MC68356UM/AD

User's Manual

MOTOROLA

PREFACE

The complete documentation package for the MC68356 consists of MC68356UM/AD, the *MC68356 Signal Processing Communications Engine User's Manual;* MC68356/D, the *MC68356 Signal Processing Communications Engine Product Brief;* DSP56KFAMUM/AD the *DSP56000 Digital Signal Processor Family Manual;* and M68000PM/AD, the *M68000 Programmer's Reference Manual.* Information on Plastic Ball Grid Array (PBGA) layout considerations is available in AN1231/D, "Plastic Ball Grid Array Application Note", available from your local Motorola sales office.

The *MC68356 Signal Processing Communications Engine User's Manual* describes the programming, capabilities, registers, and operation of the MC68356; the *MC68356 Signal Processing Communications Engine Product Brief* provides a brief description of the MC68356 capabilities; the *DSP56000 Digital Signal Processor Family Manual* describes programming and the instruction set for the DSP engine; and the *M68000 Programmer's Reference Manual* describes programming and the instruction set for the IMP processor.

This user's manual is organized as follows:

Section 1	Introduction
Section 2	Signal Description
Section 3	IMP and DSP Clocking and Low Power Modes
Section 4	68000 Core
Section 5	Memory Map
Section 6	System Integration Block
Section 7	Communications Processor
Section 8	PCMCIA Controller
Section 9	DSP Memory Modules and Operating Modes
Section 10	DSP Port A
Section 11	DSP Host Port
Section 12	DSP Serial Ports
Section 13	IEEE 1149 Test Access Port (TAP)
Section 14	Electrical Specifications
Section 15	Ordering Information and Mechanical Data
Appendix A	SCC Performance
Appendix B	Development Tools and Support
Appendix C	DSP Bootstrap Program

Applications and Technical Information

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

— Sales Offices —

UNITED STATES

ALABAMA, Huntsville	(205) 464-6800
ARIZONA, Tempe	(602) 897-5056
CALIFORNIA, Agoura Hills	(818) 706-1929
CALIFORNIA, Los Angeles	(310) 417-8848
CALIFORNIA, Irvine	(714) 753-7360
CALIFORNIA, Roseville	(916) 922-7152
CALIFORNIA, San Diego	(619) 541-2163
CALIFORNIA, Sunnyvale	(408) 749-0510
COLORADO, Colorado Springs	(719) 599-7497
COLORADO, Denver	(303) 337-3434
CONNECTICUT, Wallingford	(203) 949-4100
FLORIDA, Maitland	(407) 628-2636
FLORIDA, Pompano Beach/ Fort 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) 481-810(
MASSACHUSETTS, Woburn	(617) 932-970(
MICHIGAN, Detroit	(313) 347-680(
MINNESOTA, Minnetonka	(612) 932-150(
MISSOURI, St. Louis	(314) 275-738(
NEW JERSEY, Fairfield	(201) 808-240(
NEW YORK, Fairport	(716) 425-400(
NEW YORK, Hauppauge	(516) 361-700(
NEW YORK, Poughkeepsie/Fishkill	(914) 473-810;
NORTH CAROLINA, Raleigh	(919) 870-435!
OHIO, Cleveland	(216) 349-310(
OHIO, Columbus Worthington	(614) 431-849;
OHIO, Dayton	(513) 495-680(
OKLAHOMA, Tulsa	(800) 544-949(
OREGON, Portland	(503) 641-368'
PENNSYLVANIA, Colmar	(215) 997-102(
Philadelphia/Horsham	(215) 957-410(
TENNESSEE, Knoxville	(615) 690-559;
TEXAS, Austin	(512) 873-200(
TEXAS, Houston	(800) 343-269;
TEXAS, Plano	(214) 516-510(
VIRGINIA, Richmond	(804) 285-210(
WASHINGTON, Bellevue	(206) 454-416(
Seattle Access	(206) 622-996(
WISCONSIN, Milwaukee/Brookfield	(414) 792-012;

