is called the HDLC event register when the SCC is operating as an HDLC controller. It is an 8-bit register used to report events recognized by the HDLC channel and to generate interrupts. Upon recognition of an event, the HDLC controller sets its corresponding bit in the HDLC event register. Interrupts generated by this register may be masked in the HDLC mask register.

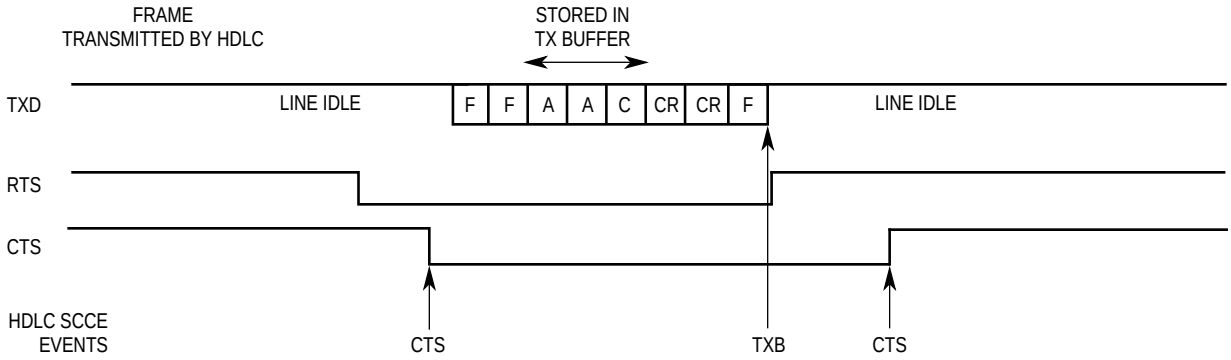
The HDLC event register is a memory-mapped register that may be read at any time. A bit is cleared by writing a one; writing a zero does not affect a bit's value. More than one bit may be cleared at a time. All unmasked bits must be cleared before the CP will clear the internal interrupt request. This register is cleared at reset.

An example of the timing of various events in the HDLC event register is shown in Figure 4-29.



NOTES:
1. RXB event assumes receive buffers are 6 bytes each.
2. The second IDL event occurs after 15 ones are received in a row.

LEGEND:
F = Flag A = Address byte C = Control byte I = Information byte CR = CRC byte



NOTE: TXB event shown assumes all three bytes were put into a single buffer.
Example shows one additional opening flag. This is programmable.

Figure 4-29. HDLC Interrupt Events Example

7	6	5	4	3	2	1	0
CTS	CD	IDL	TXE	RXF	BSY	TXB	RXB

CTS—Clear-To-Send Status Changed

A change in the status of the $\overline{CTS}$ line was detected on the HDLC channel. The SCC status register may be read to determine the current status.

CD—Carrier Detect Status Changed

A change in the status of the $\overline{CD}$ line was detected on the HDLC channel. The SCC status register may be read to determine the current status.

IDL—IDLE Sequence Status Changed

A change in the status of the serial line was detected on the HDLC channel. The SCC status register may be read to determine the current status.

TXE—Tx Error

An error ($\overline{\text{CTS}}$ lost or underrun) occurred on the transmitter channel.

RXF—Rx Frame

A complete frame has been received on the HDLC channel. This bit is set no sooner than two receive clocks after receipt of the last bit of the closing flag.

BSY—Busy Condition

A frame was received and discarded due to lack of buffers.

TXB—Tx Buffer

A buffer has been transmitted on the HDLC channel. This bit is set no sooner than when the second-to-last bit of the closing flag begins its transmission, if the buffer is the last in the frame. Otherwise, it is set after the last byte of the buffer has been written to the transmit FIFO.

RXB—Rx Buffer

A buffer has been received on the HDLC channel that was not a complete frame. This bit will only be set if the I bit in the Tx BD was set.

4.5.12.13 HDLC Mask Register

The SCC mask register (SCCM) is referred to as the HDLC mask register when the SCC is operating as an HDLC controller. It is an 8-bit read-write register that has the same bit formats as the HDLC event register. If a bit in the HDLC mask register is a one, the corresponding interrupt in the event register will be enabled. If the bit is zero, the corresponding interrupt in the event register will be masked. This register is cleared upon reset.

4.5.13 BISYNC Controller

The byte-oriented binary synchronous communication (BISYNC) protocol was originated by IBM for use in networking products. The three classes of BISYNC frames are transparent, non-transparent with header, and non-transparent without header (see Figure 4-30). The transparent mode in BISYNC allows full binary data to be transmitted, with any possible character pattern allowed. Each class of frame starts with a standard two octet synchronization pattern and ends with a block check code (BCC). The end of text character (ETX) is used to separate the text and BCC fields.

NON TRANSPARENT WITH HEADER

SYN1	SYN2	SOH	HEADER	STX	TEXT	ETX	BCC

NON TRANSPARENT WITHOUT HEADER

SYN1	SYN2	STX	TEXT	ETX	BCC

TRANSPARENT

SYN1	SYN2	DLE	STX	TRANSPARENT TEXT	DLE	ETX	BCC

Figure 4-30. Typical BISYNC Frames

The bulk of the frame is divided into fields whose meaning depends on the frame type. The BCC is either a 16-bit CRC (CRC-16) format if 8-bit characters are used or a longitudinal check (a sum check) in combination with vertical redundancy check (parity) if 7-bit characters are used. In transparent operation, to allow the BISYNC control characters to be present in the frame as valid text data, a special character (DLE) is defined, which informs the receiver that the character following the DLE is a text character, not a control character (from the control character table). If a DLE is transmitted as valid data, it must be preceded by a DLE character. This procedure is sometimes called byte-stuffing.

The physical layer of the BISYNC communications link must provide a means of synchronizing the receiver and transmitter, which is usually accomplished by sending at least one pair of synchronization characters prior to every frame.

BISYNC has the unusual property that a transmit underrun need not be an error. If an underrun occurs, the synchronization pattern is transmitted until data is once again ready to transmit. The receiver discards the additional synchronization characters as they are received, provided the V bit is set in the BISYNC-BISYNC SYNC register. In non-transparent operation, all synchronization characters (SYNCs) are discarded if the V bit is set in the BISYNC-BISYNC SYNC register. In transparent operation, all DLE-SYNC pairs are discarded. (Note that correct operation in this case assumes that, on the transmit side, the underrun does not occur between the DLE and its following character, a failure mode prevented in the MC68302.)

By appropriately setting the SCC mode register, any of the SCC channels may be configured to function as a BISYNC controller. The BISYNC controller handles the basic functions of the BISYNC protocol in normal mode and in transparent mode.

The SCC in BISYNC mode can work with IDL, GCI (IOM2), PCM highway, or NMSI interfaces. When the SCC in BISYNC mode is used with a modem interface (NMSI), the SCC outputs are connected directly to the external pins. The modem interface uses seven dedicated pins: transmit data (TXD), receive data (RXD) receive clock (RCLK), transmit clock (TCLK), carrier detect ($\overline{\text{CD}}$), clear to send ($\overline{\text{CTS}}$), and request to send ($\overline{\text{RTS}}$). Other modem lines can be supported using the parallel I/O pins.

The BISYNC controller consists of separate transmit and receive sections whose operations are asynchronous with the M68000 core and may be either synchronous or asynchronous with respect to the other SCCs. Each clock can be supplied from either the internal baud

rate generator or from external pins. More information on the baud rate generator is available in 4.5.2 SCC Configuration Register (SCON).

- Flexible Data Buffers
- Eight Control Character Recognition Registers
- Automatic SYNC1 and SYNC2 Detection
- SYNC/DLE Stripping and Insertion
- CRC16 and LRC Generation/Checking
- Parity (VRC) Generation/Checking
- Supports BISYNC Transparent Operation (Use of DLE Characters)
- Supports Promiscuous (Totally Transparent) Reception and Transmission
- Maintains Parity Error Counter
- External SYNC Support
- Reverse Data Mode
- Four Commands

4.5.13.1 Bisync Channel frame Transmission Processing

The BISYNC transmitter is designed to work with almost no intervention from the M68000 core. When the M68000 core enables the BISYNC transmitter, it will start transmitting SYN1–SYN2 pairs (located in the data synchronization register) or idle as programmed in the BISYNC mode register. The BISYNC controller polls the first buffer descriptor (BD) in the transmit channel's BD table. When there is a message to transmit, the BISYNC controller will fetch the data from memory and start transmitting the message (after first transmitting the SYN1–SYN2 pair).

When a BD's data has been completely transmitted, the last in message (L) bit is checked. If both the L bit and transmit BCS bit are set in that BD, the BISYNC controller will append the CRC16/LRC. Subsequently, the BISYNC controller writes the message status bits into the BD and clears the ready bit. It will then start transmitting SYN1–SYN2 pairs or IDLEs as programmed in the BISYNC mode register. When the end of the current BD has been reached and the last bit is not set (working in multibuffer mode), only the ready bit is cleared. In both cases, an interrupt is issued according to the interrupt (I) bit in the BD. By appropriately setting the I bit in each BD, interrupts can be generated after the transmission of each buffer, a specific buffer, or each block. The BISYNC controller will then proceed to the next BD in the table.

If no additional buffers have been presented to the BISYNC controller for transmission, an in-frame underrun is detected, and the BISYNC controller begins transmitting SYNCs. If the BISYNC controller was in transparent mode, the BISYNCs controller transmits DLE-SYNC pairs. This case is not an error. However, if an underrun occurs within a buffer, the BISYNC controller will transmit a SYN1–SYN2 pair followed by either SYNCs or idles according to the SYNF bit in the BISYNC mode register. This case is an underrun error and is described further in 4.5.13.8 BISYNC Error-Handling Procedure.

Characters are included in the block check sequence (BCS) calculation on a per-buffer basis. Each buffer can be independently programmed to be included or excluded from the BCS calculation, and any characters to be excluded from the BCS calculation must reside in a separate buffer. The BISYNC controller can reset the BCS generator before transmitting a specific buffer. When functioning in transparent mode, the BISYNC controller automatically inserts a DLE before transmitting a DLE character. In this case, only one DLE is used in the calculation of the BCS.

The IMP may also be used to transmit characters in a promiscuous (totally transparent) mode. See 4.5.16 Transparent Controller.

4.5.13.2 Bisync Channel Frame Reception Processing

Although the BISYNC receiver is designed to work with almost no intervention from the M68000 core, it allows user intervention on a per-byte basis if necessary. The BISYNC receiver can perform CRC16, longitudinal redundancy check (LRC), or vertical redundancy check (VRC) checking, SYNC stripping in normal mode, DLE-SYNC stripping and stripping of the first DLE in DLE-DLE pairs in transparent mode, and control character recognition. A control character is one belonging to the control characters shown in Figure 4-31.

When the M68000 core enables the BISYNC receiver, it will enter hunt mode. In this mode, as data is shifted into the receiver shift register one bit at a time, the contents of the register are compared to the contents of the SYN1–SYN2 fields in the data synchronization register. If the two are not equal, the next bit is shifted in, and the comparison is repeated. When the registers match, the hunt mode is terminated, and character assembly begins. The BISYNC controller is now character synchronized and will perform SYNC stripping and message reception. The BISYNC controller will revert to the hunt mode when it is issued the ENTER HUNT MODE command, upon recognition of some error condition, or upon reception of an appropriately defined control character.

When receiving data, the BISYNC controller updates the BCS bit (CR) in the BD for every byte transferred. When the data buffer has been filled, the BISYNC controller clears the empty (E) bit in the BD and generates an interrupt if the interrupt (I) bit in the BD is set. If the incoming data exceeds the length of the data buffer, the BISYNC controller will fetch the next BD in the table and, if it is empty, will continue to transfer data to this BD's associated data buffer.

When a BCS is received, it is checked and written to the data buffer. The BISYNC controller sets the last bit, writes the message status bits into the BD, and clears the empty bit. Then it generates a maskable interrupt, indicating that a block of data has been received and is in memory. Note that the SYNC in the nontransparent mode or DLE-SYNC pairs in the transparent mode (i.e., an underrun condition) are not included in the BCS calculations.

The IMP may also be used to receive characters in a promiscuous (totally transparent) mode. See 4.5.16 Transparent Controller.

4.5.13.3 Bisync Memory Map

When configured to operate in BISYNC mode, the IMP overlays the structure listed in Table 4-8 onto the protocol-specific area of that SCC parameter RAM. Refer to 2.8 MC68302

Memory Map for the placement of the three SCC parameter RAM areas and Table 4-5 for the other parameter RAM values.

Table 4-9. BISYNC Specific Parameter RAM

Address	Name	Width	Description
SCC Base + 9C	RCRC	Word	Temp Receive CRC
SCC Base + 9E	CRCC	Word	CRC Constant
SCC Base + A0 #	PRCRC	Word	Preset Receiver CRC 16/LRC
SCC Base + A2	TCRC	Word	Temp Transmit CRC
SCC Base + A4 #	PTCRC	Word	Preset Transmitter CRC 16/LRC
SCC Base + A6	RES	Word	Reserved
SCC Base + A8	RES	Word	Reserved
SCC Base + AA #	PAREC	Word	Receive Parity Error Counter
SCC Base + AC #	BSYNC	Word	BISYNC SYNC Character
SCC Base + AE #	BDLE	Word	BISYNC DLE Character
SCC Base + B0 #	CHARACTER1	Word	CONTROL Character 1
SCC Base + B2 #	CHARACTER2	Word	CONTROL Character 2
SCC Base + B4 #	CHARACTER3	Word	CONTROL Character 3
SCC Base + B6 #	CHARACTER4	Word	CONTROL Character 4
SCC Base + B8 #	CHARACTER5	Word	CONTROL Character 5
SCC Base + BA #	CHARACTER6	Word	CONTROL Character 6
SCC Base + BC #	CHARACTER7	Word	CONTROL Character 7
SCC Base + BE #	CHARACTER8	Word	CONTROL Character 8

Initialized by the user (M68000 core).

The M68000 core configures each SCC to operate in one of four protocols by the MODE1–MODE0 bits in the SCC mode register. MODE1–MODE0 = 11 selects the BISYNC mode of operation. The SYN1–SYN2 synchronization characters are programmed in the data synchronization register (see 4.5.4 SCC Data Synchronization Register (DSR)).

The BISYNC controller uses the same basic data structure as the other protocol controllers. Receive and transmit errors are reported through their respective BDs. The status of the line is reflected in the SCC status register, and a maskable interrupt is generated upon each status change.

There are two basic ways of handling the BISYNC channels. First, data may be inspected on a per-byte basis, with the BISYNC controller interrupting the M68000 core upon receipt of every byte of data. Second, the BISYNC controller may be operated so that software is only necessary for handling the first two to three bytes of data; subsequent data (until the end of the block) can be handled by the BISYNC controller without interrupting the M68000 core. See 4.5.13.14 Programming the BISYNC Controllers for more information.

4.5.13.4 BISYNC Command Set

The following commands are issued to the command register.

STOP TRANSMIT Command

After a hardware or software reset and the enabling of the channel using the SCC mode register, the channel is in the transmit enable mode and starts polling the first BD in the table approximately every eight transmit clocks.

The STOP TRANSMIT command aborts transmission after the contents of the FIFO are transmitted (up to three bytes) without waiting until the end of the buffer is reached. The TBD# is not advanced. SYNC characters consisting of SYNC-SYNC or DLE-SYNC pairs (according to the transmitter mode) will be continually transmitted until transmission is re-enabled by issuing the RESTART TRANSMIT command. The STOP TRANSMIT com-

mand may be used when it is necessary to abort transmission and transmit an EOT control sequence. The EOT sequence should be the first buffer presented to the BISYNC controller for transmission after re-enabling transmission.

The STOP TRANSMIT command must be issued before the SCC mode register is used to disable the transmitter if the transmitter is to be re-enabled at a later time.

NOTE

The BISYNC controller will remain in the transparent or normal mode after receiving the STOP TRANSMIT or RESTART TRANSMIT commands.

RESTART TRANSMIT Command

The RESTART TRANSMIT command is used to begin or resume transmission from the current Tx BD number (TBD#) in the channel's Tx BD table. When this command is received by the channel, it will start polling the ready bit in this BD. This command is expected by the BISYNC controller after a STOP TRANSMIT command, after the STOP TRANSMIT command and the disabling of the channel in its mode register, or after a transmitter error (underrun or CTS lost) occurs.

If the transmitter is being re-enabled, the RESTART TRANSMIT command must be used and should be followed by the enabling of the transmitter in the SCC mode register.

RESET BCS CALCULATION Command

The RESET BCS CALCULATION command resets the receive BCS accumulator immediately. For example, it may be used to reset the BCS after recognizing a control character, signifying that a new block is commencing (such as SOH).

ENTER HUNT MODE Command

After a hardware or software reset and the enabling of the channel in the SCC mode register, the channel is in the receive enable mode and will use the first BD in the table.

The ENTER HUNT MODE command is used to force the BISYNC controller to abort reception of the current block, generate an RX interrupt (if enabled) as the buffer is closed, and enter the hunt mode. In hunt mode, the BISYNC controller continually scans the input data stream for the SYN1–SYN2 sequence as programmed in the data synchronization register. After receiving the command, the current receive buffer is closed, and the BCS is reset. Message reception continues using the next BD.

If an enabled receiver has been disabled (by clearing ENR in the SCC mode register), the ENTER HUNT MODE command must be given to the channel before setting ENR again.

4.5.13.5 BISYNC Control Character Recognition

The BISYNC controller can recognize special control characters. These characters are used to “customize” the BISYNC protocol implemented by the BISYNC controller and may be used to aid its operation in a DMA-oriented environment. Their main use is for receive buffers longer than one byte. In single-byte buffers, each byte can easily be inspected, and control character recognition should be disabled.

The purpose of the control characters table is to enable automatic recognition (by the BISYNC controller) of the end of the current block. See 4.5.13.14 Programming the BISYNC Controllers for more information. Since the BISYNC controller imposes no restrictions on the format of the BISYNC blocks, user software must respond to the received characters and inform the BISYNC controller of mode changes and certain protocol events (e.g., resetting the BCS). However, correct use of the control characters table allows the remainder of the block to be received without interrupting the user software.

Up to eight control characters may be defined. These characters inform the BISYNC controller that the end of the current block has been reached and whether a BCS is expected following this character. For example, the end of text (ETX) character implies both an end of block (ETB) and a BCS should be received. An enquiry (ENQ) character designates end of block without a subsequent BCS. All the control characters are written into the data buffer.

The BISYNC controller uses a table of 16-bit entries to support control character recognition. Each entry consists of the control character, an end-of-table bit, a BCS expected bit, and a hunt mode bit. The control characters table is shown in Figure 4-31. To disable the entire control characters table, write $8000 to the first table entry.

	15	14	13	12	11	10	9	8	7 … 0
OFFSET + 0	E	B	H						CHARACTER1
OFFSET +2	E	B	H						CHARACTER2
OFFSET + 4	E	B	H						CHARACTER3
•									
•									
•									
OFFSET + E	E	B	H						CHARACTER8

Figure 4-31. BISYNC Control Characters Table

CHARACTER8–CHARACTER1—Control Character Value

These fields define control characters.

NOTE

When using 7-bit characters with parity, the parity bit should be included in the control character value.

E—End of Table

0 = This entry is valid. The lower eight bits will be checked against the incoming character.

1 = The entry is not valid. No valid entries exist beyond this entry.

NOTE

In tables with eight control characters, E should be zero in all eight positions.

B—BCS Expected

0 = The character is written into the receive buffer. The buffer is then closed.
1 = The character is written into the receive buffer. The receiver waits for one LRC or two CRC bytes of BCS and then closes the buffer. This should be used for ETB, ETX, and ITB.

NOTE

A maskable interrupt is generated after the buffer is closed.

H—Enter Hunt Mode

0 = The BISYNC controller will maintain character synchronization after closing this buffer.
1 = The BISYNC controller will enter hunt mode after closing the buffer. When the B bit is set, the controller will enter hunt mode after the reception of the BCS.

4.5.13.6 BSYNC-BISYNC SYNC Register

The 16-bit, memory-mapped, read-write BSYNC register is used to define the BISYNC stripping and insertion of the SYNC character. When an underrun occurs during message transmission, the BISYNC controller will insert SYNC characters until the next data buffer is available for transmission. When the BISYNC receiver is not in hunt mode and a SYN1 character has been received and discarded, the receiver will discard a SYNC character if the valid (V) bit is set.

NOTE

Normal Bisync operation requires that SYN1=SYN2=SYNC (and the V bit must be set in the BISYNC SYNC register).

When using 7-bit characters with parity, the parity bit should be included in the SYNC register value.

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

4.5.13.7 BDLE-BISYNC DLE Register

The 16-bit, memory-mapped, read-write BDLE register is used to define the BISYNC stripping and insertion of the DLE character. When the BISYNC controller is in transparent mode and an underrun occurs during message transmission, the BISYNC controller inserts DLE-SYNC pairs until the next data buffer is available for transmission.

When the BISYNC receiver is in transparent mode and a DLE character is received, the receiver discards this character and excludes it from the BCS if the valid (V) bit is set. If the second (next) character is a SYNC character, the BISYNC controller discards it and ex-

cludes it from the BCS. If the second character is a DLE, the BISYNC controller will write it to the buffer and include it in the BCS. If the character is not a DLE or SYNC, the BISYNC controller will examine the control characters table and act accordingly. If the character is not in the table, the buffer will be closed with the DLE follow character error (DL) bit set. If the V bit is not set, the receiver will treat the character as a normal character.

NOTE

When using 7-bit characters with parity, the parity bit should be included in the DLE register value.

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

4.5.13.8 BISYNC Error-Handling Procedure

The BISYNC controller reports message reception and transmission error conditions using the channel BDs, the error counters, and the BISYNC event register. The modem interface lines can also be directly monitored in the SCC status register.

Transmission Errors:

1. Transmitter Underrun. When this error occurs, the channel terminates buffer transmission, closes the buffer, sets the underrun (UN) bit in the BD, and generates the TXE interrupt (if enabled). The channel resumes transmission after the reception of the RESTART TRANSMIT command. Underrun cannot occur between frames or during a DLE-XXX pair in transparent mode. The FIFO size is three bytes in BISYNC.
2. Clear-To-Send Lost During Message Transmission. When this error occurs and the channel is not programmed to control this line with software, the channel terminates buffer transmission, closes the buffer, sets the CTS lost (CT) bit in the BD, and generates the TXE interrupt (if enabled). The channel will resume transmission after the reception of the RESTART TRANSMIT command.

Reception Errors:

1. Overrun Error. The BISYNC controller maintains an internal three-byte FIFO for receiving data. The CP begins programming the SDMA channel (if the data buffer is in external memory) and updating the CRC when the first word is received into the FIFO. If a FIFO overrun occurs, the BISYNC controller writes the received data byte to the internal FIFO over the previously received byte. The previous character and its status bits are lost. Following this, the channel closes the buffer, sets the overrun (OV) bit in the BD, and generates the RX interrupt (if enabled). The receiver then enters hunt mode immediately.
2. Carrier Detect Lost During Message Reception. When this error occurs and the channel is not programmed to control this line with software, the channel terminates message reception, closes the buffer, sets the carrier detect lost (CD) bit in the BD, and generates the RX interrupt (if enabled). This error is the highest priority; the rest of the message is lost and no other errors are checked in the message. The receiver then enters hunt mode immediately.
3. Parity Error. When this error occurs, the channel writes the received character to the

buffer and sets the PR bit in the BD. The channel terminates message reception, closes the buffer, sets the PR bit in the BD, and generates the RX interrupt (if enabled). The channel also increments the parity error counter (PAREC), and the receiver then enters hunt mode immediately.

4. CRC Error. The channel updates the CRC error (CR) bit in the BD every time a character is received, with a byte delay (eight serial clocks) between the status update and the CRC calculation. When using control character recognition to detect the end of the block and cause the checking of the CRC that follows, the channel closes the buffer, sets the CR bit in the BD, and generates the RX interrupt (if enabled).

Error Counter

The CP main controller maintains one 16-bit (modulo–2**16) error counter for each BISYNC controller. It can be initialized by the user when the channel is disabled. The counter is as follows:

—PAREC—Parity Error Counter (on received characters)

4.5.13.9 BISYNC Mode Register

Each SCC mode register is a 16-bit, memory-mapped, read-write register that controls the SCC operation. The term BISYNC mode register refers to the protocol-specific bits (15–6) of the SCC mode register when that SCC is configured for BISYNC. The read-write BISYNC mode register is cleared by reset.

15	14	13	12	11	10	9	8	7	6	5 0
PM	EXSYN	NTSYN	REVD	BCS	—	RTR	RBCS	SYNF	ENC	COMMON SCC MODE BITS

PM—Parity Mode

0 = Odd Parity
1 = Even Parity

This bit is valid when the BCS bit is cleared. When odd parity is selected, the transmitter will count the number of ones in the 7-bit data character. If the total is not an odd number, then the parity bit is made equal to one to make an odd number of ones. Then, if the receiver counts an even number of ones, an error in transmission has occurred. In the same manner, for even parity, an even number must result from the calculation performed at both ends of the line.

EXSYN—External Sync Mode

When this mode is selected, the receiver expects external logic to indicate the beginning of the data field using the $\overline{\text{CD1}}$/L1SY1 pin, if SCC1 is used, and the $\overline{\text{CD2}}$ and $\overline{\text{CD3}}$ pins, respectively, if SCC2 or SCC3 are used in this mode. In this mode, there will be no carrier detect function for the SCC.

When the channel is programmed to work through the serial channels physical interface (IDL or GCI) and EXSYN is set, the layer-1 logic carries out the synchronization using the L1SY1 pin. In PCM mode, the L1SY1–L1SY0 pins are used.

In NMSI mode, the $\overline{\text{CD}}$ pins (and the $\overline{\text{CD}}$ timing) are used to synchronize the data. $\overline{\text{CD}}$ should be asserted on the second data bit of the frame when used as a sync.

If this bit is cleared, the BISYNC controller will look for the SYN1–SYN2 sequence in the data synchronization register.

NTSYN—No Transmit SYNC

When this bit is set, the SCC operates in a promiscuous, totally transparent mode. See 4.5.16 Transparent Controller for details.

REVD—Reverse DATA

When this bit is set, the receiver and transmitter will reverse the character bit order, transmitting the most significant bit first. This bit is valid in promiscuous mode.

BCS—Block Check Sequence

0 = LRC
For even LRC, the PRCRC and PTCRC preset registers in the BISYNC-specific parameter RAM should be initialized to zero before the channel is enabled. For odd LRC, the PRCRC and PTCRC registers should be initialized to ones. The LRC is formed by the Exclusive OR of each 7-bits of data (not including synchronization characters), and the parity bit is added after the final LRC calculation.

The receiver will check character parity when BCS is programmed to LRC and the receiver is not in transparent mode. The transmitter will transmit character parity when BCS is programmed to LRC and the transmitter is not in transparent mode. Use of parity in BISYNC assumes the use of 7-bit data characters.

1 = CRC16
The PRCRC and PTCRC preset registers should be initialized to a preset value of all zeros or all ones before the channel is enabled. In both cases, the transmitter sends the calculated CRC non-inverted, and the receiver checks the CRC against zero. Eight-bit characters (without parity) are configured when CRC16 is chosen. The CRC16 polynomial is as follows:

$$X^{16} + X^{15} + X^{2} + 1$$

Bit 10—Reserved for future use.

RTR—Receiver Transparent Mode

0 = The receiver is placed in normal mode with SYNC stripping and control character recognition operative.
1 = The receiver is placed in transparent mode. SYNCs, DLEs, and control characters are only recognized after a leading DLE character. The receiver will calculate the CRC16 sequence, even if programmed to LRC while in transparent mode. PRCRC should be first initialized to the CRC16 preset value before setting this bit.

RBCS—Receive Block Check Sequence

The BISYNC receiver internally stores two BCS calculations with a byte delay (eight serial clocks) between them. This enables the user to examine a received data byte and then decide whether or not it should be part of the BCS calculation. This is useful when control

character recognition and stripping is desired to be performed in software. The bit should be set (or reset) within the time taken to receive the following data byte. When this bit is reset, the BCS calculations exclude the latest fully received data byte. When RBCS is set, the BCS calculations continue normally.

0 = Disable receive BCS
1 = Enable receive BCS

SYNF—Transmit SYN1–SYN2 or IDLE between Messages and Control the RTS Pin

0 = Send ones between messages; $\overline{\text{RTS}}$ is negated between messages. The BISYNC controller can transmit ones in both NRZ and NRZI encoded formats.
1 = Send SYN1–SYN2 pairs between messages; $\overline{\text{RTS}}$ is always asserted.

ENC—Data Encoding Format

0 = Non-return to zero (NRZ). A one is a high level; a zero is a low level.
1 = Non-return to zero inverted (NRZI). A one is represented by no change in the level; a zero is represented by a change in the level. The receiver decodes NRZI, but a clock must be supplied. The transmitter encodes NRZI.

COMMON SCC MODE BITS—See 4.5.3 SCC Mode Register (SCM) for a description of the DIAG1, DIAG0, ENR, ENT, MODE1, and MODE0 bits.

4.5.13.10 BISYNC Receive Buffer Descriptor (Rx BD)

- The CP reports information about the received data for each buffer using BD. The Rx BD is shown in Figure 4-32. The CP closes the current buffer, generates a maskable interrupt, and starts to receive data into the next buffer after one of the following events:
- Receiving a user-defined control character
- Detecting an error
- Detecting a full receive buffer
- Issuing the ENTER HUNT MODE command

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET + 0	E	X	W	I	C	B	—	—	—	—	—	DL	PR	CR	OV	CD
OFFSET +2	DATA LENGTH															
OFFSET +4	RX BUFFER POINTER (24-bits used, upper 8 bits must be 0)															
OFFSET + 6																

Figure 4-32. BISYNC Receive Buffer Descriptor

The first word of the Rx BD contains control and status bits.

E—Empty

0 = The data buffer associated with this BD has been filled with received data, or data reception has been aborted due to an error condition. The M68000 core is free to examine or write to any fields of this BD.
1 = The data buffer associated with this BD is empty. This bit signifies that the BD and its associated buffer are available to the CP. After it sets this bit, the M68000 core

should not write to any fields of this BD when this bit is set. The empty bit will remain set while the CP is currently filling the buffer with received data.

X—External Buffer

0 = The buffer associated with this BD is in internal dual-port RAM.
1 = The buffer associated with this BD is in external memory.

W—Wrap (Final BD in Table)

0 = This is not the last BD in the Rx BD table.
1 = This is the last BD in the Rx BD table. After this buffer has been used, the CP will receive incoming data into the first BD in the table. Setting this bit allows the use of fewer than eight BD to conserve internal RAM.

NOTE

The user is required to set the wrap bit in one of the first eight BDs; otherwise, errant behavior may occur.

I—Interrupt

0 = No interrupt is generated after this buffer has been used.
1 = The RX bit in the BISYNC event register will be set when this buffer has been closed by the BISYNC controller, which can cause an interrupt.

The following status bits are written by the CP after the received data has been placed into the associated data buffer.

C—Control Character

The last byte in the buffer is a user-defined control character.

0 = The last byte of this buffer does not contain a control character.
1 = The last byte of this buffer contains a control character.

B—BCS Received

The last bytes in the buffer contain the received BCS.

0 = This buffer does not contain the BCS.
1 = This buffer contains the BCS. A control character may also reside one byte prior to this BCS.

Bits 9–5—Reserved for future use.

DL—DLE Follow Character Error

While in transparent mode, a DLE character was received, and the next character was not DLE, SYNC, or a valid entry in the control characters table.

PR—Parity Error

A character with a parity error was received and is the last byte of this buffer.

CR—BCS Error

BCS error (CR) is updated every time a byte is written into the buffer. The CR bit includes the calculation for the current byte. By clearing the RBCS bit in the BISYNC mode register within eight serial clocks, the user can exclude the current character from the message BCS calculation. The data length field may be read to determine the current character's position.

OV—Overrun

A receiver overrun occurred during message reception.

CD—Carrier Detect Lost

The carrier detect signal was negated during message reception.

Data Length

The data length is the number of octets that the CP has written into this BD's data buffer, including the BCS (if selected). In BISYNC mode, the data length should initially be set to zero by the user and is incremented each time a received character is written to the data buffer.

NOTE

The actual buffer size should be greater than or equal to the MR-BLR.

Rx Buffer Pointer

The receive buffer pointer, which always points to the first location of the associated data buffer, may be even or odd. The buffer may reside in either internal or external memory.

NOTE

For correct operation of the function codes, the upper 8 bits of the pointer must be initialized to zero.

4.5.13.11 BISYNC Transmit Buffer Descriptor (Tx BD).

Data is presented to the CP for transmission on an SCC channel by arranging it in buffers referenced by the channel's Tx BD table. The CP confirms transmission (or indicates error conditions) using the BDs to inform the processor that the buffers have been serviced. The Tx BD is shown in Figure 4-33.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET + 0	R	X	W	I	L	TB	B	BR	TD	TR	—	—	—	—	UN	CT
OFFSET + 2	DATA LENGTH															
OFFSET + 4	TX BUFFER POINTER (24-bits used, upper 8 bits must be 0)															
OFFSET +6																

Figure 4-33. BISYNC Transmit Buffer Descriptor

The first word of the Tx BD contains status and control bits. These bits are prepared by the user before transmission and are set by the CP after the buffer has been transmitted.

R—Ready

0 = This buffer is not currently ready for transmission. The user is free to manipulate this BD (or its associated buffer). The CP clears this bit after the buffer has been fully transmitted or after an error condition has been encountered.
1 = The data buffer has been prepared for transmission by the user (but not yet transmitted). No fields of this BD may be written by the user once this bit is set.

X—External Buffer

0 = The buffer associated with this BD is in internal dual-port RAM.
1 = The buffer associated with this BD is in external memory.

W—Wrap (Final BD in Table)

0 = This is not the last BD in the Tx BD table.
1 = This is the last BD in the Tx BD table. After this buffer has been used, the CP will transmit data from the first BD in the table.

NOTE

The user is required to set the wrap bit in one of the first eight BDs; otherwise, errant behavior may occur.

I—Interrupt

0= No interrupt is generated after this buffer has been serviced.
1= Either TX or TXE in the BISYNC event register will be set when this buffer has been serviced by the CP, which can cause an interrupt.

L—Last in Message

0 = The last character in the buffer is not the last character in the current block.
1 = The last character in the buffer is the last character in the current block. The transmitter will enter (remain in) normal mode after sending the last character in the buffer and the BCS (if enabled).

TB—Transmit BCS

This bit is valid only when the L bit is set.

0 = Transmit the SYN1–SYN2 sequence or IDLE (according to the SYNF bit in the BISYNC mode register) after the last character in the buffer.
1 = Transmit the BCS sequence after the last character. The BISYNC controller will also reset the BCS generator after transmitting the BCS sequence.

B—BCS Enable

0 = Buffer consists of characters to be excluded from the BCS accumulation.
1 = Buffer consists of characters to be included in the BCS accumulation.

BR—BCS Reset

0 = The BCS accumulation is not reset.
1 = The transmitter BCS accumulation is reset (used for STX or SOH) before sending the data buffer.

TD—Transmit DLE

0 = No automatic DLE transmission before the data buffer.
1 = The transmitter will transmit a DLE character before sending the data buffer, which saves writing the first DLE to a separate data buffer when working in transparent mode. The transmitter also checks for a DLE character in the middle of a buffer. If found, it will insert an extra DLE (if the V-bit is set in the BISYNC SYNC register).

TR—Transparent Mode

0 = The transmitter will enter (remain in) the normal mode after sending the data buffer. In this mode, the transmitter will automatically insert SYNCs in an underrun condition.
1 = The transmitter enters or remains in transparent mode after sending the data buffer. In this mode, the transmitter automatically inserts DLE-SYNC pairs in the underrun condition. Underrun occurs when the BISYNC controller finishes a buffer with L set to zero and the next BD is not available. The transmitter also checks all characters before sending them; if a DLE is detected, another DLE is automatically sent. The user must insert a DLE or program the BISYNC controller to insert it (using TD) before each control character required. The transmitter will calculate the CRC16 BCS even if the BCS bit in the BISYNC mode register is programmed to LRC. The PTCRC should be initialized to the CRC16 preset before setting this bit.

The following status bits are written by the CP after it has finished transmitting the associated data buffer.

UN—Underrun

The BISYNC controller encountered a transmitter underrun condition while transmitting the associated data buffer.

CT—CTS Lost

$\overline{\text{CTS}}$ in NMSI mode or L1GR in IDL/GCI mode was lost during message transmission.

Data Length

The data length is the number of octets that the CP should transmit from this BD's data buffer. It is never modified by the CP. The data length should be greater than zero.

Tx Buffer Pointer

The transmit buffer pointer, which always points to the first byte of the associated data buffer, may be even or odd. The buffer may reside in either internal or external memory.

NOTE

For correct operation of the function codes, the upper 8 bits of the pointer must be initialized to zero.

4.5.13.12 BISYNC Event Register

The SCC event register (SCCE) is referred to as the BISYNC event register when the SCC is programmed as a BISYNC controller. It is an 8-bit register used to report events recog-

nized by the BISYNC channel and to generate interrupts. On recognition of an event, the BISYNC controller sets the corresponding bit in the BISYNC event register. Interrupts generated by this register may be masked in the BISYNC mask register.

The BISYNC event register is a memory-mapped register that may be read at any time. A bit is cleared by writing a one (writing a zero does not affect a bit's value). More than one bit may be cleared at a time. All unmasked bits must be cleared before the CP will negate the internal interrupt request signal. This register is cleared at reset.

7	6	5	4	3	2	1	0
CTS	CD	—	TXE	RCH	BSY	TX	RX

CTS—Clear-To-Send Status Changed

A change in the status of the serial line was detected on the BISYNC channel. The SCC status register may be read to determine the current status.

CD—Carrier Detect Status Changed

A change in the status of the serial line was detected on the BISYNC channel. The SCC status register may be read to determine the current status.

Bit 5—Reserved for future use.

TXE—Tx Error

An error (CTS lost or underrun) occurred on the transmitter channel.

RCH—Receive Character

A character has been received and written to the buffer.

BSY—Busy Condition

A character was received and discarded due to lack of buffers. The receiver will resume reception after an ENTER HUNT MODE command.

TX—Tx Buffer

A buffer has been transmitted. This bit is set on the second to last bit of BCC or data.

RX—Rx Buffer

A complete buffer has been received on the BISYNC channel.

4.5.13.13 BISYNC Mask Register

The SCC mask register (SCCM) is referred to as the BISYNC mask register when the SCC is operating as a BISYNC controller. It is an 8-bit read-write register that has the same bit format as the BISYNC event register. If a bit in the BISYNC mask register is a one, the corresponding interrupt in the event register will be enabled. If the bit is zero, the corresponding interrupt in the event register will be masked. This register is cleared upon reset.

4.5.13.14 Programming the BISYNC Controllers

There are two general techniques that the software may employ to handle data received by the BISYNC controllers. The simplest way is to allocate single-byte receive buffers, request (in the status word in each BD) an interrupt on reception of each buffer (i.e., byte), and implement the BISYNC protocol entirely in software on a byte-by-byte basis. This simple approach is flexible and may be adapted to any BISYNC implementation. The obvious penalty is the overhead caused by interrupts on each received character.

A more efficient method is as follows. Multibyte buffers are prepared and linked to the receive buffer table. Software is used to analyze the first (two to three) bytes of the buffer to determine what type of block is being received. When this has been determined, reception can continue without further intervention to the user's software until a control character is encountered. The control character signifies the end of the block, causing the software to revert back to a byte-by-byte reception mode.

To accomplish this, the RCH bit in the BISYNC mask register should initially be set, enabling an interrupt on every byte of data received. This allows the software to analyze the type of block being received on a byte-by-byte basis. After analyzing the initial characters of a block, the user should either set the receiver transparent mode (RTR) bit in the BISYNC mode register or issue the RESET BCS CALCULATION command. For example, if DLE-STX is received, transparent mode should be entered. By setting the appropriate bit in the BISYNC mode register, the BISYNC controller automatically strips the leading DLE from <DLE-character> sequences. Thus, control characters are only recognized when they follow a DLE character. The RTR bit should be cleared after a DLE-ETX is received.

Alternatively, after receiving an SOH, the RESET BCS CALCULATION command should be issued. This command causes the SOH to be excluded from BCS accumulation and the BCS to be reset. Note that the RBCS bit in the BISYNC mode register (used to exclude a character from the BCS calculation) is not needed here since SYNCs and leading DLEs (in transparent mode) are automatically excluded by the BISYNC controller.

After recognizing the type of block above, the RCH interrupt should be masked. Data reception then continues without further interruption of the M68000 core until the end of the current block is reached. This is defined by the reception of a control character matching that programmed in the receive control characters table.

The control characters table should be set to recognize the end of the block as follows:

Control Characters	E	B	H
ETX	0	1	1
ITB	0	1	0
ETB	0	1	1
ENQ	0	0	0
Next Entry	1	X	X

After the end of text (ETX), a BCS is expected; then the buffer should be closed. Hunt mode should be entered when line turnaround occurs. ENQ characters are used to abort transmis-

sion of a block. For the receiver, the ENQ character designates the end of the block, but no CRC is expected.

Following control character reception (i.e., end of the block), the RCH bit in the BISYNC mask register should be set, re-enabling interrupts for each byte of received data.

4.5.14 DDCMP Controller

The byte-oriented digital data communications message protocol (DDCMP) was originated by DEC for use in networking products. The three classes of DDCMP frames are transparent (or maintenance) messages, data messages, and control messages (see Figure 4-34). Each class of frame starts with a standard two octet synchronization pattern and ends with a CRC. Depending upon the frame type, a separate CRC may be present for the header as well as the data portions of the frame. These CRCs use the same 16-bit generator polynomial as that used in HDLC.

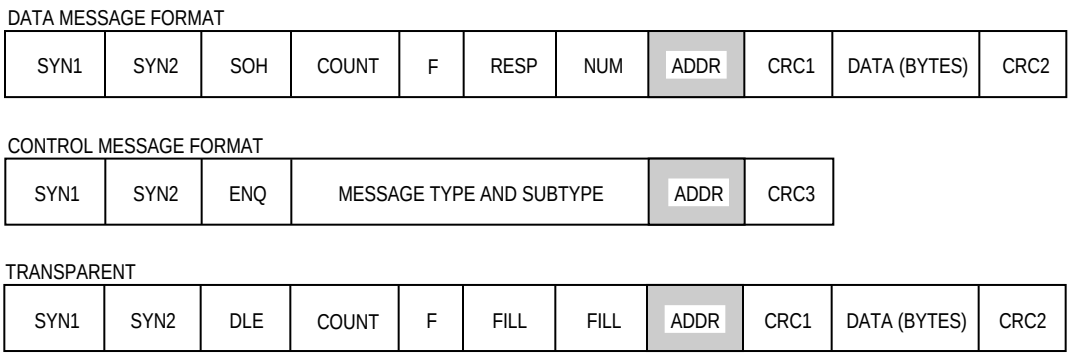


Figure 4-34. Typical DDCMP Frames

The most notable feature of the DDCMP frame is that the frame length is transmitted within the frame itself. Thus, any character pattern can be transmitted in the data field since the character count is responsible for ending the frame, not a special character. For this to work properly, the header containing the frame length must be protected, causing a need for a CRC in the frame header.

The bulk of the frame is divided into fields whose meaning depends on the frame type. Defined control characters are only used in the fixed-length frame headers (the fields between the synchronization octets and the first CRC). The following fields are one byte each: SYN1, SYN2, SOH, RESP, NUM, ADDR, ENQ, DLE, and FILL. The following fields are two bytes each: COUNT + F, CRC1, CRC2, and CRC3. The DATA field is a variable number of bytes, as defined in the COUNT field.

DDCMP communications can be either synchronous or asynchronous, with both types using the same frame format. Synchronous DDCMP frames require the physical layer to transmit the clock along with data over the link. Asynchronous DDCMP frames are composed of asynchronous UART characters, which together form the frame. The receiver and transmitter clocks are not linked; the receiver resynchronizes itself every byte using the start and stop bits of each UART character.

By setting its SCC mode register (SCM), any of the SCC channels may be configured to function as a DDCMP controller. The DDCMP link can be either synchronous (by programming the MODE1–MODE0 bits of the SCC mode register to DDCMP) or asynchronous (by programming the MODE1–MODE0 bits of the SCC mode register to asynchronous and setting the DDCMP bit in the UART mode register). The DDCMP controller handles the basic functions of the DDCMP protocol in both cases.

The SCC in DDCMP mode can work in either IDL, GCI, PCM highway, or NMSI interfaces. When the SCC is used with a modem interface (NMSI), the serial outputs are connected directly to the external pins. The modem interface uses seven dedicated pins: transmit data (TXD), receive data (RXD), receive clock (RCLK), transmit clock (TCLK), carrier detect (CD), clear to send (CTS), and request to send (RTS). Other modem lines can be supported through the parallel I/O pins.

The DDCMP controller consists of separate transmit and receive sections whose operations are asynchronous with the M68000 core and may be either synchronous or asynchronous with respect to the other SCCs. Each clock can be supplied either from the baud rate generator or externally. More information on the baud rate generator is available in 4.5.2 SCC Configuration Register (SCON).

The DDCMP controller key features are as follows:

- Synchronous or Asynchronous DDCMP Links Supported
- Flexible Data Buffers
- Four Address Comparison Registers with Mask
- Automatic Frame Synchronization
- Automatic Message Synchronization by Searching for SOH, ENQ, or DLE
- CRC16 Generation/Checking
- NRZ/NRZI Data Encoding
- Maintenance of Four 16-Bit Error Counters

4.5.14.1 DDCMP Channel Frame Transmission Processing

The DDCMP transmitter is designed to work with almost no intervention from the M68000 core (see Figure 4-35).

When the M68000 core enables the DDCMP transmitter and the link is synchronous, it starts transmitting SYN1–SYN2 pairs (programmed in the data synchronization register) or IDLEs as determined in the DDCMP mode register. The DDCMP controller polls the first buffer descriptor (BD) in the channel's transmit BD table. When there is a message to transmit, the DDCMP controller fetches the data from memory and starts transmitting the message (after first transmitting the SYN1–SYN2 pair when the link is synchronous).

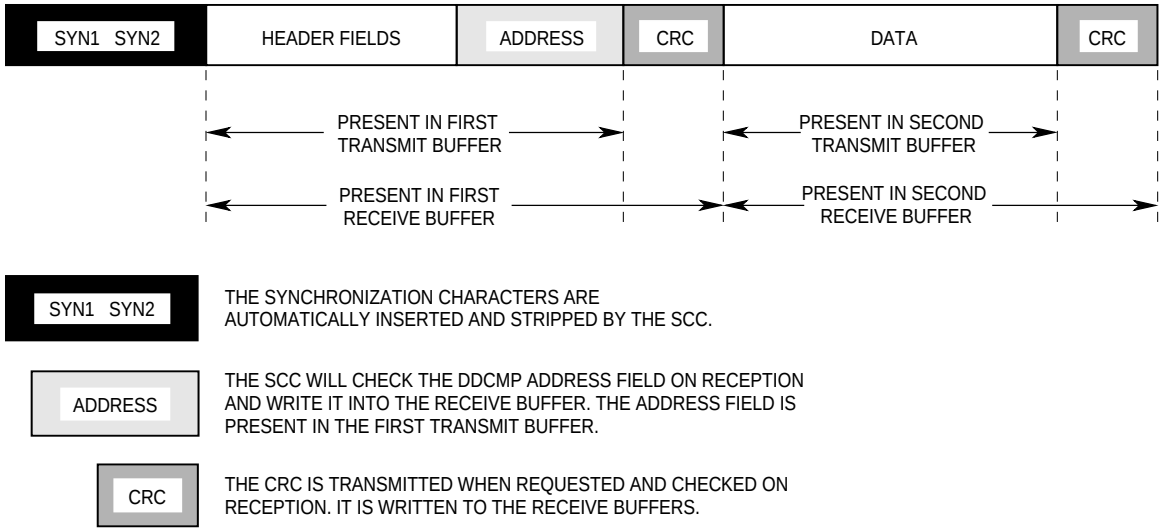


Figure 4-35. DDCMP Transmission/Reception Summary

When a BD has been completely transmitted, the transmit CRC (TC) bit is checked in the BD. If set, the DDCMP controller appends one of the block checks: CRC1, CRC2, or CRC3 for the header field, data message, or control messages, respectively. Next, the DDCMP controller writes the buffer's status bits into the BD and clears the ready bit in the BD. It then proceeds to the next BD in the table. When the last bit (L) is set and the TC bit is set in that BD, the DDCMP controller appends the CRC2 block check to the data field. This bit is also used for transmitting CRC3 in control messages. Next, it writes the buffer status bits into the BD and clears the ready bit. Finally, on synchronous links, either SYN1–SYN2 pairs or IDLEs (as programmed in the DDCMP mode register) are transmitted. When the end of the current BD has been reached and the last bit is not set (working in multibuffer mode or sending back-to-back messages), only the status bits are written. In either case, when a BD has been completely transmitted, an interrupt is issued if the interrupt (I) bit in the BD is set and the event is not masked in the DDCMP mask register. The appropriate setting of the I bit in each BD allows the user to be interrupted after transmission of each buffer, a specific buffer, or each message.

4.5.14.2 DDCMP Channel Frame Reception Processing.

The DDCMP receiver is also designed to work with almost no intervention from the M68000 core (see Figure 4-35).

The DDCMP receiver performs automatic SYN1–SYN2 synchronization on synchronous links and start/stop synchronization on asynchronous links. Automatic message synchronization is achieved by searching for the special starting characters SOH, ENQ, or DLE and making address comparisons with a mask. When the M68000 core enables the DDCMP receiver on synchronous links, it enters hunt mode. In this mode, as data is shifted into the receiver shift register one bit at a time, the contents of the register are compared to the SYN1–SYN2 fields of the data synchronization register (see 4.5.4 SCC Data Synchronization Register (DSR)). If the two are not equal, the next bit is shifted in, and the comparison is repeated. When the registers match, hunt mode is terminated, and character assembly

begins. However, if a character is not SOH, ENQ, DLE, or SYNC, hunt mode is again entered. On asynchronous links, byte synchronization is achieved by the start/stop protocol of the UART. The DDCMP controller is now byte-synchronized and performs SYN1–SYN2 stripping, until receiving one of the three user-defined special starting bytes (SOH for data messages, ENQ for control messages, and DLE for maintenance messages).

If a match is detected, the DDCMP controller fetches the next BD and, if it is empty, starts to transfer the incoming header to the BD's associated data buffer. The DDCMP controller counts the bytes of the fixed-length header and compares the received header address field to the four user-defined values after masking the result with the address mask. When a match is detected, the DDCMP controller continues to transfer the incoming message to the data buffer. The header CRC field (CRC1) is checked and is written to the data buffer. The DDCMP controller updates the CRC error (CR) bit, sets the header (H) bit, writes the message type and status bits into the BD, and clears the empty bit. It next generates a maskable receive block interrupt (RBK), indicating that a header has been received and is in memory. If the header was a control message, the DDCMP controller waits for a new message.

If there is no match in address comparison and the header is error free, the DDCMP controller will use the same buffer for the next message. To maintain synchronization, the DDCMP controller counts the data length based on the count field contained in the header.

When the data buffer has been filled, the DDCMP controller clears the empty bit in the BD and generates a maskable received buffer interrupt (RBD). If the incoming message exceeds the length of the data buffer, the DDCMP controller fetches the next BD in the table, and, if it is empty, continues to transfer the rest of the message to the new data buffer. When the message ends, the CRC2 field is checked and written to the data buffer. The DDCMP controller sets the last bit, writes the message type and other status bits into the BD, and clears the empty bit. Following this, it generates an RBK, indicating that a message has been received and is in memory. The DDCMP controller then waits for a new message.

4.5.14.3 DDCMP Memory Map

When configured to operate in DDCMP mode, the IMP overlays the structure illustrated in Table 4-9 onto the protocol-specific area of that SCC's parameter RAM. Refer to 2.8 MC68302 Memory Map for the placement of the three SCC parameter RAM areas and to Table 4-5 for the other parameter RAM values.

Table 4-10. DDCMP Specific Parameter RAM

Address	Name	Width	Description
SCC Base + 9C SCC Base + 9E SCC Base + A0 # SCC Base + A2	RCRC CRCC PCRC TCRC	Word Word Word Word	Temp Receive CRC CRC16 Constant Preset CRC16 Temp Transmit CRC
SCC Base + A4 # SCC Base + A5 # SCC Base + A6 SCC Base + A7 # SCC Base + A8 SCC Base + A9 #	DSYN1 DSOH Reserved DENQ Reserved DDLE	Byte Byte Byte Byte Byte Byte	DDCMP SYN1 Character DDCMP SOH Character DDCMP ENQ Character DDCMP DLE Character
SCC Base + AA # SCC Base + AC # SCC Base + AE # SCC Base + B0 #	CRC1EC CRC2EC NMARC DISMC	Word Word Word Word	CRC1 Error Counter CRC2 Error Counter Nonmatching Address Received Counter Discard Message Counter
SCC Base + B2 SCC Base + B4	RMLG RMLG_CNT	Word Word	Received Message Length Received Message Length Counter
SCC Base + B6 # SCC Base + B8 # SCC Base + BA # SCC Base + BC # SCC Base + BE #	DMASK DADDR1 DADDR2 DADDR3 DADDR4	Word Word Word Word Word	User-Defined Frame Address Mask User-Defined Frame Address User-Defined Frame Address User-Defined Frame Address User-Defined Frame Address

Initialized by the user (M68000 core).

4.5.14.4 DDCMP Programming Model

The M68000 core configures each SCC to operate in one of four protocols by the MODE1–MODE0 bits in the SCC mode register. If MODE1–MODE0 = 10, DDCMP operation is selected with synchronous links. For asynchronous links, MODE1–MODE0 = 01 (ASYNC) should be selected, and the DDCMP bit in the UART mode register should be set. The SYN1–SYN2 synchronization characters are programmed in the data synchronization register (DSR). See 4.5.4 SCC Data Synchronization Register (DSR) for more programming information. The DDCMP controller uses the same basic data structure as the UART, HDLC, and BISYNC controllers.

The DDCMP controller generates and checks the CRC16 message trailer. It can be preset to ones or zeros by writing to the preset CRC (PCRC) register before enabling the receiver or the transmitter. The received message length (RMLG) is the header byte count value as determined by the receiver, and the received message length counter (RMLG _ CNT) is the temporary received data downcounter.

Receive and transmit errors are reported in their respective BDs. The line status signals ($\overline{CD}$ and $\overline{CTS}$) may be read in the SCC status register and a maskable interrupt is generated upon each status change (see 4.5.2 SCC Configuration Register (SCON)).

4.5.14.5 DDCMP Command Set.

The following commands are issued to the command register:

STOP TRANSMIT Command

After a hardware or software reset and the enabling of the channel in the SCC mode register, the channel is in the transmit enable mode and starts polling the first BD in the table approximately every eight transmit clocks.

The channel STOP TRANSMIT command disables the transmission of messages on the transmit channel. If this command is received by the DDCMP controller during message

transmission, message transmission is aborted after the contents of the FIFO (up to three bytes) are transmitted. The TBD# is not advanced. No new BD is accessed, and no new messages are transmitted for this channel. Upon receipt of this command, the transmitter aborts the message transmission (if currently transmitting) and then transmits SYN1–SYN2 pairs or IDLEs as determined by the DDCMP mode register.

The STOP TRANSMIT command must be issued before the SCC mode register is used to disable the transmitter if the transmitter is to be re-enabled at a later time.

RESTART TRANSMIT Command

The RESTART TRANSMIT command re-enables the transmission of characters on the transmit channel. This command is expected by the DDCMP controller after a STOP TRANSMIT command, after a STOP TRANSMIT command followed by the disabling of the channel in its SCC mode register, or after a transmitter error (underrun or CTS lost during data or maintenance message header fields). The DDCMP controller will resume transmission from the current transmitter BD number (TBD#) in the channel's transmit BD table.

If the channel is being re-enabled, the RESTART TRANSMIT command must be used and should be followed by the enabling of the transmitter in the SCC mode register.

ENTER HUNT MODE Command

After a hardware or software reset and the enabling of the channel in the SCC mode register, the channel is in the receive enable mode and will use the first BD in the table.

The ENTER HUNT MODE command is used to force the DDCMP controller to abort reception of the current message, generate an RBD interrupt (if enabled) as the buffer is closed, and enter hunt mode. In hunt mode, the DDCMP controller continually scans the input data stream for the SYN1–SYN2 sequence on synchronous links. Then for synchronous or asynchronous links, the DDCMP controller scans the input bytes for the starting byte of one of the messages. After receiving the command, the current receive buffer is closed, and the CRC is reset. Message reception continues using the next BD.

If an enabled receiver has been disabled (by clearing ENR in the SCC mode register), the ENTER HUNT MODE command must be given to the channel before setting ENR again.

4.5.14.6 DDCMP Control Character Recognition

The DDCMP controller can recognize three special control characters. These characters are used to synchronize the message and allow the DDCMP controller to function in a DMA-controlled environment.

DSYN1—DDCMP Sync Character Register

The 8-bit DSYN1 register should be written with the same value that was written in the SYN1 byte of the data synchronization register (DSR). DSYN1 is a memory-mapped read-write register.

NOTE

For correct operation of DDCMP, DSYN1, SYN1, and SYN2 must be the same value.

DSOH—DDCMP SOH Register

The 8-bit DSOH register is used to synchronize data messages by the DDCMP controller. When the DDCMP controller is not in hunt mode (byte synchronization is now established), it searches for the SOH character to start processing data messages. The DDCMP controller transfers the header and the data fields of the message to the buffer, checks the header and data CRCs, counts the data field up to the value contained in the header byte count field, and compares the header address field against the user-defined addresses. The DSOH register is a memory-mapped read-write register.

DENQ—DDCMP ENQ Register

The 8-bit DENQ register is used to synchronize control messages by the DDCMP controller. When the DDCMP controller is not in hunt mode (byte synchronization is established), it searches for the ENQ character to start processing control messages. The DDCMP controller transfers the message to the buffer, checks the CRC, and compares the message address field against the user-defined addresses. The DENQ register is a memory-mapped read-write register.

DDLE—DDCMP DLE Register

The 8-bit DDLE register is used to synchronize maintenance messages by the DDCMP controller. When the DDCMP controller is not in hunt mode (byte synchronization is established), it searches for the DLE character to start processing the maintenance messages. The DDCMP controller transfers the header and the data fields of the message to the buffer, checks the header and data CRCs, counts the data field up to the value contained in the header byte count field, and compares the header address field against the user-defined addresses. The DDLE register is a memory-mapped read-write register.

4.5.14.7 DDCMP Address Recognition.

Each DDCMP controller has five 16-bit registers to support address recognition: one mask register and four address registers (DMASK, DADDR1, DADDR2, DADDR3, and DADDR4). The DDCMP controller reads the message address from the receiver, masks it with the user-defined DMASK bits, and then checks the result against the four address register values. A one in DMASK indicates a bit position where a comparison should take place; a zero masks the comparison. For 8-bit address comparison, the high byte of DMASK should be zero.

4.5.14.8 DDCMP Error-Handling Procedure

The DDCMP controller reports message reception and transmission errors using the channel BDs, the error counters, and the DDCMP event register. The modem interface lines can also be directly monitored with the SCC status register.

Transmission errors:

1. Transmitter Underrun. When this error occurs, the channel terminates buffer transmission, closes the buffer, sets the underrun (UN) bit in the BD, and generates the transmit error (TXE) interrupt (if enabled). The channel will resume transmission after the reception of the RESTART TRANSMIT command. The FIFO size is three bytes.

NOTE

This error can occur only on synchronous links.

2. Clear-To-Send Lost (Collision) During Message Transmission. When this error occurs and the channel is not programmed to control this line with software, the channel terminates buffer transmission, closes the buffer, sets the CTS lost (CT) bit in the BD, and generates the transmit error (TXE) interrupt (if enabled). The channel resumes transmission after the reception of the RESTART TRANSMIT command.

Reception Errors:

1. Carrier Detect Lost During Message Reception. When this error occurs and the channel is not programmed to control this line with software, the channel terminates message reception, closes the buffer, sets the carrier detect lost (CD) bit in the BD, and generates the receive block (RBK) interrupt (if enabled). This error has the highest priority. The rest of the message is lost, and other errors in that message are not checked.
   The channel will enter hunt mode immediately. It is possible that a SYN1–SYN2-(SOH,DLE,ENQ) sequence in data will be incorrectly interpreted as the start of the next header, but this "header" will have a CRC error.
2. Overrun Error. The DDCMP controller maintains an internal three-byte FIFO for receiving data. The CP begins programming the SDMA channel (if the data buffer is in external memory) and updating the CRC when the first word is received into the FIFO. If the receive FIFO overrun error occurs, the channel writes the received data byte to the internal FIFO on top of the previously received byte. The previous data byte is lost. Then the channel closes the buffer, sets the overrun (OV) bit in the BD, and generates the receive block (RBK) interrupt (if enabled).
   The channel will enter hunt mode immediately. It is possible that a SYN1–SYN2-(SOH,DLE,ENQ) sequence in data will be incorrectly interpreted as the start of the next header, but this “header” will have a CRC error.
3. CRC1 (Header CRC) Error. When this error occurs, the channel writes the received CRC to the data buffer, closes the buffer, sets the CRC error (CR) bit in the BD, generates the RBK interrupt (if enabled), increments the error counter (CRC1EC), and enters hunt mode.
   When this error occurs on data-and maintenance-message header fields, the channel will enter hunt mode immediately. It is possible that a SYN1–SYN2-(SOH,DLE,ENQ) sequence in data will be incorrectly interpreted as the start of the next header, but this “header” will have a CRC error.
4. CRC2 (Data or Maintenance CRC) or CRC3 (Control Message) Error. When this error occurs, the channel writes the received CRC to the data buffer, closes the buffer, sets the CRC error (CR) bit in the BD, and generates the RBK interrupt (if enabled). The channel also increments the CRC2EC counter and enters hunt mode.
5. Framing Error. A framing error is detected by the DDCMP controller when no stop bit is detected in a received data string. When this error occurs, the channel writes the received character to the buffer, closes the buffer, sets the framing error (FR) bit in the BD, and generates the RBK interrupt (if enabled). When this error occurs, parity is not checked for this character.

The channel will enter hunt mode immediately. It is possible that a SYN1–SYN2-(SOH,DLE,ENQ) sequence in data will be incorrectly interpreted as the start of the next header, but this “header” will have a CRC error.

NOTE

This error can occur only on asynchronous links.

6. Parity Error. When a parity error occurs, the channel writes the received character to the buffer, closes the buffer, sets the parity error (PR) bit in the BD, and generates the RBK interrupt (if enabled).
   The channel will enter hunt mode immediately. It is possible that a SYN1–SYN2-(SOH,DLE,ENQ) sequence in data will be incorrectly interpreted as the start of the next header, but this “header” will have a CRC error.

NOTE

This error can occur only on asynchronous links.

Error Counters

The CP maintains four 16-bit (modulo 2**16) error counters for each DDCMP controller. They can be initialized by the user when the channel is disabled. The counters are as follows:

—CRC1EC—CRC1 Error Counter
—CRC2EC—CRC2/CRC3 Error Counter
—NMARC — Nonmatching Address Received Counter (updated only when the frame is error-free)
—DISMC — Discarded Messages (received messages when there are no free buffers and the frame is error-free)

4.5.14.9 DDCMP Mode Register

Each SCC mode register is a 16-bit, memory- mapped, read-write register that controls the SCC operation. The term DDCMP mode register refers to the protocol-specific bits (15–6) of the SCC mode register when that SCC is configured for DDCMP. The read-write DDCMP mode register is cleared by reset.

15	14	13	12	11	10	9	8	7	6	5 … 0
NOS3	NOS2	NOS1	NOS0	—	V.110	—	—	SYNF	ENC	COMMON SCC MODE BITS

NOS3–NOS0—Minimum Number of SYN1—SYN2 Pairs between or before Messages (1 to 16 SYNC Pairs)

If NOS3–NOS0 = 0000, then 1 SYNC pair will be transmitted; if NOS3–NOS0 = 1111, then 16 SYNC pairs will be transmitted.

NOTE

With appropriate programming of the transmit BD (TC = 1 and L = 0), it is possible to transmit back-to-back messages.

Bits 11, 9–8—Reserved for future use.

V.110—V.110 Mode

0 = DDCMP mode; synchronous DDCMP is chosen.
1 = V.110 mode; the V.110 protocol description is in 4.5.15 V.110 Controller.

SYNF—Transmit SYN1–SYN2 or IDLE between Messages and Control the RTS Pin

0 = Send ones between messages. $\overline{RTS}$ is negated between messages.

NOTE

The DDCMP controller can transmit ones in both NRZ and NRZI data encoded formats. The minimum number of ones transmitted is 17.

1 = Send SYN1–SYN2 pairs between messages. $\overline{RTS}$ is always asserted. Note that SYN1 and SYN2 may be the same character.

ENC—Data Encoding Format

0 = Nonreturn to Zero (NRZ). A one is a high level; a zero is a low level.
1 = Nonreturn to Zero Inverted (NRZI). A one is represented by no change in the level; a zero is represented by a change in the level. The receiver decodes NRZI, but a clock must be supplied. The transmitter encodes NRZI.

COMMON SCC MODE BITS—See 4.5.3 SCC Mode Register (SCM) for a description of the DIAG1, DIAG0, ENR, ENT, MODE1, and MODE0 bits.

4.5.14.10 DDCMP Receive Buffer Descriptor (Rx BD)

The CP reports information about the received data for each buffer using the BDs. The Rx BD is shown in Figure 4-36. The CP closes the current buffer, generates a maskable interrupt, and starts to receive data in the next buffer after any of the following events:

- Receiving the received message length number of bytes (RMLG)
- Detecting an error
- Detecting a full receive buffer
- Issuing the ENTER HUNT MODE command

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET + 0	E	X	W	I	L	H	T2	T1	—	—	CF	FR	PR	CR	OV	CD
OFFSET + 2	DATA LENGTH															
OFFSET + 4	RX BUFFER POINTER (24-bits used, upper 8 bits must be 0)															
OFFSET +6																

Figure 4-36. DDCMP Receive Buffer Descriptor

The first word of the Rx BD contains control and status bits. Bits 15–12 are written by the user before the buffer is linked to the Rx BD table, and bits 5–0 and 11–8 are set by the IMP

following message reception. Bit 15 determines whether the M68000 core or the CP may currently access the BD.

E—Empty

0 = The data buffer associated with this BD has been filled with received data, or data reception has been aborted due to an error condition. The M6800 core is free to examine or write to any fields of the BD.
1 = The data buffer associated with this BD is empty. This bit signifies that the BD and its associated buffer are available to the DDCMP controller. The M68000 core should not write to any fields of this BD after it sets this bit. Note that the empty bit will remain set while the DDCMP controller is currently filling the buffer with received data.

X—External Buffer

0 = The buffer associated with this BD is in internal dual-port RAM.
1 = The buffer associated with this BD is in external memory.

W—Wrap (Final BD in Table)

0 = This is not the last BD in the Rx BD table.
1 = This is the last BD in the Rx BD table. After this buffer has been used, the DDCMP controller places incoming data into the first BD in the table.

NOTE

The user is required to set the wrap bit in one of the first eight BDs; otherwise, errant behavior may occur.

I—Interrupt

0 = No interrupt is generated after this buffer has been closed.
1 = The RBD or RBK bits in the DDCMP event register will be set when this buffer has been closed by the DDCMP controller, which can cause interrupts.

The following status bits are written by the DDCMP controller after it has finished receiving data in the associated data buffer.

L—Last in Message

0 = The buffer is not the last in a message.
1 = The buffer is the last in a message.

H—Header in Buffer

0 = The buffer does not contain a message header.
1 = The buffer contains a message header.

NOTE

To correctly identify buffers containing headers, the buffer size should be eight or more bytes in length so that the header will fit in a single buffer.

T2,T1—Message Type

00 = Data message
01 = Control message
10 = Maintenance message
11 = Reserved

Bits 7–6—Reserved for future use.

CF—CRC Follow Error

The character following the CRC for this message was not one of SOH, ENQ, DLE, SYN, or IDLE. The receiver then enters hunt mode.

FR—Framing Error

A character with a framing error was received. The associated character may be found at the last location in this buffer. A framing error is detected by the UART controller when no stop bit is detected in the receive data string.

NOTE

This error can occur only on asynchronous DDCMP links.

PR—Parity Error

A character with a parity error was received. The associated character may be found at the last location in this buffer.

NOTE

This error can occur only on asynchronous DDCMP links.

CR—Rx CRC Error

A message with a CRC error was received in the header (CRC1) or data (CRC2) fields or a control message (CRC3).

OV—Overrun

A receiver overrun occurred during message reception.

CD—Carrier Detect Lost

The CD signal was negated during message reception. This bit is valid only when working in NMSI mode.

Data Length

The data length is the number of octets that the DDCMP controller has written to this BD's data buffer. It is written by the CP once as the BD is closed.

NOTE

The actual buffer size should be greater than or equal to eight (to ensure the header is received in one buffer).

Rx Buffer Pointer

This pointer contains the address of the associated data buffer and may be even or odd. The buffer may reside in either internal or external memory.

NOTE

For correct operation of the function codes, the upper 8 bits of the pointer must be initialized to zero.

4.5.14.11 DDCMP Transmit Buffer Descriptor (Tx BD)

Data is presented to the CP for transmission over an SCC channel by arranging it in buffers referenced by the channel's Tx BD table. The CP confirms transmission (or indicates error conditions) using the BDs to inform the M68000 core that the buffers have been serviced. The Tx BD is shown in Figure 4-37.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET + 0	R	X	W	I	L	TC	OL	—	—	—	—	—	—	—	UN	CT
OFFSET + 2	DATA LENGTH															
OFFSET + 4	TX BUFFER POINTER (24-bits used, upper 8 bits must be 0)															
OFFSET +6																

Figure 4-37. DDCMP Transmit Buffer Descriptor

The first word contains status and control bits. Bits 15–9 are prepared by the user before transmission. Bits 1–0 are set by the DDCMP controller after the buffer has been transmitted. Bit 15 is set by the user when the buffer and BD have been prepared and is cleared by the DDCMP controller when the message is transmitted.

R—Ready

0 = This buffer is not currently ready for transmission. The user is free to manipulate this BD (or its associated buffer). The DDCMP controller clears this bit after the buffer has been completely transmitted (or after an error condition is encountered).
1 = The data buffer has been prepared for transmission by the user (but not yet transmitted). No fields of this BD may be written by the user once this bit is set.

X—External Buffer

0 = The buffer associated with this BD is in internal dual-port RAM.
1 = The buffer associated with this BD is in external memory.

W—Wrap (Final BD in Table)

0 = This is not the last BD in the Tx BD table.
1 = This is the last BD in the Tx BD table. After this buffer has been used, the DDCMP controller will transmit data from the first BD in the table.

NOTE

The user is required to set the wrap bit in one of the first eight BDs; otherwise, errant behavior may occur.

I—Interrupt

0 = No interrupt is generated after this buffer has been serviced.
1 = Either TX or TXE in the DDCMP event register will be set when this buffer has been serviced by the DDCMP controller, which can cause interrupts.

L—Last

0 = This buffer is not the last in the message.
1 = The last bit is set by the processor to indicate that this buffer is the last buffer in the current message.

NOTE

The DDCMP controller checks the TC bit, not the last bit, to determine whether to append the CRC sequence. The DDCMP controller will transmit the programmable number of SYN1–SYN2 pairs before transmitting the next buffer (message) when the last bit is set.

TC—Tx CRC

0 = Do not transmit a CRC sequence after the buffer's last data byte.
1 = Transmit a CRC16 sequence after the buffer's last data byte.

When the last bit is not set but TC is set (e.g., in a header buffer), the DDCMP controller will append the next buffer immediately following the CRC sequence. The preset value for the CRC16 calculation is located in the PCRC register and should be initialized to all zeros or all ones.

OL—Optional Last

This bit allows the user to transmit abutted messages in DDCMP.

0 = Normal operation. The SYNF bit in the DDCMP mode register determines the pattern transmitted between messages.
1 = Abutted messages. The CP checks the ready bit of the next Tx BD after processing the current BD, and, if set, abuts the next message to the current message. If the ready bit is not set, the SYNF bit determines the transmitted pattern.

Bits 8–2—Reserved for future use.

The following status bits are written by the DDCMP controller after it has finished transmitting the associated data buffer.

UN—Underrun

The DDCMP controller encountered a transmitter underrun condition while transmitting the associated data buffer.

NOTE

This error can occur only on synchronous links.

CT—CTS Lost

$\overline{\text{CTS}}$ in NMSI mode or grant in IDL/GCI mode was lost during message transmission.

Data Length

The data length is the number of octets that the DDCMP controller should transmit from this BD's data buffer. It is never modified by the CP. The data length should be greater than zero.

Tx Buffer Pointer

This pointer, which contains the address of the associated data buffer, may be even or odd. The buffer may reside in either internal or external memory.

NOTE

For correct operation of the function codes, the upper 8 bits of the pointer must be initialized to zero.

4.5.14.12 DDCMP Event Register

The SCC event register (SCCE) is referred to as the DDCMP event register when the SCC is configured for DDCMP. It is an 8-bit register used to report events recognized by the DDCMP channel and to generate interrupts. On recognition of an event, the DDCMP controller sets its corresponding bit in this register. Interrupts generated by this register may be masked in the DDCMP mask register.

The DDCMP event register is a memory-mapped register that may be read at any time. A bit is cleared by writing a one (writing zero does not affect a bit's value). More than one bit may be cleared at a time. All unmasked bits must be cleared before the CP will clear the internal interrupt request. This register is cleared at reset.

7	6	5	4	3	2	1	0
CTS	CD	—	TXE	RBK	BSY	TX	RBD

CTS—Clear-To-Send Status Changed

A change in the status of the $\overline{\text{CTS}}$ line was detected on the DDCMP channel. The SCC status register may be read to determine the current status.

CD—Carrier Detect Status Changed

A change in the status of the CD line was detected on the DDCMP channel. The SCC status register may be read to determine the current status.

Bit 5—Reserved for future use.

TXE—Tx Error

An error (CTS lost or underrun) occurred on the transmitter channel.

RBK—Receive Block

A complete block has been received on the DDCMP channel. A block is defined as reception of a complete header, a complete message, or a receiver error condition.

BSY—Busy Condition

A data byte was received and discarded due to lack of buffers. The receiver will enter hunt mode automatically.

TX—Tx Buffer

A buffer has been transmitted over the DDCMP channel. This bit is set no sooner than at the beginning of the transmission of the second-to-last data (or CRC) bit in the frame, if this buffer was the last in a frame. Otherwise, it is set as the last data byte is written to the transmit FIFO.

RBD—Rx Buffer

A buffer has been received on the DDCMP channel that was not a complete block.

4.5.14.13 DDCMP Mask Register

The SCC mask register (SCCM) is referred to as the DDCMP mask register when the SCC is operating as a DDCMP controller. It is an 8-bit read-write register that has the same bit format as the DDCMP event register. If a bit in the DDCMP mask register is a one, the corresponding interrupt in the event register will be enabled. If the bit is zero, the corresponding interrupt in the event register will be masked. This register is cleared upon reset.

4.5.15 V.110 Controller

V.110 is an ISDN protocol that interfaces non-ISDN terminal equipment into one of the B channels of the 2B + D basic rate ISDN S-interface. V.110 offers the ability to transmit data from non-ISDN terminal equipment over one of the B channels through an ISDN to other non-ISDN terminal equipment. A common application of V.110 rate adaption is providing a connection between RS-232 based terminal equipment over an ISDN.

Since V.110 uses the B channels, a V.110 application requires that an ISDN call first be set up over the D channel. This call setup uses the LAPD protocol, which can be implemented using another SCC in the MC68302 (see 4.5.12 HDLC Controller).

V.110 can adapt slower equipment up to the 64 kbps B channel rate. If the non-ISDN terminal equipment is synchronous, a one- or two-step process is required. If asynchronous equipment is adapted, a three-step process is required. The standard allows for the adaption of asynchronous rates up to 19.2 kbps and synchronous rates up to 56 kbps.

Only the third step (RA2) is required for adaption of synchronous 48- and 56-kbps rates commonly used by synchronous modems. RA1 and RA2 are needed for synchronous rates of 600, 1200, 2400, 4800, 7200, 9600, 12 kbps, 14.4 kbps, and 19.2 kbps. Asynchronous terminals require all three steps because rate-adapting asynchronous terminals require the additional task of compensating for slight underspeed or overspeed of the terminal with respect to the ISDN clock rate. This is why the RA0 function refers to stop-bit manipulation.

Data is transmitted over the ISDN in a defined 80-bit frame format. This frame includes all data from the terminal (including start and stop bits for the asynchronous terminals) as well as various framing and control functions.

Another rate adaption protocol, called V.120, is an alternative protocol to V.110. V.120 is an extension of the LAPD protocol and may be implemented on the MC68302 using an SCC configured in HDLC mode.

4.5.15.1 Bit Rate Adaption of Synchronous Data Signaling Rates up to 19.2 kbps

The V.110 synchronous bit rate adaption block diagram within the terminal adaptor is shown in Figure 4-38.

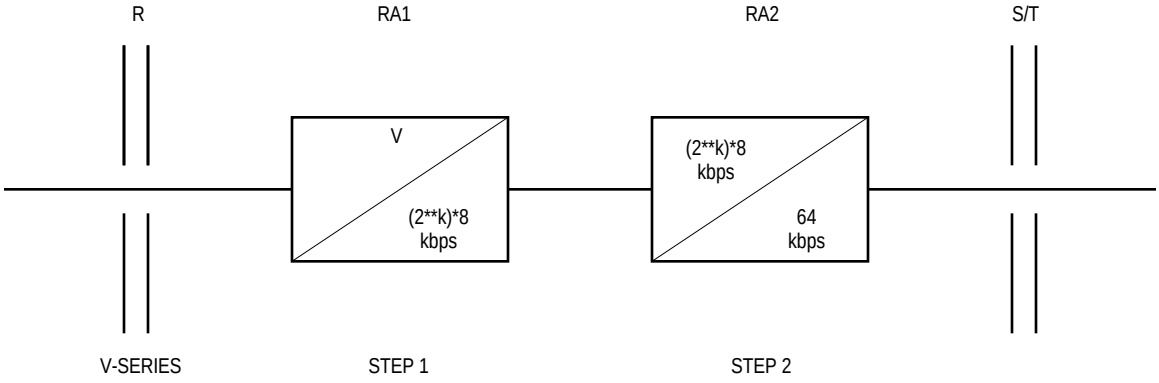


Figure 4-38. Two-Step Synchronous Bit Rate Adaption

This function may be implemented with two SCCs, one of which is configured for V.110 operation. Step 1 (RA1) rate adaption can be achieved using one SCC channel programmed to promiscuous (totally transparent) mode (see 4.5.13 BISYNC Controller). This SCC will transfer the data between the R interface and IMP memory. The M68000 core must be programmed to format the data in memory according to the V.110 protocol to create the V.110 80-bit frame. Another SCC is used to transfer the data between IMP memory and the S/T interface. This SCC should be programmed for V.110 operation, which provides the conversion of the data rate to 64 kbps. Data may be transmitted and received on 1, 2, or 4 bits of an ISDN B channel as programmed in the SIMASK register.

NOTE

V.110 contains a requirement (under further study) for control information on the R interface (i.e., RTS, CTS, CD, DTR, and DSR), conveyed by the S bits in the V.110 frame, not to have a different transmission delay than the user data conveyed by the D8–D1 bits. This very time-critical aspect of the standard is not supported by the IMP. In this case, provision would need to be made by the user to guarantee correct sampling times for this information to correspond with the user data. The IMP, however, can detect changes in these signals and issue appropriate interrupts to the M68000 core, allowing the function to be fully implemented in a slightly longer time period.

4.5.15.2 Rate Adaption of 48- and 56-kbps User Rates to 64 kbps

This function may again be implemented with two SCCs; however, in this case, the SCC connected to the B channel is programmed to promiscuous (totally transparent) mode rather than for V.110 operation (see 4.5.9 SCC Transparent Mode). The M68000 core will need to format the framing pattern in the 48-kbps conversion case. For the 56-kbps rate conversion, however, the B channel mask (SIMASK) in the serial channel physical interface can be used.

4.5.15.3 Adaption for Asynchronous Rates up to 19.2 kbps

The V.110 asynchronous bit rate adaption block diagram within the terminal adaptor is shown in Figure 4-39.

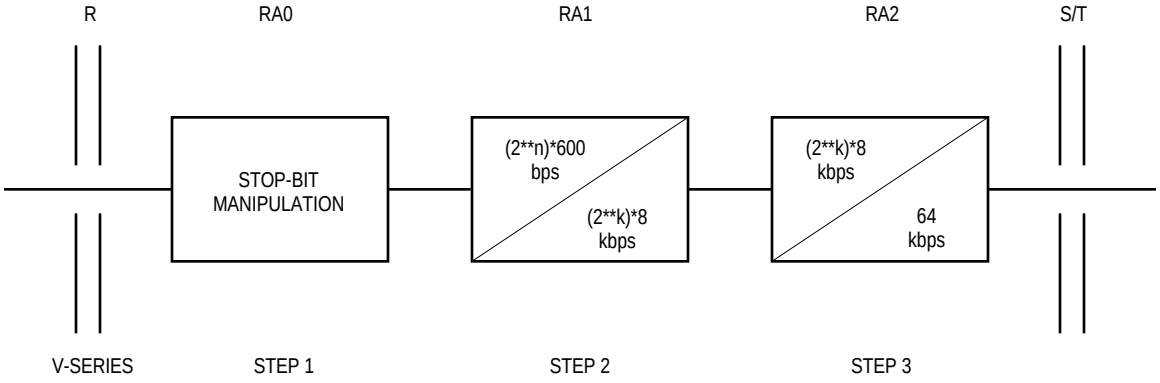


Figure 4-39. Three-Step Asynchronous Bit Rate Adaption

This function may be implemented in two SCCs. One SCC operates as a UART; the other SCC operates as a V.110 controller. The M68000 core formats the data for transmission by the V.110 at the 64 kbps data rate. Thus, the RA1 step is hidden in software.

4.5.15.4 V.110 Controller Overview.

By the appropriate setting of its SCC mode register, any of the SCC channels may be configured to function as a V.110 controller. MODE1–MODE0 bits the SCC mode register should be programmed to DDCMP, and the V.110 bit in the DDCMP mode register should be set. The V.110 controller has the ability to receive and transmit V.110 80-bit frames. The processing of those frames is handled by the M68000 core in software.

The V.110 receiver will synchronize on the 17-bit alignment pattern of the frame:

00000000 1xxxxxxx 1xxxxxxx 1xxxxxxx 1xxxxxxx

1xxxxxxx 1xxxxxxx 1xxxxxxx 1xxxxxxx 1xxxxxxx

After achieving frame synchronization, the receiver will transfer the frame data to a receive buffer (the leading one will be the MSB so that the programmer does not have to swap the bits). The V.110 controller will write nine bytes of data to the buffer (discarding the first byte of all zeros). The M68000 core should unformat the data in memory according to the V.110 protocol to create the data buffer; it may then use another SCC controller to transmit this data to the R interface.

The V.110 transmitter will transmit a data buffer transparently with a bit swap (the MSB will be transmitted first) onto a B channel. The data buffer should contain the 17-bit alignment pattern. Another SCC controller may be used to receive data from the R interface. The M68000 core should then format the data according to the V.110 protocol to create the V.110 80-bit frame data buffer. The V.110 controller will then transmit it onto the B channel.

The V.110 controller operates on the ISDN physical interface using either IDL or GCI (IOM-2) over one of the B channels. NMSI and PCM physical interfaces are also possible. The data synchronization register (DSR) should be programmed to 'xxxxxxx1 00000000'b to achieve the proper frame synchronization (see 4.5.4 SCC Data Synchronization Register (DSR)).

4.5.15.5 V.110 Programming Model

The M68000 core configures each SCC to operate in one of the protocols by the MODE1–MODE0 bits in the SCC mode register. If MODE1–MODE0 = 10, the synchronous link DDCMP protocol is selected. The V.110 bit should also be set in the DDCMP mode register. The V.110 controller uses the same basic data structure as the DDCMP controller, the same command set, and the same event and mask registers for interrupt generation.

4.5.15.6 Error-Handling Procedure

The V.110 controller reports frame reception and transmission error conditions using the channel buffer descriptors (BDs) and the V.110 event register.

Transmission Errors:

1. Transmitter Underrun. When this error occurs, the channel terminates buffer transmission, closes the buffer, sets the underrun (UN) bit in the BD, and generates the transmit error (TXE) interrupt (if enabled). The channel will resume transmission after the reception of the RESTART TRANSMIT command. The FIFO size is three bytes.

Reception Errors:

1. Overrun Error. The V.110 controller maintains an internal three-byte length FIFO for receiving data. When the receive FIFO overrun error occurs, the channel writes the received data byte to the internal FIFO on top of the previously received byte (the previous data byte is lost). Then the channel closes the buffer, sets the overflow (OV) bit in the BD, and generates the receive frame (RXF) interrupt (if enabled). The channel will automatically enter hunt mode.
2. Synchronization Error. A synchronization error is detected by the V.110 controller when the MSB of every byte is not one. When this error occurs, the channel writes the received byte to the buffer and continues to receive the V.110 frame. When the frame ends, the channel closes the buffer, sets the synchronization error (SE) bit in the BD, and generates the RXF interrupt (if enabled). The channel will automatically enter the hunt mode.

4.5.15.7 V.110 Receive Buffer Descriptor (Rx BD)

The CP reports information about the received data for each buffer using the BDs. The Rx BD is shown in Figure 4-40. The CP closes the current buffer, generates a maskable interrupt, and starts to receive data in the next buffer after any of the following events:

- Receiving of 10 bytes (80-bit frame)
- Detecting of an error
- Issuing the ENTER HUNT MODE command

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET + 0	E	X	W	—	—	—	—	—	—	—	—	—	SE	—	OV	—
OFFSET + 2	DATA LENGTH															
OFFSET + 4	RX BUFFER POINTER (24-bits used, upper 8 bits must be 0)															
OFFSET +6																

Figure 4-40. V.110 Receive Buffer Descriptor

The first word of the Rx BD contains control and status bits. Bits 15–13 are written by the user before the buffer is linked to the Rx BD table, and bits 1 and 3 are set by the IMP following message reception. Bit 15 is set by the M68000 core when the buffer is available to the V.110 controller and is cleared by the V.110 controller after filling the buffer.

E—Empty

0 = The data buffer associated with this BD has been filled with received data, or data reception has been aborted due to an error condition. The M68000 core is free to examine or write to any fields of the BD.

1 = The data buffer associated with this BD is empty. This bit signifies that the BD and its associated buffer are available to the V.110 controller. The M68000 core should not write to any fields of this BD after it sets this bit. The empty bit will remain set while the V.110 controller is currently filling the buffer with received data.

X—External Buffer

0 = The buffer associated with this BD is in internal dual-port RAM.

1 = The buffer associated with this BD is in external memory.

W—Wrap (Final BD in Table)

0 = This is not the last BD in the Rx BD table.

1 = This is the last BD in the Rx BD table. After this buffer has been used, the V.110 controller receives incoming data by placing it in the first BD in the table.

NOTE

The user is required to set the wrap bit in one of the first eight BDs; otherwise, errant behavior may occur.

Bits 12–4, 2, 0—Reserved for future use.

SE—Synchronization Error

A frame with a synchronization error was received. A synchronization error is detected by the V.110 controller when the MSB of a byte (except the all-zeros byte) is not one.

OV—Overrun

A receiver overrun occurred during message reception.

Data Length

The data length is the number of octets that the V.110 controller has written to this BD data buffer. It is written by the CP once as the BD is closed. The V.110 controller will write nine bytes of data to the buffer. It will not write the all-zeros byte to the buffer.

NOTE

The actual buffer size should be greater than or equal to 10 bytes.

Rx Buffer Pointer

This pointer contains the address of the associated data buffer. The buffer may reside in either internal or external memory.

NOTE

The Rx buffer pointer must be even, and the upper 8 bits of the pointer must be zero for the function codes to operate correctly.

4.5.15.8 V.110 Transmit Buffer Descriptor (Tx BD)

Data is presented to the CP for transmission on an SCC channel by arranging it in buffers referenced by the channel's Tx BD table. The CP confirms transmission (or indicates error conditions) using the BDs to inform the M68000 core whether the buffers have been serviced. The Tx BD is shown in Figure 4-41.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET + 0	E	X	W	I	L	0	—	—	—	—	—	—	—	—	UN	—
OFFSET + 2	DATA LENGTH															
OFFSET + 4	TX BUFFER POINTER (24-bits used, upper 8 bits must be 0)															
OFFSET +6																

Figure 4-41. V.110 Transmit Buffer Descriptor

The first word contains status and control bits. Bits 15–10 are prepared by the user before transmission. Bit 1 is set by the V.110 controller after the buffer has been transmitted. Bit 15 is set by the user and cleared by the V.110 controller.

R—Ready

0 = This buffer is not currently ready for transmission. The user is free to manipulate this BD and its associated buffer. The V.110 controller clears this bit after the buffer has been fully transmitted (or after an error condition is encountered).
1 = The data buffer has been prepared for transmission by the user (but not yet transmitted). No fields of this BD may be written by the user once this bit is set.

X—External Buffer

0 = The buffer associated with this BD is in internal dual-port RAM.
1 = The buffer associated with this BD is in external memory.

W—Wrap (Final BD in Table)

0 = This is not the last BD in the Tx BD table.
1 = This is the last BD in the Tx BD table. After this buffer has been used, the V.110 controller will transmit data from the first BD in the table.

NOTE

The user is required to set the wrap bit in one of the first eight BDs; otherwise, errant behavior may occur.

I—Interrupt

0 = No interrupt is generated after this buffer has been serviced.
1 = Either TX or TXE in the V.110 event register will be set when this buffer has been serviced by the V.110 controller, which can cause interrupts.

L—Last

0 = This buffer is not the last in the message.
1 = This bit is set by the processor to indicate that this buffer is the last buffer in the current frame. The V.110 controller will transmit ones until the next BD is ready.

Bit 10—Must be set to zero by the user.

Bits 9–2, 0—Reserved for future use.

The following bits are written by the V.110 controller after it has finished transmitting the associated data buffer.

UN—Underrun

The V.110 controller encountered a transmitter underrun condition while transmitting the associated data buffer.

Data Length

The data length is the number of octets that the V.110 controller should transmit from this BD's data buffer. It is never modified by the CP. The data length should be greater than zero.

Tx Buffer Pointer

This pointer, which contains the address of the associated data buffer, may be even or odd. The buffer may reside in either internal or external memory.

NOTE

For correct operation of the function codes, the upper 8 bits of the pointer must be initialized to zero.

4.5.15.9 V.110 Event Register

The SCC event register (SCCE) is referred to as the V.110 event register when the SCC is configured as a V.110 controller. It is an 8-bit register used to report events recognized by the V.110 channel and to generate interrupts. On recognition of an event, the V.110 control-

ler sets its corresponding bit in this register. Interrupts generated by this register may be masked in the V.110 mask register.

The V.110 event register is a memory-mapped register that may be read at any time. A bit is cleared by writing a one (writing a zero does not affect a bit's value). More than one bit may be cleared at a time. All unmasked bits must be cleared before the CP will clear the internal interrupt request signal. This register is cleared at reset.

7	6	5	4	3	2	1	0
—	—	—	TXE	RXF	BSY	TX	—

Bits 7–5, 0—Reserved for future use.

TXE—Tx Error

An error (underrun) occurred on the transmitter channel.

RXF—Receive Frame

A complete frame has been received on the V.110 channel.

BSY—Busy Condition

A data byte was received and discarded due to lack of buffers. The receiver will automatically enter hunt mode.

TX —Tx Buffer

A buffer has been transmitted.This bit is set on the second to last bit of an 80-bit frame.

4.5.15.10 V.110 Mask Register

The SCC mask register (SCCM) is referred to as the V.110 mask register when the SCC is operating as a V.110 controller. It is an 8-bit read-write register that has the same bit format as the V.110 event register. If a bit in the V.110 mask register is a one, the corresponding interrupt in the event register will be enabled. If the bit is zero, the corresponding interrupt in the event register will be masked. This register is cleared upon reset.

4.5.16 Transparent Controller

The transparent controller allows transmission and reception of serial data over an SCC without any modification to that data stream. Transparent mode provides a clear channel on which no bit-level manipulation is performed by the SCC. Any protocol run over transparent mode is performed in software. The job of an SCC in transparent mode is to function simply as a high-speed serial-to-parallel and parallel-to-serial converter. This mode is also referred to as totally transparent or promiscuous operation.

There are several basic applications for transparent mode. First, some data may need to be moved serially but requires no protocol superimposed. An example of this is voice data. Second, some board-level applications require a serial-to-parallel and parallel-to-serial conversion. Often this conversion is performed to allow communication between chips on the same board. The SCCs on the MC68302 can do this very efficiently with very little M68000 core intervention. Third, some applications require the switching of data without interfering with

the protocol encoding itself. For instance, in a multiplexer, data from a high-speed time-multiplexed serial stream is multiplexed into multiple low-speed data streams. The concept is to switch the data path but not alter the protocol encoded on that data path.

By appropriately setting the SCC mode register, any of the SCC channels can be configured to function as a transparent controller. Transparent mode is achieved by setting the SCM MODE1–MODE0 bits to 11 and setting the NTSYN bit (bit 13) to 1. Note that, if the NTSYN bit is cleared, normal BISYNC operation will occur rather than transparent operation. See 4.5.13 BISYNC Controller for a description of BISYNC operation.

The SCC in transparent mode can work with IDL, GCI (IOM-2), PCM highway, or NMSI interfaces. When the SCC in transparent mode is used with a modem interface (NMSI), the SCC outputs are connected directly to the external pins. The modem interface uses seven dedicated pins: transmit data (TXD), receive data (RXD), receive clock (RCLK), transmit clock (TCLK), carrier detect or carrier detect sync ($\overline{\text{CD}}$), clear to send ($\overline{\text{CTS}}$), and request to send ($\overline{\text{RTS}}$).

The transparent controller consists of separate transmit and receive sections whose operations are asynchronous with the M68000 core and may be either synchronous or asynchronous with respect to the other SCCs. Transparent mode on the MC68302 is a synchronous protocol; thus, a clock edge must be provided with each bit of data received or transmitted. Each clock can be supplied from either the internal baud rate generator or from external pins. More information on the baud rate generator is available in 4.5.2 SCC Configuration Register (SCON).

The main transparent controller features are as follows:

- Flexible Data Buffers
- External Sync Pin or BISYNC-Like Frame Sync on Receive
- Reverse Data Mode
- Interrupts on Buffers Transmitted or Received
- Clear to Send and Carrier Detect Lost Reporting
- Maskable Interrupt Available on Each Character Received
- Three Commands

4.5.16.1 Transparent Channel Buffer Transmission Processing

When the M68000 core enables the transparent transmitter, it will start transmitting ones. The transparent controller then polls the first BD in the transmit channel's BD table approximately every 16 transmit clocks. When there is a buffer to transmit, the transparent controller will fetch the data from memory and start transmitting the buffer. Transmission will not begin until the internal transmit FIFO is preloaded and the SCC achieves synchronization. See 4.5.16.5 Transparent Synchronization for details of the synchronization process.

When a BD's data is completely transmitted, the last bit (L) is checked in the BD. If the L bit is cleared, then the transmitter moves immediately to the next buffer to begin its transmission, with no gap on the serial line between buffers. Failure to provide the next buffer in time results in a transmit underrun, causing the TXE bit in the transparent event register to be set.

If the L bit is set, the frame ends, and the transmission of ones resumes until a new buffer is made ready. $\overline{RTS}$ is negated during this period. Regardless of whether or not the next buffer is available immediately, the next buffer will not begin transmission until achieving synchronization.

The transmit buffer length and starting address may be even or odd; however, since the transparent transmitter reads a word at a time, better performance can be achieved with an even buffer length and starting address. For example, if a transmit buffer begins on an odd-byte boundary and is 10 bytes in length (the worst case), six word reads will result, even though only 10 bytes will be transmitted.

Any whole number of bytes may be transmitted. If the REVD bit in the transparent mode register is set, each data byte will be reversed in its bit order before transmission.

If the interrupt (I) bit in the Tx BD is set, then the TX bit will be set in the transparent event register following the transmission of the buffer. The TX bit can generate a maskable interrupt.

4.5.16.2 Transparent Channel Buffer Reception Processing

When the M68000 core enables the transparent receiver, it will enter hunt mode. In this mode, it waits to achieve synchronization before receiving data. See 4.5.16.5 Transparent Synchronization for details.

Once data reception begins, the transparent receiver begins moving data from the receive FIFO to the receive buffer, always moving a 16-bit word at a time. After each word is moved to memory, the RCH bit in the transparent event register is set, which can generate a maskable interrupt, if desired. The transparent receiver continues to move data to the receive buffer until the buffer is completely full, as defined by the byte count in MRBLR. The receive buffer length (stored in MRBLR) and starting address must always be even, so the minimum receive buffer length must be 2.

After a buffer is filled, the transparent receiver moves to the next Rx BD in the table and begins moving data to its associated buffer. If the next buffer is not available when needed, a busy condition is signified by the setting of the BSY bit in the transparent event register, which can generate a maskable interrupt.

Received data is always packed into memory a word at a time, regardless of how it is received. For example, in NMSI mode, the first word of data will not be moved to the receive buffer until after the sixteenth receive clock occurs. In PCM highway mode, the same principle applies except that the clocks are only internally active during an SCC time slot. For example, if each SCC time slot is seven bits long, the first word of data will not be moved to the receive buffer until after the second bit of the third time slot, regardless of how much time exists between individual time slots.

Once synchronization is achieved for the receiver, the reception process continues unabated until a busy condition occurs, a $\overline{CD}$ lost condition occurs, or a receive overrun occurs. The busy condition error should be followed by an ENTER HUNT MODE command to the

channel. In all three error cases, the reception process will not proceed until synchronization has once again been achieved.

If the REVD bit in the transparent mode register is set, each data byte will be reversed in its bit order before it is written to memory.

If the interrupt (I) bit in the Rx BD is set, then the RX bit will be set in the transparent event register following the reception of the buffer. The RX bit can generate a maskable interrupt.

4.5.16.3 Transparent Memory Map

When configured to operate in transparent mode, the IMP overlays the structure illustrated in Table 4-11 onto the protocol specific area of that SCC parameter RAM. Refer to 2.8 MC68302 Memory Map for the placement of the three SCC parameter RAM areas and Table 4-5 for the other parameter RAM values.

Table 4-11. Transparent-Specific Parameter RAM

Address	Name	Width	Description
SCC BASE + 9C	RES	WORD	Reserved
SCC BASE + 9E	RES	WORD	Reserved
SCC BASE + A0	RES	WORD	Reserved
SCC BASE + A2	RES	WORD	Reserved
SCC BASE + A4	RES	WORD	Reserved
SCC BASE + A6	RES	WORD	Reserved
SCC BASE + A8	RES	WORD	Reserved
SCC BASE + AA	RES	WORD	Reserved
SCC BASE + AC	RES	WORD	Reserved
SCC BASE + AE	RES	WORD	Reserved
SCC BASE + B0	RES	WORD	Reserved
SCC BASE + B2	RES	WORD	Reserved
SCC BASE + B4	RES	WORD	Reserved
SCC BASE + B6	RES	WORD	Reserved
SCC BASE + B8	RES	WORD	Reserved
SCC BASE + BA	RES	WORD	Reserved
SCC BASE + BC	RES	WORD	Reserved
SCC BASE + BE	RES	WORD	Reserved

Although there are no transparent-specific parameter RAM locations that must be initialized by the user, the general SCC parameter RAM must still be initialized (see Table 4-5).

The transparent controller uses the same basic data structure as the other protocol controllers. Receive and transmit errors are reported through receive and transmit BDs. The status of the line is reflected in the SCC status register, and a maskable interrupt is generated upon each status change.

If in-line synchronization is used with the transparent controller, then the DSR (see 4.5.4 SCC Data Synchronization Register (DSR)) must be initialized with the SYN1–SYN2 synchronization characters. See 4.5.16.5 Transparent Synchronization for details.

4.5.16.4 Transparent Commands

The following commands are issued to the command register.

STOP TRANSMIT Command

After a hardware or software reset and the enabling of the channel using the SCC mode register, the channel is in the transmit enable mode and starts polling the first BD in the table approximately every 16 transmit clocks.

The STOP TRANSMIT command aborts transmission. If this command is received by the transparent controller during a buffer transmission, transmission of that buffer is aborted after the FIFO contents (up to four words) are transmitted. The TBD# is not advanced. Ones are continuously transmitted until transmission is re-enabled by issuing the RESTART TRANSMIT command.

The STOP TRANSMIT command must be issued before the SCC mode register is used to disable the transmitter if the transmitter is to be re-enabled at a later time.

RESTART TRANSMIT Command

The RESTART TRANSMIT command is used to begin or resume transmission from the current Tx BD number (TBD#) in the channel's Tx BD table. When this command is received by the channel, it will start polling the ready bit in this BD. This command is expected by the transparent controller after a STOP TRANSMIT command, after a STOP TRANSMIT command and the disabling of the channel in its mode register, or after a transmitter error (underrun or CTS lost occurs).

If the transmitter is being re-enabled, the RESTART TRANSMIT command must be used and should be followed by the enabling of the transmitter in the SCC mode register.

ENTER HUNT MODE Command

After a hardware or software reset and the enabling of the channel in the SCC mode register, the channel is in the receive enable mode and will use the first BD in the table.

The ENTER HUNT MODE command is used to force the transparent controller to abort reception of the current block, generate an RX interrupt (if enabled) as the buffer is closed, and enter the hunt mode. In the hunt mode, the transparent controller waits for a synchronization to occur on the SCC (see 4.5.16.5 Transparent Synchronization). After receiving the ENTER HUNT MODE command, the current receive buffer is closed. Reception continues using the next BD.

If an enabled receiver has been disabled (by clearing ENR in the SCC mode register), the ENTER HUNT MODE command must be given to the channel before setting ENR again.

4.5.16.5 Transparent Synchronization

Once the SCC is enabled for transparent operation in the SCM, the transmit and receive buffer descriptors are made ready for the SCC, and the transmit FIFO has been preloaded by the SDMA channel (signaled by the $\overline{\text{RTS}}$ pin in NMSI and PCM modes), one additional process must occur before data can be transmitted and received. This process is called

transparent synchronization. Transparent synchronization gives the user bit-level control over when the transmission and reception can begin.

The method of synchronization is controlled by the DIAG1–DIAG0 bits in the SCM, the EXSYN bit in the SCM, and, in some cases, the data synchronization register (DSR). The resulting timing is dependent on the physical interface chosen.

NOTE

See D.8 Using the MC68302 Transparent Mode for timing diagrams and additional details concerning mode.

Five ways exist to achieve transparent synchronization.

1. With the physical interface in the NMSI mode, the SCC may be configured with the EXSYN bit set, and the DIAG1–DIAG0 bits set to software operation. In this case, the $\overline{CTS}$ pin is ignored, and the $\overline{CD}$ pin is used to control both transmission and reception. For the transmitter, once $\overline{RTS}$ is asserted and the transmitter samples $\overline{CD}$ as low, the transmission will begin after a fixed 6.5 transmit clock delay. For the receiver, the first valid bit of data received occurs one bit prior to the receive clock that sampled $\overline{CD}$ as low. Note that $\overline{CD}$ is sampled on a rising TCLK for the transmitter and a rising RCLK for the receiver.
   Once $\overline{CD}$ is sampled as low by the receiver and transmitter, further toggling of $\overline{CD}$ will have no effect since synchronization has already been achieved.
2. With the physical interface in NMSI mode, the SCC may be configured with the EXSYN bit cleared and the DIAG1–DIAG0 bits set to software operation. For the transmitter, the transmission will begin without any synchronization. For the receiver, reception will begin as soon as the receive data pattern matches the SYN1–SYN2 pattern programmed into the DSR. Thus, the receiver uses an in-line synchronization method.
3. If case 2 is used but the DIAG1–DIAG0 bits are set to normal operation, then the normal operation characteristics as described in the SCM register also apply. The transmitter will be controlled by the $\overline{CTS}$ pin, and the receiver will wait for the SYN1–SYN2 pattern once the $\overline{CD}$ pin is detected as low.

NOTE

In cases 2 and 3 above, the SYN2 character is written into the receive data buffer.

4. With the physical interface configured for PCM highway mode, the SCC may be configured with the EXSYN bit set and the DIAG1–DIAG0 bits set to either software operation or normal operation. In this case, the L1SY1–L1SY0 pins carry out the synchronization. For the transmitter, once data is preloaded into the transmit FIFO, the rising edge of the L1SY1–L1SY0 pins will cause a transmission to occur. This transmission will be comprised of one leading $FF byte, followed by the first bit of the transmit buffer. For the receiver, reception will begin at the beginning of a time slot.

NOTE

This technique is not valid for the PCM envelope sync method when the time slots are less than six bits in length. In such a case, the user may clear EXSYN to cause transmission to begin, and then, for the receiver, provide the required SYN1–SYN2 sequence. To accomplish this, the user may configure the SCC to loopback mode with the EXSYN bit cleared and the DSR set to $FFFF. Then after 16 serial clocks, the receiver and transmitter are synchronized, and the SCC may be dynamically reconfigured to normal or software operation. At this point, reception begins immediately, and transmission begins after the transmit BD is made ready.

5. With the physical interface configured for IDL or GCI mode, the SCC may be configured with the EXSYN bit set and the DIAG1–DIAG0 bits set to either software operation or normal operation. In this case, the data will be byte-aligned to the B or D channel time slots.

Once synchronization is achieved for the transmitter, it will remain in effect until an error occurs, a STOP TRANSMIT command is given, or a buffer has completed transmission with the Tx BD last (L) bit set. Once synchronization is achieved for the receiver, it will remain in effect until an error occurs or the ENTER HUNT MODE command is given.

4.5.16.6 Transparent Error-Handling Procedure

The transparent controller reports message reception and transmission error conditions using the channel BDs and the transparent event register. The modem interface lines can also be directly monitored in the SCC status register.

Transmission Errors:

1. Transmitter Underrun—When this error occurs, the channel terminates buffer transmission, closes the buffer, sets the underrun (UN) bit in the BD, and generates the TXE interrupt (if enabled). The channel resumes transmission after the reception of the RESTART TRANSMIT command. Underrun can occur after a transmit frame for which the L bit in the Tx BD was not set. In this case, only the TXE bit is set. The FIFO size is four words in transparent mode.
2. Clear-To-Send Lost During Message Transmission—When this error occurs and the channel is not programmed to control this line with software, the channel terminates buffer transmission, closes the buffer, sets the CTS lost (CT) bit in the BD, and generates the TXE interrupt (if enabled). The channel will resume transmission after the reception of the RESTART TRANSMIT command.

Reception Errors:

1. Overrun Error—The transparent controller maintains an internal three-word FIFO for receiving data. The CP begins programming the SDMA channel (if the data buffer is in external memory) when the first word is received into the FIFO. If a FIFO overrun occurs, the transparent controller writes the received data word to the internal FIFO over the previously received word. The previous word is lost. Next, the channel closes

the buffer, sets the overrun (OV) bit in the BD, and generates the RX interrupt (if enabled). The receiver then enters hunt mode immediately.

2. Carrier Detect Lost During Message Reception—When this error occurs and the channel is not programmed to control this line with software, the channel terminates reception, closes the buffer, sets the carrier detect lost (CD) bit in the BD, and generates the RX interrupt (if enabled). This error has the highest priority; the rest of the message is lost and no other errors are checked. The receiver then enters hunt mode immediately.
3. Busy Condition—If the RISC controller tries to use an Rx BD that is not empty, the busy condition is encountered. No data is received and the current Rx BD is left unmodified. After new buffers are provided, the user should issue the ENTER HUNT MODE command.

4.5.16.7 Transparent Mode Register

Each SCC mode register is a 16-bit, memory-mapped, read-write register that controls the SCC operation. The term transparent mode register refers to the protocol-specific bits (15–6) of the SCC mode register when that SCC is configured for transparent mode. The transparent mode register is cleared by reset. All undefined bits should be written with zero.

15	14	13	12	11	10	9	8	7	6	5 0
—	EXSYN	NTSYN	REVD	—	—	—	—	—	—	COMMON SCC MODE BITS

Bit 15—Reserved for future use; should be written with zero.

EXSYN — External Sync Mode

When this mode is selected, the receiver and transmitter expect external logic to indicate the beginning of the data field by using the $\overline{CD1}$/L1SY1 pin, if SCC1 is used, and the $\overline{CD2}$ and $\overline{CD3}$ pins, respectively, if SCC2 or SCC3 is used. In this mode, there is no carrier detect function for the SCC.

When the channel is programmed to work through the serial channels physical interface (IDL or GCI) and EXSYN is set, the layer 1 logic carries out the synchronization using the L1SY1 pin. In PCM mode, the L1SY1–L1SY0 pins are used. In NMSI mode, the $\overline{CD}$ pins (and the $\overline{CD}$ timing) are used to synchronize the data. $\overline{CD}$ should go low on the second valid data bit of the receive data stream.

If this bit is cleared, the receiver will look for the SYN1–SYN2 sequence in the data synchronization register to achieve synchronization, and the transmitter uses the $\overline{CTS}$ pin according to the DIAG1–DIAG0 bits in the SCM. The receiver also uses the $\overline{CD}$ pin according to the DIAG1–DIAG0 bits in the SCM.

NTSYN—No Transmit SYNC

This bit must be set for the SCC to operate in a totally transparent (promiscuous) mode.

REVD—Reverse Data

When this bit is set, the receiver and transmitter will reverse the character bit order, transmitting the most significant bit first.

Bits 11–6—Reserved for future use; should be written with zero.

COMMON SCC MODE BITS—See 4.5.3 SCC Mode Register (SCM) for a description of the DIAG1, DIAG0, ENR, ENT, MODE1, and MODE0 bits.

4.5.16.8 Transparent Receive Buffer Descriptor (RxBD)

The CP reports information about the received data for each buffer using BD. The Rx BD is shown in Figure 4-42. The CP closes the current buffer, generates a maskable interrupt, and starts to receive data into the next buffer after one of the following events:

- Detecting an error
- Detecting a full receive buffer
- Issuing the ENTER HUNT MODE command

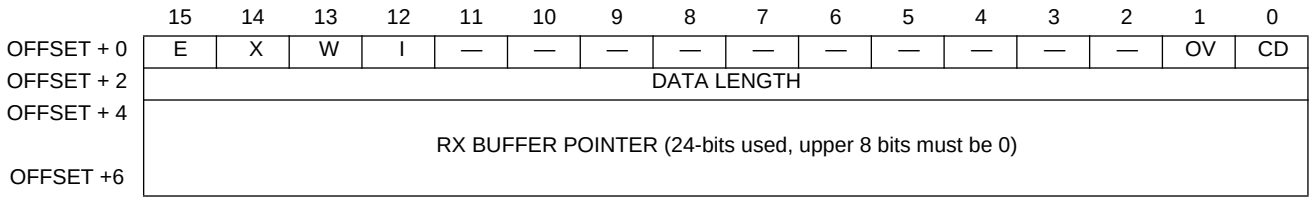


Figure 4-42. Transparent Receive Buffer Descriptor

The first word of the Rx BD contains control and status bits.

E—Empty

0 = The data buffer associated with this BD has been filled with received data, or data reception has been aborted due to an error condition. The M68000 core is free to examine or write to any fields of this BD.

1 = The data buffer associated with this BD is empty. This bit signifies that the BD and its associated buffer are available to the CP. After it sets this bit, the M68000 core should not write to any fields of this BD when this bit is set. The empty bit will remain set while the CP is currently filling the buffer with received data.

X—External Buffer

0 = The buffer associated with this BD is in internal dual-port RAM.

1 = The buffer associated with this BD is in external memory.

W—Wrap (Final BD in Table)

0 = This is not the last BD in the Rx BD table.

1 = This is the last BD in the Rx BD table. After this buffer has been used, the CP will receive incoming data into the first BD in the table. Setting this bit allows the use of fewer than eight BDs to conserve internal RAM.

NOTE

The user is required to set the wrap bit in one of the first eight BDs; otherwise, errant behavior may occur.

I—Interrupt

0 = No interrupt is generated after this buffer has been used.
1 = When this buffer has been closed by the transparent controller, the RX bit in the transparent event register will be set, which can cause an interrupt.

The following status bits are written by the CP after the received data has been placed into the associated data buffer.

Bits 11–2—Reserved for future use. Should be written with zero by the user.

OV—Overrun

A receiver overrun occurred during reception.

CD—Carrier Detect Lost

The carrier detect signal was negated during buffer reception.

Data Length

The data length is the number of octets that the CP has written into this BD's data buffer. It is written only once by the CP as the buffer is closed.

NOTE

The actual buffer size should be greater than or equal to the MRBLR.

Rx Buffer Pointer

The receive buffer pointer, which always points to the first location of the associated data buffer, must be even. The buffer may reside in either internal or external memory.

NOTE

For correct operation of the function codes, the upper 8 bits of the pointer must be initialized to zero.

4.5.16.9 Transparent Transmit Buffer Descriptor (Tx BD)

Data is presented to the CP for transmission on an SCC channel by arranging it in buffers referenced by the channel's Tx BD table. The CP confirms transmission (or indicates error conditions) using the BDs to inform the processor that the buffers have been serviced. The Tx BD is shown in Figure 4-43.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET + 0	R	X	W	I	L	—	—	—	—	—	—	—	—	—	UN	CT
OFFSET + 2	DATA LENGTH															
OFFSET + 4	TX BUFFER POINTER (24-bits used, upper 8 bits must be 0)															
OFFSET +6																

Figure 4-43. Transparent Transmit Buffer Descriptor

The first word of the Tx BD contains status and control bits. These bits are prepared by the user before transmission and are set by the CP after the buffer has been transmitted.

R—Ready

0 = This buffer is not currently ready for transmission. The user is free to manipulate this BD (or its associated buffer). The CP clears this bit after the buffer has been fully transmitted or after an error condition has been encountered.
1 = The data buffer has been prepared for transmission by the user (but not yet transmitted). No fields of this BD may be written by the user once this bit is set.

X—External Buffer

0 = The buffer associated with this BD is in internal dual-port RAM.
1 = The buffer associated with this BD is in external memory.

W—Wrap (Final BD in Table)

0 = This is not the last BD in the Tx BD table.
1 = This is the last BD in the Tx BD table. After this buffer has been used, the CP will transmit data from the first BD in the table.

NOTE

The user is required to set the wrap bit in one of the first eight BDs; otherwise, errant behavior may occur.

I—Interrupt

0 = No interrupt is generated after this buffer has been serviced.
1 = When this buffer is serviced by the CP, the TX or TXE bit in the transparent event register will be set, which can cause an interrupt.

L—Last in Message

0 = The last byte in the buffer is not the last byte in the transmitted block. Data from the next transmit buffer (if ready) will be transmitted immediately following the last byte of this buffer.
1 = The last byte in the buffer is the last byte in the transmitted block. After this buffer is transmitted, the transmitter will require synchronization before the next buffer can be transmitted.

Bits 10—2 are reserved for future use; they should be written with zero.

The following status bits are written by the CP after it has finished transmitting the associated data buffer.

UN—Underrun

The transparent controller encountered a transmitter underrun condition while transmitting the associated data buffer.

CT—CTS Lost

CTS in NMSI mode or L1GR in IDL/GCI mode was lost during frame transmission.

Data Length

The data length is the number of octets that the CP should transmit from this BD's data buffer. The data length, which should be greater than zero, may be even or odd. This value is never modified by the CP.

Tx Buffer Pointer

The transmit buffer pointer, which always points to the first byte of the associated data buffer, may be even or odd. The buffer may reside in either internal or external memory.

NOTE

For correct operation of the function codes, the upper 8 bits of the pointer must be initialized to zero.

4.5.16.10 Transparent Event Register

The SCC event register (SCCE) is referred to as the transparent event register when the SCC is programmed as a transparent controller. It is an 8-bit register used to report events recognized by the transparent channel and to generate interrupts. On recognition of an event, the transparent controller sets the corresponding bit in the transparent event register. Interrupts generated by this register may be masked in the transparent mask register.

The transparent event register is a memory-mapped register that may be read at any time. A bit is cleared by writing a one (writing a zero does not affect a bit's value). More than one bit may be cleared at a time. All unmasked bits must be cleared before the CP will negate the internal interrupt request signal. This register is cleared at reset.

7	6	5	4	3	2	1	0
CTS	CD	—	TXE	RCH	BSY	TX	RX

CTS—Clear-To-Send Status Changed

A change in the status of the serial line was detected on the transparent channel. The SCC status register may be read to determine the current status.

CD—Carrier Detect Status Changed

A change in the status of the serial line was detected on the transparent channel. The SCC status register may be read to determine the current status.

Bit 5—Reserved for future use.

TXE—Tx Error

An error (CTS lost or underrun) occurred on the transmitter channel.

RCH—Receive Character

A word has been received and written to the receive buffer.

BSY—Busy Condition

A word was received and discarded due to lack of buffers. The receiver will resume reception after an ENTER HUNT MODE command.

TX—Tx Buffer

A buffer has been transmitted. If the L bit in the Tx BD is set, TX is set no sooner than on the second-to-last bit of the last byte being transmitted on the TXD pin. If the L bit in the Tx BD is cleared, TX is set after the last byte was written to the transmit FIFO.

RX—Rx Buffer

A complete buffer has been received on the transparent channel. RX is set no sooner than 10 serial clocks after the last bit of the last byte in the buffer is received on the RXD pin.

4.5.16.11 Transparent Mask Register

The SCC mask register (SCCM) is referred to as the transparent mask register when the SCC is operating as a transparent controller. It is an 8-bit read-write register that has the same bit format as the transparent event register. If a bit in the transparent mask register is a one, the corresponding interrupt in the event register will be enabled. If the bit is zero, the corresponding interrupt in the event register will be masked. This register is cleared at reset.

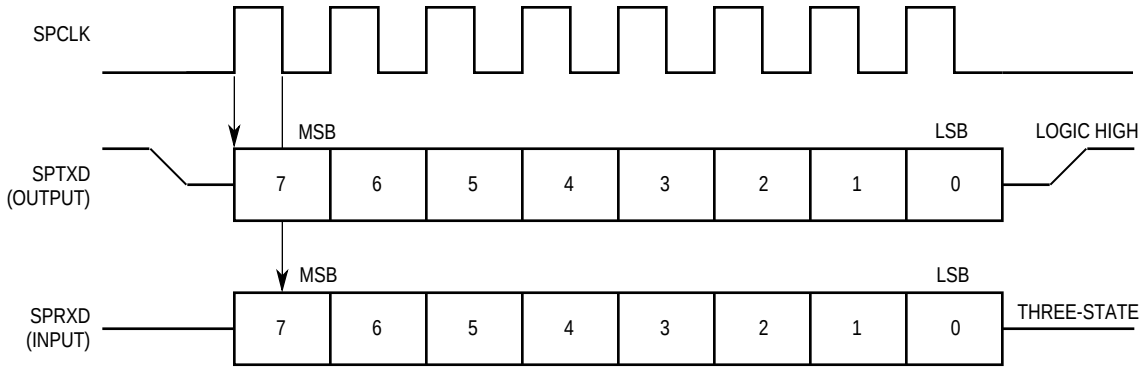
4.6 SERIAL COMMUNICATION PORT (SCP)

The SCP (see Figure 4-44) is a full-duplex, synchronous, character-oriented channel that provides a three-wire interface (receive, transmit, and clock). The SCP consists of independent transmitter and receiver sections and a common clock generator. The transmitter and receiver sections use the same clock, which is derived from the main clock by a separate on-chip baud rate generator. Since the MC68302 is an SCP master for this serial channel, it generates both the enable and the clock signals.

The SCP allows the MC68302 to exchange status and control information with a variety of serial devices, using a subset of the Motorola serial peripheral interface (SPI). The SCP is compatible with SPI slave devices. These devices include industry-standard CODECs as well as other microcontrollers and peripherals.

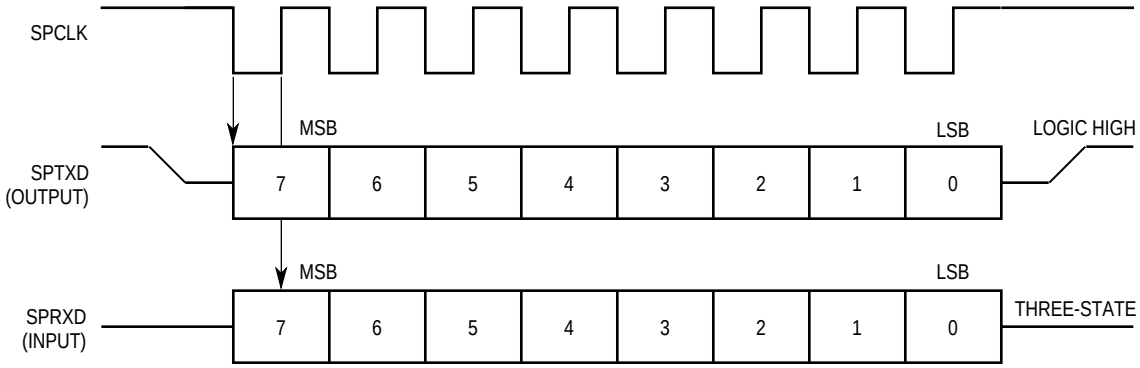
The SCP enable signals, which can be implemented using the general-purpose I/O pins, are used to enable one of several potential SCP slave devices. The clock signal (SPCLK) shifts the received data (SPRXD) in and shifts the transmitted data (SPTXD) out. The clock is gated; it operates only while data is being transferred and is idle otherwise.

Two successive byte transmissions over the SCP cannot occur immediately back-to-back. A minimum delay of two to eight bit times is imposed by the SCP, depending on the SCP clock rate (Communication processor priorities and software handling of interrupts may contribute extra delays). Higher SCP clock rates give higher minimum delay.



NOTE: Transmitted data bits shift on rising edges; received bits are sampled on falling edges.

(a) CI=0



NOTE: Transmitted data bits shift on falling edges; received bits are sampled on rising edges.

(b) CI=1

Figure 4-44. SCP Timing

The SCP can be configured to operate in a local loopback mode, which is useful for local diagnostic functions.

Note that the least significant bit of the SCP is labeled as data bit 0 on the serial line; whereas, other devices, such as the MC145554 CODEC, may label the most significant bit as data bit 0. The MC68302 SCP bit 7 (most significant bit) is shifted out first.

The SCP key features are as follows:

- Three-Wire Interface (SPTXD, SPRXD, and SPCLK)
- Full-Duplex Operation
- Clock Rate up to 4.096 MHz
- Programmable Clock Generator
- Local Loopback Capability for Testing

4.6.1 SCP Programming Model

The SCP mode register consists of the upper eight bits of SPMODE. The SCP mode register, an internal read-write register that controls both the SCP operation mode and clock source, is cleared by reset.

15	14	13	12	11	10	9	8
STR	LOOP	CI	PM3	PM2	PM1	PM0	EN

STR—Start Transmit

When set, this bit causes the SCP controller to transmit eight bits from the SCP transmit/receive buffer descriptor (BD) and to receive eight bits of data in this same BD. This bit is cleared automatically after one system clock cycle.

LOOP—Loop Mode

When set, the loop mode bit selects local loopback operation. The ones complement of the transmitter output is internally connected to the receiver input; the receiver and transmitter operate normally except that SPRXD is ignored. When cleared, this bit selects normal operation.

CI—Clock Invert

When set, the CI bit inverts the SCP clock polarity. When CI is zero, transmitted data bits shift on rising clock edges, and received bits are sampled on falling edges. When the SCP is idle, the clock is low. While CI is one, transmitted data bits are shifted on falling edges, and received bits are sampled on rising edges. In this case, when the SCP is idle, the clock is high.

PM3–PM0—Prescale Modulus Select

The prescale modulus select bits specify the divide ratio of the prescale divider in the SCP clock generator. The divider value is 4*(“PM3–PM0” + 1) giving a clock divide ratio of 4 to 64 in multiples of 4. With a 16.384-MHz system clock, the maximum SCP clock is 4.096 MHz.

EN—Enable SCP

When set, this bit enables the SCP operation and connects the external pins SPRXD/$\overline{\text{CTS3}}$, SPTXD/$\overline{\text{RTS3}}$, and SPCLK/$\overline{\text{CD3}}$ internally to the SCP (see Figure 4-45). When cleared, the SCP is put into a reset state consuming minimal power, and the three pins are connected back to SCC3.

NOTE

When the DIAG1–DIAG0 bits of SCC3 are programmed to normal operation control of the $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ lines and the ENT or ENR bits of SCC3 are set, the user may not modify the EN bit.

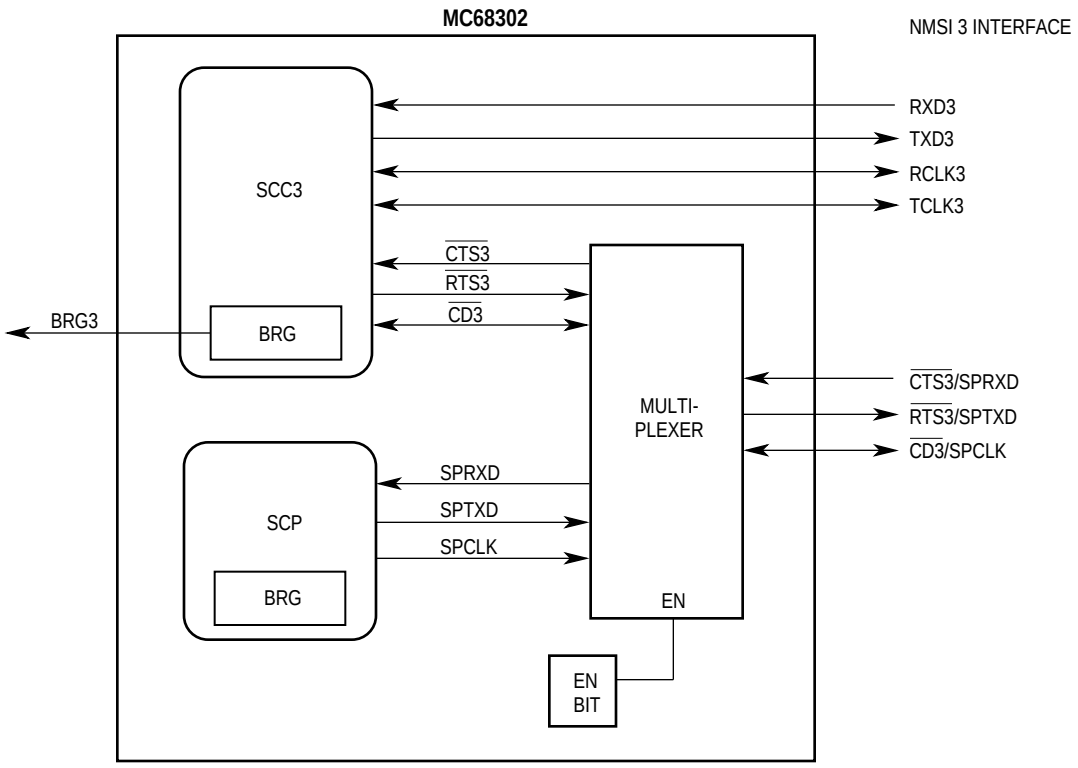


Figure 4-45. SCP vs. SCC Pin Multiplexing

4.6.2 SCP Transmit/Receive Buffer Descriptor

The transmit/receive BD contains the data to be transmitted (written by the M68000 core) and the received data (written by the SCP). The done (D) bit indicates that the received data is valid and is cleared by the SCP.

15	14 … 8	7 … 0
D	RESERVED	DATA

4.6.3 SCP Transmit/Receive Processing

The IMP SCP always functions in the master mode. Thus, in a typical exchange of messages, the IMP transmits a message to an external peripheral (SCP slave) which, in turn, sends back a reply. When the IMP works with more than one slave, it can use the general-purpose parallel I/O pins as enable (select) signals. To begin the data exchange, the M68000 core writes the data to be transmitted into the transmit/receive BD, and sets the done bit. The M68000 core should then set the start transmit (STR) bit in the SPMODE register to start transmission of data. STR is cleared by hardware after one system clock cycle.

Upon recognizing the STR bit, the SCP also begins receiving eight bits of data. It writes the data into the transmit/receive BD, clears the done bit, and issues a maskable interrupt to the IMP interrupt controller. When working in a polled environment, the done bit should be set

by the M68000 core before setting the STR bit so that received replies may be easily recognized by the software.

4.7 SERIAL MANAGEMENT CONTROLLERS (SMCS)

The SMC key features are as follows:

- Two Modes of Operation:
  - —IDL—SMC1 supports the maintenance channel and SMC2 supports the auxiliary channel
  - —GCI (IOM-2)—SMC1 supports the monitor channel and SMC2 supports the C/I channel
- Full-Duplex Operation
- Local Loopback Capability for Testing

4.7.1 Overview

The SMCs are two synchronous, full-duplex serial management control (SMC) ports. The SMC ports may be configured to operate in either Motorola interchip digital link (IDL) or general circuit interface (GCI) modes. GCI is also known as ISDN oriented modular 2 (IOM-2). See 4.4 Serial Channels Physical Interface for the details of configuring the IDL and GCI interfaces. The SMC ports are not used when the physical serial interface is configured for PCM highway or NMSI modes.

4.7.1.1 Using IDL with the SMCs

In this mode, SMC1 transfers the maintenance (M) bits of the IDL to and from the internal RAM, and SMC2 transfers the auxiliary (A) bits to and from the internal RAM. The CP generates a maskable interrupt upon reception/transmission of eight bits. The SMC1 and SMC2 receivers can be programmed to work in hunt-on-zero mode, in which the receiver will search the line signals for a zero bit. When it is found, the receiver will transfer data to the internal RAM.

4.7.1.2 Using GCI with the SMCs

In this mode, SMC1 controls the GCI monitor channel.

SMC1 Transmission

The monitor channel is used to transfer commands to the layer-1 component. The M68000 core writes the data byte into the SMC1 Tx BD. SMC1 will transmit the data on the monitor channel.

The SMC1 channel transmitter can be programmed to work in one of two modes:

Transparent Mode

In this mode, SMC1 transmits the monitor channel data and the A and E control bits transparently into the channel. When the M68000 core has not written new data to the buffer, the SMC1 transmitter will retransmit the previous monitor channel data and the A and E control bits.

Monitor Channel Protocol

In this mode, SMC1 transmits the data and handles the A and E control bits according to the GCI monitor channel protocol. When using the monitor channel protocol, the user may issue the TIMEOUT command to solve deadlocks in case of bit errors in the A and E bit positions on data line. The IMP will transmit an abort on the E bit.

SMC1 Reception

The SMC1 receiver can be programmed to work in one of two modes:

Transparent Mode

In this mode, SMC1 receives the data, moves the A and E control bits transparently into the SMC1 receive BD, and generates a maskable interrupt. The SMC1 receiver discards new data when the M68000 core has not read the receive BD.

Monitor Channel Protocol

In this mode, SMC1 receives data and handles the A and E control bits according to the GCI monitor channel protocol. When a received data byte is stored by the CP in the SMC1 receive BD, a maskable interrupt is generated.

When using the monitor channel protocol, the user may issue the TRANSMIT ABORT REQUEST command. The IMP will then transmit an abort request on the A bit.

SMC2 Controls the GCI Command/Indication (C/I) Channel

SMC2 Transmission

The M68000 core writes the data byte into the SMC2 Tx BD. SMC2 will transmit the data continuously on the C/I channel to the physical layer device.

SMC2 Reception

The SMC2 receiver continuously monitors the C/I channel. When a change in data is recognized and this value is received in two successive frames, it will be interpreted as valid data. The received data byte is stored by the CP in the SMC2 receive BD, and a maskable interrupt is generated.

The receive and transmit clocks are derived from the same physical clock (L1CLK) and are only active while serial data is transferred between the SMC controllers and the serial interface.

When SMC loopback mode is chosen, SMC transmitted data is routed to the SMC receiver. Transmitted data appears on the L1TXD pin, unless the SDIAG1–SDIAG0 bits in the SIMODE register are programmed to “loopback control” (see 4.4 Serial Channels Physical Interface).

4.7.2 SMC Programming Model

The operating mode of both SMC ports is defined by SMC mode, which consists of the lower eight bits of SPMODE. As previously mentioned, the upper eight bits program the SCP.

7	6	5	4	3	2	1	0
—	SMD3	SMD2	SMD1	SMD0	LOOP	EN2	EN1

Bit 7—This bit is reserved and should be set to zero.

SMD3–SMD0—SMC Mode Support

X00X = GCI—The monitor channel is not used.
001X = GCI—The monitor channel data and the A and E control bits are internally controlled according to the monitor channel protocol.
101X = GCI—The monitor channel data and the A and E control bits are received and transmitted transparently by the IMP.
X100 = IDL—The M and A channels are in hunt-on-zero mode.
X101 = IDL—Only the M channel is in hunt-on-zero mode.
X110 = IDL—Only the A channel is in hunt-on-zero mode.
X111 = IDL—Regular operation; no channel is in hunt-on-zero mode.

LOOP—Local Loopback Mode

0 = Normal mode
1 = Local loopback mode. In GCI mode, EN1 and EN2 must also be set.

EN2—SMC2 Enable

0 = Disable SMC2
1 = Enable SMC2

EN1—SMC1 Enable

0 = Disable SMC1
1 = Enable SMC1

4.7.3 SMC Commands

The following commands issued to the CP command register (see 4.3 Command Set) are used only when GCI is selected for the serial channels physical interface.

TRANSMIT ABORT REQUEST Command

This receiver command may be issued when the IMP implements the monitor channel protocol. When issued, the IMP sends an abort request on the A bit.

TIMEOUT Command

This transmitter command may be issued when the IMP implements the monitor channel protocol. It is issued because the device is not responding or because GCI A bit errors are detected. When issued, the IMP sends an abort request on the E bit.

4.7.4 SMC Memory Structure and Buffers Descriptors

The CP uses several memory structures and memory-mapped registers to communicate with the M68000 core. All the structures detailed in the following paragraphs reside in the dual-port RAM of the IMP (see Figure 3-4). The SMC buffer descriptors allow the user to define one data byte at a time for each transmit channel and receive one data byte at a time for each receive channel.

4.7.4.1 SMC1 Receive Buffer Descriptor

The CP reports information about the received byte using this (BD).

15	14	13	12	11 10	9	8	7 0
E	L	ER	MS	—	AB	EB	DATA

E—Empty

0 = This bit is cleared by the CP to indicate that the data byte associated with this BD is now available to the M68000 core.

1 = This bit is set by the M68000 core to indicate that the data byte associated with this BD is empty.

In GCI mode, when the IMP implements the monitor channel protocol, the IMP will wait until this bit is set by the M68000 core before acknowledging the monitor channel data. In other modes (transparent GCI and IDL), additional received data bytes will be discarded until the empty bit is set by the M68000 core.

L—Last (EOM)

This bit is valid only in GCI mode when the IMP implements the monitor channel protocol. This bit is set when the end-of-message (EOM) indication is received on the E bit.

NOTE

When this bit is set, the data byte is not valid.

ER—Error Condition

This bit is valid only in GCI mode when the IMP implements the monitor channel protocol and the L bit is set. This bit is set when an error condition occurs on the monitor channel protocol. A new byte is transmitted before the IMP acknowledges the previous byte.

MS—Data Mismatch

This bit is valid only in GCI mode when the IMP implements the monitor channel protocol. This bit is set when two different consecutive bytes are received and is cleared when the last two consecutive bytes match. The IMP waits for the reception of two identical consecutive bytes before writing new data to the receive BD.

Bits 11–10—Reserved for future use.

AB—Received A Bit

This bit is valid only in GCI mode when the monitor channel is in transparent mode.

EB—Received E Bit

This bit is valid only in GCI mode when the monitor channel is in transparent mode.

Data—Data Field

The data field contains the byte of data received by SMC1.

4.7.4.2 SMC1 Transmit Buffer Descriptor

The CP reports information about this transmit byte through the BD.

15	14	13	12 — 10	9	8	7 — 0
R	L	AR	—	AB	EB	DATA

R—Ready

0 = This bit is cleared by the CP after transmission. The Tx BD is now available to the M68000 core.

1 = This bit is set by the M68000 core to indicate that the data byte associated with this BD is ready for transmission.

In GCI mode, when the IMP implements the monitor channel protocol, it will clear this bit after receiving an acknowledgment on the A bit. When the SMC1 data should be transmitted and this bit is cleared, the channel will retransmit the previous data until new data is provided by the M68000 core.

L—Last (EOM)

This bit is valid only in GCI mode when the IMP implements the monitor channel protocol. When this bit is set, the SMC1 channel will transmit the buffer's data and then the end of message (EOM) indication on the E bit.

AR—Abort Request

This bit is valid only in GCI mode when the IMP implements the monitor channel protocol. This bit is set by the IMP when an abort request was received on the A bit. The SMC1 transmitter will transmit EOM on the E bit.

Bits 12–10—Reserved for future use.

AB—Transmit A Bit Value

This bit is valid only in GCI mode when the monitor channel is in transparent mode.

EB—Transmit E Bit Value

This bit is valid only in GCI mode when the monitor channel is in transparent mode.

Data—Data Field

The data field contains the data to be transmitted by SMC1.

4.7.4.3 SMC2 Receive Buffer Descriptor

In the IDL mode, this BD is identical to the SMC1 receive BD. In the GCI mode, SMC2 is used to control the C/I channel.

15	14 — 6	5 — 2	1	0
E	RESERVED	C/I	0	0

E—Empty

0 = This bit is cleared by the CP to indicate that the data bits associated with this BD are now available to the M68000 core.

1 = This bit is set by the M68000 core to indicate that the data bits associated with this BD have been read.

NOTE

Additional data received will be discarded until the empty bit is set by the M68000 core.

Bits 14–6—These bits are reserved and should be set to zero by the M68000 core.

C/I—Command/Indication Channel Data

Bits 1–0—The CP always writes these bits with zeros.

4.7.4.4 SMC2 Transmit Buffer Descriptor

In the IDL mode, this BD is identical to the SMC1 transmit BD. In the GCI mode, SMC2 is used to control the C/I channel.

15	14 … 6	5 … 2	1	0
R	RESERVED	C/I	0	0

R—Ready

0 = This bit is cleared by the CP after transmission to indicate that the BD is now available to the M68000 core.

1 = This bit is set by the M68000 core to indicate that the data associated with this BD is ready for transmission.

Bits 14–6—Reserved for future use; should be set to zero by the user.

C/I—Command/Indication Channel Data

Bits 1–0—These bits should be written with zeros by the M68000 core.

4.7.5 SMC Interrupt Requests

SMC1 and SMC2 send individual interrupt requests to the IMP interrupt controller when one of the respective SMC receive buffers is full or when one of the SMC transmit buffers is empty. Each of the two interrupt requests from each SMC is enabled when its respective SMC channel is enabled in the SPMODE register. Interrupt requests from SMC1 and SMC2 can be masked in the interrupt mask register. See 3.2 Interrupt Controller for more details.

SECTION 5
SIGNAL DESCRIPTION

This section defines the MC68302 pinout. The input and output signals of the MC68302 are organized into functional groups and are described in the following sections. The MC68302 is offered in a 132-pin (13 x 13) pin grid array (PGA), a 132-lead plastic quad flat package (PQFP), and a 144-lead thin quad flat package (TQFP).

The MC68302 uses a standard M68000 bus for communication between both on-chip and external peripherals. This bus is a single, continuous bus existing both on-chip and off-chip the MC68302. Any access made internal to the device is visible externally. Any access made external is visible internally. Thus, when the M68000 core accesses the dual-port RAM, the bus signals are driven externally. Likewise, when an external device accesses an area of external system memory, the chip-select logic can be used to generate the chip-select signal and $\overline{\text{DTACK}}$.

5.1 FUNCTIONAL GROUPS

The input and output signals of the MC68302 are organized into functional groups as shown in Table 5-1 and Figure 5-1.

Table 5-1. Signal Definitions

Functional Group	Signals	Number
Clocks	XTAL, EXTAL, CLKO	3
System Control	$\overline{\text{RESET}}$, $\overline{\text{HALT}}$, $\overline{\text{BERR}}$, BUSW, DISCPU	5
Address Bus	A23–A1	23
Data Bus	D15–D0	16
Bus Control	$\overline{\text{AS}}$, R/$\overline{\text{W}}$, $\overline{\text{UDS}}$/A0, $\overline{\text{LDS/DS}}$, $\overline{\text{DTACK}}$	5
Bus Control	$\overline{\text{RMC}}$, IAC, $\overline{\text{BCLR}}$	3
Bus Arbitration	$\overline{\text{BR}}$, $\overline{\text{BG}}$, $\overline{\text{BGACK}}$	3
Interrupt Control	$\overline{\text{IPL2}}$–$\overline{\text{IPL0}}$, FC2–FC0, $\overline{\text{AVEC}}$	7
NMSI1/ISDN I/F	RXD, TXD, RCLK, TCLK, $\overline{\text{CD}}$, $\overline{\text{CTS}}$, $\overline{\text{RTS}}$, BRG1	8
NMSI2/PAIO	RXD, TXD, RCLK, TCLK, $\overline{\text{CD}}$, $\overline{\text{CTS}}$, $\overline{\text{RTS}}$, SDS2	8
NMSI3/SCP/PAIO	RXD, TXD, RCLK, TCLK, $\overline{\text{CD}}$, $\overline{\text{CTS}}$, $\overline{\text{RTS}}$, PA12	8
IDMA/PAIO	DREQ, DACK, DONE	3
IACK/PBIO	IACK7, IACK6, IACK1	3
Timer/PBIO	TIN2, TIN1, $\overline{\text{TOUT2}}$, $\overline{\text{TOUT1}}$, $\overline{\text{WDOG}}$	5
PBIO	PB11–PB8	4

Table 5-1. Signal Definitions

Chip Select	CS3–CS0	4
Testing	$\overline{FRZ}$ (2 Spare)	3
V_{DD}		8
GND		13

All pins except EXTAL, CLKO, and the layer 1 interface pins in IDL mode support TTL levels. EXTAL, when used as an input clock, needs a CMOS level. CLKO supplies a CMOS level output. The IDL interface is specified as a CMOS electrical interface.

All outputs (except CLKO and the GCI pins) drive 130 pF. CLKO is designed to drive 50 pF. The GCI output pins drive 150 pF.

5.2 POWER PINS

The IMP has 21 power supply pins. Careful attention has been paid to reducing IMP noise, potential crosstalk, and RF radiation from the output drivers. Inputs may be +5 V when V_{DD} is 0 V without damaging the device.

- V_{DD} (8)—There are 8 power pins.
- GND (13)—There are 13 ground pins.

NOTE

The Input High Voltage and Input Low Voltage for EXTAL and the values for power are specified in the DC Electrical Characteristics.A valid clock signal oscillates between a low voltage of between V_{SS}-0.3 and .6 volts and a high voltage of between 4.0 and V_{DD} volts. This EXTAL signal must be present within 20 mS after V_{DD} reaches its minimum specified level of 4.5 volts.

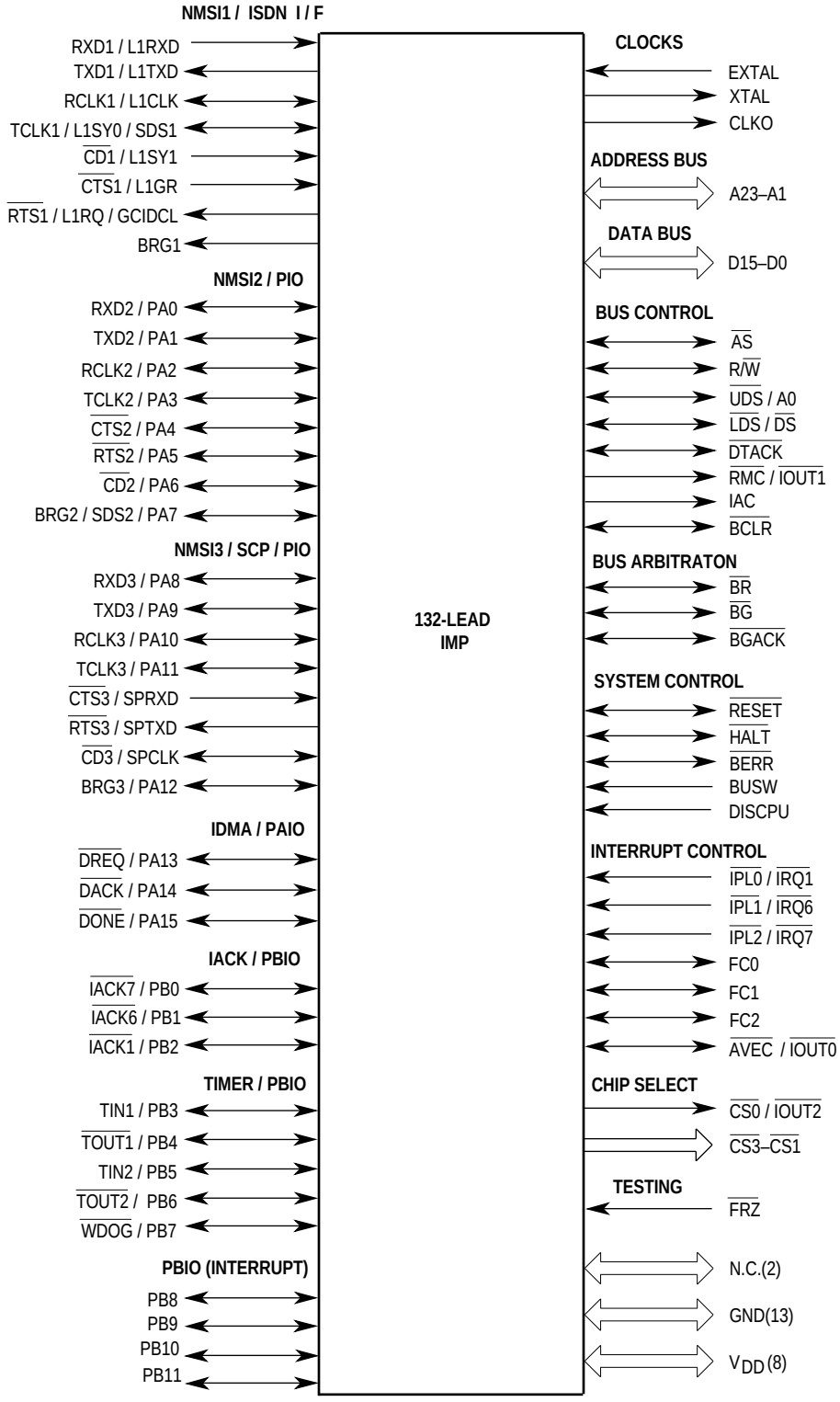


Figure 5-1. Functional Signal Groups

5.3 CLOCKS

The clock pins are shown in Figure 5-2.

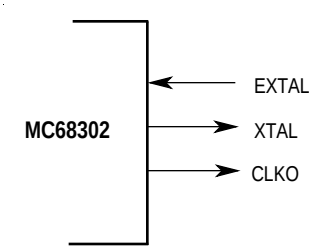


Figure 5-2. Clock Pins

EXTAL—External Clock/Crystal Input

This input provides two clock generation options (crystal and external clock). EXTAL may be used (with XTAL) to connect an external crystal to the on-chip oscillator and clock generator. If an external clock is used, the clock source should be connected to EXTAL, and XTAL should be left unconnected. The oscillator uses an internal frequency equal to the external crystal frequency. The frequency of EXTAL may range from 8 MHz to 25 MHz. When an external clock is used, it must provide a CMOS level at this input frequency.

NOTE

The Input High Voltage and Input Low Voltage for EXTAL and the values for Power are specified in the DC Electrical Characteristics.A valid clock signal oscillates between a low voltage of between V_{SS}-0.3 and .6 volts and a high voltage of between 4.0 and V_{DD} volts. This EXTAL signal must be present within 20 mS after V_{DD} reaches its minimum specified level of 4.5 volts.

The frequency range of the original MC68302 was 8 MHz to 16.67 MHz. Thus, in this manual, many references to the frequency “16.67 MHz” are made when the maximum operating frequency of the MC68302 is discussed. Similarly, the value “8 MHz” is often used when discussing the minimum operating frequency. When using faster versions of the MC68302, such as 20 or 25 MHz, all references to 16.67 MHz may be replaced with 20 or 25 MHz. When using versions with slower minimum operating frequencies, such as 4 MHz, all references to 8 MHz may be replaced with 4 MHz. Note, however, that resulting parameters such as baud rates and timer periods change accordingly.

XTAL—Crystal Output

This output connects the on-chip oscillator output to an external crystal. If an external clock is used, XTAL should be left unconnected. See 3.7 On-Chip Clock Generator for more details.

CLKO—Clock Out

This output clock signal is derived from the on-chip clock oscillator. This clock signal is internally connected to the clock input of the M68000 core, the communication processor,

and system integration block. All M68000 bus timings are referenced to the CLKO signal. CLKO supports both CMOS and TTL output levels. The output drive capability of the CLKO signal is programmable in the CKCR register (see 3.9 Clock Control Register) to one-third, two-thirds, or full strength, or this output can be disabled.

5.4 SYSTEM CONTROL

The system control pins are shown in Figure 5-3.

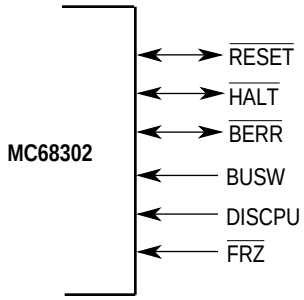


Figure 5-3. System Control Pins

RESET

This bidirectional, open-drain signal, acting as an input and asserted along with the HALT pin, starts an initialization sequence called a total system reset that resets the entire MC68302. RESET and HALT should remain asserted for at least 100 ms at power-on reset, and at least 10 clocks otherwise. The on-chip system RAM is not initialized during reset except for several locations initialized by the CP.

An internally generated reset, from the M68000 RESET instruction, causes the RESET line to become an output for 124 clocks. In this case, the M68000 core is not reset; however, the communication processor is fully reset, and the system integration block is almost fully reset (refer to Table 2-6 and Table 2-9 for a list of the unaffected registers). The user may also use the RESET output signal in this case to reset all external devices.

During total system reset, the address, data, and bus control pins are all three-stated, except for CS3–CS0, which are high, and IAC, which is low. The BG pin output is the same as that on the BR input. The general-purpose I/O pins are configured as inputs, except for WDOG, which is an open-drain output. The NMSI1 pins are all inputs, except for RTS1 and TXD1, which output a high value. RTS3, NC1, and NC3 are also high. CLKO is active and BRG1 is CLO/3.

NOTE

The RESET pin should not be asserted externally without also asserting the HALT pin. To reset just the internal MC68302 peripherals, the RESET instruction may be used. If the RESET instruction is to be used, then the pull-up resistor on RESET should not be greater than 1.2 k ohms.

Besides the total system reset and the RESET instruction, some of the MC68302 peripherals have reset bits in one of their registers that cause that particular peripheral to be reset to the same state as a total system reset or the RESET instruction. Reset bits may be found in the CP (in the CR), the IDMA (in the CMR), timer 1 (in the TMR1), and timer 2 (in the TMR2).

$\overline{\text{HALT}}$—Halt

When this bidirectional, open-drain signal is driven by an external device, it will cause the IMP bus master (M68000 core, SDMA, or IDMA) to stop at the completion of the current bus cycle. If the processor has stopped executing instructions due to a double-fault condition, this line is driven by the processor to indicate to external devices that the processor has stopped. An example of a double-fault condition is the occurrence of a bus error followed by a second bus error during the bus error exception stacking process. This signal is asserted with the $\overline{\text{RESET}}$ signal to cause a total MC68302 system reset. If $\overline{\text{BERR}}$ is asserted with the $\overline{\text{HALT}}$ signal, a retry cycle is performed.

$\overline{\text{BERR}}$—Bus Error

This bidirectional, open-drain signal informs the bus master (M68000 core, SDMA, IDMA, or external bus master) that there is a problem with the cycle currently being executed. This signal can be asserted by the on-chip hardware watchdog (bus timeout because of no $\overline{\text{DTACK}}$), by the chip-select logic (address conflict or write-protect violation), or by external circuitry. If $\overline{\text{BERR}}$ is asserted with the $\overline{\text{HALT}}$ signal, a retry cycle is performed.

BUSW—Bus Width Select

This input defines the M68000 processor mode (MC68000 or MC68008) and the data bus width (16 bits or 8 bits, respectively). BUSW may only be changed upon a total system reset. In 16-bit mode, all accesses to internal and external memory by the MC68000 core, the IDMA, SDMA, and external master may be 16 bits, according to the assertion of the UDS and LDS pins. In 8-bit mode, all M68000 core and IDMA accesses to internal and external memory are limited to 8 bits. Also in 8-bit mode, SDMA accesses to external memory are limited to 8 bits, but CP accesses to the CP side of the dual-port RAM continue to be 16 bits. In 8-bit mode, external accesses to internal memory are also limited to 8 bits at a time.

Low = 8-bit data bus, MC68008 core processor

High = 16-bit data bus, MC68000 core processor

DISCPU—Disable CPU (M68000 core)

The MC68302 can be configured to work solely with an external CPU. In this mode the on-chip M68000 core CPU should be disabled by asserting the DISCPU pin high during a total system reset ($\overline{\text{RESET}}$ and $\overline{\text{HALT}}$ asserted). DISCPU may only be changed upon a total system reset.

The DISCPU pin, for instance, allows use of several IMPs to provide more than three SCC channels without the need for bus isolation techniques. Only one of the IMP M68000 cores is active and services the other IMPs as peripherals (with their respective cores disabled). Refer to 3.8.4 Disable CPU Logic (M68000) for more details.

$\overline{FRZ}$—Freeze Activity

The $\overline{FRZ}$ pin is used to freeze the activity of selected peripherals. This is useful for system debugging purposes. Refer to 3.8 System Control for more details on which peripherals are affected. $\overline{FRZ}$ should be continuously negated during total system reset.

5.5 ADDRESS BUS PINS (A23–A1)

The address bus pins are shown in Figure 5-4.

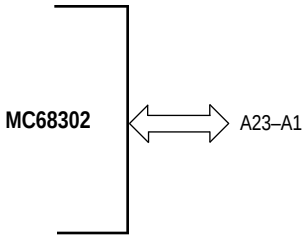


Figure 5-4. Address Bus Pins

A23—A1 form a 24-bit address bus when combined with $\overline{UDS}$/A0. The address bus is a bidirectional, three-state bus capable of addressing 16M bytes of data (including the IMP internal address space). It provides the address for bus operation during all cycles except CPU space cycles. In CPU space cycles, the CPU reads a peripheral device vector number.

These lines are outputs when the IMP (M68000 core, SDMA or IDMA) is the bus master and are inputs otherwise.

5.6 DATA BUS PINS (D15—D0)

The data bus pins are shown in Figure 5-5.

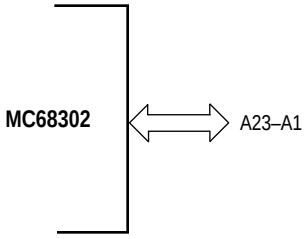


Figure 5-5. Data Bus Pins

This 16-bit, bidirectional, three-state bus is the general-purpose data path. It can transmit and accept data in either word or byte lengths. For all 16-bit IMP accesses, byte 0, the high-order byte of a word, is available on D15–D8, conforming to the standard M68000 format.

When working with an 8-bit bus (BUSW is low), the data is transferred through the low-order byte (D7–D0). The high-order byte (D15–D8) is not used for data transfer, but D8-D15 are outputs during write cycles and are not three-stated.

5.7 BUS CONTROL PINS

The bus control pins are shown in Figure 5-6.

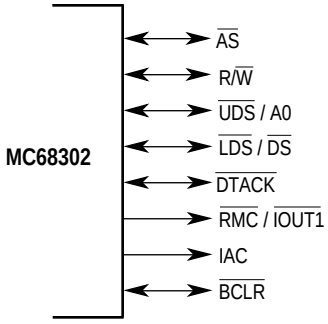


Figure 5-6. Bus Control Pins

AS—Address Strobe

This bidirectional signal indicates that there is a valid address on the address bus. This line is an output when the IMP (M68000 core, SDMA or IDMA) is the bus master and is an input otherwise.

R/W—Read/Write

This bidirectional signal defines the data bus transfer as a read or write cycle. It is an output when the IMP is the bus master and is an input otherwise.

UDS/A0—Upper Data Strobe/Address 0

This bidirectional line controls the flow of data on the data bus. When using a 16-bit data bus, this pin functions as upper data strobe (UDS). When using an 8-bit data bus, this pin functions as A0. When used as A0 (i.e., the BUSW pin is low), then the pin takes on the timing of the other address pins, as opposed to the strobe timing. This line is an output when the IMP is the bus master and is an input otherwise.

LDS/DS—Lower Data Strobe/Data Strobe

This bidirectional line controls the flow of data on the data bus. When using a 16-bit data bus, this pin functions as lower data strobe (LDS). When using an 8-bit data bus, this pin functions as DS. This line is an output when the IMP (M68000 core, SDMA or IDMA) is the bus master and is an input otherwise.

DTACK—Data Transfer Acknowledge

This bidirectional signal indicates that the data transfer has been completed. DTACK can be generated internally in the chip-select logic either for an IMP bus master or for an external bus master access to an external address within the chip-select ranges. It will also be generated internally during any access to the on-chip dual-port RAM or internal regis-

ters. If $\overline{\text{DTACK}}$ is generated internally, then it is an output. It is an input when the IMP accesses an external device not within the range of the chip-select logic or when programmed to be generated externally.

$\overline{\text{RMC}}$/$\overline{\text{IOUT1}}$—Read-Modify-Write Cycle Indication/Interrupt Output 1

This signal functions as $\overline{\text{RMC}}$ in normal operation. $\overline{\text{RMC}}$ is an output signal that is asserted when a read-modify-write cycle is executed. It indicates that the cycle is indivisible.

When the M68000 core is disabled, this pin operates as $\overline{\text{IOUT1}}$. $\overline{\text{IOUT2}}$–$\overline{\text{IOUT0}}$ provide the interrupt request output signals from the IMP interrupt controller to an external CPU when the M68000 core is disabled.

IAC—Internal Access

This output indicates that the current bus cycle accesses an on-chip location. This includes the on-chip 4K byte block of internal RAM and registers (both real and reserved locations), and the system configuration registers ($0F0–$0FF). The above-mentioned bus cycle may originate from the M68000 core, the IDMA, or an external bus master. Note that, if the SDMA accesses the internal dual-port RAM, it does so without arbitration on the M68000 bus; therefore, the IAC pin is not asserted in this case. The timing of IAC is identical to that of the $\overline{\text{CS3}}$–$\overline{\text{CS0}}$ pins.

IAC can be used to disable an external address/data buffer when the on-chip dual-port RAM and registers are accessed, thus preventing bus contention. Such a buffer is optional and is only required in larger systems. An external address/data buffer with its output enable (E) and direction control (dir) may be placed between the two bus segments as shown in Figure 5-7. The IAC signal saves the propagation delay and logic required to OR all the various system chip-select lines together to determine when to enable the external buffers.

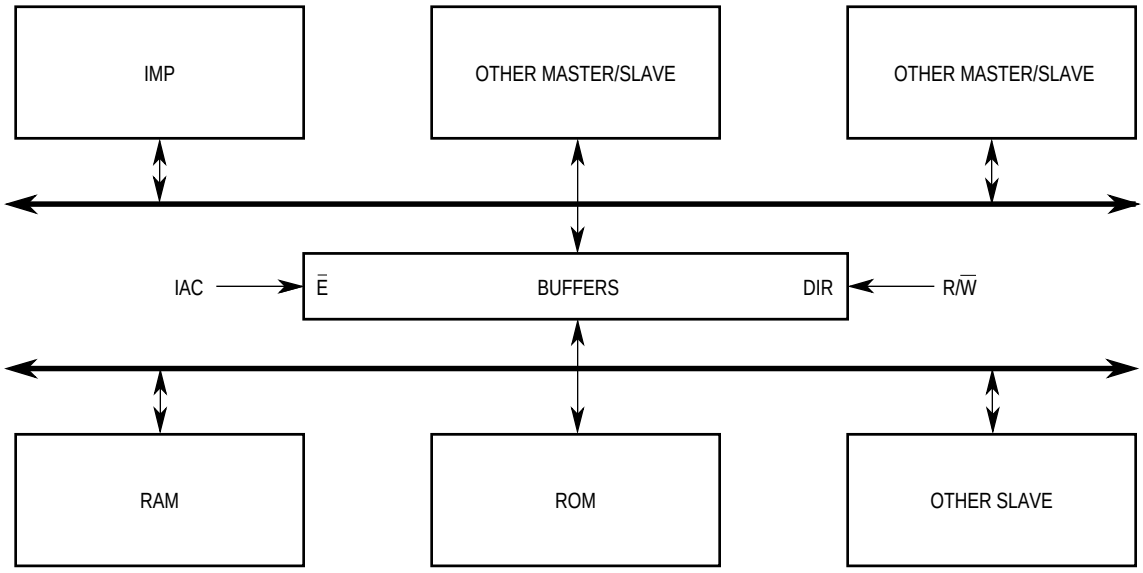


Figure 5-7. External Address/Data Buffer

$\overline{\text{BCLR}}$—Bus Clear

This open-drain output indicates that the M68000 core or the serial DMA (SDMA) requests the external bus master to release the bus. The core may be configured to assert this signal when it has a pending interrupt to execute. The SDMA asserts this signal when one of the SCCs is requesting DMA service.

When the M68000 core is disabled, this signal is an input to the independent DMA (IDMA) and is interpreted as a bus release request. It remains an output from the SDMA in this mode.

5.8 BUS ARBITRATION PINS

The bus arbitration pins are shown in Figure 5-8.

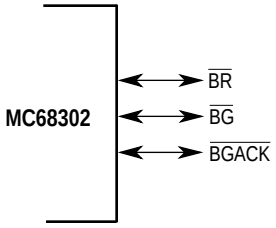


Figure 5-8. Bus Arbitration Pins

$\overline{\text{BR}}$—Bus Request

This input signal indicates to the on-chip bus arbiter that an external device desires to become the bus master. See 3.8.5.2 External Bus Arbitration for details. This signal is an open-drain output request signal from the IDMA and SDMA when the internal M68000 core is disabled.

$\overline{\text{BG}}$—Bus Grant

This output signal indicates to all external bus master devices that the processor will release bus control at the end of the current bus cycle to an external bus master. This signal is an input to the IDMA and SDMA when the internal M68000 core is disabled. During total system reset, $\overline{\text{BG}} = \overline{\text{BR}}$.

$\overline{\text{BGACK}}$—Bus Grant Acknowledge

This bidirectional signal indicates that some other device besides the M68000 core has become the bus master. This signal is an input when an external device or the M68000 core owns the bus. This signal is an output when the IDMA or SDMA has become the master of the bus. If the SDMA steals a cycle from the IDMA, the $\overline{\text{BGACK}}$ pin will remain asserted continuously.

NOTE

$\overline{\text{BGACK}}$ should always be used in the external bus arbitration process. See 3.8.5.2 External Bus Arbitration for details.

5.9 INTERRUPT CONTROL PINS

The interrupt control pins are shown in Figure 5-9.

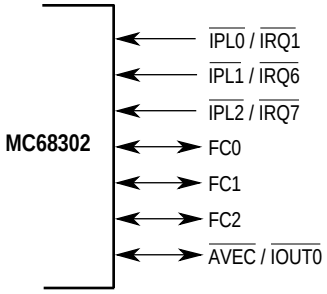


Figure 5-9. Interrupt Control Pins

These inputs have dual functionality:

- IPL0/IRQ1
- IPL1/IRQ6
- IPL2/IRQ7—Interrupt Priority Level 2–0/Interrupt Request 1,6,7

As IPL2–IPL0 (normal mode), these input pins indicate the encoded priority level of the external device requesting an interrupt. Level 7 is the highest (nonmaskable) priority; whereas, level 0 indicates that no interrupt is requested. The least significant bit is IPL0, and the most significant bit is IPL2. These lines must remain stable until the M68000 core signals an interrupt acknowledge through FC2–FC0 and A19–A16 to ensure that the interrupt is properly recognized.

As IRQ1, IRQ6, and IRQ7 (dedicated mode), these inputs indicate to the MC68302 that an external device is requesting an interrupt. Level 7 is the highest level and cannot be masked. Level 1 is the lowest level. Each one of these inputs (except for level 7) can be programmed to be either level-sensitive or edge-sensitive. The M68000 always treats a level 7 interrupt as edge sensitive.

FC2–FC0—Function Codes 2–0

These bidirectional signals indicate the state and the cycle type currently being executed. The information indicated by the function code outputs is valid whenever AS is active.

These lines are outputs when the IMP (M68000 core, SDMA, or IDMA) is the bus master and are inputs otherwise. The function codes output by the M68000 core are predefined; whereas, those output by the SDMA and IDMA are programmable. The function code lines are inputs to the chip-select logic and IMP internal register decoding in the BAR.

AVEC/IOUT0—Autovector Input/Interrupt Output 0

In normal operation, this signal functions as the input AVEC. AVEC, when asserted during an interrupt acknowledge cycle, indicates that the M68000 core should use automatic vectoring for an interrupt. This pin operates like VPA on the MC68000, but is used for au-

tomatic vectoring only. AVEC instead of DTACK should be asserted during autovectoring and should be high otherwise.

When the M68000 core is disabled, this pin operates as IOUT0. IOUT2–IOUT0 provide the interrupt request output signals from the IMP interrupt controller to an external CPU when the M68000 core is disabled.

5.10 MC68302 BUS INTERFACE SIGNAL SUMMARY

Table 5-2 and Table 5-3 summarize all bus signals discussed in the previous paragraphs. They show the direction of each pin for the following bus masters: M68000 core, IDMA, SDMA (includes DRAM refresh), and external. Each bus master can access either internal dual-port RAM and registers or an external device or memory. When an external bus master accesses the internal dual-port RAM or registers, the access may be synchronous or asynchronous.

When the M68000 core is disabled, BR and BG change their direction, and BCLR becomes bidirectional.

Table 5-2. Bus Signal Summary—Core and External Master

		M68000 Core Master Access To		External Master Access To	
Signal Name	**Pin Type**	**Internal Memory Space**	**External Memory Space**	**Internal Memory Space**	**External Memory Space**
A23–A1, FC2–FC0, AS, UDS, LDS, R/W, RMC	I/O	O	O	I	I
BCLR	I/O Open Drain	O	O	O	O
IAC	O	O	O	O	O
D15–D0 Read	I/O	O	I	O	I
D15–D0 Write	I/O	O	O	I	I
DTACK	I/O	O	**	O	**
BR	I/O Open Drain	I	I	I	I
BG	I/O	O	O	O	O
BGACK	I/O	I	I	I	I
HALT	I/O Open Drain	I/O	I/O	I	I
RESET	I/O Open Drain	I/O	I/O	I	I
BERR	I/O Open Drain	I/O***	I/O***	I/O***	I/O***
IPL2–IPL0	I	I	I	I	I
AVEC	I	I	I	I	I
IOUT2–IOUT0	O	O	O	O	O

**If DTACK is generated automatically (internally) by the chip-select logic, then it is an output. Otherwise, it is an input.
***BERR is an open-drain output, and may be asserted by the IMP when the hardware watchdog is used or when the chip-select logic detects address conflict or write protect violation. BERR may be asserted by external logic in all cases.

Table 5-3. Bus Signal Summary—IDMA and SDMA

		IDMA Master Access To		SDMA Master Access To	
Signal Name	**Pin Type**	**Internal Memory Space**	**External Memory Space**	**Internal Memory Space**	**External Memory Space**
A23–A1, FC–FC0, $\overline{AS}$, $\overline{UDS}$, $\overline{LDS}$, R/$\overline{W}$, $\overline{RMC}$	I/O	O	O	N/A	O
$\overline{BCLR}$	I/O Open Drain	I #	I #	N/A	O
IAC	O	O	O	N/A	O
D15—D0 Read	I/O	O	I	N/A	I
D15—D0 Write	I/O	O	O	N/A	O
$\overline{DTACK}$	I/O	O	**	N/A	**
$\overline{BR}$	I/O	O ##	O ##	N/A	O ##
$\overline{BG}$	I/O	I ##	I ##	N/A	I ##
$\overline{BGACK}$	I/O	O	O	N/A	O
$\overline{HALT}$	I/O Open Drain	I	I	N/A	I
$\overline{RESET}$	I/O Open Drain	I	I	N/A	I
$\overline{BERR}$	I/O Open Drain	I/O***	I/O***	N/A	I/O***

**If $\overline{DTACK}$ is generated automatically (internally) by the chip-select logic, then it is an output. Otherwise, it is an input.

***$\overline{BERR}$ is an open-drain output, and may be asserted by the IMP when the hardware watchdog is used or when the chip-select logic detects address conflict or write protect violation. $\overline{BERR}$ may be asserted by external logic in all cases.

Applies to disable CPU mode only. The internal signal IBCLR is used otherwise.

Applies to disable CPU mode only, otherwise N/A.

5.11 PHYSICAL LAYER SERIAL INTERFACE PINS

The physical layer serial interface has 24 pins, and all but one of them have multiple functionality. The pins can be used in a variety of configurations in ISDN or non-ISDN environments. Table 5-3 shows the functionality of each group of pins and their internal connection to the three SCC and one SCP controllers. The physical layer serial interface can be configured for non-multiplexed operation (NMSI) or multiplexed operation that includes IDL, GCI, and PCM highway modes. IDL and GCI are ISDN interfaces. When working in one of the multiplexed modes, the NMSI1/ISDN physical interface can be connected to all three SCC controllers.

Table 5-4. Serial Interface Pin Functions

First Function	Connected To	Second Function	Connected To
NMSI1 (8)	SCC1 Controller	ISDN Interface	SCC1/SCC2/SCC3
NMSI2 (8)	SCC2 Controller	PIO—Port A	Parallel I/O
NMSI3 (5)	SCC3 Controller	PIO—Port A	Parallel I/O
NMSI3 (3)	SCC3 Controller	SCP	SCP Controller

NOTE: Each one of the parallel I/O pins can be configured individually.

5.12 TYPICAL SERIAL INTERFACE PIN CONFIGURATIONS

Table 5-4 shows typical configurations of the physical layer interface pins for an ISDN environment. Table 5-6 shows potential configurations of the physical layer interface pins for a non-ISDN environment. The IDMA, IACK, and timer pins can be used in all applications either as dedicated functions or as PIO pins.

Table 5-5. Typical ISDN Configurations

Pins	Connected To	Used As
NMSI1 or ISDN I/F	SCC1 and SCC3	SCC1 Used as ISDN D-ch SCC3 Used as ISDN B2-ch
NMSI2	SCC2	SCC2 is Connected to Terminal
NMSI3	PA12–PA8 SCP	PIO (Extra Modem Signals and SCP Select Signals) Status/Control Exchange

NOTES:
1. ISDN environment with SCP port for status/control exchange and with existing terminal (for rate adaption).
2. D-ch is used for signaling.
3. B1-ch is used for voice (external CODEC required).
4. B2-ch is used for data transfer.

Table 5-6. Typical Generic Configurations

Pins	Connected To	Used As
NMSI1 or ISDN I/F	SCC1	Terminal with Modem
NMSI2	SCC2	Terminal with Modem
NMSI3 (5)	SCC3	Terminal without Modem
NMSI3 (3)	SCP	Status/Control Exchange

NOTE: Generic environment with three SCC ports (any protocol) and the SCP port. SCC3 does not use modem control signals.

5.13 NMSI1 OR ISDN INTERFACE PINS

The NMSI1 or ISDN interface pins are shown in Figure 5-10.

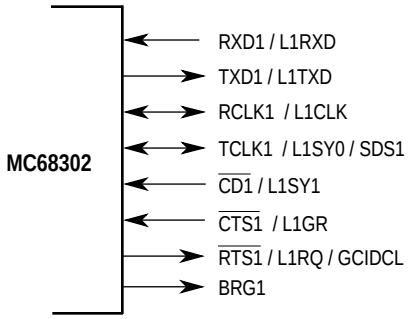


Figure 5-10. NMSI1 or ISDN Interface Pins

These eight pins can be used either as NMSI1 in nonmultiplexed serial interface (NMSI) mode or as an ISDN physical layer interface in IDL, GCI, and PCM highway modes. The input buffers have Schmitt triggers.

Table 5-7 shows the functionality of each pin in NMSI, GCI, IDL, and PCM highway modes.

Table 5-7. Mode Pin Functions

Signal Name	NMSI1		GCI		IDL		PCM	
RXD1/L1RXD	I	RXD1	I	L1RXD	I	L1RXD	I	L1RXD
TXD1/L1TXD	O	TXD1	O	L1TXD	O	L1TXD	O	L1TXD
RCLK1/L1CLK	I/O	RCLK1	I	L1CLK	I	L1CLK	I	L1CLK
TCLK1/L1SY0	I/O	TCLK1	O	SDS1	O	SDS1	I	L1SY0
$\overline{\text{CD1}}$/L1SY1	I	$\overline{\text{CD1}}$	I	L1SYNC	I	L1SYNC	I	L1SY1
$\overline{\text{CTS1}}$/L1GR	I	$\overline{\text{CTS1}}$	I	L1GR	I	L1GR		
$\overline{\text{RTS1}}$/L1RQ	O	$\overline{\text{RTS1}}$	O	GCIDCL	O	L1RQ	O	$\overline{\text{RTS}}$
BRG1	O	BRG1	O	BRG1	O	BRG1	O	BRG1

NOTES:
1. In IDL and GCI mode, SDS2 is output on the PA7 pin.
2. CD1 may be used as an external sync in NMSI mode.
3. $\overline{\text{RTS}}$ is the $\overline{\text{RTS1}}$, $\overline{\text{RTS2}}$, or RTS3 pin according to which SCCs are connected to the PCM highway.

RXD1/L1RXD—Receive Data/Layer-1 Receive Data

This input is used as the NMSI1 receive data in NMSI mode and as the receive data input in IDL, GCI, and PCM modes.

TXD1/L1TXD—Transmit Data/Layer-1 Transmit Data

This output is used as NMSI1 transmit data in NMSI mode and as the transmit data output in IDL, GCI, and PCM modes. TXD1 may be configured as an open-drain output in NMSI mode. L1TXD in IDL and PCM mode is a three-state output. In GCI mode, it is an open-drain output.

RCLK1/L1CLK—Receive Clock/Layer-1 Clock

This pin is used as an NMSI1 bidirectional receive clock in NMSI mode or as an input clock in IDL, GCI, and PCM modes. In NMSI mode, this signal is an input when SCC1 is working with an external clock and is an output when SCC1 is working with its baud rate generator. The RCLK1 output can be three-stated by setting bit 12 in the CKCR register (see 3.9 Clock Control Register).

TCLK1/L1SY0/SDS1—Transmit Clock/PCM Sync/Serial Data Strobe 1

This pin is used as an NMSI1 bidirectional transmit clock in NMSI mode, as a sync signal in PCM mode, or as the SDS1 output in IDL/GCI modes. In NMSI mode, this signal is an input when SCC1 is working with an external clock and is an output when SCC1 is working with its baud rate generator. The TCLK1 output can be three-stated by setting bit 13 in the CKCR register (see 3.9 Clock Control Register).

NOTE

When using SCC1 in the NMSI mode with the internal baud rate generator operating, the TCLK1 and RCLK1 pins will always output the baud rate generator clock unless disabled in the CKCR register. Thus, if a dynamic selection between an internal and external clock source is required in an application, the clock pins should be disabled first in the CKCR register before switching the TCLK1 and RCLK1 lines. On SCC2 and SCC3, contention may be avoided by disabling the clock line outputs in the PACNT register.

In PCM mode, L1SY1–L1SY0 are encoded signals used to create channels that can be independently routed to the SCCs.

Table 5-8. PCM Mode Signals

L1SY1	L1SY0	Data (L1RXD, L1TXD) is Routed to SCC
0	0	L1TXD is Three-Stated, L1RXD is Ignored
0	1	CH-1
1	0	CH-2
1	1	CH-3

NOTE: CH-1, 2, and 3 are connected to the SCCs as determined in the SIMODE register.

In IDL/GCI modes, the SDS2–SDS1 outputs may be used to route the B1 and/or B2 channels to devices that do not support the IDL or GCI buses. This is configured in the serial interface mode (SIMODE) and serial interface mask (SIMASK) registers.

$\overline{\text{CD1}}$/L1SY1—Carrier Detect/Layer-1 Sync

This input is used as the NMSI1 carrier detect ($\overline{\text{CD}}$) pin in NMSI mode, as a PCM sync signal in PCM mode, and as an L1SYNC signal in IDL/GCI modes.