Field Applications Engineering Available Through All Sales Offices

CANADA

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

INTERNATIONAL

AUSTRALIA, Melbourne	(61-3)887-0711
AUSTRALIA, Sydney	(61(2)906-3855
BRAZIL, Sao Paulo	55(11)815-4200
CHINA, Beijing	86 505-2180
FINLAND, Helsinki	358-0-35161191
Car Phone	358(49)211501
FRANCE, Paris/Vanves	33(1)40 955 900
GERMANY, Langenhagen/ Hanover	49(511)789911
GERMANY, Munich	49 89 92103-0
GERMANY, Nuremberg	49 911 64-3044
GERMANY, Sindelfingen	49 7031 69 910
GERMANY,Wiesbaden	49 611 761921
HONG KONG, Kwai Fong	852-4808333
Tai Po	852-6668333
INDIA, Bangalore	(91-812)627094
ISRAEL, Tel Aviv	972(3)753-8222
ITALY, Milan	39(2)82201
JAPAN, Aizu	81(241)272231
JAPAN, Atsugi	81(0462)23-0761
JAPAN, Kumagaya	81(0485)26-2600
JAPAN, Kyushu	81(092)771-4212
JAPAN, Mito	81(0292)26-2340
JAPAN, Nagoya	81(052)232-1621
JAPAN, Osaka	81(06)305-1801
JAPAN, Sendai	81(22)268-4333
JAPAN, Tachikawa	81(0425)23-6700
JAPAN, Tokyo	81(03)3440-3311
JAPAN, Yokohama	81(045)472-2751
KOREA, Pusan	82(51)4635-035
KOREA, Seoul	82(2)554-5188
MALAYSIA, Penang	60(4)374514
MEXICO, Mexico City	52(5)282-2864
MEXICO, Guadalajara	52(36)21-8977
Marketing	52(36)21-9023
Customer Service	52(36)669-9160
NETHERLANDS, Best	(31)49988 612 11
PUERTO RICO, San Juan	(809)793-2170
SINGAPORE	(65)2945438
SPAIN, Madrid	34(1)457-8204
or	34(1)457-8254
SWEDEN, Solna	46(8)734-8800
SWITZERLAND, Geneva	41(22)7991111
SWITZERLAND, Zurich	41(1)730 4074
TAIWAN, Taipei	886(2)717-7089
THAILAND, Bangkok	(66-2)254-4910
UNITED KINGDOM, Aylesbury	44(296)395-252

FULL LINE REPRESENTATIVES

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

HYBRID COMPONENTS RESELLERS

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

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Table Number	Title	Page Number

SECTION 1
INTRODUCTION

The MC68356 is the first commercially available monolithic device to offer a general purpose digital signal processor, CISC microprocessor, and RISC microprocessor on a single chip. Included in the list of features of the MC68356 are a static integrated multiprotocol communications processor (IMP) whose features form a superset of the popular MC68302, a static DSP56002 based 24-bit digital signal processor with greatly expanded on-board memory spaces, a PCMCIA slave interface with a UART block that emulates the UART 16550, and a fully programmable low-power management system. These features make the MC68356 ideally suited for applications requiring DSP technology, portability, low power consumption, and high integration. The planned modem version of the MC68356 features fully functional ROM resident modem datapump code implementing popular dial line modem protocols, including the new V.34 standard.

1.1 MC68356 KEY FEATURES

The following list summarizes the key features of the MC68356. The features are divided into separate sections for the communications processor and DSP portions of the device. New features, which do not exist on the current versions of the MC68302 and the DSP56002, are highlighted in **bold** text.