If the $\overline{\text{CD1}}$ pin has changed for more than one receive clock cycle, the IMP asserts the appropriate bit in the SCC1 event register. If the SCC1 channel is programmed not to support $\overline{\text{CD1}}$ automatically (in the SCC1 mode register), then this pin may be used as an external interrupt source. The current value of $\overline{\text{CD1}}$ may be read in the SCCS1 register. See 4.5.8.3 SCC Status Register (SCCs) for details. $\overline{\text{CD1}}$ may also be used as an external sync in NMSI mode.

$\overline{\text{CTS1}}$/L1GR—Clear to Send/Layer-1 Grant

This input is the NMSI1 $\overline{\text{CTS}}$ signal in the NMSI mode or the grant signal in the IDL/GCI mode. If this pin is not used as a grant signal in GCI mode, it should be connected to V_{DD}.

If the $\overline{\text{CTS1}}$ pin has changed for more than one transmit clock cycle, the IMP asserts the appropriate bit in the SCC1 event register and optionally aborts the transmission of that frame.

If SCC1 is programmed not to support $\overline{\text{CTS1}}$ (in the SCC1 mode register), then this pin may be used as an external interrupt source. The current value of the $\overline{\text{CTS1}}$ pin may be read in the SCCS1 register. See 4.5.8.3 SCC Status Register (SCCs) for details.

$\overline{\text{RTS1}}$/L1RQ/GCIDCL—Request to Send/Layer-1 Request/GCI Clock Out

This output is the NMSI1 $\overline{\text{RTS}}$ signal in NMSI mode, the IDL request signal in IDL mode, or the GCI data clock output in GCI mode.

$\overline{\text{RTS1}}$ is asserted when SCC1 (in NMSI mode) has data or pad (flags or syncs) to transmit.

In GCI mode this pin is used to output the GCI data clock.

BRG1—Baud Rate Generator 1

This output is always the baud rate generator clock of SCC1. (This pin used to be NC2.) The BRG clock output on the BRG pins is 180 degrees out of phase with the internal BRG clock output on the RCLK and TCLK pins. This statement applies to all BRG pins: BRG1, BRG2, and BRG3. The BRG1 output can be disabled by setting bit 11 in the CKCR register (see 3.9 Clock Control Register). When BRG1 is disabled the pin is driven high.

5.14 NMSI2 PORT OR PORT A PINS

The NMSI2 port or port A pins are shown in Figure 5-11.

These eight pins can be used either as the NMSI2 port or as a general-purpose parallel I/O port. Each one of these pins can be configured individually to be general-purpose I/O pins or a dedicated function in NMSI2. When they are used as NMSI2 pins, they function exactly as the NMSI1 pins in NMSI mode.

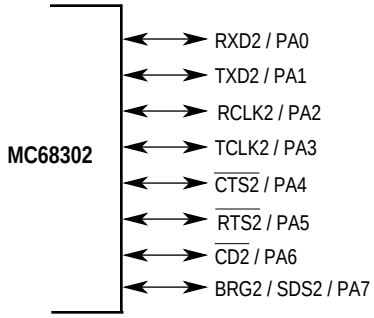


Figure 5-11. NMSI2 Port or Port A Pins

The PA7 signal in dedicated mode becomes serial data strobe 2 (SDS2) in IDL and GCI modes. In IDL/GCI modes, the SDS2–SDS1 outputs may be used to route the B1 and/or B2 channels to devices that do not support the IDL or GCI buses. This is configured in the SIMODE and SIMASK registers. If SCC2 is in NMSI mode, this pin operates as BRG2, the output of the SCC2 baud rate generator, unless SDS2 is enabled to be asserted during the B1 or B2 channels of ISDN (bits SDC2–SDC1 of SIMODE). SDS2/BRG2 may be temporarily disabled by configuring it as a general-purpose output pin. The input buffers have Schmitt

triggers. TCLK2 acts as the SCC2 baud rate generator output if SCC2 is in one of the multiplexed modes.

- RXD2/PA0
- TXD2/PA1
- RCLK2/PA2
- TCLK2/PA3
- $\overline{\text{CTS2}}$/PA4
- $\overline{\text{RTS2}}$/PA5
- $\overline{\text{CD2}}$/PA6
- SDS2/PA7/BRG2

Table 5-9. Baud Rate Generator Outputs

Source	NMSI	GCI	IDL	PCM
SCC1	BRG1	BRG1	BRG1	BRG1
SCC2	BRG2	TCLK2	TCLK2	TCLK2
SCC3	BRG3	TCLK3	TCLK3	TCLK3

NOTE: In NMSI mode, the baud rate generator outputs can also appear on the RCLK and TCLK pins as programmed in the SCON register.

5.15 NMSI3 PORT OR PORT A PINS OR SCP PINS

The NMSI3 port or port A pins or SCP pins are shown in Figure 5-12.

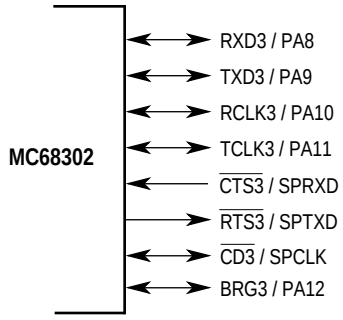


Figure 5-12. NMSI3 Port or Port A Pins or SCP Pins

These eight pins can be used either as the NMSI3 port or as the NMSI3 port (less three modem lines) and the SCP port. If the SCP is enabled (EN bit in SPMODE register is set), then the three lines are connected to the SCP port. Otherwise, they are connected to the SCC3 port.

Each of the port A I/O pins can be configured individually to be general-purpose I/O pins or a dedicated function in NMSI3. When they are used as the NMSI3 pins, they function exactly

as the NMSI1 pins (see the previous description). The input buffers have Schmitt triggers. TCLK3 acts as the SCC3 baud rate generator output if SCC3 is in one of the multiplexed modes.

- RXD3/PA8
- TXD3/PA9
- RCLK3/PA10
- TCLK3/PA11

SPRXD/$\overline{\text{CTS3}}$—SCP Receive Serial Data/NMSI3 Clear-to-Send Pin

This signal functions as the SCP receive data input or may be used as the NMSI3 $\overline{\text{CTS}}$ input pin.

SPTXD/$\overline{\text{RTS3}}$—SCP Transmit Serial Data/NMSI3 Request-to-Send Pin

This output is the SCP transmit data output or may be used as the NMSI3 $\overline{\text{RTS}}$ pin.

SPCLK/$\overline{\text{CD3}}$—SCP Clock/NMSI3 CD Pin

This bidirectional signal is used as the SCP clock output or the NMSI3 $\overline{\text{CD3}}$ input pin.

PA12/BRG3

This pin functions as bit 12 of port A or may be used as the SCC3 baud rate generator output clock when SCC3 is operating in NMSI mode.

5.16 IDMA OR PORT A PINS

The IDMA or port A pins are shown in Figure 5-13.

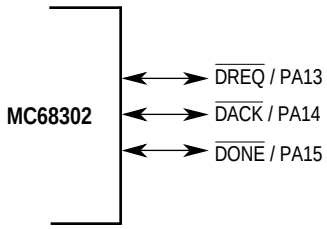


Figure 5-13. IDMA or Port A Pins

Each of these three pins can be used either as dedicated pins for the IDMA signals or as general-purpose parallel I/O port A pins. Note that even if one or more of the IDMA pins are used as general-purpose I/O pins, the IDMA can still be used. For example, if $\overline{\text{DONE}}$ is not needed by the IDMA, it can be configured as a general-purpose I/O pin. If the IDMA is used for memory-to-memory transfers only, then all three pins can be used as general-purpose I/O pins. The input buffer of $\overline{\text{DACK}}$ has a Schmitt trigger.

$\overline{\text{DREQ}}$/PA13—DMA Request

This input is asserted by a peripheral device to request an operand transfer between that peripheral device and memory. In the cycle steal request generation mode, this input is edge-sensitive. In burst mode, it is level-sensitive.

$\overline{\text{DACK}}$/PA14—DMA Acknowledge

This output, asserted by the IDMA, signals to the peripheral that an operand is being transferred in response to a previous transfer request.

$\overline{\text{DONE}}$/PA15—DONE

This bidirectional, open-drain signal is asserted by the IDMA or by a peripheral device during any IDMA bus cycle to indicate that the data being transferred is the last item in a block. The IDMA asserts this signal as an output during a bus cycle when the byte count register is decremented to zero. Otherwise, this pin is an input to the IDMA to terminate IDMA operation.

5.17 IACK OR PIO PORT B PINS

The IACK or PIO port B pins are shown in Figure 5-14.

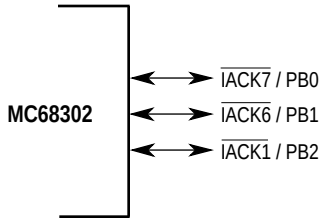


Figure 5-14. IACK or PIO Port B Pins

Each one of these three pins can be used either as an interrupt acknowledge signal or as a general-purpose parallel I/O port. Note that the IMP interrupt controller does not require the use of the IACK pins when it supplies the interrupt vector for the external source. The input buffers have Schmitt triggers.

$\overline{\text{IACK7}}$/PB0

$\overline{\text{IACK6}}$/PB1

$\overline{\text{IACK1}}$/PB2—Interrupt Acknowledge/Port B I/O

As $\overline{\text{IACK1}}$, $\overline{\text{IACK6}}$, and $\overline{\text{IACK7}}$, these active low output signals indicate to the external device that the MC68302 is executing an interrupt acknowledge cycle. The external device must then place its vector number on the lower byte of the data bus or use AVEC for autovectoring (unless internal vector generation is used).

5.18 TIMER PINS

The timer pins are shown in Figure 5-15.

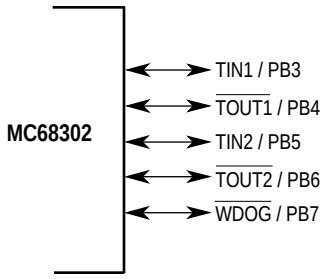


Figure 5-15. Timer Pins

Each of these five pins can be used either as a dedicated timer function or as a general-purpose port B I/O port pin. Note that the timers do not require the use of external pins. The input buffers have Schmitt triggers.

TIN1/PB3—Timer 1 Input

This input is used as a timer clock source for timer 1 or as a trigger for the timer 1 capture register. TIN1 may also be used as the external clock source for any or all three SCC baud rate generators.

$\overline{\text{TOUT1}}$/PB4—Timer 1 Output

This output is used as an active-low pulse timeout or an event overflow output (toggle) from timer 1.

TIN2/PB5—Timer 2 Input

This input can be used as a timer clock source for timer 2 or as a trigger for the timer 2 capture register.

$\overline{\text{TOUT2}}$/PB6—Timer 2 Output

This output is used as an active-low pulse timeout or as an event overflow output (toggle) from timer 2.

$\overline{\text{WDOG}}$/PB7—Watchdog Output

This active-low, open-drain output indicates expiration of the watchdog timer. $\overline{\text{WDOG}}$ is asserted for a period of 16 clock (CLKO) cycles and may be externally connected to the $\overline{\text{RESET}}$ and $\overline{\text{HALT}}$ pins to reset the MC68302. $\overline{\text{WDOG}}$ is never asserted by the on-chip hardware watchdog (see the $\overline{\text{BERR}}$ signal description). The $\overline{\text{WDOG}}$ pin function is enabled after a total system reset. It may be reassigned as the PB7 I/O pin in the PBCNT register.

5.19 PARALLEL I/O PINS WITH INTERRUPT CAPABILITY

The four parallel I/O pins with interrupt are shown in Figure 5-16.

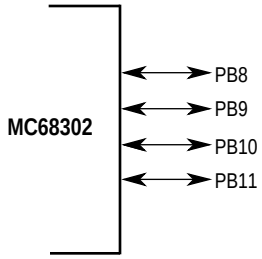


Figure 5-16. Port B Parallel I/O Pins with Interrupt

PB11–PB8—Port B Parallel I/O pins

These four pins may be configured as a general-purpose parallel I/O ports with interrupt capability. Each of the pins can be configured either as an input or an output. When configured as an input, each pin can generate a separate, maskable interrupt on a high-to-low transition. PB8 may also be used to request a refresh cycle from the DRAM refresh controller rather than as an I/O pin. The input buffers have Schmitt triggers.

5.20 CHIP-SELECT PINS

The chip-select pins are shown in Figure 5-17.

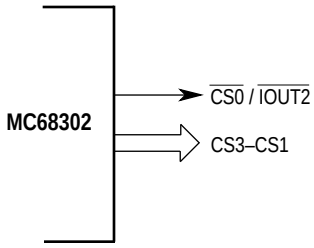


Figure 5-17. Chip-Select Pins

$\overline{CS0}$/$\overline{IOUT2}$—Chip-Select 0/Interrupt Output 2

In normal operation, this pin functions as CS0. CS0 is one of the four active-low output pins that function as chip selects for external devices or memory. It does not activate on accesses to the internal RAM or registers (including the BAR, SCR, or CKCR registers).

When the M68000 core is disabled, this pin operates as $\overline{IOUT2}$. $\overline{IOUT2}$—$\overline{IOUT0}$ provide the interrupt request output signals from the IMP interrupt controller to an external CPU when the M68000 core is disabled.

$\overline{CS3}$–$\overline{CS1}$—Chip Selects 3–1

These three active-low output pins function as chip selects for external devices or memory. $\overline{CS3}$—$\overline{CS0}$ do not activate on accesses to the internal RAM or registers (including the BAR SCR, or CKCR registers).

5.21 NO-CONNECT PINS

NC1 and NC3 output high values and are reserved for future use.

5.22 WHEN TO USE PULLUP RESISTORS

Pins that are output-only do not require external pullups. The bidirectional bus control signals require pullups since they are three-stated by the MC68302 when they are not being driven. Open-drain signals always require pullups.

Unused inputs should not be left floating. If they are input-only, they may be tied directly to V_{CC} or ground, or a pullup or pulldown resistor may be used. Unused outputs may be left unconnected. Unused I/O pins may be configured as outputs after reset and left unconnected.

If the MC68302 is to be held in reset for extended periods of time in an application (other than what occurs in normal power-on reset or board test sequences) due to a special application requirement (such as V_{DD} dropping below required specifications, etc.), then three-stated signals and inputs should be pulled up or down. This decreases stress on the device transistors and saves power.

See the $\overline{RESET}$ pin description for the condition of all pins during reset.

SECTION 6
ELECTRICAL CHARACTERISTICS

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 (CLKO pin) and possibly to one or more other signals.

6.1 MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Supply Voltage	V_{DD}	- 0.3 to + 7.0	V
Input Voltage	V_{in}	- 0.3 to + 7.0	V
Operating Temperature Range MC68302 MC68302C	T_A	 0 to 70 - 40 to 85	°C
Storage Temperature Range	T_{stg}	- 55 to + 150	°C

This device contains circuitry to protect the inputs against damage due to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum-rated voltages to his 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_{DD})

6.2 THERMAL CHARACTERISTICS

Characteristic	Symbol	Value	Unit
Thermal Resistance for PGA	θ_{JA}	25	°C/W
	θ_{JC}	2	°C/W
Thermal Resistance for CQFP	θ_{JA}	40	°C/W
	θ_{JC}	15	°C/W
Thermal Resistance for PQFP	θ_{JA}	42	°C/W
	θ_{JC}	20	°C/W

$T_J = T_A + (P_D \cdot {}_A)$
$P_D = (V_{DD} \cdot I_{DD}) + P_{I/O}$

where:
$P_{I/O}$ is the power dissipation on pins.
For T_A = 70°C and $P_{I/O}$ + 0 W, 16.67 MHz, 5.5 V, and CQFP package, the worst case value of T_J is:
T_J = 70°C + (5.5 V · 30 mA · 40°C/W) = 98.65C

6.3 POWER CONSIDERATIONS

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

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

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} = $I_{DD} \times V_{DD}$, Watts—Chip Internal Power

$P_{I/O}$ = Power Dissipation on Input and Output Pins—User Determined

For most applications $P_{I/O} < 0.3 \bullet P_{INT}$ and can be neglected.

If $P_{I/O}$ is neglected, an approximate relationship between P_D and T_J is

$$P_D = K \div (T_J + 273°C) \quad (2)$$

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

$$K = P_D \bullet (T_A + 273°C) + \theta_{JA} \bullet P_D^2 \quad (3)$$

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

6.4 POWER DISSIPATION

Characteristic	Symbol	Typ	Max	Unit
Power Dissipation at 25 MHz- Rev C.8µ (see Notes 1 & 2)	PD	85	130	mA
Power Dissipation at 25 MHz -Rev C.65µ (see Notes 1 & 2)	PD	65	90	mA
Power Dissipation at 20 MHz-Rev C.8µ (see Notes 1 & 2)	PD	65	100	mA
Power Dissipation at 20 MHz-Rev C.65µ (see Notes 1 & 2)	PD	50	80	mA
Power Dissipation at 16.67 MHz-Rev C.8µ (see Notes 1 & 2)	PD	54	85	mA
Power Dissipation at 16.67 MHz-Rev C.65µ (see Notes 1 & 2)	PD	44	70	mA
Power Dissipation at 4 MHz-Rev C.8µ (see Notes 1,2 & 3)	PD	40	-	mA
Power Dissipation at 4 MHz-Rev C.65µ (see Notes 1, 2 & 3)	PD	27	-	mA
Power Dissipation at 3.3V 20 MHz-Rev C.65µ (see Note 2)	PD	30	60	mA
Power Dissipation at 3.3V 16.67 MHz-Rev C.65µ (see Note 2)	PD	25	50	mA

NOTES:

1. Values measured with maximum loading of 130 pF on all output pins. Typical means 5.0 V at 25°C. Maximum means guaranteed maximum over maximum temperature (85°C) and voltage (5.5 V).
2. The IMP is tested with the M68000 core executing, all three baud rate generators enabled and clocking at a rate of 64 kHz, and the two general-purpose timers running with a prescaler of 256. Power measurements are not significantly impacted by baud rate generators or timers until their clocking frequency becomes a much more sizable fraction of the system frequency than in these test conditions.
3. The M68000 core will not operate at 4 MHz. This is only for low power mode.

6.5 DC ELECTRICAL CHARACTERISTICS

Characteristic	Symbol	Min	Max	Unit
Input High Voltage (Except EXTAL)	V_{IH}	2.0	V_{DD}	V
Input Low Voltage (Except EXTAL)	V_{IL}	V_{SS} - 0.3	0.8	V
Input High Voltage (EXTAL)	V_{CIH}	4.0	VDD	V
Input Low Voltage (EXTAL)	V_{CIL}	V_{SS} - 0.3	0.6	V
Input Leakage Current	II_N	—	20	µA
Input Capacitance All Pins	C_{IN}	—	15	pF
Three-State Leakage Current (2.4/0.5 V)	I_{TSI}	—	20	µA
Open Drain Leakage Current (2.4 V)	I_{OD}	—	20	µA
Output High Voltage (I_{OH} = 400 µA) (see Note)	V_{OH}	V_{DD}–1.0	—	V
Output Low Voltage	V_{OL}			
(I_{OL} = 3.2 mA) A1–A23, PB0–PB11, FC0–FC2, $\overline{CS0}$–$\overline{CS3}$		—	0.5	V
$\overline{IAC}$, $\overline{AVEC}$, $\overline{BG}$, RCLK1, RCLK2, RCLK3, TCLK1, TCLK2, TCLK3, $\overline{RTS1}$, $\overline{RTS2}$, $\overline{RTS3}$, SDS2, PA12, RXD2, RXD3, CTS2, CD2, CD3 $\overline{DREQ}$, BRG1				
(I_{OL} = 5.3 mA) $\overline{AS}$, $\overline{UDS}$, $\overline{LDS}$, R/$\overline{W}$, $\overline{BERR}$		—	0.5	
$\overline{BGACK}$, $\overline{BCLR}$, $\overline{DTACK}$, $\overline{DACK}$, $\overline{RMC}$, D0–D15, $\overline{RESET}$				
(I_{OL} = 7.0 mA) TXD1, TXD2, TXD3		—	0.5	
(I_{OL} = 8.9 mA) $\overline{DONE}$, $\overline{HALT}$, $\overline{BR}$ (as output)		—	0.5	
(I_{OL} = 3.2 mA) CLKO		—	0.4	
Output Drive CLKO	O_{CLK}	—	50	pF
Output Drive ISDN I/F (GCI Mode)	O_{GCI}	—	150	pF
Output Drive All Other Pins	O_{ALL}	—	130	pF
Output Drive Derating Factor for CLKO of 0.030 ns/pF	O_{KF}	20	50	pF
Output Drive Derating Factor for CLKO of 0.035 ns/pF	O_{KF}	50	130	pF
Output Drive Derating Factor for All Other Pins 0.035 ns/pF	O_{KF}	20	130	pF
Output Drive Derating Factor for All Other Pins 0.055 ns/pF	O_{KF}	130	220	pF
Power	V_{DD}	4.5	5.5	V
Common	V_{SS}	0	0	V

NOTE: The maximum I_{OH} for a given pin is one-half the I_{OL} rating for that pin. For an I_{OH} between 400 µA and I_{OL}/2 mA, the minimum V_{OH} is calculated as: V_{DD} - (1 +.05 V/mA(I_{OH} -.400 mA)).

NOTE: All AC specs are assume an output load of 130pf (except for CLKO).

6.6 DC ELECTRICAL CHARACTERISTICS—NMSI1 IN IDL MODE

Characteristic	Symbol	Min	Max	Unit	Condition
Input Pin Characteristics: L1CLK, L1SY1, L1RXD, L1GR					
Input Low Level Voltage	V_{IL}	-10%	+ 20%	V	(% of V_{DD})
Input High Level Voltage	V_{IH}	V_{DD} - 20%	V_{DD} + 10%	V	
Input Low Level Current	II_L	—	± 10	µA	$V_{in} = V_{ss}$
Input High Level Current	II_H	—	± 10	µA	$V_{in} = V_{DD}$
Output Pin Characteristics: L1TXD, SDS1- SDS2, L1RQ					
Output Low Level Voltage	V_{OL}	0	1.0	V	I_{OL} = 5.0 mA
Output High Level Voltage	V_{OH}	V_{DD} - 1.0	V_{DD}	V	I_{OH} = 400 µA

6.7 AC ELECTRICAL SPECIFICATIONS—CLOCK TIMING

(see Figure 6-1, Figure 6-2, Figure 6-3, and Figure 6-4)

			16.67 MHz		20 MHz		25 MHz		
Num.	**Characteristic**	**Symbol**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Unit**
	Frequency of Operation	f	8	16.67	8	20	8	25	MHz
1	Clock Period (EXTAL) (See note 3)	t_{cyc}	60	125	50	125	40	125	ns
2, 3	Clock Pulse Width (EXTAL)	t_{CL}, t_{CH}	25	62.5	21	62.5	16	62.5	ns
4, 5	Clock Rise and Fall Times (EXTAL)	t_{Cr}, t_{Cf}	—	5	—	4	—	4	ns
5a	EXTAL to CLKO Delay (See Notes 1 and 2)	t_{CD}	2	11	2	9	2	7	ns

NOTE:
1. CLKO loading is 50 pF max.
2. CLKO skew from the rising and falling edges of EXTAL will not differ from each other by more than 1 ns, if the EXTAL rise time equals the EXTAL fall time.
3. You may not stop the clock input at any time.

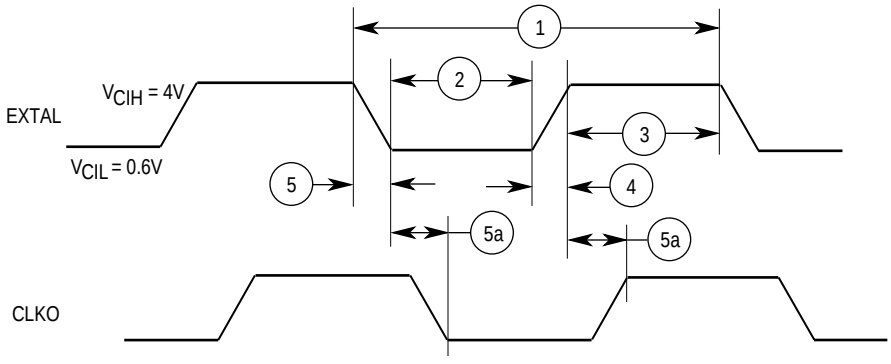


Figure 6-1. Clock Timing Diagram

6.8 AC ELECTRICAL SPECIFICATIONS—IMP BUS MASTER CYCLES

(see Figure 6-2, Figure 6-3, Figure 6-4, and Figure 6-5))

			16.67 MHz		20 MHz		25 MHz		
Num.	**Characteristic**	**Symbol**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Unit**
6	Clock High to FC, Address Valid	$t_{CHFCADV}$	0	45	0	40	0	30	ns
7	Clock High to Address, Data Bus High Impedance (Maximum)	t_{CHADZ}	—	50	—	42	—	33	ns
8	Clock High to Address, FC Invalid (Minimum)	t_{CHAFI}	0	—	0	—	0	—	ns
9	Clock High to $\overline{AS}$, $\overline{DS}$ Asserted (see Note 1)	t_{CHSL}	3	30	3	25	3	20	ns
11	Address, FC Valid to $\overline{AS}$, $\overline{DS}$ Asserted (Read) $\overline{AS}$ Asserted Write (see Note 2)	t_{AFCVSL}	15	—	12	—	10	—	ns
12	Clock Low to $\overline{AS}$, $\overline{DS}$ Negated (see Note 1)	t_{CLSH}	—	30	—	25	—	20	ns
13	$\overline{AS}$, $\overline{DS}$ Negated to Address, FC Invalid (see Note 2)	t_{SHAFI}	15	—	12	—	10	—	ns
14	$\overline{AS}$ (and $\overline{DS}$ Read) Width Asserted (see Note 2)	t_{SL}	120	—	100	—	80	—	ns
14A	$\overline{DS}$ Width Asserted, Write (see Note 2)	t_{DSL}	60	—	50	—	40	—	ns
15	$\overline{AS}$, $\overline{DS}$ Width Negated (see Note 2)	t_{SH}	60	—	50	—	40	—	ns
16	Clock High to Control Bus High Impedance	t_{CHCZ}	—	50	—	42	—	33	ns
17	$\overline{AS}$, $\overline{DS}$ Negated to R/$\overline{W}$ Invalid (see Note 2)	t_{SHRH}	15	—	12	—	10	—	ns
18	Clock High to R/$\overline{W}$ High (see Note 1)	t_{CHRH}	—	30	—	25	—	20	ns
20	Clock High to R/$\overline{W}$ Low (see Note 1)	t_{CHRL}	—	30	—	25	—	20	ns
20A	$\overline{AS}$ Asserted to R/$\overline{W}$ Low (Write) (see Notes 2 and 6)	t_{ASRV}	—	10	—	10	—	7	ns
21	Address FC Valid to R/$\overline{W}$ Low (Write) (see Note 2)	t_{AFCVRL}	15	—	12	—	10	—	ns
22	R/$\overline{W}$ Low to $\overline{DS}$ Asserted (Write) (see Note 2)	t_{RLSL}	30	—	25	—	20	—	ns
23	Clock Low to Data-Out Valid	t_{CLDO}	—	30	—	25	—	20	ns
25	$\overline{AS}$, $\overline{DS}$, Negated to Data-Out Invalid (Write) (see Note 2)	t_{SHDOI}	15	—	12	—	10	—	ns
26	Data-Out Valid to $\overline{DS}$ Asserted (Write) (see Note 2)	t_{DOSL}	15	—	12	—	10	—	ns
27	Data-In Valid to Clock Low (Setup Time on Read) (see Note 5)	t_{DICL}	7	—	6	—	5	—	ns
28	$\overline{AS}$, $\overline{DS}$ Negated to $\overline{DTACK}$ Negated (Asynchronous Hold) (see Note 2)	t_{SHDAH}	0	110	0	95	0	75	ns
29	$\overline{AS}$, $\overline{DS}$ Negated to Data-In Invalid (Hold Time on Read)	t_{SHDII}	0	—	0	—	0	—	ns
30	$\overline{AS}$, $\overline{DS}$ Negated to $\overline{BERR}$ Negated	t_{SHBEH}	0	—	0	—	0	—	ns
31	$\overline{DTACK}$ Asserted to Data-In Valid (Setup Time) (see Notes 2 and 5)	t_{DALDI}	—	50	—	42	—	33	ns
32	$\overline{HALT}$ and $\overline{RESET}$ Input Transition Time	t_{RHr}, t_{RHf}	—	150	—	150	—	150	ns
33	Clock High to $\overline{BG}$ Asserted	t_{CHGL}	—	30	—	25	—	20	ns
34	Clock High to $\overline{BG}$ Negated	t_{CHGH}	—	30	—	25	—	20	ns
35	$\overline{BR}$ Asserted to $\overline{BG}$ Asserted (see Note 11)	t_{BRLGL}	2.5	4.5	2.5	4.5	2.5	4.5	clks
36	$\overline{BR}$ Negated to $\overline{BG}$ Negated (see Note 7)	t_{BRHGH}	1.5	2.5	1.5	2.5	1.5	2.5	clks

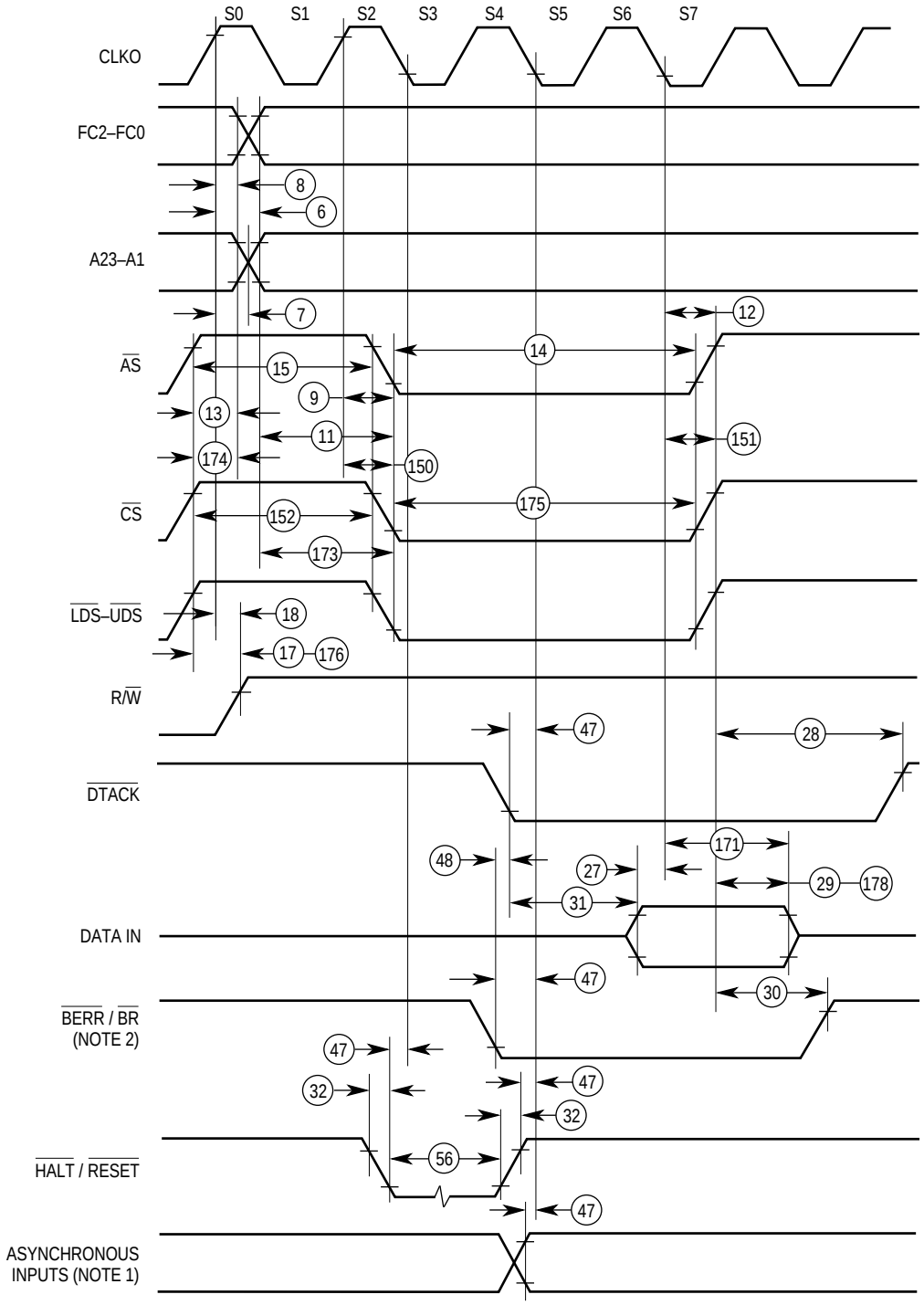
			16.67 MHz		20 MHz		25 MHz		
Num.	**Characteristic**	**Symbol**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Unit**
37	$\overline{BGACK}$ Asserted to $\overline{BG}$ Negated	t_{GALGH}	2.5	4.5	2.5	4.5	2.5	4.5	clks
37A	$\overline{BGACK}$ Asserted to $\overline{BR}$ Negated (see Note 8)	t_{GALBRH}	10	1.5	10	1.5	10	1.5	ns/ clks
38	$\overline{BG}$ Asserted to Control, Address, Data Bus High Impedance ($\overline{AS}$ Negated)	t_{GLZ}	—	50	—	42	—	33	ns
39	$\overline{BG}$ Width Negated	t_{GH}	1.5	—	1.5	—	1.5	—	clks
40	$\overline{BGACK}$ Asserted to Address Valid	t_{GALAV}	15	—	15	—	15	—	ns
41	$\overline{BGACK}$ Asserted to $\overline{AS}$ Asserted	t_{GALASA}	30	—	30	—	20	—	ns
44	$\overline{AS}$, $\overline{DS}$ Negated to $\overline{AVEC}$ Negated	t_{SHVPH}	0	50	0	42	0	33	ns
46	$\overline{BGACK}$ Width Low	t_{GAL}	1.5	—	1.5	—	1.5	—	clks
47	Asynchronous Input Setup Time (see Note 5)	t_{ASI}	10	—	10	—	7	—	ns
48	$\overline{BERR}$ Asserted to $\overline{DTACK}$ Asserted (see Notes 2 and 3)	t_{BELDAL}	10	—	10	—	7	—	ns
53	Data-Out Hold from Clock High	t_{CHDOI}	0	—	0	—	0	—	ns
55	R/$\overline{W}$ Asserted to Data Bus Impedance Change	t_{RLDBD}	0	—	0	—	0	—	ns
56	$\overline{HALT}$/$\overline{RESET}$ Pulse Width (see Note 4)	t_{HRPW}	10	—	10	—	10	—	clks
57	$\overline{BGACK}$ Negated to $\overline{AS}$, $\overline{DS}$, R/$\overline{W}$ Driven	t_{GASD}	1.5	—	1.5	—	1.5	—	clks
57A	$\overline{BGACK}$ Negated to FC	t_{GAFD}	1	—	1	—	1	—	clks
58	$\overline{BR}$ Negated to $\overline{AS}$, $\overline{DS}$, R/$\overline{W}$ Driven (see Note 7)	t_{RHSD}	1.5	—	1.5	—	1.5	—	clks
58A	$\overline{BR}$ Negated to FC (see Note 7)	t_{RHFD}	1	—	1	—	1	—	clks
60	Clock High to $\overline{BCLR}$ Asserted	t_{CHBCL}	—	30	—	25	—	20	ns
61	Clock High to $\overline{BCLR}$ High Impedance (See Note 10)	t_{CHBCH}	—	30	—	25	—	20	ns
62	Clock Low (S0 Falling Edge during read) to $\overline{RMC}$ Asserted	t_{CLRML}	—	30	—	25	—	20	ns
63	Clock High (during write) to $\overline{RMC}$ Negated	t_{CHRMH}	—	30	—	25	—	20	ns
64	$\overline{RMC}$ Negated to $\overline{BG}$ Asserted (see Note 9)	t_{RMHGL}	—	30	—	25	—	20	ns

NOTES:

1. For loading capacitance of less than or equal to 50 pF, subtract 4 ns from the value given in the maximum columns.
2. Actual value depends on clock period since signals are driven/latched on different CLKO edges. To calculate the actual spec for other clock frequencies, the user may derive the formula for each specification. First, derive the margin factor as: M = N(P/2) - Sa
   where N is the number of one-half CLKO periods between the two events as derived from the timing diagram, P is the rated clock period of the device for which the specs were derived (e.g., 60 ns with a 16.67-MHz device or 50 ns with a 20 MHz device), and Sa is the actual spec in the data sheet. Thus, for spec 14 at 16.67 MHz:
   M = 5(60 ns/2) - 120 ns = 30 ns.
   Once the margin (M) is calculated for a given spec, a new value of that spec (Sn) at another clock frequency with period (Pa) is calculated as:
   Sn = N(Pa/2) - M
   Thus for spec 14 at 12.5 MHz:
   Sn = 5(80 ns/2) - 30 ns = 170 ns.
   These two formulas assume a 50% duty cycle. Otherwise, if N is odd, the previous values N(P/2) and N(Pa/2) must be reduced by X, where X is the difference between the nominal pulse width and the minimum pulse width of the EXTAL input clock for that duty cycle.
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 a

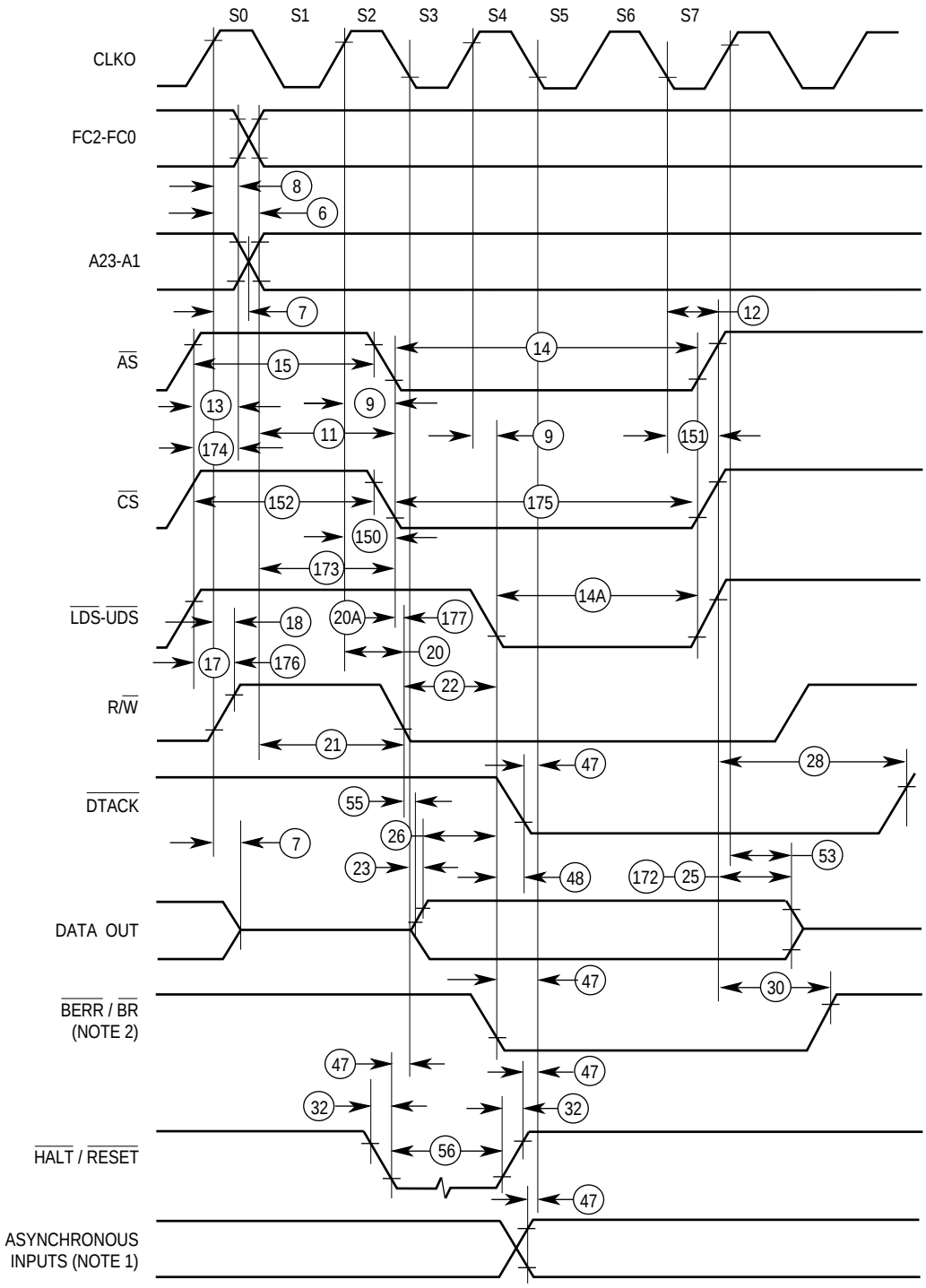
synchronous input using the asynchronous input setup time (#47).

4. For power-up, the MC68302 must be held in the reset state for 100 ms to allow stabilization of on-chip circuit. After the system is powered up #56 refers to the minimum pulse width required to reset the processor.
5. If the asynchronous input setup (#47) requirement is satisfied for $\overline{\text{DTACK}}$, the $\overline{\text{DTACK}}$ 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 $\overline{\text{AS}}$ and R/$\overline{\text{W}}$ are equally loaded (±20%), subtract 5 ns from the values given in these columns.
7. The MC68302 will negate $\overline{\text{BG}}$ and begin driving the bus if external arbitration logic negates $\overline{\text{BR}}$ before asserting $\overline{\text{BGACK}}$.
8. The minimum value must be met to guarantee proper operation. If the maximum value is exceeded, $\overline{\text{BG}}$ may be reasserted.
9. This specification is valid only when the RMCST bit is set in the SCR register.
10. Occurs on S0 of SDMA read/write access when the SDMA becomes bus master.
11. Specification may be exceeded during the TAS instruction if the RMCST bit in the SCR is set.



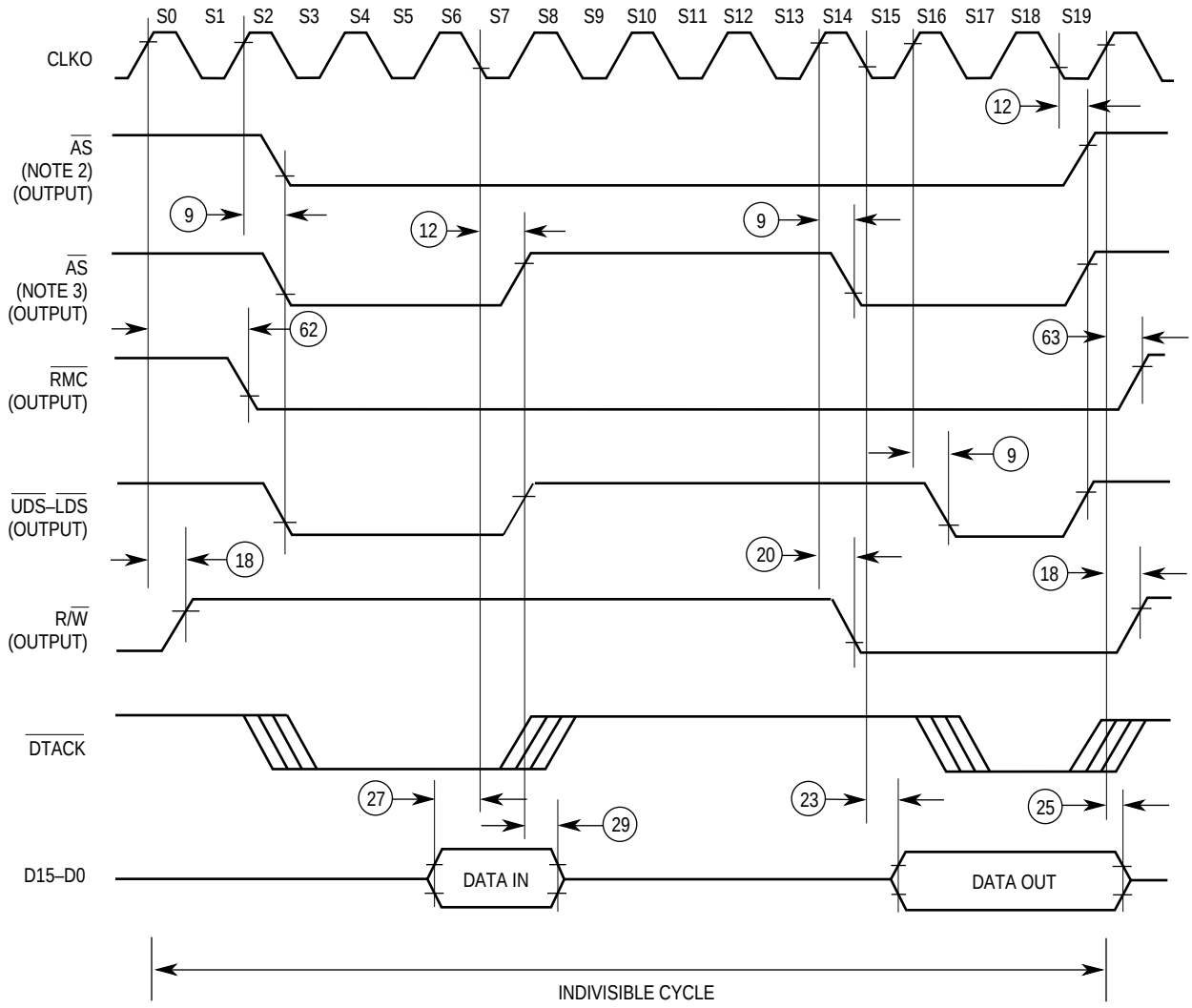
NOTES:
1. Setup time for the asynchronous inputs IPL2–IPL0 guarantees their recognition at the next falling edge of the clock.
2. BR need fall at this time only to insure being recognized at the end of the bus cycle.
3. Timing measurements are referenced to and from a low voltage of 0.8 volt and a high voltage of 2.0 volts, unless otherwise noted. The voltage swing through this range should start outside and pass through the range such that the rise or fall is linear between 0.8 volts and 2.0 volts.

Figure 6-2. Read Cycle Timing Diagram



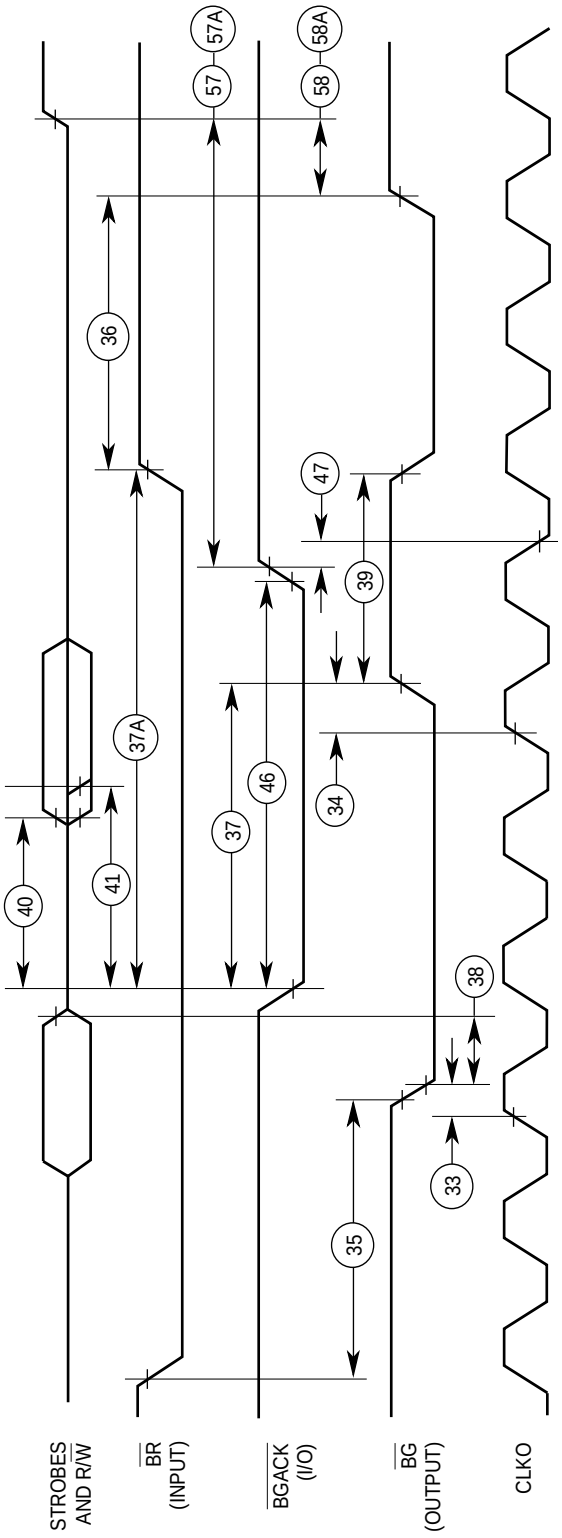
NOTES:
1. Timing measurements are referenced to and from a low voltage of 0.8 volt and a high voltage of 2.0 volts, unless otherwise noted. The voltage swing through this range should start outside and pass through the range such that the rise or fall is linear between between 0.8 volt and 2.0 volts.
2. Because of loading variations, R/W may be valid after AS even though both are initiated by the rising edge of S2 (specification #20A)
3. Each wait state is a full clock cycle inserted between S4 and S5.

Figure 6-3. Write Cycle Timing Diagram



NOTES:
1. For other timings than RMC, see Figures 6-2 and 6-3.
2. RMCST = 0 in the SCR.
3. RMCST = 1 in the SCR.
4. Wait states may be inserted between S4 and S5 during the write cycle and between S16 and S17 during the read cycle.
5. Read-modify-write cycle is generated only by the TAS instruction.

Figure 6-4. Read-Modify-Write Cycle Timing Diagram



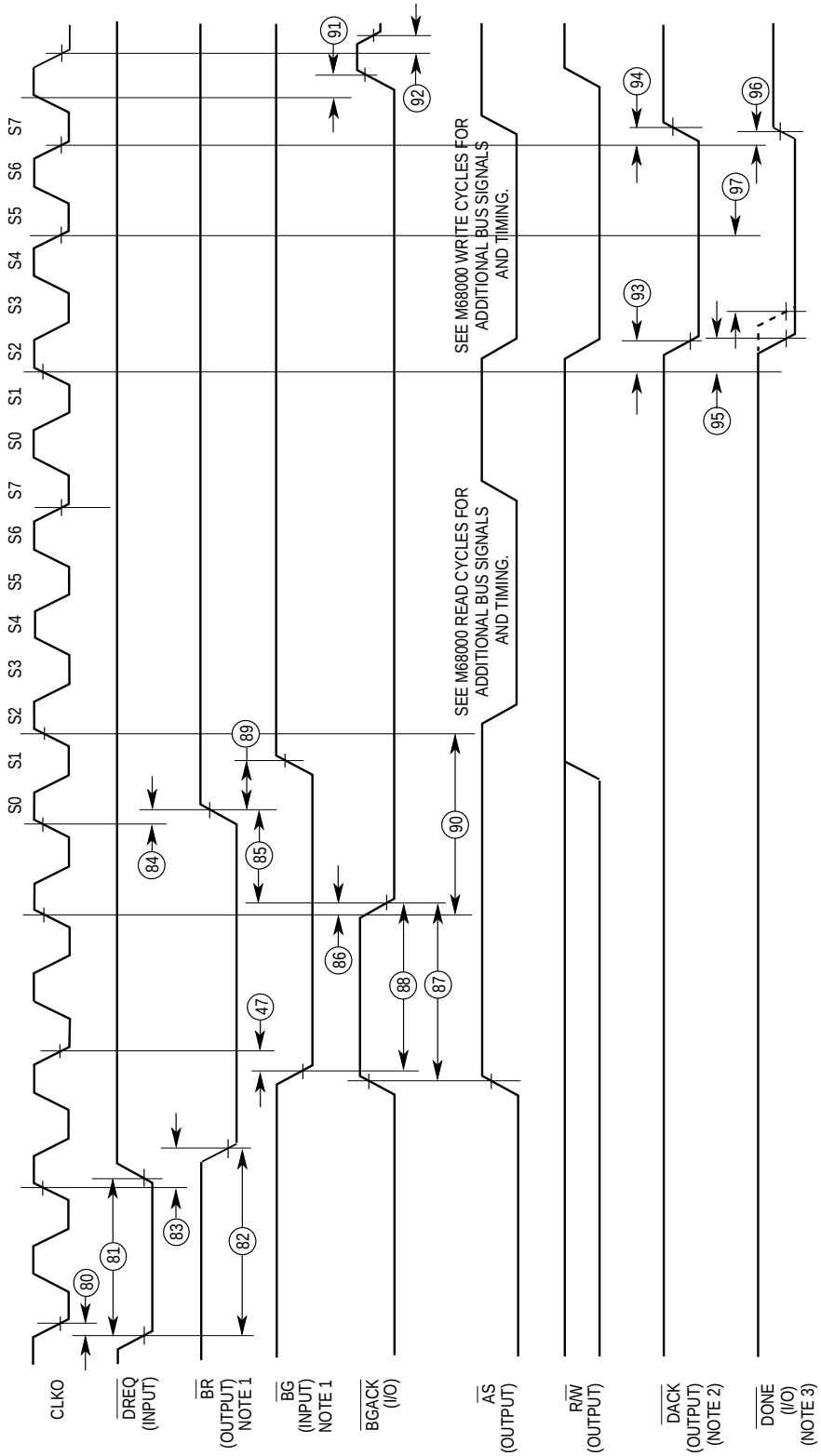
NOTE: Setup time to the clock (#47) for the asynchronous inputs BERR, BGACK, BR, DTACK, AND IPL2–IPL0 guarantees their recognition at the next falling edge of the clock.

Figure 6-5. Bus Arbitration Timing Diagram

6.9 AC ELECTRICAL SPECIFICATIONS—DMA (see Figure 6-6 and Figure 6-7)

Num.	Characteristic	Symbol	16.67 MHz		20 MHz		25 MHz		Unit
			Min	Max	Min	Max	Min	Max	
80	$\overline{DREQ}$ Asynchronous Setup Time (see Note 1)	t_{REQASI}	15	—	15	—	10	—	ns
81	$\overline{DREQ}$ Width Low (see Note 2)	t_{REQL}	2	—	2	—	2	—	clks
82	$\overline{DREQ}$ Low to $\overline{BR}$ Low (see Notes 3 and 4)	$t_{REQLBRL}$	—	2	—	2	—	2	clks
83	Clock High to $\overline{BR}$ Low (see Notes 3 and 4)	t_{CHBRL}	—	30	—	25	—	20	ns
84	Clock High to $\overline{BR}$ High Impedance (see Notes 3 and 4)	t_{CHBRZ}	—	30	—	25	—	20	ns
85	$\overline{BGACK}$ Low to $\overline{BR}$ High Impedance (see Notes 3 and 4)	t_{BKLBRZ}	30	—	25	—	20	—	ns
86	Clock High to $\overline{BGACK}$ Low	t_{CHBKL}	—	30	—	25	—	20	ns
87	$\overline{AS}$ and $\overline{BGACK}$ High (the Latest One) to $\overline{BGACK}$ Low (when $\overline{BG}$ Is Asserted)	t_{ABHBKL}	1.5	2.5 +30	1.5	2.5 +25	1.5	2.5 +20	clks ns
88	$\overline{BG}$ Low to $\overline{BGACK}$ Low (No Other Bus Master) (see Notes 3 and 4)	t_{BGLBKL}	1.5	2.5 +30	1.5	2.5 +25	1.5	2.5 +20	clks ns
89	$\overline{BR}$ High Impedance to $\overline{BG}$ High (see Notes 3 and 4)	t_{BRHBGH}	0	—	0	—	0	—	ns
90	Clock on which $\overline{BGACK}$ Low to Clock on which $\overline{AS}$ Low	$t_{CLBKLAL}$	2	2	2	2	2	2	clks
91	Clock High to $\overline{BGACK}$ High	t_{CHBKH}	—	30	—	25	—	20	ns
92	Clock Low to $\overline{BGACK}$ High Impedance	t_{CLBKZ}	—	15	—	15	—	10	ns
93	Clock High to $\overline{DACK}$ Low	t_{CHACKL}	—	30	—	25	—	20	ns
94	Clock Low to $\overline{DACK}$ High	t_{CLACKH}	—	30	—	25	—	20	ns
95	Clock High to $\overline{DONE}$ Low (Output)	t_{CHDNL}	—	30	—	25	—	20	ns
96	Clock Low to $\overline{DONE}$ High Impedance	t_{CLDNZ}	—	30	—	25	—	20	ns
97	$\overline{DONE}$ Input Low to Clock Low (Asynchronous Setup)	t_{DNLTCH}	15	—	15	—	10	—	ns

NOTES:
1. $\overline{DREQ}$ is sampled on the falling edge of CLK in cycle steal and burst modes.
2. If #80 is satisfied for $\overline{DREQ}$, #81 may be ignored.
3. $\overline{BR}$ will not be asserted while $\overline{BG}$, $\overline{HALT}$, or $\overline{BERR}$ is asserted.
4. Specifications are for DISABLE CPU mode only.
5. $\overline{DREQ}$, $\overline{DACK}$, and $\overline{DONE}$ do not apply to the SDMA channels.
6. DMA and SDMA read and write cycle timing is the same as that for the M68000 core.



NOTES:
1. BR and BG shown above are only active in disable CPU mode; otherwise, they do not apply to the diagram.
2. Assumes the ECO bit in the CMR = 1.
3. For the case when DONE is an input, assumes ECO bit in the CMR = 1.

Figure 6-6. DMA Timing Diagram (IDMA)

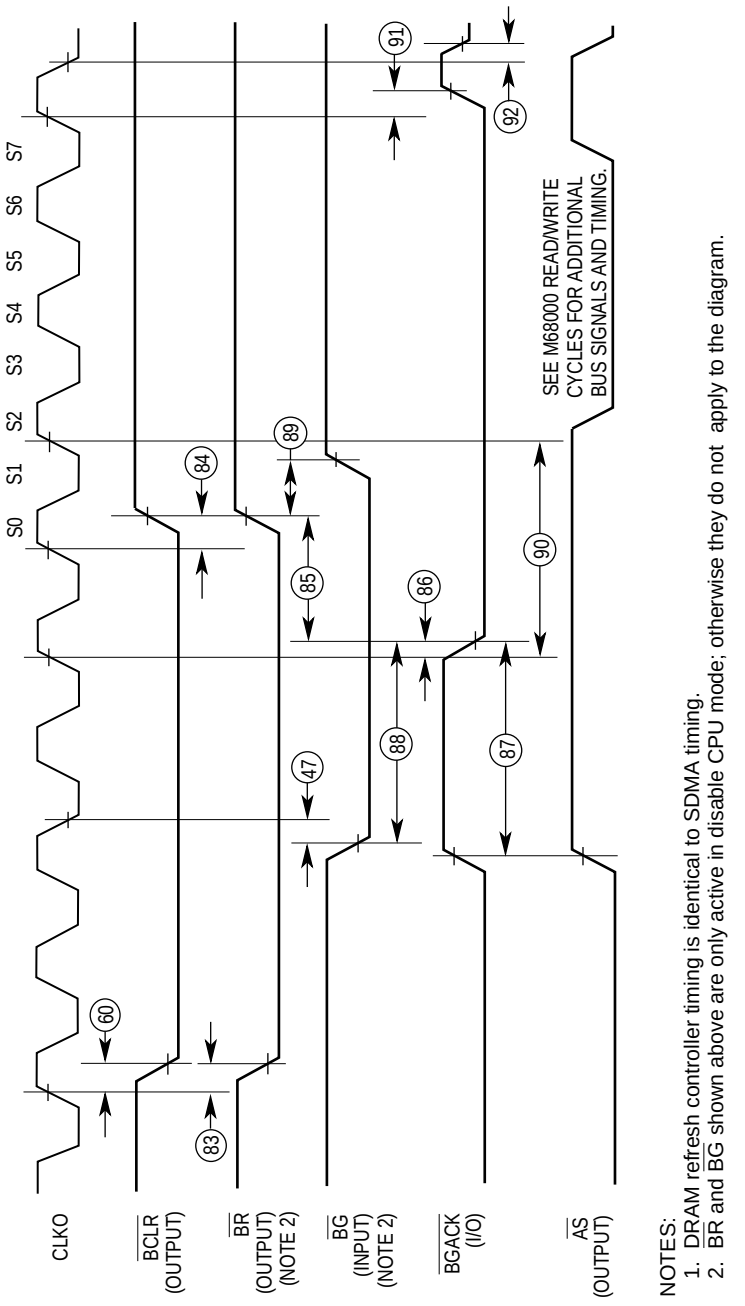


Figure 6-7. DMA Timing Diagram (SDMA)

6.10 AC ELECTRICAL SPECIFICATIONS—EXTERNAL MASTER INTERNAL ASYNCHRONOUS READ/WRITE CYCLES

(see Figure 6-8 and Figure 6-9)

			16.67 MHz		20 MHz		25 MHz		
Num.	**Characteristic**	**Symbol**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Unit**
100	R/$\overline{W}$ Valid to $\overline{DS}$ Low	t_{RWVDSL}	0	—	0	—	0	—	ns
101	$\overline{DS}$ Low to Data-In Valid	t_{DSLDIV}	—	30	—	25	—	20	ns
102	$\overline{DTACK}$ Low to Data-In Hold Time	t_{DKLDH}	0	—	0	—	0	—	ns
103	$\overline{AS}$ Valid to $\overline{DS}$ Low	t_{ASVDSL}	0	—	0	—	0	—	ns
104	$\overline{DTACK}$ Low to $\overline{AS}$, $\overline{DS}$ High	t_{DKLDSH}	0	—	0	—	0	—	ns
105	$\overline{DS}$ High to $\overline{DTACK}$ High	t_{DSHDKH}	—	45	—	40	—	30	ns
106	$\overline{DS}$ Inactive to $\overline{AS}$ Inactive	t_{DSIASI}	0	—	0	—	0	—	ns
107	$\overline{DS}$ High to R/$\overline{W}$ High	t_{DSHRWH}	0	—	0	—	0	—	ns
108	$\overline{DS}$ High to Data High Impedance	t_{DSHDZ}	—	45	—	40	—	30	ns
108A	$\overline{DS}$ High to Data-Out Hold Time (see Note)	t_{DSHDH}	0	—	0	—	0	—	ns
109A	Data Out Valid to $\overline{DTACK}$ Low	t_{DOVDKL}	15	—	15	—	10	—	ns

NOTE: If $\overline{AS}$ is negated before $\overline{DS}$, the data bus could be three-stated (spec 126) before $\overline{DS}$ is negated.

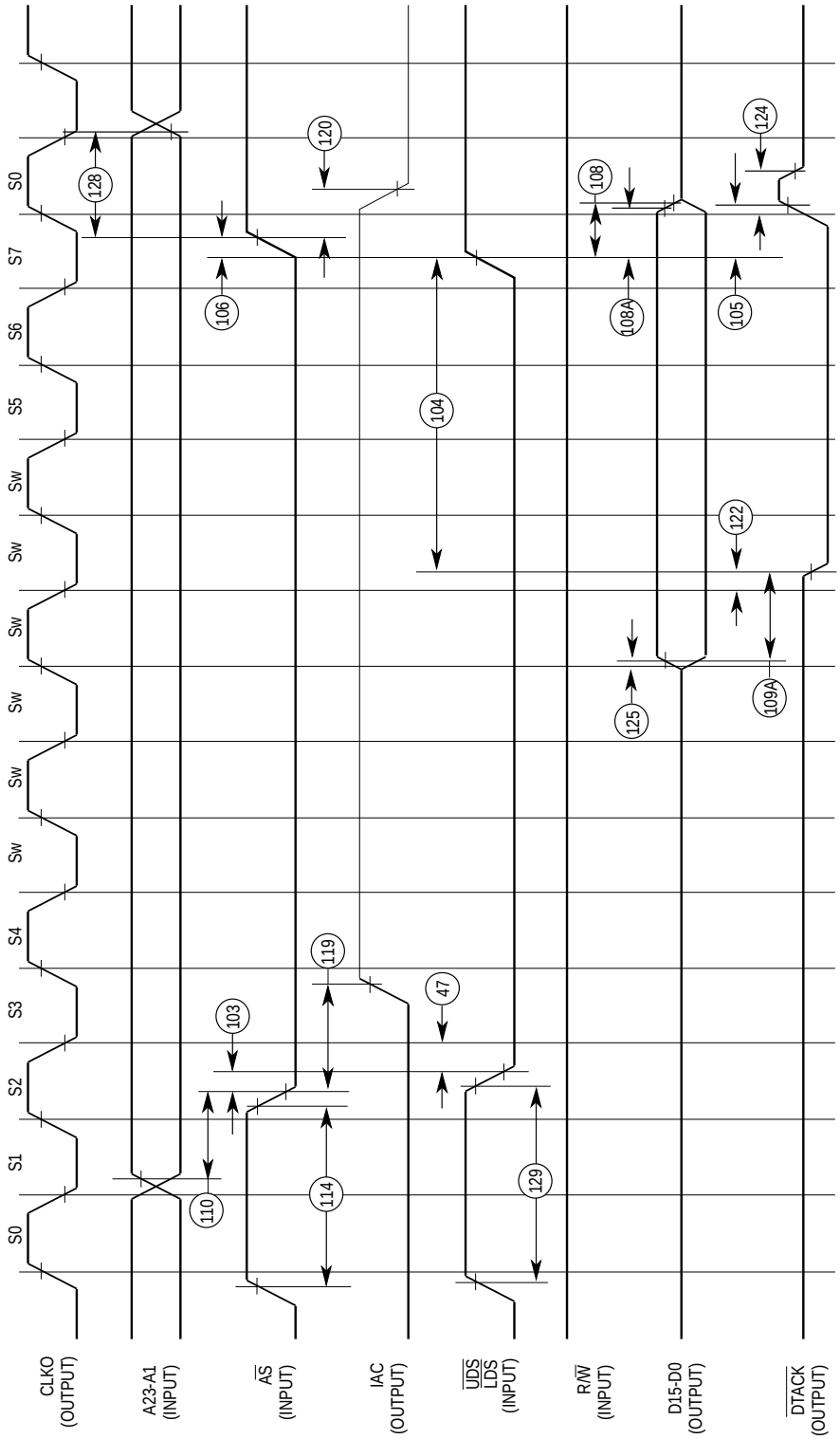


Figure 6-8. External Master Internal Asynchronous Read Cycle Timing Diagram

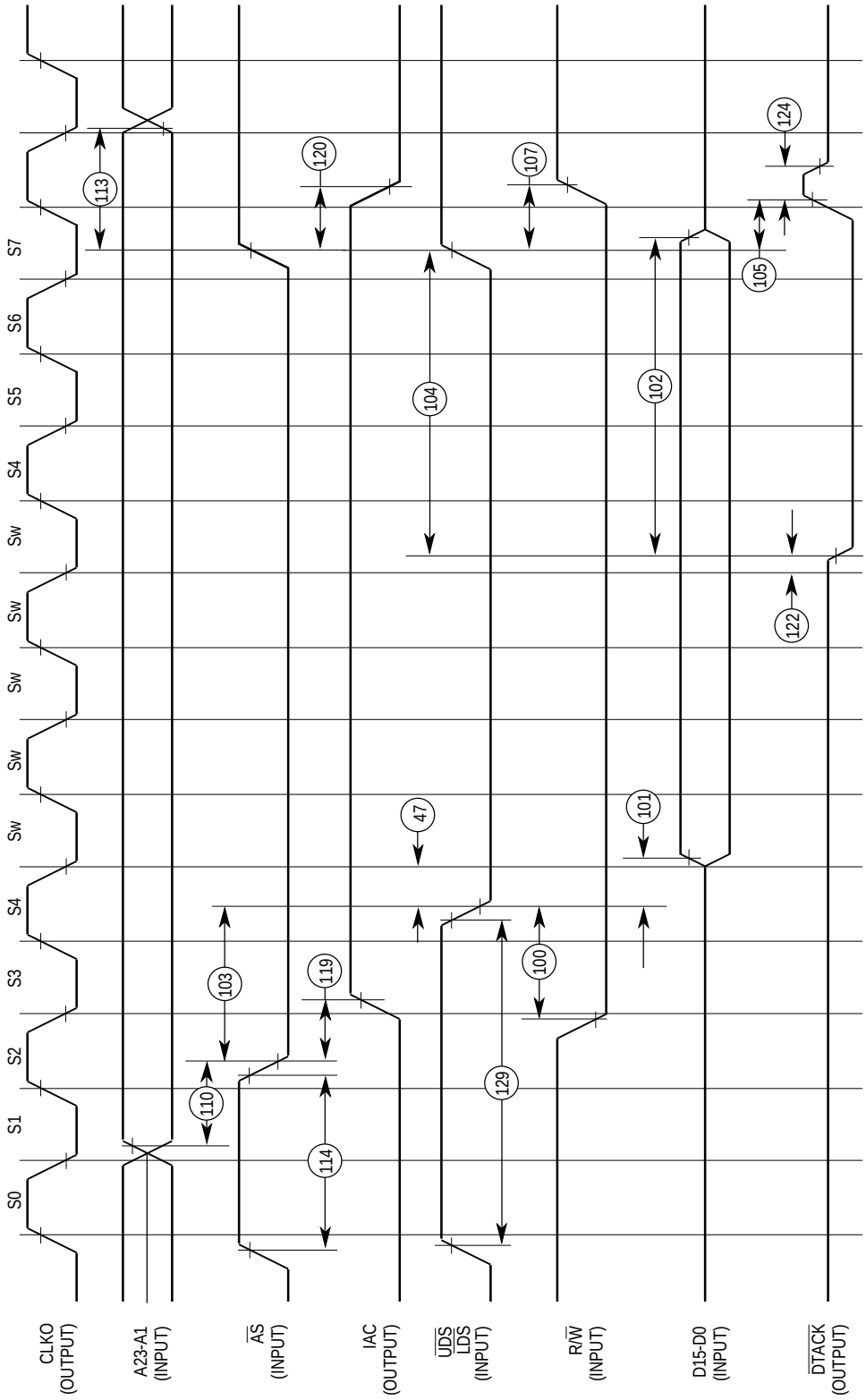


Figure 6-9. External Master Internal Asynchronous Write Cycle Timing Diagram

6.11 AC ELECTRICAL SPECIFICATIONS—EXTERNAL MASTER INTERNAL SYNCHRONOUS READ/WRITE CYCLES

(see Figure 6-10, Figure 6-11, and Figure 6-12)

			16.67 MHz		20 MHz		25 MHz		
Num.	**Characteristic**	**Symbol**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Unit**
110	Address Valid to $\overline{AS}$ Low	t_{AVASL}	15	—	12	—	10	—	ns
111	$\overline{AS}$ Low to Clock High	t_{ASLCH}	30	—	25	—	20	—	ns
112	Clock Low to $\overline{AS}$ High	t_{CLASH}	—	45	—	40	—	30	ns
113	$\overline{AS}$ High to Address Hold Time on Write	t_{ASHAH}	0	—	0	—	0	—	ns
114	$\overline{AS}$ Inactive Time	t_{ASH}	1	—	1	—	1	—	clk
115	$\overline{UDS}$/$\overline{LDS}$ Low to Clock High (see Note 2)	t_{SLCH}	40	—	33	—	27	—	ns
116	Clock Low to $\overline{UDS}$/$\overline{LDS}$ High	t_{CLSH}	—	45	—	40	—	30	ns
117	R/$\overline{W}$ Valid to Clock High (see Note 2)	t_{RWVCH}	30	—	25	—	20	—	ns
118	Clock High to R/$\overline{W}$ High	t_{CHRWH}	—	45	—	40	—	30	ns
119	$\overline{AS}$ Low to IAC High	t_{ASLIAH}	—	40	—	35	—	27	ns
120	$\overline{AS}$ High to IAC Low	t_{ASHIAL}	—	40	—	35	—	27	ns
121	$\overline{AS}$ Low to $\overline{DTACK}$ Low (0 Wait State)	t_{ASLDTL}	—	45	—	40	—	30	ns
122	Clock Low to $\overline{DTACK}$ Low (1 Wait State)	t_{CLDTL}	—	30	—	25	—	20	ns
123	$\overline{AS}$ High to $\overline{DTACK}$ High	t_{ASHDTH}	—	45	—	40	—	30	ns
124	$\overline{DTACK}$ High to $\overline{DTACK}$ High Impedance	t_{DTHDTZ}	—	15	—	15	—	10	ns
125	Clock High to Data-Out Valid	t_{CHDOV}	—	30	—	25	—	20	ns
126	$\overline{AS}$ High to Data High Impedance	t_{ASHDZ}	—	45	—	40	—	30	ns
127	$\overline{AS}$ High to Data-Out Hold Time	t_{ASHDOI}	0	—	0	—	0	—	ns
128	$\overline{AS}$ High to Address Hold Time on Read	t_{ASHAI}	0	—	0	—	0	—	ns
129	$\overline{UDS}$/$\overline{LDS}$ Inactive Time	t_{SH}	1	—	1	—	1	—	clk
130	Data-In Valid to Clock Low	t_{CLDIV}	30	—	25	—	20	—	ns
131	Clock Low to Data-In Hold Time	t_{CLDIH}	15	—	12	—	10	—	ns

NOTES:

1. Synchronous specifications above are valid only when SAM = 1 in the SCR.
2. It is required that this signal not be asserted prior to the previous rising CLKO edge (i.e., in the previous clock cycle). It must be recognized by the IMP no sooner than the rising CLKO edge shown in the diagram.

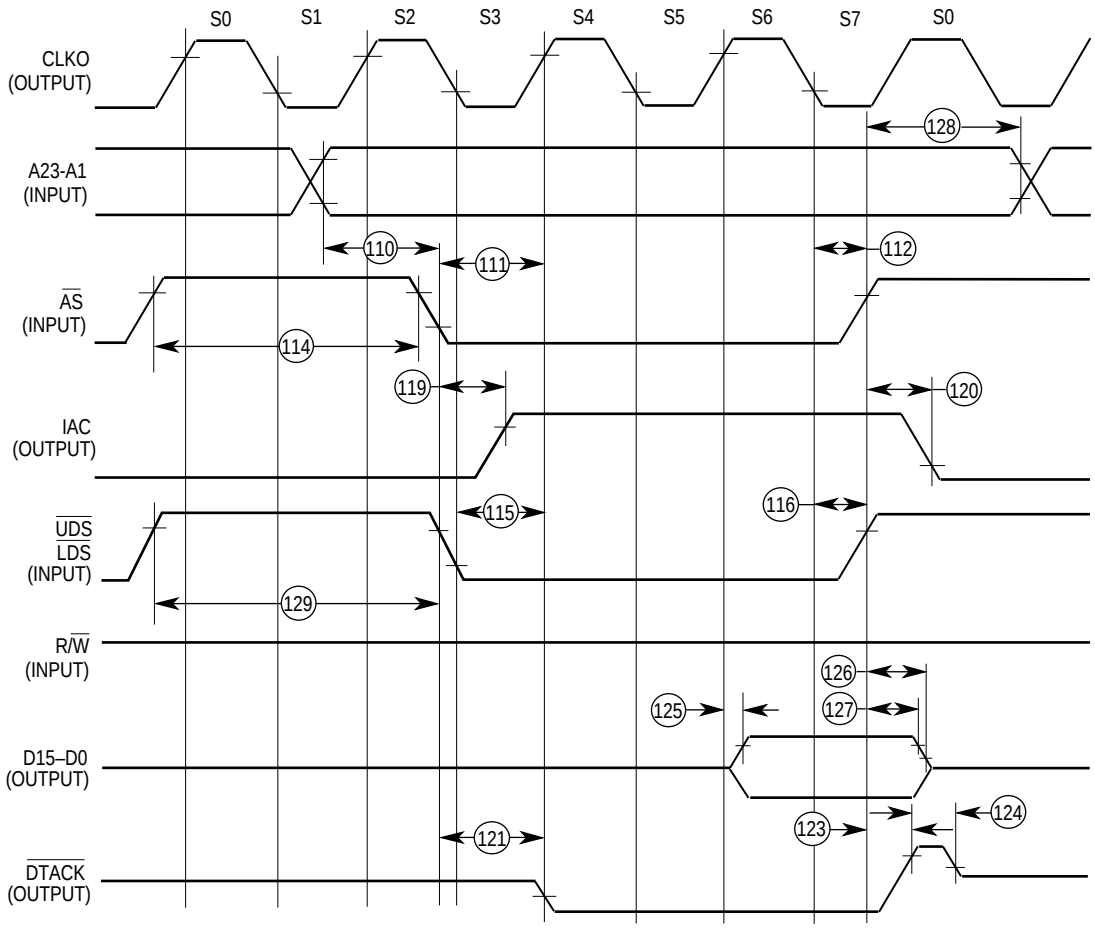


Figure 6-10. External Master Internal Synchronous Read Cycle Timing Diagram

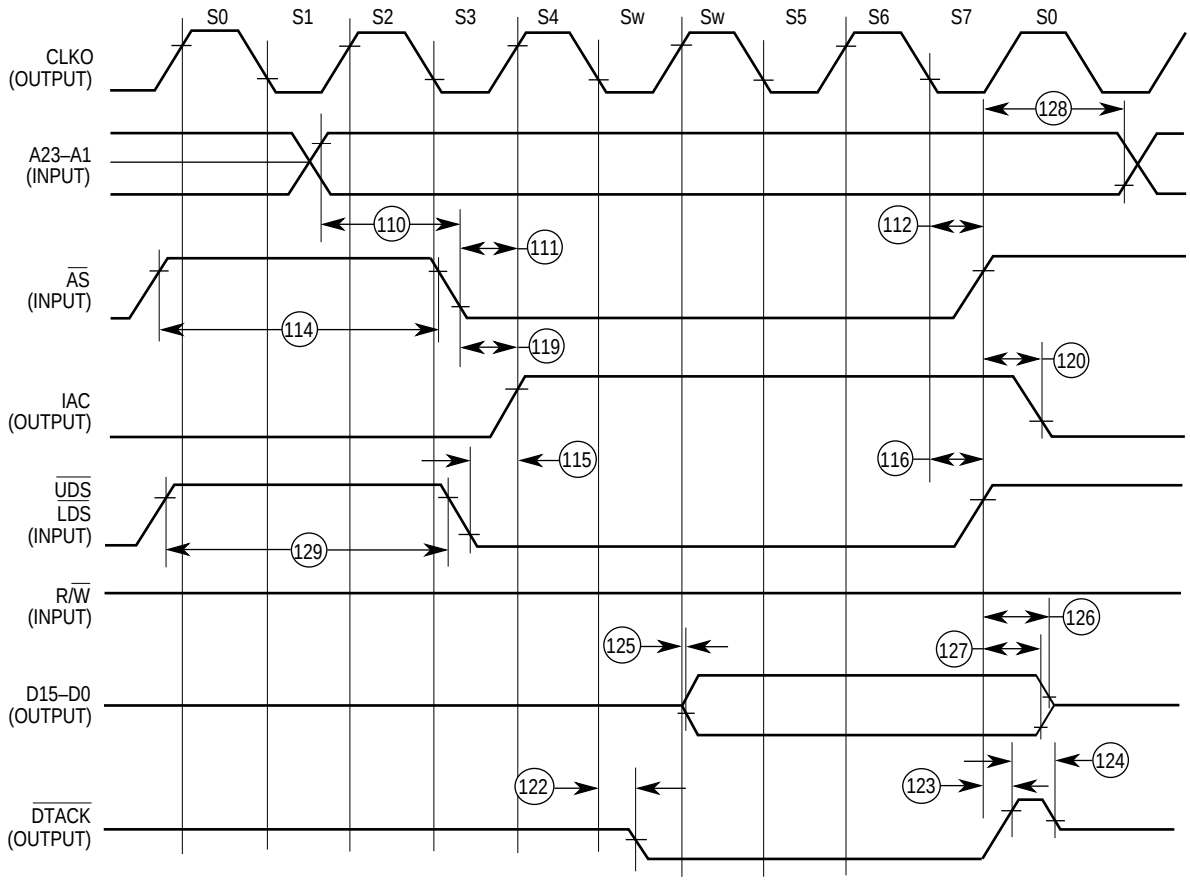


Figure 6-11. External Master Internal Synchronous Read Cycle Timing Diagram (One Wait State)

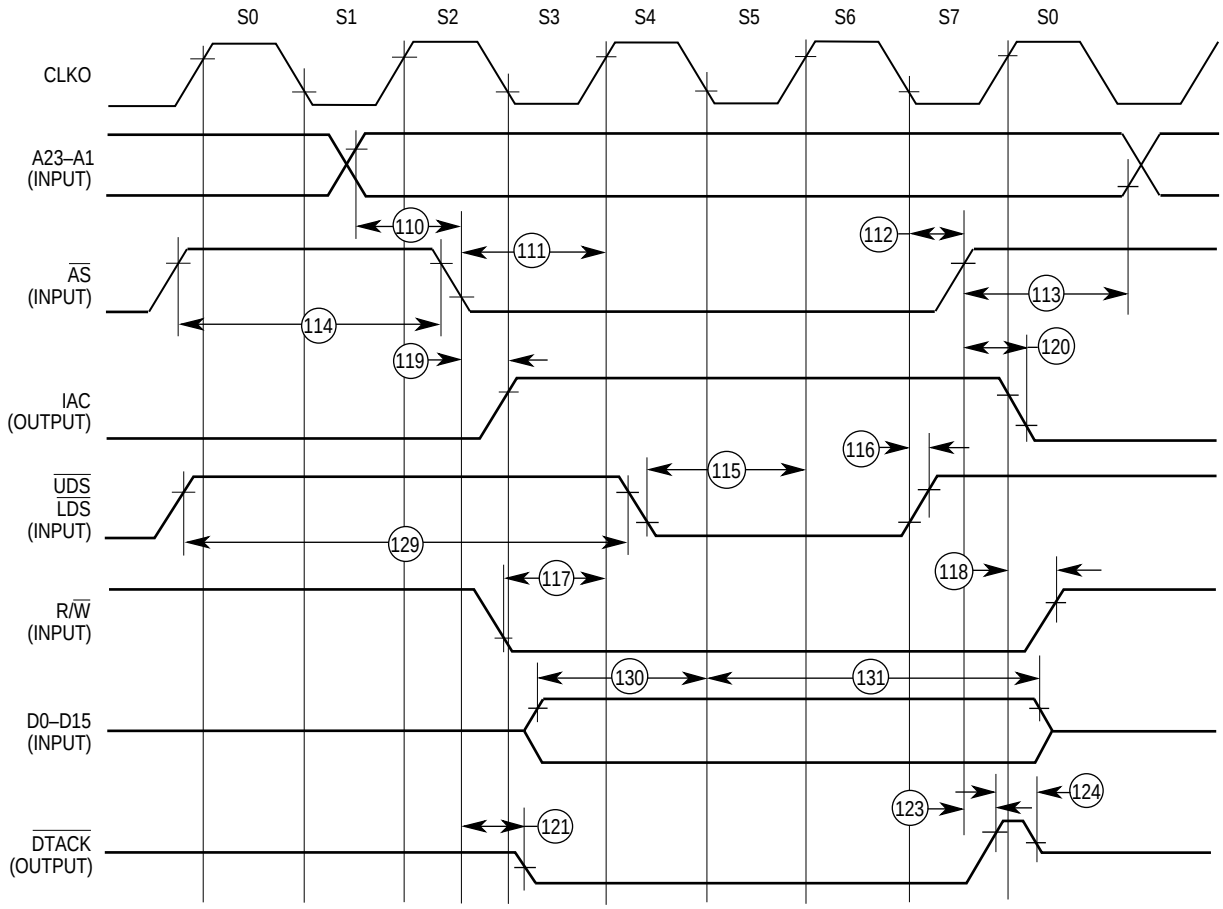


Figure 6-12. External Master Internal Synchronous Write Cycle Timing Diagram

6.12 AC ELECTRICAL SPECIFICATIONS—INTERNAL MASTER INTERNAL READ/WRITE CYCLES (see Figure 6-13)

			16.67 MHz		20 MHz		25 MHz		
Num.	**Characteristic**	**Symbol**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Unit**
140	Clock High to IAC High	t_{CHIAH}	—	40	—	35	—	27	ns
141	Clock Low to IAC Low	t_{CLIAL}	—	40	—	35	—	27	ns
142	Clock High to $\overline{\text{DTACK}}$ Low	t_{CHDTL}	—	45	—	40	—	30	ns
143	Clock Low to $\overline{\text{DTACK}}$ High	t_{CLDTH}	—	40	—	35	—	27	ns
144	Clock High to Data-Out Valid	t_{CHDOV}	—	30	—	25	—	20	ns
145	$\overline{\text{AS}}$ High to Data-Out Hold Time	t_{ASHDOH}	0	—	0	—	0	—	ns

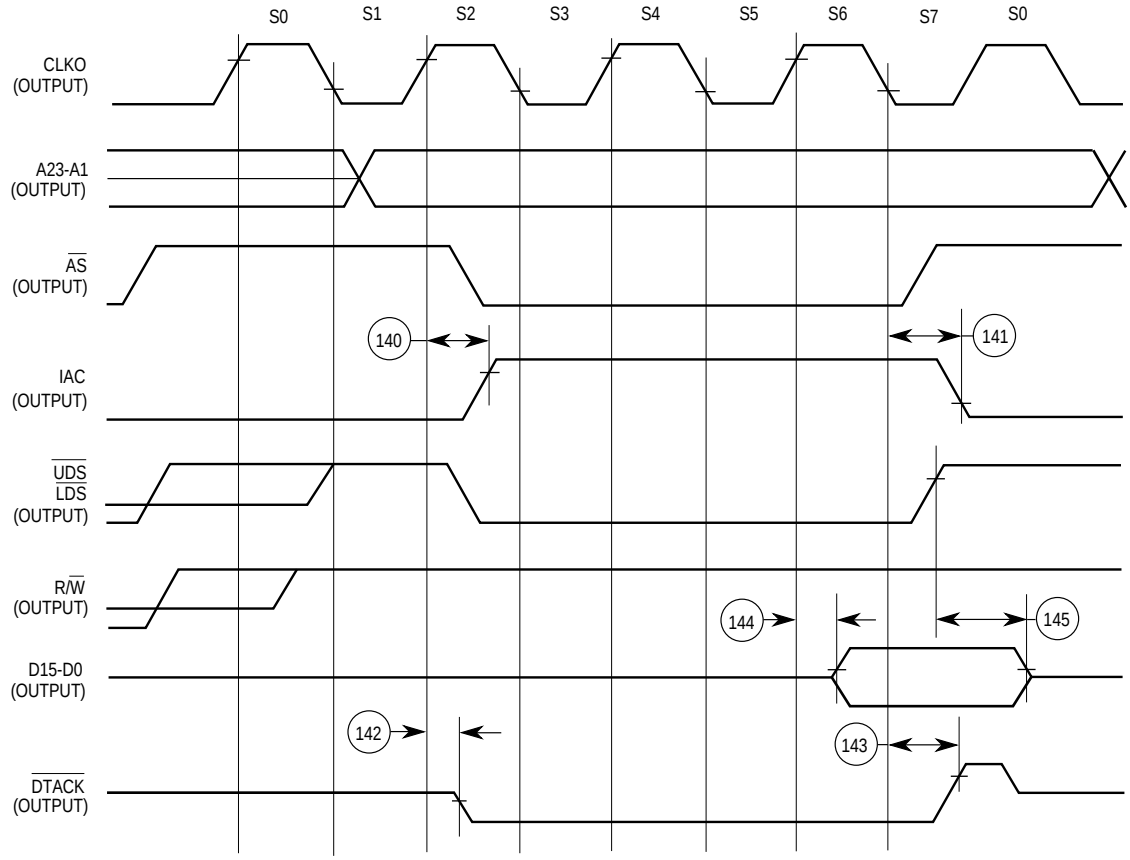


Figure 6-13. Internal Master Internal Read/Write Cycle Timing Diagram

6.13 AC ELECTRICAL SPECIFICATIONS—CHIP-SELECT TIMING INTERNAL MASTER (see Figure 6-14)

			16.67 MHz		20 MHz		25 MHz		
Num.	Characteristic	Symbol	Min	Max	Min	Max	Min	Max	Unit
150	Clock High to $\overline{CS}$, $\overline{IACK}$ Low (see Note 2)	$t_{CHCSIAKL}$	0	40	0	35	0	27	ns
151	Clock Low to $\overline{CS}$, $\overline{IACK}$ High (see Note 2)	$t_{CLCSIAKH}$	0	40	0	35	0	27	ns
152	$\overline{CS}$ Width Negated	t_{CSH}	60	—	50	—	40	—	ns
153	Clock High to $\overline{DTACK}$ Low (0 Wait State)	t_{CHDTKL}	—	45	—	40	—	30	ns
154	Clock Low to $\overline{DTACK}$ Low (1–6 Wait States)	t_{CLDTKL}	—	30	—	25	—	20	ns
155	Clock Low to $\overline{DTACK}$ High	t_{CLDTKH}	—	40	—	35	—	27	ns
156	Clock High to $\overline{BERR}$ Low (see Note 1)	t_{CHBERL}	—	40	—	35	—	27	ns
157	Clock Low to $\overline{BERR}$ High Impedance (see Note 1)	t_{CLBERH}	—	40	—	35	—	27	ns
158	$\overline{DTACK}$ High to $\overline{DTACK}$ High Impedance	$t_{DTKHDTKZ}$	—	15	—	15	—	10	ns
171	Input Data Hold Time from S6 Low	t_{IDHCL}	5	—	5	—	5	—	ns
172	$\overline{CS}$ Negated to Data-Out Invalid (Write)	t_{CSNDOI}	10	—	10	—	7	—	ns
173	Address, FC Valid to $\overline{CS}$ Asserted	t_{AFVCSA}	15	—	15	—	15	—	ns
174	$\overline{CS}$ Negated to Address, FC Invalid	t_{CSNAFI}	15	—	15	—	12	—	ns
175	$\overline{CS}$ Low Time (0 Wait States)	t_{CSLT}	120	—	100	—	80	—	ns
176	$\overline{CS}$ Negated to R/$\overline{W}$ Invalid	t_{CSNRWI}	10	—	10	—	7	—	ns
177	$\overline{CS}$ Asserted to R/$\overline{W}$ Low (Write)	t_{CSARWL}	—	10	—	10	—	8	ns
178	$\overline{CS}$ Negated to Data-In Invalid (Hold Time on Read)	t_{CSNDII}	0	—	0	—	0	—	ns

NOTE:

1. This specification is valid only when the ADCE or WPVE bits in the SCR are set.
2. For loading capacitance less than or equal to 50 pF, subtract 4 ns from the maximum value given.
3. Since $\overline{AS}$ and $\overline{CS}$ are asserted/negated on the same CLKO edges, no $\overline{AS}$ to $\overline{CS}$ relative timings can be specified. However, $\overline{CS}$ timings are given relative to a number of other signals, in the same manner as $\overline{AS}$. See Figure 6-2 and Figure 6-3 for diagrams.