Integrated Multiprotocol Processor (IMP) Features:

- CPM68000 Core Processor
  - **—Fully Static Version of the HCMOS MC68000/MC68008 Core**
  - **—25MHz at 5V, Future Version - 20MHz at 3.3 V**
- System Integration Block Including:
  - **—Fully Programmable PLL for System Clock Generation with Low Power Clock Output Drivers**
  - —Independent DMA (IDMA)
  - —Interrupt Controller with Two Modes of Operation
  - **—Up to 45 Parallel I/O Pins with Up to 8 Extra Input Pins**
  - —Three Timers Including a Watchdog Timer
  - —Four Programmable Chip-Select Lines
  - **—Glueless Interface to SRAM, EPROM, Flash EPROM, and EEPROM**
  - **—Programmable Interrupt Timer with Independent Clocking**
  - —Programmable Address Mapping of the Dual-Port RAM and IMP Registers

—System Control:

--Bus Arbitration Logic with Low-Interrupt Latency Support

--System Status and Control Logic

--Disable CPU Logic (M68000)

--Hardware Watchdog

--DRAM Refresh Controller

- Communication Processor (CP)

—RISC Controller

—Six Serial DMA (SDMA) Channels

—1152 Bytes of Dual Port RAM

—Serial Channels Physical Interface:

--Motorola Interchip Digital Link (IDL)

--General Circuit Interface (GCI)

--Pulse Code Modulation (PCM) Highway

--Non-Multiplexed Serial Interface (NMSI) Implementing Standard Modem Signals

—SCP for Synchronous Communication

—Two Serial Management Controllers (SMC's) to Support IDL and GCI Channels

—Three Serial Communication Channels (SCC's) Protocol Support Includes:

--HDLC

--UART

--UART 16550 Emulation

--BISYNC

--Totally Transparent Mode

—Autobaud Support Up to 115 Kbps with Multiple SCC's

—Internal Connection Option between SCC1 and DSP SCI with Separate Transmit and Receive Clocks

- PCMCIA Support

—PCMCIA 2.1 Compatible Slave Interface (8- or 16-bit Options)

—Support for Ring Detect and Other Indications

—Support for Power-Down Modes

—Support for Direct Access by PCMCIA Host to 68000 Bus for Fast Data Transfers

- 16550 Emulation Block

—Complete H/W and S/W Emulation of the 16550 UART

—DMA Support for the 68000 Side of the 16550 Emulation Controller

- Low Power Control
  - **—On Chip PLL Can Be Used with 32-KHz Crystal**
  - **—Ability to Change the System Clock Frequency "On the Fly" Using a Programmable 4-bit Clock Divider**
  - **—New Low-Power Modes**
  - **—Minimum Current Drain in μA Range**
  - **—One External Crystal Can Generate Clocks for both IMP and DSP**

Digital Signal Processor Features:

- DSP56K Central Processing Unit
  - —30 MIPS at 60MHz - 5V, Future Version at 22.5 MIPS (45 MHZ) at 3.3 V
  - —Single Instruction Cycle 24 x 24-Bit Parallel Multiply-Accumulator
  - —Highly Parallel Instruction Set with Unique Addressing Modes
  - —Zero-Overhead Nested Do Loops
  - —Fast Auto-Return Interrupts
  - —On-Chip Emulator (OnCE) for Unobtrusive, Full-Speed Debugging
  - —Fully Static Logic, Operating Frequency Down to DC
  - —Low-Power CMOS Design
  - —STOP and WAIT Low-Power Standby Modes
- Low Power Control
  - —On Chip PLL Can Be Used with 32-KHz Crystal
  - —Slow-Go Modes "On the Fly" using the 4-Bit Clock Divider Output
  - —Minimum Current Drain in μA Range
  - **—One External Crystal Can Generate Clocks for Both IMP and DSP**
- Expanded DSP Memory Spaces:
  - **—5.25K x 24 Program RAM**
  - —64 x 24 Program Bootstrap ROM
  - **—3K x 24 X-Data RAM**
  - **—2.5K x 24 Y-Data RAM**
  - —2 x 256 x 24 Data μ and A Law and Sine ROM
- Full Speed Memory Expansion Port with 16-Bit Address and 24-Bit Data Bus
- 8-bit Host Interface
  - **—Internally Connects to 68K Bus**
  - **—Configurable Interface Logic Selects between DMA Transfers and Interrupt-Driven Transfers**
  - **—Accessible by External 68000 Bus Masters**
- Synchronous Serial Interface (SSI) Port