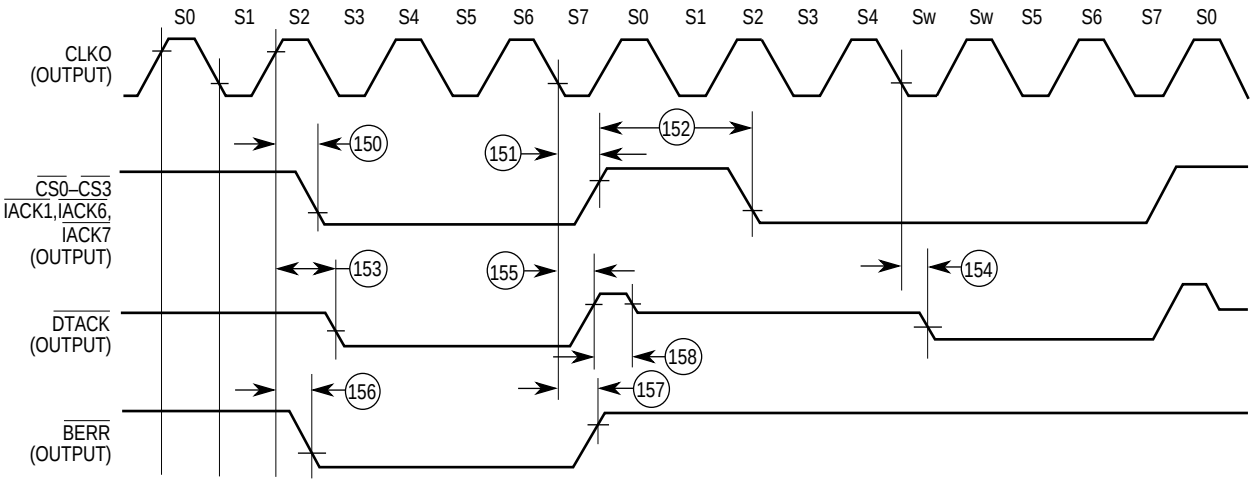


Figure 6-14. Internal Master Chip-Select Timing Diagram

6.14 AC ELECTRICAL SPECIFICATIONS—CHIP-SELECT TIMING EXTERNAL MASTER (see Figure 6-15)

			16.67 MHz		20 MHz		25 MHz		
Num.	**Characteristic**	**Symbol**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Unit**
154	Clock Low to $\overline{DTACK}$ Low (1-6 Wait States)	t_{CLDTKL}	—	30	—	25	—	20	ns
160	$\overline{AS}$ Low to $\overline{CS}$ Low	t_{ASLCSL}	—	30	—	25	—	20	ns
161	$\overline{AS}$ High to $\overline{CS}$ High	t_{ASHCSH}	—	30	—	25	—	20	ns
162	Address Valid to $\overline{AS}$ Low	t_{AVASL}	15	—	12	—	10	—	ns
163	R/$\overline{W}$ Valid to $\overline{AS}$ Low (see Note 1)	t_{RWVASL}	15	—	12	—	10	—	ns
164	$\overline{AS}$ Negated to Address Hold Time	t_{ASHAI}	0	—	0	—	0	—	ns
165	$\overline{AS}$ Low to $\overline{DTACK}$ Low (0 Wait State)	$t_{ASLDTKL}$	—	45	—	40	—	30	ns
167	$\overline{AS}$ High to $\overline{DTACK}$ High	$t_{ASHDTKH}$	—	30	—	25	—	20	ns
168	$\overline{AS}$ Low to $\overline{BERR}$ Low (see Note 2)	$t_{ASLBERL}$	—	30	—	25	—	20	ns
169	$\overline{AS}$ High to $\overline{BERR}$ High Impedance (see Notes 2 and 3)	$t_{ASHBERH}$	—	30	—	25	—	20	ns

NOTES:
1. The minimum value must be met to guarantee write protection operation.
2. This specification is valid when the ADCE or WPVE bits in the SCR are set.
3. Also applies after a timeout of the hardware watchdog.

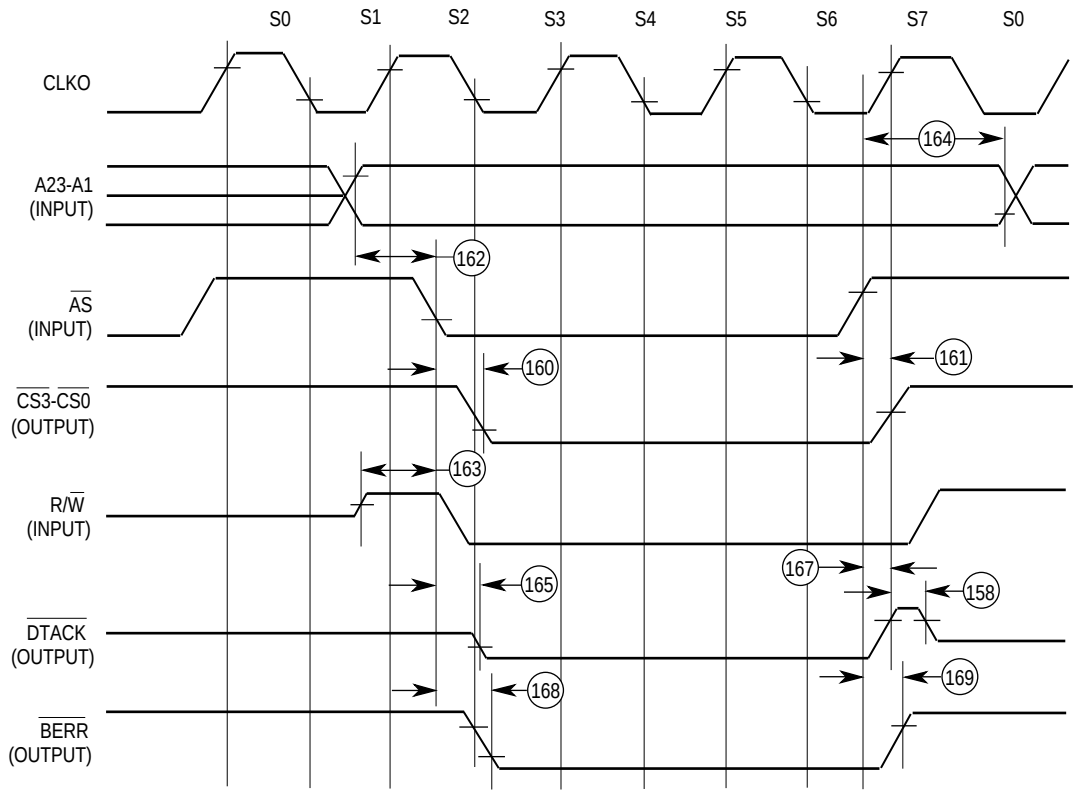


Figure 6-15. External Master Chip-Select Timing Diagram

6.15 AC ELECTRICAL SPECIFICATIONS—PARALLEL I/O

(see Figure 6-16**)**

			16.67 MHz		20 MHz		25 MHz		
Num.	**Characteristic**	**Symbol**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Unit**
180	Input Data Setup Time (to Clock Low)	t_{DSU}	20	—	20	—	14	—	ns
181	Input Data Hold Time (from Clock Low)	t_{DH}	10	—	10	—	19	—	ns
182	Clock High to Data-out Valid (CPU Writes Data, Control, or Direction)	t_{CHDOV}	—	35	—	30	—	24	ns

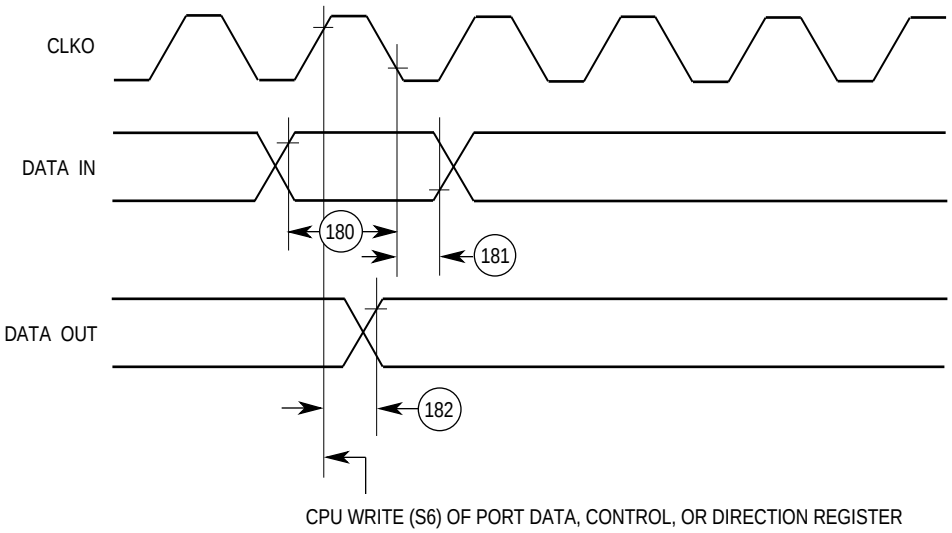


Figure 6-16. Parallel I/O Data-In/Data-Out Timing Diagram

6.16 AC ELECTRICAL SPECIFICATIONS—INTERRUPTS

(seeFigure 6-17**)**

			16.67 MHz		20 MHz		25 MHz		
Num.	**Characteristic**	**Symbol**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Unit**
190	Interrupt Pulse Width Low $\overline{IRQ}$ (Edge Triggered Mode) or PB8-11	t_{IPW}	50	—	42	—	34	—	ns
191	Minimum Time Between Active Edges	t_{AEMT}	3	—	3	—	3	—	clk

NOTE: Setup time for the asynchronous inputs $\overline{IPL2}$–$\overline{IPL0}$ and $\overline{AVEC}$ guarantees their recognition at the next falling edge of the clock.

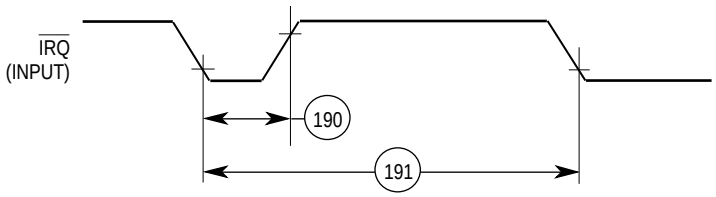


Figure 6-17. Interrupts Timing Diagram

6.17 AC ELECTRICAL SPECIFICATIONS—TIMERS

(see Figure 6-18**)**

			16.67 MHz		20 MHz		25 MHz		
Num.	**Characteristic**	**Symbol**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Unit**
200	Timer Input Capture Pulse Width	t_{TPW}	50	—	42	—	34	—	ns
201	TIN Clock Low Pulse Width	t_{TICLT}	50	—	42	—	34	—	ns
202	TIN Clock High Pulse Width and Input Capture High Pulse Width	t_{TICHT}	2	—	2	—	2	—	clk
203	TIN Clock Cycle Time	t_{cyc}	3	—	3	—	3	—	clk
204	Clock High to TOUT Valid	t_{CHTOV}	—	35	—	30	—	24	ns
205	$\overline{FRZ}$ Input Setup Time (to Clock High) (see Note 1)	t_{FRZSU}	20	—	20	—	14	—	ns
206	$\overline{FRZ}$ Input Hold Time (from Clock High)	t_{FRZHT}	10	—	10	—	7	—	ns

NOTES:
1. $\overline{FRZ}$ should be negated during total system reset.
2. The TIN specs above do not apply to the use of TIN1 as a baud rate generator input clock. In such a case, specifications 1–3 may be used.

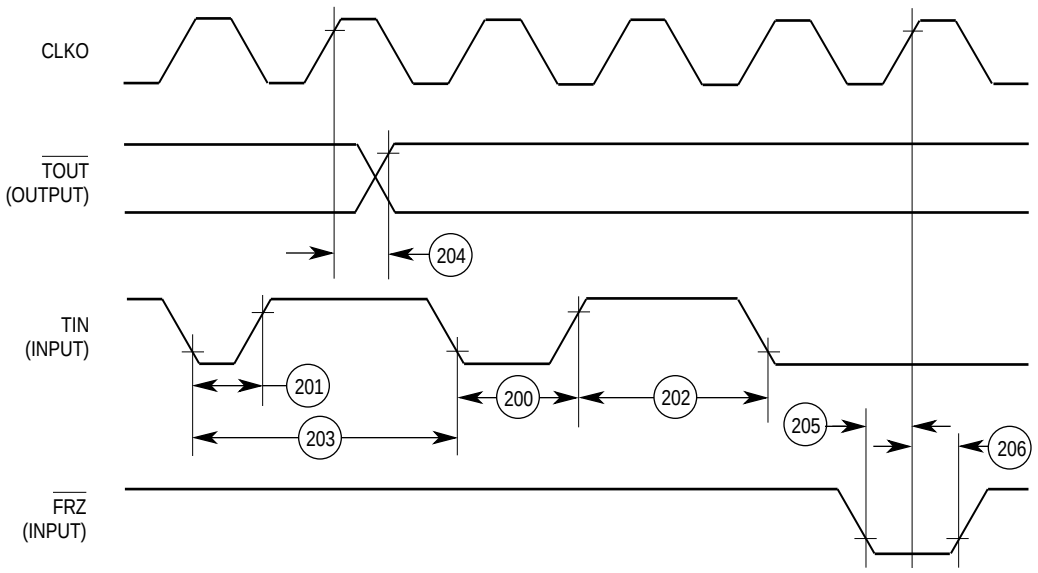


Figure 6-18. Timers Timing Diagram

6.18 AC ELECTRICAL SPECIFICATIONS—SERIAL COMMUNICATIONS PORT (see Figure 6-19).

		16.67 MHz		20 MHz		25 MHz		
Num.	**Characteristic**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Unit**
250	SPCLK Clock Output Period	4	64	4	64	4	64	clks
251	SPCLK Clock Output Rise/Fall Time	0	15	0	10	0	8	ns
252	Delay from SPCLK to Transmit (see Note 1)	0	40	0	30	0	24	ns
253	SCP Receive Setup Time (see Note 1)	40	—	30	—	24	—	ns
254	SCP Receive Hold Time (see Note 1)	10	—	8	—	7	—	ns

NOTES:
1. This also applies when SPCLK is inverted by CI in the SPMODE register.
2. The enable signals for the slaves may be implemented by the parallel I/O pins.

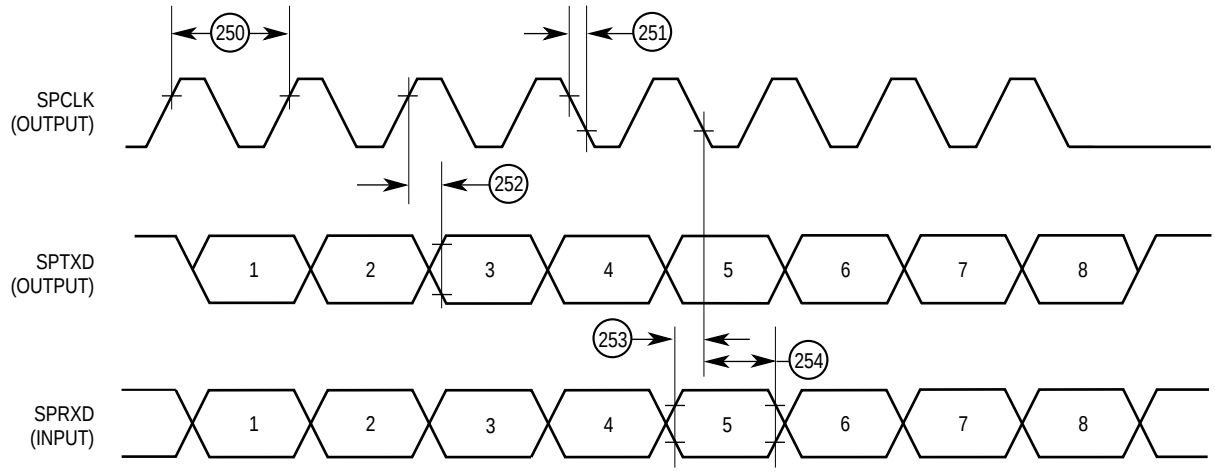


Figure 6-19. Serial Communication Port Timing Diagram

6.19 AC ELECTRICAL SPECIFICATIONS—IDL TIMING (All timing measurements, unless otherwise specified, are referenced to the L1CLK at 50% point of V_{DD}) (see Figure 6-20).

		16.67 MHz		20 MHz		25 MHz		
Num.	**Characteristic**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Unit**
260	L1CLK (IDL Clock) Frequency (see Note 1)	—	6.66	—	8	—	10	MHz
261	L1CLK Width Low	55	—	45	—	37	—	ns
262	L1CLK Width High (see Note 3)	P+10	—	P+10	—	P+10	—	ns
263	L1TXD, L1RQ, SDS1–SDS2 Rising/Falling Time	—	20	—	17	—	14	ns
264	L1SY1 (sync) Setup Time (to L1CLK Falling Edge)	30	—	25	—	20	—	ns
265	L1SY1 (sync) Hold Time (from L1CLK Falling Edge)	50	—	40	—	34	—	ns
266	L1SY1 (sync) Inactive Before 4th L1CLK	0	—	0	—	0	—	ns
267	L1TxD Active Delay (from L1CLK Rising Edge)	0	75	0	65	0	50	ns
268	L1TxD to High Impedance (from L1CLK Rising Edge) (see Note 2)	0	50	0	42	0	34	ns
269	L1RxD Setup Time (to L1CLK Falling Edge)	50	—	42	—	34	—	ns
270	L1RxD Hold Time (from L1CLK Falling Edge)	50	—	42	—	34	—	ns
271	Time Between Successive IDL syncs	20	—	20	—	20	—	L1CLK
272	L1RQ Valid before Falling Edge of L1SY1	1	—	1	—	1	—	L1CLK
273	L1GR Setup Time (to L1SY1 Falling Edge)	50	—	42	—	34	—	ns
274	L1GR Hold Time (from L1SY1 Falling Edge)	50	—	42	—	34	—	ns
275	SDS1–SDS2 Active Delay from L1CLK Rising Edge	10	75	10	65	7	50	ns
276	SDS1–SDS2 Inactive Delay from L1CLK Falling Edge	10	75	10	65	7	50	ns

NOTES:

1. The ratio CLKO/L1CLK must be greater than 2.5/1.
2. High impedance is measured at the 30% and 70% of V_{DD} points, with the line at $V_{DD}/2$ through 10K in parallel with130 pF.
3. Where P = 1/CLKO. Thus, for a 16.67-MHz CLKO rate, P = 60 ns.

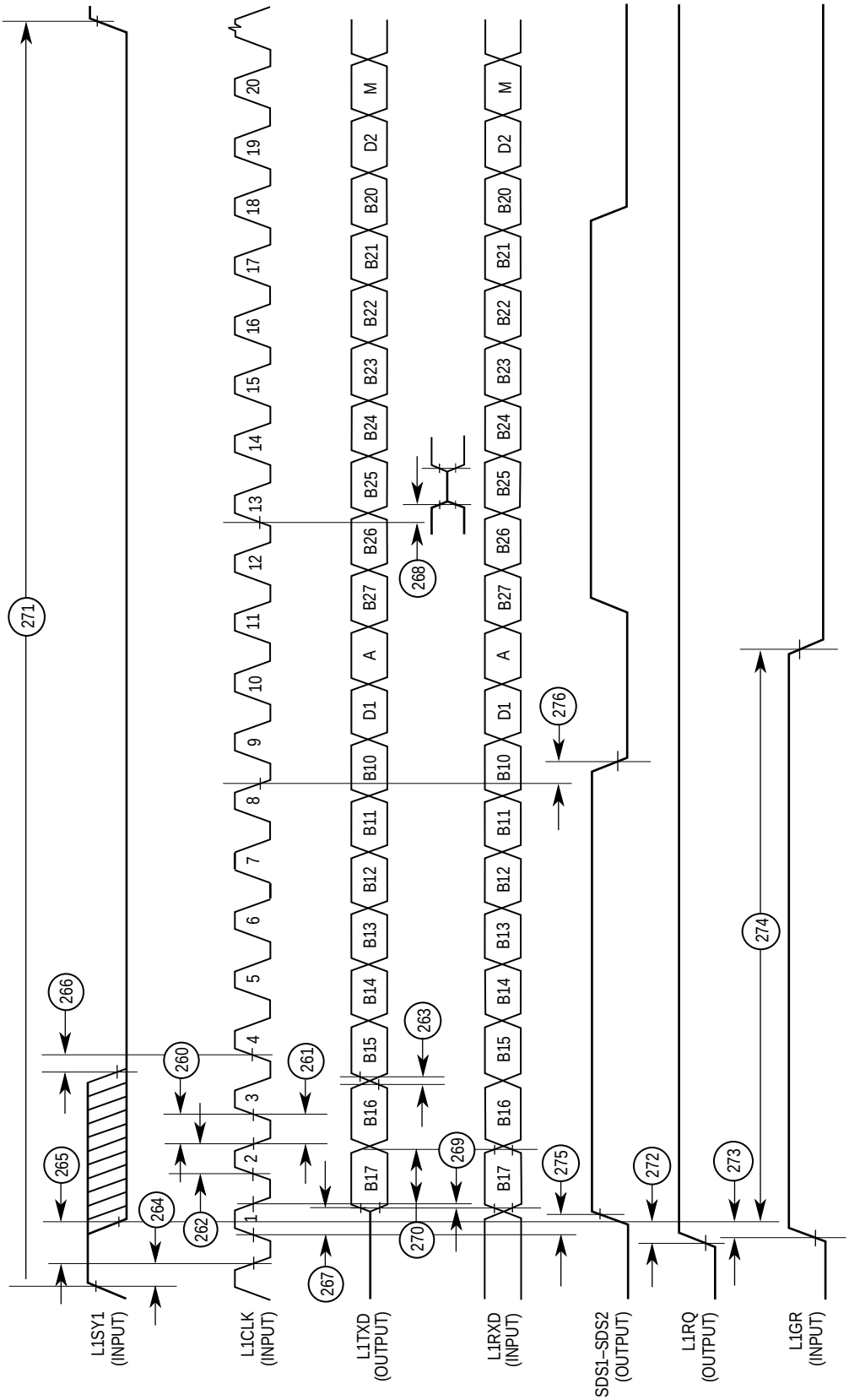


Figure 6-20. IDL Timing Diagram

6.20 AC ELECTRICAL SPECIFICATIONS—GCI TIMING

GCI supports the NORMAL mode and the GCI channel 0 (GCN0) in MUX mode. Normal mode uses 512 kHz clock rate (256K bit rate). MUX mode uses 256 x n - 3088 kbs (clock rate is data rate x 2). The ratio CLKO/L1CLK must be greater than 2.5/1 (see Figure 6-21).

		16.67 MHz		20 MHz		25 MHz		
Num.	**Characteristic**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Unit**
	L1CLK GCI Clock Frequency (Normal Mode) (see Note 1)	—	512	—	512	—	512	kHz
280	L1CLK Clock Period Normal Mode (see Note 1)	1800	2100	1800	2100	1800	2100	ns
281	L1CLK Width Low/High Normal Mode	840	1450	840	1450	840	1450	ns
282	L1CLK Rise/Fall Time Normal Mode (see Note 4)	—	—	—	—	—	—	ns
	L1CLK (GCI Clock) Frequency (MUX Mode) (see Note 1)	—	6.668	—	6.668	—	6.668	MHz
280	L1CLK Clock Period MUX Mode (see Note 1)	150	—	150	—	150	—	ns
281	L1CLK Width Low MUX Mode	55	—	55	—	55	—	ns
281A	L1CLK Width High MUX Mode (see Note 5)	P+10	—	P+10	—	P+10	—	ns
282	L1CLK Rise/Fall Time MUX Mode (see Note 4)	—	—	—	—	—	—	ns
283	L1SY1 Sync Setup Time to L1CLK Falling Edge	30	—	25	—	20	—	ns
284	L1SY1 Sync Hold Time from L1CLK Falling Edge	50	—	42	—	34	—	ns
285	L1TxD Active Delay (from L1CLK Rising Edge) (see Note 2)	0	100	0	85	0	70	ns
286	L1TxD Active Delay (from L1SY1 Rising Edge) (see Note 2)	0	100	0	85	0	70	ns
287	L1RxD Setup Time to L1CLK Rising Edge	20	—	17	—	14	—	ns
288	L1RxD Hold Time from L1CLK Rising Edge	50	—	42	—	34	—	ns
289	Time Between Successive L1SY1in Normal SCIT Mode	64 192	— —	64 192	— —	64 192	— —	L1CLK L1CLK
290	SDS1–SDS2 Active Delay from L1CLK Rising Edge (see Note 3)	10	90	10	75	7	60	ns
291	SDS1–SDS2 Active Delay from L1SY1 Rising Edge (see Note 3)	10	90	10	75	7	60	ns
292	SDS1–SDS2 Inactive Delay from L1CLK Falling Edge	10	90	10	75	7	60	ns
293	GCIDCL (GCI Data Clock) Active Delay	0	50	0	42	0	34	ns

NOTES:

1.The ratio CLKO/L1CLK must be greater than 2.5/1.

2. Condition C_L = 150 pF. L1TD becomes valid after the L1CLK rising edge or L1SY1, whichever is later.

3.SDS1–SDS2 become valid after the L1CLK rising edge or L1SY1, whichever is later.

4.Schmitt trigger used on input buffer.

5.Where P = 1/CLKO. Thus, for a 16.67-MHz CLKO rate, P = 60 ns.

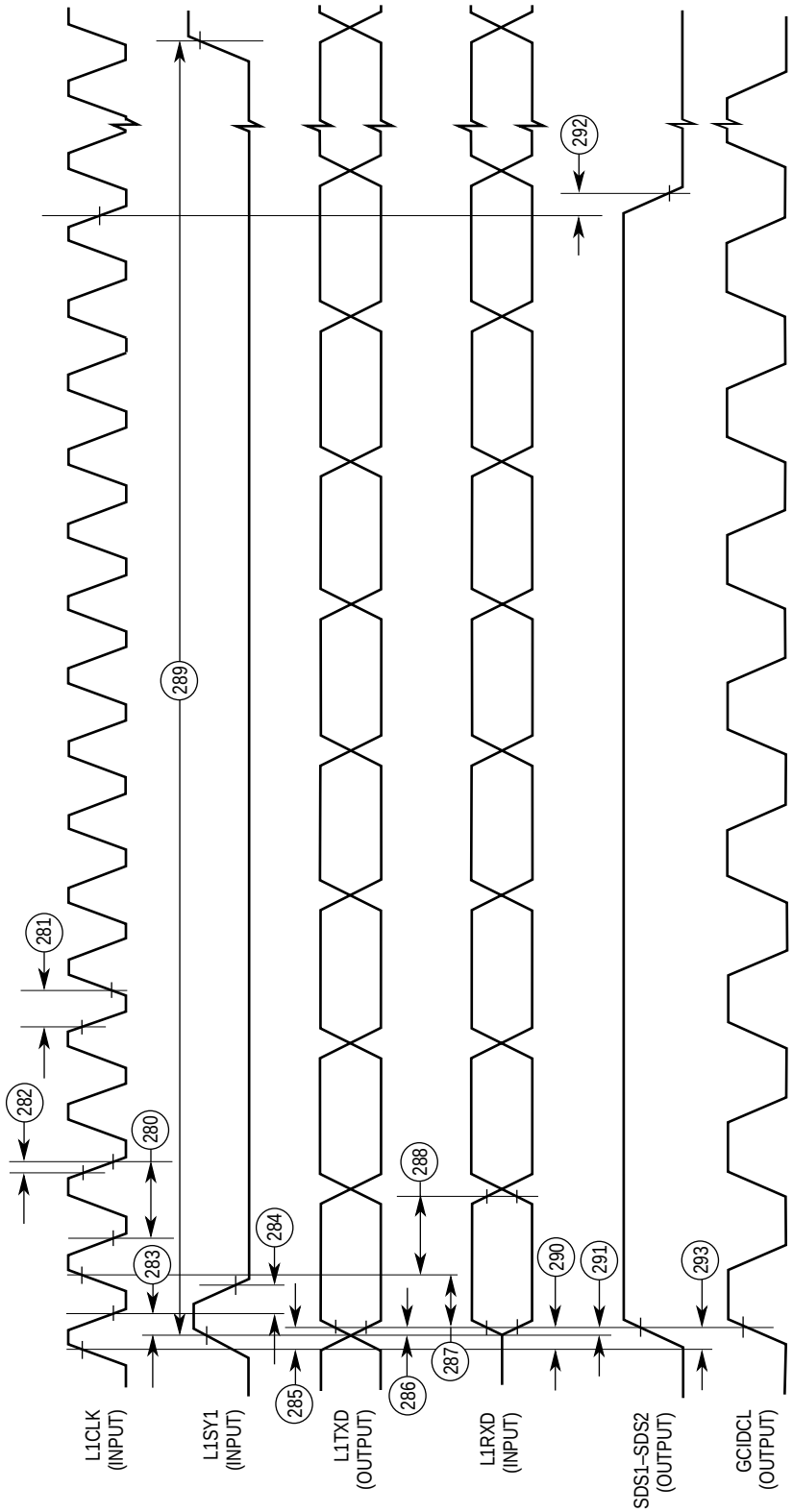


Figure 6-21. GCI Timing Diagram

6.21 AC ELECTRICAL SPECIFICATIONS—PCM TIMING

There are two sync types:

Short Frame—Sync signals are one clock cycle prior to the data
Long Frame—Sync signals are N-bits that envelope the data, N > 0; see Figure 6-22 and Figure 6-23).

		16.67 MHz		20 MHz		25 MHz		
Num.	**Characteristic**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Unit**
300	L1CLK (PCM Clock) Frequency (see Note 1)	—	6.66	—	8.0	—	10.0	MHz
301	L1CLK Width Low	55	—	45	—	37	—	ns
301A	L1CLK Width High (see Note 4)	P+10	—	P+10	—	P+10	—	ns
302	L1SY0–L1SY1 Setup Time to L1CLK Rising Edge	0	—	0	—	0	—	ns
303	L1SY0–L1SY1 Hold Time from L1CLK Falling Edge	40	—	33	—	27	—	ns
304	L1SY0–L1SY1 Width Low	1	—	1	—	1	—	L1CLK
305	Time Between Successive Sync Signals (Short Frame)	8	—	8	—	8	—	L1CLK
306	L1TxD Data Valid after L1CLK Rising Edge (see Note 2)	0	70	0	60	0	47	ns
307	L1TxD to High Impedance (from L1CLK Rising Edge)	0	50	0	42	0	34	ns
308	L1RxD Setup Time (to L1CLK Falling Edge) (see Note 3)	20	—	17	—	14	—	ns
309	L1RxD Hold Time (from L1CLK Falling Edge) (see Note 3)	50	—	42	—	34	—	ns

NOTES:

1. The ratio CLK/L1CLK must be greater than 2.5/1.
2. L1TxD becomes valid after the L1CLK rising edge or the sync enable, whichever is later, if long frames are used. This note should only be used if the user can guarantee that only one sync pin (L1SY0 and L1SY1) is changed simultaneously in the selection and de-selection of the desired PCM channel time slot. A safe example of this is using only PCM CH-1. Another example is using CH-1 and CH-2 only, where CH-1 and CH-2 are not contiguous on the PCM highway.
3. Specification valid for both sync methods.
4. Where P = 1/CLKO. Thus, for a 16.67-MHz CLKO rate, P = 60 ns.

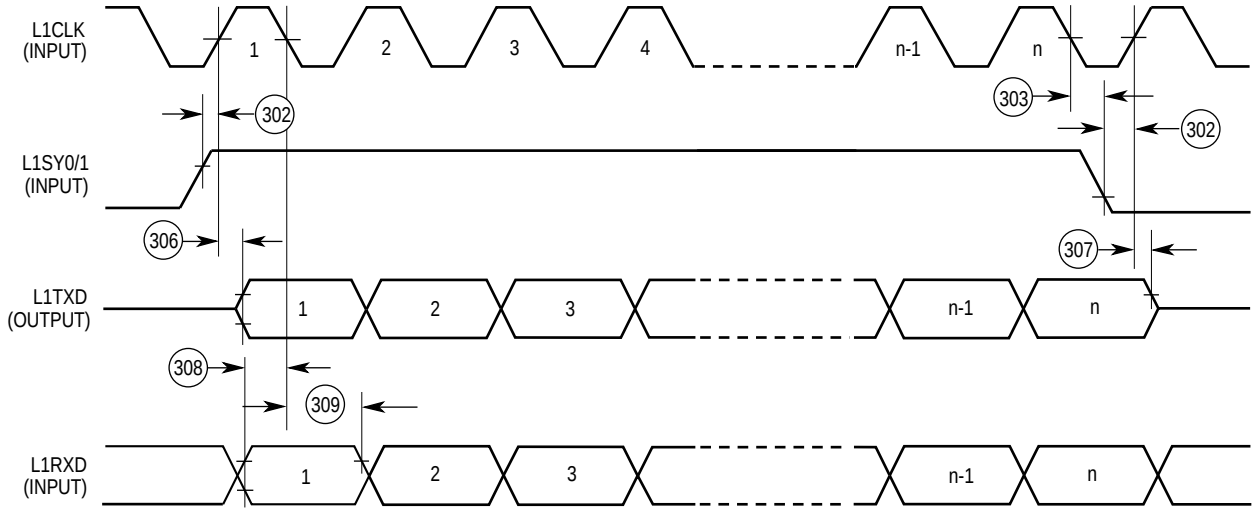
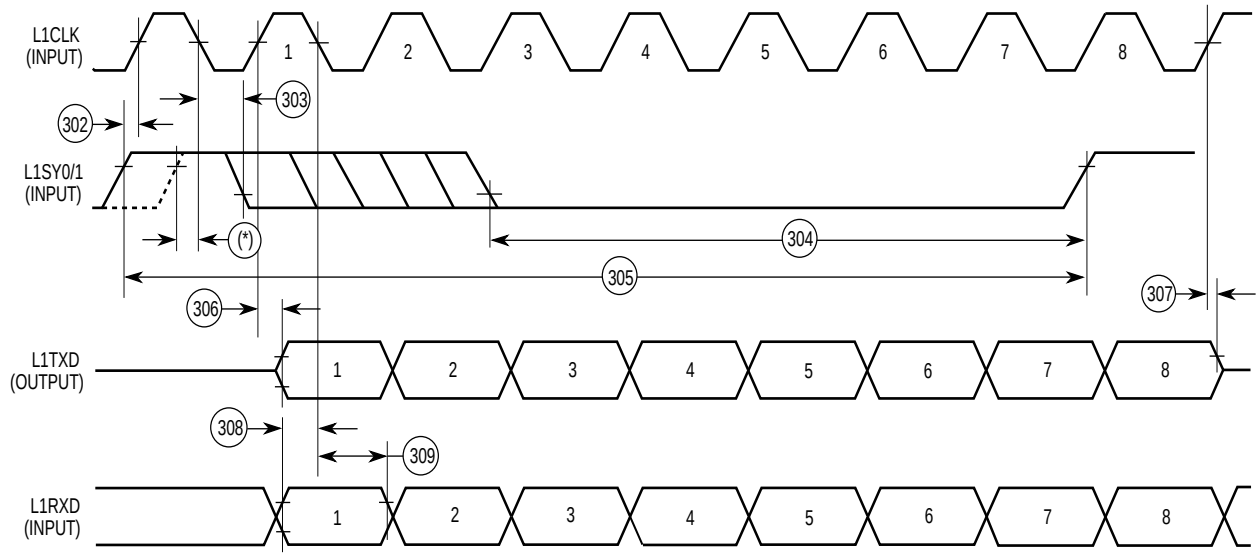


Figure 6-22. PCM Timing Diagram (SYNC Envelopes Data)



NOTE: (*) If L1SYn is guaranteed to make a smooth low to high transition (no spikes) while the clock is high, setup time can be defined as shown (min 20 ns).

Figure 6-23. PCM Timing Diagram (SYNC Prior to 8-Bit Data)

6.22 AC ELECTRICAL SPECIFICATIONS—NMSI TIMING

The NMSI mode uses two clocks, one for receive and one for transmit. Both clocks can be internal or external. When t he clock is internal, it is generated by the internal baud rate generator and it is output on TCLK or RCLK. All the timing is related to the external clock pin. The timing is specified for NMSI1. It is also valid for NMSI2 and NMSI3 (see Figure 6-24).

		16.67 MHz		16.67 MHz		20 MHz		20 MHz		25 MHz		25 MHz		
Num.	**Characteristic**	**Internal Clock**		**External Clock**		**Internal Clock**		**External Clock**		**Internal Clock**		**External Clock**		**Unit**
		Min	**Max**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	
315	RCLK1 and TCLK1 Frequency (see Note 1)	—	5.55	—	6.668	—	6.66	—	8	—	8.33	—	10	MHz
316	RCLK1 and TCLK1 Low (see Note 4)	65	—	P+10	—	55	—	P+10	—	45	—	P+10	—	ns
316a	RCLK1 and TCLK1 High	65	—	55	—	55	—	45	—	45	—	35	—	ns
317	RCLK1 and TCLK1 Rise/Fall Time (see Note 3)	—	20	—	—	—	17	—	—	—	14	—	—	ns
318	TXD1 Active Delay from TCLK1 Falling Edge	0	40	0	70	0	30	0	50	0	25	0	40	ns
319	$\overline{\text{RTS1}}$ Active/Inactive Delay from TCLK1 Falling Edge	0	40	0	100	0	30	0	80	0	25	0	65	ns
320	$\overline{\text{CTS1}}$ Setup Time to TCLK1 Rising Edge	50	—	10	—	40	—	7	—	35	—	7	—	ns
321	RXD1 Setup Time to RCLK1 Rising Edge	50	—	10	—	40	—	7	—	35	—	7	—	ns
322	RXD1 Hold Time from RCLK1 Rising Edge (see Note 2)	10	—	50	—	7	—	40	—	7	—	35	—	ns
323	$\overline{\text{CD1}}$ Setup Time to RCLK1 Rising Edge	50	—	10	—	40	—	7	—	35	—	7	—	ns

NOTES:

1. The ratio CLKO/TCLK1 and CLKO/RCLK1 must be greater than or equal to 2.5/1 for external clock. The input clock to the baud rate generator may be either an internal clock or TIN1, and may be as fast as EXTAL. However, the output of the baud rate generator must provide a CLKO/TCLK1 and CLKO/RCLK1 ratio greater than or equal to 3/1. In asynchronous mode (UART), the bit rate is 1/16 of the TCLK1/RCLK1 clock rate.
2. Also applies to $\overline{\text{CD}}$ hold time when $\overline{\text{CD}}$ is used as an external sync in BISYNC or totally transparent mode.
3. Schmitt triggers used on input buffers.
4. Where P = 1/CLKO. Thus, for a 16.67-MHz CLKO rate, P = 60 ns.

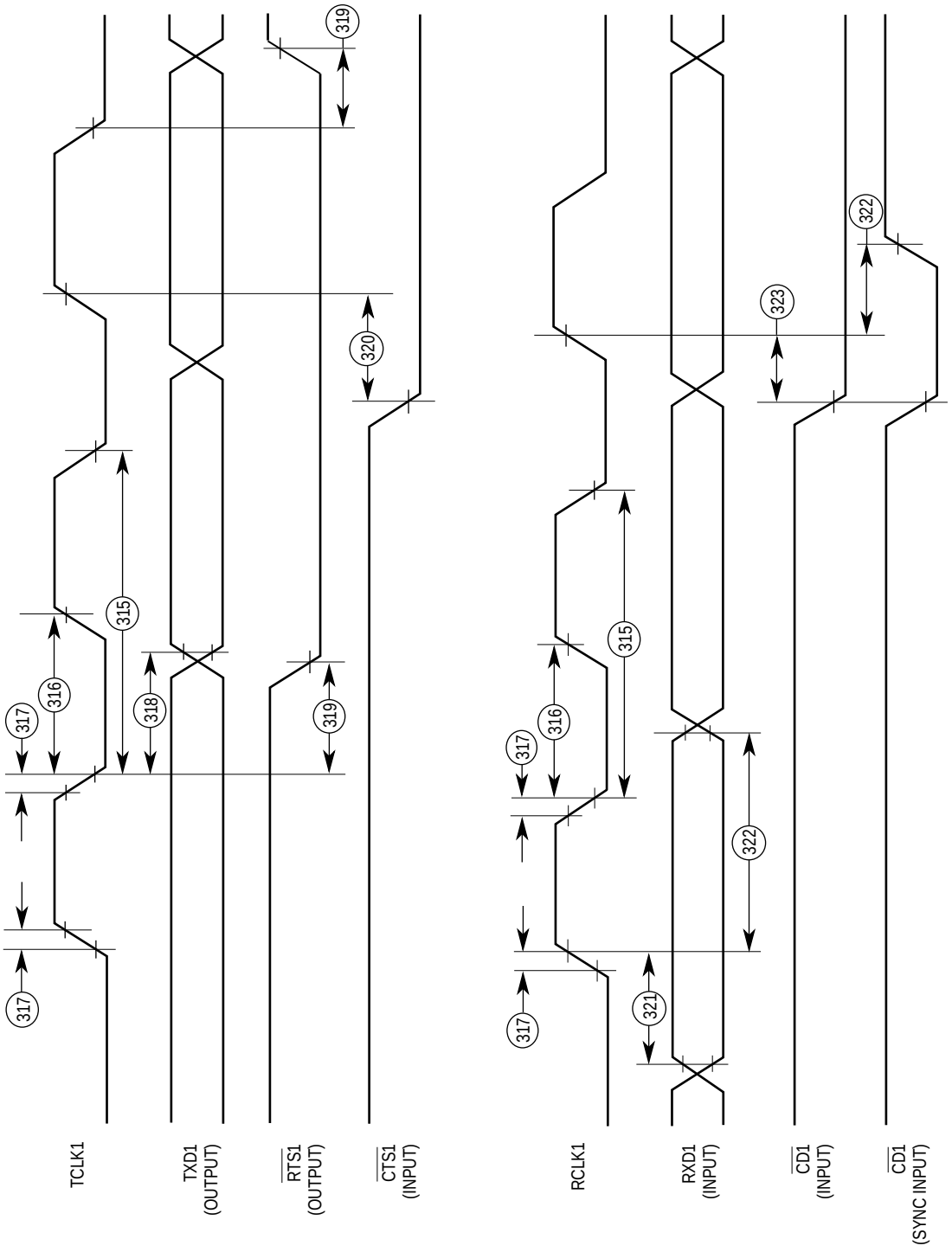
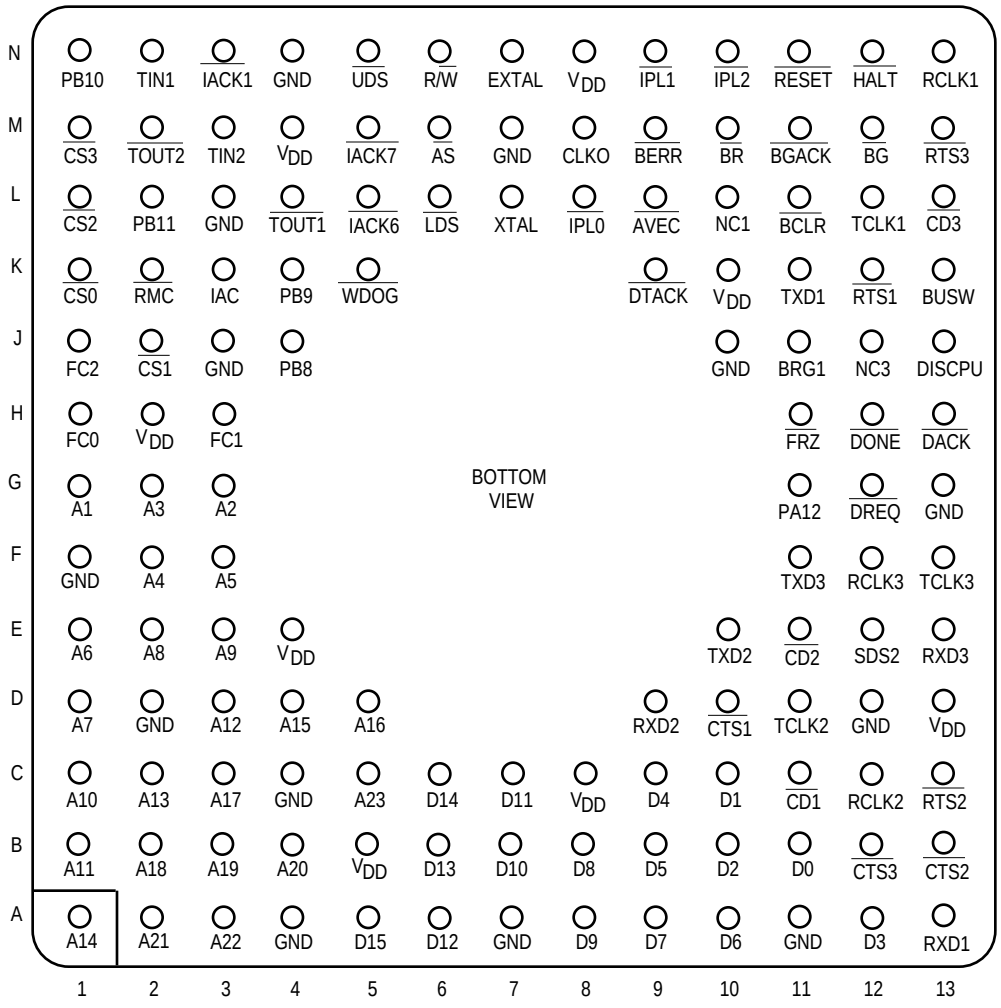


Figure 6-24. NMSI Timing Diagram

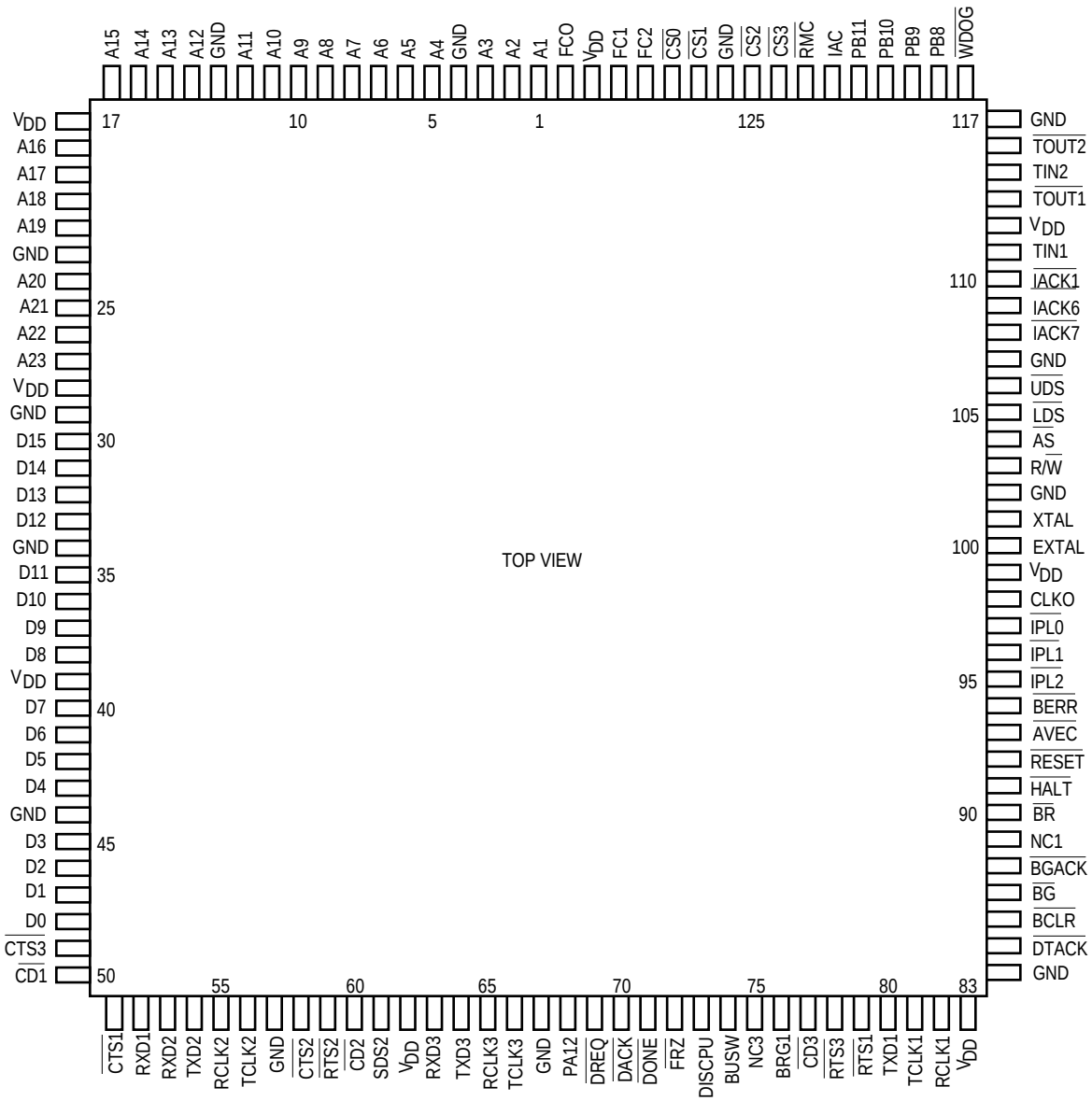
SECTION 7
MECHANICAL DATA AND ORDERING INFORMATION

7.1 PIN ASSIGNMENTS

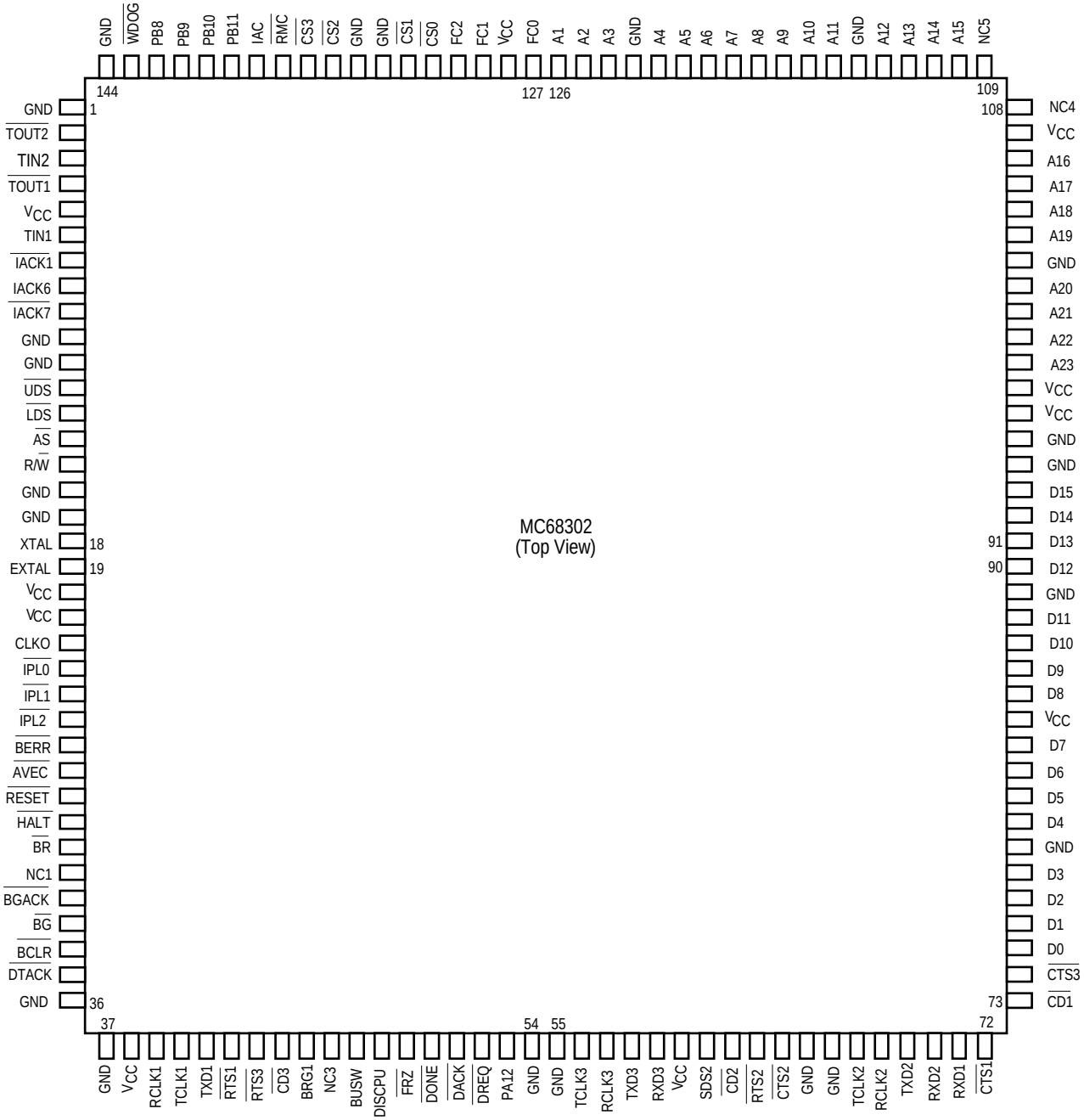
7.1.1 Pin Grid Array (PGA)



7.1.2 Plastic Surface Mount (PQFP)

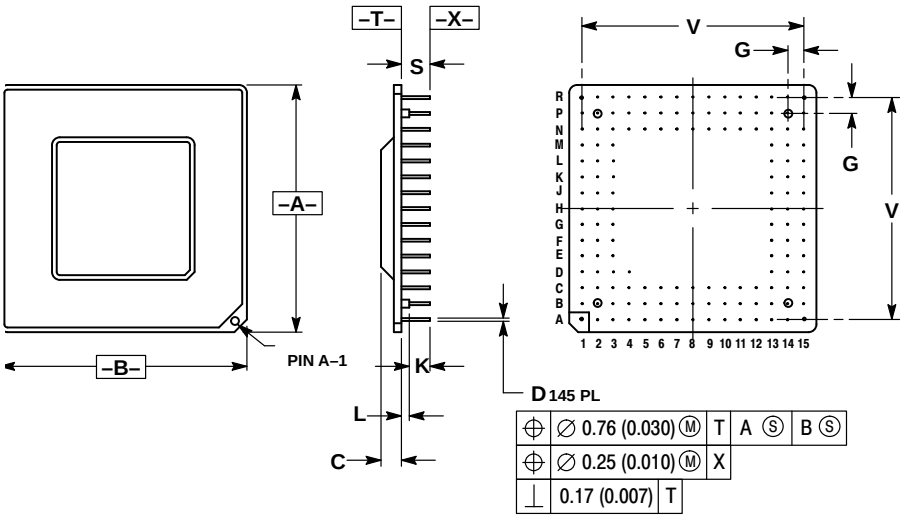


7.1.3 Thin Surface Mount (TQFP)



7.2 PACKAGE DIMENSIONS

7.2.1 Pin Grid Array (PGA)



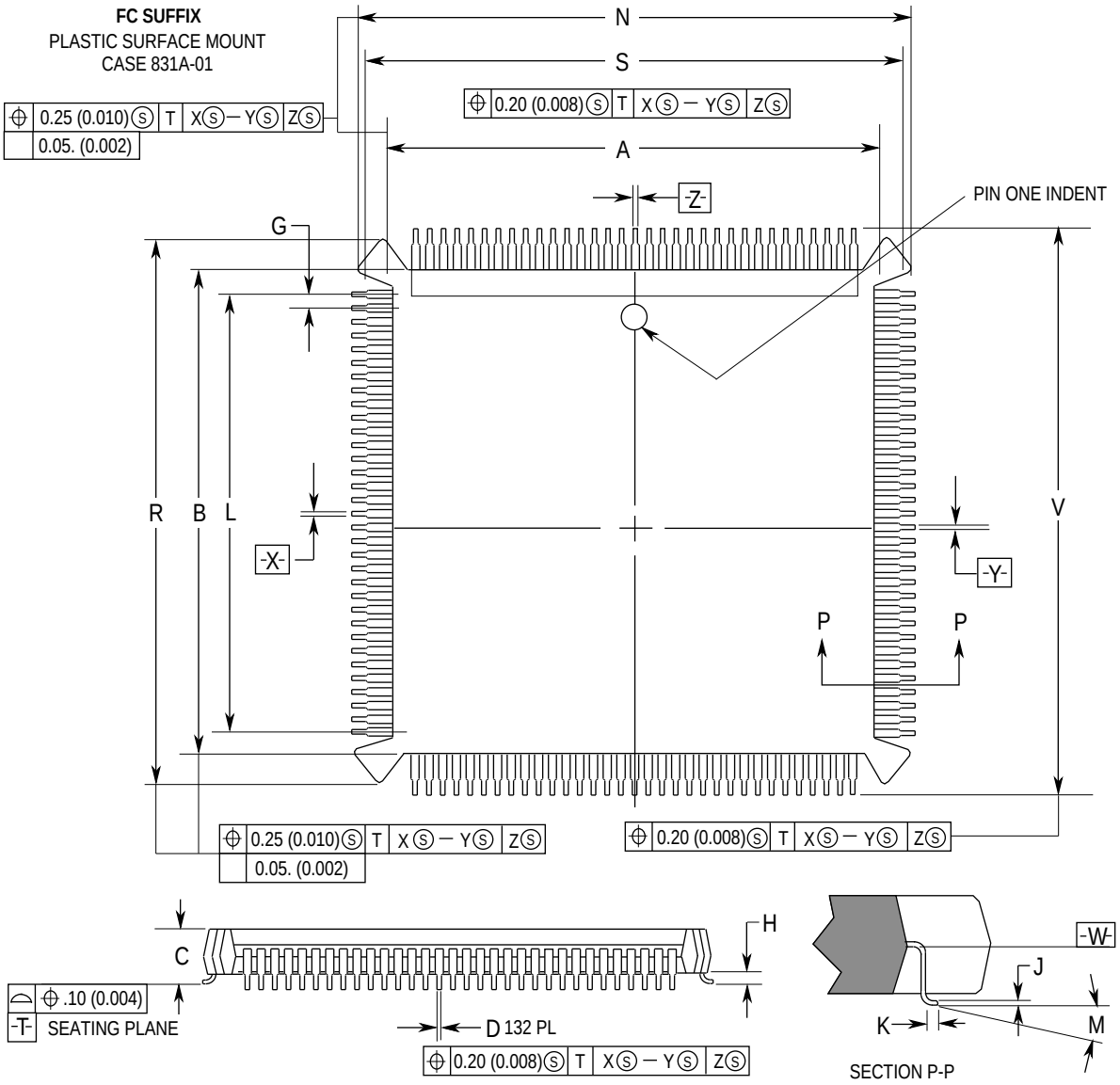
NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: INCH.
3. DIMENSION D INCLUDES LEAD FINISH.

	INCHES		MILLIMETERS	
DIM	MIN	MAX	MIN	MAX
A	1.550	1.570	39.37	39.88
B	1.550	1.570	39.37	39.88
C	0.115	0.135	2.92	3.43
D	0.017	0.022	0.43	0.55
G	0.100 BSC		2.54 BSC	
K	0.120	0.140	3.05	3.55
L	0.040	0.060	1.02	1.52
S	0.170	0.195	4.32	4.95
V	1.400 BSC		35.56 BSC	

CASE 768E-01
ISSUE O

DATE 04/04/94

7.2.2 Plastic Surface Mount (PQFP)

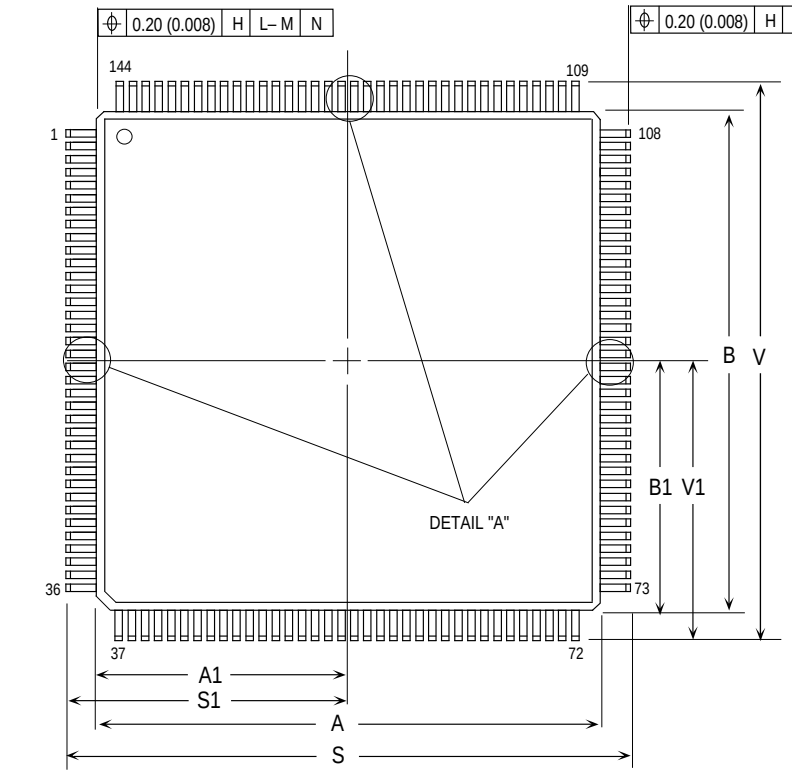


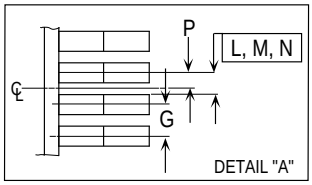
	MILLIMETERS		INCHES	
DIM	MIN	MAX	MIN	MAX
A	24.06	24.20	0.947	0.953
B	24.06	24.20	0.947	0.953
C	4.07	4.57	0.160	0.180
D	0.21	0.30	0.008	0.012
G	0.64 BSC		0.025 BSC	
H	0.51	1.01	0.020	0.040
J	0.16	0.20	0.006	0.008
K	0.51	0.76	0.020	0.030
M	0°	8°	0°	8°
N	27.88	28.01	1.097	1.103
R	27.88	28.01	1.097	1.103
S	27.31	27.55	1.075	1.085
V	27.31	27.55	1.075	1.085

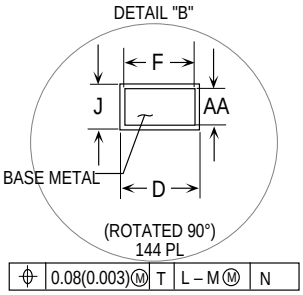
NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: INCHES
3. DIM A, B, N, AND R DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE MOLD PROTRUSION FOR DIMENSIONS A AND B IS 0.25 (0.010), FOR DIMENSIONS N AND R IS 0.18 (0.007).
4. DATUM PLANE -W- IS LOCATED AT THE UNDERSIDE 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. DIM S AND V TO BE DETERMINED AT SEATING PLANE, DATUM -T-.
7. DIM A, B, N AND R TO BE DETERMINED AT DATUM PLANE -W-.

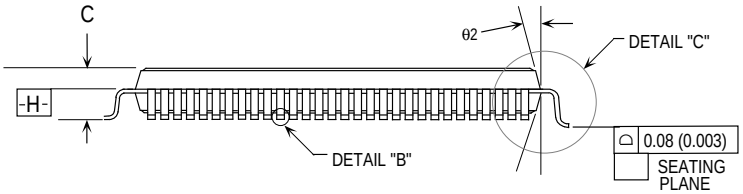
7.2.3 Thin Surface Mount (TQFP)

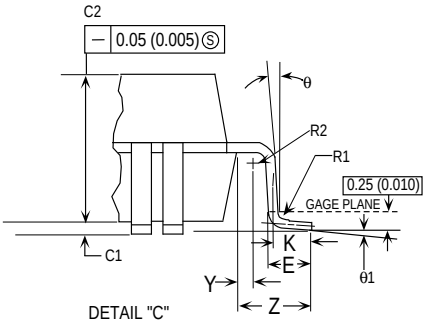
CASE 918-02
144 TQFP











NOTES:

1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: MILLIMETER.
3. DATUM PLANE **-H-** IS LOCATED AT BOTTOM OF LEAD AND COINCIDENT WITH THE LEAD WHERE THE LEAD EXITS THE PLASTIC BODY AT THE BOTTOM OF THE PARTING LINE.
4. DATUMS **-L-**, **-M-**, AND **-N-** TO BE DETERMINED AT DATUM PLANE **-H-**.
5. DIMENSIONS S AND V TO BE DETERMINED AT SEATING PLANE **-T-**.
6. DIMENSIONS A AND B DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 0.25 (0.010 PER SIDE. DIMENSIONS A AND B DO NOT INCLUDE MOLD MISMATCH AND ARE DETERMINED AT DATUM LINE **-H-**.
7. DIMENSION D DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL NOT CAUSE THE D DIMENSION TO EXCEED 0.35 (0.014).

	MILLIMETERS		INCHES	
DIM	MIN	MAX	MIN	MAX
A	20.00 BSC		0.790 BSC	
A1	10.00 BSC		0.394 BSC	
B	20.00 BSC		0.790 BSC	
B1	10.00 BSC		0.394 BSC	
C	1.40	1.60	0.055	0.063
C1	0.05	0.15	0.002	0.006
C2	1.35	1.45	0.053	0.057
D	0.17	0.27	0.007	0.011
E	0.45	0.75	0.018	0.030
F	0.17	0.23	0.007	0.009
G	0.50 BSC.		0.20 BSC.	
J	0.09	0.20	0.004	0.008
K	0.50 REF		0.020 REF	
P	0.25 BSC		0.010 BSC	
R1	0.13	0.20	0.005	0.008
R2	0.13	0.20	0.005	0.008
S	22.00 BSC		0.866 BSC	
S1	11.00 BSC		0.433 BSC	
V	22.00 BSC		0.866 BSC	
V1	11.00 BSC		0.433 BSC	
Y	0.25 REF		0.010 REF	
Z	1.00 REF		0.039 REF	
AA	0.09	0.16	0.004	0.006
θ	0°		0°	
θ1	0°	7°	0°	7°
θ2	11°	13°	11°	13°

7.3 ORDERING INFORMATION

Package Type	Frequency (MHz)	Voltage	Temperature	Order Number
Pin Grid Array (RC Suffix)	16.67 16.67 20 20 25	5.0	0°C to 70°C - 40°C to + 85°C 0°C to 70°C - 40°C to + 85°C 0°C to 70°C	MC68302RC16 MC68302CRC16 MC68302RC20 MC68302CRC20 MC68302RC25
Plastic Surface Mount (FC Suffix)	16.67 16.67 20 20 25	5.0	0°C to 70°C - 40°C to + 85°C 0°C to 70°C - 40°C to + 85°C 0°C to 70°C	MC68302FC16 MC68302CFC16 MC68302FC20 MC68302CFC20 MC68302FC25
Thin Surface Mount (PV Suffix)	16.67 20 16.67	5.0 5.0 3.3	0°C to 70°C 0°C to 70°C 0°C to 70°C	MC68302PV16 MC68302PV20 MC68302PV16V

APPENDIX A
SCC PERFORMANCE

The MC68302 at 16.67 MHz was originally designed to support unrestricted operation of multiple (three) serial communications controllers (SCCs) servicing differing communications protocols with cost-effective usage of silicon at data rates of 256 kbps for HDLC-framed or transparent data handling and 128 kbps for BISYNC, DDCMP, V.110, or asynchronous framed data. The resultant design has exceeded these design goals.

Since the MC68302 serial channels will likely service serial channels time-division multiplexed into higher bandwidth channels, such as the U.S. and Japanese T1 and European CEPT primary rate channels, the physical clocking of the serial channels can be accomplished at the higher speeds required by these channels—up to a maximum of 40% (a 1:2.5 ratio) of the system clock frequency. This gives up to a 6.67-MHz serial clock for a 16.67-MHz IMP system clock. At this same 16.67-MHz system clock speed, the MC68302 can therefore handle the 1.544-MHz and 2.048-MHz clocking frequencies of T1 and CEPT lines as well as the 4.096-Mbps signaling rates of the common ISDN interchip local buses, such as IDL and GCI (also known as IOM-2).

Thus, the MC68302 is well equipped to handle, for instance, three 256-kbps channels multiplexed on a T1 or CEPT primary rate channel.

Where an application requires even higher bandwidth channels, such as the 384-kbps H0 channels, the 1.536-Mbps H11 channel, or higher, the following restriction should be observed: bus latency of the SDMA must be less than 20 system clocks. (The $\overline{\text{BCLR}}$ signal may be helpful here.)

NOTE

A previous revision of this manual showed two additional restrictions which have now been eliminated.

If the above restriction is adhered to, the following table shows experimental performance data obtained with the device.

The frequency ratio stated represents the system (EXTAL) frequency to serial bit rate frequency. A user exceeding this bit rate will begin to experience SCC underruns and/or overruns. Some users may wish to tolerate an occasional underrun/overrun to slightly increase performance.

For example, a ratio of 1:10 in the following table shows that an MC68302 system clock of 16.67 MHz can support an MC68302 serial rate of 1.67 Mbps. Typically, this 1.67-Mbps rate would be achieved with 1 bit every 1.67-MHz serial clock. However, it may also be achieved

with a faster serial clock (subject to the clocking limits of the SCC mentioned previously) as long as the bit rate over a 9-bit (17-bit period for HDLC or transparent) period averages out to 1.67 Mbps.

High-Speed Channels		Low-Speed Channels		Comments/Restrictions
Number	Frequency Ratio	Number	Frequency Ratio	
1 HDLC	1:7	—	—	Buffers in Dual-Port RAM
1 HDLC	1:9	—	—	
2 HDLC	1:22	—	—	
3 HDLC	1:37	—	—	
1 HDLC	1:9 1:9 1:10 1:10 1:9 1:10 1:9 1:10 1:9 1:9 1:10	UART 2 UART 2 UART BISYNC DDCMP DDCMP HDLC HDLC 2 HDLC 2 HDLC 2 HDLC	1:20 (*16) 1:396 (*16) 1:10 (*16) 1:98 1:366 1:241 1:98 1:57 1:224 1:128,238 1:128	Half-Duplex Bisync
2 HDLC	1:22 1:23 1:24 1:24 1:25 1:23 1:24 1:23 1:24	UART UART UART BISYNC BISYNC DDCMP DDCMP HDLC HDLC	1:329 (*16) 1:305 (*16) 1:24 (*16) 1:496 1:177 1:1151 1:229 1:241 1:113	Half-Duplex Bisync Half-Duplex Bisync
3 UART	1:2.5	—	—	UART Uses 16x Clock
3 BISYNC	1:60	—	—	Half-Duplex Bisync
3 DDCMP	1:75	—	—	
1 Transp	1:10	—	—	
2 Transp	1:23	—	—	
3 Transp	1:35	—	—	

NOTES:
1. SCC performance calculation example with a 16.67-MHz system clock:
   One HDLC channel can operate with a ratio of 1:9. Thus,16.67-MHz/9 gives 1.8 Mbps, and a 20-MHz system clock gives 2.22 Mbps.
2. "Difficult" buffer parameters were chosen as shown below. Use of less difficult parameters does not significantly improve performance. The SCCs transmit and receive from/into multiple buffers per frame. Tx BD 1 address is ODD and has 59 bytes; whereas, the Tx BD2 address is EVEN and has 60 bytes. In HDLC mode, Rx BD1 address is EVEN, but is ODD in other modes. Rx BD1 is (frame length-1) bytes long and Rx BD2 is 1 byte long.
3. The external RAM access time is two wait states. As a general rule, the addition of a wait state only decreases maximum performance by about 1%.
4. The last address or control character in the table was checked.
5. For the DDCMP, the frame length was 6 and 59 bytes for the two BDs.
6. When the performance of a high-speed channel together with a low-speed channel was measured, the high-speed

channel was always SCC1.

7. This data applies to MC68302 masks 2B14M, 3B14M, or later.
8. The following explanation concerns a fast HDLC channel and two slower channels: When the fast HDLC is 1:9, two HDLCs can run at 1:224. Thus, with a 16.67-MHz dock, SCC1 can run at 1.85 Mbps; SCC2 and SCC3 can run at 74 kbps. Two HDLCs can also run without equal values: one at 1:128 and one at 1:238. When the fast HDLC is 1:10, two HDLCs can run at 1:128. When the fast HDLC is 1:9, two UARTs can run at 1:396 (*16). When the fast HDLC is 1:10, two UARTs can run at 1:10 (*16).
9. Performance results above showed no receive overruns or transmit underruns in several minutes of continuous transmission/reception. Reduction of the above ratios by a single value (e.g., 1:35 reduced to 1:34) did show an overrun or underrun within several minutes.
10. All results assume the DRAM refresh controller is not operating; otherwise, performance is slightly reduced.
11. Unless specifically stated, all table results assume continuous full-duplex operation. Results for half-duplex were not measured, but will be roughly 2x better.

Since operation at very high data rates is characteristic of HDLC-framed channels rather than BISYNC-, DDCMP-, or async-framed channels, the user can also use the MC68302 in conjunction with either the Motorola MC68605 1984 CCITT X.25 LAPB controller, the MC68606 CCITT Q.921 multilink LAPD controller, or the MC145488 dual data link controller. These devices fully support operation at T1/CEPT rates (and above) and can operate with their serial clocks "gated" onto subchannels of such an interface. These devices are full M68000 bus masters. The MC68605 and MC68606 perform the full data-link layer protocol as well as support various transparent modes within HDLC-framed operation. The MC145488 provides HDLC-framed and totally transparent operation on two full-duplex channels.

APPENDIX B
DEVELOPMENT TOOLS AND SUPPORT

B.1 MOTOROLA SOFTWARE OVERVIEW

A software development package is offered as a set of independent modules which provides the following features:

- IMP Chip Evaluation
  A development board, described in B.5 302 Family ADS System, can be used as a prototyping vehicle for user code.
- IMP Chip Drivers
  Written in C, the source code of the IMP drivers is available on electronic media. These drivers were developed to support the needs of the protocol implementations described below.
- Protocol Implementations
  Modules implementing common ISO/OSI layer 2 and 3 protocols are available under license. These are in the form of binary relocatable object code as well as source code written in C.
- Portability
  All of the modules may be ported to different MC68302 implementations and combined with user-developed code. Well-defined software interface documentation is available for all provided modules.

B.2 MOTOROLA SOFTWARE MODULES

Chip driver routines written in C illustrate initialization of the IMP, interrupt handling, and the management of data transmission and reception on all channels.

In addition to the chip drivers, protocol modules are provided. Layer 2 modules include LAPB and LAPD. (Support for layer 2 BISYNC and DDCMP functions are also provided with the chip driver package.) The layer 3 module supported is the X.25 packet layer protocol.

Since the modules do require some minimal operating system services, they are written to use our EDX operating system. Use of EDX with the protocol modules is not required as long as some other operating system support is provided by the user.

All modules communicate with each other using message passing. This technique simplifies the interfaces between the different modules and enables them to interface with other user-added modules with relative ease. Detailed descriptions of the software interfaces are available.

Each module operates completely independent of its environment. Independence from the hardware environment is achieved by the fact that each module is individually configurable and relocatable anywhere in system memory. Calls are used for requesting resources or services from the host operating system (memory management and message passing), which creates independence from the firmware. Modules may be used with any combination of other modules, including those added by the user (see Figure B-1).

The chip driver module features are as follows:

- All software modules use the driver module as their interface to the MC68302
- Illustrates initialization of the MC68302 and interrupt handling
- Provides complete configuration of the MC68302 on system start (or restart)
- Provides services for the MC68302 serial ports

  —- Supports linked lists of frames for both transmit and receive
  — Provides error handling and recovery for both transmit and receive
  — Supports all three serial ports in different protocol configurations
  — Provides internal loopback (frames not sent to MC68302)
- Provides timer services
- Can be configured to send “trace” messages to a system log file
- Supports up to 24 serial ports

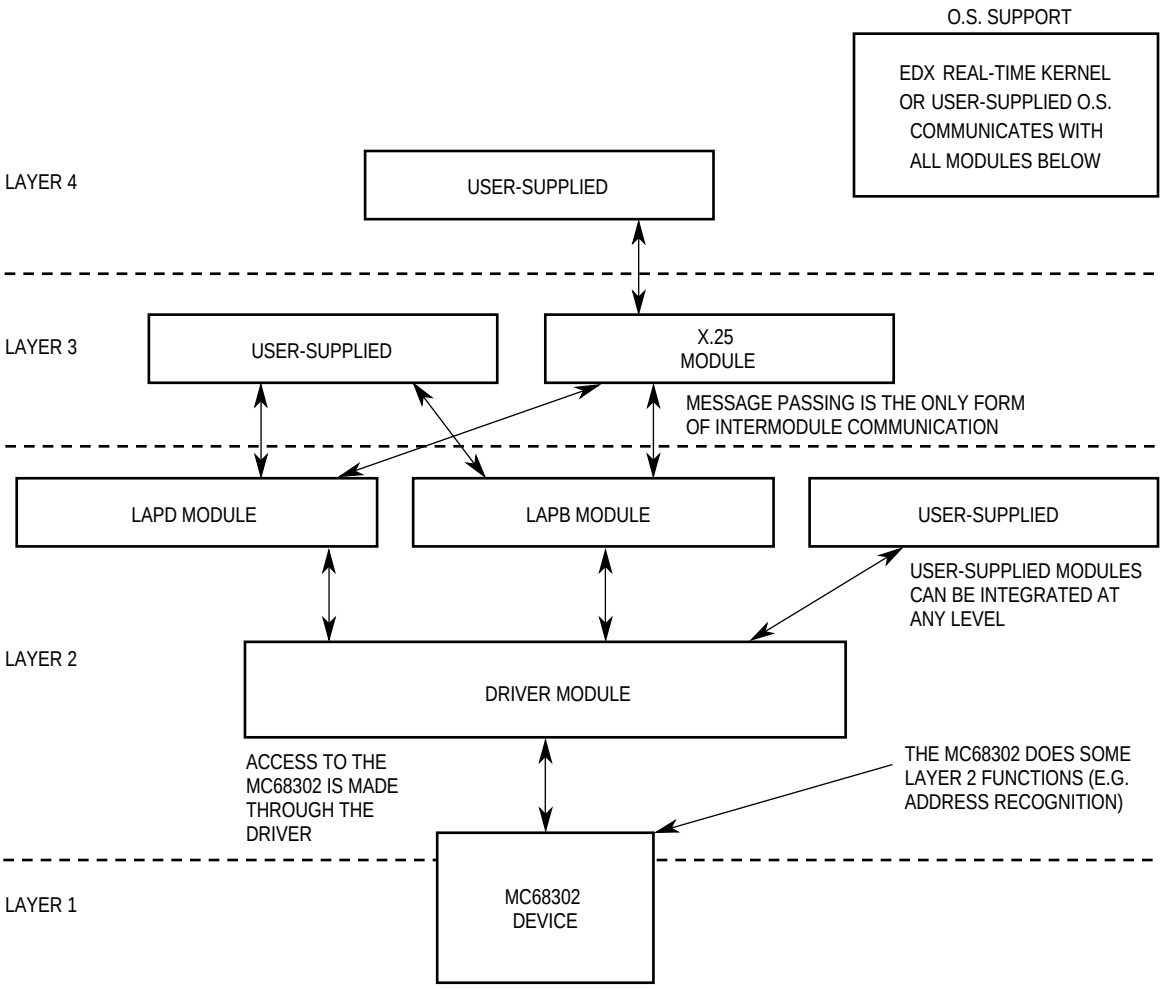


Figure B-1. Software Overview

The LAPD module features are as follows:

- Fully implements *1988 CCITT Recommendation Q.920/Q.921*
- Supports up to twelve physical channels—8192 logical links per channel
- Supports management and broadcast links
- Applicable for user and network applications
- Uses dedicated transmit pool for fast control frame generation
- Dynamic modification of protocol parameters
- Independent of layer 1 and layer 3 implementation
- Independent of layer 2 management implementation
- Message-oriented interface
- Independent configuration of upper and lower layer modules interfacing with each link
- Special mode for internal loopback between pairs of links

- Supports external loopback between two channels or on the same channel
- Trace option for reporting to system management each primitive issued by the LAPD module to the layer 3 module or to layer 2 management

The LAPB module features are as follows:

- Implements *1988 CCITT Recommendation X25,* chapters 2.1 through 2.4
- Supports up to twelve distinct physical channels with each operating as an independent station
- Modulo 8 or modulo 128 operation
- Applicable for DTE and DCE applications
- Uses dedicated transmit pool for fast control frame generation
- Dynamic modification of protocol parameters
- Independent of layer 1 and layer 3 implementation
- Message-oriented interface
- Independent configuration of upper and lower layer modules interfacing with each LAPB link
- Special mode for internal loopback (frames are not sent to the driver)
- Supports external loopback between two MC68302 serial channels or on the same MC68302 serial channel
- Trace option for reporting to system management each primitive issued by the LAPB module to layer 3 or to layer 2 management

The X.25 module features are as follows:

- Fully implements 1988 CCITT Recommendation X.25, chapters 3.1 through 7.3
- May be used with both layer 2 modules: LAPD or LAPB
- Unlimited number of layer 2 entities (interfaces)
- Supports up to 4095 logical channels for each interface
- Many DTE/DCE interface parameters (configurable for each interface):

  —- DTE/DCE
  —- Modulo 8/128
  —- Window size
  —- Maximum receive and transmit packet lengths

- Layer 4 message fragmentation/assembly using M-BIT
- Q-BIT support
- All standard CCITT X.25 facilities
- Compatible with X.213 interface primitives
- Link parameters (configurable separately for each interface):

  —- Interface ID

—DTE/DCE
—- Permanent virtual circuit/virtual call
—- D-BIT support
—- On-line registration support
—- TOA/NPI address mode
—- Fast select facility support
—- Logical channel (and group) numbers
—- Protocol parameters (w, T11, T12, T21, T22, N12, N13)
—- Maximum data packet length (unlimited)

- Message-oriented interface
- Independent of layer 2 and layer 4 implementation

The EDX module features are as follows:

- The EDX event driven executive is an operating system kernel that provides:

  —Multitasking with simple task scheduling
  —Message passing between tasks
  —Memory allocation

- EDX was designed to be:

  —Fast (written mostly in M68000 Assembler)
  —Efficient (uses about 1.5 kbytes of ROM)

- EDX was designed for use in communications environments. It provides “soft” real-time scheduling, not the “hard” real-time scheduling needed in many event control applications.

The approximate compiled object code sizes are as follows:

—Chip Drivers	24K
—LAPD	24K
—LAPB	20K
—X.25	36K
—EDX	1.5K

A RAM scratchpad area (around 4 kbytes) and buffer space are also required for the modules (except for EDX). Code sizes may be reduced from the figures above if optional module features are not compiled.

An additional user-interface module provides a menu-driven user interface to the IMP and each functional protocol module. This module may be used for chip evaluation or as a tool for debugging user-developed applications. Menu options will allow a user to examine the appropriate module's memory structures (or the register set and on-chip dual-port RAM of the IMP) or to issue specific commands. The commands may result in specific IMP commands or in the execution of protocol-defined primitives in one of the protocol modules.

The chip driver module is available in C source code form. The source code requires no license.

The LAPB, LAPD, and X.25 modules are available in C source code form. The source code for these modules requires a license from Motorola.

The EDX kernel is available in assembler source code form. The source of the EDX kernel requires a license from Motorola.

Contact your local Motorola sales office to see license terms.

B.3 THIRD-PARTY SOFTWARE SUPPORT

Since the IMP is a memory-mapped device based on a full M68000 core, existing compilers, source-level debuggers, assemblers, and linkers designed for the M68000 Family may be successfully used. Many third parties supply such tools. Contact your local representative for a list of third party suppliers.

B.4 IN-CIRCUIT EMULATION SUPPORT

Full in-circuit emulation support is available from multiple third parties, but is not discussed in this manual.

B.5 302 FAMILY ADS SYSTEM

The M68302FADS is an integrated Family Applications Development System (FADS) designed to aid hardware and software developers of the MC68302, MC68LC302, MC68PM302, and MC68EN302 in quickly evaluating and developing applications for these devices. All of the hardware resources needed to download and debug application software are provided, such as large blocks of flash and static RAM for the processors, serial ports, clock generation circuitry, logic analyzer connectors, expansion connectors as well as monitor/debugger hardware and software. The logic analyzer connectors provide the user with access to all of the processors' pins in order to monitor bus activity. The expansion connectors let the user attach hardware applications and to use board resources to verify a design.

To serve as a convenient platform for software development, the M68302FADS is provided with a monitor/debugger for the Integrated Multiprotocol Processor (IMP) section. The monitor/debugger provides the following operations: memory dump and set (with optional disassembly of 68K instructions), single instruction execution, breakpoints, and downloads. The debugger interface can work together in separate Windows 3.0 DOS shells on the same x86 based PC to communicate with the on-board IMP hardware. Future support for SUN platforms is planned.

The M68302FADS board has sockets for and is shipped with the MC68302RC25, MC68LC302RC20, MC68PM302RC20. An adapter card for the MC68EN302 will be available separately.

- General Features
  - —Supports the 68302 family of processors: MC68302, MC68LC302, MC68PM302 and MC68EN302 (with an adaptor).
  - —On board IMP(68302) debugger software with host debugger interface.
  - —Separate external clock generators for the IMP(68302)

—PCMCIA port connector with extender card to plug directly into PCMCIA sockets.

—Expansion connectors providing all the 68302 family device signals.

—68000 bus signals brought out to logic analyzer connectors.

—Single +5Vdc power supply with onboard 5V to +/-12Volt converter.

- IMP(68302) Support Features Included

  —MC68302 at 25 MHz

  —MC68LC302 at 20 MHz

  —MC68PM302 at 20 MHz.

  —512 kbyte, zero wait state static RAM, expandable up to 1 Mbyte. (16 bit orientation)

  —1 Mbyte FLASH. (16 bit orientation)

  —2 kbyte EEPROM. (8 bit orientation)

  —MC68681 DUART, with two RS232 serial ports.

  —Serial RS-232 terminal connectors

  —RESET and ABORT controls.

  —RUN and HALT status indicators (LED’s).