- Serial Communication Interface (SCI+ Port)
  - **—Separate RX and TX Clocks**
  - **—Internal Connection Option — Internally Connects SCC1 of IMP to SCI of DSP**
  - **—DTE to Datapump Direct Mode — Direct Connection of SCC2 to DSP SCI for Support of Synchronous Modem Protocols**
- Direct Access from DSP to 68000 Bus
  - **—Allows the DSP to Read and Write Locations on the 68000 Bus**

1.2 MC68356 ARCHITECTURE OVERVIEW

The architecture of the MC68356 is shown in Figure 1-1. The top portion of Figure 1-1 is the integrated multiprotocol processor section (IMP), which is based on the MC68302. The lower half of the block diagram is the DSP section of the MC68356 and includes large memory blocks of DSP program and data RAM.

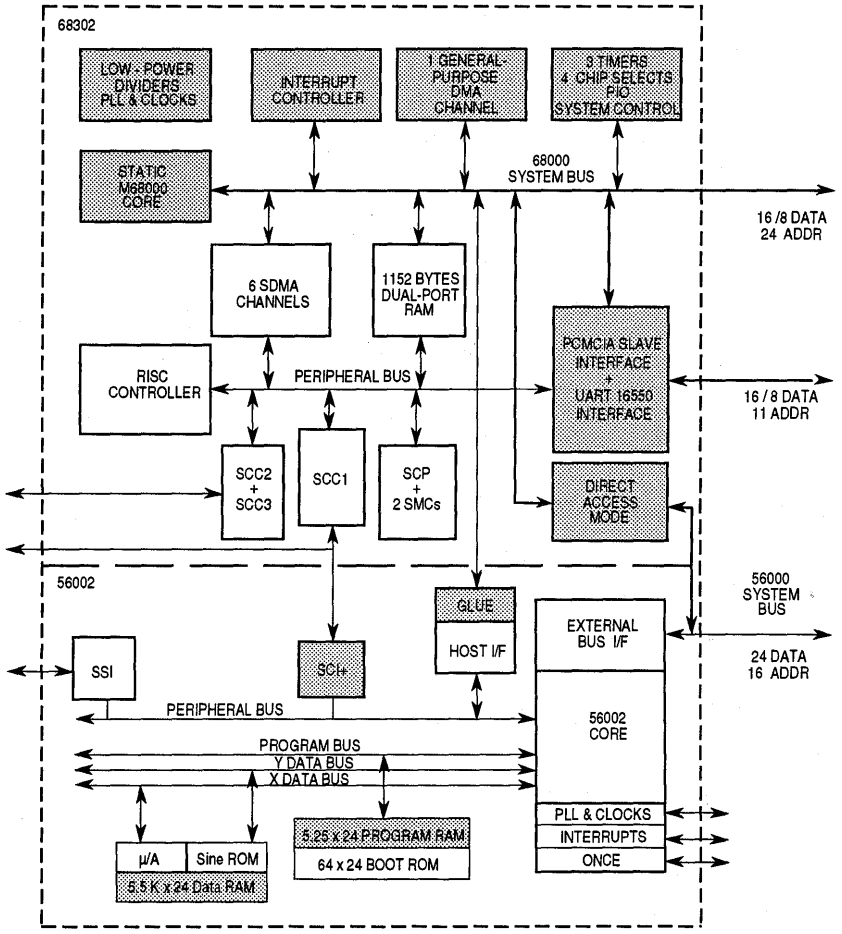


Figure 1-1. MC68356 Block Diagram

1.2.1 Integrated Multiprotocol Processor (IMP) Portion of the MC68356

The integrated multiprotocol processor (IMP) portion, which is based on the MC68302, consists of a communications processor section with three serial communications channels (SCC) that handle popular communications protocols such as HDLC, asynchronous (UART), BYSYNC, and synchronous protocols. Two serial management channels (SMC) and an interchip serial interface are also included. Flexible physical interfacing options ease connections to ISDN GCI and IDL interfaces as well as other muxed interfacing options. In addition, the system integration block includes an IDMA, six serial DMA channels, two general purpose timers, a watchdog timer, a periodic interrupt timer, an interrupt controller, four chip selects and general purpose I/O pins.

The original dynamic 68000 core used on the MC68302 has been replaced with a new static core that has been modified to improve power conservation and increase performance. The new core will operate at any oscillator clock frequency between 0 Hz and the maximum allowed clock frequency.

A PLL clock generation module has been added on the MC68356 to allow new power saving modes and oscillator/crystal options to optimize system power savings and oscillator parts count. The PLL, when enabled, can be driven by any oscillator or crystal frequency between 25 kHZ and 6 MHZ. The PLL output clock frequency is determined by the multiplication factor that is written to the PLL by software. The PLL block also includes a frequency divider on the output stage, which allows software to adjust the output system clock without forcing the PLL to lose lock. An option is also available to enable the IMP clock generation module to generate the DSP system clock, which eliminates the need for a separate DSP oscillator or crystal circuit.

The performance of the autobauding function has been improved to allow full use of two SCC's while autobauding at speeds up to 115.2 Kbps on a third SCC.

A PCMCIA block provides a glueless slave interface to a PCMCIA bus conforming to revision 2.1 of the PCMCIA specification. The PCMCIA interface can be configured as either an 8- or 16-bit slave interface. In a PCMCIA application, the initialization routine requires the PCMCIA card to provide the host with information about the configuration of the card. This information is called the card information structure (CIS). The MC68356 PCMCIA interface allows the CIS to be stored in 68K memory. The host can also directly access the 68000 bus by using the direct access mode logic which will allow the host to carry out burst mode accesses to 68000 bus memory locations. Once initialization is complete, application data can be transferred via the 16550 emulation or directly into the 68K memory.

The UART 16550 block allows the MC68356 to appear as a standard COM port to an x86 style PC. The 16550 emulation block supports the pins, the register set, timing, and interrupt structure of the original 16550 device. The 16550 emulation block may be used for PCMCIA applications or on a standard AT-card to interface to the AT bus.

1.2.2 DSP Portion of the MC68356

The DSP module of the MC68356 is based on a 60 MHz (at 5 V), 30 MIPS version of the DSP56002. The DSP56002 is a 24 bit fixed point DSP which includes a full Harvard archi-

tecture with separate internal X and Y data buses as well as a separate program bus. The DSP56002 features fast auto-return interrupts, zero-overhead nested Do loops, and an On-Chip Emulator (OnCE) for unobtrusive, full speed debugging. The internal memory spaces have been expanded to include 5.25K x 24 words of program RAM, and 5.5K x 24 words of X-Y data RAM. The expanded internal memory space of the DSP module offers these advantages:

1. Increased performance gained from being able to take advantage of the internal parallelism of the 56002. External DSP memory accesses are reduced or eliminated. The 56002's Harvard architecture allows it to simultaneously fetch program, X-data, and Y-data memory. This feature cannot be fully utilized for a given instruction when more than one type of external memory is used, because the DSP56002 has a single external system bus. The larger memory blocks improve overall performance by increasing the likelihood that the DSP56002 can perform simultaneous accesses to the three DSP memory spaces.
2. Reduced Memory Costs — The expanded internal memory spaces coupled with the host port boot mode and the DSP to IMP direct access mechanism allow most applications to store DSP programs in cheaper 68K memory which can be downloaded into the DSP56002 when needed. If all DSP programs can be run from the MC68356 internal memory system, the need for external DSP memory is eliminated. If external DSP memory is still required, the system can be configured such that the high speed DSP algorithms are stored internally, allowing the DSP to use slower RAM for its external memory requirements. The reduction or elimination of external DSP memory results in lower power and reduced system cost.

The host interface of the original DSP56002 is internally connected to the 68K bus. Internal interface logic is provided to allow the host port to be accessed via the IMP IDMA unit or via interrupt request logic to the 68000 core. In addition, the host port can be accessed by external 68000 bus masters.