  —Bus expansion connector pin out, is compatible with the 302ADS

  —ADI port connector.

.

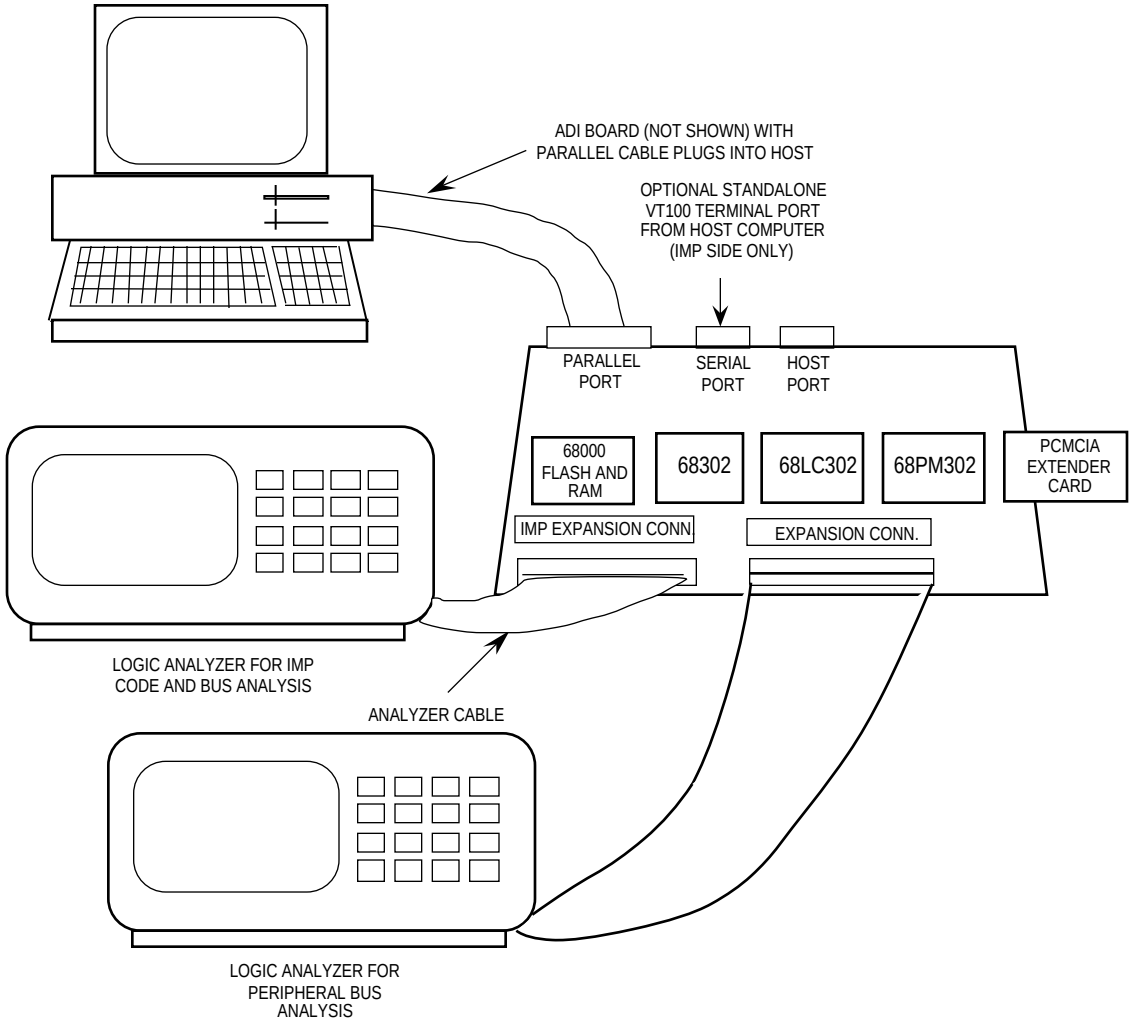


Figure B-2. MC68302FADS

APPENDIX C
RISC MICROCODE FROM RAM

The MC68302 RISC processor has an option to execute microcode from the 576-byte user RAM in the on-chip dual-port RAM. In this mode, the 576-byte user RAM cannot be accessed by the M68000 core or other M68000 bus masters. Also in this mode, port A pins are optionally available to the RISC processor as well as the M68000 core. Figure C-1 shows these resulting changes.

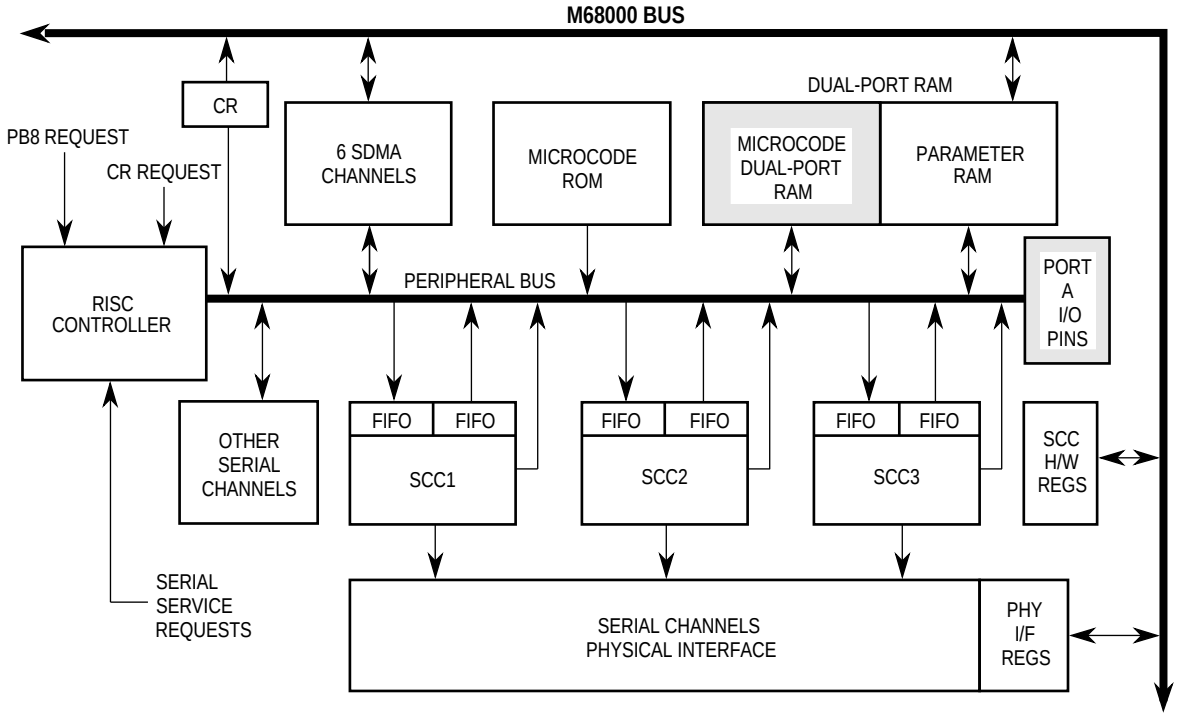


Figure C-1. CP Architecture Running RAM Microcode

Once the microcode has been loaded into the dual-port RAM by the M68000 core or other bus master, the microcode from RAM option is enabled in the reserved register at location $0F8. When the user writes $0001 to location $0F8, the RISC processor will execute microcode from RAM once the CP is reset in the command register. Hereafter, the RISC processor can freely address both the dual-port RAM and its own private ROM.

Microcode for a new application or protocol is developed only under special arrangement and coordination with Motorola.

Some RAM microcode routines are also available for purchase. The following is an overview of available functions. Contact a Motorola sales office for detailed specifications of the microcode routines.

C.1 SS7 PROTOCOL SUPPORT

The HDLC routines are enhanced and extended to provide special Signaling System #7 (SS7) support. In addition to the HDLC features of the MC68302, the following key features are included:

- Automatic Fill-In Signal Unit (FISU) Transmission and Reception
- Automatic Link Status Signal Unit (LSSU) Retransmission
- Octet Counting Mode Support
- Two Signal Unit Counters (error and error-free signal units)

Any or all three SCCs can become an SS7 protocol controller. The SS7 microcode increases that portion of layer 2 SS7 already supported by the HDLC features in the MC68302. This implementation of SS7, however, is not a complete layer 2 implementation. Rather, the additional features included in the SS7 controller are those that would have been especially difficult and/or time-consuming to support by the M68000 core alone, due to their continuous or real-time nature.

C.2 CENTRONICS TRANSMISSION CONTROLLER

The RISC processor implements a Centronics parallel interface using the PA7–PA0, PA12–PA8, and the PA15 parallel l/O pins. All SCC channels and protocols, including DRAM refresh, are supported concurrently with this feature, but SCC2, if used, must be configured in a multiplexed mode, and several of its buffer descriptors are used by the Centronics interface. Once a buffer is configured in memory, the protocol handles the movement of all data from the buffer to the interface.

This package implements the transmitter function of the Centronics interface. (Another package can implement the receiver function.)

The features supported are as follows:

- Three Flexible Data Buffers for Transmission
- Controls the Eight Data Lines and Data Strobe
- Supports the PE, SLCT, $\overline{\text{FAULT}}$, $\overline{\text{BUSY}}$ and $\overline{\text{INPUT}}$ $\overline{\text{PRIME}}$ Conditions
- Supports the Start Print and Stop Print Commands
- Interrupts May Be Generated After Each Buffer or Upon an Error Condition
- Transmission Rates up to 260 kbytes/sec.

C.3 CENTRONICS[1] RECEPTION CONTROLLER

With this package, SCC2 is turned into an interface that can receive Centronics data. This package, for example, could be used in a laser printer design requiring a Centronics interface option. The package uses SCC2 (its pins and registers) in addition to PA12, PA13 or PA11, PA14, and PA15.

The main features are as follows:

- Flexible Message-Oriented Data Structure
- Flexible Control Character Comparison
- Controls the $\overline{\text{BUSY}}$ and $\overline{\text{ACK}}$ Signals
- Three Timing Relationships Possible between $\overline{\text{BUSY}}$ and $\overline{\text{ACK}}$
- Supports the SET_BUSY and CLEAR_BUSY Commands
- Programmable Duration (in clock cycles) of the $\overline{\text{ACK}}$ Signal
- Receive Buffer May Be Closed upon a Programmable Silence Period

C.4 PROFIBUS CONTROLLER

Process field bus (PROFIBUS) is a UART-based master-slave protocol that specifies data rates starting at 9.6 kbps. The PROFIBUS microcode running on the IMP RISC controller assists the M68000 core in handling some of the time-critical PROFIBUS link layer functions, leaving more of the M68000 core available for the application software.

The main PROFIBUS controller features are as follows:

- Automatic Frame Synchronization by Searching for the Start Delimiter
- Frame Preceding IDLE Sequence Generation/Checking
- Flexible Frame-Oriented Data Buffers
- Separate Interrupts for Frames and Buffers (Receive and Transmit)
- Maintenance of Six 16-Bit Error Counters
- Two Address Comparison Registers with Mask
- Frame Error, Noise Error, Parity Error Detection
- Detection of IDLE in the Middle of a Frame
- Check Sum Generation/Checking
- End Delimiter (ED) Generation/Checking
- Three Commands

C.5 AUTOBAUD SUPPORT PACKAGE

The package allows a UART to automatically detect the baud rate of a serial stream and adjust to it. This is useful in modem applications that must support an autobaud capability. It is

1. Centronics is a trademark of Centronics.

designed to comply with the requirements of the Hayes AT command set; however, it can be used with simpler character schemes as well (such as a carriage return).

The SCC receiver synchronizes on the falling edge of the START bit. Once a start bit is detected, each bit received is processed by the AutoBaud controller. The AutoBaud controller measures the length of the START bit to determine the receive baudrate and compares the length to values in a user supplied lookup table. After the baudrate is determined, the AutoBaud controller assembles the character and compares it against two user-defined characters. If a match is detected, the AutoBaud controller interrupts the host and returns the determined nominal start value from the lookup table. The AutoBaud controller continues to assemble the characters and interrupt the host until the host stops the reception process. The incoming message should contain a mixture of even and odd characters so that the user has enough information to decide on the proper character format (length and parity). The host then uses the returned nominal start value from the lookup table, modifies the SCC Configuration Register (SCON) to generate the correct baudrate, and reprograms the SCC to UART mode.

Many rates are supported including: 150, 300, 600, 1200, 2400, 4800, 9600, 14.4K, 19.2K, 38.4K, 57.6K, 64K, 96K, and 115.2K. To estimate the performance of the AutoBaud microcode package, the performance table in Appendix A of the MC68302 user's manual can be used. The maximum full-duplex rate for a BISYNC channel is one-tenth of the system clock rate. So a 16.67 MHz 68302 can support 115.2k autobaudrate with another low-speed channel (<50 kbps) and a 20 MHz MC68302 can support 115.2k AutoBaudrate with 2 low-speed channels. The performance can vary depending on system loading, configuration, and echoing mode.

C.6 MICROCODE FROM RAM INITIALIZATION SEQUENCE

1. Perform a total system reset of the MC68302.
2. Write $0700 to the BAR. The base address of the internal dual-port RAM after this action is $700000 (hex). If a different base address is desired, the S-record file addresses should be modified to the desired address.
3. Load the S-record file data into the internal dual-port RAM. (In a production environment, the microcode may be copied from EPROM directly to the internal dual-port RAM.)
4. Write $0001 to address $0F8 in supervisory space.
5. Write a software reset command to the CR.
6. Continue with the normal initialization sequence.

APPENDIX D
MC68302 APPLICATIONS

This appendix describes different applications for the MC68302.

D.1 MINIMUM SYSTEM CONFIGURATION

The following paragraphs describe a minimum 16-bit MC68302 system. As Figure D-1 shows, this system can be easily built with very few components.

D.1.1 System Configuration

The crystal circuit shown in Figure D-1 is a typical configuration. The values used are not required by Motorola. Some deviation of the capacitance or resistance values is allowed. Crystal parameters need not be anything special—Co < 10 pF and Rx = 50 Ω is used. Of course, an oscillator could be used in place of the crystal circuit.

$\overline{\text{AVEC}}$ is pulled high since autovectoring for external interrupts is not needed. If external devices were added (not shown) the MC68302 interrupt controller could handle the interrupt vector generation for up to seven external sources using $\overline{\text{IRQ7}}$, $\overline{\text{IRQ6}}$, $\overline{\text{1RQ1}}$, PB11, PB10, PB9, and PB8. The $\overline{\text{IPL2}}$-$\overline{\text{IPL0}}$ lines are also pulled high (inactive) since no external interrupts are required.

BUSW is pulled high for 16-bit operation and may not be modified dynamically. Choice of 8-bit operation could be used to eliminate two of the memory chips.

$\overline{\text{BERR}}$ is pulled high since it is an open-drain signal. It will be asserted low by the MC68302 if the hardware watchdog terminates a stalled bus cycle.

$\overline{\text{BR}}$ is tied high since no external bus masters exist in this design.

$\overline{\text{BGACK}}$ is pulled high (inactive). It is asserted low during an IDMA or SDMA bus cycle.

$\overline{\text{FRZ}}$ is tied high since the MC68302 freeze debugging logic is not used in this design.

DISCPU is tied low to allow the M68000 core to function normally. Tying this pin high causes the part to enter the disable CPU mode when the core is disabled and the part is an intelligent peripheral.

All unused lines should be terminated.

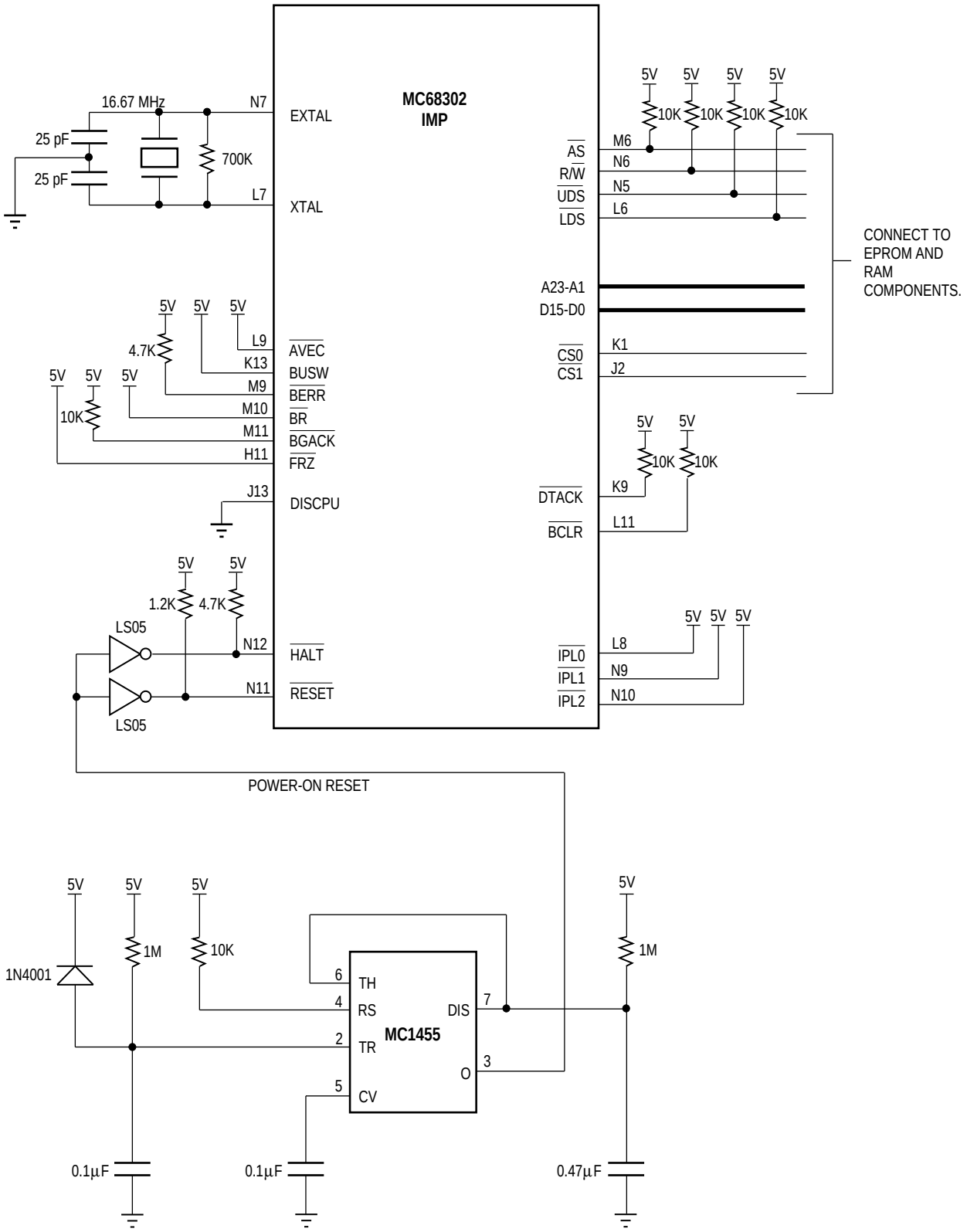


Figure D-1. MC68302 Minimum System Configuration (Sheet 1 of 2)

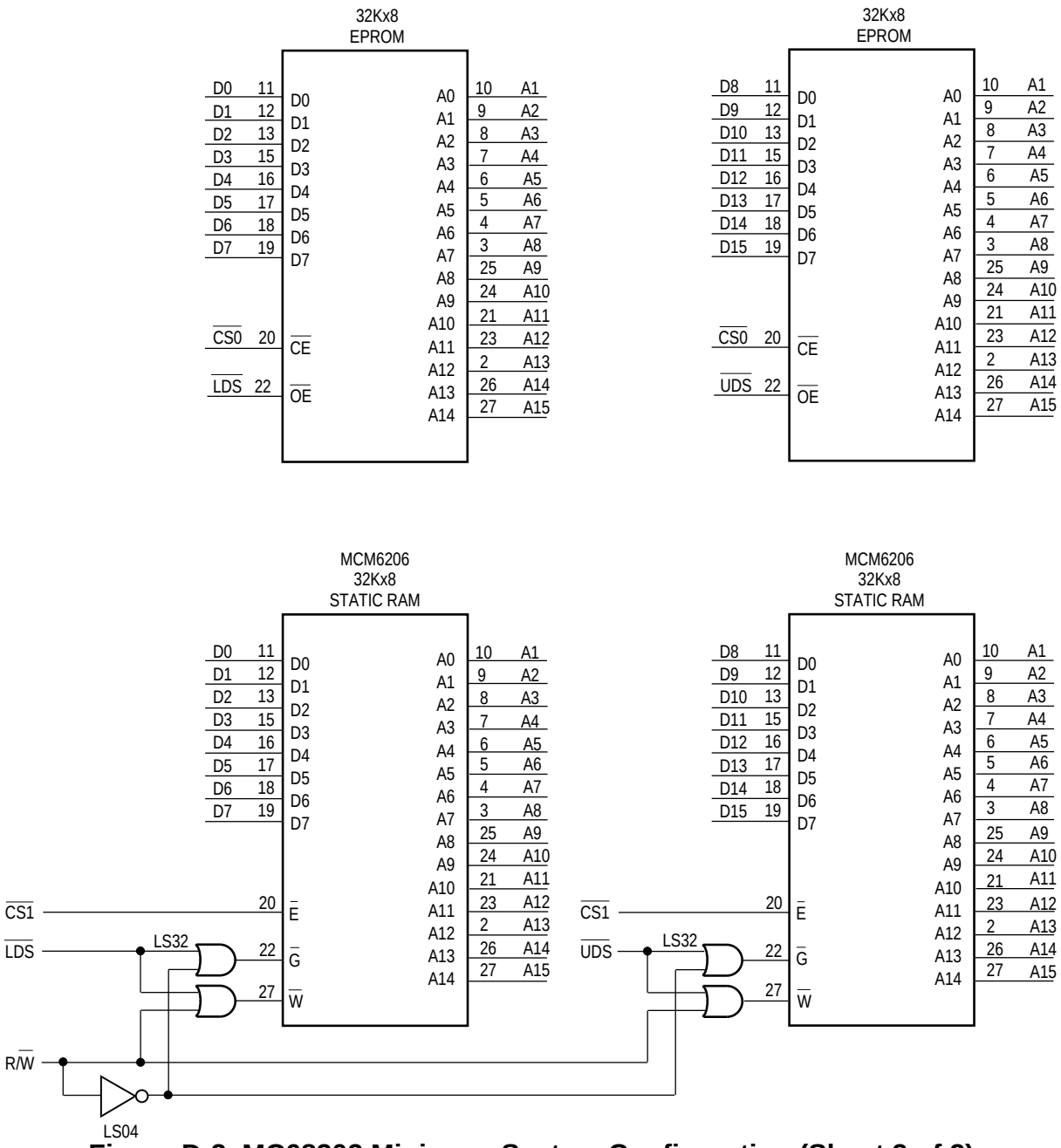


Figure D-2. MC68302 Minimum System Configuration (Sheet 2 of 2)

D.1.2 Reset Circuit

$\overline{\text{HALT}}$ and $\overline{\text{RESET}}$ are both open-drain signals. When both signals are externally asserted, the entire MC68302 is reset. $\overline{\text{HALT}}$ is asserted by the MC68302 when the HALT instruction is executed, and $\overline{\text{RESET}}$ is asserted when the RESET instruction is executed.

The reset circuit shown uses an MC1455 timer to generate a 0.5-sec pulse *until* DIS reaches the 2/3 V_{CC} threshold level. After that, DIS discharges the 0.47-µF capacitor. When DIS falls below 2/3 V_{CC}, then the output pin (O) is pulled low, ending the reset to the MC68302. This

circuit will also place the MC68302 into reset if V_{CC} drops below a threshold, an advantage over a simple RC circuit.

D.1.3 Memory Interface

$\overline{AS}$, R/$\overline{W}$, $\overline{UDS}$, $\overline{LDS}$, A23–A1, D15–D0, $\overline{CS0}$, and $\overline{CS1}$ are used in the memory interface. The bidirectional control signals have pullups so that they are inactive when the MC68302 is not actively driving them. $\overline{CS0}$ and $\overline{CS1}$ are only outputs and do not need pullups.

$\overline{DTACK}$ is generated internally by the wait-state generation logic, and therefore it does not need to be externally connected. It is driven by the wait-state generation logic during chip-select accesses and is pulled up externally through a resistor.

$\overline{BCLR}$ is an open-drain signal that is output by the MC68302 when the SDMA requests the bus or (if enabled) when an interrupt request is pending. It is therefore pulled up externally.

D.1.4 Memory Circuit

The EPROM design is a straight connection. Larger EPROM sizes may be easily substituted; it is also possible to use a single 16-bit EPROM, if desired, and reduce the component count by one.

$\overline{CS0}$ is used to select the EPROM since $\overline{CS0}$ is designed to be used for a boot ROM. $\overline{CS0}$ comes up enabled for the first 8 kbytes of the system address space. Before the program jumps outside of this space, it should configure the $\overline{CS0}$ range for 32 kbytes and enable the RAM chip select to cover the RAM range and desired starting address. To switch the RAM down to low-order memory to allow exception vectors to be altered, see D.2 Switching the External ROM and RAM Using the MC68302**.**

$\overline{LDS}$ and $\overline{UDS}$ define the lower and upper bytes of the program word, respectively, and are connected to the EPROM output enable.

The RAM design is similar to the EPROM design, except that the R/$\overline{W}$ signal is used to qualify the output enable and the $\overline{LDS}$/$\overline{UDS}$ signals are used to qualify whether to write the particular byte of the data word (the M68000 supports byte operations).

$\overline{CS1}$ is used to select the RAM.

No external buffers are required in this small system design.

D.1.5 Memory Timing Analysis

The design as shown will work with zero wait states with a 100-ns EPROM and a static RAM, such as the MCM6202. The RAM writes are controlled by the enable ($\overline{E}$) rather than by the write ($\overline{W}$). Two of the MC68302 required specifications deserve special note.

The time between $\overline{CS}$ low and data valid must be calculated for proper timing of read cycles. This time is approximately 2 1/2 clocks. $\overline{CS}$ is asserted by the MC68302 after the rising edge of S2. Data must be valid a setup time before the S6 falling edge. Thus, for a 16.67-MHz device, the equation for the time between $\overline{CS}$ low and data valid is as follows:

2-1/2 clocks –data setup time–$\overline{CS}$ maximum asserted time = 150 - 7 - 40 = 103 ns.

For proper timing of write cycles, it is important to know how much time elapses between $\overline{CS}$ negated and the data out hold time. With a 16.67-MHz MC68302, this value is a 10-ns minimum. Thus, on a $\overline{CS}$-controlled RAM write cycle, the RAM required data hold time must be <= 10 ns. For the MCM6206, it is 0 ns.

D.2 SWITCHING THE EXTERNAL ROM AND RAM USING THE MC68302

When designing with the MC68302 (or simply the M68000), one of the tasks often required is to switch the locations of the ROM and RAM in the system. This is because the reset vector needs to be supplied from ROM; whereas, the rest of the exception vectors are normally included in RAM so that they can be modified. The exception vector area extends from $0 to $FF.

The MC68302 makes this switch easy to perform, requiring no additional glue logic. Two chip-select lines and a few temporary locations of the dual-port RAM are used. For this example, 256 kbytes of EPROM and 64 kbytes of RAM are used.

D.2.1 Conditions at Reset

We physically connect $\overline{CS0}$ to the ROM and $\overline{CS1}$ to the RAM. After reset, the MC68302 defaults to having only one chip select enabled ($\overline{CS0}$), with a base address of $0 and a range of 8 kbytes. The ROM should be programmed with an initial set of interrupt vectors, and the reset vector should point to some location within the first 8 k of ROM. $\overline{CS1}$ is disabled at reset so the RAM cannot be accessed. Also, the base address register (BAR) in the MC68302 has not been written; therefore, the dual-port RAM and other on-chip peripherals cannot be accessed. The status register (SR) of the M68000 core is $2700, putting the core in supervisor mode with interrupts masked. The SR is left in this condition until the ROM and RAM are switched.

D.2.2 First Things First

The following situation now exists:

- ROM—Begins at $0; only first 8 kbytes are valid for programming.
- RAM—Undefined.
- MC68302—Undefined (except BAR at $0F2 and SCR at $0F4).

Once the reset vector has been taken, the BAR at location $0F2 should be written to locate the MC68302 dual-port RAM and on-chip peripheral registers in memory. In this example, $700000 is chosen as the starting point of the 4 k block of the MC68302 internal address space; thus, we write $0700 to BAR. BAR exists internal to the MC68302 and is easily accessed even though it overlaps memory with ROM. $\overline{CS0}$ *will not* activate on accesses to the BAR.

Now that the MC68302 can be fully accessed, the ROM and RAM chip selects can be modified. First, the option register (OR) of $\overline{CS0}$ can be extended to include all 256 kbytes by writing the base address mask in OR0 with 11111100000b. At this time, the DTACK field may be changed to reduce the number of wait states from six to something lower. Also, while writ-

ing this register, the compare function code (CFC) bit should be cleared, since data stored in the program ROM will need to be accessed and moved to other areas of memory. (At reset, the function code for $\overline{CS0}$ defaults to 110b to select supervisor program, and the function code comparison is enabled). Thus, OR0 could be written with $5F80 for this example.

Next, the RAM addresses should be defined. To set the range of the RAM for 64 kbytes, to configure it for zero wait states, to allow both reads and writes, and to disable function code comparisons, OR1 should be set to $1FE0. To initially place the RAM at $400000, enable the RAM and set the function code to supervisor data; BR1 should be set to $A801.

Now that the RAM is enabled, an initial set of vectors may be placed in it (i.e., copied from the ROM). Of course, they will not be accessed as vectors until the RAM is moved to location 0. Any exceptions that occur during this time will still have their vectors derived from the ROM vector table.

The following situation now exists:

ROM—$0 to $03FFFF

RAM—$400000 to $40FFFF

MC68302—$700000 to $700FFF

D.2.3 Switching Process

To perform the switch, jump from the ROM to the dual-port RAM, reconfigure chip selects 1 and 0, and then jump back to the ROM. It is important that the RAM be moved first to ensure that an exception vector table is always present. During the brief period while the locations in $\overline{CS1}$ and $\overline{CS0}$ overlap, $\overline{CS0}$ will take precedence.

To perform the switch we need to execute a short dual-port RAM program. Thus, we copy the following data (instructions) from ROM to the dual-port RAM starting at location $700000:

```
MOVE.W     #$A001, ($700834).L   ; Place CS1 at $0 by writing to BR1
MOVE.W     #$C201, ($700830).L   ; Place CS0 at $100000 by writing to BR0
JMP       ($Address in ROM).L
```

Thus, first copy the binary coding of the preceding program from ROM to the dual-port RAM starting at location $700000.

Next, execute

```
JMP       ($700000).L            ; Jump to Dual-Port RAM
```

which then causes the following final situation:

RAM—$000000 to $00FFFF

ROM—$100000 to $13FFFF

MC68302—$700000 to $700FFF

The initialization of chip selects 0 and 1 and the switch of ROM and RAM are now complete.

D.3 MC68302 BUFFER PROCESSING AND INTERRUPT HANDLING

The following paragraphs describe how to build an algorithm to process the buffers for the MC68302 serial communication controller (SCC) channels.

D.3.1 Buffer Descriptors Definition

Data buffers used by the MC68302 are controlled by buffer descriptors (BDs). The general structure of a BD is shown in Figure D-3. The processing of buffers by software is done by examining BDs. Thus, BDs are the focus of this discussion.

Transmit BD

OFFSET + 0	READY	OTHER STATUS AND CONTROL
OFFSET + 2	DATA LENGTH (NUMBER OF BYTES TO SEND)	
OFFSET + 4	32-BIT BUFFER POINTER (24 BITS USED)	

Receive BD

OFFSET + 0	EMPTY	OTHER STATUS AND CONTROL
OFFSET + 2	DATA LENGTH (0000 INITIALLY)	
OFFSET + 4	32-BIT BUFFER POINTER (24 BITS USED)	

Figure D-3. Transmit and Receive BD Structure

Each transmit BD has a very important bit called the “ready” bit. This bit is set by the M68000 user program to signify to the SCC that the BD has data ready for sending. Similarly, the “empty” bit tells whether a receive BD is empty and can be used by the SCC for locating an empty buffer to store incoming data.

In the MC68302, up to 8 receive BDs and 8 transmit BDs can be defined per SCC. These BDs are stored in predefined places in the MC68302 dual-port RAM. The “wrap” bit is set in the last BD, causing the SCC to wrap back around to the first BD when processing of the last BD is complete. Thus, each set of BDs form a circular queue. An example is shown in Figure D-4.

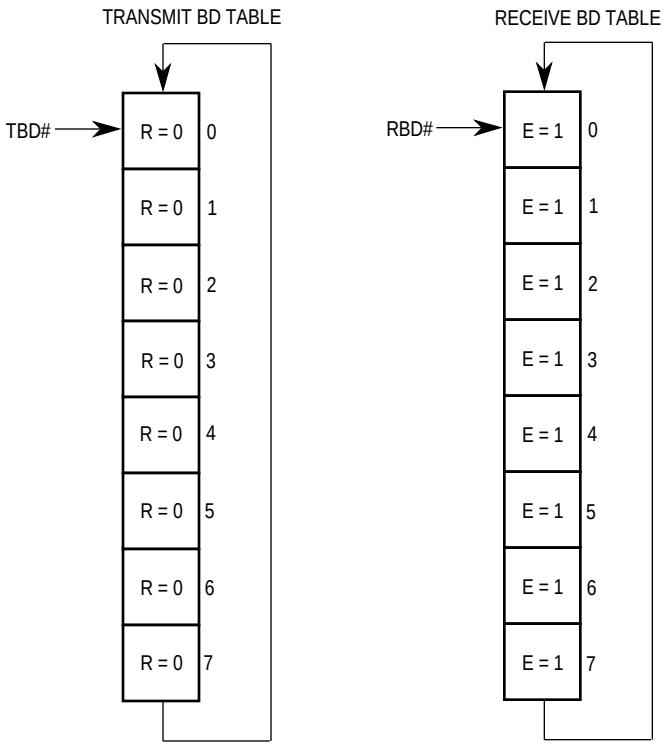


Figure D-4. Transmit and Receive BD Tables

The TBD# and RBD# are pointers used by the SCC to show which BD the SCC is currently using (or which one it will be using next). TBD# and RBD# always sequence around the queue of BDs in a circular fashion and are initialized to the first BD as shown.

D.3.2 MC68302 Buffer Processing

The communications processor (CP) processes the transmit BDs in a straightforward fashion. Once the transmit side of an SCC is enabled, the CP starts with the first BD in that SCC's transmit BD table, periodically checking the ready bit to see if that BD is ready. Once that BD is ready, the CP will process that BD, reading a word at a time from its associated buffer, doing certain required protocol processing on the data, and moving resultant data to the SCC transmit FIFO. When the first buffer has been processed, the CP moves on to the next BD, again waiting for that BD's ready bit to be set. Thus, the CP does no look-ahead BD processing, nor does it skip over BDs that are not ready. When the CP sees the wrap bit set in a BD, it goes back to the beginning of the BD table after processing of this BD is complete. After using a BD, the CP sets its ready bit to not-ready; thus, the CP will not use a BD twice. The BD must be confirmed by the M68000 core before being used again.

The CP uses the receive BDs in a similar fashion. Once the receive side of an SCC is enabled, the CP starts with the first BD in that SCC's receive BD table. Once data arrives from the serial line into the SCC, the CP performs certain required protocol processing on the data and moves the resultant data (either by bytes or by words, depending on the protocol) to the buffer pointed to by the first BD. Use of a BD is complete when there is no more room

in the buffer or when certain events, such as an error or an end-of-frame, are detected. Whatever the reason, the buffer is then said to be closed, and additional data will be stored using the next BD. Whenever the CP needs to begin using a BD because new data is arriving, it will check the empty bit of that BD. If the current BD is not empty, it will report a busy error. However, the CP will not move from the current BD until it becomes empty. When the CP sees the “wrap” bit set in a BD, the CP goes back to the beginning of the BD table after use of this BD is complete. After using a BD, the CP sets its empty bit to not-empty; thus, the CP will never use a BD twice. The BD must be processed by the M68000 core before being used again.

D.3.3 New Pointers

To control the buffering of the SCCs, the three BD pointers to be used by software are defined. Two pointers are used for the transmit BDs, and one pointer is used for the receive BDs (see Figure D-5).

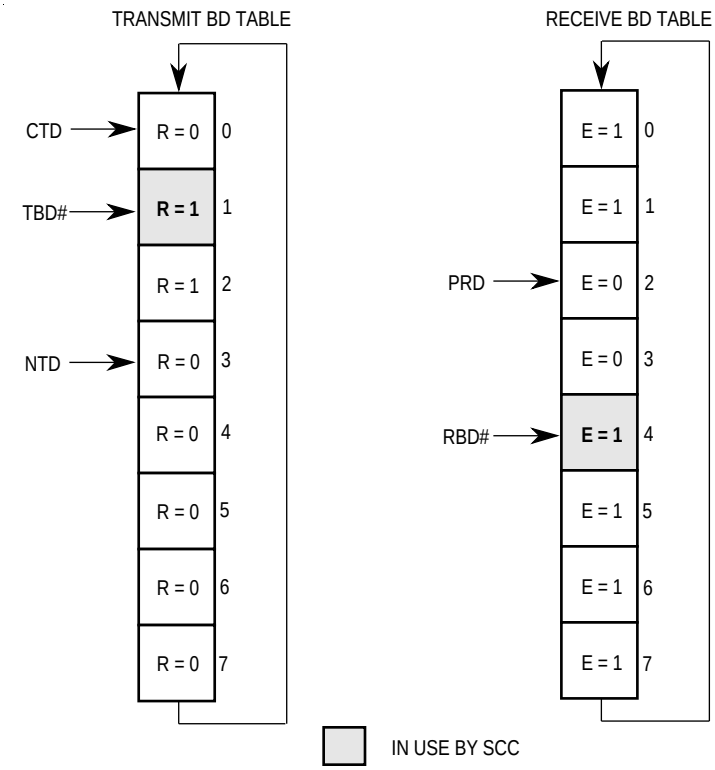


Figure D-5. Pointer during Execution

New transmit data (NTD) shows the next transmit BD that will be receiving data. This is the first pointer to move in the transmit process.

Confirm transmit data (CTD) shows the next transmit BD that will be confirmed. To confirm a buffer, check for errors after transmission and then mark the BD as available for USQ. This is the last pointer to move in the transmit process.

Process receive data (PRD) shows the next receive BD that will be processed. To process a buffer, check for errors, move or use the data, and then mark the BD as available for use. This is the last pointer to move in the receive process.

D.3.4 Initial Conditions

Initially the BDs should be configured like those shown in Figure D-4. All three transmit pointers should point to the first transmit BD in the table, and both receive pointers should point to the first receive BD. Also, the data length fields of all the BDs should be zero for the purposes of the following algorithm.

D.3.5 Transmit Algorithm

Use the following routine to configure new data to be transmitted.

Execute anytime:

1. Make sure that (NTD -> Ready) = 0. If ready = 1, there is no more room to link in buffers to be transmitted, since there are already eight BDs ahead of the SCC.
2. Make sure that the (NTD -> data length) = 0 to prevent overwriting a buffer that has not been confirmed.
3. Link in the desired buffer to this BD.
4. Set up the control information in the BD, but do not set the ready bit.
5. Do not allow the confirm transmit process to interrupt the following steps:

   a. Set the data length to the exact number of bytes (odd or even) that should be transmitted from this buffer.
   b. Set the ready bit of the BD.
   c. Move NTD to point to the next BD. (If the wrap bit is set, point to the first BD.)
6. The confirm process interrupt may now be unmasked.

D.3.6 Interrupt Routine

When an interrupt occurs in the SCC event register perform the following:

1. Read the SCC event register.
2. Clear any unmasked bits that will be dealt with in this interrupt routine (write ones to those bits). Those events normally include receiving a buffer or frame, transmitting a buffer, getting a transmit error, and any change in general status conditions such as carrier detect or clear-to-send.
3. Deal with the general status conditions as desired.

If the interrupt was due to receiving a buffer or frame, perform the following receive process:

1. While (PRD -> Empty) = 0, do the following:
   /* Buffer has been filled if Empty = 0*/

   a. Check for errors in the BD.
   b. The user may or may not want to analyze data in the buffer at this time.
   c. Move data out of buffer or change the BD pointer to a new location.
   d. Clear out the data length field so that it is zero to start with.

e. Clear all status bits to zero (the default condition).
f. Set (PRD -> Empty) = 1.
g. Move PRD to point to the next BD (If the wrap bit is set, point to the first BD).

If the interrupt was due to a buffer being transmitted, perform the following confirm process:

2. While (CTD -> Ready) = 0 and (CTD -> data length) > 0, do the following:
   /* Look at frames that have been transmitted, but not confirmed */

   a. Check for errors in the BD.
   b. Clear all status bits to zero (the default condition).
   c. Clear out the data length field so that it is zero to start with. This procedure allows new data be placed in this transmit buffer by the transmit algorithm.
   d. Move CTD to point to the next BD. (If the wrap bit is set, point to the first BD.)

Finally:

3. Clear the SCC bit in the in-service register (ISR) of the interrupt controller. This is standard procedure.
4. Return from interrupt.

D.3.7 Final Comments

Note that nowhere in the algorithm does RBD# or TBD# need to be read. The empty and ready bits provide all the necessary information.

Whether receiving buffers or confirming buffers, the interrupt routines deal with all the buffers they can before the interrupt routine is exited. This approach is not mandatory, but if not all buffers are dealt with, then some provision needs to be made for getting the work done without waiting for another interrupt.

For frame-oriented protocols such as HDLC, the user may wish to only process frames, not buffers. In this case, the algorithm would be the same, but the interrupt mask would be set for frame interrupts only.

D.3.8 HDLC Code Listing

The following code shows the initialization of the HDLC protocol and the implementation of the buffer processing algorithms just described. HDLC frames are continuously transmitted and received in the SCC loopback mode.

```
************************** BUFFER PROCESSING CODE**************************
************************************EQU TABLE*******************************
* The following three values are application dependent
BASE       EQU       $0700000          ;This is set according to the val in BAR
INIT       EQU       $0030300          ;Initialization Routine
INT_VEC    EQU       $0031000          ;Interrupt Vector for SCC1
* Commonly used Registers and Parameters
BAR        EQU       $0F2              ;Base Address Register
SCR        EQU       $0F4              ;System Control Register
CKCR       EQU       $0F6              ;Clock Control Register
GIMR       EQU       BASE +  $812      ;Global Interrupt Mode Register
IPR        EQU       BASE +  $814      ;Interrupt Pending Register
```

```
IMR       EQU      BASE +  $816      ;Interrupt Mask Register
ISR       EQU      BASE +  $818      ;In-Service Register
SIMODE    EQU      BASE +  $8B4      ;Serial Interface Mode Register
SCON1     EQU      BASE +  $882      ;SCC1 Configuration Register
SCM1      EQU      BASE +  $884      ;SCC1 Mode Register
SCCE1     EQU      BASE +  $888      ;SCC1 Event Register
SCCM 1    EQU      BASE +  $88A      ;SCC1 Mask Register
EN_SCC    EQU      $0C               ;ENR and ENT bits in SCM

***SCC1 Parameter Table ***
ST_BD     EQU      0                 ;Status and Control in BD
SS_BD     EQU      1                 ;Status in BD
LN_BD     EQU      2                 ;Data Length in BD
PT_BD     EQU      4                 ;Buffer pointer in BD
SZ_BD     EQU      $08               ;Size of BD = 8 bytes
FCR_1     EQU      BASE+$0480        ;RFCR and TFCR for SCC1
MRBLR_1   EQU      BASE+$0482        ;Max Rx Buffer Length
RXBD_01   EQU      BASE+$0400        ;RX BD 0 in SCC1
TXBD_01   EQU      BASE+$0440        ;TX BD 0 in SCC1
READY     EQU      $07               ;Ready bit in the 1st byte of TX BD
EMPTY     EQU      $07               ;Empty bit in the 1st byte of RX BD
WRAP      EQU      $05               ;Wrap bit in the 1st byte of BD
* The following values are application dependent for this example
RXBF_01   EQU      $030000           ;Address of the first RX buffer
TXBF_01   EQU      $030080           ;Address of the first TX buffer
BD_CNT    EQU      $08               ;Number of BDs used
SZ_BF     EQU      $10               ;Size of buffer =16 bytes
N_DATA    EQU      6                 ;Number of data to be sent in a buffer
* SCC1 HDLC Parameters
CMSKL_1   EQU      BASE+$04A0        ;CRC Mask Low
CMSKH_1   EQU      BASE+$04A2        ;CRC Mask High
DISFC_1   EQU      BASE+$04A8        ;Discard Frame Counter
CRCEC_1   EQU      BASE+$04AA        ;CRC Error Counter
ABTSC_1   EQU      BASE+$04AC        ;Abort Sequence Counter
NMARC _ 1 EQU      BASE+$04AE        ;Nonmatching Address Receive Counter
RETRC_1   EQU      BASE+$04B0        ;Frame Retransmission Counter
MFLR_1    EQU      BASE+$04B2        ;Max Frame Length Register
HMASK 1   EQU      BASE+$04B6        ;User-Defined Frame Address Mask

********************** Typical M68302 Initialization Code********************
* Register Initialized values in ADS board before execution
* USP = 00080000 ISP = 000040000 (Stack pointer not used)
          ORG      INIT              ;PC = 00030300
          MOVE.W   #$2700,SR         ;SR = 2700, mask off interrupts
* Set Base Address = $700000
* Now all 68302 on-chip peripherals begin at address $700xxx
          MOVE.W   #$0700,BAR        ;BAR = 0700
* Set System Control Register
          MOVE.L   #0,SCR            ;Nothing special for this example
***Setups for interrupt ***
          MOVE.W   #$0A0,GIMR        ;Normal mode, v7–v5 = 5
          MOVE.W   #0,IMR            ;Mask off all for now
          MOVE.W   #$FFFF,IPR        ;Clear IPR

*** Set up Serial Interface Connection ***
```

```
* Set up PACNT, PBCNT, etc., ignore for this example, only SCC1 is used
* Select Serial Interface Mode: normal operation, NMSI mode
          MOVE.W    #0,SIMODE          ;Same as default after reset

*** SCC1 Initialization ***
*  Interrupt Vector: SCC1 interrupt handler is at INT_VEC = $31000
*  v7–v5 =5, v4–v0 = $ad -> vector = $ad -> Exception vector = ($ad<<2) = $2b4
          MOVE.L    #INT_VEC, $02B4

* Determine Configuration
* Use Baud Rate Generator for transmit and receive, Rate is 130 kbps.
          MOVE.W    #$07E,SCON1

* Select SCC Mode
* HDLC, Loopback mode, CRC16, RTS negate between frames, NRZ mode.
          MOVE.W    #$10,SCM1

* Set up Parameter RAM
          MOVE.W    #0,FCR_1           ; Clear RFCR and TFCR
          MOVE.W    #$08,MRBLR_1       ; Max Buffer Length = 8
          MOVE.W    #$F0B8,CMSKL_1     ; 16 bit CRC
          MOVE.W    #$070,MFLR_1       ; Max Frame Length = $70 bytes
          MOVE.W    #0,HMASK_1         ; Do not check address
          MOVE.W    #0,DISFC_1         ; Clear the counter
          MOVE.W    #0,CRCEC_1         ; Clear the counter
          MOVE.W    #0,ABTSC 1         ; Clear the counter
          MOVE.W    #0,NMARC_1         ; Clear the counter
          MOVE.W    #0,RETRC_1         ; Clear the counter

* Clear Event Register
          MOVE.B    #$FF,SCCE1

* Determine Maskable Interrupt Events by setting SCCM
* Allow the following interrupt: TXE, RFX, TXB, and RFB
          MOVE.B    #$1B,SCCM1

* Clear M68000 data registers
          CLR.L     D0
          CLR.L     D1
          CLR.L     D2
          CLR.L     D3
          CLR.L     D4
          CLR.L     D5

***Prepare Buffer Descriptors ***
*SCC1 Rx Buffer Descriptors Initialization values before execution:
*00700400 D000 0000 0003 0000 D000 0000 0003 0010
*00700410 D000 0000 0003 0020 D000 0000 0003 0030
*00700420 D000 0000 0003 0040 D000 0000 0003 0050
*00700430 D000 0000 0003 0060 F000 0000 0003 0070
          LEA.L     RXBD_01,A0         ;A0 points to the first RXBD of SCC1
          LEA.L     RXBF_01,A1         ;A1 points to the first buffer
          MOVE.W    #$D000,D1          ;D1 is used for setting the status of BD
*                                      ;Empty = 1, External = 1, Int=1
          MOVE.W    #$F000,D2          ;D2 is for the last BD, Wrap = 1
```

```
            MOVE.B    #BD_CNT,D3        ;# of BD used = 8
            SUBQ.B    #2,D3
Set RXBD    MOVE.W    D1,ST_BD(A0)      ;Set Control and Status Bits
            MOVE.W    #0,LN_BD(A0)      ;Length = 0
            MOVE.L    A1,PT BD(A0)      ;Set buffer pointer
            ADDQ.L    #SZ_BD,A0         ;Next BD
            ADDA.L    #SZ_BF,A1         ;Next BF
            DBRA      D3,SetRXBD

* Set the last BD
            MOVE.W    D2,ST_BD(A0)      ;Wrap = 1
            MOVE.W    #0,LN_BD(A0)      ;Length = 0
            MOVE.L    A1,PT_BD(A0)      ;Buffer Pointer

*SCC1 Tx Buffer Descriptors Initialization values before execution
*00700440 5C00 0000 0003 0080 5C00 0000 0003 0090
*00700450 5C00 0000 0003 00A0 5C00 0000 0003 00B0
*00700460 5C00 0000 0003 00C0 5C00 0000 0003 00D0
*00700470 5C00 0000 0003 00E0 7C00 0000 0003 00F0

            LEA.L     TXBD_01,A0        ;A0 points to the first TXBD of SCC1
            LEA.L     TXBF_01,A1        ;A1 points to the first buffer
            MOVE.W    #$5C00, D1        ;D1 is used for setting the status of BD
*                                       ;Ready = 0, External = 1, Interrupt = 1,
*                                       ;Last = 1, TxCRC = 1
            MOVE.W    #$7C00,D2         ;D2 is for the last BD, Wrap = 1
            MOVE.L    #BD_CNT,D3        ;# of BD used = 8
            SUBQ.B    #$2,D3            ;Count from 6 to 0
SetTXBD     MOVE.W    D1,ST_BD(A0)      ;Set Control and Status Bits
            MOVE.W    #0,LN_BD(A0)      ;Length = 0
            MOVE.L    A1,PT_BD(A0)      ;Set buffer pointer
            ADDQ.L    #SZ_BD,A0         ;Next BD
            ADDA.L    #SZ_BF,A1         ;Next BF
            DBRA      D3,SetTXBD




*Set the last BD
            MOVE.W    D2,ST_BD(A0)      ;Wrap = 1
            MOVE.W    #0,LN_BD(A0)      ;Length = 0
            MOVE.L    A1,PT_BD(A0)      ;Buffer Pointer

***Prepare Tx Buffers: In this example each frame fits into one buffer ***
*00030080 0001 0203 0405 0607 0809 0A0B 0C0D 0E0F
*00030090 0001 0203 0405 0607 0809 0A0B 0C0D 0E0F
* . . .
*000300F0 0001 0203 0405 0607 0809 0A0B 0C0D 0E0F
            LEA.L     TXBF_01,A0        ;A1 points to the first buffer
            MOVE.L    #SZ_BD,D1         ;D1 is used to count the BD
            SUBO.B    #1,D1
NxtBF       CLR.L     D2                ;D2 is used as content and counter of BF
NxtBT       MOVE.B    D2,(A0)+
            ADDQ.B    #$1,D2
            CMPI.B    #SZ_BF,D2         ;Number of data in a buffer is 16
```

```
          BLT       NxtBT             ;Next Byte
          DBRA      D1,NxtBF          ;Next Buffer

***Now ready to go ***
***Set IMR and Enable SCC1 ***
          MOVE.W    #$2000,1MR        ;Allow SCC1 interrupt only
          MOVE.W    #$2000,SR         ;Unmask interrupts
          ORI.W     #EN_SCC,SCM1      ;ENT = ENR = 1
          JMP       MAIN              ;Go to Main routine for Tx and Rx

*****************************Main Routine ******************************
* Set up BD pointers
MAIN      LEA.L     TXBD_01,A1        ;A1 = CTD pointer
          LEA.L     TXBD_01,A2        ;A2 = NTD pointer
          LEA.L     RXBD_01,A3        ;A3= PRD pointer

* The following is an infinite loop that prepares data to be sent
* when a Tx BD is available to be used.
          CLR.L      D3               ;D3 is used to count Tx frames
*                                       transmitted in the loop
TxReady   BTST.B    #READY,(A2)       ;Test Ready Bit
          BNE.B     TxReady           ;If Ready = 0, the BD has been sent
Confirm   CMPI.W    #$0,LN_BD(A2)     ;test NTD ->data length
          BNE.B     Confirm           ;If  length = 0, the BD has been
*                                       confirmed

*Set TXBD if it is to be changed, e.g.,
*          ORI.W    #$5C00,ST_BD(A2)



* Mask off interrupt for the following operations
          MOVE.W    #$2700,SR
          MOVE.W    #N_DATA,LN_BD(A2) ;Set data length to 6
          BSET.B    #READY,ST_BD(A2)  ;Set Ready bit
          ADDO.L    #1,D3             ;Inc Tx frame count
          BTST.B    #WRAP,ST_BD(A2)   ;Test Wrap bit
          BNE.B     Wrapit            ;If Wrap . 1, wrap a
          ADDQ.W    #SZ_BD,A2         ;Move NTD to next BD
          BRA.B     Unmask
Wrapit    LEA.L     TXBD _ 01,A2      ;Wrap back to the first TX BD
Umask     MOVE.W    #$2000,SR         ;Unmask interrupt
          JMP       TxReady

*******************SCC1 interrupt handler *******************
          ORG       INT_VEC           ;Interrupt Vector for SCC1
* Check events:  Handle RX then TX then Errors
          CLR.L     D1                ;Clear D1
          MOVE.B    SCCE1,D1          ;SCCE1 =>, D1
          MOVE.L    D1,D2             ;SCCE1 =>, D2
          ANDI.W    #9,D2             ;Are RXF or RXB set?
          CMPI.W    #0,D2             ;If they are set
          BNE.B     RX_INT            ;Handle receiver's interrupt
CK_TX     MOVE.L    D1,D2             ;SCCE1 =>, D2
          ANDI.W    #$12,D2           ;Are TXF or TXB set?
```

```
          CMPI.W    #0,D2             ;If they are set
          BNE.B     TX_ INT           ;Handle Transmitter's interrupt
* The handling of other events, e.g., CTS, CD IDL, BSY, is left  to
* the users as desired.
OthrlNT   MOVE.W #$2000,1SR           ;Clear SCC1 bit in ISR
          RTE

***************Receiver portion of SCC1 interrupt routine***************
* This routine handles received (nonempty) BD: set data length = 0,
* clear status bits, set empty =1, and update PRD
* Clear the identified events as soon as possible, so no lost
* events occur during the interrupt handling
RX_INT    MOVE.B    #9,SCCE1          ;Clear RXF and RXB in SCCE1
* While Not-Empty continue to process the next Rx BD, Else Exit.
NxtPRD    BTST.B    #EMPTY,ST_BD(A3)  ;Test PRD -, Empty Bit
*         BNE.B     EXIT_RX           ;Don't need to process if the
*                                     ;Rx BD is still empty.
*** Check status in RXBD for erratic events ***
* If status bits are all 0 then continue, else SHUTDOWN the receiving
* process. This in turn shuts down the whole program, since all of
* Rx BDs will soon be unavailable (all BDs Empty = 8). Thus, the
* status of this BD will be saved for examination later.
          CMPI.B    #0,SS_BD(A3)      ;Check status bits
          BNE.B     EXIT_RX
* Status bits are all 0
          CLR.W     LN_BD(A3)         ;data length = 0
          CLR.B     SS_BD(A3)         ;Clear out all status bits
          BSET.B    #EMPTY,ST_BD(A3)  ;Empty = 1
          BTST.B    #WRAP,ST_BD(A3)   ;Test Wrap bit
          BNE.B     Wrap_R
          ADDQ.W    #SZ_BD,A3         ;Increment PRD to next BD
          BRA.B     NxtPRD            ;Back to while loop
Wrap_R    LEA.L     RXBD_01,A3        ;Wrap back to the first Rx BD
          BRA.B     NxtPRD            ;Back to the while loop
EXIT_RX   JMP       CK_TX             ;Exit receiver potion of the handler

***************Conflrmer portion of SCC1 Interrupt routine ***************
* This routine handles transmitted (Not-ready) BD: set data length = 0,
* clear status bits, set ready = 1, and update CTD.
* Same as the Rx Interrupt handler, the first thing to do is to
* clear the identified events
TX_INT    MOVE.B    #$1 2,SCCE1       ;Clear TXF and TXB in SCCE1
* While Not-Ready continue to process the next Rx BD, Else Exit.
* The Ready bit should be cleared by the CP
NxtCTD    BTST.B    #READY,ST BD(A1)  ;Test PRD-> Ready Bit
          BNE.B     EXIT_TX           ;Don't need to process if the
*                                     ;Tx BD is Ready.
* Check data length, length must be > 0 to continue the confirming process
          CMPI.W    #0,LN_BD(A1)      ;Test CTD –> data length
          BEQ.B     EXIT_TX
          CLR.W     LN_BD(A1)         ;data length = 0

***Check status in TXBD for erratic events ***
* If status bits are all 0 then continue, else SHUTDOWN the confirming
* process. This in turn shuts down the whole program, since soon
```

```
* none of the Rx BDs will be available (all BDs Empty = 0). Thus, the
* status of this BD will be saved for examination later.
          CMPI.B    #0,SS_BD(A1)      ;Check status bits
          BNE.B     EXIT_TX
* Status bits are all 0
          CLR.B     SS_BD(A1)         ;Clear out all status bits
          BTST.B    #WRAP,ST_BD(A1)   ;Test Wrap bit
          BNE.B     Wrap_T            ;
          ADDQ.W    #SZ_BD,A1         ;Increment CTD to next BD
          BRA.B     NxtCTD            ;Back to while loop
Wrap_T    LEA.L     TXBD_01,A1        ;Wrap back to the first Tx BD
          BRA.B     NxtCTD            ;Back to the while loop
EXIT_TX   JMP       OthrlNT           ;Exit confirmer potion of the handler
* Back to the main handler, that handles the rest of the events

********************Data after transmission ********************
* 00030000 0001 0203 0405 141A xxxx xxxx xxxx xxxx # Receiver Buffer 0
* 00030010 0001 0203 0405 141A xxxx xxxx xxxx xxxx # Receiver Buffer 1
* 00030020 0001 0203 0405 141A xxxx xxxx xxxx xxxx # Receiver Buffer 2
* 00030030 0001 0203 0405 141A xxxx xxxx xxxx xxxx # Receiver Buffer 3
* 00030040 0001 0203 0405 141A xxxx xxxx xxxx xxxx # Receiver Buffer 4
* 00030050 0001 0203 0405 141A xxxx xxxx xxxx xxxx # Receiver Buffer 5
* 00030060 0001 0203 0405 141A xxxx xxxx xxxx xxxx # Receiver Buffer 6
* 00030070 0001 0203 0405 141A xxxx xxxx xxxx xxxx # Receiver Buffer 7
• 00030080 0001 0203 0405 0607 0809 0A0B 0C0D 0E0F # Transmit Buffer
• Notice that 141A is the 16-bit CRC
END
```

D.4 CONFIGURING A UART ON THE MC68302

The following paragraphs discuss a working example of software that configures the MC68302 SCC3 for the UART mode. The code receives data from the UART receiver on a character-by-character basis and retransmits it out of the UART transmitter. The code, which runs as is on the ADS302 board, is an excellent starting point for understanding how to program the UART mode or to create a simple UART handler to support a debug monitor.

D.4.1 Purpose of the Code

This code is really a character “echo” generator — every character received is retransmitted under control of the M68000 core on the MC68302. An automatic echo mode is available with each SCC in the SCM register. Why do it manually in software? The answer is that developing this concept in software is a good way to begin a more complete design since it demonstrates reception, transmission, and some real-time issues.

The code receives and transmits data on a character-by-character basis, with interrupts generated on each character received. Although the MC68302 UART mode has much more flexibility and power than what is used by this code, many applications just require a simple low-speed UART channel for debugging during the design. SCC3 is often chosen to run at 9600 baud for this purpose as shown in the example.

The code is shown in standard M68000 assembler. Efficiency was sacrificed, when necessary, to enhance readability. The code is comprised of three basic parts. The first part initializes the MC68302 with everything required to set up SCC3 for the UART mode. The

initialization corresponds to the recommended order described in 4.5.7 SCC Initialization. The second part is a set of loops waiting for data to arrive to be retransmitted out of the SCC3. The third part is the SCC3 receive interrupt handler. Transmit interrupts are masked in this example.

D.4.2 Organization of Buffers

In the MC68302, there is no such thing as an receive register (Rx) or transmit register (Tx). Rather, a flexible structure called a buffer descriptor (BD) is used. In this example, two Rx BDs and two Tx BDs are used. Each BD is set up to point to a one-byte location for data. Thus, the receiver and transmitter are double-buffered. The number of Rx or Tx BDs can be changed simply by changing the number of BDs initialized in the code (and two other lines documented in the code). However, using at least two BDs has advantages as noted in the following paragraphs.

The structure of the buffers is shown in Figure D-6.

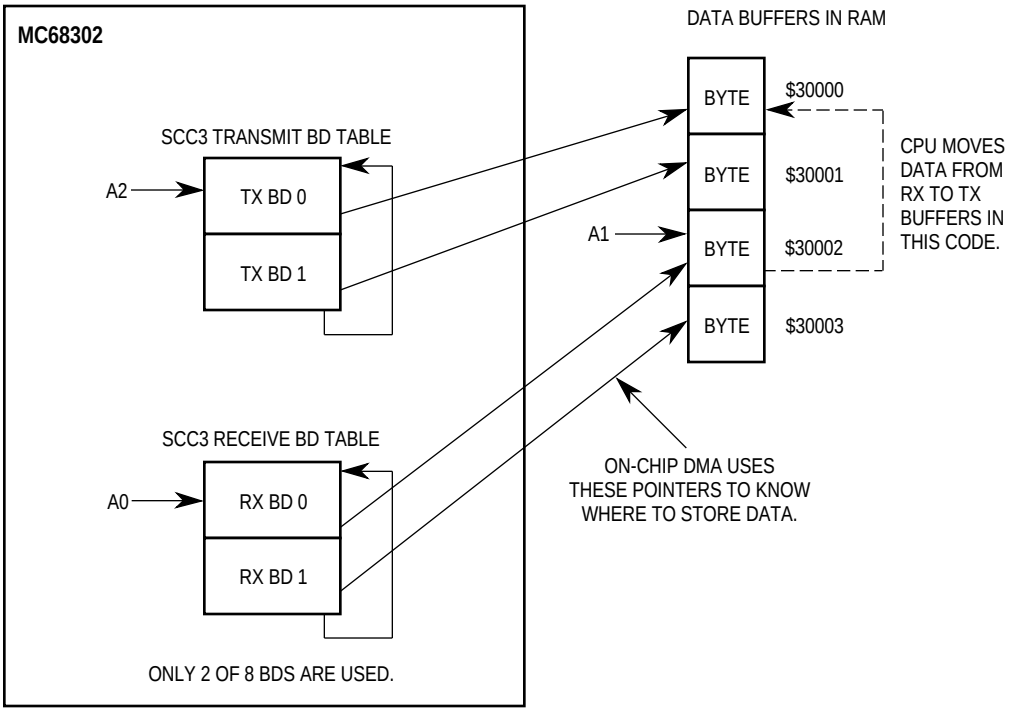


Figure D-6. Transmit and Receive BD Tables and Buffers

To make the application more general, the data buffers were located in external RAM; however, internal RAM could have been used. Each BD points to just one byte in memory as shown. Note that the data buffers do not need to be consecutive as shown.

From one to eight BDs may be used for both the transmit and receive operation. Use of eight BDs for the transmit side of SCC3 requires that the SCP and SMCs not be used. As data rates increase substantially beyond the 9600 baud of this example, the use of more BDs and more data bytes per BD becomes justified. Why were two Rx BDs and two Tx BDs chosen rather than just one each?

Reason 1: When doing character-by-character handling of a software echo operation, the use of at least three BDs (i.e., really a three-byte software FIFO) takes care of the latency between recognizing a character has come in and getting that character transmitted out as other characters continue to come in. Use of four BDs in total gives plenty of software FIFO margin.

Reason 2: Using two Tx BDs means that in a full-load situation, the next BD will always have its ready bit set when the RISC controller checks it, meaning that there will never be any added delay in getting characters transmitted out of the UART.

Reason 3: Using two Rx BDs gives extra time for the software to handle the interrupt for one character while the next character is being received into the next buffer, and gives this example the opportunity to show how to step through the BDs in software.

D.4.3 Assumptions about the System

The code, which was run on the ADS302 board, assumes that the MC68302 peripherals are placed at the default position of $700000 (i.e., BAR is written with $700). It also assumes that SCC3 is used. Either of the above assumptions can be modified if desired. The code was assembled with a simple freeware M68000 assembler called X68000.

D.4.4 UART Features Not Discussed

The following UART capabilities were not discussed in this example: fractional stop bit transmission, inserting flow control characters into the transmit data stream, recognizing special control characters, using $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ to control transmission and reception, use of an external clock, use of multiple bytes per buffer, sending breaks, sending idles before messages, recognizing addresses, freezing transmission, idle timeout, and others. Refer to 4.5.11 UART Controller for more details.

D.4.5 UART Code Listing

```
* SIMPLE UART “ECHO” CODE
* Register initialization values before execution:
*
* PC = 00030300 SR = 2700
* USP = 00080000 *ISP = 00004000 (Stack pointers not used)
*
* D1 = SCCE3 holding register
* D3 = count of characters received
* D4 = count of characters transmitted
* D5 = count of BSY conditions occurring (no receive buffers available)
* D6 = 1 or greater, if character(s) waiting to be transmitted, 0 otherwise
* A0 = current Rx BD pointer
* A1 = current “next TX byte” to send pointer
* A2 = current Tx BD pointer
* A3 = temp
* SCC3 Tx Buffer Descriptors initialization:
* 00700640 5000 0000 0003 0000
* 00700648 7000 0000 0003 0001 (wrap bit set)
* SCC3 Rx Buffer Descriptors initialization:
* 00700600 d000 0000 0003 0002
```

```
* 00700608 1000 0000 0003 0003 (wrap bit set)


BAR             EQU       $0F2
GIMR            EQU       $700812
IPR             EQU       $700814
IMR             EQU       $700816
ISR             EQU       $700818
PACNT           EQU       $70081e
SIMODE          EQU       $7008b4
SCON3           EQU       $7008a2
SCM3            EQU       $700&4
SCCE3           EQU       $7008a8
SCCM3           EQU       $7008aa

*SCC3 Initialization Code
          ORG      $30300
          MOVE.W #$700.BAR           ;BAR = 0700
* Base Address - S700000, so ALL MC68302 on-chip peripherals begin at
* address S700xxx.

          MOVE.W   #$00A0, GIMR      ;GIMR = 00a0
          MOVE.W   #$FFFF,IPR        ;clear IPR
          MOVE.L   #$30500,$2a0      ;SCC3 vector initialization
          MOVE.W   #$0300,PACNT      ;PACNT = 0300
* Causes the SCC3 TXD3 and RXD3 pins to be enabled. TCLK3 and RCLK3
* pins are left as parallel l/O pins.
          MOVE.W   #50,SIMODE        ;SIMODE = 0000 (its reset value)
* SCC3 is set up for NMSI (i.e. modem) operation. No multiplexed
* modes are used on the other SCCs.
          MOVE.W   #$00d8,SCON3      ;SCON3=00d8 for ~9600 baud at 16.67 MHz
* Baud Rate generator is used for transmit and receive. Rate is 9556bps.
          MOVE.W   #$171,SCM3        ;SCM3 = 0171
* No parity. Normal UART operation. 8-bit characters. 2 Stop bits.
* The CD and CTS lines not used to enable reception and transmission,
* but do cause a status change in the SCC3 Event register.
          MOVE.L  #$50000000,S700640 ;Set up Tx BD 0 Status and Count
          MOVE.L  #$30000,$700644    ;Set up Tx BD 0 Buffer Address
          MOVE.L  #$70000000,$700648 ;Set up Tx BD 1 Status and Count
          MOVE.L  #$30001,$70064c    ;Set up Tx BD 1 Buffer Address
* Set up 2 Tx BDs
          MOVE.L  #$d0000000,$700600 ;Set up Rx BD 0 Status and Count
          MOVE.L  #$30002,$700604    ;Set up Rx BD 0 Buffer Address
          MOVE.L  #$f0000000,$700608 ;Set up Rx BD 1 Status and Count
          MOVE.L  #$30003,$70060c    ;Set up Rx BD 1 Buffer Address
* Set up 2 Rx BDs
          MOVE.W  #$0,$700680        ;clear RFCR/TFCR (Function code setup)
* Must be initialized to a value other than 7, or won't work with chip selects
          MOVE.W  #$1,$700682        ;MRBLR = 0001 (one-byte receive
                                      buffers)
* This combined with the “l” bit set in the Rx BD, gives interrupts on each
* character received.

          MOVE.W #$4,$70069c         ;MAX_IDL don't care since MRBLR =1.
* Normally set to a small value, it closes a receive buffer if a certain number
```

```
* of idles are received without a new * character. Note that 0 is the maximum
* value. In this case the buffer will always be closed after 1 character, so
* MAX_IDL is a don't care.
          MOVE.W  #$1,$7006A0         ;BRKCR = 1. Only one break char. sent if
* STOP TRANSMIT command executed. The STOP TRANSMIT command is not used in
* this code.
          MOVE.W  #$0,$7006A2         ;PAREC = 0000
          MOVE.W  #$0,$7006A4         ;FRMEC = 0000
          MOVE.W  #$0,$7006A6         ;NOSEC = 0000
          MOVE.W  #$0,S7006A8         ;BRKEC = 0000
* Initialize counters to zero
          MOVE.W  #$0,$7006aa         ;UADDR1 = 0000
          MOVE.W  #$0,$7006ac         ;UADDR2 = 0000
* UART Address characters not used in normal operation.
          MOVE.W  #$8000,$7006B0      ;CHARACTER1 = 8000
* Initialize CHARACTER8
          MOVE.W  #$0000,$7006BE      ;CHARACTER8 = 0000
* No control characters are initialized to cause special interrupts.
* The flow-control facility (XON-XOFF) is not used.
          MOVE.B    #$FF,SCCE3        ;clear SCCE3
          MOVE.B    #$15,SCCM3        ;SCCM3 = 15
*Interrupts in SCCM3 are allowed only for BRK (break character
*received), BSY (no receive buffer available), and RX (buffer
* received). Note that buffers are 1 character in this application.
          MOVE.W    #$0100,1MR        ;IMR=0100. Allow SCC3 interrupts only.
          MOVE.W    #$17d,SCM3        ;SCM3 = 017d
* Set ENR and ENT bib in SCM3.
          CLR.L     D0                ;clear all used data registers
          CLR.L     D1
          CLR.L     D3
          CLR.L     D4
          CLR.L     D5
          CLR.L     D6
          MOVEA.L   #$700600,A0       ;Load A0 as current Rx BD pointer
          MOVEA.L   #$30002,A1        ;Load A1 as current “next Tx Byte” ptr
          MOVEA.L   #$700640,A2       ;Load A2 as current Tx BD pointer
          MOVE.W    #$2000,SR         ;Enable interrupts. Stay in spvr mode.
*************************************************************************
* Transmit Code
* OUTERLOOP
OUTLOOP   CMPI.B    #$0,D6            ; Something to send (is D6> =1?)
          BEQ.B     OUTLOOP           ; Stay in outerloop if 0
          SUBQ.B    #$1,D6            ; Decrement send status by 1 (0 = empty)
* INNERLOOP:
INLOOP    BTST.B    #$07,(A2)         ; Test Ready bit of Tx BD
          BNE.B     INLOOP            ; Fall thru if 0, else wait in innerloop
* The following sets up and sends out the transmit buffer
          ADDO.W    #$1,D4            ; Increment number of chars transmitted
          MOVEA.L   A2,A3             ; A3 will be used to find Tx data buffer
          ADDQ.W    #$4,A3            ; Inc A3 to point to Tx data pointer
          MOVEA.L   (A3),A3           ; A3 now points to Tx Data buffer
          MOVE.B    (A1),(A3)         ; Move char from Rx Buffer to Tx Buffer
          ADDQ.W    #2,A2             ; Increment A2 to point to byte count
          MOVE.W    #$1,(A2)          ; Set TxBD Byte count to 1
          SUBO.W    #2,A2             ; A2 now points to beginning of Tx BD
```

```
          BSET.B    #$7,(A2)          ;Set Ready bit of Tx BD
* The Tx BD sand data status is not checked since the only one is CTS lost,
* which is not applicable, since CTS is ignored in this application.
* The following updates A2 to point to the next Tx BD
          BTST.B    #$05,(A2)         ;test Wrap bit
          BNE.B     REINIT2           ;If set, reinit A2 to 700640
          ADDA.W    #$08,A2           ;else inc A2 by 8 to next Tx BD
          BRA.B     CONT              ;Jump to Continue on
REINIT2 MOVEA.L     #$700640,A2       ;Reinitialize A2
* Determine what the next byte to "echo" will be and then go to OUTERLOOP
 CONT     ADDO.W    #$1,A1            ;Increment A1 to next byte to send
          CMPA.L    #$30004,A1        ;Is A1 = 30004? ***
          BEQ.B     NEWA1             ;If so, go to NEWA1
          BRA.B     OUTLOOP           ;Jump back to outerloop and wait
NEWA1     SUBO.W    #$02.A1           ;Set A1 back to 30002 ?***
          BRA.B     OUTLOOP           ;Jump back to outerloop and wait
* The two lines with ***  above are dependent on the number of Rx BDs used.
* If the number is increased, these values should be increased by the same
* amount. These are the only lines dependent on the Rx BD or Tx BD setup.
*************************************************************************
*SCC3 Interrupt Routine
          ORG       $30500
          CLR.L     D1                ;clear D1
          MOVE.B    SCCE3,D1          ;Move SCCE3 status to D1
          MOVE.B    #$15,SCCE3        ;Clear only BRK. BSY and RX in SCCE3.

          BTST.B    #$2.D1            ;Is BSY set?
          BNE.B     BUSY              ;Jump to BUSY handler if set


* Test Break:
BRKTEST   BTST.B    #$4,D1            ;Is BRK set?
          BNE.B     BREAK             ;Jump to BREAK handler if set
*Test Receive:
RECTEST   BTST.B    #$0,D1            ;Is RX set?
          BNE.B     RECEIVE           ;Jump to RECEIVE handler if set
          JMP       ALMDONE           ;Jump to About Done (impossible)
* Busy handler:
BUSY      ADDQ.B    #1,D5             ;Inc Busy counter (no receive buffers)
          BSET.B    #$F,(A0)          ;set Empty bit of current Rx BD
          JMP       BRKTEST           ;Jump to test for BREAK

*Break handler:
BREAK     NOP                         ;This code ignores received breaks
* The UART BRKEC will record the number of breaks received
          JMP       RECTEST           ;Jump to test for RECEIVE
*Receive handler:
RECEIVE   ADDQ.W    #1,D3             ;Increment number of chars received
          ADDQ.B    #1,D6             ;D6 inc by 1 (character ready to send)

          ADDQ.W    #1,A0             ;Inc A0 to point to Rx BD byte status
          CMPI.B    #$0,(A0)          ;Does status = 00?
          BNE.B     BSTAT             ;Jump to Bad Status it not 00
INCPTR    SUBQ.W    #1,A0             ;Dec A0 to point to beginning of Rx BD
          ANDI.W    #$FF00,(A0)       ;Clear out Rx BD status
```

```
          BSET.B    #$07,(A0)         ;Set Empty bit of Rx BD
          BTST.B    #$05,(A0)         ;test Wrap bit
          BNE.B     REINIT0           ;It set, reinit A0 to 700600
          ADDA.W    #$08,A0           ;else inc A0 by 8 to next Rx BD
          BRA.B     ALMDONE           ;Jump to Almost Done
REINIT0 MOVEA.L     #$700600,A0       ;Reinitialize A0
          BRA.B     ALMDONE           ;Jump to almost Done
*Bad Status:
BSTAT     NOP                         ;Bad status handler would go here
*Note that the UART FRMEC, NOSEC, and PAREC counters record bad status.
          BRA.B     INCPTR            ;Jump back to Receive handler
* Almost Done:
ALMDONE   MOVE.W    #$0100,1SR        ;Clear SCC3 bit in the ISR
          RTE                         ;end of interrupt handler

END
```

D.5 INDEPENDENT DMA IN THE MC68302

Moving of data between a high-speed peripheral controller and memory is optimized when a direct memory access (DMA) controller is used. The MC68302 contains an independent direct memory access (IDMA) controller. Engineers developing system architectures requiring both the M68000 microprocessor and DMA can use the MC68302 to obtain both building blocks in one package.

The registers associated with the programming of the IDMA are as follows: six IDMA registers, four interrupt controller registers, and one parallel l/O register (see Table D-1).

Table D-1. IDMA Registers

Acronym	Register	Address
CMR	Channel Mode Register	Base + $802
SAPR	Source Address Pointer Register	Base + $804
DAPR	Destination Address Pointer Register	Base + $808
BCR	Byte Count Register	Base + $80c
CSR	Channel Status Register	Base + $80e
FCR	Function Code Register	Base + $810
GIMR	Global Interrupt Mode Register	Base + $812
IPR	Interrupt Pending Register	Base + $814
IMR	Interrupt Mask Register	Base + $816
ISR	Interrupt In-Service Register	Base + $818
PACNT	Port A Control Register	Base + $81e

D.5.1 IDMA Overview

The IDMA can transfer data between any combination of memory and l/O, in either byte or word quantities with even or odd source and destination addresses. Each IDMA transfer requires two fundamental operations: reading data from a source address and writing data to a destination address. A pointer to the source data is contained in the source address point-

er register (SAPR). The pointer to the destination is located in the destination address pointer register (DAPR). The byte count register (BCR) specifies the number of bytes to be transferred and is decremented once for each byte transferred. Note that the six SDMA channels on the MC68302 have a higher priority than the IDMA (unless the IDMA is performing a maximum rate burst).

D.5.2 IDMA Software Initialization

Information that describes the data block to be moved, the transfer methods, and the control options is loaded into the channel mode register (CMR). See Table D-2 for these required selections. Loading the SAPR, DAPR, and BCR are also part of the initialization procedure. If the functions of the $\overline{\text{DREQ}}$, $\overline{\text{DACK}}$, and $\overline{\text{DONE}}$ pins are required, then their corresponding bits must be enabled in the port A control register (PACNT).

D.5.3 IDMA Bus Arbitration Signals

Whenever the IDMA wants to transfer data, it must arbitrate for the external bus. Two internal signals are used: IDMA bus request ($\overline{\text{IDBR}}$) and IDMA bus grant ($\overline{\text{IDBG}}$). When no external resource controls the bus ($\overline{\text{BGACK}}$ is negated) nor any internal resource (such as the SCCs) requires the use of the bus, then the IDMA can be granted the bus when it asserts (internally) $\overline{\text{IDBR}}$. Obtaining bus mastership occurs when 1) $\overline{\text{AS}}$ and $\overline{\text{BGACK}}$ are high (negated) and 2) external $\overline{\text{BR}}$ and $\overline{\text{BG}}$ are not indicating an external master wants the bus. To indicate that the IDMA has taken control of the bus, $\overline{\text{BGACK}}$ is asserted while $\overline{\text{BR}}$ and $\overline{\text{BG}}$ are not asserted. The IDMA function code bits in the function code register (FCR) can be further used to distinguish between the source and destination accesses.

D.5.4 Triggering External IDMA Transfers

The data request ($\overline{\text{DREQ}}$) input signal should be asserted by a peripheral when it requires service. The MC68302 responds with data acknowledge ($\overline{\text{DACK}}$) output signal which asserts when the IDMA is moving data from/to the peripheral device. A third signal, $\overline{\text{DONE}}$, indicates when the last DMA transfer cycle is in progress. The bidirectional $\overline{\text{DONE}}$ signal is asserted by the IDMA when the BCR has decremented to zero or can be asserted by the peripheral to terminate the data transfer.

D.5.5 Performing Internally Generated IDMA Transfers

The procedure to move blocks of data from one memory space to another can be achieved by first loading the SAPR, DAPR, BCR, and CMR. The DMA transfer sequence begins by setting the STR bit (see Table D-2) in the CMR. The transfer stops when the BCR has decremented to zero, the external $\overline{\text{DONE}}$ pin is asserted externally, or the software clears the STR bit in the CMR. The percentage of bus usage can be controlled by the BT bits to keep the IDMA from exceeding 12.5, 25, 50, or 75 percent of the available bus bandwidth. The transfer can also be interrupted by the $\overline{\text{BCLR}}$ signal, which can be activated through the SDMA channels, or by an interrupt (see 3.8.3 System Control Bits).

Table D-2. Channel Mode Register Bits

Operation	CMR Bits															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
No Action Because an Internal Transfer		X			0											
$\overline{\text{DREQ}}$, $\overline{\text{DACK}}$, $\overline{\text{DONE}}$ Are Used To Read the Source Portion of the Transfer		0			1											
$\overline{\text{DREQ}}$, $\overline{\text{DACK}}$, $\overline{\text{DONE}}$ Are Used To Read the Destination Portion of the Transfer		1			1											
Allow Interrupt on No Error (see Notes 1, 2)			1													
Inhibit Interrupt on No Error (see Note 2)			0													
Allow Interrupt on Bus Error				1												
Inhibit Interrupt on Bus Error				0												
Internal Request Mode (see BT bits)					0	0										
Internal Request Mode at Maximum Rate (Burst)					0	1										
External Request Burst Transfer Mode (Note 3)					1	0										
External Request Cycle Steal Mode (see Note 4)					1	1										
Source Address Pointer Increment Off							0									
Source Address Pointer Increment On							1									
Destination Address Pointer Increment Off								0								
Destination Address Pointer Increment On								1								
Source Size—Reserved									0	0						
Source Size = Byte									0	1						
Source Size = Word									1	0						
Source Size—Reserved									1	1						
Destination Size—Reserved											0	0				
Destination Size = Byte											0	1				
Destination Size = Word											1	0				
Destination Size—Reserved											1	1				
IDMA Transfer of up to 75% of Bus Bandwidth (see Note 5)													0	0		
IDMA Transfer of up to 50% Of Bus Bandwidth (see Note 5)													0	1		
IDMA Transfer of up to 25% Of Bus Bandwidth (see Note 5)													1	0		
IDMA Transfer of up to 12.5% Of Bandwidth (see Note 5)													1	1		
Normal Operation (Software Reset OH)															0	
Software Reset - Abort External Pending Cycles															1	
Stop IDMA at the End of Current Transfer																0
Start (or Restart) IDMA Channel Transfer																1

NOTES:
1. When external pin $\overline{\text{DONE}}$ asserts, an interrupt request is generated.
2. The DONE bit is set in the CSR at the conclusion of data transfer.
3. Transfer occurs at maximum rate. Hardware signal $\overline{\text{DREQ}}$ is level sensitive.
4. Hardware signal $\overline{\text{DREQ}}$ is edge sensitive.
5. These percentages are valid only when bits 11 and 10 = 00.

D.5.6 External Cycles Examples

If the MC68302 is the current bus master and no other internal or external resources are arbitrating for the bus, then the IDMA will obtain bus mastership and perform the data movement cycles when the $\overline{\text{DREQ}}$ signal meets the asynchronous setup time prior to the falling edge of clock.

Figure D-7 shows the sequence of a peripheral requesting a data transfer, the IDMA reading data from memory and then writing the data to the peripheral. The source and destination size are the same for this case. $\overline{\text{DREQ}}$ is sampled on the falling edge of CLK and causes the $\overline{\text{BGACK}}$ signal to assert at the conclusion of the current M68000 core bus cycle. The IDMA performs a read cycle using the SAPR to obtain data and then performs a write cycle to place data in the address specified in the DAPR. The fact that $\overline{\text{DACK}}$ asserts in the write cycle indicates that data is being transferred to the requesting peripheral. Note that the $\overline{\text{BR}}$ and $\overline{\text{BG}}$ pins are not affected by the IDMA in this example. They only activate when the device is in the disable CPU mode (see 3.8.4 Disable CPU Logic (M68000)).

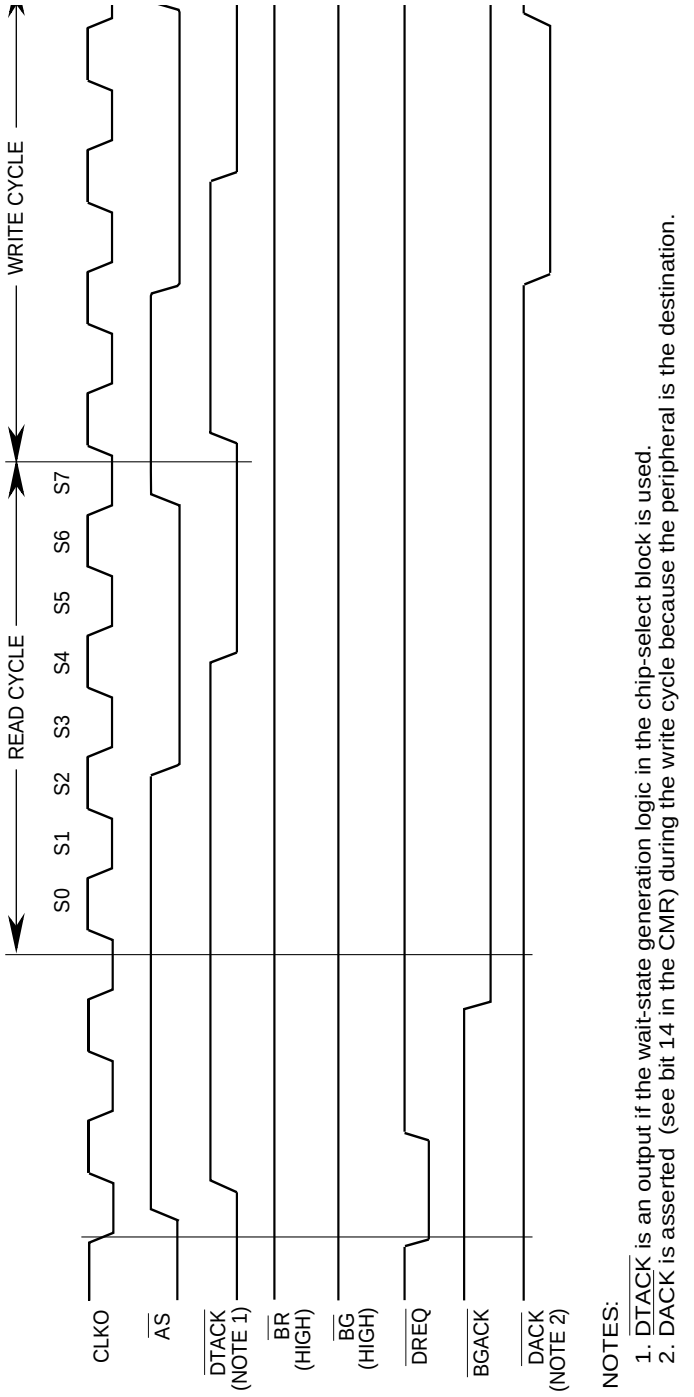
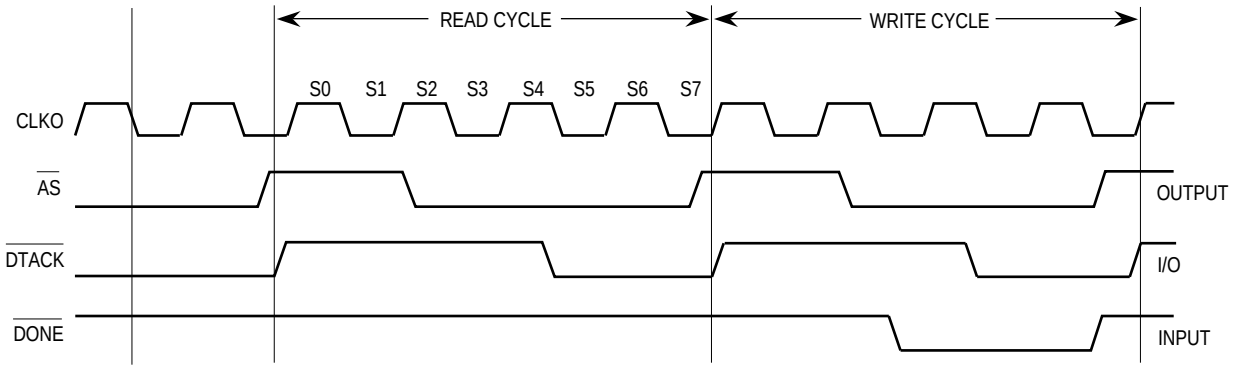


Figure D-7. Typical IDMA External Cycles (Normal Operation)

Figure D-8 shows the peripheral terminating the current IDMA transfer. Note that the DONE signal has been asserted by the peripheral. The IDMA will indicate the transfer has been completed by setting bits in the CSR (see Table D-3).



NOTE: If the byte count had reached zero, DONE would be asserted by the IDMA, indicating normal transfer termination.

Figure D-8. Typical IDMA External Cycles Showing Block Transfer Termination

Table D-3. Channel Status Register Bits

Operation	Bit	7	6	5	4	3	2	1	0
Done Not Synchronized—DONE is asserted as an input during the first access of an 8-bit peripheral when operating with 1 6-bit memory.	DNS					1			
Bus Error Source—A bus error occurred during the read portion of an IDMA cycle.	BES	Reserved					1*		
Bus Error Destination—A bus error occurred during the write portion of an IDMA cycle.	BED							1*	
Normal Channel Transfer Done—The BCR decremented to zero or the external peripheral asserted DONE and no errors occurred during any IDMA cycle.	DONE								1*

* These bits are cleared by writing a one or setting RST in the CMR.

Figure D-9 depicts the typical cycles used on a 16-bit bus when the source data size and destination data size are not equal. In this example, the source size is byte and the destination size is word. The IDMA performs two read cycles to obtain data and then performs a write cycle to place data into the destination location. If the CMR SAPI bit was set, then each byte read increments the SAPR by two. Hence, the SAPR is always pointing to the leftmost or rightmost byte of the 16-bit bus. This type of transfer duplicates the function of an M68000 MOVEP instruction.

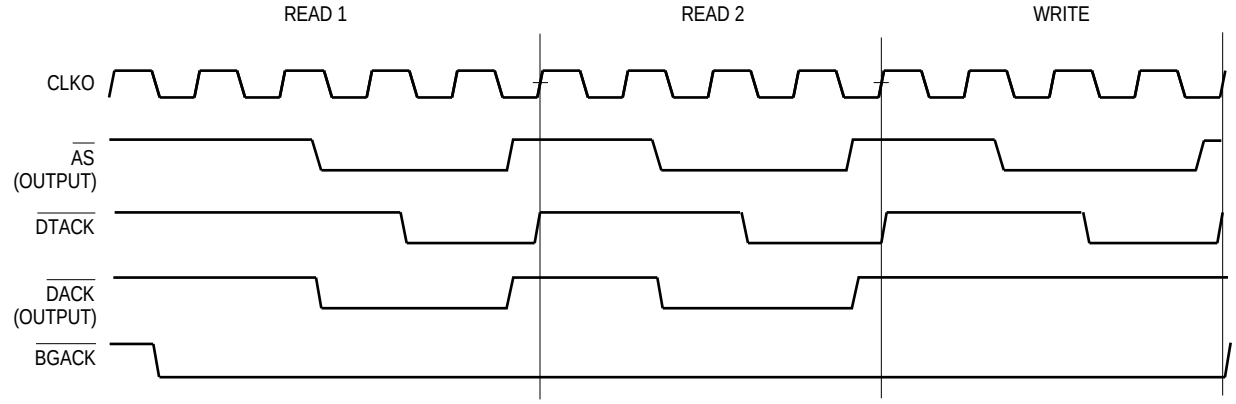
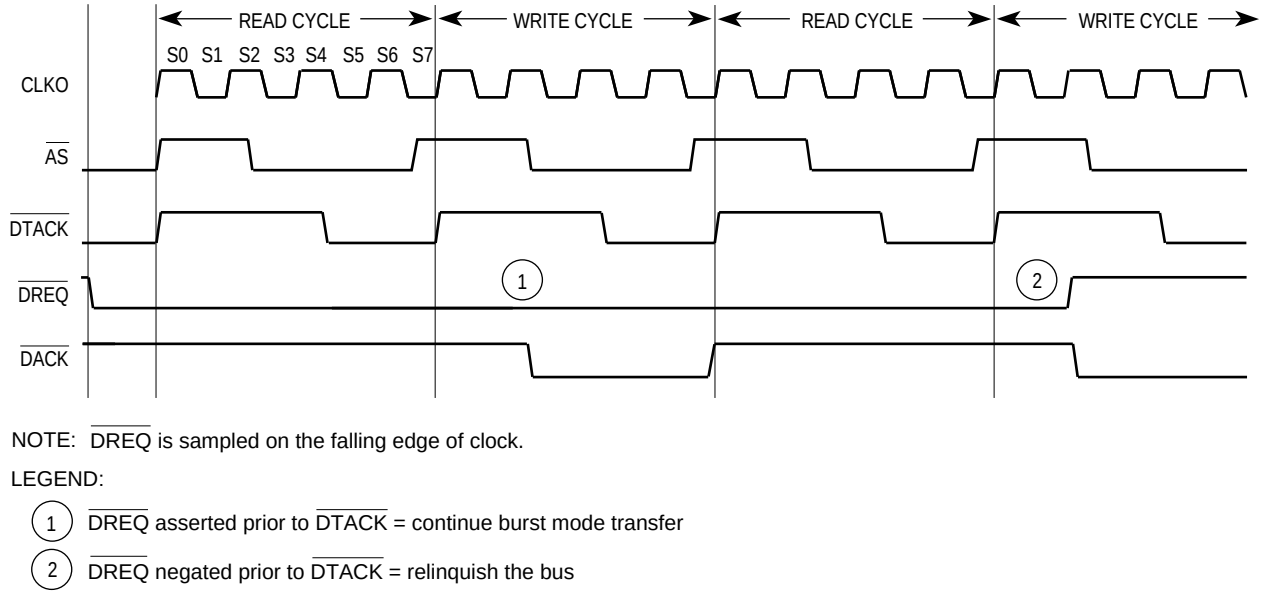


Figure D-9. Typical IDMA Source to Word Destination IDMA Cycles

Figure D-10 illustrates the activation of external burst mode by using the $\overline{\text{DREQ}}$ signal as a level-sensitive input to the IDMA. If $\overline{\text{DREQ}}$ is asserted when the IDMA is accessing the peripheral (indicated by $\overline{\text{DACK}}$ being asserted) as shown in Figure D-10, then the IDMA will continue servicing the peripheral by performing another sequence of operand transfer cycles. The external burst mode stops when the $\overline{\text{DREQ}}$ signal is deasserted prior to the trailing edge of S3 in the cycle where $\overline{\text{DACK}}$ is asserted.



NOTE: $\overline{\text{DREQ}}$ is sampled on the falling edge of clock.

LEGEND:

(1) $\overline{\text{DREQ}}$ asserted prior to $\overline{\text{DTACK}}$ = continue burst mode transfer

(2) $\overline{\text{DREQ}}$ negated prior to $\overline{\text{DTACK}}$ = relinquish the bus

Figure D-10. Burst Mode Cycles

D.5.7 Internal Interrupt Sequence

An interrupt acknowledge cycle (IACK) occurs when an allowed internal or external interrupt request is pending and the priority of the interrupt is higher than the current microprocessor run level. The interrupt acknowledge cycle begins at the conclusion of instruction execution in state S0. All internal resources, including the IDMA, generate INRQ requests at level 4.

The four registers used in interrupt processing are as follows:

1. The interrupt mask register (IMR) contains the flags that, when set, allow the INRQ source to initiate service.
2. The interrupt pending register (IPR) contains bits that correspond to the INRQ source requesting service.
3. The interrupt in-service register (ISR) indicates which internal interrupts are currently being processed (usually only one at a time).
4. The global interrupt mode register (GIMR) has bits that specify interrupt modes such as the edge or level of an input that triggers an interrupt.

A level 4 interrupt may be generated by the IDMA upon completion of a data block transfer. Interrupt processing of IDMA transfers is possible by 1) setting the IDMA interrupt enable (bit 11) in the IMR and 2) setting one or both interrupt enable (INTN and INTE) bits in the CMR (see Table D-1). Once in the interrupt handler, four bits in the CSR indicate the reason for termination of an IDMA data block (see Table D-2).

D.5.8 Final Notes

Only three signals, $\overline{\text{DREQ}}$, $\overline{\text{DACK}}$, and $\overline{\text{DONE}}$, are used to control the handshake between the peripheral and the IDMA. $\overline{\text{DONE}}$ is not needed unless the number of bytes to transmit is not known. The $\overline{\text{DACK}}$ signal can select the peripheral device when data is being transferred to or from it, and wait-state logic can generate $\overline{\text{DTACK}}$ to indicate when valid data is on the bus. Software controls data transfer by setting bits in the CMR as shown in Table D-2 and loading the SAPR and DAPR. It then waits for completion of the transfer by either monitoring the bits in the CSR (see Table D-3) or by using interrupt processing.

D.6 MC68302 MULTIPROTOCOL CONTROLLER TIED TO IDL BUS FORMS AND ISDN VOICE/DATA TERMINAL

The following paragraphs discuss how the MC68302 can be tied to the interchip digital link (IDL) bus, which enables connectivity to a family of ISDN chips. The IDL bus connects the MC68302 integrated controller with the MC145475 S/T interface and the MC145554 CO-DEC to form a basic rate ISDN voice/data terminal (see Figure D-11).

The MC68302 is the first device to combine the benefits of the M68000 microprocessor with a flexible communications architecture. This CMOS device incorporates an MC68000 or MC68008 core processor, a communications RISC processor with associated peripherals, and a system integration block.

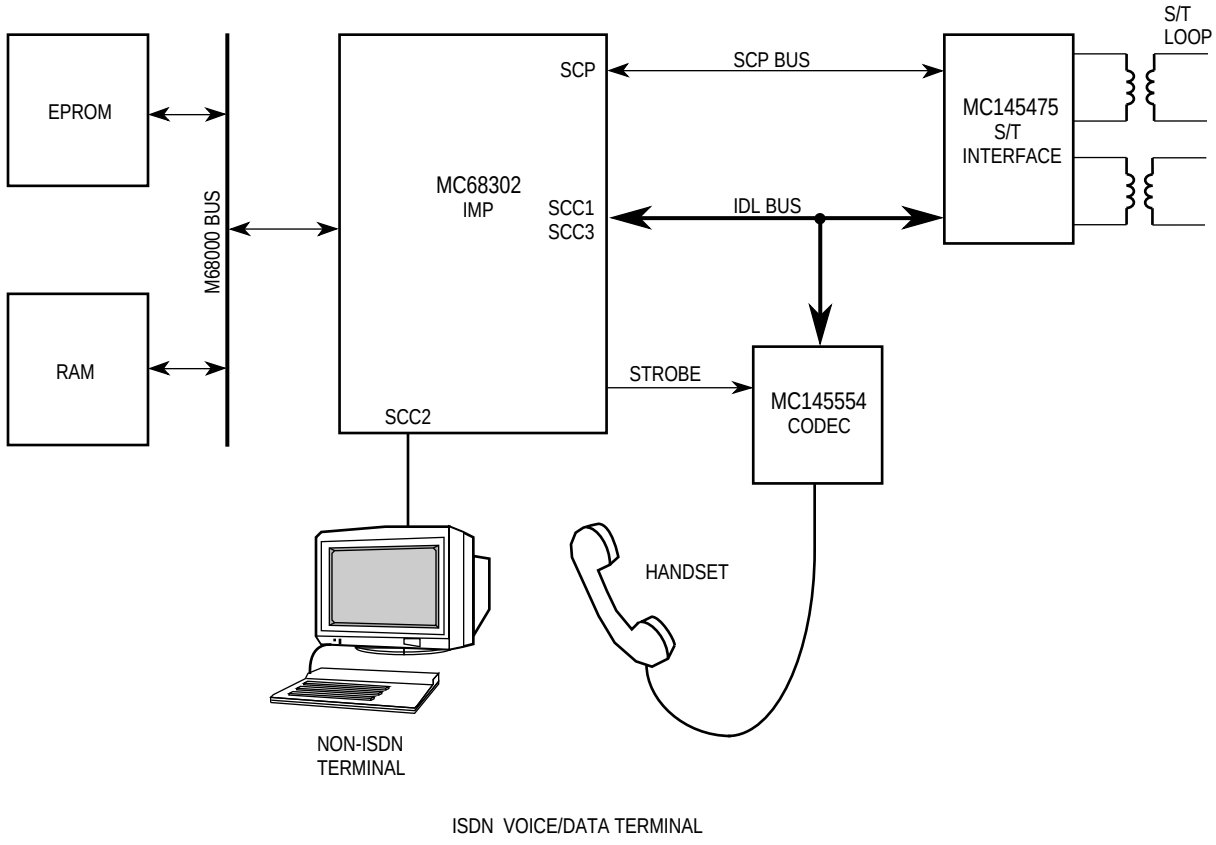


Figure D-11. ISDN Voice/Data Terminal

D.6.1 M68000 Core

The M68000 core processor on the MC68302 is instruction and timing compatible with the standard MC68000 (16-bit) or MC68008 (8-bit) versions of the M68000 Family. The core supports bus lock during read-modify-write cycles, a low latency interrupt mechanism, and bus width configuration. It does not support the older M6800 peripherals.

D.6.2 Communications Processor

The communication processor consists of a RISC processor, three serial communication controllers (SCCs), six DMA channels for the three SCCs, a programmable physical interface, a programmable serial communication port (SCP), and two serial management controllers (SMCs). The RISC processor, a separate processor from the M68000 core processor, is dedicated to the service of the SCCs, SCP, and SMCs.

The MC68302 supports three, full-duplex, independent SCCs, which support HDLC, UART, BISYNC, DDCMP, and V.110 protocols as well as transparent mode.

The physical interface supports a standard nonmultiplexed interface for each of the three SCCs (TXD, RXD, TCLK, RCLK, CTS, RTS, and CD) as well as several multiplexed modes.In multiplexed modes, up to three SCCs can be time-multiplexed onto the same serial channel. The multiplexed modes include IDL, GCI, and PCM highway.

The SCP is a full-duplex, synchronous, character-oriented channel that provides a three-wire interface. It is used to control and program SPI-type devices. The SCP implements a subset of Motorola's SPI interface.

The two SMCs are used to exchange control information multiplexed with the 2B + D data in the IDL or GCI buses.

D.6.3 System Integration Block

The system integration block incorporates general-purpose peripherals that eliminate the glue logic found in most M68000 systems. It includes an independent DMA controller (IDMA), an interrupt controller, parallel I/O ports, 1152-byte dual-port RAM, two timers, one watchdog timer, chip-select lines and wait-state generation logic, a bus arbiter, low power modes, core disable logic, on-chip clock generator, and a hardware watchdog.

D.6.4 IDL Bus

The IDL was developed to maximize the portability of the various chips required in an ISDN system. It provides a consistent interface definition across which a family of ISDN chips will be able to transport data (see Figure D-12).

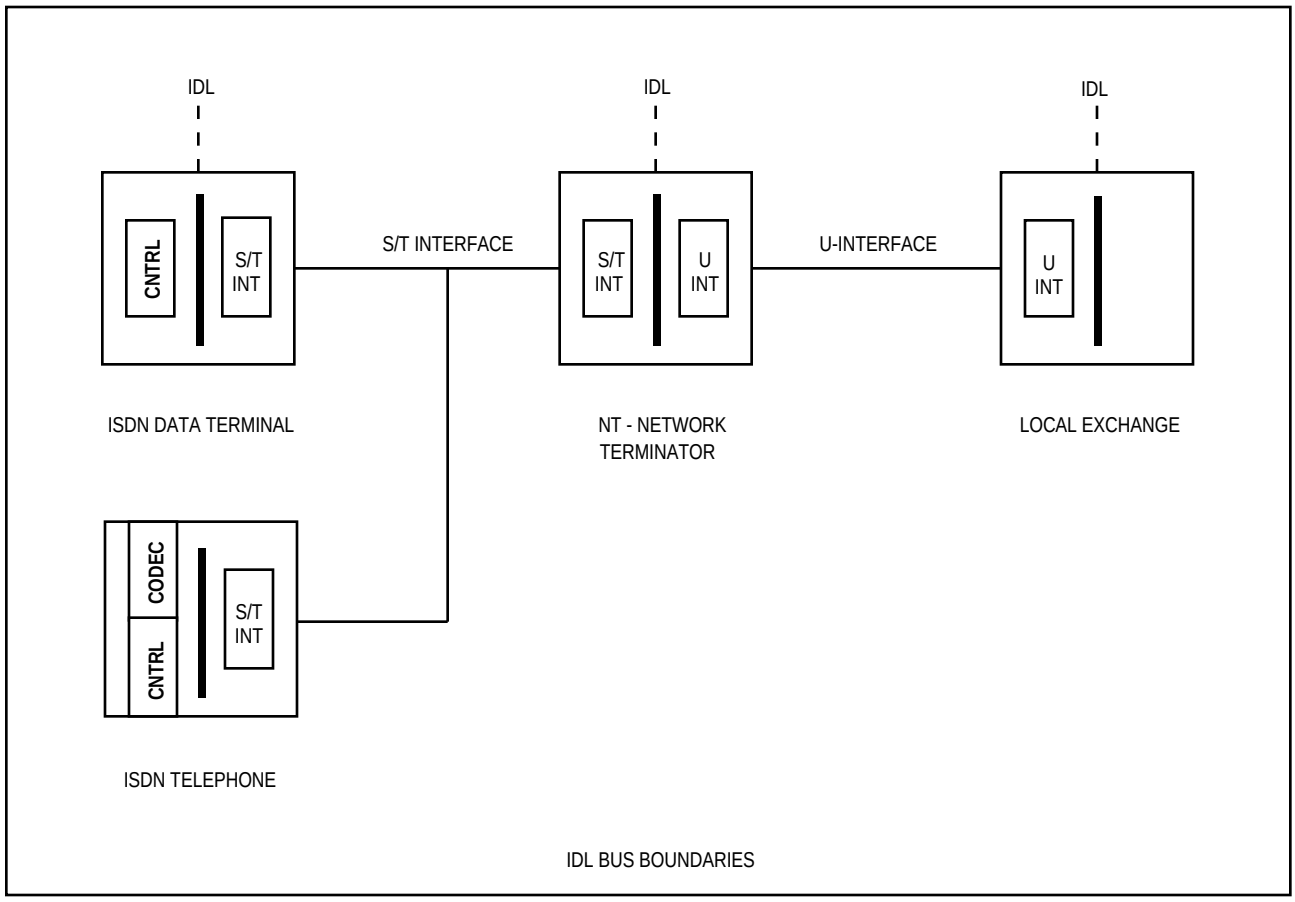


Figure D-12. IDL Bus Boundaries

Motorola offers a full line of ISDN/IDL compatible chips that enable modular and portable design of ISDN equipment:

- MC145472 — ISDN U interface transceiver conforms to the American Standard for ISDN basic access.
- MC145474 — ISDN S/T interface transceiver conforms to CCITT 1.430 and ANSI T1.605 recommendations.
- MC145475 — ISDN S/T interface transceiver conforms to CCITT 1.430 and ANSI T1.605 recommendations. Supports NT1 Star Operation.
- MC145554/7— Mu-Law and A-Law Companding PCM CODEC Filter (16-pin package).
- MC145564/7— Mu-Law and A-Law Companding PCM CODEC Filter (20-pin package).

D.6.5 IDL Bus Specification

The IDL bus (see Figure D-12) consists of four wires and provides data exchange with 125-µsec frame period with clock speeds of 1.544 to 2.56 MHz. The IDL supports a total of 160 kbps of full-duplex data consisting of two B-channels (2 x 64 kbps), one D-channel (16 kbps), one M-channel (8 kbps), and one A-channel (8 kbps). The A and M channels are used to convey maintenance and auxiliary data.

The IDL bus connects one IDL master to one or more IDL slaves. The IDL bus timing is driven by the master device.

The bus signals are as follows:

- CLOCK — always provided from the master to the slave. It provides the bit timing for the data traveling across the IDL.
- SYNC —provides the framing for the IDL. The SYNC occurs once per 125-µsec frame and is active high one full clock cycle in the bit immediately preceding the data transaction.
- TXDATA — drives the data from one chip to another. The line is in high impedance when no data transaction occurs.
- RXDATA — the input line that receives data from the TXDATA of another part.

D.6.6 IMP/IDL Interconnection

The MC68302 directly connects to the IDL bus with no glue logic. The MC68302 is an IDL slave (accepts IDL timing from the bus). In the application described, the IDL master device is the MC145475 S/T interface chip (see Figure D-14).

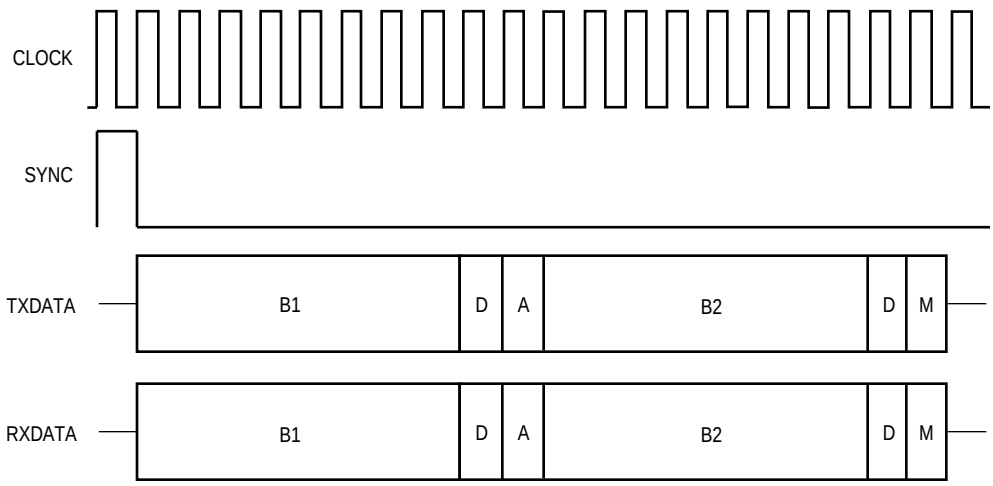


Figure D-13. IDL Frame Structure

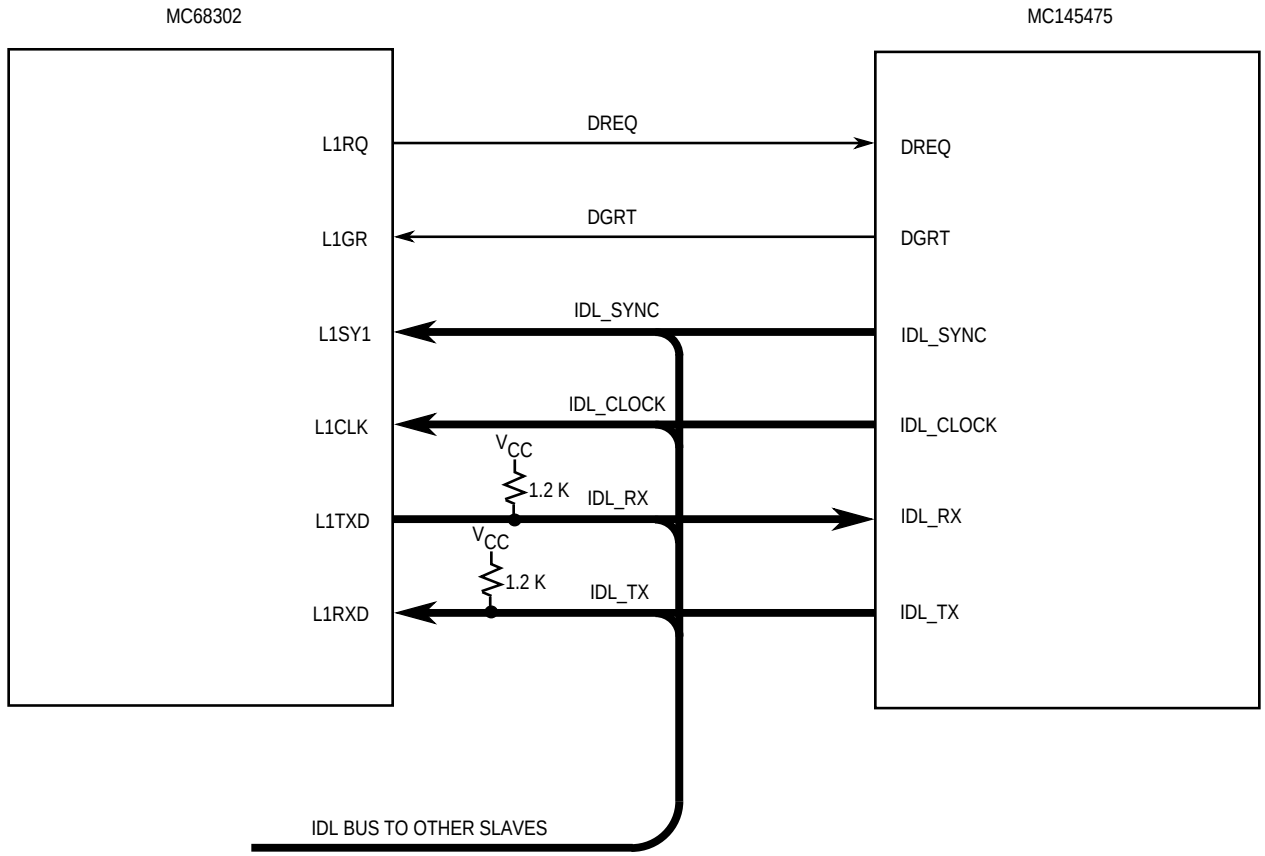


Figure D-14. IDL Bus to Other Slaves

To support access to the S/T interface D-channel common resource, the MC145475 provides two additional signals. The DREQ signal is asserted by the device wanting to transmit a frame on the D-channel. The availability of the D-channel is signaled by the MC145475 by asserting DGRT.

In case of a collision, DGRT will be negated, signaling that D-channel transmission must be aborted. If the MC68302 is programmed to do so, an automatic retransmission will take place when the D-channel is available again.

The MC68302 generates two serial data strobe signals, SDS1 and SDS2, which are programmed to envelope the B1 or/and B2 time slots. The strobe is helpful in gating a device to the proper B-channel without the need for additional frame synchronization logic.

The SDSs are programmable and each can be set to envelope B1, B2, or B1 lB2 by setting the SDC1 and SDC2 bits in the serial interface mode register (SIMR). In this application, SDS is used to gate a CODEC to the proper B-channel.

D.6.7 Serial Interface Configuration

To allow the MC68302 to be tied to the IDL bus, the serial interface (see Figure D-15) must be configured to the IDL mode. A value of $0C76 written to the SIMR will set the following configuration: IDL mode, D-channel routed to SCC3, B1-channel routed to SCC1, SCC2 working in NMSI mode (routed to its external pins), and SDS1 enveloping B2-channel (CODEC strobe).

This configuration is suitable for terminal adaptor configuration, in which the non-lSDN terminal is connected to SCC2. The output/input frames of the rate adaption protocol are transmitted/received by SCC1 over B1 channel, and SCC3 handles the signaling over the D-channel (call establishment, disconnect, etc.). The B2 channel is routed to a CODEC by the serial data strobe 1 and is used for voice calls

The serial interface mask register allows selection of subrates of the 64-kbps B-channel capacity by selecting the used bits of the eight B-channel bits.

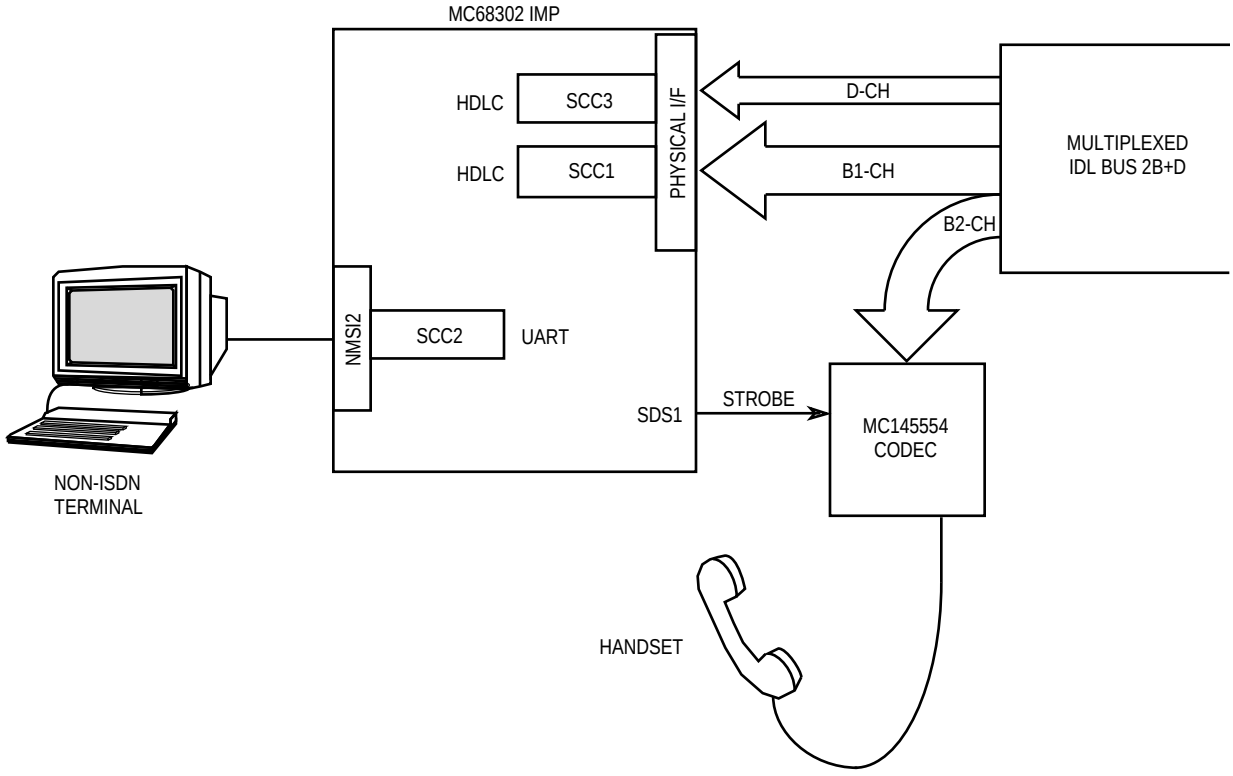


Figure D-15. Serial Interface Configuration

D.6.8 SCC Configuration

SCC1 will be set according the rate adaptation protocol implemented. If V.120 is used, the protocol should be set to HDLC. If V.110 is used, program SCC1 to V.110 (configured under the DDCMP mode). The following values are suitable:

Register	Value	Comments
Configuration Register (SCONE)	$30FF	External receive and transmit clocks will be provided from the IDL through the serial Interface.
Mode Register (SCM1) for V.120	$000C	HDLC mode.
Mode Register (SCM1) for V. 110	$040E	V.110 mode.
Sync Register (DSR1)	$7E7E	HDLC flag.
Sync Register (DSR1)	$0100	V.110 sync pattern
Mask Register (SCCM1)		The mask should be set according to the interrupt events handled.

SCC2 will be set to handle a 9600 baud UART from a non-ISDN terminal. The following values are suitable:

Register	Value	Comments
Configuration Register (SCON2)	$30FF	Set internal receive and transmit clocks, 9600 baud if 16.6-MHz parallel clock used.
Mode Register (SCM2)	$010D	UART mode, 8 bits, no parity
Sync Register (DSR2)	$7E7E	Bits 14-12 control stop-bit shaving.
Mask Register (SCCM2)		The mask should be set in accordance with the interrupt events the software wants to handle.

SCC3 will be set to handle the HDLC frames of LAPD protocol over the D-channel. (Multiplexing SCC3 into lDL leaves the SCP pins available for the SCP port.) The following values are suitable:

Register	Value	Comments
Configuration Register (SCON3)	$3000	External receive and transmit clocks (will be provided from the IDL through the serial interface)
Mode Register (SCM3)	$010C	HDLC mode with retransmit option.
Sync Register (DSR3)	$7E7E	HDLC flag.
Mask Register (SCCM3)	$1B	The mask should be set in accordance with the interrupt events the software wants to handle.

NOTE

In addition to programming the SCCs registers, the protocol-specific parameter RAM of each SCC should be initialized by the control software according to the protocol selected.

D.6.9 Parallel I/O Port A Configuration

To implement a DCE interface for the non-ISDN terminal, PA2 and PA5 should be configured as general-purpose outputs (drive $\overline{CTS}$ and $\overline{CD}$).

PA7 and PA8 should be configured as general-purpose outputs (drive SCP_EN and RESET).

PA4 and PA6 may be configured either as general-purpose inputs or as dedicated modem pins.

If a general-purpose input is chosen, the state of the terminal $\overline{DTR}$ and $\overline{RTS}$ lines will be polled by application software by reading the parallel I/O data register.

If PA4 and PA6 are to be used as general-purpose inputs, the following registers should be set:

Register	Value	Comments
PACNT	$000B	Set PA0 PA1, and PA3 to the dedicated mode. (RXD2, TXD2, and TCLK2 connected to SCC2.)
PADDR	$01A4	Set PA2, PA5, PA7, and PA8 as output pins. ($\overline{RCLK2}$ and $\overline{RTS2}$ are outputs.)
PADAT		Will be set by the application software to drive PA2, PA5, PA7, and PA8 to the proper state. Will be read to check $\overline{DTR}$ and $\overline{RTS}$.

If the dedicated mode is chosen, the software can use the delta mechanism of the SCC to generate an interrupt upon any change in the state of $\overline{RTS}$ or $\overline{DTR}$. In this case, the SCC can be configured to use its $\overline{CTS}$ (connected to the terminal $\overline{DTR}$ line) and $\overline{CD}$ (connected to the terminal $\overline{RTS}$ line) under automatic control or under software control by setting the DIAG bits in the mode register.

If $\overline{CTS2}$ (PA4) and $\overline{CD2}$ (PA6) are to be used as dedicated pins, the following registers should be set:

Register	Value	Comments
PACNT	$005B	Set PA0, PA1, PA3, PA4, and PA5 to the dedicated mode. (RXD2, TXD2, TCLK2, CTS2 and CD2 connected to SCC2.)
PADDR	$01A4	Set PA2, PA5, PA7, and PA8 as output pins. ($\overline{RCLK2}$ and $\overline{RTS2}$ are outputs.)
PADAT		Will be set by the application software to drive PA2, PA5, PA7, and PA8 to the proper state.

D.6.10 SCP Bus

The SCP (see Figure D-15) is an industry standard bus used for controlling and programming external devices. The SCP is a four-wire bus consisting of transmit path, receive path, associated clock, and enable signal (see Figure D-16). The clock determines the rate of the exchange of data in both the transmit and receive directions, and the enable signal governs when this exchange occurs.

The interconnection between the MC68302 SCP and the MC145475 is straightforward. The MC68302 is an SCP master device and will generate the clock timing and the SCP_ENABLE for the SCP transaction. The SCP_EN signal is driven by a parallel output pin and must be handled by software (asserted for each transaction between the MC68302 and the MC145475).

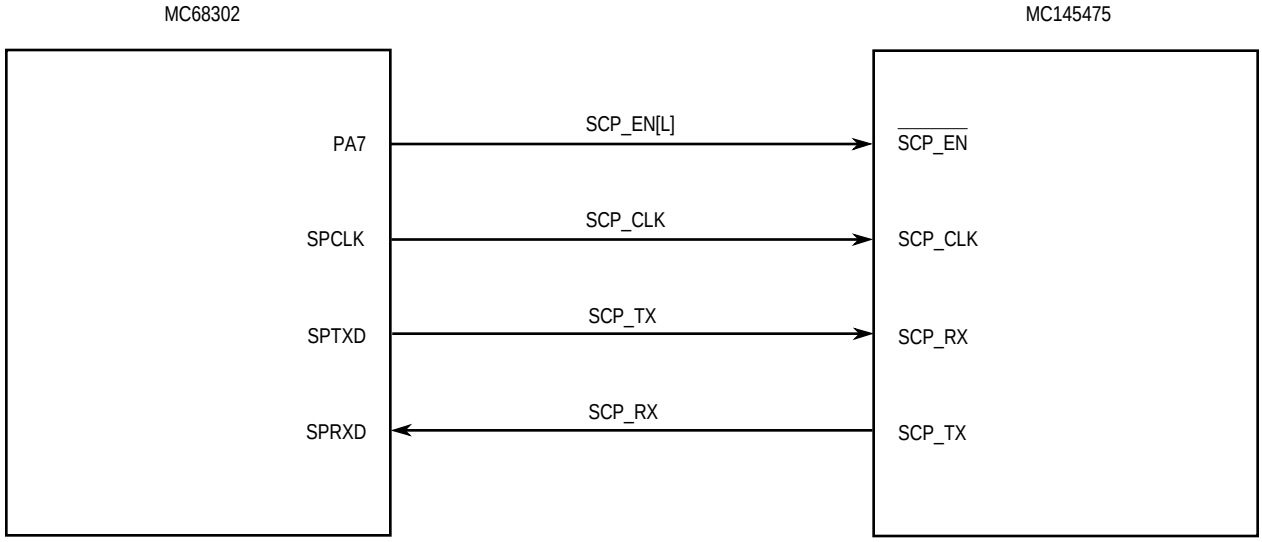


Figure D-16. SCP Bus Interconnection

D.6.11 SCP Configuration

The SCP is configured by setting the SCP mode register. Setting the SCP mode register to $2F will configure the SCP to the clock invert mode (clock is idle high, transmitted data is shifted on falling edges and received bits are sampled on rising edges), and the internal baud rate generator will divide the system clock by 32.

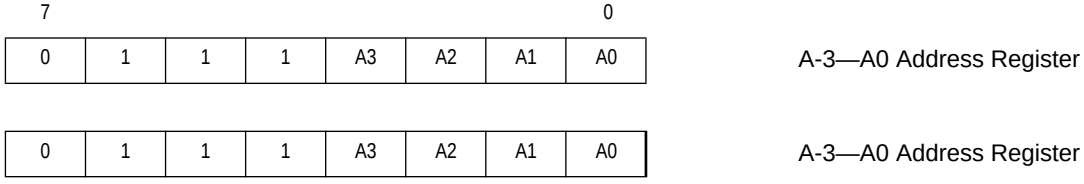
Each data transaction is triggered by setting the start bit high (after software has configured the SCP buffer descriptor).

D.6.12 SCP Data Transactions

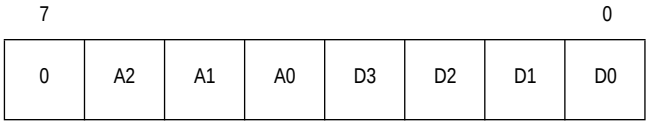
The MC145475 has two sets of read/write programmable registers. One is the nibble (4 bits) register set and the other is the byte (8 bits) register set.

The registers are set to default value on reset. Read/write operation is made via the SCP channel.

A byte register write is made by writing two bytes with the following format:

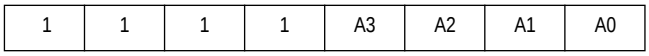


A nibble register write is made by writing one byte with the following format:

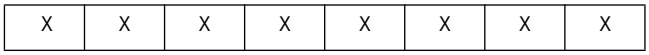


A3–A0 Address Register
D3–D0 Data Register

A byte register read is made by writing two bytes with the following format:



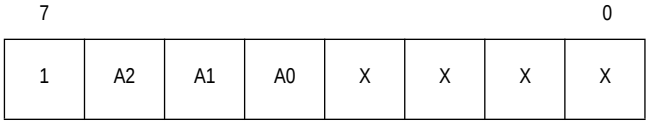
A3–A0 Address Register



X–Don't Care

Data read from the register will be received during the second transaction.

A nibble register read is made by writing one byte with the following format:



A2–A0 Address Register
X– Don't Care

Data read from the register will be received during the second transaction.

NOTE

The SCP_EN signal must be asserted prior to each SCP transaction and negated after completion.

D.6.13 Additional IMP To S/T Chip Connections

In addition to the IDL bus and the SCP bus, two discrete signals connect the MC145475 S/T chip to the MC68302 (see Figure D-17).

$\overline{\text{IRQ}}$ —The active-low signal sends an interrupt request from the MC145475 to the MC68302 core. This is an active-low signal that is asserted when one or more of the following events occurs:

— Change in the received information state (INFO n) of the S/T receiver.
— Multiframe reception.
D-channel collision.

Each event can be masked and/or cleared by a write/read operation on the corresponding register. The $\overline{\text{IRQ}}$ signal can be connected to the IRQ1 pin of the MC68302 to generate a level 1 interrupt.

$\overline{\text{RESET}}$—This active-low signal initializes the MC145475, forces all internal state machines to the initial state, and forces all internal nibble and byte registers (except BR4 and

BR5) to their default value. RESET should be asserted after power-on and negated for normal operation.

In the application described, $\overline{\text{RESET}}$ is driven from a parallel output pin, PA8, of the MC68302.

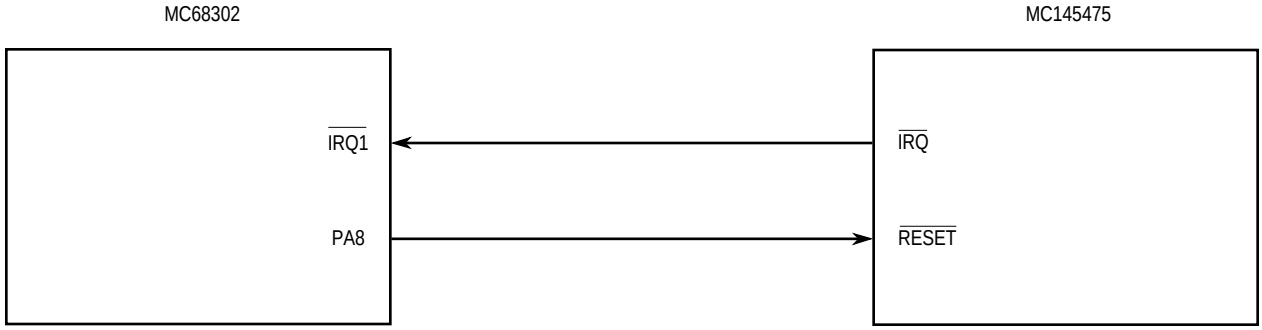


Figure D-17. Discrete Signal Interconnection

D.6.14 Initialization of the MC145475

The following tables show the initialization sequence after power-on reset (assuming the registers have default values).

If configured as NT, a possible initialization sequence is as follows:

Register	Value (hex)	Comments
Byte Register #7 (BR7)	$0C	IDL master mode; 2-MHz IDL clock (suitable for the MC145554 CODEC).
Nibble Register #4 (NR4)	—	Interrupt mask—according to the application.

If configured as TE, a possible initialization sequence is as follows:

Register	Value (hex)	Comments
Byte Register #7 (BR7)	$04	2-MHz IDL clock (suitable for the MC145554 CODEC).
Nibble Register #4 (NR4)	—	Interrupt mask—according to the application.
Nibble Register #5 (NR5)	$C	Enables B1 and B2 on S/T loop (with accordance to signaling on the D-channel).

NOTE

In the application described, power-on reset to the S/T chip is provided from one of the parallel output pins (PA8). It is the responsibility of software to reset the S/T chip after power-up.

Nibble register 2 allows activation (TE/NT) and deactivation (NT) of the S/T loop. To activate the loop, write $8 to NR2. To deactivate the loop, write $4 to NR2.

D.6.15 MC145554 CODEC Filter

A CODEC is a device that performs voice digitization and reconstruction as well as the band limiting and smoothing required for PCM systems.

The MC145554 is a Mu-law device that can be tied directly to the IDL bus. The SDS strobe generated by the MC68302 serial interface can be used to route the CODEC to the proper B-channel.

NOTE

Nibble register 2 allows activation (TE/NT) and deactivation (NT) of the S/T loop. To activate the loop, write $8 to NR2. To deactivate the loop, write $4 to NR2.

The analog interface between the CODEC and the handset shown in Figure D-18 is suitable for a carbon microphone. If another low-output microphone is used, an amplifier must be included to drive the input of the CODEC to the proper levels.

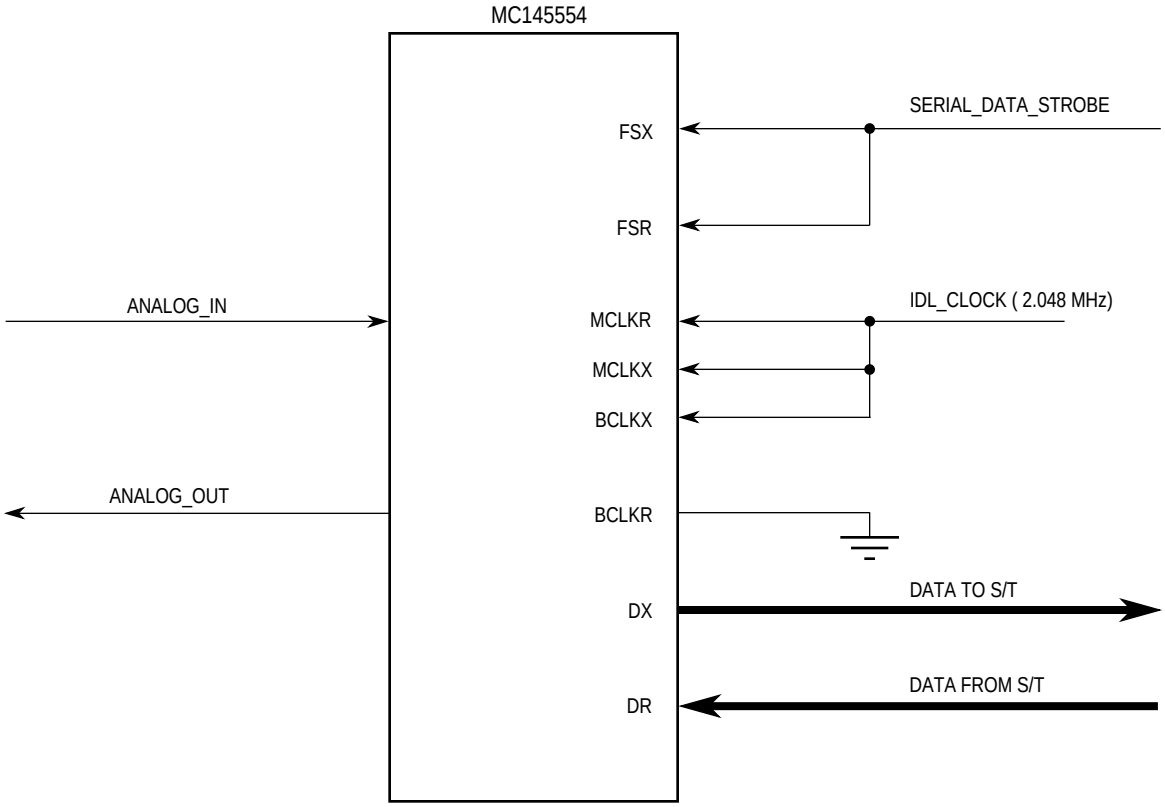


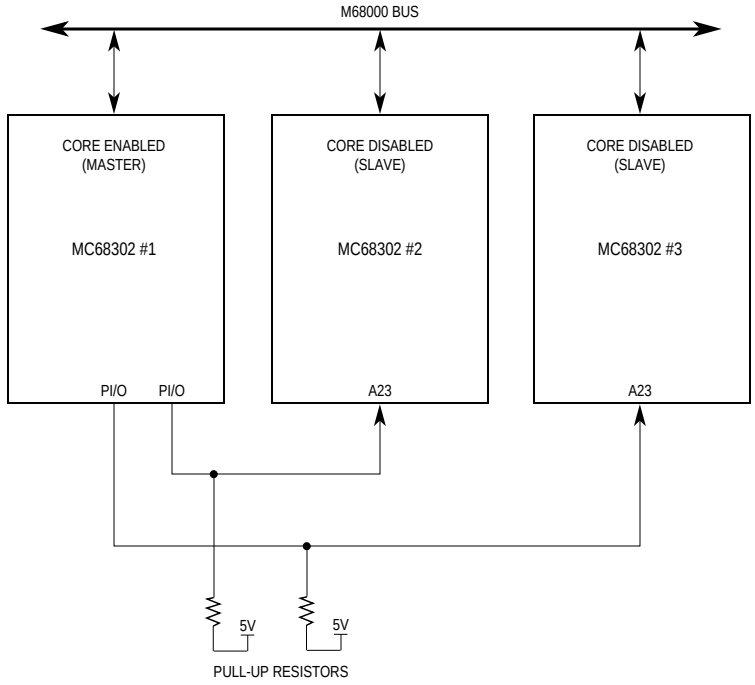
Figure D-18. CODEC/IDL Electrical Connection

D.7 INTERFACING A MASTER MC68302 TO ONE OR MORE SLAVE MC68302S

When more than three SCC channels are required in an application, multiple MC68302s can be used to solve the problem. One possible solution would be to design a board with several

fully functioning MC68302s, each having an isolated bus and the ability to send data and messages between them (e.g., through a shared RAM). However, another approach is possible.

By using the MC68302 “disable CPU logic” feature, enabled with the DISCPU pin, the MC68302 can be converted into an intelligent slave peripheral that no longer has its M68000 core operating. The SDMA channels and IDMA channel request the bus externally through the bus request ($\overline{BR}$) pin. (When not in slave mode, these channels request the bus internally to the on-chip bus arbiter, with no external indication visible.) A typical slave mode example is shown in Figure D-19. A single master MC68302 (i.e., one with the M68000 core enabled) can access and control one or more slave MC68302s. (i.e., ones with the M68000 core disabled.)



NOTE: A23 is not used by the slaves.

Figure D-19. Typical Slave Mode Example

Use of the “disable CPU logic” feature in a multiple MC68302 system depends mainly on the amount of protocol processing required by the M68000 core. If the data rates are high and the amount of protocol processing required on each channel is significant, the M68000 core may be the limiting factor in communications performance. Thus, further increases in serial rates will not yield additional packets/sec performance. In such a case, a faster processor (such as the MC68020/MC68030) could be used to control all three MC68302 devices in slave mode.

The bus utilization of the SDMA channels on the three MC68302 devices is not usually a significant factor. For instance, if three SCC channels are running full duplex at 64 kbps, the respective SDMA channels consume less than 1 percent of the M68000 bus. You can calculate this figure for your design by determining how often a bus cycle to memory is required

for a given protocol. For HDLC and transparent, this is every 16 bits; whereas, in the other protocols, it is every 8 bits on receive and every 16 bits on transmit.

This application does not address the issue of choosing whether a master-slave arrangement is the best approach for your application, but rather looks at the design issues that need to be addressed once the approach seems reasonable. (Most applications usually find the master-slave approach shown in Figure D-19 quite acceptable for data rates up to 64 kbps on each of the nine SCCs. Any comments on rates beyond this value tend to be less general and more application dependent.)

D.7.1 Synchronous vs. Asynchronous Accesses

When a device is in slave mode, there are two different ways it can be accessed: synchronously and asynchronously.

When the MC68302 enters slave mode, it is initialized with asynchronous accesses. The term “asynchronous” refers to the fact that the master does not need to supply signals to the slave on particular system clock edges, as required in synchronous. However, asynchronous accesses are always and only three waits states for both reads and writes. In general, this is the suggested method of accessing the MC68302 in slave mode. These accesses are much simpler than in synchronous timing. Any performance loss is minimal since, once the MC68302 is initialized, accesses to it are rare. The SDMA channels, which are unaffected by this choice, can still operate with zero wait states and comprise most of the slave MC68302 activity.

If the SAM bit in the SCR is set, the following accesses to the slave MC68302 will be synchronous. Writes will be zero wait states, and reads can be either zero or one wait state, based on the EMWS bit in the SCR. The EMWS bit is almost always required to meet synchronous setup times. Synchronous accesses should be used when one master MC68302 is accessing another slave MC68302 without any intervening glue (like buffers). In this case, the synchronous timings can be met with proper clocking.

D.7.2 Clocking

The master clock out (CLKO) line should be connected to the EXTAL input of the slaves, allowing the master MC68302 to perform zero wait state writes and one wait state reads from the slave MC68302 at full 16.67-MHz operation in synchronous mode. The guaranteed propagation delay between EXTAL and CLKO is 2-11 ns at 16.67-MHz operation. The typical delay is about 5 ns.

If a single external clock is required to drive the inputs to all the MC68302 EXTAL pins, the skew between EXTAL and CLKO must be watched by the designer. In the case of synchronous accesses, this will certainly slow the overall frequency that can be designed.

D.7.3 Programming the Base Address Registers (BARs)

The next issue is how to program the three BARs on each of the MC68302s considering they have the same address ($0F2). Figure D-19 shows an easy method of programming the BAR. The master MC68302 first programs its BAR at $0000F2. For each slave, it writes the slave's parallel l/O line with a zero and writes to $8000F2 to program the slave's BAR. This

method works because the parallel I/O lines default as inputs to the MC68302 and can therefore all be pulled high initially. After the slave BARs are programmed, the parallel l/O lines on the master should be reconfigured as inputs; otherwise, a contention could occur on A23 when a slave's DMA is accessing the bus.

This method is the easiest because it requires no external glue. It costs one parallel l/O line per slave on the master MC68302 and reduces the address space of each slave from 16 MB to 8 MB, neither of which should be a problem in most systems. If A23 is really needed on the slaves, it can be regained, but extra logic is required.

D.7.4 Dealing with Interrupts

The following example is the easiest method for dealing with interrupts from the slaves. It assumes that any other external interrupt sources are sent directly to the master MC68302 without using the interrupt controllers on the slaves.

1. The internal interrupts cause the slaves to force out level 4 on their $\overline{IOUT2}$-$\overline{IOUT0}$ pins. (AVEC, RMC, and CS0 are not available on the slaves.)
2. IOUT2 from the slave is connected to the master PB8, PB9, PB10, or PB11 pin if the master is in normal interrupt mode, or to $\overline{IRQ1}$ or $\overline{IRO6}$ if it is in dedicated interrupt mode. Thus, in normal mode the slave interrupts will arrive at level 4, and in dedicated mode they will arrive at either level 1 or 6.
3. The best method of operation is for the slaves not to generate the vector during the interrupt acknowledge cycle. The master MC68302 can generate it and then read the IPR of the slave to determine the actual source of the interrupt.

NOTE

Why not use the vector generation enable (VGE) bit in each slave MC68302 to allow the slaves to generate different vectors for each of their internal peripherals? This could be done; however, the slaves must be tricked into responding to an unique interrupt level (or else multiple vectors could collide on the bus simultaneously). The external decoding and address buffer logic required to do this slows down the interface timing (and adds expense). Rather, the VGE bit is intended for applications where a single MC68302 is a slave to another processor such as the MC68020.

To use the interrupt controller on a slave MC68302 (in either normal or dedicated mode) to handle interrupt levels 1, 6, or 7 from an external peripheral, connect the slave's $\overline{IOUT2}$-$\overline{IOUT0}$ pins directly to the master's $\overline{IPL2}$-$\overline{IPL0}$ pins. The master MC68302 will supply the vector for levels 1, 6, and 7, and level 4 of the slave will be interpreted as an error vector (00000). Upon branching to this vector, the master MC68302 software should then check the slave's IPR to identify the source.

D.7.5 Arbitration

If only one slave is present, no arbiter is required because the $\overline{BR}$ as an output on the slave can be sent directly to the $\overline{BR}$ input on the master. Figure D-20 shows a dual master-slave system using this arbitration scheme. Note that the $\overline{BCLR}$ pin from the slave MC68302 can be used to give the 12 SDMA channels in the system priority over the two IDMA channels in the system. $\overline{BCLR}$ is asserted whenever an SDMA channel wants the bus, and, will tem-

porarily halt any IDMA accesses. The SDMAs only use the M68000 bus for a single bus cycle before giving up the bus, so priority between SDMAs is not an issue. In this system, if an SDMA channel from the slave requests service at the same time as an SDMA channel from the master, the slave SDMA will go first.

With multiple slaves (i.e., multiple external bus masters from the standpoint of the master MC68302), external logic must prioritize the various $\overline{BR}$ signals. The $\overline{BGACK}$ and $\overline{BCLR}$ signals can be connected as shown in Figure D-20, but the $\overline{BR}$ and $\overline{BG}$ signals must be routed to the external bus arbiter.

D.7.6 Final Notes

It is important for the slave to negate its $\overline{BR}$ pin quickly after it asserts $\overline{BGACK}$ to meet the master's bus arbitration timing specifications. Thus, a 820Ω pullup resistor is used in Figure D-20.

Will the slave SDMA channels move SCC data to/from the low half of memory or the high half of memory? If the decision is the low half of memory, the following notes about A23 should be considered.

Since A23 on the master and the slave are not driven by the address bus, they must be at the proper level before beginning accesses to and from the slave MC68302. Thus, the slave's A23 pullup may have to be reduced from 10k to 1k to give a fast enough rise time (for instance, a slave SDMA access is immediately followed by a master access to the slave).

The decision of whether to pull the master's A23 pin up or down is made based on whether the SDMA on the slave will be storing its SCC data into the high half or the low half of memory, respectively. A 1k resistor may be required in this case as well. However, if the master MC68302 chip-select logic is used by the slave, the chip-select comparison for the A23 pin can be disabled. (If this trick is used, it is important that no peripherals or memory be mapped to the chip select's corresponding area in the upper half of system memory).

D.8 USING THE MC68302 TRANSPARENT MODE

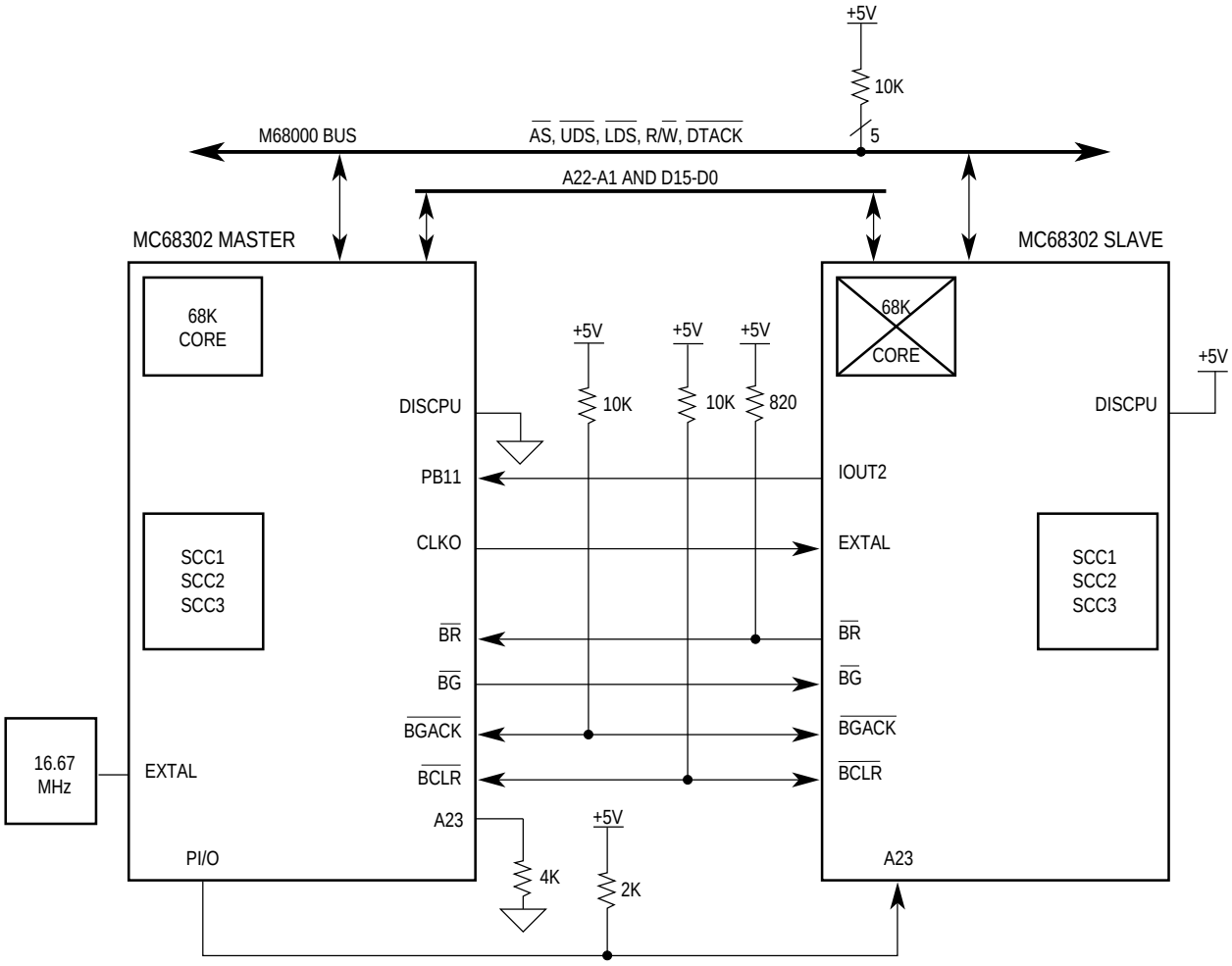
The following paragraphs describe different ways that the totally transparent (promiscuous) mode on the MC68302 serial channels can be used.

D.8.1 Transparent Mode Definition

Transparent mode allows the transmission and reception of serial data over a serial communications controller (SCC) without any modification to that data stream. Contrast this with HDLC mode, where zero bits are occasionally inserted during transmission and stripped during reception, and with UART mode, where start and stop bits are inserted and stripped. Transparent mode provides a clear channel on which no bit-level manipulation is performed by the SCC. Any protocol run on transparent mode is performed in software. The job of an SCC in transparent mode is to function simply as a high-speed serial-parallel converter.

Transparent mode on the MC68302 also means a synchronous protocol; thus, a clock edge must be provided with each bit of data received or transmitted. Contrast this with the UART

protocol, where a 16x oversampling is employed to effectively extract the clock information from the data. With transparent mode (and all the other protocols), this clock can be generated internally with a baud rate generator or can be provided externally.



NOTE: The M68000 core on the master controls all six SCCs.

Figure D-20. Dual Master-Slave System

D.8.2 Applications for Transparent Mode

There are several basic applications for the use of transparent mode.

First, some data may need to be moved serially but may not require protocol superimposed — for example, voice data. There is no reason to encode voice data, and no error correction is needed. With voice data, an occasional dropped bit will not interfere with the data stream in any significant way. The MC68302 transparent mode works well for this type of application.

Second, some board-level applications require a serial-to-parallel and parallel-to-serial conversion. Often this is done to allow communication between chips on the same board. The

SCCs on the MC68302 can do this very efficiently because of their sophisticated DMA capability, and very little MC68000 core intervention is required.

Third, some applications require the switching of data without interfering with the protocol encoding itself. For instance, in a multiplexer, data from a high-speed time-multiplexed serial stream is multiplexed into multiple low-speed data streams. In this case, the idea is to switch the data path but not alter the protocol encoded on that data path.

Finally, some applications require a special protocol that does not fall under the category of HDLC, UART, etc. In some instances, transparent mode can be used; however, care should be taken to understand the capabilities of transparent mode before trying this. The most important issue is how this new protocol recognizes its frames on the receiving end. Transparent mode on the MC68302 was designed to work well receiving continuous streams of data (no gaps in the data exist over time). This is different from receiving transparent frames; although there is some support for this, it is limited on the MC68302.

D.8.3 Physical Interface to Accompany Transparent Mode

Before discussing the details of transparent mode timing, we need to chose the physical interface to go with the transparent mode. The timings associated with transparent mode differ based on the physical interface chosen.

The MC68302 supports the following four physical interfaces:

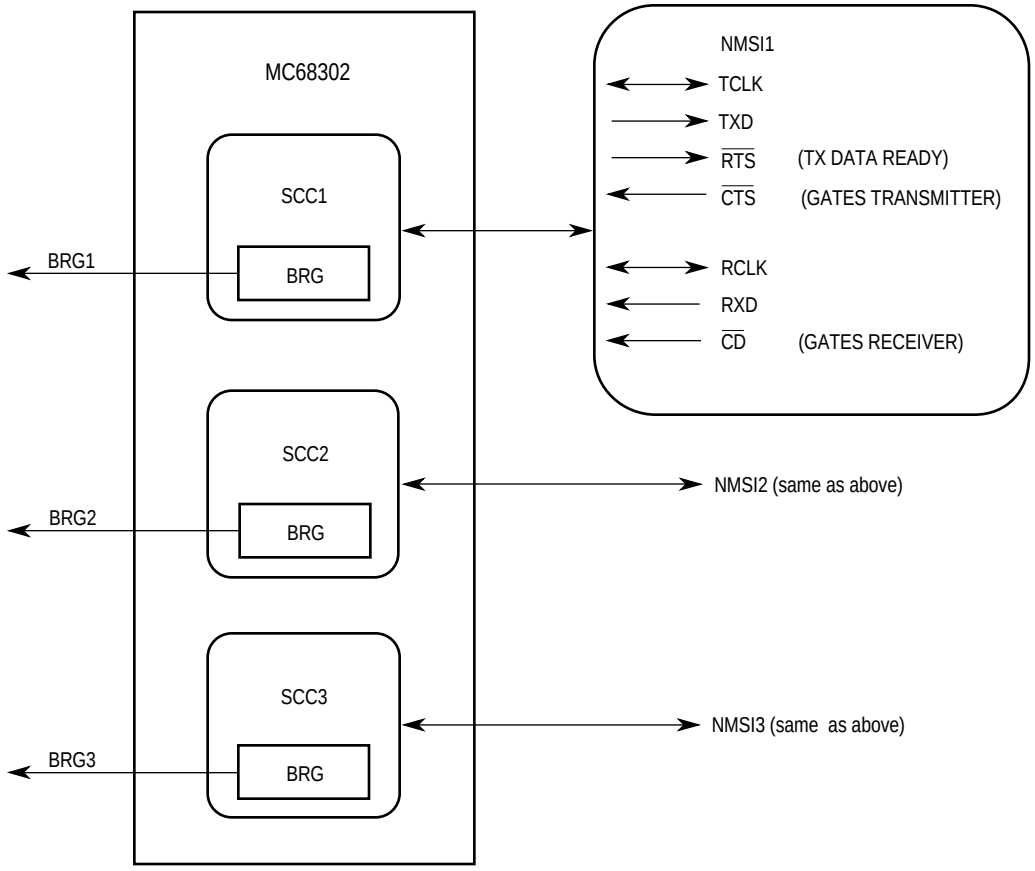
- Nonmultiplexed Serial Interface—NMSI
- Pulse Code Modulation Highway—PCM
- Interchip Digital Link—IDL
- General Circuit Interface—GCI

You will probably chose either an NMSI or PCM highway interface, unless you are designing an ISDN-based system. If you are designing an ISDN-based system, you will probably choose either an IDL or GCI interface. The following paragraphs discuss all the interfaces, but special attention is given to the NMSI.

NOTE

The following discussion assumes some knowledge of the interfaces. For more applications information on these interfaces, refer to 4.4 Serial Channels Physical Interface**.**

The most commonly used physical interface on the MC68302 is the nonmultiplexed serial interface (NMSI). The NMSI consists of seven basic modem (RS-232) signals: $\overline{\text{TXD}}$, $\overline{\text{TCLK}}$, $\overline{\text{RXD}}$, $\overline{\text{RCLK}}$, $\overline{\text{RTS}}$, $\overline{\text{CTS}}$, and $\overline{\text{CD}}$. Each of the three SCCs can have its own set of these signals, as shown in Figure D-21.



NMSI — Nonmultiplexed serial interface (also called the modem I/F).

Figure D-21. NMSI Pin Definitions

The other three physical interfaces, PCM highway, IDL, and GCI, are called multiplexed interfaces since they allow data from one, two, or all three SCCs to be time multiplexed together on the same pins. If a multiplexed interface is chosen, the first SCC to use that interface must be SCC1, since the three multiplexed modes share pins with SCC1 (see Figure D-22). After choosing a multiplexed mode, you can decide independently whether SCC2 and SCC3 should be part of the multiplexed interface or whether they should have their own set of NMSI pins.

If you are working in ISDN, transparent mode can be quite useful in sending and receiving transparent data over the 2B + D interface. IDL and GCI allow the SCCs to transmit and receive data on the two 64 kbps B channels and on the one 16 kbps D channel in basic rate ISDN. If you are not interfacing to a 2B + D ISDN environment, you can probably rule out using IDL and GCI.

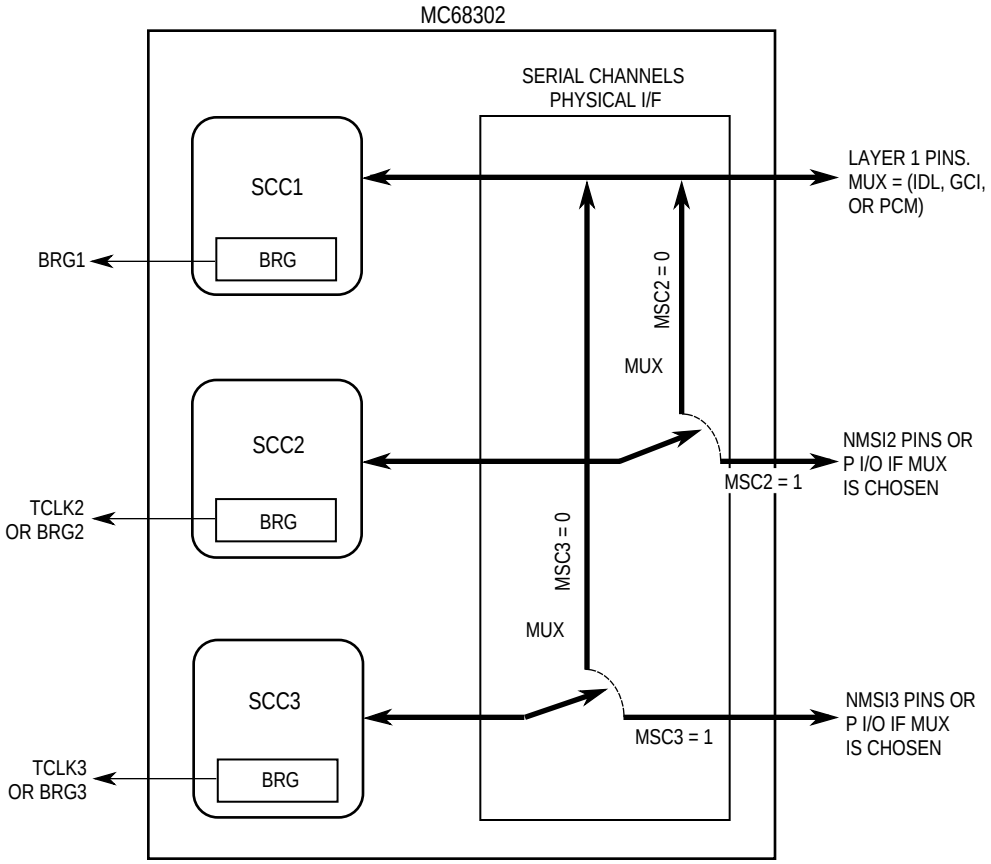


Figure D-22. Multiplexed Modes Example

PCM highway is any time-division multiplexed serial interface (i.e., data is transferred over time slots). The most common examples of a PCM highway (serial bus) are a T1 line or a CEPT line; however, there are many other possible PCM highway configurations. T1 has 24 8-bit channels and is clocked at 1.544 MHz. CEPT has 32 8-bit channels and is clocked at 2.048 MHz. In both cases, the actual rate of an 8-bit channel is 64 kbps. If more data throughput is needed, multiple 8-bit time slots can be grouped together for faster data transfer. You may wish to choose PCM highway, even if you only have one SCC using the interface (and therefore one long time slot) as will be shown.

The following examples illustrate how the different types of interfaces can be combined.

Example 1. If you need multiple transparent channels on separate physical interfaces, then all of them can be NMSI or all but one as NMSI. Thus, you can choose from the following four combinations:

1. NMSI1, NMSI2, and NMSI3
2. PCM, NMSI2, and NMSI3
3. IDL, NMSI2, and NMSI3
4. GCI, NMSI2, and NMSI3

Example 2. If you only need one transparent channel (and you had all three physical interface available), your six choices are as follows:

1. NMSI1 using SCC1
2. PCM (i.e., NMSI1 is converted into PCM pins) using SCC1
3. GCI (i.e., NMSI1 is converted into GCI pins) using SCC1
4. IDL (i.e., NMSI1 is converted into IDL pins) using SCC1
5. NMSI2 using SCC2
6. NMSI3 using SCC3

Example 3. If you need to interface one, two, or three transparent channels to a single time-multiplexed bus, then the choice is simply PCM highway using SCC1 and (either SCC2 or SCC3 or both).

Example 4. If you need to interface one, two, or three transparent channels to an lSDN basic rate bus, then the choices are as follows:

1. IDL using SCC1 and (either SCC2 or SCC3 or both)
2. GCI using SCC1 and (either SCC2 or SCC3 or both)

NOTE

The preceding four examples of physical interface combinations apply equally well to other MC68302-supported protocols such as HDLC.

Since the purposes of GCI and IDL are clear, the real challenge is choosing between NMSI and PCM. What are the advantages of PCM over NMSI? To really answer that, you will have to take a more detailed look at the timings discussed in the following paragraphs. However, one general statement can be made: PCM mode allows better control over how data is gated into and out of the SCC, but requires that data is transmitted and received simultaneously on the SCC. (There are no separate TCLK and RCLK pins in PCM mode. Instead, there is one clock pin (L1 CLK) that clocks transmit and receive data whenever the syncs are activated).

The choice of physical interface is made in the serial interface mode register (SIMODE); $0000 is the default value and sets all three SCCs to use the NMSI interface. As another example, the value $0009 chooses SCC1 and SCC2 to use the PCM mode and SCC3 to use the NMSI interface.

D.8.4 General Transparent Mode Behavior

Transparent mode is entered by selecting the BISYNC mode and setting the NTSYN bit in the SCC mode register (SCM). In most applications, it is also customary to set the EXSYN bit in the SCM as well. No other SCM bits are valid in transparent mode except REVD, which allows the bit ordering for each byte of the transmitted and received data to be reversed before sending it out or storing it in memory.

All transfers to and from memory in transparent mode are 16 bits to maximize performance. All bits in the transmit buffer are transmitted out of the SCC in transparent mode, regardless

of the physical interface configuration. Similarly, all bits in the receive buffer will be filled with real transparent data (full packing is always performed), regardless of the physical interface configuration.

If no data is available to transmit, transparent mode will transmit ones. The decision of whether to set the last (L) bit in the Tx BD is left to the user. If multiple buffers are to be sent back-to-back with no gaps in between, the L bit should be cleared in all buffers except for the last buffer. In this case, failure to provide buffers in time will result in a transmit underrun. If the L bit is set, the frame will end without error, and the transmission of ones will resume.

The transmit byte count and buffer alignment need not be even, but the SDMA channel will always read words on an even-byte boundary, even if it has to discard one of the two bytes. For example, if a transmit buffer begins on an odd-byte boundary and is 10 bytes in length (worst case), six word reads will result, even though only 10 bytes will be transmitted.

The receive buffer length (stored in MRBLR) and starting address must be even. All transfers to memory will be of word length and, unless an error occurs, a buffer will not be closed until it contains MRBLR/two words (the byte count will be equal to MRBLR). This raises an important point. Data received will only be transmitted to memory every 16 clocks. If a non-multiple of 16 bits is sent in a frame, the residue bits will not be transmitted to memory until additional bits arrive, and it will be impossible to demarcate frames unless their length is predetermined. (If a SYNC character is received with the data, the BISYNC mode can be used to receive an odd number of bytes with odd-length receive buffers and pointers allowed. (For more detailed information, refer to D.8.6 Other NMSI Modes**.)**

When the enable transmitter (ENT) bit is set, the process of polling the Tx BD begins by the RISC. The frequency of this polling is determined by the SCC's transmit clock. If the clock is stopped, no polling will occur. When the ready bit of the first Tx BD is set, the RISC initiates the SDMA activity of filling up the transmit FIFO with three words of data. Once the FIFO is full, the $\overline{\text{RTS}}$ signal is asserted, and the physical interface signals take control to determine the exact timing of the transmitted data. Once the physical interface says “go”, typically one final $FF is transmitted before data begins; however, whether $FF is transmitted depends on the mode chosen.

When the enable receiver (ENR) bit is set and 16 bits of valid data (as defined by the physical interface signals) have been clocked into the receiver, the RISC checks to see if the first receive buffer is available and, if the buffer is available, begins moving the data to it. The receive FIFO is three words deep, but a single open entry in the FIFO causes an SDMA service request. There are three types of receive errors: overrun (receive FIFO overflow), busy (new data arrived without a receive buffer being available), and $\overline{\text{CD}}$ lost (which is not possible in any example configuration discussed in this appendix). These errors are reported in the SCC event register (SCCE) or the Rx BD.

Whenever a buffer has been transmitted with the interrupt (I) bit set in the Tx BD, the TX event in the SCCE register will be set. This TX bit can cause an interrupt if the corresponding bit in the SCCM is set. Similarly, whenever a buffer has been received with the interrupt (I) bit set in the Rx BD, the RX event in the SCCE register will be set. Also, whenever a word of data is written to the receive buffer, the RCH bit is set in the SCCE.

D.8.5 Transparent Mode with the NMSI Physical Interface

NMSI has two independent data signals, TXD and RXD, and two independent clocking signals, TCLK and RCLK. TCLK and RCLK may be individually chosen to be generated internally or externally to the MC68302.

NMSI also has three control signals: $\overline{\text{RTS}}$, $\overline{\text{CTS}}$, and $\overline{\text{CD}}$. First, let's discuss their properties in general. The SCC forces $\overline{\text{RTS}}$ low when it is ready to transmit data, but the SCC waits until it sees $\overline{\text{CTS}}$ is low before doing this. After the frame has been transmitted, the $\overline{\text{RTS}}$ signal is negated (high). The $\overline{\text{CTS}}$ signal should stay low during the entire time $\overline{\text{RTS}}$ is low, or transmission is aborted and a $\overline{\text{CTS}}$ lost error is indicated in the transmit buffer descriptor (Tx BD). On the receiving side, the CD signal going low tells the MC68302 to gate data into this SCC. Once low, $\overline{\text{CD}}$ should remain low for the entire frame, or reception is terminated and a $\overline{\text{CD}}$ lost error is signaled in the receive buffer descriptor (Rx BD).

Sometimes the $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ input functions described above are not appropriate for an application. In this case, the software operation mode in the SCC mode register (SCM) can be chosen by programming the DIAG1-DIAG0 bits. In the software operation mode, as far as the SCC is concerned, $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ are always low. However, the real value of the $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ lines externally can be read in the SCCS register once the transmitter and receiver are enabled, and changes in these lines can generate interrupts via the SCCE register. Software operation mode does not affect $\overline{\text{RTS}}$ because, since $\overline{\text{RTS}}$ is an output, $\overline{\text{RTS}}$ can always be ignored by the external logic.

In totally transparent mode (and also BISYNC mode), the $\overline{\text{CD}}$ signal can become a synchronization input. When discussing the $\overline{\text{CD}}$ signal during totally transparent mode in this document, $\overline{\text{CD}}$ will be referred to as "$\overline{\text{CD}}$ (sync)". The totally transparent mode is initiated by setting the EXSYN bit in the SCM. With EXSYN set, a high-to-low transition on $\overline{\text{CD}}$ (sync) defines the start of *both transmission and reception* of transparent mode frames. Subsequent high and low transitions of $\overline{\text{CD}}$ (sync) have *no* effect on the reception of data. The only way to reinitiate the SYNC process is to issue an ENTER HUNT MODE command to the channel, and then force another high-to-low transition on $\overline{\text{CD}}$ (sync).

Figure D-23 shows the simplest NMSI transmit case. NTSYN and EXSYN are set to enable transparent mode, and the L bit is set. Software operation mode (DIAG1 = 1 and DIAG0 = 1) is chosen to eliminate using $\overline{\text{CTS}}$ to control transmission. However, since EXSYN = 1, $\overline{\text{CD}}$ becomes $\overline{\text{CD}}$ (sync), and transmission cannot begin until $\overline{\text{CD}}$ (sync) is low (which can be accomplished by grounding $\overline{\text{CD}}$ (sync)). Thus, there will only be a 1-bit delay between $\overline{\text{RTS}}$ being asserted by the transmitter and actual data being transmitted. Since the L bit is set, $\overline{\text{RTS}}$ is negated after the last byte in the frame.

In the preceding example, if multiple buffers had been ready with their L bits cleared, $\overline{\text{RTS}}$ would have remained asserted, and the next buffer's data would have begun immediately. If multiple buffers had been ready with their L bits set, $\overline{\text{RTS}}$ would have been asserted again after a delay of at least 17 idle bits on the line (the exact number of bits is load dependent).

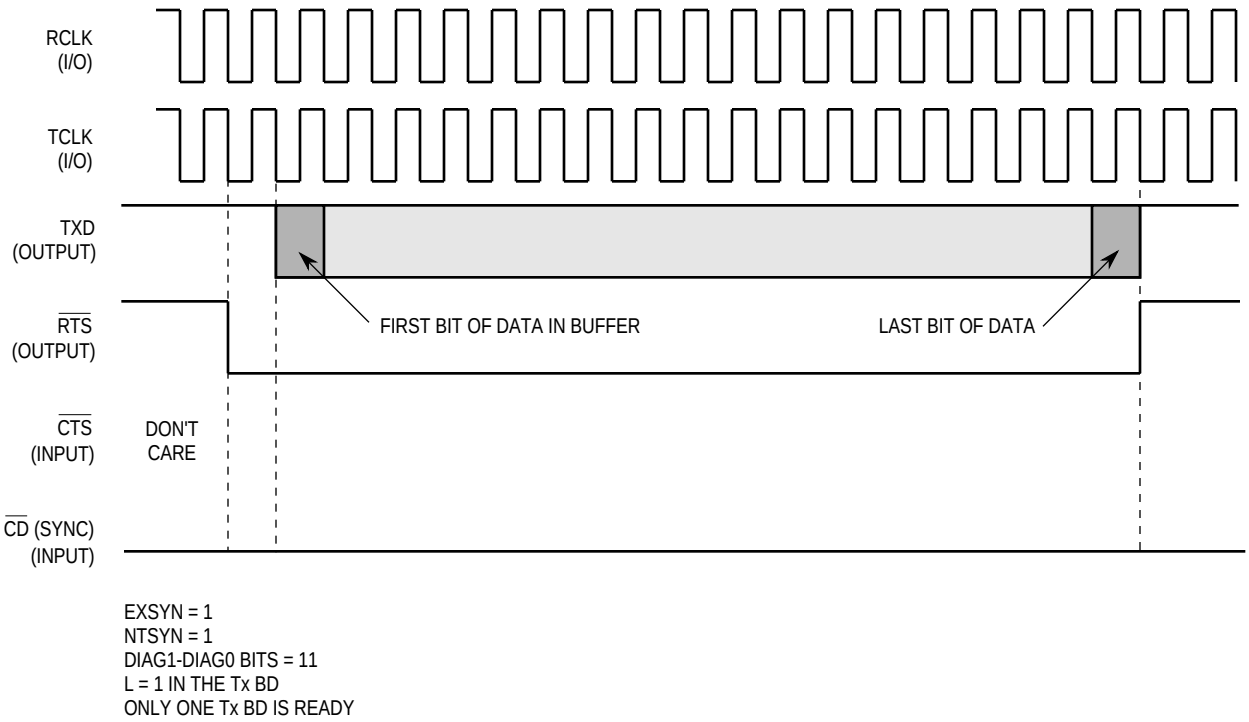


Figure D-23. Simplest Transmit Case in NMSI

Figure D-24 shows the simplest NMSI receive case. NTSYN and EXSYN are set to enable transparent mode. The choice of software operation in this case is irrelevant because the $\overline{CD}$ (sync) pin is grounded externally, causing the SCC receiver to continuously receive data. As soon as the ENR bit in the SCM is set, bits begin being received into the SCC. Data bits are sampled at every rising edge of RCLK, and no byte alignment is performed on receive. This configuration is appropriate for receiving a constant, uninterrupted stream of bit data.

In the two examples shown in Figure D-23 and Figure D-24, if the $\overline{CD}$ (sync) pin is needed as a parallel I/O line, clear the corresponding bit in the port A control PACNT) register. When the bit is set, data will transmit using the $\overline{CTS}$ and receive using $\overline{CD}$ (sync), which, in this case, is always internally asserted to the SCC.

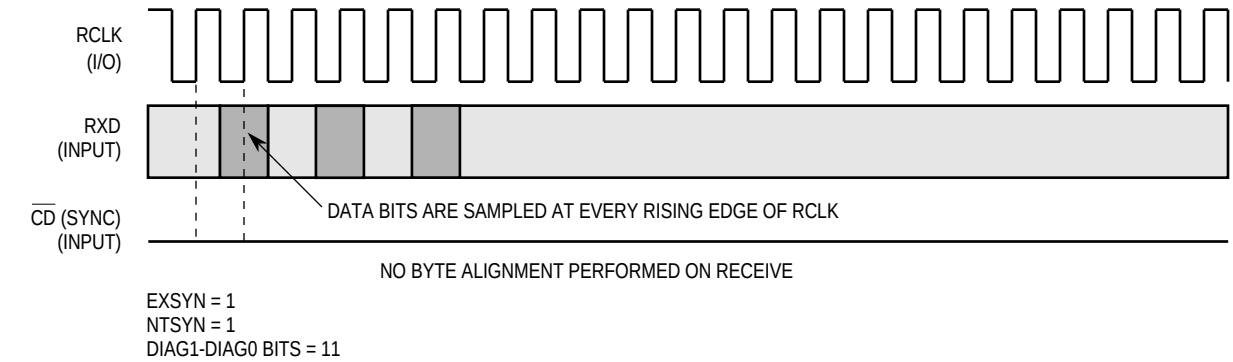


Figure D-24. Simplest Receive Case in NMSI

Figure D-25 shows how $\overline{CTS}$ can be used in the NMSI transmit case. NTSYN and EXSYN are set to enable transparent mode. Instead of software operation for $\overline{CTS}$ and $\overline{CD}$, normal (automatic) operation is chosen. $\overline{RTS}$ is asserted when the transmit FIFO is full. From then on, data is held off until $\overline{CTS}$ is sampled low. From that sample point, there is a 3.5 TCLK delay before the first bit of the data buffer is transmitted. Ones are transmitted until the first bit of the data buffer is transmitted.

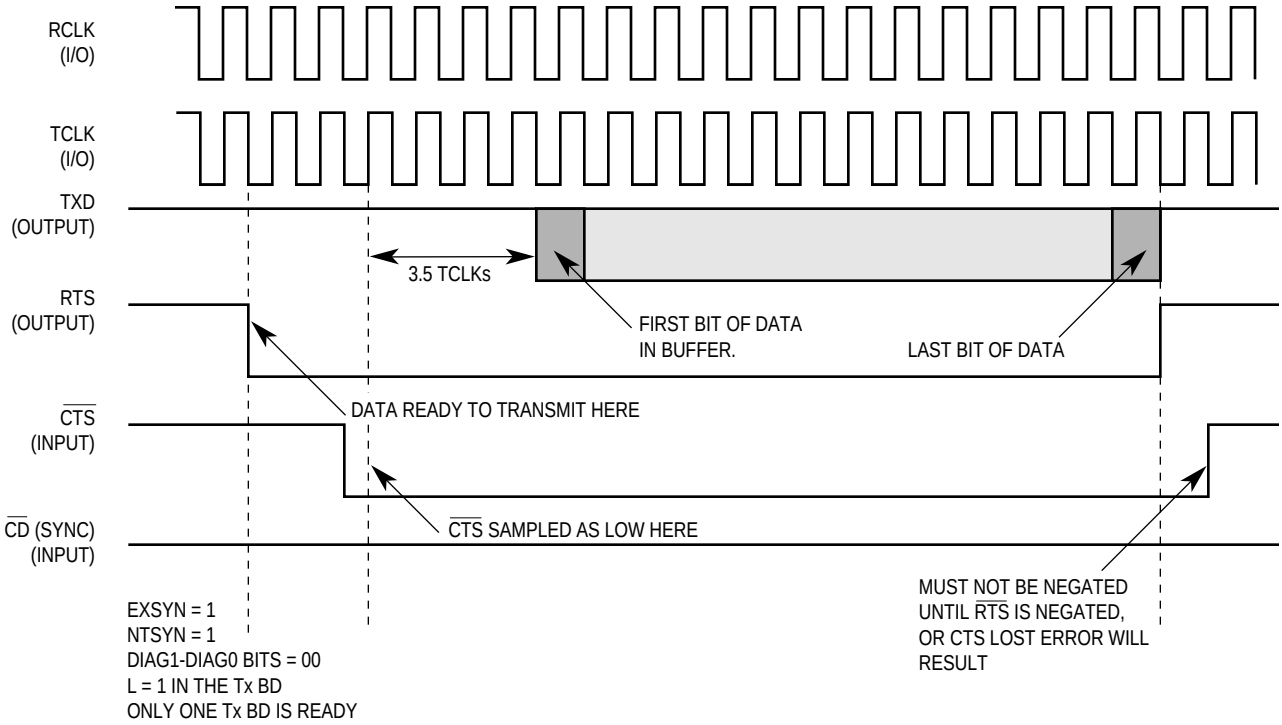


Figure D-25. Using $\overline{CTS}$ In the NMSI Transmit Case

In the case shown in Figure D-25, it is important that $\overline{CTS}$ not go high for the duration of the buffer transmission. If multiple buffers are all ready with their L bits cleared, transmission of frames will continue back-to-back. If $\overline{CTS}$ negated during any of these buffers, transmission will cease, and that buffer will report a $\overline{CTS}$ lost condition. Ones will be transmitted at that time. Once a restart transmit command is given, transmission of the next buffer can begin once $\overline{CTS}$ is reasserted.