The MC68356 also supports a 56002 to 68000 bus direct access mechanism to allow the 56000 core to access the 68000 bus via direct access logic to store and retrieve data at 68000 bus speeds. The mechanism is convenient for storing data in slower, cheaper 68000 memory space.

The addition of separate Rx and Tx clocking to the SCI port (now called the SCI+) and its connection to SCC1 provide the MC68356 with another link between the IMP and DSP. The modifications to the host interface and the SCI port improve communication between the communications processor and DSP modules of the MC68356, and eliminate the need for additional logic that would normally be required to interface a DSP to the serial interface of an embedded controller.

1.3 DESIGNING WITH MC68356

Since the MC68356 provides the functions of a higher performance MC68302 and a powerful DSP core in a single chip, many designers who currently use the MC68302 with a DSP in the same system may want to upgrade their current and future designs with the MC68356. The following paragraphs briefly discuss the issues that would be faced in such an effort.

1.3.1 Hardware Compatibility Issues

The MC68356 is not pin compatible with either the MC68302 or the DSP56002, since it is a combination of the two devices. Some of the pins in these devices are removed on the MC68356 as described in Section 2 Signal Description. New pins are added, and for the IMP, new pin multiplexing options have been added as well.

The MC68356 is available in the PBGA package (plastic ball grid array) which has more pins and a different physical interface than the QFP and PGA packages used by the MC68302 and DSP56002.

1.3.2 Glueless Interfaces

Three new pins ($\overline{OE}$, $\overline{WEH}$, $\overline{WEL}$) have been added to facilitate glueless interfacing to SRAM, EPROM, Flash EPROM, and EEPROM on the 68000 bus.

1.3.3 Software Compatibility Issues

The instruction sets of the IMP (68302) and DSP (56002) in the MC68356 are identical to their discrete counterparts.

Previous users of the MC68302 and DSP56002 will notice that very few features of these components have been removed during the integration. The main issue in upgrading a system operating with a separate DSP56002 and MC68302 is to modify the software to take full advantage of the new features of the MC68356.

1.4 APPLICATIONS FOR MC68356

Having a powerful DSP in combination with an integrated communication controller allows the MC68356 to be especially useful in communications applications. High-speed fax and modem applications are well-suited for this device. Other applications include medical instruments, PBXs, base stations, PDAs, and wireless communication systems.

1.4.1 General Purpose Application

Figure 1-2 shows a general purpose application that includes a T1 connection running HDLC, an asynchronous channel, and a PCMCIA or ISA bus interface to a host computer. On the DSP side, the diagram shows that multiple codecs or analog to digital converters can be connected in parallel to the DSP synchronous serial interface (SSI), as well as to the external DSP bus.