Once $\overline{CTS}$ deasserts after $\overline{RTS}$, the $\overline{RTS}$-$\overline{CTS}$ protocol can begin again as soon as the next buffer is made ready, but a minimum of 17 idle bits will occur between frames, regardless of how soon $\overline{CTS}$ is reasserted. Remember that when EXSYN is set, $\overline{CD}$ (sync) must be low for transmission to begin. In this case, it is grounded; whereas, in the following case, EXSYN is actively switching.

Figure D-26 shows how $\overline{CD}$ (sync) can be used to control transmission. EXSYN and NTSYN are once again set to enable transparent mode, and the L bit is set. Since software operation mode (DIAG1 = 1 and DIAG0 = 1) is chosen, the $\overline{CTS}$ pin value is ignored. Once $\overline{CD}$ (sync) is latched low, data begins transmission in 6.5 TCLKs. Notice that the rising edge of $\overline{CD}$ (sync) and subsequent falling edges of $\overline{CD}$ (sync) (not shown) have no effect, since synchronization has already been achieved.

Since the L bit is set, once the frame ends, the synchronization process must occur once again. Also, to force resynchronizations instead of waiting for the transmission to finish, a STOP TRANSMIT command can be given, followed by a RESTART TRANSMIT command; however, the STOP TRANSMIT command will abort the current buffer.

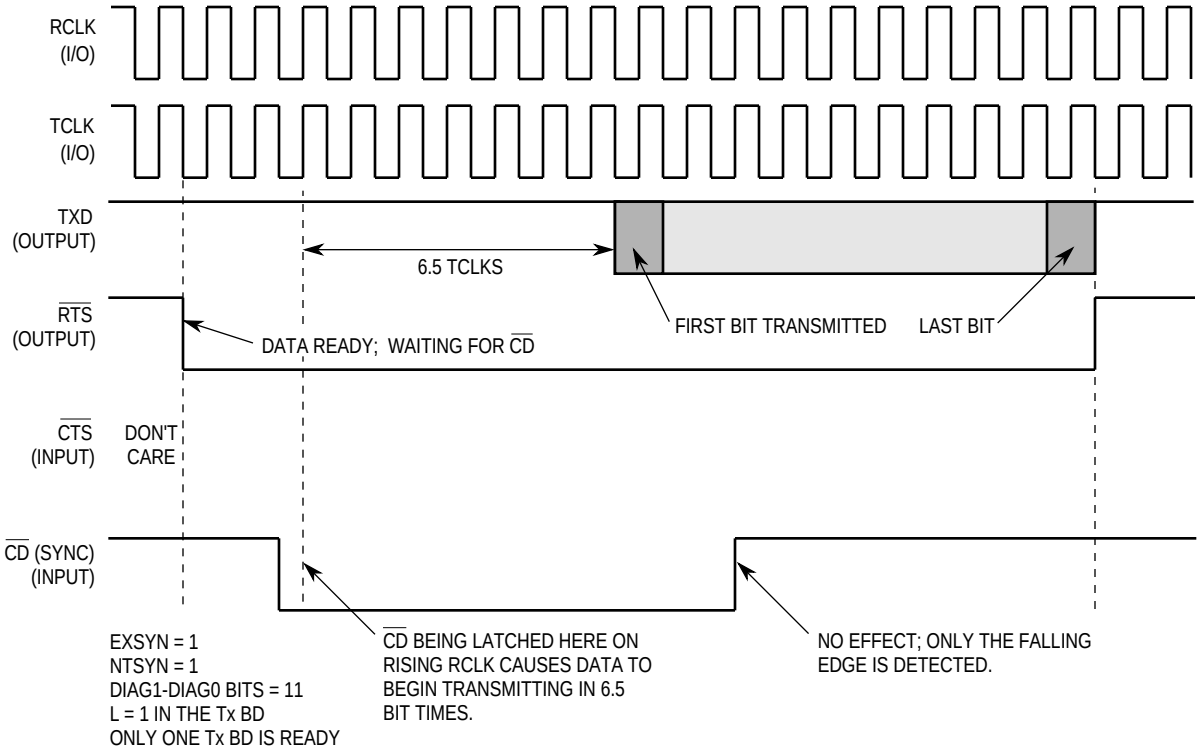


Figure D-26. Using $\overline{CD}$ (Sync) In the NMSI Transmit Case

Figure D-27 shows how $\overline{CD}$ (sync) is used in the NMSI receive case. Setting the EXSYN bit causes $\overline{CD}$ (sync) to control the reception of data. $\overline{CD}$ (sync) should be latched low on the rising clock (RCLK) of the second *bit of the frame.* (Latching $\overline{CD}$ (sync) during the 2nd bit of the frame allows external BISYNC sync detection logic, which also uses EXSYN, extra time to present the external sync to the SCC.) Once synchronization is achieved, it will never be lost unless an ENTER HUNT MODE command is given, a receive overrun occurs, or the receiver is disabled and re-enabled (ENR bit is cleared, ENTER HUNT MODE command is issued, and ENR is set). Once synchronization is lost, a new frame can be resynchronized using $\overline{CD}$ (sync).

Notice that we have been discussing the receive and transmit cases separately. The receive and transmit halves of the SCC really are separate and distinct; however, in transparent mode, the receive and transmit halves of the SCC share the $\overline{CD}$ (sync) pin, which is not true in normal NMSI.

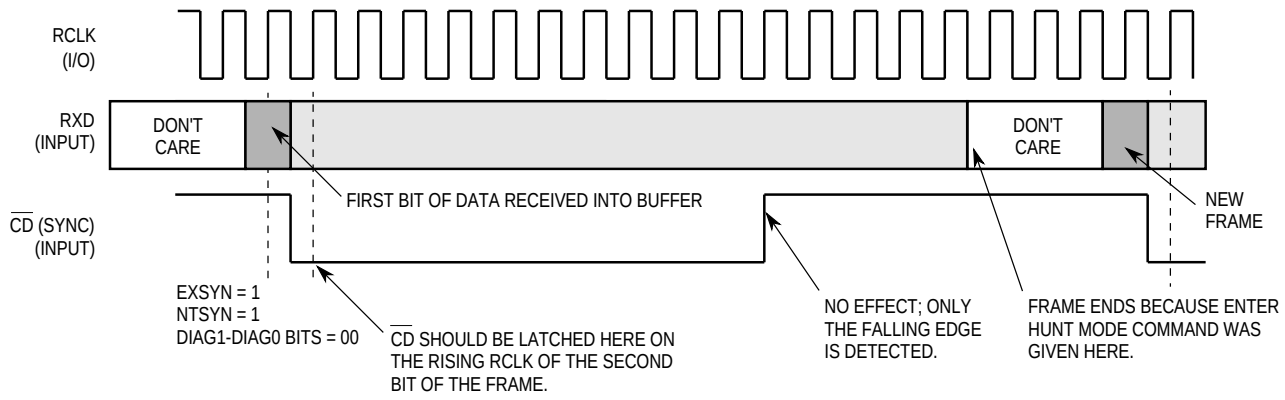


Figure D-27. Using CD (Sync) in the NMSI Receive Case

Figure D-28 shows an external loopback example which illustrates how the receive and transmit cases function together. $\overline{RTS}$ is externally connected to $\overline{CD}$ (sync), and $\overline{CTS}$ is a don't-care since software operation (DIAG1 = 1 and DIAG0 = 1) is chosen.

If you combine the 6.5 cycle TCLK delay on transmission with the $\overline{CD}$ (sync) signal being provided on the second bit of the frame (–1.5 clock delay (see Figure D-27), then the first byte stored in the receive buffer is an $FF (i.e., 6.5 — (–1.5) = 8 clock delay).

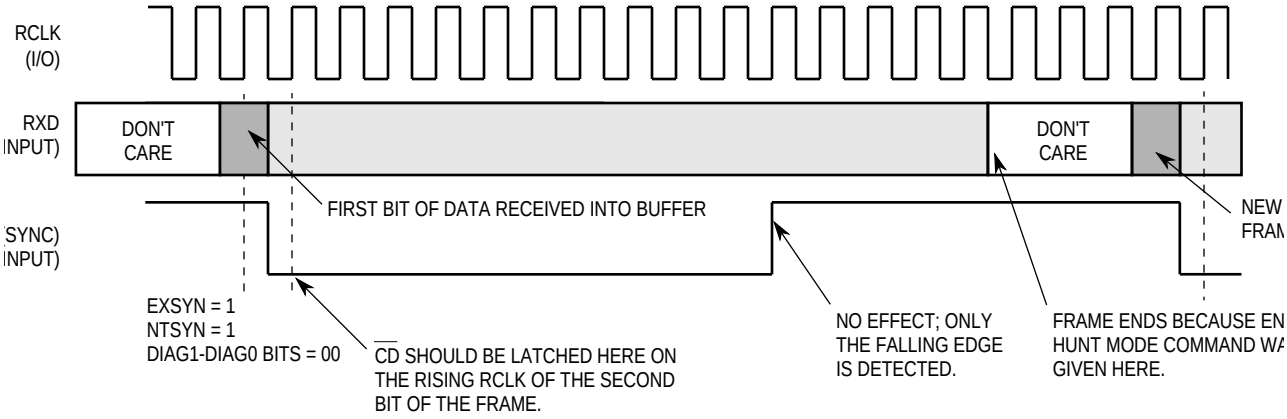


Figure D-28. External Loopback with $\overline{RTS}$ Connected to $\overline{CD}$

D.8.6 Other NMSI Modes

Two transparent mode variations are available for use with an NMSI: BISYNC mode and TRANSYNC mode. These modes are described in detail in the following paragraphs.

D.8.6.1 BISYNC MODE. You can solve the problem of demarcating frames if you can afford to exclude two special characters from the data. This, of course, is only possible when you can define the protocol at both ends of the communication link. Normal BISYNC mode can be used to do this. Normal BISYNC mode has the advantage of being able to store an odd number of bytes in the receive buffer, rather than words. A disadvantage, however, is lower serial performance.

For instance, you could create a protocol that sends and receives the following:

syn-syn-Data-Data-ETX-syn-syn-syn-syn-Data-ETX-syn-syn

where syn is a defined one-byte sync character, ETX is the one-byte control character on which to close the receive buffer (BISYNC terminology), and data is any other byte value.

On reception, the first three bytes can be stored in one receive buffer, and the next two bytes can be stored in a different receive buffer (ETX is not discarded from the receive buffer). The receiver in BISYNC mode will strip the syn bytes and use the ETX character to generate a control character interrupt to close the current receive buffer. Thus, data bytes must not contain syn or ETX characters!

On transmission, the first three bytes may be stored in one transmit buffer and the next two bytes in the next transmit buffer. At least two sync characters must be transmitted before every frame. If the next buffer is not ready immediately, additional syncs will be inserted (as shown in the preceding example by two extra syncs (for a total of four) between the two frames). If the transmitter underruns, the syn character will be inserted. One advantage of this BISYNC-type configuration is that an underrun is not fatal and has no effect on reception of correct data.

Assuming that syn = $45 and ETX = $44, registers should be set up as follows for a loopback test of BISYNC mode:

DSR = $4545—This register contains a pair of sync characters. In this register, they are termed SYN1-SYN2. They do not have to be the same byte repeated twice, although there is no particular advantage in using different bytes. Since any characters can be used as the sync characters, it is even possible to choose HDLC flags ($7E7E) if desired.

SCM = $089F—This setting selects normal BISYNC, loopback enabled, BCS disabled, and transmit SYN1-SYN2 pairs between frames. It is also possible to transmit ones between frames instead of SYN1-SYN2, but a sync character must be included at the end of each transmit buffer to ensure that the receiver sees a sync at the end of the frame.

BSYNC = $8045—This defines the underrun sync on the transmit side and ensures the receiver will not put the sync in the receive buffer.

BDLE = $00ff—Disables the BISYNC DLE function.

PRCRC, PTCRC = $0000—It is good practice to clear these even though they are not used.

Control Char Table Entry 1 = $2044—Closes the receive buffer upon reception of an ETX character.

Control Char Table Entry 2 = $8000—This setting disables control char table entries 2-8.

Note that the preceding list of registers is not meant to suggest that these registers should be written in this order. Refer to 4.5.7 SCC Initialization for the proper order.

Other registers, such as MRBLR, RFCR, TFCR, SCCM, IMR, and SIMODE, still have to be defined, but their values are of no particular relevance to the loopback operation.

In the Tx BD, the last (L) bit should be set, and the TB, BR, TD, and TR bits should be cleared. In each Rx BD, the CR bit, which indicates a bad BCS, should be ignored.

If all the frames are of a fixed length, you do not need to use ETX. Instead, disable the whole control character table, and set the MRBLR to the frame length. If MRBLR = 2, for example, then you can send and receive the following frame types:

syn-syn-Data-Data-syn-syn-syn-syn-Data-Data-syn-syn

where syn is a one-byte sync character that cannot be sent as data.

To be able to send the sync character within the data stream requires full BISYNC capabilities in a mode called BISYNC transparent, which is not discussed in this subsection.

D.8.6.2 TRANSYNC MODE. In the normal transparent mode examples discussed previously, both the NTSYN and the EXSYN bits were set in the SCM register. Also, different ways of using BISYNC mode have been described in which both the NTSYN and EXSYN bits are cleared. However, what happens if you set NTSYN and clear EXSYN? The answer is a combination of transparent and BISYNC modes that is referred to here as TRANSYNC.

On the transmission side, normal transparent operation takes place with no sync characters transmitted. On the receive side, however, reception will not be synchronized until the pattern in the DSR is matched on the line. In other words, the $\overline{CD}$ (sync) function is eliminated on transmit and is replaced with the DSR matching function on receive. Recall that $\overline{CD}$ and $\overline{CTS}$ can still control transmission and reception in TRANSYNC mode if the DIAG1-DIAG0 bits are set for normal mode and not software operation mode.

NOTE

When NTSYN is cleared and EXSYN is set, the result is normal BISYNC mode except that the external synchronization function, $\overline{CD}$ (sync), is required for proper reception. Syncs are transmitted in this mode, but are not required on receive. This is the opposite of TRANSYNC mode.

D.8.7 Gating Clocks in NMSI Mode

If the behavior of CTS and CD (sync) are not what is needed for an application, there is always the possibility of gating clocks to the SCC. The term “gating clocks” usually means providing clocks to an SCC only while it is in the act of transmitting or receiving, but at no other time. Gating clocks is a requirement in some multidrop applications and can be useful for many special applications. Gating clocks is only possible if the clocks are inputs to the SCC since the internal SCC baud rate generators do not support gating clocks.

The gating of clocks can provide extra control over the transmission and reception of data, albeit with extra logic external to the MC68302. The SCCs are designed with static logic; thus, the clock signal may be held in a constant high/low state for any period of time. Whenever clocks are provided externally (and especially when they are gated), care should be taken to avoid glitches, excessive ringing, and very long rise/fall times in a very noisy environment. If the minimum clock high/low time is violated, erratic operation can result, which can cause an SCC to immediately transition to an error state such as underrun or overrun.

A CP reset in the command register (CR) may be required to continue operation with an SCC that has underrun or overrun.

There are two special points worth noting about gating clocks:

1. When gating clocks to the transmitter, be aware that clocks are required to the transmitter for the Tx BD to be polled and the transmit FIFO to be initially filled with data. You cannot enable the transmitter and begin providing clocks 1 ms later and expect the transmit data to go out on the first clock provided. Neither is there a set number of clocks to count on since the number of clocks is dependent on the data rate, the RISC loading, the system bus latency, and the exact time in relation to the RISC poll that the Tx BD ready bit was set. Therefore, it is best to use $\overline{\text{CTS}}$ or $\overline{\text{CD}}$ (sync) to gate data out of the chip while initially providing a number of transmit clocks.
2. If you cleared the L bit in the Tx BDs, then you can freely gate the clocks at any time (for instance between buffers) and pick up right where you left off. If each frame is an individual entity and the timing of the frame to follow it is unknown, then the L bit should be set, and after $\overline{\text{RTS}}$ is negated by the MC68302 and $\overline{\text{CTS}}$ is negated by your external logic, you may disable clocking the transmitter. However, be sure to give the transmitter more clocks (20 should be enough in most systems), before expecting the $\overline{\text{RTS}}$ pin to go low again, signifying that data is ready to be transmitted. The $\overline{\text{CTS}}$ or $\overline{\text{CD}}$ (sync) pins may once again be used to control the exact timing of the subsequent data transmission.
3. Data may be freely gated to the receiver when the CD (sync) signal is used. However, remember that nine extra clocks will need to be provided after the frame (which must be a multiple of 16 bits) to ensure that the receive data passes through the SCC to the FIFO where it can be extracted by the SDMA and moved to memory.

Providing less than nine extra clocks harms nothing, but the last word of the frame will not be received until the next frame begins. If you need to get this last word of data into the buffer without waiting for the next frame, an ENTER HUNT MODE command may be given. This command will clear out the residue bits, allowing a new synchronization to occur for the next frame. The ENTER HUNT MODE command should be given between the 9th and 16th serial clock, after the last bit of the frame has been clocked into the MC68302 and at least three serial clocks before the next CD (sync). Serial clocks do not need to be running while this command is executed.

If normal $\overline{\text{CD}}$ is used, as opposed to $\overline{\text{CD}}$ (sync), the $\overline{\text{CD}}$ pin should remain asserted for at least six additional clocks after the last bit of the frame is clocked in. After $\overline{\text{CD}}$ is negated, an additional three receive clocks should also be provided before the clocks are stopped. Additional clocks after $\overline{\text{CD}}$ is negated force ones into the SCC, regardless of the state of the RXD pin. When you want a new frame to be clocked into the MC68302, $\overline{\text{CD}}$ should be asserted one clock prior to the first bit of the new frame. (Note that when normal $\overline{\text{CD}}$ is used, the frame length need not be a multiple of 16 bits).

D.8.8 Using Transparent Mode with PCM Highway Mode

In PCM highway mode, the physical interface controls the multiplexing of data from one, two, or all three SCCs onto the NMSI1 pins. In PCM highway mode, the NMSI1 pins take on the names and functions shown in Table D-4.

Table D-4. PCM Highway Pin Names and Functions

Name	Function	Input/Output
L1RXD	Receive Data	Input
L1TXD	Transmit Data	Output
L1CLK	Receive and Transmit Clock	Input
L1SY0	Sync Signal 0	Input
L1SY1	Sync Signal 1	Input
RTS1-RTS3	Three $\overline{\text{RTS}}$ Signals	Outputs

Notice that there is only one clock signal for PCM highway. This clock functions as both a receive and transmit clock. Thus, if receive data needs to be clocked into the MC68302 at a different time or speed than transmit data is being clocked out, then PCM highway is not an appropriate interface and NMSI should be used instead. Also, L1CLK is an input only. To drive it from the MC68302, a timer or baud rate generator output pin must be externally connected to the L1CLK pin.

The two sync signals are L1SY0 and L1SY1. They are also inputs to the MC68302. Together, they select one of three PCM channels to which data is routed or select no channel (see Table D-5).

Table D-5. PCM Highway Channel Selection with LISY0 and L1SY1

L1SY1	L1SY0	Selected Channel
0	0	No Channel Selected
0	1	PCM Channel 1 Selected
1	0	PCM Channel 2 Selected
1	1	PCM Channel 3 Selected

A PCM channel can be routed to any SCC, as selected in the SIMODE register. This routing ability gives one extra layer of indirection; thus, hardware which must generate the L1SY1 and L1SY0 signals externally does not have to be modified if a change in the SCC data routing is required.

Two different modes exist for using the L1SY1 and L1SY0 pins: one-clock-prior mode and envelope mode. In one-clock-prior mode, the sync signals go active for a single clock period prior to an 8-bit time slot. In envelope mode, the sync signals go active on the first bit of the time slot and stay active for the entire time slot. Thus, envelope mode is more general since it allows time slots to be between one and N bits long. Figure D-29 illustrates the operation of envelope mode. The three PCM channels defined in the figure show some of the flexibility available in the PCM mode.

The PCM highway interface has three $\overline{RTS}$ signals. One of these signals is asserted when an SCC wants to transmit over the PCM Highway just like in NMSI mode), and stays continuously asserted until the entire frame is transmitted (regardless of how many time slots the transmission takes). Which $\overline{RTS}$ signal asserts depends on which SCC is transmitting; there is one $\overline{RTS}$ signal for each SCC. Notice, however, that there is no $\overline{CTS}$ signal, so there is nothing to hold off the transfer. If the $\overline{RTS}$ signals are not needed, they can be ignored or reassigned as parallel I/O lines.

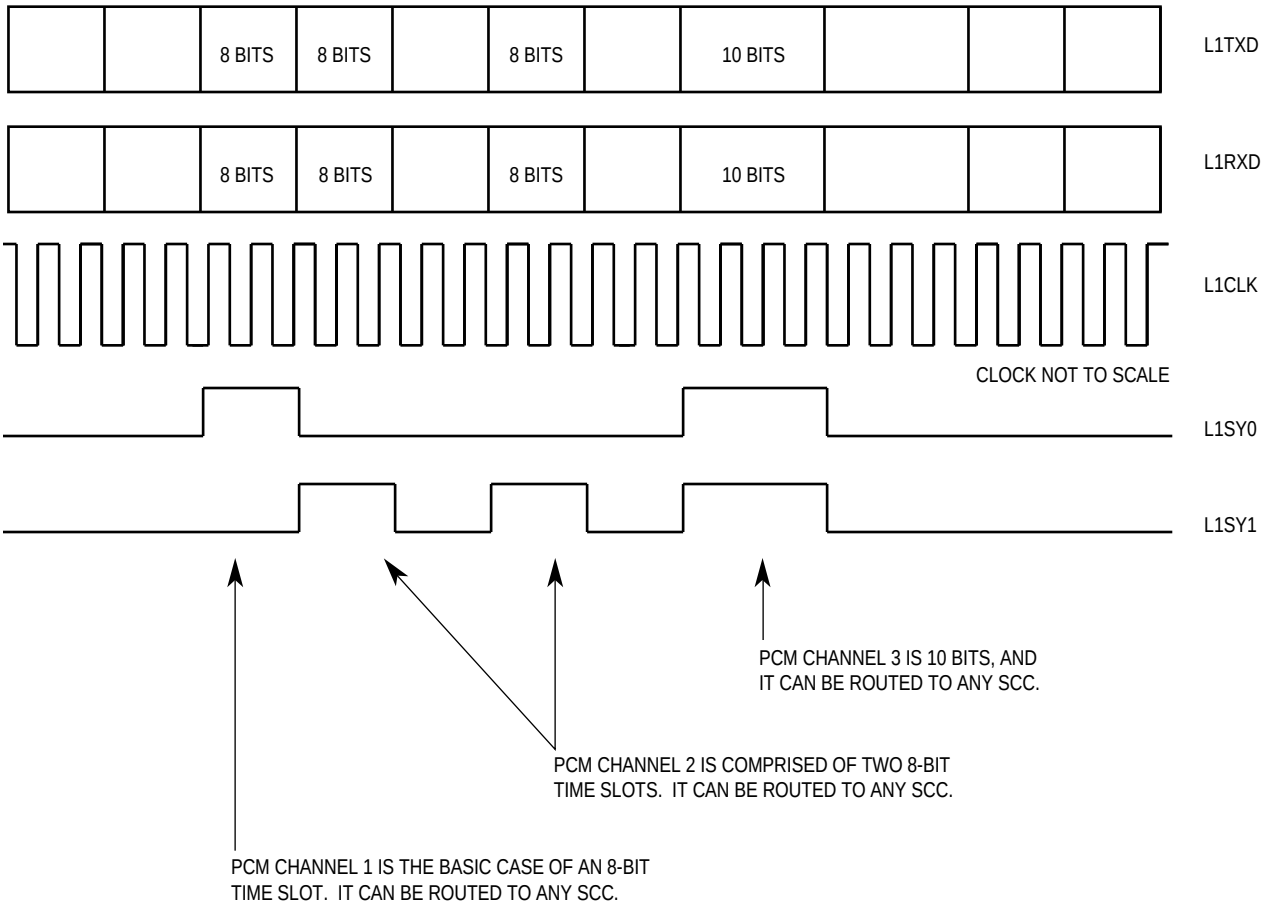


Figure D-29. Routing Channels in PCM Envelope Mode

What other signals are missing from PCM mode? First, there is no $\overline{CD}$ signal for the receiver. The receiver is enabled whenever the ENR bit is set. However, you could say there is a $\overline{CD}$ (sync) of sorts that is implemented with the L1SY1 and L1SY0 pins. Two pins are used since not only is the timing important, but also the selection of the PCM channel as well.

The way transparent mode works with a PCM highway interface is very similar to the operation of the gated clocks example discussed previously. Whether or not a time slot environment is present, PCM mode gives greater control over what intervals transparent data can be transmitted and received. However, in PCM mode, the clocks are gated by the physical interface on the MC68302 as opposed to external hardware.

The L1SY1-L1SY0 signals determine 1) when clocks are sent to the particular SCC and 2) when the synchronization signal is sent to the SCC. If the L1SY1-L1SY0 signals are not active, then the SCC is not being clocked. The rising edge of L1SY1-L1SY0 starts the clocking and sends an internal synchronization pulse to the SCC. These facts are very important in determining when the first byte of real data from a buffer will be transmitted onto the PCM.

In transparent mode, if data is not ready to transmit, $FFs will be sent during the time slot. Once data transfer begins, data will be clocked out during every clock of the time slot. When the time slot ends, the SCC will wait without being clocked until the next time slot arrives. Similarly, on the receive side, data and the clock will only be presented to the SCC when that SCC's time slot is active.

When using transparent mode with PCM, it is often of interest to know exactly when the very first buffer of transmit data will go out. The $\overline{\text{RTS}}$ signal gives an important clue here. Once the $\overline{\text{RTS}}$ signal for an SCC is asserted, the next rising edge of the L1SY1-L1SY0 pins for this channel (i.e., the SCC's next time slot) will begin clocking out the following pattern:

$FF, data1, data2, data3, . . .

where data1 is the first byte of data stored in the transmit data buffer. For example, if the PCM was configured with individual 8-bit time slots for this SCC, $FF would be clocked on the first time slot, data1 on the second, etc. **It is assumed in that this buffer in this example does not immediately follow the previous buffe**r. A string of buffers with their L bits cleared will follow each other immediately without any delay—only the first will have this delay.

From the time the ENT bit is set and a buffer is ready to transmit, it can take a number of serial clocks (usually less than 36) for $\overline{\text{RTS}}$ to be asserted (it could be more for higher data rates). Thus, this clock delay must be taken into account if data transmission delays need to be consistent. The delay can be accounted for by synchronizing the setting of the ENT bit with the time slot itself. If the time slot is long, it should be sufficient to set the ENT bit before one time slot to guarantee data transmission during the next time slot (see Figure D-30). The algorithm can work as follows:

1. After the last buffer is transmitted, give STOP TRANSMIT command.
2. Clear ENT.
3. Give RESTART TRANSMIT command.
4. Set ready bit of next Tx BD to transmit.
5. Generate interrupt to MC68302 on falling L1SY1 /L1SY0 pin.
6. Now that time slot is inactive, set ENT bit.

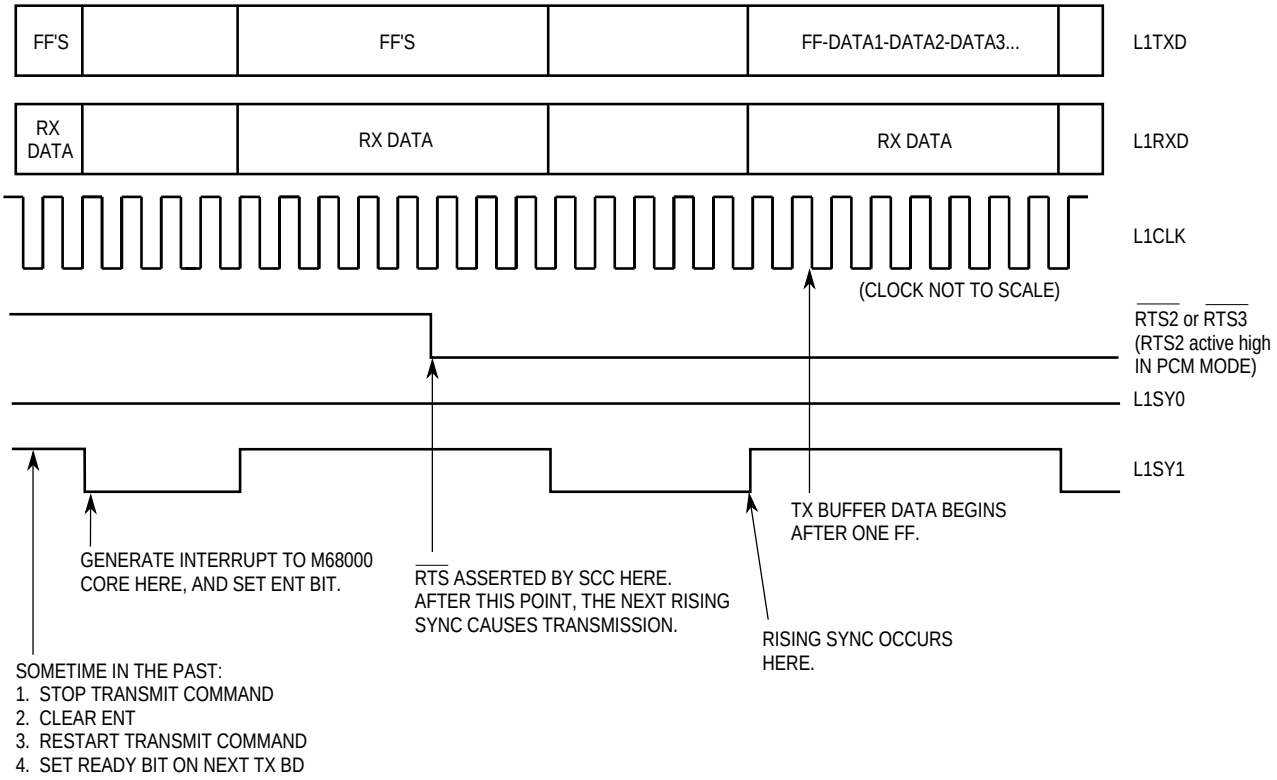


Figure D-30. PCM Transmission Timing Technique

A synchronized setting of the ENT bit causes the transmit FIFO to be filled in a fixed time if the following two conditions are true: 1) the SDMA is the highest priority bus master in the system (i.e., there is no external bus master during this period) and 2) the SCC needed for the transmission is the only SCC currently being used. Both of these factors become negligible if the serial clock rate is much slower than the system clock rate (e.g., a 1 to 50 ratio). (A slow serial clock rate means that deviations are much less than a serial bit time and have no effect on transmission delay.)

The preceding example showed ENT being set before the time slot. If there are more than 36 serial clocks following the setting of the ENT bit, it is possible to set the ENT bit during the time slot and see the same behavior. The assertion of the $\overline{\text{RTS}}$ signal can be used to verify that a sufficient number of clocks occurred after the setting of ENT.

In the transparent operation, assertion of $\overline{\text{RTS1}}$ is slightly different from that of $\overline{\text{RTS2}}$ and $\overline{\text{RTS3}}$. The description of $\overline{\text{RTS}}$ in Table D-4, the text on D-73, Figure D-29, and the text on D-74 is correct for $\overline{\text{RTS2}}$ and $\overline{\text{RTS3}}$, but not exactly correct for $\overline{\text{RTS1}}$. $\overline{\text{RTS1}}$ has the opposite polarity in PCM mode. $\overline{\text{RTS1}}$ goes low when SIMODE is programmed as the PCM mode, and then goes high when the SCC is about ready to transmit.

If the time slots are not long enough to guarantee transmission after the second time slot, then the synchronized setting of the ENT bit should at least guarantee a fixed delay to the start of data. In this case, there will be additional time slots with $FF data until the data1 byte is transmitted.

There is nothing to synchronize on the receive side. As soon as the ENR bit is set, the first time slot for this SCC will begin the reception process. As with the NMSI mode, a word is not written to the buffer until the 9th clock after the serial clock that clocked in the last bit of this word (see D.8.7 Gating Clocks in NMSI Mode)**.** Recall that clocks can only be counted *during* time slots.

D.8.9 PCM Mode Final Thoughts

Since all synchronous protocols work with PCM mode, it is possible to use the regular BI-SYNC mode to send syncs in the data stream, as described earlier.

When using totally transparent mode with PCM, both the NTSYN and EXSYN bits should normally be set. The DIAG1-DIAG0 bits can be set for either normal mode or software operation with no difference in behavior.

PCM mode does affect the SCCS register. In the SCCS register, the $\overline{\text{CD}}$ and $\overline{\text{CTS}}$ bits are always zero when the ENR and ENT bits, respectively, are set. The ID bit is not valid in transparent mode, regardless of the physical interface chosen.

Care should always be taken to avoid glitches or ringing on the L1CLK line. If a glitched or ringing L1CLK line causes an extra clock to be inserted during a time slot, there is no way to resynchronize the byte alignment in envelope mode until the ENT synchronization algorithm described previously is followed. This potential problem does lead to one slight advantage of the one-clock-prior method over the envelope sync. With the one-clock-prior method, it is more likely that the glitched clock will only misalign the transfer/reception of a single byte of data, rather than the whole data stream. (However, this cannot be guaranteed—predicting device behavior out-of-spec is extremely difficult.)

D.8.10 Using Transparent Mode with IDL and GCI

Transparent mode can be freely used with the ISDN physical interfaces. Using transparent mode with the ISDN interfaces is especially useful in the B-channels, since the D channel LAPD protocol is typically supported with the SCC in HDLC mode. Transparent data may be sent and received over the 64-kbps B1 channel, the 64-kbps B2 channel, a combined B1-B2 channel with a 128-kbps bandwidth, or subportions of either the B1 or B2 channel or both. (The desired subportions are defined in the SIMASK register.)

With the ISDN interfaces, as with the other types of interfaces, if the SCC is not transmitting data, it will transmit $FFs. If the NTSYN and the EXSYN bits are set in the SCM, data will be byte-aligned within the B1 or B2 channels. Thus, it will only be transmitted once the SCC transmit FIFO is filled and the *beginning* of the B1 or B2 channel occurs.

A special case occurs when the B1 and B2 channels are combined into a single 128-kbps channel. In this case, although data will only appear on byte boundaries, the transmit buffer's data could begin in either the B1 or B2 channels, depending on the timing involved. If this is a problem, the following rule may be observed. If the ENT bit is set at a consistent time during the GCI/IDL frame and if ready bit of the Tx BD is set at a consistent time relative to the GCI/IDL frame (preferably before the ENT bit is set), a consistent starting point of byte alignment (either B1 or B2) can be obtained. If data is then transmitted in a continuous

stream with the L bit in all Tx BDs cleared, then the byte alignment timing will remain constant.

D.8.11 Initializing Transparent Mode

Full examples of the assembler code required to initialize the HDLC and UART protocols are given in D.3 MC68302 Buffer Processing and Interrupt Handling and D.4 Configuring A Uart on the MC68302. A transparent mode initialization follows the same flow as these subsections except that different values would be used. The HDLC and UART examples also show writing of the BAR and full configuration of the interrupt controller to allow SCC interrupts, etc., which are not duplicated here.

The following example shows a step-by-step list of the SCC-related registers as they would be initialized to create a transparent SCC2 channel in the NMSI mode. The registers in this example are configured for external loopback with TCLK2 externally connected to RCLK2, TXD2 externally connected to RXD2, $\overline{\text{CTS2}}$ a don't care, and $\overline{\text{RTS2}}$ externally connected to $\overline{\text{CD2}}$ (sync). The functionality of this configuration is the same as that shown in Figure D-28. This example may be easily checked on the ADS302 board, either with the menu interface software already on the ADS302 board or with user-written software downloaded to the ADS302 board. The external connections can be made by placing three jumper cables on row B of the serial bus connector P8: B5-to-B6, B7-to-B8, and B10-to-B11.

1. To use SCC2 in the NMSI mode, we need to chose NMSI2 pins instead of parallel I/O pins. To do this, we write a one to the PACNT register in every bit position that we want an SCC pin to be active. For this example, we will assume all seven NMSI2 pins are active.
   PACNT = $xx7F
2. The SIMODE register is set to its default setting. This configuration chooses NMSI mode on all three SCCs. Actually, all we need is that NMSI mode be selected for SCC2.
   SIMODE = $0000
3. The SCON register configures the clocking options. Here we chose to generate about a 65-kHz clock on the TCLK2 pin with the internal baud rate generator. RCLK2 will take its input externally; thus, we connect the TCLK2 pin externally to RCLK2. SCON2 = $1200
4. The setting shown for SCM2 sets the EXSYN and NTSYN bits, sets the DIAG1-DIAG0 bits for software operation, and sets the protocol to BISYNC (which is actually transparent since the NTSYN bit is set).
   Since we are implementing an external loopback with the MC68302, the DIAG1-DIAG0 bits are *not set* for loopback mode. Setting the DIAG1-DIAG0 bits for loopback mode causes *internal* loopback. (To implement an internal loopback, externally connect only $\overline{\text{RTS2}}$ to $\overline{\text{CD2}}$ (sync), set SCON2 to $0200, and SCM2 to $6013. Later, the very last step is to set SCM2 to $601 F. With internal loopback, RCLK and TCLK should be directly supplied with the same clock source—either both from the internal baud rate generators or both from the same externally generated clock source.)
   SCM2 = $6033
5. The DSR2 does not need to be written and can be left at its default value since we are

using transparent mode with the EXSYN bit set in SCM2.
DSR2 = $7E7E

6. Setting SCCE2 to $FF clears out any current status in the event register. SCCE2 = $FF
7. You must indicate in the SCCM2 register which (if any) transparent events you wish to cause interrupts. Bit 5 should be set to zero. Bit 3 is only valid in BISYNC mode and has no meaning in transparent mode. All other bits are valid. For this example, we will disable interrupts, but all events will still be set in SCCE2.
SCCM2 = $00
8. Setting IMR to $0400 allows interrupts from SCC2 to be enabled in the interrupt controller; however, since SCCM2 = $00, any SCC2 interrupt requests are prevented from reaching the interrupt controller, and this step has no effect.
IMR = $0400
9. Initialize the receive and transmit function codes to 000, and set the receive buffer size to 10 (hex) bytes. These are the general-purpose parameter RAM values for SCC2.
RFCR = $00
TFCR = $00
MRBLR = $0010
10. Initialize the BISYNC parameters. These parameters do not involve transparent mode; however, it is a good idea to initialize them in case BISYNC mode is ever accidentally entered by clearing the NTSYN bit in SCM2. The values for BSYNC and BDLE are arbitrary and were chosen so that the two registers have different values. The control characters table is disabled for good measure, although this too is not used in transparent mode.
PRCRC = $0000
PTCRC = $0000
PAREC = $0000
BSYNC = $0033
BDLE = $0044
CHARACTER1 = $8000
11. The Tx BD buffer starts at address $30000. It is 18 (hex) bytes long. (Notice that the MRBLR value equal to $0010 does not restrict the transmit buffer size.) The status $D800 says that the buffer is ready and in external RAM. Since the I bit is set, the TX bit in the SCCE2 register will be set upon completion, and this is the “last” buffer in this transmission. The next Tx BD is set up so that it is not ready; transmission will halt after one Tx BD.
Tx BD = $D800 $0018 $0003 $0000
Tx BD = $5800 $xxxx $xxxx $xxxx (This Tx BD is not yet ready.)
12. Two empty Rx BDs are needed to receive the transmit frame. Both are currently empty, and data is to be stored in external RAM buffers. Since the I bits are set, the RX bit in the SCCE2 register will be set when each buffer is filled with data. The third Rx BD is not ready yet. If it was ready, it would be filled with all $FFs (idles) after the first two buffers were filled.
Rx BD = $D000 $0000 $0004 $0000
Rx BD = $D000 $0000 $0004 $0010

Rx BD = $5000 $xxxx $xxxx $xxxx (This Rx BD is not yet available.)

13. The final change is to set the ENT and ENR bits in the SCM2 register, causing the transfers to begin.
    SCM2 = $603F

With the above configuration, the data in the receive buffers will be as follows:

Buffer1: $FF, data1, data2, ..., datae, dataf.
Buffer2: data10, data11, ..., data17, data18, (and then 7 $FF bytes).

Four events in the SCCE2 event register will be set:

—RX—One or more receive buffers have been used (in this case two).
—TX—One or more buffers have been transmitted (in this case one).
—BSY—Receive data was discarded due to lack of receive buffers (occurred because the third Rx BD was not empty).
—RCH—A word of data has been written to a receive buffer (occurred each time a word was written).

D.8.12 Special Uses of Transparent Mode

The following paragraphs discuss two special cases where transparent mode can be used to extend the capabilities of the MC68302 UART mode.

D.8.12.1 5- OR 6-BIT UART. One special protocol of note that can be accomplished with transparent mode is the building of a 5- or 6-bit UART. The UART on the SCCs offers 7- and 8-bit modes only.

A 5- or 6-bit UART can be accomplished with software and the transparent mode. Software is responsible for inserting and deleting start and stop bits; the transparent mode provides oversampling.

Select transparent mode for 8x the desired bit rate. For every bit of data to transmit, write a byte of data to memory. A character will then be encoded as one byte of zeros for the start bit, a byte of either zeros or ones for every bit in the character, and a byte of ones for each stop bit. When there is no data to send, the transmitter will send out ones during the idle period if the buffer had its L bit set.

Reception is more software intensive. The data is received at 8x the desired rate, and the software must extract the start, stop, and character bits from the data stream. There is no easy way to “byte align” the received data to byte boundaries in memory (as explained in D.8.5 Transparent Mode with the NMSI Physical Interface) without some added external hardware.

D.8.12.2 SYNCHRONOUS UART. A synchronous UART may also be built with transparent mode. As with a 5- or 6-bit UART, all data including start and stop bits must be placed into the transmit buffer and the true data extracted from the raw transparent data that includes start and stop bits. (V.14 applications would also require the detection of deleted stop bits.)

A synchronous UART built with transparent mode has one simplifying difference from a 5- or 6-bit UART. Since a clock is provided with each bit, the transparent mode can be clocked at 1x the data rate, reducing the transparent bandwidth and the buffer memory requirements. The only disadvantage is that the M68000 core must build the transmit buffer up a bit at a time using bit instructions.

D.8.13 SCP as a Transparent Mode Alternative

One often overlooked feature of the MC68302 is the SCP. If your application needs byte-at-a-time transparent mode operation, the SCP may fit the bill. The SCP, a subset of the Motorola synchronous SPI protocol interface, transmits and receives bytes in a transparent mode.

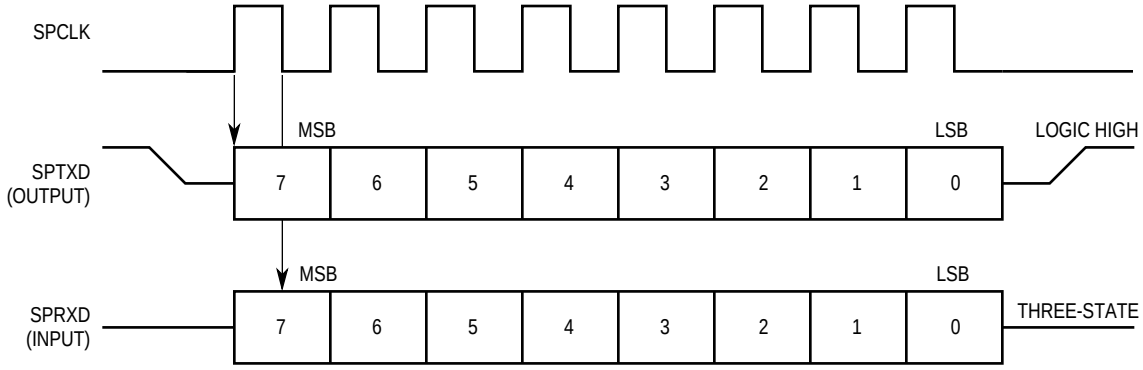
Each time the M68000 core sets the start bit in the SCP, one byte of data is shifted from the SCP buffer descriptor out on the SPTXD pin (see Figure D-31). At the same time that transmit data is being clocked out, receive data is being clocked into the MC68302.

The advantages of using the SCP instead of an SCC in transparent mode are simplicity and the saving of SCCs for other functions. There are two disadvantages, however. First, the M68000 core must be individually involved in each byte transferred (an interrupt may be generated per byte); thus, data cannot be sent back-to-back without at least some delay. Second, the SCP functions in a clock master mode only; therefore, the device communicating with the SCP must be able to accept an external input clock.

It is possible for the SCP to interface externally to one of the MC68302's SCCs. For instance, this type interface could be used to convert HDLC-encoded data from a serial format to a parallel format so that it can be moved over the M68000 bus.

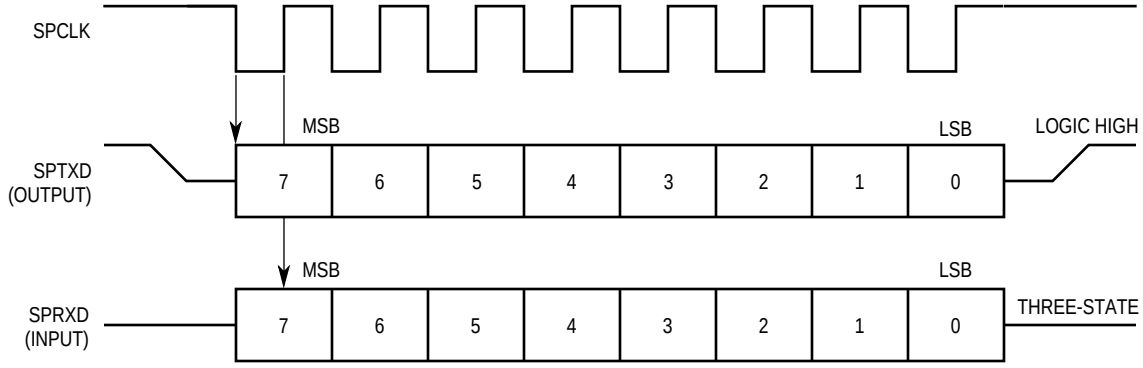
D.8.14 Transparent Mode Summary

The totally transparent mode on the MC68302 can be used in many ways. Once the desired physical interface is chosen (NSMI, PCM, IDL, or GCI), the vast number of possibilities can begin to be narrowed down. These possibilities were described in D.8 Using the MC68302 Transparent Mode**,** with emphasis on the NMSI and PCM modes. A step-by-step register initialization was given for a transparent loopback application on SCC2. It was also stated that occasionally BISYNC mode can be used in place of an SCC in transparent mode.



NOTE: Transmitted data bits shift on rising edges; received bits are sampled on falling edges.

(a) CI = 0



NOTE: Transmitted data bits shift on falling edges; received bits are sampled on rising edges.

(b) CI = 1

Figure D-31. SCP Timing

D.9 AN APPLETALK® NODE WITH THE MC68302 AND MC68195

The following paragraphs describe a hardware design that uses the MC68195 LocalTalk Adaptor (LA) to interface the MC68302 to AppleTalk. The LA is designed to work directly with the MC68302 for this purpose. The design is also suitable to those wishing to build a proprietary HDLC-based LAN.

The design as shown works with a set of LocalTalk chip drivers, that allow frames to be sent and received on the network. If these drivers are to be used, the interface between the LA and the MC68302 must be identical to that shown in Figure D-32.

D.9.1 Overview of the Board

The board interfaces two of the MC68302 SCCs to the AppleTalk LAN. SCC1 and SCC2 are used for this example, but any of the three SCCs could have been originally chosen for use with the board. The MC68195 LA provides the FM0 encoding/decoding and digital phase-locked loop functionality. It also provides a direct interface to the MC68302 and the line driver/receiver chips, as shown. The LA input clock is required to be 10x the desired data rate. Thus, for AppleTallk, a 2.304-MHz crystal was used.

The physical portion of the board was modeled after the Macintosh™ Plus serial interface. Thus, the RS-422 driver/receiver function was implemented with the 26LS32 receiver and 26LS30 line driver. The connectors used were the standard mini-DIN 8, which is the same as those used in the Macintosh. The connections to this connector followed the Macintosh Plus serial interface diagram, except that HSKo (pin 1) was simply pulled high through a 100 Ω resistor. This does not inhibit LocalTalk functionality.

D.9.2 Important Side Notes

The reset circuit chosen was a simple RC delay. Normally, the reset function would be part of the system reset circuitry, but this function was not available through the original connector on the ADS302 board, so it was built on the LA board. The LA has an internal Schmitt trigger on the reset input to facilitate this configuration.

Note that there is a pullup on the SCC2 $\overline{\text{RTS2}}$ pin because SCC2 on the MC68302, unlike SCC1, has its pins multiplexed with parallel I/O pins. These pins default to the input state upon reset. If this pullup is omitted, the $\overline{\text{RTS2}}$ pin might be low between the moment of reset and the moment at which the software configures this pin to the $\overline{\text{RTS2}}$ function (i.e., writing the PACNT register). During this window of time, a low value on $\overline{\text{RTS2}}$ would cause the LA to illegally transfer out onto the LocalTalk network. The pullup safely avoids this problem. A pullup is not needed on the MC68302 $\overline{\text{RTS1}}$ or $\overline{\text{RTS3}}$ since they reset to the inactive (high) state.

Five parallel I/O signals (PA7-PA11) are used to configure the LA into various loopback, bypass, or clock enable modes, making it easy for the MC68302 to put the LA into various modes for testing. In a final implementation, some of these signals could be pulled directly high or low. As previously described, these signals float until initialized in software by the MC68302; however, since these briefly floating inputs do not cause a problem in this system, pullups and pulldowns were not added.

The channel enable signals ($\overline{\text{CHEN}}$) were pulled continuously high in this application since there was no need to disable LA operation. In a final system, these could be easily connected to MC68302 I/O pins as needed.

·The general-purpose inputs (GPI) were simply pulled high fort his example. They could have been used to support asynchronous operation over the mini-DIN 8 connector, if desired.

AppleTalk is a registered trademark of Apple Computer, Inc.
Macintosh is a trademark of Apple Computer, Inc.

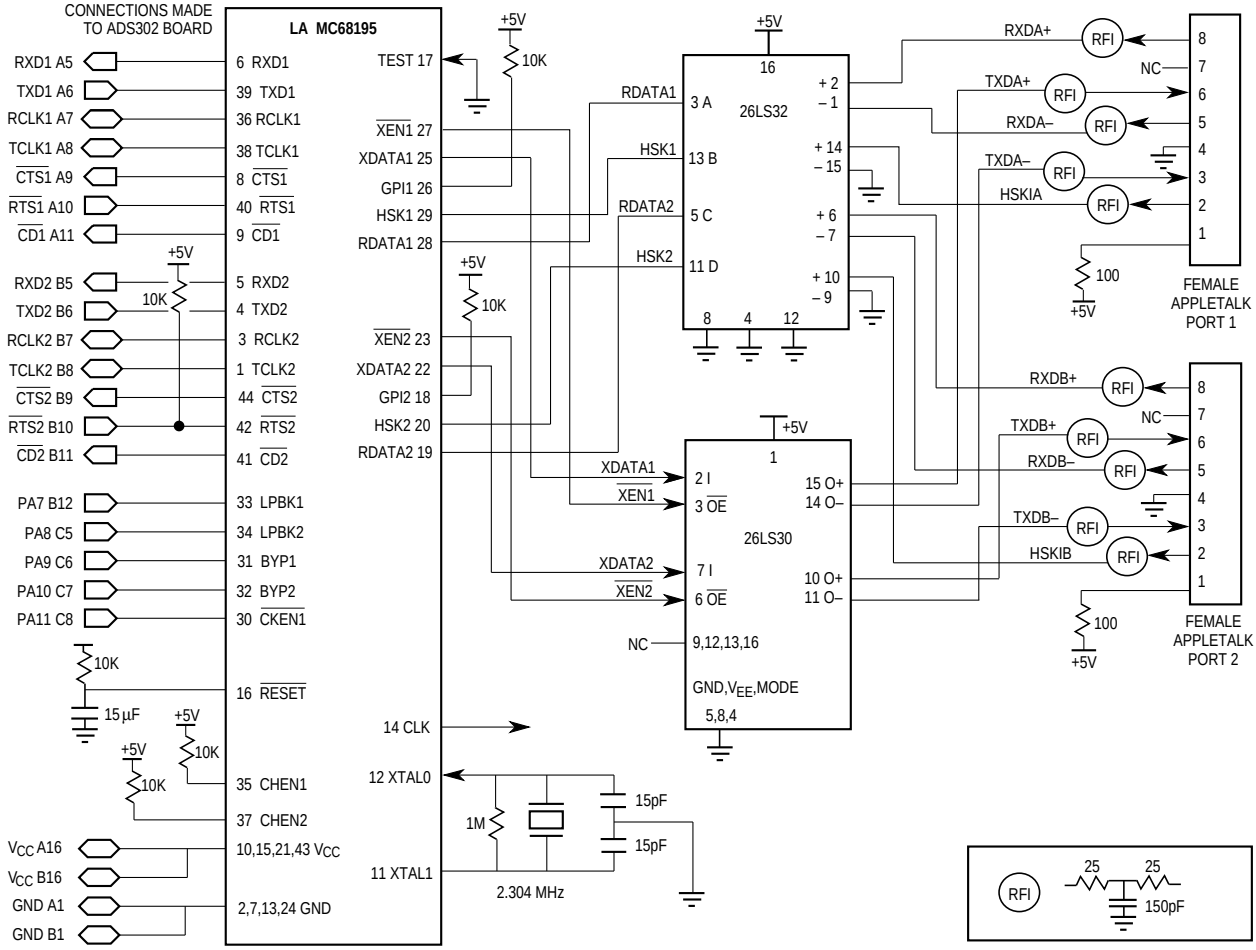


Figure D-32. Local Talk Adaptor Board

APPENDIX E
SCC PROGRAMMING REFERENCE

This appendix is intended to be used as a quick programming reference for setting up the SCCs. Each of the following three subsections pulls together the information required to set up an SCC to operate in HDLC mode, UART mode, or transparent mode. For detailed information about a particular register, parameter, or function, please refer to the appropriate subsection in Section 4 Communications Processor (CP).

NOTE

All SCC buffer descriptor tables are shown with eight Rx BDs and eight Tx BDs. If the DRAM refresh controller is used, then only six Tx BDs are available for SCC2. Also, if the SCP or SMCs are used, only four Tx BDs are available for SCC3. In these cases, the wrap bit should be set in the last Tx BD.

E.1 HDLC PROGRAMMING REFERENCE SECTION

This subsection deals with the registers and parameters required to program an SCC for HDLC. A generic algorithm for programming the SCC is also presented.

E.1.1 HDLC Programming Model

The programming model and memory map for the HDLC protocol is shown in Table E-1 (a). The offsets for each SCC are given above each table. Some parameters are common to all protocols. The HDLC parameters are shown for those entries that are protocol specific. Table E-1 (b). depicts the general and protocol-specific parameter RAM for each SCC. The SCC registers are shown in Table E-1 (c), and the communications processor registers are shown in Table E-1 (d). Note that reserved bits in registers should be written as zeros.

Table E-1 (a). HDLC Programming Mode Receive and Transmit Buffer Descriptors for SCCx

Initialized by User	Offset Hex	Name	Initialized by User	Offset Hex	Name
Yes	00	Rx BD 0 Control/Status	Yes	40	Tx BD 0 Control/Status
	02	Rx BD 0 Data Count	Yes	42	Tx BD 0 Data Count
Yes	04	Rx BD 0 Data Pointer (High Word)	Yes	44	Tx BD 0 Data Pointer (High Word)
Yes	06	Rx BD 0 Data Pointer (Low Word)	Yes	46	Tx BD 0 Data Pointer (Low Word)
Yes	08	Rx BD 1 Control/Status	Yes	48	Tx BD 1 Control/Status
	0A	Rx BD 1 Data Count	Yes	4A	Tx BD 1 Data Count
Yes	0C	Rx BD 1 Data Pointer (High Word)	Yes	4C	Tx BD 1 Data Pointer (High Word)
Yes	0E	Rx BD 1 Data Pointer (Low Word)	Yes	4E	Tx BD 1 Data Pointer (Low Word)
Yes	10	Rx BD 2 Control/Status	Yes	50	Tx BD 2 Control/Status
	12	Rx BD 2 Data Count	Yes	52	Tx BD 2 Data Count
Yes	14	Rx BD 2 Data Pointer (High Word)	Yes	54	Tx BD 2 Data Pointer (High Word)
Yes	16	Rx BD 2 Data Pointer (Low Word)	Yes	56	Tx BD 2 Data Pointer (Low Word)
Yes	18	Rx BD 3 Control/Status	Yes	58	Tx BD 3 Control/Status
	1A	Rx BD 3 Data Count	Yes	5A	Tx BD 3 Data Count
Yes	1C	Rx BD 3 Data Pointer (High Word)	Yes	5C	Tx BD 3 Data Pointer (High Word)
Yes	1E	Rx BD 3 Data Pointer (Low Word)	Yes	5E	Tx BD 3 Data Pointer (Low Word)
Yes	20	Rx BD 4 Control/Status	Yes	60	Tx BD 4 Control/Status
	22	Rx BD 4 Data Count	Yes	62	Tx BD 4 Data Count
Yes	24	Rx BD 4 Data Pointer (High Word)	Yes	64	Tx BD 4 Data Pointer (High Word)
Yes	26	Rx BD 4 Data Pointer (Low Word)	Yes	66	Tx BD 4 Data Pointer (Low Word)
Yes	28	Rx BD 5 Control/Status	Yes	68	Tx BD 5 Control/Status
	2A	Rx BD 5 Data Count	Yes	6A	Tx BD 5 Data Count
Yes	2C	Rx BD 5 Data Pointer (High Word)	Yes	6C	Tx BD 5 Data Pointer (High Word)
Yes	2E	Rx BD 5 Data Pointer (Low Word)	Yes	6E	Tx BD 5 Data Pointer (Low Word)
Yes	30	Rx BD 6 Control/Status	Yes	70	Tx BD 6 Control/Status
	32	Rx BD 6 Data Count	Yes	72	Tx BD 6 Data Count
Yes	34	Rx BD 6 Data Pointer (High Word)	Yes	74	Tx BD 6 Data Pointer (High Word)
Yes	36	Rx BD 6 Data Pointer (Low Word)	Yes	76	Tx BD 6 Data Pointer (Low Word)
Yes	38	Rx BD 7 Control/Status	Yes	78	Tx BD 7 Control/Status
	3A	Rx BD 7 Data Count	Yes	7A	Tx BD 7 Data Count
Yes	3C	Rx BD 7 Data Pointer (High Word)	Yes	7C	Tx BD 7 Data Pointer (High Word)
Yes	3E	Rx BD 7 Data Pointer (Low Word)	Yes	7E	Tx BD 7 Data Pointer (Low Word)

NOTE: The offset is from the MC68302 base address + ($400 for SCC1, $500 for SCC2, or $600 for SCC3).

Table E-1 (b). HDLC Programming Model (Continued) General Parameter and HDLC Protocol-Specific RAM for SCCx

Initialized by User	Offset Hex	Name		Initialized by User	Offset Hex	Name
Yes	80	RFCR	TFCR	Yes	A2	CRC_Mask_H
Yes	82	MRBLR			A4	Temp Transmit CRC Low
	84	Rx Internal State			A6	Temp Transmit CRC High
	86	Reserved	Rx Internal Buffer No.	Yes	A8	DISFC
	88 8A	Rx Internal Data Pointer (High Word) Rx Internal Data Pointer (Low Word)		Yes	AA	CRCEC
	8C	Rx Internal Byte Count		Yes	AC	ABTSC
	8E	Rx Temp		Yes	AE	NMARC
	90	Tx Internal State		Yes	B0	RETRC
	92	Reserved	Tx Internal Buffer No.	Yes	B2	MFLR
	94 96	Tx Internal Data Pointer (High Word) TX Internal Data Pointer (Low Word)			B4	MAX_cnt
	98	Tx Internal Byte Count		Yes	B6	HMASK
	9A	Tx Temp		Yes	B8	HADDR1
	9C	Temp Receive CRC Low		Yes	BA	HADDR2
	9E	Temp Receive CRC High		Yes	BC	HADDR3
Yes	A0	CRC_Mask_L		Yes	BE	HADDR4

NOTE: The offset is from the MC68302 base address + ($400 for SCC1, $500 for SCC2, or $600 for SCC3).

Table E-1 (c) SCCx Register Set

Initialized by User	Offset Hex	Name	
	00	Reserved	
Yes	02	SCC Configuration Register (SCON)	
Yes	04	SCC Mode Register (SCM)	
Yes	06	SCC Data Synchronization Register (DSR)	
Yes	08	Event Register (SCCE)	Reserved
Yes	0A	Mask Register (SCCM)	Reserved
	0C	Status Register (SCCS)	Reserved
	0E	Reserved	

NOTE: The offset is from the MC68302 base address + ($880 for SCC1, $890 for SCC2, or $8A0 for SCC3).

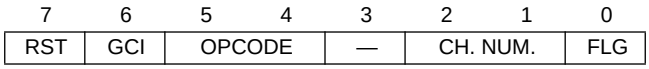
Table E-1 (d).General Registers (Only One Set)

Initialized by User	Offset Hex	Name	
	860	Command Register (CR)	Reserved
Yes	8B2	Serial Interface Mask Register (SIMASK)	
Yes	8B4	Serial Interface Mask Register (SIMODE)	

NOTE: The offset is from the MC68302 base address.

E.1.1 COMMUNICATIONS PROCESSOR (CP) REGISTERS. The CP has one set of three registers that configure the operation of the serial interface for all three SCCs. These registers are discussed in the following paragraphs.

E.1.1.1.1 Command Register CR). The command register is an 8-bit register located at offset $860 (on D15-D8 of a 16-bit data bus). This register is used to issue commands to the CP. The user should set the FLG bit when a command is written to the command register. The CP clears the FLG bit during command processing to indicate that it is ready for the next command. Reserved bits in registers should be written as zeros.



RST—Software Reset Command (set by the user and cleared by the CP)

0 = No software reset command issued or cleared by CP during software reset sequence.
1 = Software reset command (FLG bit should also be set if it is not already set).

GCI—GCI Commands

0 = Normal operation.
1 = The OPCODE bits are used for GCI commands (user should set CH. NUM. to 10 and FLG to 1).

OPCODE—Command Opcode

00 = STOP TRANSMIT Command.
01 = RESTART TRANSMIT Command.
10 = ENTER HUNT MODE Command.
11 = Reset receiver BCS generator (used only in BISYNC mode).

BIT 3—Reserved (should be set to zero by the user when the command register is written)

CH. NUM.—Channel Number

00 = SCC1.
01 = SCC2.
10 = SCC3.
11 = Reserved.

FLG—Command Semaphore Flag (set by the user and cleared by the CP upon command completion)

0 = CP is ready to receive a new command (should be checked before issuing the next command to the CP)
1= Command register contains a command to be executed or one that is currently being executed.

E.1.1.1.2 Serial Interface Mode Register (SIMODE). This 16-bit register is located at offset $8B4. The SIMODE register is used to configure the serial interface mode for the three SCCs.

15	14	13	12	11	10	9	8
SETZ	SYNC/SCIT	SDIAG1	SDIAG0	SDC2	SDC1	B2RB	B2RA

7	6	5	4	3	2	1	0
B1RB	B1RA	DRB	DRA	MSC3	MSC2	MS1	MS0

SETZ—Set L1 TXD to Zero (valid only for the GCI interface)

0 = Normal operation.
1 = L1 TXD output set to a logic zero (used in GCI activation).

SYNC/SCIT—SYNC Mode/SCIT Select Support

0 = One pulse wide prior to the 8-bit data.
1 = N pulses wide and envelopes the N-bit data.

SDIAG1, SDIAG0—Serial Interface Diagnostic Mode

00 = Normal operation.
01 = Automatic echo.
10 = Internal loopback.
11 = Loopback control.

SDC2—Serial Data Strobe Control 2

0 = SDS2 signal is asserted during the B2 channel.
1 = SDS1 signal is asserted during the B2 channel.

SDC1—Serial Data Strobe Control 1

0 = SDS1 signal is asserted during the B1 channel.
1 = SDS2 signal is asserted during the B1 channel.

B2RB, B2RA—B2 Channel Route in IDL/GCI Mode or CH-3 Route in PCM Mode

00 = Channel not supported.
01 = Route channel to SCC1.
10 = Route channel to SCC2 (a MSC2 is cleared).
11 = Route channel to SCC3 (if MSC3 is cleared).

B1 RB, B1 RA—B1 Channel Route in IDL/GCI Mode or CH-2 Route in PCM Mode

00 = Channel not supported.
01 = Route channel to SCC1.
10 = Route channel to SCC2 (a MSC2 is cleared).
11 = Route channel to SCC3 (if MSC3 is cleared).

DRB, DRA—D Channel Route in IDL/GCI Mode or CH-1 Route in PCM Mode

00 = Channel not supported.
01 = Route channel to SCC1.
10 = Route channel to SCC2 (a MSC2 is cleared).
11 = Route channel to SCC3 (if MSC3 is cleared).

MSC3—SCC3 Connection

0 = SCC3 is connected to the multiplexed serial interface.
1 = SCC3 is not connected to the multiplexed serial interface.

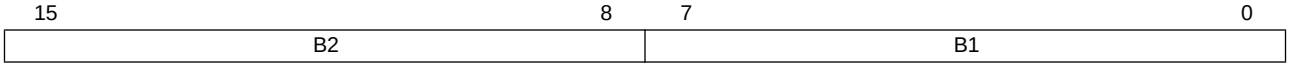
MSC2—SCC2 Connection

0 = SCC2 is connected to the multiplexed serial interface.
1 = SCC2 is not connected to the multiplexed serial interface.

MS1, MS0—Mode Supported

00 = NMSI mode.
01 = PCM mode.
10 = IDL mode.
11 = GCI interface.

E.1.1.1.3 Serial Interface Mask Register (SIMASK). This 16-bit register is located at offset $8B2. The SIMASK register is used to configure which bits on the B1 and B2 channels are used in the GCI and IDL modes. Bit 0 of SIMASK is the first bit transmitted and received on B1.



E.1.1.2 PER SCC REGISTERS. Each of the three SCCs has a set of the following six registers. These registers configure the SCC and the protocol operation. Some parameters and register bits are protocol independent. The HDLC functions have been given for those parameters and bits that are protocol specific.

E.1.1.2.1 Serial Configuration Register (SCON). This 16-bit register is located at offset $882 (SCC1), $892 (SCC2), and $8A2 (SCC3). The SCON register is used to select the clock source and baud rate for the SCC.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WOMS	EXTC	TCS	RCS	CD10	CD9	CD8	CD7	CD6	CD5	CD4	CD3	CD2	CD1	CD0	DIV4

WOMS—Wired-OR Mode Select

0 = TXD driver operates normally.
1 = TXD driver functions as an open-drain output and may be wired together with other TXD pins.

EXTC—External Clock Source

0 = The internal main clock is the source for the baud rate generator.
1 = The external clock on the TIN1 pin is the source for the baud rate generator.

TCS—Transmit Clock Source

0 = Transmit clock source is the baud rate generator output.
1 = Transmit clock source is the clock signal on TCLK pin.

RCS—Receive Clock Source

0 = Receive clock source is the baud rate generator output.
1 = Receive clock source is the clock signal on RCLK pin.

CD10—CD0—Clock Divider

Used to preset the 11-bit counter that is decremented at the prescaler output rate.

DIV4—SCC Clock Prescaler Divide by 4

0 = Divide-by-1 prescaler.
1 = Divide-by-4 prescaler.

E.1.1.2.2 SCC Mode Register (SCM). This 16-bit register is located at offset $884 (SCC1), $894 (SCC2), and $8A4 (SCC3). The SCM register configures the operation of the SCC and defines HDLC specific parameters. Note that reserved bits in registers should be written as zeros.

15	14	13	12	11	10	9	8
NOF3	NOF2	NOF1	NOF0	C32	FSE	—	RTE

7	6	5	4	3	2	1	0
FLG	ENC	DIAG1	DIAG0	ENR	ENT	MODE1	M0DE0

NOF3-NOF0—Number of Flags

Minimum number of flags between frames or before frames specifies the number of flags (0-15) to be inserted between frames or before a frame is transmitted.

C32—CRC16/CRC32

0 = 16-bit CRC.
1 = 32-bit CRC.

FSE—Flag Sharing Enable

0 = Normal operation.
1 = Transmits a single shared flag between back-to-back frames if NOF3-NOF2 = 0. Other values of NOF3-NOF2 are decremented by 1.

BIT 9—Reserved for future use

RTE—Retransmit Enable

0 = No retransmission.
1 = Retransmit enabled.

FLG—Transmit Flags/Idles between Frames and Control the $\overline{\text{RTS}}$ Pin

0 = Send ones between frames; $\overline{\text{RTS}}$ negated between frames.
1 = Send flags between frames; $\overline{\text{RTS}}$ is always asserted.

ENC—Data Encoding Format

0 = Non-return to zero (NRZ).
1 = Non-return to zero inverted (NRZI).

DIAG1, DIAG0—Diagnostic Mode

00 = Normal operation.
01 = Loopback mode.
10 = Automatic echo.
11 = Software operation.

ENR—Enable Receiver

0 = Receiver is disabled.
1 = Receiver is enabled.

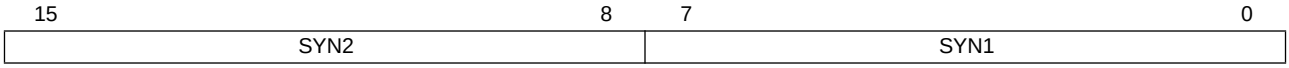
ENT—Enable Transmitter

0 = Transmitter is disabled.
1 = Transmitter is enabled.

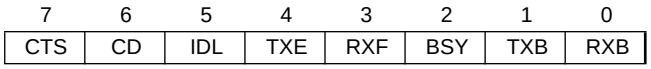
MODE1, MODE0—Channel Mode

00 = HDLC.
01 = Asynchronous (UART and DDCMP).
10 = Synchronous DDCMP and V.110.
11 = BISYNC and Promiscuous (Transparent).

E.1.1.2.3 SCC Data Synchronization Register (DSR). This 16-bit register is located at offset $886 (SCC1), $896 (SCC2), and $8A6 (SCC3). The DSR specifies the pattern used for the frame synchronization procedure. For HDLC, the DSR should be set to $7E7E. The DSR value after reset is $7E7E.



E.1.1.2.4 HDLC Event Register (SCCE). This 8-bit register is located at offset $888 (SCC1), $898 (SCC2), and $8A8 (SCC3) on D15-D8 of a 16-bit data bus. The SCCE is used to report events recognized by the HDLC channel. Bits mst be cleared by the user to avoid missing interrupt events. Bits are cleared by writing ones to the corresponding bit positions



$\overline{\text{CTS}}$—Clear-To-Send Status Changed

0 = No interrupt.
1 = A change in the status of $\overline{\text{CTS}}$ was detected.

$\overline{\text{CD}}$—Carrier Detect Status Changed

0 = No interrupt.
1 = A change in the status of $\overline{\text{CD}}$ was detected.

IDL—IDLE Sequence Status Changed

0 = No interrupt.
1 = A change in the status of the receive data serial line was detected.

TXE—Tx Error

0 = No interrupt
1 = An error ($\overline{\text{CTS}}$ lost or underrun) occurred on the transmitter channel.

RXF—Rx Frame

0 = No interrupt.
1 = A complete frame has been received on the HDLC channel.

BSY—Busy Condition

0 = No interrupt.
1 = A frame was received and discarded due to lack of buffers.

TXB—Tx Buffer

0 = No interrupt.
1 = A buffer has been transmitted on the HDLC channel (set only if the I bit in the Tx buffer descriptor is set).

RXB—Rx Buffer

0 = No interrupt
1 = A buffer that was not a complete frame was received on the HDLC channel (set only if the I bit in the Rx buffer descriptor is set).

E.1.1.2.5 HDLC Mask Register (SCCM). This 8-bit register is located at offset $88A (SCC1), $89A (SCC2), and $8AA (SCC3) on D15-D8 of a 16-bit data bus. The SCCM is used to enable and disable interrupt events reported by the SCCE. The mask bits correspond to the interrupt event bit shown in the SCCE. A bit should be set to one to enable the corresponding interrupt in the SCCE.

7	6	5	4	3	2	1	0
CTS	CD	IDL	TXE	RXF	BSY	TXB	RXB

E.1.1.2.6 HDLC Status Register (SCCS). This 8-bit register is located at offset $88C (SCC1), $89C (SCC2), and $8AC (SCC3) on D15-D8 of a 16-bit data bus. The SCCS register reflects the current status of the RXD, $\overline{CD}$, and $\overline{CTS}$ lines as seen by the SCC.

7	6	5	4	3	2	1	0
—	—	—	—	—	ID	CD	CTS

ID—Idle Status on the Receiver Line (valid only when the ENR bit is set and the receive clock is running)

0 = Receiver line is not idling.
1 = Either $\overline{CD}$ is not asserted or the receiver line is idling while $\overline{CD}$ is asserted.

$\overline{CD}$—Carrier Detect Status Changed (valid only when the ENR bit is set and the receive clock is running)

0 = $\overline{CD}$ is asserted.
1 = $\overline{CD}$ is not asserted.

$\overline{CTS}$—Clear-To-Send Status Changed (valid only when the ENT bit is set and the transmit clock is running)

0 = $\overline{CTS}$ is asserted.
1 = $\overline{CTS}$ is not asserted.

E.1.1.3 GENERAL AND HDLC PROTOCOL-SPECIFIC PARAMETER RAM. Each SCC has 32 words of parameter RAM used to configure receive and transmit operation, store temporary parameters for the CP, and maintain counters. The first 14 words are general parameters, which are the same for each protocol. The last 18 words are specific to the protocol selected. The following sections discuss the parameters that the user must initialize to configure the HDLC operation.

E.1.1.3.1 RFCR/TFCR—Rx Function Code/Tx Function Code. This 16-bit parameter contains the function codes of the receive data buffers and transmit data buffers. The user must initialize the function codes (FC2-FC0) to a value less than 7.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	FC2	FC1	FC0	0	0	0	0	0	FC2	FC1	FC0	0	0	0	0

E.1.1.3.2 MRBLR—Maximum Rx Buffer Length. This16-bit parameter defines the maximum receiver buffer length for each of the eight receive buffer descriptors.

E.1.1.3.3 CRC Mask_L and CRC Mask_H. This 32-bit parameter contains the constant values used for the 16-bit and 32-bit CRC calculation. For a 16-bit CRC, CRC_MASK_L should be set to $F0B8 and CRC_MASK_H is not used. For a 32-bit CRC, the user should set CRC_MASK_L = $DEBB and CRC_MASK_H = $20E3.

E.1.1.3.4 DISFC—Discard Frame Counter. This 16-bit parameter is incremented when a frame is discarded due to lack of receive buffers.

E.1.1.3.5 CRCEC—CRC Error Counter. This 16-bit parameter is incremented when a CRC error is detected in an incoming frame.

E.1.1.3.6 ABTSC—Abort Sequence Counter. This 16-bit parameter is incremented when an abort sequence is detected in an incoming frame,

E.1.1.3.7 NMARC—Nonmatching Address Received Counter. This 16-bit parameter is incremented when an error-free frame that does not match the user-defined addresses is detected.

E.1.1.3.8 RETRC—Frame Retransmission Counter. This 16-bit parameter is incremented when a frame is retransmitted due to a collision.

E.1.1.3.9 MFLR—Maximum Frame Length Register. This16-bit parameter defines the maximum length of an incoming receive frame.

E.1.1.3.10 HMASK—HDLC Frame Address Mask. This 16-bit parameter is the user-defined frame address mask register. A one should be written to each bit for which the address comparison is to occur. Bits 15-8 contain the least significant address byte, and bits 7-0 contain the most significant address byte.

E.1.1.3.11 HADDR1, HADDR2, HADDR3, and HADDR4—HDLC Frame Address. These four 16-bit parameters are the user-defined frame address registers. Bits 15-8 contain the least significant address byte, and bits 7-0 contain the most significant address byte.

E.1.1.4 RECEIVE BUFFER DESCRIPTORS. Each SCC has eight receive buffer descriptors. Each buffer descriptor consists of four words as shown below. Reserved bits in registers should be written as zeros.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET + 0	E	X	W	I	L	F	—	—	—	—	LG	NO	AB	CR	OV	CD
OFFSET +2	DATA LENGTH															
OFFSET +4	RX BUFFER POINTER															
OFFSET +6																

E.1.1.4.1 Receive BD Control/Status Word. To initialize the buffer, the user should write bits 15-12 and clear bits 11-10 and 5-0. The IMP clears bit 15 when the buffer is closed and sets bits 5-0 depending on which error occurred.

E—Empty

0 = This data buffer is full or has been closed due to an error condition.
1 = This data buffer is empty; must be set by the user to enable reception into this buffer.

X—External Buffer

0 = The data buffer associated with this BD is in internal dual-port RAM.
1 = The data buffer associated with this BD is in external memory.

W—Wrap (final BD in table)

0 = This is not the last BD in the receive BD table.
1 = This is the last BD in the receive BD table.

I—Interrupt (The RXF bit in the event register is set when a complete frame is received, independent of the I bit.)

0 = The RXB bit in the event register is not set when this buffer is closed.
1 = The RXB bit (or RXF bit, if this is the last buffer in a frame) in the event register is set when this buffer is closed.

L—Last in Frame

0 = This buffer is not the last buffer in a frame.
1 = This buffer is the last buffer in a frame.

F—First in Frame

0 = This buffer is not the first buffer in a frame.
1 = This buffer is the first buffer in a frame.

Bits 9–6—Reserved for future use

LG—Rx Frame Length Violation

0 = No frame length violation occurred.
1 = A frame length violation was detected. Up to the number of bytes specified in the maximum frame length will be written to the buffer (or buffers, if multiple buffers per frame).

NO—Rx Nonoctet Aligned Frame

0 = An octet aligned frame was received.
1 = A nonoctet aligned frame was received.

AB—Rx Abort Sequence

0 = No abort was received.
1 = A minimum of seven ones was received during frame reception.

CR—Rx CRC Error

0 = This frame does not contain a CRC error.
1 = This frame contains a CRC error.

OV—Overrun

0 = No receiver overrun occurred.
1 = A receiver overrun condition occurred during frame reception.

CD—Carrier Detect Lost (valid only in NMSI mode)

0 = No CD lost was detected.
1 = CD was negated during frame reception.

E.1.1.4.2 Receive Buffer Data Length. This 16-bit value is written by the IMP to indicate the number of data bytes received into the data buffer.

E.1.1.4.3 Receive Buffer Pointer. This 32-bit value is written by the user to indicate the address where the data is to be stored.

E.1.1.5 TRANSMIT BUFFER DESCRIPTORS. Each SCC has eight transmit buffer descriptors. Each buffer descriptor consists of four words as shown below. Reserved bits in registers should be written as zeros.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET + 0	R	X	W	I	L	TC	—	—	—	—	—	—	—	—	UN	CT
OFFSET +2	DATA LENGTH															
OFFSET +4	TX BUFFER POINTER															
OFFSET +6																

E.1.1.5.1 Transmit BD Control/Status Word. To initialize the buffer, the user should write bits 15-10 and bits 1-0. The IMP clears bit 15 when the buffer is transmitted or closed due to an error and sets bits 1-0 depending on which error occurred.

R—Ready

0 = This data buffer is not currently ready for transmission.
1 = This data buffer has been prepared by the user for transmission but has not yet been fully transmitted. Must be set by the user to enable transmission of the buffer.

X—External Buffer

0 = The data buffer associated with this BD is in internal dual-port RAM.
1 = The data buffer associated with this BD is in external memory.

W—Wrap (final BD in table)

0 = This is not the last BD in the transmit BD table.
1 = This is the last BD in the transmit BD table.

I—Interrupt

0 = The TXB bit in the event register is not set when this buffer is closed.
1 = The TXB bit in the event register is set if this buffer closed without an error. If an error occurred, then TXE is set.

L—Last in Frame

0 = This buffer is not the last buffer in a frame.
1 = This buffer is the last buffer in a frame.

TC—Tx CRC

0 = Transmit the closing flag after the last data byte.
1 = Transmit the CRC sequence after the last data byte.

Bits 9-2—Reserved for future use

UN—Underrun

0 = No transmitter underrun occurred.
1 = A transmitter underrun condition occurred while transmitting the associated data buffer.

CT—CTS Lost

0 = No CTS or L1GR lost was detected during frame transmission.
1 = CTS in NMSI mode or L1GR in IDL/GCI mode was lost during frame transmission.

E.1.1.5.2 Transmit Buffer Data Length. This 16-bit value is written by the user to indicate the number of data bytes to be transmitted from the data butter.

E.1.1.5.3 Transmit Buffer Pointer. This 32-bit value is written by the user to indicate the address of the first byte of data in the data buffer.

E.1.2 Programming the SCC for HDLC

This section gives a generic algorithm for programming an SCC to handle HDLC. The algorithm is intended to show what must be done and in what order to initialize the SCC and prepare the SCC for transmission and reception. The algorithm is not specific and assumes that the IMP and other on-chip peripherals have been initialized as required by the system hardware (timers, chip selects, etc.).

E.1.2.1 CP INITIALIZATION.

1. Write the port A and port B control registers (PACNT and PBCNT) to configure SCC2 or SCC3 serial interface pins as peripheral pins, if SCC2 or SCC3 is used.
2. Write SIMODE to configure the SCCs physical interface.
3. Write SIMASK if IDL or GCI multiplexed mode was selected in SIMODE.

E.1.2.2 GENERAL AND HDLC PROTOCOL-SPECIFIC RAM INITIALIZATION.

4. Write RFCR/TFCR.

5. Write MRBLR.
6. Write CRC_Mask_L and CRC_Mask_H.
7. Write DISFC.
8. Write CRCEC.
9. Write ABTSC.
10. Write NMARC.
11. Write RETRC.
12. Write MFLR.
13. Write HMASK.
14. Write HADDR1, HADDR2, HADDR3, and HADDR4.

E.1.2.3 SCC INITIALIZATION.

15. Write SCON.
16. Write SCM without setting the ENR and ENT bits.
17. Write DSR.
18. Write SCCE with $FF to clear any previous events.
19. Write SCCM.
20. Write IMR.

E.1.2.4 SCC OPERATION.

21. Write the Rx buffer descriptor control/status, buffer pointer high, and buffer pointer low words for all of the buffer descriptors that are going to be used. Set the W bit in the last buffer descriptor to be used in the queue.
22. Prepare transmit buffers as required to transmit data on the SCC. Set the R bit in each Tx buffer descriptor's control/status word when the data buffer is ready for transmission. Set the W bit in the last Tx buffer descriptor in the table so that the IMP will use the first Tx buffer descriptor (after the user sets the R bit) for the next transmission.
23. Write SCM, setting the ENR and ENT bits to enable reception and transmission on the SCC.
24. Prepare more transmit buffers as required to transmit data on the SCC.

E.1.2.5 SCC INTERRUPT HANDLING.

1. Read the SCC event register.
2. Clear any unmasked bits that will be used in this interrupt routine.
3. Handle the interrupt events as required by the system.
4. Clear the appropriate SCC bit in the in-service register (ISR) of the interrupt controller.
5. Return from the interrupt.

E.2 UART PROGRAMMING REFERENCE SECTION

This subsection deals with the registers and parameters required to program an SCC as a UART. At the end of this subsection is a generic algorithm for programming the SCC.

E.2.1 UART Programming Model

The programming model and memory map for the UART protocol is shown in Table E-2. The offsets for each SCC are given above each table. Some parameters are common to all protocols. The UART parameters are shown for those entries that are protocol specific. Table E-2(b) depicts the general and protocol-specific parameter RAM for each SCC. The SCC registers are shown in Table E-2(c), and the communications processor registers are shown in Table E-2(d). Note that reserved bits in registers should be written as zeros.

Table E-1 (a). UART Programming Model
Receive and Transmit Buffer Descriptors for SCCx

Initialized by User	Offset Hex	Name	Initialized by User	Offset Hex	Name
Yes	00	Rx BD 0 Control/Status	Yes	40	Tx BD 0 Control/Status
	02	Rx BD 0 Data Count	Yes	42	Tx BD 0 Data Count
Yes	04	Rx BD 0 Data Pointer (High Word)	Yes	44	Tx BD 0 Data Pointer (High Word)
Yes	06	Rx BD 0 Data Pointer (Low Word)	Yes	46	Tx BD 0 Data Pointer (Low Word)
Yes	08	Rx BD 1 Control/Status	Yes	48	Tx BD 1 Control/Status
	0A	Rx BD 1 Data Count	Yes	4A	Tx BD 1 Data Count
Yes	0C	Rx BD 1 Data Pointer (High Word)	Yes	4C	Tx BD 1 Data Pointer (High Word)
Yes	0E	Rx BD 1 Data Pointer (Low Word)	Yes	4E	Tx BD 1 Data Pointer (Low Word)
Yes	10	Rx BD 2 Control/Status	Yes	50	Tx BD 2 Control/Status
	12	Rx BD 2 Data Count	Yes	52	Tx BD 2 Data Count
Yes	14	Rx BD 2 Data Pointer (High Word)	Yes	54	Tx BD 2 Data Pointer (High Word)
Yes	16	Rx BD 2 Data Pointer (Low Word)	Yes	56	Tx BD 2 Data Pointer (Low Word)
Yes	18	Rx BD 3 Control/Status	Yes	58	Tx BD 3 Control/Status
	1A	Rx BD 3 Data Count	Yes	5A	Tx BD 3 Data Count
Yes	1C	Rx BD 3 Data Pointer (High Word)	Yes	5C	Tx BD 3 Data Pointer (High Word)
Yes	1E	Rx BD 3 Data Pointer (Low Word)	Yes	5E	Tx BD 3 Data Pointer (Low Word)
Yes	20	Rx BD 4 Control/Status	Yes	60	Tx BD 4 Control/Status
	22	Rx BD 4 Data Count	Yes	62	Tx BD 4 Data Count
Yes	24	Rx BD 4 Data Pointer (High Word)	Yes	64	Tx BD 4 Data Pointer (High Word)
Yes	26	Rx BD 4 Data Pointer (Low Word)	Yes	66	Tx BD 4 Data Pointer (Low Word)
Yes	28	Rx BD 5 Control/Status	Yes	68	Tx BD 5 Control/Status
	2A	Rx BD 5 Data Count	Yes	6A	Tx BD 5 Data Count
Yes	2C	Rx BD 5 Data Pointer (High Word)	Yes	6C	Tx BD 5 Data Pointer (High Word)
Yes	2E	Rx BD 5 Data Pointer (Low Word)	Yes	6E	Tx BD 5 Data Pointer (Low Word)
Yes	30	Rx BD 6 Control/Status	Yes	70	Tx BD 6 Control/Status
	32	Rx BD 6 Data Count	Yes	72	Tx BD 6 Data Count
Yes	34	Rx BD 6 Data Pointer (High Word)	Yes	74	Tx BD 6 Data Pointer (High Word)
Yes	36	Rx BD 6 Data Pointer (Low Word)	Yes	76	Tx BD 6 Data Pointer (Low Word)
Yes	38	Rx BD 7 Control/Status	Yes	78	Tx BD 7 Control/Status
	3A	Rx BD 7 Data Count	Yes	7A	Tx BD 7 Data Count
Yes	3C	Rx BD 7 Data Pointer (High Word)	Yes	7C	Tx BD 7 Data Pointer (High Word)
Yes	3E	Rx BD 7 Data Pointer (Low Word)	Yes	7E	Tx BD 7 Data Pointer (Low Word)

NOTE: The offset is from the MC68302 base address + ($400 for SCC1, $500 for SCC2, or $600 for SCC3).