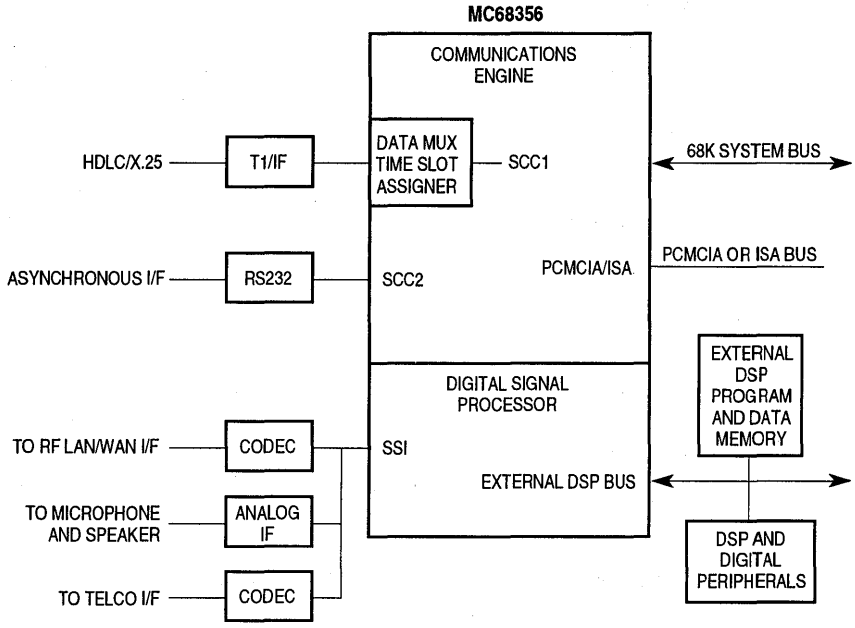


Figure 1-2. General Purpose Application

1.4.2 MC68356 Minimum Memory System Configuration

Figure 1-3 shows the minimum memory system configuration required for an MC68356 design. This system uses an EPROM (flash or regular) to store program code for the system, and SRAM for no wait state program and data memory. No external decode logic is needed for the memory interface because of the internal chip selects which include the $\overline{WEH}$, $\overline{WEL}$ and $\overline{OE}$ decodes. Since the DSP processor has 5.25K of program SRAM and 5.5K of data SRAM residing on-chip, all program and data code can be stored in the 68000 memories and downloaded via the internal host port connection to the DSP using the host port boot mode. Later transfers of program and data code in 68K memory can be initiated by the DSP using the DSP to 68000 direct access mechanism without requiring service from the 68000 core.

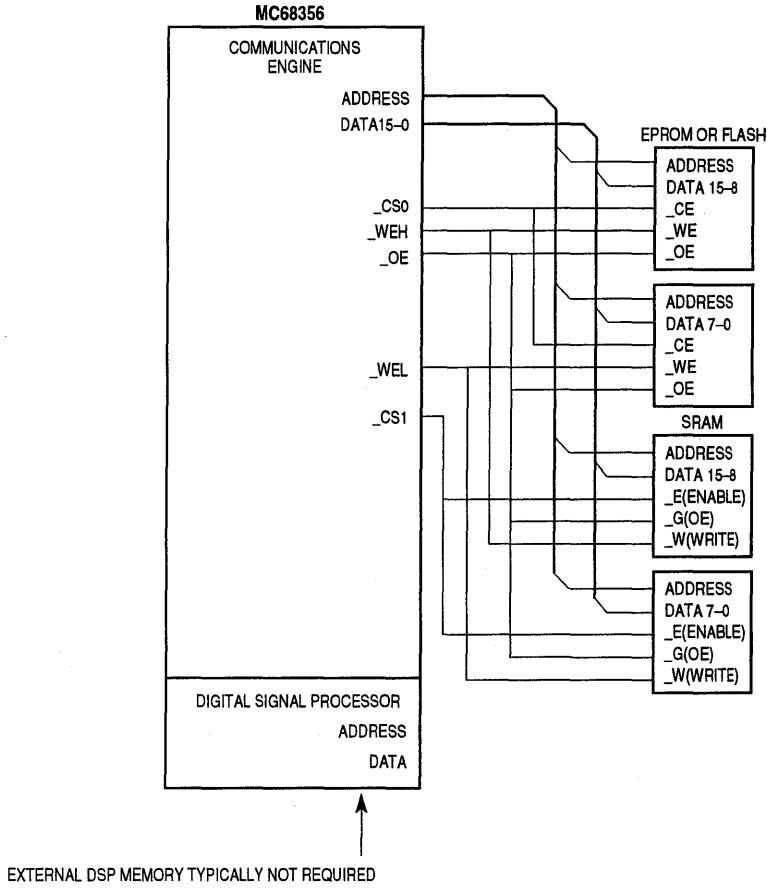
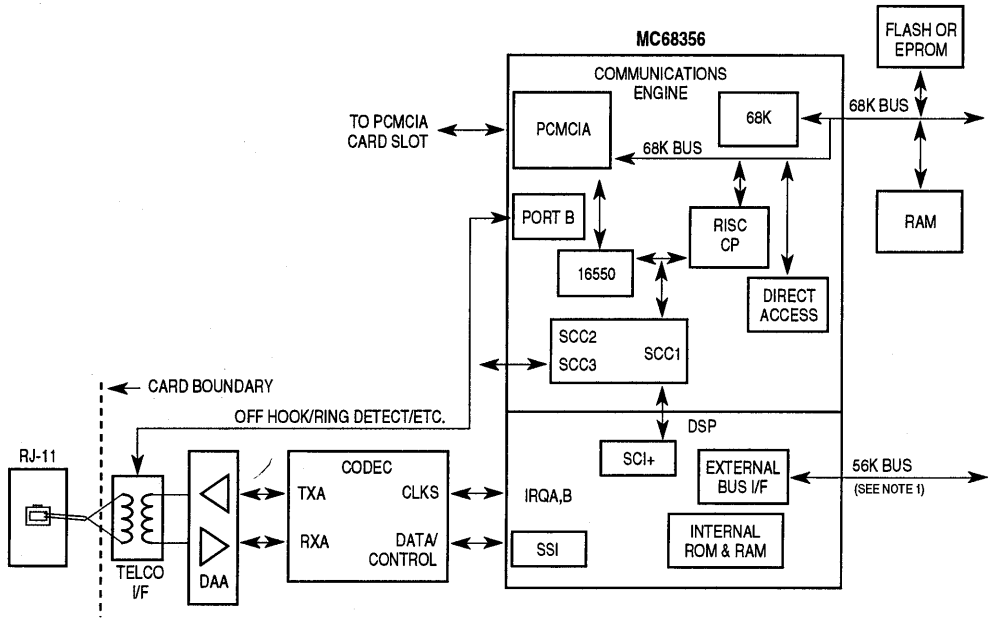