Table E-1 (b). UART Programming Model (Continued)
General Parameter and UART Protocol-Specific RAM for SCCx

Initialized by User	Offset Hex	Name		Initialized by User	Offset Hex	Name
Yes	80	RFCR	TFCR	Yes	A2	Receive Parity Error Counter
Yes	82	MRBLR		Yes	A4	Receive Framing Error Counter
	84	Rx Internal State		Yes	A6	Receive Noise Counter
	86	Reserved	Rx Internal Buffer No.	Yes	A8	Receive Break Condition Counter
	88 8A	Rx Internal Data Pointer (High Word) Rx Internal Data Pointer (Low Word)		Yes	AA	UART Address Character 1
	8C	Rx Internal Byte Count		Yes	AC	UART Address Character 2
	8E	Rx Temp			AE	Receive Control Character Register
	90	Tx Internal State		Yes	B0	Control Character 1
	92	Reserved	Tx Internal Buffer No.	Yes	B2	Control Character 2
	94 96	Tx Internal Data Pointer (High Word) TX Internal Data Pointer (Low Word)		Yes	B4	Control Character 3
	98	Tx Internal Byte Count		Yes	B6	Control Character 4
	9A	Tx Temp		Yes	B8	Control Character 5
Yes	9C	Maximum IDLE Characters		Yes	BA	Control Character 6
	9E	Temporary Receive IDLE Counter		Yes	BC	Control Character 7
Yes	A0	Break Counter Register		Yes	BE	Control Character 8

NOTE: The offset is from the MC68302 base address + ($400 for SCC1, $500 for SCC2, or $600 for SCC3).

Table E-1 (c). SCCx Register Set

Initialized by User	Offset Hex	Name	
	00	Reserved	
Yes	02	SCC Configuration Register (SCON)	
Yes	04	SCC Mode Register (SCM)	
Yes	06	SCC Data Synchronization Register (DSR)	
Yes	08	Event Register (SCCE)	Reserved
Yes	0A	Mask Register (SCCM)	Reserved
	0C	Status Register (SCCS)	Reserved
	0E	Reserved	

NOTE: The offset is from the MC68302 base address + ($880 for SCC1, $890 for SCC2, or $8A0 for SCC3).

Table E-1 (d). General Registers (Only One Set)

Initialized by User	Offset Hex	Name	
	860	Command Register (CR)	Reserved
Yes	8B2	Serial Interface Mask Register (SIMASK)	
Yes	8B4	Serial Interface Mask Register (SIMODE)	

NOTE: The offset is from the MC68302 base address.

E.2.1.1 COMMUNICATIONS PROCESSOR (CP) REGISTERS. The CP has one set of three registers that configure the operation of the serial interface for all three SCCs. These registers are discussed in the next three subsections.

E.2.1.1.1 Command Register (CR). The command register is an 8-bit register located at offset $860 (on D15-D8 of a 16-bit data bus). This register is used to issue commands to the CP. The user should set the FLG bit when a command is written to the command register. The CP clears the FLG bit during command processing to indicate that it is ready for the next command.

7	6	5 4	3	2 1	0
RST	GCI	OPCODE	—	CH. NUM.	FLG

RST—Software Reset Command (set by the user and cleared by the CP)

0 = No software reset command issued or cleared by CP during software reset sequence.
1 = Software reset command (FLG bit should also be set if it is not already set).

GCI—GCI Commands

0 = Normal operation.
1 = The OPCODE bits are used for GCI commands (user should set CH. NUM. to 10 and FLG to 1).

OPCODE—Command Opcode

00 = STOP TRANSMIT Command.
01 = RESTART TRANSMIT Command.
10 = ENTER HUNT MODE Command.
11 = Reset receiver BCS generator (used only in BISYNC mode).

BIT 3—Reserved (should be set to zero by the user when the command register is written)

CH. NUM.—Channel Number

00 = SCC1.
01 = SCC2.
10 = SCC3.
11 = Reserved.

FLG—Command Semaphore Flag (set by the user and cleared by the CP upon command completion)

0 = CP is ready to receive a new command (should be checked before issuing the next command to the CP).
1 = Command register contains a command to be executed or one that is currently being executed.

E.2.1.1.2 Serial Interface Mode Register (SIMODE).

15	14	13	12	11	10	9	8
SETZ	SYNC/SCIT	SDIAG1	SDIAG0	SDC2	SDC1	B2RB	B2RA

7	6	5	4	3	2	1	0
B1RB	B1RA	DRB	DRA	MSC3	MSC2	MS1	MS0

SETZ—Set L1TXD to Zero (valid only for the GCI interface)

0 = Normal operation.
1 = L1TXD output set to a logic zero (used in GCI activation).

SYNC/SCIT—SYNC Mode/SCIT Select Support

0 = One pulse wide prior to the 8-bit data.
1 = N pulses wide and envelopes the N-bit data.

SDIAG1, SDIAG0—Serial Interface Diagnostic Mode

00 = Normal operation.
01 = Automatic echo.
10 = Internal loopback.
11 = Loopback control.

SDC2—Serial Data Strobe Control 2

0 = SDS2 signal is asserted during the B2 channel.
1 = SDS1 signal is asserted during the B2 channel.

SDC1—Serial Data Strobe Control 1

0 = SDS1 signal is asserted during the B1 channel.
1 = SDS2 signal is asserted during the B1 channel.

B2RB, B2RA—B2 Channel Route in IDL/GCI Mode or CH-3 Route in PCM Mode

00 = Channel not supported.
01 = Route channel to SCC1.
10 = Route channel to SCC2 (if MSC2 is cleared).
11 = Route channel to SCC3 (if MSC3 is cleared).

B1 RB, B1 RA—B1 Channel Route in IDL/GCI Mode or CH-2 Route in PCM Mode

00 = Channel not supported.
01 = Route channel to SCC1.
10 = Route channel to SCC2 (if MSC2 is cleared).
11 = Route channel to SCC3 (if MSC3 is cleared).

DRB, DRA—D Channel Route in IDL/GCI Mode or CH-1 Route in PCM Mode

00 = Channel not supported.
01 = Route channel to SCC1.
10 = Route channel to SCC2 (if MSC2 is cleared).
11 = Route channel to SCC3 (if MSC3 is cleared).

MSC3—SCC3 Connection

0 = SCC3 is connected to the multiplexed serial interface.
1 = SCC3 is not connected to the multiplexed serial interface.

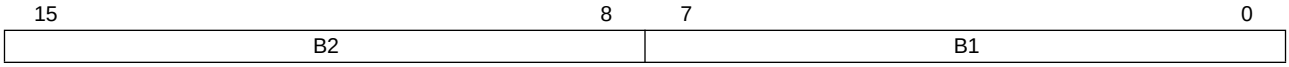
MSC2—SCC2 Connection

0 = SCC2 is connected to the multiplexed serial interface.
1 = SCC2 is not connected to the multiplexed serial interface.

MS1, MS0—Mode Supported

00 = NMSI mode.
01 = PCM mode.
10 = IDL mode.
11 = GCI interface.

E.2.1.1.3 Serial Interface Mask Register (SIMASK). This 16-bit register is located at offset $8B2. The SIMASK register is used to configure which bits on the B1 and B2 channels are used in the GCI and IDL modes. Bit 0 of SIMASK is the first bit transmitted and received on B1.



E.2.1.2 PER SCC REGISTERS. Each of the three SCCs has a set of the following six registers. These registers configure the SCC and the protocol operation. Some parameters and register bits are protocol independent. The UART functions have been given for those parameters and bits that are protocol specific.

E.2.1.2.1 Serial Configuration Register (SCON). This 16-bit register is located at offset $882 (SCC1), $892 (SCC2), and $8A2 (SCC3). The SCON register is used to select the clock source and baud rate for the SCC.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WOMS	EXTC	RCS	TCS	CD10	CD9	CD8	CD7	CD6	CD5	CD4	CD3	CD2	CD1	CD0	DIV4

WOMS—Wired-OR Mode Select

0 = TXD driver operates normally.
1 = TXD driver functions as an open-drain output and may be wired together with other TXD pins.

EXTC—External Clock Source

0 = The internal main clock is the source for the baud rate generator.
1 = The external clock on the TIN1 pin is the source for the baud rate generator.

TCS—Transmit Clock Source

0 = Transmit clock source is the baud rate generator output.
1 = Transmit clock source is the clock signal on TCLK pin.

RCS—Receive Clock Source

0 = Receive clock source is the baud rate generator output.
1 = Receive clock source is the clock signal on TCLK pin.

CD10-CD0—Clock Divide

Used to preset the 11-bit counter that is decremented at the prescaler output rate.

DIV4—SCC Clock Prescaler Divide by 4

0 = Divide-by-1 prescaler.
1 = Divide-by-4 prescaler.

E.2.1.2.2 SCC Mode Register (SCM). This 16-bit register is located at offset $884 (SCC1), $894 (SCC2), and $8A4 (SCC3). The SCM register configures the operation of the SCC and defines UART specific parameters.

15	14	13	12	11	10	9	8
TPM1	TPM0	RPM	PEN	UM1	UM0	FRZ	CL

7	6	5	4	3	2	1	0
RSTM	SL	DIAG1	DIAG0	ENR	ENT	MODE1	MODE0

TPM1,TMP0—Transmitter Parity Mode

00 = Odd parity; always send an odd number of ones.
01 = Force low parity; always send a zero in the parity bit position.
10 = Even parity; always send an even number of ones.
11 = Force high parity; always send a one in the parity bit position.

RPM—Receiver Parity Mode

0 = Odd parity.
1 = Even parity.

PEN—Parity Enable

0 = No parity.
1 = Parity is enabled for the transmitter and receiver.

UM1, UM0—UART Mode 1-0

00 = Normal UART operation.
01 = Nonautomatic multidrop mode. No automatic address recognition is performed.
10 = DDCMP protocol is implemented over the asynchronous channel.
11 = Automatic multidrop mode. The IMP automatically checks the value of the incoming address character and accepts the data following it only if the address matches the 8-bit value in either UADDR1 or UADDR2.

FRZ—Freeze Transmission

0 = Normal operation.
1 = The IMP stops transmitting after transmitting any data already transferred to the transmit FIFO.

CL—Character Length

0 = 7-bit character length. On receive, bit 7 is written to memory as a zero.
1 = 8-bit character length.

RTSM—RTS Mode

0 = $\overline{\text{RTS}}$ is asserted whenever the transmitter is enabled and there are characters to transmit.
1 = $\overline{\text{RTS}}$ is asserted whenever the transmitter is enabled.

SL—Stop Length

0 = One stop bit.
1 = Two stop bits.

DIAG1, DIAG0—Diagnostic Mode

00 = Normal operation.
01 = Loopback mode.
10 = Automatic echo.
11 = Software operation.

ENR—Enable Receiver

0 = Receiver is disabled.
1 = Receiver is enabled.

ENT—Enable Transmitter

0 = Transmitter is disabled.
1 = Transmitter is enabled.

MODE1, MODE0—Channel Mode

00 = HDLC.
01 = Asynchronous (UART and DDCMP).
10 = Synchronous DDCMP and V.110.
11 = BISYNC and Promiscuous (Transparent).

E.2.1.2.3 SCC Data Synchronization Register (DSR). This 16-bit register is located at offset $886 (SCC1) $896 (SCC2) and $8A6 (SCC3). Bits 14-12 of the DSR are used to program the length of the last stop bit transmitted. These bits are decoded as follows:

DSR(14-12)—Fractional Stop Bits

111 16/16 (default value after reset).
110 15/16.
101 14/16.
100 13/16.
011 12/16.
010 11/16.
001 10/16.
000 9/16.

E.2.1.2.4 UART Event Register (SCCE). This 8-bit register is located at offset $888 (SCC1) $898 (SCC2) and $8A8 (SCC3) on D15-D8 of a 16-bit data bus. The SCCE is used to report events recognized by the UART channel. Bits must be cleared by the user to avoid missing interrupt events. Bits are cleared by writing ones to the corresponding bit positions.

7	6	5	4	3	2	1	0
CTS	CD	IDL	BRK	CCR	BSY	TX	RX

CTS—Clear-To-Send Status Changed

0 = No interrupt.
1 = A change in the status of $\overline{\text{CTS}}$ was detected.

CD—Carrier Detect Status Changed

0 = No interrupt.
1 = A change in the status of $\overline{\text{CD}}$ was detected.

IDL—IDLE Sequence Status Changed

0 = No interrupt.
1 = A change in the status of the receive data serial line was detected.

BRK—Break Character Received

0 = No interrupt.
1 = Break character received.

CCR—Control Character Received

0 = No interrupt
1 = Control character received (with reject (R) character = 1) and stored in the receive control character register (RCCR).

BSY—Busy Condition

0 = No interrupt.
1 = A character was received and discarded due to lack of buffers.

TX—Tx Buffer

0 = No interrupt.
1 = A buffer has been transmitted on the UART channel (set only if the I bit in the Tx buffer descriptor is set).

RX—Rx Buffer

0 = No interrupt.
1 = A buffer was received on the UART channel (set only if the I bit in the Rx buffer descriptor is set).

E.2.1.2.5 UART Mask Register (SCCM). This 8-bit register is located at offset $88A (SCC1), $89A (SCC2), and $8AA (SCC3) on D15-D8 of a 16-bit data bus. The SCCM is used to enable and disable interrupt events reported by the SCCE. The mask bits correspond to the interrupt event bit shown in the SCCE. A bit should be set to one to enable the corresponding interrupt in the SCCE.

7	6	5	4	3	2	1	0
CTS	CD	IDL	BRK	CCR	BSY	TX	RX

E.2.1.2.6 UART Status Register (SCCS). This 8-bit register is located at offset $88C (SCC1), $89C (SCC2), and $8AC (SCC3), on D15-D8 of a 16-bit data bus. The SCCS register reflects the current status of the RXD, $\overline{CD}$, and $\overline{CTS}$ lines as seen by the SCC.

7	6	5	4	3	2	1	0
—	—	—	—	—	ID	CD	CTS

ID—Idle Status on the Receiver Line (valid only when the ENR bit is set and the receive clock is running)

0 = Receiver Line is not idling.
1 = Either $\overline{CD}$ is not asserted or the receiver line is idling while $\overline{CD}$ is asserted.

$\overline{CD}$—Carrier Detect Status Changed (valid only when the ENR bit is set and the receive clock is running)

0 = $\overline{CD}$ is asserted.
1 = $\overline{CD}$ is not asserted.

$\overline{CTS}$—Clear-To-Send Status Changed (valid only when the ENT bit is set and the transmit clock is running)

0 = $\overline{CTS}$ is asserted.
1 = $\overline{CTS}$ is not asserted.

E.2.1.3 GENERAL AND UART PROTOCOL-SPECIFIC PARAMETER RAM. Each SCC has 32 words of parameter RAM used to configure receive and transmit operation, store temporary parameters for the CP, and maintain counters. The first 14 words are general parameters, which are the same for each protocol. The last 18 words are specific to the protocol selected. The following sections discuss the parameters that the user must initialize to configure the UART operation.

E.2.1.3.1 RFCR/TFCR—Rx Function Code/Tx Function Code. This 16-bit parameter contains the function codes of the receive data buffers and transmit data buffers. The user must initialize the function codes (FC2-FC0) to a value less than 7.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	FC2	FC1	FC0	0	0	0	0	0	FC2	FC1	FC0	0	0	0	0

E.2.1.3.2 MRBLR—Maximum Rx Buffer Length. This 16-bit parameter defines the maximum receiver buffer length for each of the eight receive buffer descriptors.

E.2.1.3.3 MAX_IDL—Maximum IDLE Characters. This 16 bit parameter contains the maximum number of IDLE characters that the SCC will wait before closing a receive buffer.

E.2.1.3.4 BRKCR—Break Count Register. This 16-bit parameter defines the number of break characters to be transmitted when the STOP TRANSMIT command is executed.

E.2.1.3.5 PAREC—Receive Parity Error Counter. This 16-bit parameter is incremented when a character is received with a parity error.

E.2.1.3.6 FRMEC—Receive Framing Error Counter. This 16 bit parameter is incremented when no stop bit is detected in a received data string.

E.2.1.3.7 NOSEC—Receive Noise Counter. This 16-bit parameter is incremented when the three samples taken in the middle of a bit are not identical.

E.2.1.3.8 BRKEC—Receive Break Condition Counter. This 16-bit parameter is incremented when a break character string is detected.

E.2.1.3.9 UADDR1 and UADDR2. These 16-bit parameters contain the addresses used for address recognition.

E.2.1.3.10 RCCR—Receive Control Character Register. The RCCR is a 16-bit register. If a received character matches a defined control character for which the reject bit is set, the UART controller will write the control character into the lower 8 bits (D7-D0) of RCCR and generate a maskable interrupt. The upper 8-bits of RCCR are reserved.

E.2.1.3.11 Character1-Character7—Control Character 1-7. These seven 16-bit table entries define the control characters that should be compared to the incoming characters. For 7-bit characters, the eighth bit (bit 7) should be zero. Note that reserved bits in registers should be written as zeros.

15	14	13	12	11	10	9	8	7 0
E	R	—	—	—	—	—	—	CHARACTER1
E	R	—	—	—	—	—	—	CHARACTER2
E	R	—	—	—	—	—	—	CHARACTER3
E	R	—	—	—	—	—	—	CHARACTER4
E	R	—	—	—	—	—	—	CHARACTER5
E	R	—	—	—	—	—	—	CHARACTER6
E	R	—	—	—	—	—	—	CHARACTER7

E—End of Table

0 = This entry is valid.
1 = This entry is not valid. No valid entries lie beyond this entry.

R—Reject Character

0 = The control character is written into the receive data buffer, and the buffer is then closed. A new receive buffer is used if there is more data in the message.
1 = The control character is written to the RCCR instead of the received data butter. The current buffer is not closed.

E.2.1.3.12 Character8—Control Characters8. This 16-bit table entry defines a control character that should be compared to the incoming characters or is used to transmit out-of sequence characters, such as XON and XOFF.

15	14	13	12	11	10	9	8	7 … 0
E	R	REA	I	CT	0	0	A	CHARACTER8

E—End of Table

0 = This entry is valid as a receive control character.
1 = This entry contains a character to be transmitted out of sequence.

R—Reject Character

0 = Must be zero to use this entry as a flow control transmission character, otherwise, function is the same as for CHARACTERS 1-7.
1 = For receive control characters, the meaning is the same as for CHARACTERS 1-7.

REA—Ready

0 = Character is not ready for transmission
1 = Character is ready for transmission

I—Interrupt

0 = No interrupt.
1 = The TX bit in the event register will be set when this character is transmitted.

CT—Clear-to Send Lost

0 = $\overline{\text{CTS}}$ remained asserted.
1 = $\overline{\text{CTS}}$ was negated during transmission of this character.

A—Address

0 = Address bit of zero will be transmitted if a multidrop mode is chosen.
1 = Address bit of one will be transmitted if a multidrop mode is chosen.

E.2.1.4 RECEIVE BUFFER DESCRIPTORS. Each SCC has eight receive buffer descriptors. Each buffer descriptor consists of four words as shown below. Reserved bits in registers should be written as zeros.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET + 0	E	X	W	I	C	A	M	ID	—	—	BR	FR	PR	—	OV	CD
OFFSET +2	DATA LENGTH															
OFFSET +4	RX BUFFER POINTER															
OFFSET +6																

E.2.1.4.1 Receive BD Control/Status Word. To initialize the buffer, the user should write bits 15-12 and clear bits 11-0. The IMP clears bit 15 when the buffer is closed and sets bits 5-0 depending on which error occurred.

E—Empty

0 = This data buffer is full or has been closed due to an error condition.
1 = This data buffer is empty; must be set by the user to enable reception into this buffer.

X—External Buffer

0 = The data butter associated with this BD is in internal dual-port RAM.
1 = The data buffer associated with this BD is in external memory.

W—Wrap (final BD in table)

0 = This is not the last BD in the receive BD table.
1 = This is the last BD in the receive BD table.

I—Interrupt

0 = The RX bit in the event register is not set when this buffer is closed.
1 = The RX bit in the event register is set when this buffer is closed.

C—Control Character

0 = This buffer does not contain a control character.
1 = The last byte of this buffer contains a control character.

A—Address

0 = This buffer contains data only.
1 = The first byte of this data buffer is an address byte.

M—Address Match

0 = The address byte matched UADDR2.
1 = The address byte matched UADDR1.

ID—IDLE Reception

0 = Buffer not closed due to reception of maximum number of IDLE characters (MAX_IDL).
1 = Buffer closed due to reception of maximum number of IDLE characters (MAX_IDL).

Bits 7, 6, 2—Reserved for future use

BR—Break Received

0 = No break sequence was detected during reception into this buffer.
1 = A break sequence was detected during reception into this buffer.

FR—Framing Error

0 = No framing error was detected.
1 = A framing error was detected during reception of the last data byte in this buffer.

PR—Parity Error

0 = No parity error was detected.
1 = A character with a parity error was received and is located in the last byte of this buffer.

OV—Overrun

0 = No receiver overrun occurred.
1 = A receiver overrun condition occurred during buffer reception.

CD—Carrier Detect Lost (valid only in NMSI mode)

0 = No CD lost was detected.
1 = CD was negated during buffer reception.

E.2.1.4.2 Receive Buffer Data Length. This 16-bit value is written by the IMP to indicate the number of octets received into the data buffer.

E.2.1.4.3 Receive Buffer Pointer. This 32-bit value is written by the user to indicate the address where the data is to be stored.

E.2.1.5 TRANSMIT BUFFER DESCRIPTORS. Each SCC has eight transmit buffer descriptors. Each buffer descriptor consists of four words as shown below. Reserved bits in registers should be written as zeros.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET + 0	R	X	W	I	CR	A	P	—	—	—	—	—	—	—	—	CT
OFFSET +2	DATA LENGTH															
OFFSET +4	TX BUFFER POINTER															
OFFSET +6																

E.2.1.5.1 Transmit BD Control/Status Word. To initialize the buffer, the user should write bits 15-9 and clear bit 0. The IMP clears bit 15 when the buffer is transmitted or closed due to an error and sets bit 0 depending on which error occurred.

R—Ready

0 = This data buffer is not currently ready for transmission.
1 = This data buffer has been prepared by the user for transmission but has not yet been fully transmitted. Must be set by the user to enable transmission of the buffer.

X—External Buffer

0 = The data buffer associated with this BD is in internal dual port RAM.
1 = The data buffer associated with this BD is in external memory.

W—Wrap (final BD in table)

0 = This is not the last BD in the transmit BD table.
1 = This is the last BD in the transmit BD table.

I—Interrupt

0 = The TXB bit in the event register is not set when this buffer is closed.
1 = The TXB bit in the event register is set if this buffer closed without an error. If an error occurred, then TXE is set.

CR—Clear-to-Send Report

0 = The buffer following this buffer, if ready, will be transmitted with no delay; less precise CT bit reporting.
1 = Normal CTS lost (CT bit) error reporting, and two bits of idle occur between back-to-back buffers.

A—Address

0 = This buffer contains data only.
1 = This buffer contains address character(s) only.

P—Preamble

0 = No preamble sequence is sent.
1 = The UART sends a preamble sequence (set of 9 to 13 bits) before sending the data.

Bits 8–1—Reserved for future use

CT—CTS Lost

0 = No CTS or L1GR lost was detected during frame transmission.
1 = CTS in NMSI mode or L1GR in IDL/GCI mode was lost during frame transmission.

E.2.1.5.2 Transmit Buffer Data Length. This 16-bit value is written by the user to indicate the number of octets to be transmitted from the data butter.

E2.1.5.3 Transmit Buffer Pointer. This 32-bit value is written by the user to indicate the address of the first byte of data in the data buffer.

E.2.2 Programming the SCC for UART

This section gives a generic algorithm for programming an SCC to handle UART. The algorithm is intended to show what must be done and in what order to initialize the SCC and prepare the SCC for transmission and reception. The algorithm is not specific and assumes that the IMP and other on-chip peripherals have been initialized as required by the system hardware (timers, chip selects, etc.).

E.2.2.1 INITIALIZATION.

1. Write the port A and port B control registers (PACNT and PBCNT) to configure SCC2 or SCC3 serial interface pins as peripheral pins, if SCC2 or CC3 is used.
2. Write SIMODE to configure the SCCs physical interface.
3. Write SIMASK if IDL or GCI multiplexed mode was selected in SIMODE.

E.2.2.2 GENERAL AND UART PROTOCOL-SPECIFIC RAM INITIALIZATION.

4. Write RFCR/TFCR.
5. Write MRBLR.
6. Write MAX_IDLE.
7. Write BRKCR.
8. Write PAREC.
9. Write FRMEC.
10. Write NOSEC.
11. Write BRKEC.
12. Write UADDR1 and UADDR2.
13. Write CHARACTER1-8 in the control character table.

E.2.2.3 SCC INITIALIZATION.

14. Write SCON.
15. Write SCM without setting the ENR and ENT bits.
16. Write DSR.
17. Write SCCE with $FF to clear any previous events.
18. Write SCCM.
19. Write IMR.

E.2.2.4 SCC OPERATION.

20. Write the Rx buffer descriptor control/status, buffer pointer high, and buffer pointer low words for all of the buffer descriptors that are going to be used. Set the W bit in the last buffer descriptor to be used in the queue.
21. Prepare transmit buffers as required to transmit data on the SCC. Set the R bit in each Tx buffer descriptor's control/status word when the data buffer is ready for transmission. Set the W bit in the last Tx buffer descriptor in the table so that the IMP will use the first Tx buffer descriptor (after the user sets the R bit) for the next transmission.
22. Write SCM, setting the ENR and ENT bits to enable reception and transmission on the SCC.
23. Prepare more transmit buffers as required to transmit data on the SCC.

E.2.2.5 SCC INTERRUPT HANDLING.

1. Read the SCC event register.
2. Clear any unmasked bits that will be used in this interrupt routine.
3. Handle the interrupt events as required by the system.
4. Clear the appropriate SCC bit in the in-service register (ISR) of the interrupt controller.
5. Return from the interrupt.

E.3 TRANSPARENT PROGRAMMING REFERENCE SECTION

This subsection discusses the registers and parameters required to program an SCC for transparent operation. At the end of this subsection is a generic algorithm for programming the SCC.

E.3.1 Transparent Programming Model

The programming model and memory map for the transparent protocol is shown in E-1. The offsets for each SCC are given above each table. Some parameters are common to all protocols. The transparent parameters are shown for those entries that are protocol specific. Table E-3(b) depicts the general and protocol-specific parameter RAM for each SCC. The SCC registers are shown in Table E-3(c), and the communications processor registers are shown in Table E-3(d). Note that reserved bits in registers should be written as zeros.

Table E-1. (a)Transparent Programming Model Receive and Transmit Buffer Descriptors for SCCx

Initialized by User	Offset Hex	Name	Initialized by User	Offset Hex	Name
Yes	00	Rx BD 0 Control/Status	Yes	40	Tx BD 0 Control/Status
	02	Rx BD 0 Data Count	Yes	42	Tx BD 0 Data Count
Yes	04	Rx BD 0 Data Pointer (High Word)	Yes	44	Tx BD 0 Data Pointer (High Word)
Yes	06	Rx BD 0 Data Pointer (Low Word)	Yes	46	Tx BD 0 Data Pointer (Low Word)
Yes	08	Rx BD 1 Control/Status	Yes	48	Tx BD 1 Control/Status
	0A	Rx BD 1 Data Count	Yes	4A	Tx BD 1 Data Count
Yes	0C	Rx BD 1 Data Pointer (High Word)	Yes	4C	Tx BD 1 Data Pointer (High Word)
Yes	0E	Rx BD 1 Data Pointer (Low Word)	Yes	4E	Tx BD 1 Data Pointer (Low Word)
Yes	10	Rx BD 2 Control/Status	Yes	50	Tx BD 2 Control/Status
	12	Rx BD 2 Data Count	Yes	52	Tx BD 2 Data Count
Yes	14	Rx BD 2 Data Pointer (High Word)	Yes	54	Tx BD 2 Data Pointer (High Word)
Yes	16	Rx BD 2 Data Pointer (Low Word)	Yes	56	Tx BD 2 Data Pointer (Low Word)
Yes	18	Rx BD 3 Control/Status	Yes	58	Tx BD 3 Control/Status
	1A	Rx BD 3 Data Count	Yes	5A	Tx BD 3 Data Count
Yes	1C	Rx BD 3 Data Pointer (High Word)	Yes	5C	Tx BD 3 Data Pointer (High Word)
Yes	1E	Rx BD 3 Data Pointer (Low Word)	Yes	5E	Tx BD 3 Data Pointer (Low Word)
Yes	20	Rx BD 4 Control/Status	Yes	60	Tx BD 4 Control/Status
	22	Rx BD 4 Data Count	Yes	62	Tx BD 4 Data Count
Yes	24	Rx BD 4 Data Pointer (High Word)	Yes	64	Tx BD 4 Data Pointer (High Word)
Yes	26	Rx BD 4 Data Pointer (Low Word)	Yes	66	Tx BD 4 Data Pointer (Low Word)
Yes	28	Rx BD 5 Control/Status	Yes	68	Tx BD 5 Control/Status
	2A	Rx BD 5 Data Count	Yes	6A	Tx BD 5 Data Count
Yes	2C	Rx BD 5 Data Pointer (High Word)	Yes	6C	Tx BD 5 Data Pointer (High Word)
Yes	2E	Rx BD 5 Data Pointer (Low Word)	Yes	6E	Tx BD 5 Data Pointer (Low Word)
Yes	30	Rx BD 6 Control/Status	Yes	70	Tx BD 6 Control/Status
	32	Rx BD 6 Data Count	Yes	72	Tx BD 6 Data Count
Yes	34	Rx BD 6 Data Pointer (High Word)	Yes	74	Tx BD 6 Data Pointer (High Word)
Yes	36	Rx BD 6 Data Pointer (Low Word)	Yes	76	Tx BD 6 Data Pointer (Low Word)
Yes	38	Rx BD 7 Control/Status	Yes	78	Tx BD 7 Control/Status
	3A	Rx BD 7 Data Count	Yes	7A	Tx BD 7 Data Count
Yes	3C	Rx BD 7 Data Pointer (High Word)	Yes	7C	Tx BD 7 Data Pointer (High Word)
Yes	3E	Rx BD 7 Data Pointer (Low Word)	Yes	7E	Tx BD 7 Data Pointer (Low Word)

NOTE: The offset is from the MC68302 base address + ($400 for SCC1, $500 for SCC2, or $600 for SCC3)

.

Table E-1 (b). Transparent Programming Model (Continued) General Parameter and Transparent Protocol-Specific RAM for SCCx

Initialized by User	Offset Hex	Name		Initialized by User	Offset Hex	Name
Yes	80	RFCR	TFCR		A2	Reserved
Yes	82	MRBLR			A4	Reserved
	84	Rx Internal State			A6	Reserved
	86	Reserved	Rx Internal Buffer No.		A8	Reserved
	88 8A	Rx Internal Data Pointer (High Word) Rx Internal Data Pointer (Low Word)			AA	Reserved
	8C	Rx Internal Byte Count			AC	Reserved
	8E	Rx Temp			AE	Reserved
	90	Tx Internal State			B0	Reserved
	92	Reserved	Tx Internal Buffer No.		B2	Reserved
	94 96	Tx Internal Data Pointer (High Word) TX Internal Data Pointer (Low Word)			B4	Reserved
	98	Tx Internal Byte Count			B6	Reserved
	9A	Tx Temp			B8	Reserved
	9C	Reserved			BA	Reserved
	9E	Reserved			BC	Reserved
	A0	Reserved			BE	Reserved

NOTE: The offset is from the MC68302 base address + ($400 for SCC1, $500 for SCC2, or $600 for SCC3).

Table E-1 (c). SCCx Register Set

Initialized by User	Offset Hex	Name	
	00	Reserved	
Yes	02	SCC Configuration Register (SCON)	
Yes	04	SCC Mode Register (SCM)	
Yes	06	SCC Data Synchronization Register (DSR)	
Yes	08	Event Register (SCCE)	Reserved
Yes	0A	Mask Register (SCCM)	Reserved
	0C	Status Register (SCCS)	Reserved
	0E	Reserved	

NOTE: The offset is from the MC68302 base address + ($880 for SCC1, $890 for SCC2, or $8A0 for SCC3).

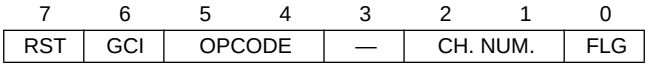
Table E-1 (d). General Registers (Only One Set)

Initialized by User	Offset Hex	Name	
	860	Command Register (CR)	Reserved
Yes	8B2	Serial Interface Mask Register (SIMASK)	
Yes	8B4	Serial Interface Mask Register (SIMODE)	

NOTE: The offset is from the MC68302 base address.

E.3.1.1 COMMUNICATIONS PROCESSOR (CP) REGISTERS. The CP has one set of three registers that configure the operation of the serial interface for all three SCCs. These registers are discussed in the following paragraphs.

E.3.1.1.1 Command Register (CR). The command register is an 8-bit register located at offset $860 (on D15-D8 of a 16-bit data bus). This register is used to issue commands to the CP. The user should set the FLG bit when a command is written to the command register. The CP clears the FLG bit during command processing to indicate that it is ready for the next command. Reserved bits in registers should be written as zeros.



RST—Software Reset Command (set by the user and cleared by the CP)

0 = No software reset command issued or cleared by CP during software reset sequence.
1 = Software reset command (FLG bit should also be set if it is not already set).

GCI—GCI Commands

0 = Normal operation.
1 = The OPCODE bits are used for GCI commands (user should set CH. NUM. to 10 and FLG to 1).

OPCODE—Command Opcode

00 = STOP TRANSMIT Command.
01 = RESTART TRANSMIT Command.
10 = ENTER HUNT MODE Command.
11 = Reset receiver BCS generator (used only in BISYNC mode).

Bit 3—Reserved (should be set to zero by the user when the command register is written)

CH. NUM.—Channel Number

00 = SCC1.
01 = SCC2.
10 = SCC3.
11 = Reserved.

FLG—Command Semaphore Flag (set by the user and cleared by the CP upon command completion)

0 = CP is ready to receive a new command (should be checked before issuing next command to the CP).
1 = Command register contains a command to be executed or one that is currently being executed.

E.3.1.1.2 Serial Interface Mode Register (SIMODE). This 16-bit register is located at offset $8B4. The SIMODE register is used to configure the serial interface operation.

15	14	13	12	11	10	9	8
SETZ	SYNC/SCIT	SDIAG1	SDIAG0	SDC2	SDC1	B2RB	B2RA

7	6	5	4	3	2	1	0
B1RB	B1RA	DRB	DRA	MSC3	MSC2	MS1	MS0

SETZ—Set L1TXD to Zero (valid only for the GCI interface)

0 = Normal Operation
1 = L1TXD output set to a logic zero (used in GCI activation)

SYNC/SCIT—SYNC Mode/SCIT Select Support

0 = One pulse wide prior to the 8-bit data.
1 = N pulses wide and envelopes the N-bit data.

SDIAG1, SDIAG0—Serial Interface Diagnostic Mode

00 = Normal operation.
01 = Automatic Echo.
10 = Internal loopback.
11 = Loopback Control.

SDC2—Serial Data Strobe Control 2

0 = SDS2 signal is asserted during the B2 channel.
1 = SDS1 signal is asserted during the B2 channel.

SDC1—Serial Data Strobe Control 1

0 = SDS1 signal is asserted during the B1 channel.
1 = SDS2 signal is asserted during the B1 channel.

B2RB, B2RA—B2 Channel Route in IDL/GCI Mode or CH-3 Route in PCM Mode

00 = Channel not supported.
01 = Route channel to SCC1.
10 = Route channel to SCC2 (if MSC2 is cleared).
11 = Route channel to SCC3 (if MSC3 is cleared).

B1RB, B1 RA—B1 Channel Route in IDL/GCI Mode or CH-2 Route in PCM Mode

00 = Channel not supported.
01 = Route channel to SCC1.
10 = Route channel to SCC2 (if MSC2 is cleared).
11 = Route channel to SCC3 (if MSC3 is cleared).

DRB, DRA— D Channel Route in IDL/GCI Mode or CH-1 Route in PCM Mode

00 = Channel not supported.
01 = Route channel to SCC1.
10 = Route channel to SCC2 (if MSC2 is cleared).
11 = Route channel to SCC3 (if MSC3 is cleared).

MSC3—SCC2 Connection

0 = SCC3 is connected to the multiplexed serial interface.
1 = SCC3 is not connected to the multiplexed serial interface.

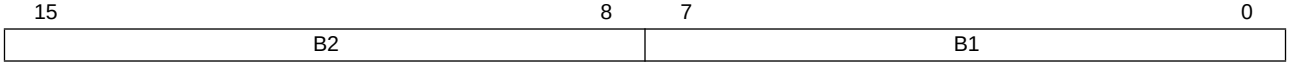
MSC2—SCC2 Connection

0 = SCC2 is connected to the multiplexed serial interface.
1 = SCC2 is not connected to the multiplexed serial interface.

MS1, MS0—Mode Supported

00 = NMSI mode.
01 = PCM mode.
10 = IDL mode.
11 = GCI interface.

E.3.1.1.3 Serial Interface Mask Register (SIMASK). This 16-bit register is located at offset $8B2. The SIMASK register is used to configure which bits on the B1 and B2 channels are used in the GCI and IDL modes. Bit 0 of SIMASK is the first bit transmitted and received on B1.



E.3.1.2 PER SCC REGISTERS. Each of the three SCCs has a set of the following six registers. These registers configure the SCC and the protocol operation. Some parameters and register bits are protocol independent. The transparent functions have been given for those parameters and bits that are protocol specific.

E.3.1.2.1 Serial Configuration Register (SCON). This 16-bit register is located at offset $882 (SCC1), $892 (SCC2), and $8A2 (SCC3). The SCON register is used to select the clock source and baud rate for the SCC.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WOMS	EXTC	TCS	RCS	CD10	CD9	CD8	CD7	CD6	CD5	CD4	CD3	CD2	CD1	CD0	DIV4

WOMS—Wired-OR Mode Select

0 = TXD driver operates normally.
1 = TXD driver functions as an open-drain output and may be wired together with other TXD pins.

EXTC—External Clock Source

0 = The internal main clock is the source of the baud rate generator.
1 = The external clock on the TIN1 pin is the source for the baud rate generator.

TCS—Transmit Clock Source

0 = Transmit clock source is the baud rate generator output.
1 = Transmit clock source is the clock signal on TCLK pin.

RCS—Receive Clock Source

0 = Receive clock source is the baud rate generator output.
1 = Receive clock source is the clock signal on TCLK pin.

CD1 0-CD0—Clock Divider

Used to preset the 11-bit counter that is decremented at the prescaler output rate.

DIV4—SCC Clock Prescaler Divide by 4

0 = Divide-by-1 prescaler.
1 = Divide-by-4 prescaler.

E.3.1.2.2 SCC Mode Register (SCM). This 16-bit register is located at offset $884 (SCC1), $894 (SCC2), and $8A4 (SCC3) The SCM register configures the operation of the SCC and defines transparent-specific parameters. Note that reserved bits in registers should be written as zeros.

15	14	13	12	11	10	9	8
—	EXSYN	NTSYN	REVD	—	—	—	—

7	6	5	4	3	2	1	0
—	—	DIAG1	DIAG0	ENR	ENT	MODE1	M0DE0

BIT 15—Reserved for future use; should be written with zero.

EXSYN—External Sync

When set, the SCC receiver uses the L1SY1/$\overline{CD1}$, $\overline{CD2}$, or $\overline{CD3}$ pins to synchronize the receiver and transmitter to the beginning of a transparent frame.

NTSYN—No Transmit SYNC

This bit must be set for the SCC to operate in the transparent mode.

REVD—Reverse Data

When this bit is set, the receiver and transmitter will reverse the character bit order, transmitting the most significant bit first.

Bits 11-6—Reserved for future use; should be written with zero.

DIAG1, DIAG0—Diagnostic Mode

00 = Normal operation.
01 = Loopback mode.
10 = Automatic echo.
11 = Software operation.

ENR—Enable Receiver

0 = Receiver is disabled.
1 = Receiver is enabled

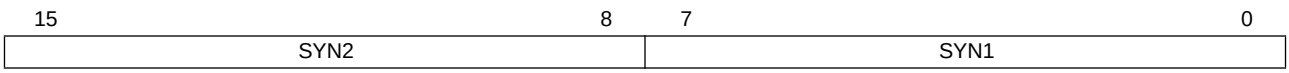
ENT—Enable Transmitter

0 = Transmitter is disabled.
1 = Transmitter is enabled.

MODE1, MODE0—Channel Mode

00 = HDLC.
01 = Asynchronous (UART and DDCMP).
10 = Synchronous DDCMP and V.110.
11 = BISYNC and Promiscuous (Transparent).

E.3.1.2.3 SCC Data Synchronization Register (DSR). This 16-bit register is located at offset $886 (SCC1), $896 (SCC2), and $8A6 (SCC3). The DSR specifies the pattern used for the receive frame synchronization procedure if the EXSYN bit is cleared. For transparent, the DSR may be set to any desired pattern. The DSR value after reset is $7E7E.



E.3.1.2.4 Transparent Event Register (SCCE). This 8-bit register is located at offset $888 (SCC1), $898 (SCC2), and $8A8 (SCC3) on D15-D8 of a 16-bit data bus. The SCCE is used to report events recognized by the transparent channel. Bits must be cleared by the user to avoid missing interrupt events. Bits are cleared by writing ones to the corresponding bit positions

7	6	5	4	3	2	1	0
CTS	CD	—	TXE	RCH	IBSY	TX	RX

CTS—Clear-To-Send Status Changed

0 = No interrupt.
1 = A change in the status of $\overline{\text{CTS}}$ was detected.

CD—Carrier Detect Status Changed

0 = No interrupt.
1 = A change in the status of $\overline{\text{CD}}$ was detected.

TXE—Tx Error

0 = No interrupt.
1 = An error ($\overline{\text{CTS}}$ lost or underrun) occurred on the transmitter channel.

RCH—Receive Character

0 = No interrupt.
1 = A 16-bit word was received on the transparent channel.

BSY—Busy Condition

0 = No interrupt.
1 = A frame was received and discarded due to lack of buffers.

TX—Tx Buffer

0 = No interrupt.
1 = A buffer has been transmitted on the transparent channel (set only if the I bit in the Tx buffer descriptor is set).

RX—Rx Buffer

0 = No interrupt.
1 = A buffer was received on the transparent channel (set only if the I bit in the Rx buffer descriptor is set).

E.3.1.2.5 Transparent Mask Register (SCCM). This 8-bit register is located at offset $88A (SCC1), $89A (SCC2), and $8AA (SCC3) on D15-D8 of a 16-bit data bus. The SCCM is used to enable and disable interrupt events reported by the SCCE. The mask bits correspond to the interrupt event bit shown in the SCCE. A bit should be set to a one to enable the corresponding interrupt in the SCCE. Note that reserved bits in registers should be written as zeros.

7	6	5	4	3	2	1	0
CTS	CD	—	TXE	RTE	IBSY	TX	RX

E.3.1.2.6 Transparent Status Register (SCCS). This 8-bit register is located at offset $88C (SCC1), $89C (SCC2), and $8AC (SCC3)on D15-D8 of a 16-bit data bus. The SCCS register reflects the current status of the CD and CTS lines as seen by the SCC.

7	6	5	4	3	2	1	0
—	—	—	—	—	—	CD	CTS

$\overline{CD}$—Carrier Detect Status Changed (valid only when the ENR bit is set and the receive clock is running)

0 = CD is asserted.
1 = CD is not asserted.

$\overline{CTS}$—Clear-To-Send Status Changed (valid only when the ENT bit is set and the transmit clock is running)

0 = $\overline{CTS}$ is asserted.
1 = $\overline{CTS}$ is not asserted.

E.3.1.3 GENERAL AND TRANSPARENT PROTOCOL-SPECIFIC PARAMETER RAM. Each SCC has 32 words of parameter RAM used to configure receive and transmit operation, store temporary parameters for the CP, and maintain counters. The first 14 words are general parameters, which are the same for each protocol. The last 18 words are specific to the protocol selected. The following subsections discuss the parameters that the user must initialize to configure the transparent operation.

E.3.1.3.1 RFCR/TFCR—Rx Function Code/Tx Function Code. This 16-bit parameter contains the function codes of the receive data buffers and transmit data buffers. The user must initialize the function codes (FC2-FC0) to a value less than 7.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	FC2	FC1	FC0	0	0	0	0	0	FC2	FC1	FC0	0	0	0	0

E.3.1.3.2 MRBLR—Maximum Rx Buffer Length. This 16-bit parameter defines the maximum receiver buffer length for each of the eight receive buffer descriptors.

E.3.1.4 RECEIVE BUFFER DESCRIPTORS. Each SCC has eight receive buffer descriptors. Each buffer descriptor consists of four words as shown below. Reserved bits in registers should be written as zeros.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET + 0	E	X	W	I	—	—	—	—	—	—	—	—	—	—	OV	CD
OFFSET +2	DATA LENGTH															
OFFSET +4	RX BUFFER POINTER															
OFFSET +6																

E.3.1.4.1 Receive BD Control/Status Word. To initialize the buffer, the user should write bits15-12 and clear bits1-0. The IMP clears bit 15 when the buffer is closed and sets bits 5-0 depending on which error occurred.

E—Empty

0 = This data buffer is full or has been closed due to an error condition.
1 = This data buffer is empty; must be set by the user to enable reception into this buffer.

X—External Buffer

0 = The data buffer associated with this BD is in internal dual-port RAM.
1 = The data buffer associated with this BD is in external memory.

W—Wrap (final BD in table)

0 = This is not the last BD in the receive BD table.
1 = This is the last BD in the receive BD table.

I—Interrupt

0 = No interrupt is generated when this buffer is closed.
1 = The RX bit in the event register is set when this buffer is closed.

Bits 11-2—Reserved for future use; should be written with zero by the user.

OV—Overrun

0 = No receiver overrun occurred.
1 = A receiver overrun condition occurred during frame reception.

CD—Carrier Detect Lost (valid only in NMSI mode)

0 = No CD lost was detected.
1 = CD was negated during frame reception.

E.3.1.4.2 Receive Buffer Data Length. This 16-bit value is written by the IMP to indicate the number of data bytes received into the data buffer.

E.3.1.4.3 Receive Buffer Pointer. This 32 bit value is written by the user to indicate the address where the data is to be stored.

E.3.1.5 TRANSMIT BUFFER DESCRIPTORS. Each SCC has eight transmit buffer descriptors. Each buffer descriptor consists of four words as shown below. Reserved bits in registers should be written as zeros.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET + 0	R	X	W	I	L	—	—	—	—	—	—	—	—	—	UN	CT
OFFSET +2	DATA LENGTH															
OFFSET +4	TX BUFFER POINTER															
OFFSET +6																

E.3.1.5.1 Transmit BD Control/Status Word. To initialize the buffer, the user should write bits 15-11 and clear bits 1-0. The IMP clears bit 15 when the buffer is transmitted or closed due to an error and sets bits 1-0 depending on which error occurred.

R—Ready

0 = This data buffer is not currently ready for transmission.
1 = This data buffer has been prepared by the user for transmission but has not yet been fully transmitted. Must be set by the user to enable transmission of the buffer.

X—External Buffer

0 = The data buffer associated with this BD is in internal dual-port RAM.
1 = The data buffer associated with this BD is in external memory.

W—Wrap (final BD in table)

0 = This is not the last BD in the transmit BD table.
1 = This is the last BD in the transmit BD table.

I—Interrupt

0 = No interrupt is generated when this buffer is closed.
1 = The TX bit in the event register is set if this buffer closed without an error. If an error occurred, then TXE is set.

L—Last in Frame

0 = This buffer is not the last buffer in the transmitted block.
1 = This buffer is the last buffer in the transmitted block.

Bits 10-2—Reserved for future use; should be written with zero by the user.

UN—Underrun

0 = No transmitter underrun occurred.
1 = A transmitter underrun condition occurred while transmitting the associated data buffer.

CT—CTS Lost

0 = No CTS or L1GR lost was detected during frame transmission.
1 = CTS in NMSI mode or L1GR in IDL/GCI mode was lost during frame transmission.

E.3.1.5.2 Transmit Buffer Data Length. This 16-bit value is written by the user to indicate the number of data bytes to be transmitted from the data buffer.

E.3.1.5.3 Transmit Buffer Pointer. This 32-bit value is written by the user to indicate the address of the first byte of data in the data buffer.

E.3.2 Programming the SCC for Transparent

This section gives a generic algorithm for programming an SCC to handle the transparent protocol. The algorithm is intended to show what must be done and in what order to initialize the SCC and prepare the SCC for transmission and reception. The algorithm is not specific and assumes that the IMP and other on-chip peripherals have been initialized as required by the system hardware (timers, chip selects, etc.).

E.3.2.1 CP INITIALIZATION.

1. Write the port A and port B control registers (PACNT and PBCNT) to configure SCC2 or SCC3 serial interface pins as peripheral pins, if SCC2 or SCC3 is used.
2. Write SIMODE to configure the SCCs physical interface.
3. Write SIMASK if IDL or GCI multiplexed mode was selected in SIMODE.

E.3.2.2 GENERAL AND TRANSPARENT PROTOCOL-SPECIFIC RAM INITIALIZATION.

4. Write RFCR/TFCR.
5. Write MRBLR.

E.3.2.3 SCC INITIALIZATION.

6. Write SCON.
7. Write SCM without setting the ENR and ENT bits.
8. Write DSR.
9. Write SCCE with $FF to clear any previous events.
10. Write SCCM.
11. Write IMR.

E.3.2.4 SCC OPERATION.

12. Write the Rx buffer descriptor control/status, buffer pointer high, and buffer pointer low words for all of the buffer descriptors that are going to be used. Set the W bit in the last buffer descriptor to be used in the queue.
13. Prepare transmit buffers as required to transmit data on the SCC. Set the R bit in each Tx buffer descriptor's control/status word when the data buffer is ready for transmission. Set the W bit in the last Tx buffer descriptor in the table so that the IMP will use the first Tx buffer descriptor (after the user sets the R bit) for the next transmission.
14. Write SCM, setting the ENR and ENT bits to enable reception and transmission on the SCC.
15. Prepare more transmit buffers as required to transmit data on the SCC.

E.3.2.5 SCC INTERRUPT HANDLING.

1. Read the SCC event register.
2. Clear any unmasked bits that will be used in this interrupt routine.
3. Handle the interrupt events as required by the system.
4. Clear the appropriate SCC bit in the in-service register (ISR) of the interrupt controller.
5. Return from the interrupt.

APPENDIX F
DESIGN CHECKLIST

When integrating the MC68302 into an application, it may be helpful to go through the following design checklist. In this checklist are a number of common problems and their resolutions that have been found while debugging real MC68302 applications.

1. **Version, Mask**
   Older versions of the MC68302—called Rev A and Rev B with masks “B14M” written on the device—do not contain all the features listed in this manual. These devices have ceased production and are longer available. A newer version called Rev C, which is identified by mask number “C65T” or later, contains all the features listed in this manual. The missing features in the old versions include clock output control and a few other minor differences.
2. **External Pin Configurations**
   A good checklist of external pin configurations may be found in D.1 Minimum System Configuration. Common problems are also listed in some of the following paragraphs.
3. **Clock Present**
   If you are using an external clock source to the 68302, make sure that it is driving the clock input within 10 msec of powerup. Otherwise, part damage can occur.
4. **$\overline{\text{AVEC}}$, $\overline{\text{DTACK}}$**
   If $\overline{\text{AVEC}}$ is not used, it needs to be pulled high (to + 5 V); otherwise, erratic behavior and bus cycles may occur. A pullup resistor may be used, if desired. If $\overline{\text{AVEC}}$ is used, it should be asserted instead of $\overline{\text{DTACK}}$ (not in addition to $\overline{\text{DTACK}}$) during interrupt acknowledge cycles.
5. **Pullup, $\overline{\text{DTACK}}$**
   Sometimes a 10K-ohm resistor may not be strong enough to pull up $\overline{\text{DTACK}}$ to provide adequate rise times to the $\overline{\text{DTACK}}$ signal. This is a loading-dependent issue.
6. **Pullup, Floating $\overline{\text{BR}}$, $\overline{\text{FRZ}}$, BUSW**
   Unexpected behavior can result if the signals $\overline{\text{BR}}$, $\overline{\text{FRZ}}$, BUSW, or other inputs are left floating. Of these, the most common mistake is to leave $\overline{\text{FRZ}}$ floating.
   If no external requests are made, $\overline{\text{BR}}$ may be pulled directly high. If external requests are made, $\overline{\text{BR}}$ may need to be pulled high through a resistor, such as 1 K ohm, to guarantee adequate $\overline{\text{BR}}$ rise time to meet bus arbitration specifications.
7. **Pullup, $\overline{\text{IPL}}$**
   $\overline{\text{IPL}}$ lines should be pulled high if not used. These signals may be pulled directly high, if desired.
8. **$\overline{\text{RESET}}$, Rise Time**
   The rise time of the $\overline{\text{RESET}}$ and $\overline{\text{HALT}}$ pins after a total system reset must be within

spec (see spec 32), or the total system reset may not be terminated correctly and unusual behavior may occur. Also, when using the RESET instruction to reset the MC68302 internal peripherals, a strong pullup (such as 1.2K ohms) may be required for proper rise times.

9. **Chip Select, $\overline{\text{DTACK}}$ Generation**
   If unexpected behavior is occurring on the $\overline{\text{DTACK}}$ line, such as early or late assertion or negation, then often the problem is traced to another component on the board that is generating $\overline{\text{DTACK}}$ at the same time as the IMP $\overline{\text{DTACK}}$ generator. An often overlooked source for $\overline{\text{DTACK}}$ generation is the emulator overlay memory (if an emulator is used), which may have been inadvertently configured to overlap with a chip-select area of the IMP. Very unusual behavior can result, especially if the number of wait states programmed for the chip select is different from that of the overlay memory.
10. **Chip Select, Option Register**
    When setting up the chip select option register to operate as an address mask, the value programmed into the base address mask field should normally have all ones at the left end and all zeros at the right end. Any zeros mixed between ones in the option register base address mask field will cause multiple responses of the chip-select pin throughout the MC68302 address space. In most applications, this is undesirable and confusing.
11. **A0–A7, D0–D7, Initialize, Reset**
    The M68000 registers A0–A7 and D0–D7 do not have predefined values upon a total system reset. The use of uninitialized (or partially uninitialized) registers can cause intermittent and erratic software behavior since the initialized register values may vary from reset to reset.
12. **BSET Instruction, Byte**
    To use the bit set instruction (BSET) to set a bit in the lower half of a word-sized register or memory location, one MUST perform a byte operation on the byte address, i.e., word_address + 1. For example, to set bit zero of a word address $3000, one must issue BSET.B #0,#$3001. There is no BSET.W instruction available in the MC68000. This also applies to the BTST instruction.
13. **Exception Vector Table, Initialize, Reset**
    Failure to provide the M68000 core exception vector table with vectors can cause erratic behavior when an exception (such as bus error) occurs. Make sure that the boot ROM/EPROM has exception vectors initialized, and that the initial reset vector causes the program to start at an address above the vector table.
14. **Stack Pointer, Initialize, Reset**
    The stack pointer must be initialized to an EVEN address; otherwise, address errors will occur when the stack is first used.
15. **Parameter RAM, Initialize, Reset**
    To use SCCs with specific protocols, both general-purpose parameters and protocol-specific parameters must be initialized. Failure to initialize the parameter RAM will result in erratic behavior since the parameter RAM does not have predefined values upon a total system reset.
16. **EQU, Parameter RAM**
    Very unusual problems with the SCCs are often traced to the fact that the source code

does not correctly “equate” locations in parameter RAM to intended addresses. A simple typo in an assembler EQU or a C #DEFlNE directive can cause 1) the intended parameter not to be set and 2) another parameter to be set to a wrong value.

17. **Function Code, Initialize, Reset**
    To use IDMA, SDMA, and/or DRAM refresh, their corresponding function code registers MUST be initialized. Setting the function codes in these registers to “111” will prevent the MC68302 chip selects from asserting. Failure to initialize these registers often results in their function codes having the value “111”, since these registers are in dual-port RAM and do not have predefined values upon a total system reset.

18. **Function Code, External Bus Master**
    When an external bus master is using the chip selects on the MC68302 with the external master's external memory accesses, make sure that the external bus master drives the function code lines to something other than “111". If the function code lines are driven or left floating to “111”, the external cycle will be interpreted by the MC68302 as an interrupt acknowledge cycle, and the chip selects will not be asserted during the cycle.

19. **BAR, Write**
    The BAR MUST BE written by an instruction following a total system reset of the MC68302 since this register resides in the MC68302, not in the memory. It is not sufficient or required for the EPROM on the target board to have the desired BAR value stored in the EPROM location $0F2 (the address of BAR). When using an emulator, a symptom of this problem can be that the code works in the emulator overlay memory, but not on the target.

20. **DRAM Refresh**
    When using the DRAM refresh unit, one cannot refresh locations $0F0–0FF of an external DRAM if an MC68302 chip select is used to select that DRAM. Locations $0F0–0FF are designated as the reserved area of the IMP that contains the BAR and SCR, and chip selects will not activate on accesses to these addresses. The remedy is simply to use a different DRAM refresh starting address besides $0. Also note that the DRAM refresh access is a byte read, not a word read.

21. **Watchdog Timer**
    If the MC68302 watchdog timer is never turned off or refreshed, an unexpected interrupt at level 4 can occur. Also, an unexpected $\overline{\text{RESET}}$ can occur if the $\overline{\text{WDOG}}$ pin is externally connected to $\overline{\text{HALT}}$ and $\overline{\text{RESET}}$. The solution is to disable the watchdog timer after reset. Note that the watchdog timer is not related to the hardware watchdog, which is a completely separate unit that monitors bus activity.

22. **Underrun, Overrun, Clock Lines, Schmitt-Triggers**
    If a transmit underrun or a receive overrun is reported but the data rates are too slow to suggest an actual underrun or overrun, the problem may be in the clock lines. Glitched or badly ringing clocks (on the TCLK or RCLK pins) can cause SCCs to enter either of the above error states. Even though Schmitt-triggers are implemented on the IMP clocks lines, a very slow rise/fall time coupled with a large amount of noise on the lines can override the hysteresis protection and affect the ability of the SCC to correctly sample and clock data. Internal clocks generated by the IMP do not cause this problem.

23. **SCCs, Diagnostics**
When an unexplained SCC problem arises, three important debugging techniques can be used to help isolate the problem area. First, to see if the problem is related to the external M68000 bus, try configuring the transmit and/or receive buffers to be located in the internal dual-port RAM. In this way, the external bus will not be required for transmission or reception. Second, to see if the problem is related to the serial interface pins, try setting the DIAG1–DIAG0 bits in the SCM register to loopback mode or software operation. These techniques allow transmission and reception to be tested without proper settings of the serial interface control pins. Loopback mode is also a good test mechanism to build into the application board test code. Third, to see if the problem is related to rise times, noise, or glitches on external serial clocks, try using the SCC internal baud rate generator or try generating a clock externally with another baud rate generator or timer and routing that clock externally to the SCC.

24. **UART, MAX_IDL**
One common problem with UART mode is the setting of the MAX_IDL value. The maximum value is $0000, and the minimum value is $0001. A large value causes the receive buffer not to close until the entire idle period is complete. At slow serial data rates, this time can be several minutes.

25. **Write Zeroes into Reserved Locations**
Zero fill all reserved locations to prevent unexpected behavior.

26. **Event Registers, Read-Modify-Write**
Do not use instructions that execute a read-modify-write cycle (e.g., ANDI, ORI, BSET, BCLR) to clear bits in the event registers. A bit in such a register is cleared by writing a one, rather than a zero, to the bit. The use of instructions that execute a read-modify-write cycle inadvertently clears all bits in the event register. The clearing of all bits in an event register could result in “lost” events. For example, when an interrupt handler intentionally or unintentionally clears all bits of an event register upon exit of the handler, all events occurring between the time of the entrance and exit of the interrupt handler are lost. A proper handling of event registers is to clear particular bits near the entrance of an interrupt handler. A proper instruction to use is the MOVE instruction (e.g., “MOVE #$01, SCCE” clears bit 0 of the SCCE register).

27. **Microcode, RAM**
If microcode from RAM is used (i.e., one of the packages available from Motorola), the microcode must be downloaded into the MC68302 dual-port RAM prior to setting location $0F8 to $0001. If this sequence is not followed, the CP will not function properly, regardless of which protocol is selected.

INDEX

A

Address
- AS 3-42
- Decode
  - Conflict 2-13
- Decode Conflict 3-44, 3-52, 3-53
- Error 2-9
- Mode 2-3
- Space 2-6

AS 2-11, 3-14, 3-42, 3-53, 3-59
Asynchronous Baud Rate 4-26
Automatic echo 4-30
AVEC 2-8, 2-11, 3-18, 3-22, 3-55, 5-12

B

Base Address Register (BAR) 2-12
Basic Rate ISDN 1-6
Baud Rate Generator 4-25, 5-19
BCLM 3-56
BCLR 3-18, 3-51, 3-54, 3-55, 3-58, 4-5, 5-10
BCLR See Interrupt
BCLR See Signals
BERR 2-13, 3-44, 3-46, 3-52, 3-53, 3-59, 3-60, 5-6
- Channel Number 3-67

BERR See Signals
BG 3-54, 3-56, 3-58, 3-59, 5-11
BGACK 3-56, 3-59, 4-4, 5-11
BISYNC 4-91
- BDLE 4-91
- BISYNC 4-91
- Carrier Detect Lost 4-92
- Clear-To-Send Lost 4-92
- Control Characters 4-89, 4-102
- CRC Error 4-93
- DLE 4-85
- DLE-DLE 4-87
- DLE-SYNC 4-87
- DSR 4-88
- Event Register 4-96, 4-98, 4-100
- EXSYN 4-93
- FIFO 4-92
- Frames 4-85
- Mask Register 4-101
- Memory Map 4-87
- Overrun Error 4-92
- Parity Error 4-92
- Programming the BISYNC 4-101
- RESET BCS CALCULATION Command 4-101
- RESTART TRANSMIT Command 4-92
- RTS 4-95
- Rx BD 4-95
- SCCE 4-100
- SCCM 4-101
- SYN1-SYN2 4-87
- SYNCs 4-85
- Transmitter Underrun 4-92
- Tx BD 4-97

BISYNC Controller 4-84
BISYNC MODE Register 4-93
Block Check Sequence 4-94
BR 3-54, 3-56, 3-58, 3-59, 5-11
Buffer 4-36
- BDs 4-34
- Buffer Descriptor 4-32
- Circular Queue 4-32
- Descriptors 2-14, 3-66
- RBD
- TBD
- Transmit BDs 4-34

Buffer Descriptor 4-32
Bus
- Arbiter 3-58
- Arbitration 3-14, 3-56, 5-11
- Bandwidth 3-5, 3-66
- Cycle 3-12
- Cycles 3-2, 3-59
- Error 2-9, 2-11, 2-13

Exception 3-66
Processing 2-7
Exceptions 3-14
Grant (BG) 5-11
Grant Acknowledge (BGACK) 5-11
Latencies 3-58
Master 2-7, 3-56, 3-58, 4-5
Request (BR) 5-11
SDMA Retry 4-41
Signal Summary 5-13
Bus Error on SDMA Access 4-41
Bus Master 4-5
BUSW 2-1, 3-39, 5-6, See Signals, See Timers

C

C/I Channel 4-141
Carrier Detect Lost 4-61
CD 4-28, 4-40
CEPT 4-19
Chip-Select 2-13
Address Decode Conflict 3-44
AS 3-42
Base Address 3-48
Base Register 3-45
BERR 3-44, 3-46
Block Sizes 3-45
CS0 3-44, 3-46, 3-55, 5-23
DTACK 3-42, 3-47, 3-48
EMWS 3-43
Option Register 3-45
Priority 3-43
Read-Only 3-48
RESET 3-44
RMSCT 3-45
Test and Set 3-45
Write Protect Violation 3-44
Write-Only 3-48
Chip-Select Timing 6-24
Circular Queue 4-32
Clear-to-Send Report 4-63
Clock
Asynchronous Baud Rate 4-26
Baud Rate Generator 4-25
CLKO 3-49, 5-5
Clock Divider 4-26
Crystal 3-50
EXTAL 3-49
Synchronous Baud Rate 4-27
XTAL 3-49
Clock Divider 4-26
CMOS Level 5-2
Command 4-5
ENTER HUNT MODE Command 4-36, 4-37, 4-54, 4-126, 4-128, 4-130
RESET BCS CALCULATION Command 4-101
RESTART TRANSMIT Command 4-73, 4-92, 4-128
STOP TRANSMIT Command 4-36, 4-37, 4-42, 4-48, 4-49, 4-69, 4-71, 4-128, 4-130
TIMEOUT Command 4-142
TRANSMIT ABORT REQUEST Command 4-142
Command Execution Latency 4-7
Command Register 4-5
Communications Processor 4-1
Configuration
MC68302 IMP Control 2-12
System Registers 2-14
CP 1-6, 4-1
CQFP 7-2
CR 4-5
CS0 3-44, 3-46, 3-55, 5-23
CTS 4-28, 4-40

D

DACK 5-20
Data Type
Asynchronous Bit Rate Adaption 4-119
FIFO 4-120
Overrun Error 4-120
Rx BD 4-120
SCCE 4-123
SCCM 4-124
Synchronization Error 4-120
Synchronous Bit Rate Adaption 4-118
Terminal Equipment 4-117
Transmitter Underrun 4-120
Tx BD 4-122
V.110 80-Bit Frames 4-119
V.110 Event Register 4-123
V.110 Mask Register 4-124

V.110 Rate Adaption 4-117
Data Types 2-3
DDCMP
Asynchronous DDCMP Mode 4-46
Carrier Detect Lost 4-109
Clear-To-Send Lost 4-109
CRC Error 4-109
DDCMP Address Recognition 4-108
DDCMP Event Register 4-112, 4-115, 4-116
DDCMP Frames 4-102
DDCMP Mask Register 4-117
DDCMP Memory Map 4-105
DDCMP Mode Register 4-110
DDLE 4-108
DENQ 4-108
DSOH 4-108
DSR 4-105
DSYN1 4-107
FIFO 4-108
Framing Error 4-109
Overrun Error 4-109
Parity Error 4-110
RTS 4-111
Rx BD 4-111
SCCE 4-116
SCCM 4-117
SYN1-SYN2 4-110
Transmitter Underrun 4-108
Tx BD 4-114
DDCMP Controller 4-102
Disable CPU 5-6
AVEC 3-55
BCLR 3-55
BG 3-54
BR 3-54
CS0 3-55
DTACK 3-55
EMWS 3-55
IOUT0/IOUT1/IOUT2 3-55
Low-Power Modes 3-55
RMC 3-55
SAM 3-55
Vector Generation Enable (VGE) 3-55
Disabled 4-39, 4-43
Disabling the SCCs 4-42
DISCPU 3-54, 5-6
DONE 5-20
DRAM Refresh 3-66, 3-67, 4-35
BERR Channel Number 3-67
Buffer Descriptors 3-66
Bus Bandwidth 3-66
Bus Exception 3-66
ERRE 3-68
PB8 3-32
SDMA 3-67
DREQ 5-20
DSR 4-32
DTACK 2-8, 3-21, 3-34, 3-47, 3-48, 3-53, 3-54, 3-55, 5-6, 5-12, See Dual-Port RAM, See Signals
Dual-Port RAM 1-5, 2-14, 3-33
BR 3-34
DTACK 3-34
EMWS 3-34
SAM 3-34

E

E 2-11, See Signals
EMWS (External Master Wait State) 3-34, 3-53, 3-54, 3-55
Enable Receiver 4-31
Enable Transmitter 4-31
ENTER HUNT MODE Command 4-6, 4-36, 4-37, 4-43, 4-50, 4-72, 4-89, 4-107
Envelope Mode 4-17
ERRE 3-68, See DRAM Refresh
Error Counters 4-55, 4-75, 4-93, 4-110
Event Registers 2-19
Exception
Bus 3-14
Bus Error 3-14
Halt 3-14
PB8 3-66
Processing 2-7, 2-11, exception vector is determined 2-8
Reset 3-14
Retry 3-14
Stack Frame 2-10
Vectors 2-8
EXRQ 3-17
EXTAL 3-62, 5-2, 5-4
External
Bus Master 2-7, 3-58

Bus Master See Bus
Master Wait State (EMWS) 3-54
External Clock 4-25
External Loopback 4-30
External Master Wait State 3-53

F

FIFO 4-2, 4-49
Framing Error 4-61
FRZ (Freeze) 3-65, 5-7
Function Codes 2-6, 3-7, 4-34, 4-36, 3-55, 5-12
Comparison 2-14
FC2-FC0 2-13, 5-12
Register 3-7

G

GCI 4-7, 5-14, 5-15
C/I Channel 4-141
Interface 4-14
IOM2 4-14
Monitor Channel Protocol 4-141
SCIT 4-14, 4-16, 4-20
SDS1 4-14
SIMASK 4-22
SIMODE 4-19
SMC Channels 4-10
TIC 4-14
TIMEOUT Command 4-142
TRANSMIT ABORT REQUEST Command 4-142
Transparent Mode 4-140
GCI Command 4-6
GCI Interface 4-14
GCI See Signals

H

HALT 2-13, 5-6, See Signals
Hardware Watchdog 3-59
AS 3-59
BERR 3-59, 3-60
HDLC
Abort Sequence 4-74
Carrier Detect Lost 4-74
Clear-To-Send Lost 4-73
CRC 4-69
CRC Error 4-75
CRC16 4-75
CRC32 4-75
CTS 4-73, 4-76
FIFO 4-73, 4-74
Flag Sharing 4-75
Flags between Frames 4-75
HDLC Address Recognition 4-72
HDLC Event Register 4-79, 4-81, 4-82
HDLC Frame 4-68
HDLC Mask Register 4-84
HDLC Memory Map 4-70
HDLC Mode Register 4-75
HMASK 4-72
Idles between Frames 4-76
MFLR 4-73
Nonoctet Aligned Frame 4-74
NRZI 4-76
Overrun Error 4-74
RESTART TRANSMIT Command 4-73
Retransmission 4-76
RTS 4-76
Rx BD 4-76
RXB 4-74, 4-79, 4-84
RXF 4-74, 4-75, 4-79, 4-84
SCCE 4-82
SCCM 4-84
STOP TRANSMIT Command 4-69, 4-71
TBD
Transmitter Underrun 4-73
Tx BD 4-81
TXB 4-81
TXE 4-73, 4-81, 4-84
HDLC Controller 4-67

I

IAC 2-13, 2-14
IACK7 3-18, 3-22, 5-21
IDL 4-7, 5-14, 5-15
ISDN Terminal Adaptor 4-11
SDS1 4-12
Signals 4-12
SIMASK 4-22
SIMODE 4-19
SMC Channels 4-10
IDL Interface 4-11
IDL See Signals

IDL Signals 4-12
Idle Status 4-40
IDMA (Independent DMA Controller) 3-54, 3-58, 5-10, 5-11, 5-20
AS 3-14
BERR 3-15
BGACK 3-14
BR 3-14
Bus
Cycle 3-12
Error 3-14
Exceptions 3-14
Bus Arbitration 3-14
DONE 3-4, 3-8, 3-13
DREQ 3-5, 3-6, 3-8, 3-12
DTACK 3-9, 3-21
External
Burst Mode 3-11
Halt 3-14, 3-15
MOVEP 3-10
Operand Packing 3-10
Registers 3-31
Reset 3-14
Retry 3-14
Transfer Rate 3-2
IMP Features
CP 1-3
IMR 3-14, 3-32, 4-39
Instruction Type Variations 2-5
Instructions
BERR 3-14, 3-15
HALT 3-14, 3-15
MOVE 2-19
MOVEP 3-10
Read-Modify-Write 2-11, 2-19
RESET 3-14
RTE 3-29
Test and Set 2-11
Internal Loopback 4-20
Internal Registers 1-6, 2-16
Internal Requests (INRQ) 3-17, See Interrupt
Interrupt
Acknowledge 2-7, 2-8, 2-11, 2-13, 3-17, 3-19, 3-20
Autovector 2-8, 2-11
AVEC 3-18, 3-22, 5-12
BCLR 3-18
Control Pins 5-11
Controller 3-15
DTACK 3-21
EXRQ 3-17
FC2-FC0 3-17
IACK7 3-18, 3-22, 5-21
IMR 3-32, 4-39
INRQ 3-17
IOUT0 5-12
IOUT0/IOUT1/IOUT2 3-55
IPL2-IPL0 3-17, 3-19
IPR 3-20, 3-26, 4-39
IRQ1 3-17
IRQ6 3-17
IRQ7 3-17, 3-19
ISR 3-28, 4-39
Masking 3-20
Mode
Dedicated 2-11
Normal 2-11
Nested 3-19
PB11 3-19, 3-21
Pending 5-10
Priority 3-20
Processing 2-11
Registers
SR 2-2, 2-11
Request 3-58
RTE 3-29
SCCE 4-39
SCCM 4-39
Spurious 2-9, 2-11
SR 3-17, 3-20
Vector 3-17, 3-21, 3-27
Vector Number 3-22
Interrupt Acknowledge 2-11
Interrupt Pending Register (IPR) 3-20
IOM2 4-14
IOUT0 5-12
IOUT0/IOUT1/IOUT2 3-55
IOUT0/IOUT1/IOUT2 3-55, See Disable CPU, See Interrupt, See Signals
IPEND 3-58, See Signals
IPL2-IPL0 3-17, 3-19, 5-11, See Interrupt, See Signals
IPR 4-39

IRQ1 3-17, 5-12, See Interrupt, See Signals
IRQ6 3-17, See Interrupt, See Signals
IRQ7 3-17, 3-19, See Interrupt, See Signals
ISDN 5-14, Terminal Adaptor 4-11
ISR 4-39

L

L1SY0 4-17
LAPB 4-67
LAPD 4-67
Loopback Control 4-20
Loopback Mode 4-29, 4-137
  External Loopback 4-30
  Internal Loopback 4-20
  Loopback Control 4-20
Lowest Power Mode 3-62
  RESET 3-62
  with External Clock 3-62
Low-Power 4-43
Low-Power Mode 3-61
  LPEN 3-63
  LPREC 3-64

M

Main Controller 4-1
MAX_IDL 4-47
MC145474 4-11
MC68000/MC68008 Modes 2-1
Microcode 3-34
Mode
  Dedicated 2-11
  Normal 2-11
Modem Signals 4-19
Monitor Channel Protocol 4-141
MOVE 2-19, See Instructions
MRBLR 4-36
Multiplexed Interfaces 4-8

N

NC1 5-23
NC1 See Signals
Nested Interrupt 3-19
NMSI 4-7, 4-19, 5-14
  BRG1 5-18
  CD1 5-17
  CTS1 5-17
  Modem Signals 4-19
  NMSI1 5-15
  NMSI2 5-18
  NMSI3 5-19
  RTS1 5-17
  SIMODE 4-19
Normal Operation 4-28

O

One-Clock-Prior Mode 4-17
Output Delays 4-30

P

Parallel I/O Port
  DREQ 3-31
  IMR 3-32
  PB11 3-31, 3-32
  PB8 3-32, 3-66
  Port A 3-29, 5-18, 5-19, 5-20
    Control Register 3-29
    Data Direction Register (PADDR) 3-30
  Port B 3-29, 5-21, 5-22
    Control Register 3-31
    Data Direction Register (PBDDR) 3-31
Parameter RAM 3-34
Parity Error 4-61
PB11 3-19, 3-21, 3-31, 3-32, See DRAM Refresh, See Interrupt, See Parallel I/O Port, See Signals
PB8 3-32, 3-66
PCM 4-7
PCM Channel 4-17
PCM Highway 5-14, 5-15
  Envelope Mode 4-17
  L1SY0 4-17
  One-Clock-Prior Mode 4-17
  PCM Channel 4-17
  PCM Highway Mode 4-16
  RTS 4-18
  SIMODE 4-19
  Time Slots 4-18
Pending Interrupt 5-10
Performance 4-23
Pin Assignments 7-1

Pin Grid Array 7-1
Port A/B
Parallel I/O 3-29
Power Consumption 3-60
Power Dissipation 6-3
Power-Saving Tips 3-60
PQFP 7-2
Priority Interrupts 3-20
Privilege State 2-6, 2-8
Protocol Parameters 2-15
Pullup Resistors 5-23

R

RAM
Dual-Port 1-5, 3-33
Parameter 3-34
System 3-34
RBD
Read-Modify-Write 2-11, 2-19, See Instructions
Read-Modify-Write Cycle 6-11, 6-12
Receive BDs 4-34
Received Control Character Register 4-51
Reception Errors 4-54
Registers
Base Address 2-12
Event 2-19
Internal 1-6, 2-16
Interrupt In-Service (ISR) 3-28
Interrupt Mask (IMR) 3-32
Interrupt Pending (IPR) 3-26
Port A
Control (PACNT) 3-29
Data Direction (PADDR) 3-30
Port B
Control (PBCNT) 3-31
Data Direction Register (PBDDR) 3-31
Status 2-2, 2-8, 2-11, 3-17, 3-20
System Configuration 2-12, 2-14
System Control (SCR) 2-12, 2-13
RESET 2-19, 3-41, 3-44, 3-62, 5-6, 5-22
Instruction 2-7, 2-13, 2-19
Reset 4-39
SMC Interrupt Requests 4-145
SMC Loopback 4-141
SMC Memory Structure 4-142
TIMEOUT Command 4-142
Total System 2-13
TRANSMIT ABORT REQUEST Command 4-142
RESET BCS CALCULATION Command 4-89
RESTART TRANSMIT Command 4-6, 4-49, 4-72, 4-89, 4-107
Reverse DATA 4-94
Revision Number 2-15
RISC Processor 4-1
RMC 2-11, 3-53, 3-55, 3-58, 5-9
RTE 3-29
RTS 4-18, 4-28

S

SAM 3-34, 3-54, 3-55
SAM See Dual-Port RAM
SCC 2-15
Asynchronous Baud Rate 4-26
Baud Rate Generator 4-25
Buffer Descriptor 4-32
CD 4-28, 4-40
Clock Divider 4-26
CTS 4-28, 4-40
Disabled 4-39, 4-43
DSR 4-32
Enable Receiver 4-31
ENTER HUNT MODE Command 4-36, 4-37
External Loopback 4-30
Function Code 4-34, 4-36
Idle Status 4-40
IPR 4-39
Low-Power 4-43
MRBLR 4-36
Normal Operation 4-28
Output Delays 4-30
Performance 4-23
Promiscuous Operation 4-124
RBD
Receive BDs 4-34
Reset 4-39
RTS 4-28
SCC Initialization 4-38
SCCE 4-39
SCCM 4-39

SCCS 4-40
SCM 4-43
SCON 4-24, 4-43
SDMA Retry 4-41
Software Operation 4-31
STOP TRANSMIT Command 4-36, 4-37, 4-42
Synchronous Baud Rate 4-27
TBD
TIN1/TIN2 5-22
Totally Transparent 4-124
Transmit Data Delays 4-28
SCC Initialization 4-38
SCC Mode Register 4-27
SCC Parameter RAM 4-35
SCC Transparent Mode 4-41
SCCE 4-39
SCCM 4-39
SCCS 4-40
SCCs 4-22
SCIT 4-14, 4-16, 4-20
SCM 4-43
SCON 4-24, 4-43
SCP 2-15
Enable Signals 4-136
Loopback Mode 4-137
SCP Master 4-136
Serial Communication Port 4-136
SPCLK 4-136
SPI Slave 4-136
SPRXD 4-136
SPTXD 4-136
SCP Port 5-19
SCR (System Control Register) 2-12, 2-13
SDLC 4-67
SDMA (Serial DMA Controller) 3-54, 3-58, 5-10, 5-11
SDMA Channel 4-3, 4-23, 4-37
BCLR 4-5
BGACK 4-4
Bus Master 4-5
SDMA Retry 4-41
SDS1 4-12, 4-14
Serial Channels Physical Interface 4-7
Serial Communication Controllers 4-22
Serial Communication Port 4-136, 6-29
SIB 3-1
Signals
Address Decode Conflict 3-44
AS 2-11, 3-14, 3-42, 3-53, 3-59
AVEC 2-8, 2-11, 3-18, 3-22, 3-55, 5-12
BCLR 3-18, 3-51, 3-54, 3-55, 3-58, 4-5, 5-10
BERR 2-13, 3-44, 3-46, 3-52, 3-53, 3-59, 3-60, 5-6
BG 3-54, 3-56, 3-58, 3-59, 5-11
BGACK 3-14, 3-56, 3-59, 4-4, 5-11
BR 3-14, 3-34, 3-54, 3-56, 3-58, 3-59, 5-11
BRG1 5-18
BUSW 2-1, 3-39, 5-6
CD 4-28, 4-40, 4-129, 4-131
CD1 5-17
CLKO 3-49, 5-5
CS 2-13, 3-53
CS0 3-44, 3-46, 3-55, 5-23
CTS 4-28, 4-40, 4-73, 4-76, 4-129
CTS1 5-17
DACK 5-20
DISCPU 3-54, 5-6
DONE 3-4, 3-8, 3-13, 5-20
DREQ 3-5, 3-6, 3-8, 3-12, 5-20
DTACK 2-8, 3-9, 3-21, 3-42, 3-47, 3-48, 3-53, 3-54, 3-55, 5-6, 5-12
E 2-11
EXTAL 3-49, 3-62, 5-2, 5-4
FC2-FC0 3-17, 5-12
FRZ 3-65, 5-7
GCI 5-15
HALT 2-13, 5-6
IAC 2-13, 2-14
IACK7 3-18, 3-22, 5-21
IDL 5-15, 5-18
IDMA 5-20
ILP0 5-11
IOUT0 5-12
IOUT0/IOUT1/IOUT2 3-55
IPEND 3-58
IPL2-IPL0 3-17, 3-19
IRQ1 3-17, 5-12
IRQ6 3-17
IRQ7 3-17, 3-19
L1SY0 4-17, 4-129
NC1 5-23
NMSI2 5-18

NMSI3 5-19
PB11 3-19
PCM Highway 5-15
Port A 5-18, 5-19, 5-20
Port B 5-21, 5-22
RESET 2-7, 2-13, 2-19, 3-41, 3-44, 3-62, 5-6, 5-22
RMC 2-11, 3-53, 3-55, 3-58, 5-9
RTS 4-18, 4-28, 4-57, 4-76, 4-95, 4-111, 4-129
RTS1 5-17
SCP 5-19
SDS1 4-12, 4-14
SPCLK 4-136
SPRXD 4-136
SPTXD 4-136
TIN1/TIN2 3-37, 5-22
TOUT1/TOUT2 3-37, 5-22
VMA 2-11
VPA 2-11, 5-12
WDOG 3-31, 3-41, 5-22
XTAL 3-49, 5-4
SIMASK 4-19, 4-22
SIMODE 4-19
SMC 2-15, 4-140
Monitor Channel Protocol 4-141
Serial Management Controllers 4-140
Transparent Mode 4-140
Using GCI 4-140
Using IDL 4-140
SMC Channels 4-10
SMC Interrupt Requests 4-145
SMC Loopback 4-141
SMC Memory Structure 4-142
SMCs 4-35
Software Operation 4-31
Software Reset Command 4-5
SPCLK 4-136
SPI Slave 4-136
SPRXD 4-136
SPTXD 4-136
SR (Status Register) 2-2, 2-8, 2-11, 3-17, 3-20
STOP TRANSMIT Command 4-6, 4-36, 4-37, 4-42, 4-88, 4-106
Supervisor
Data Space 2-12
Stack 2-9
State 2-7
Synchronous Baud Rate 4-27
System
Configuration Registers 2-12
Control Registers (SCR) 2-12, 2-13
System Control Register (SCR) 3-50
System Integration Block (SIB) 3-1
System RAM 3-34

T

T1 4-19
TBD
Thermal Characteristics 6-1
TIC 4-14
Time Slots 4-18
TIMEOUT Command 4-142
Timers 3-35
BUSW 3-39
Prescaler 3-37
RESET 3-41
Resolution 3-37
TIN1/TIN2 3-37, 5-22
TOUT1/TOUT2 3-37, 5-22
WDOG 3-31, 3-41
TIN1/TIN2 3-37, 5-22, See SCC, See Signals, See Timers
TOUT1/TOUT2 3-37, 5-22, See Signals, See Timers
TRANSMIT ABORT REQUEST Command 4-142
Transmit BDs 4-34
Transmit Data Delays 4-28
Transparent
Busy Condition 4-131
Carrier Detect Lost 4-130
CD 4-129, 4-131
Clear-To-Send Lost 4-130
CTS 4-129
DSR 4-127, 4-129
ENTER HUNT MODE Command 4-126, 4-128, 4-130
EXSYN 4-129, 4-131
EXSYN Bit 4-129
External Sync Mode 4-131
FIFO 4-130
GCI 4-130
IDL 4-130

L1SY0 4-129
NTSYN 4-131
PCM Highway 4-129
Promiscuous Operation 4-124
RESTART TRANSMIT Command 4-128
REVD 4-131
RTS 4-129
RXBD 4-132
SCCE 4-135
SCCM 4-136
STOP TRANSMIT Command 4-128, 4-130
SYN1-SYN2 4-129
Totally Transparent 4-124
Transmitter Underrun 4-130
Transparent Event Register 4-132, 4-134
Transparent Mask Register 4-136
Transparent Memory Map 4-127
Transparent Synchronization 4-128
Tx BD 4-133
Transparent Controller 4-124
Transparent Mode 4-140
Transparent Mode Register 4-131
TTL Levels 5-2

U

UART 4-41
Address Recognition 4-50
Asynchronous DDCMP 4-46
Automatic Address Recognition 4-48
Automatic Multidrop Mode 4-50
Break Characters 4-48
BREAK Sequence 4-55
BRKCR 4-48, 4-49
Character Length 4-57
Control Characters 4-51
DDCMP 4-57
DSR 4-56
ENTER HUNT MODE Command 4-54
FIFO 4-53, 4-54, 4-57
Flow Control 4-51
Fractional Stop Bits 4-46, 4-55
Frame Format 4-44
Framing Error 4-55
FRZ 4-57
Idle Characters 4-47
IDLE Sequence 4-55
Multidrop Configuration 4-45
Multidrop Environment 4-50
Multidrop Mode 4-57
Noise Error 4-55
Overrun 4-61
Parity Error 4-55
Parity Mode 4-56
Preamble Sequence 4-63
PROFIBUS 4-44
Programming Example 4-67
Reception of Idles 4-61
RTS 4-57
RX 4-54, 4-55, 4-60, 4-66
Rx BD 4-58
SCCE 4-64
SCCM 4-66
Send Break 4-53
Send Preamble 4-54
Stop Bit 4-58
STOP TRANSMIT Command 4-49
Transmission Error 4-54
TX 4-53, 4-54, 4-62
Tx BD 4-62
UADDR1 4-50
UART Event Register 4-53, 4-60, 4-62, 4-64
UART Mask Register 4-66
UART Memory Map 4-47
UART Mode Register 4-56
XOFF 4-51, 4-52, 4-67
XON 4-52, 4-67
UART Controller 4-44
User State 2-7
Using GCI 4-140
Using IDL 4-140

V

Value 3-38
Vector
Generation Enable (VGE) 3-55
Interrupt 3-17, 3-21, 3-27
Number 2-8, 2-11, 3-22
Table 2-12
Vector Generation Enable 3-55
VMA 2-1, See Signals
VPA 2-11, 5-12, See Signals

W

Wait-State 1-5
Wakeup Timer 4-54
Watchdog (WDOG) 3-31, 3-41, 5-22, See Signals, See Timers
  Hardware 3-59
  Timer 3-41
Wired-OR 4-25
Write Protect Violation 3-44, 3-52

X

XTAL 3-49, 5-4, See Clock, See Signals