Figure 1-3. MC68356 Minimum Memory System Configuration

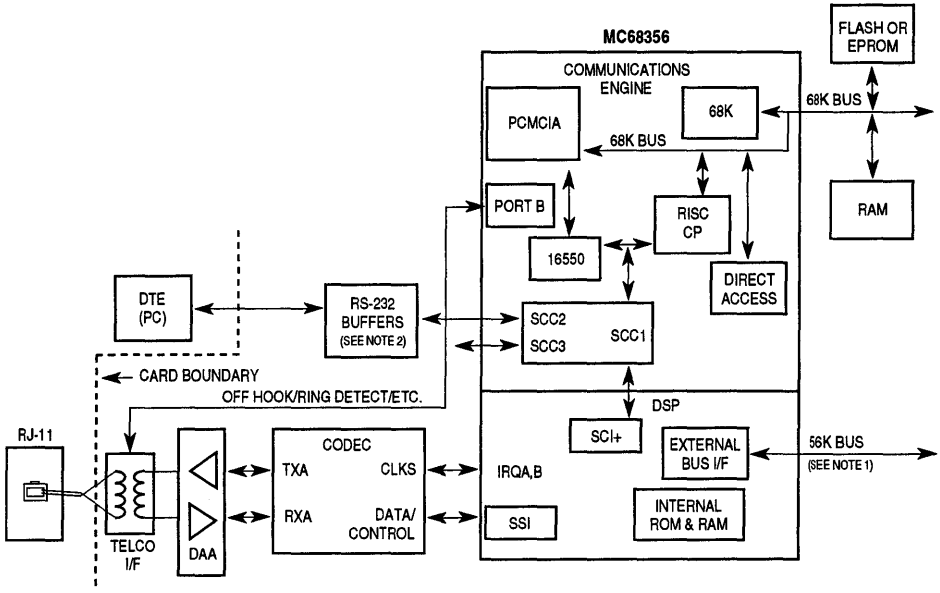
1.4.3 Modem Applications

The architecture of the MC68356 lends itself particularly well to high-speed modem applications. The MC68356 has been specifically tailored to meet the requirements of the new V.34 modem standard. Figure 1-4 shows a PCMCIA modem application using the MC68356. Figure 1-5 shows a typical "box modem" application which uses an RS232 interface to the DTE. Figure 1-6 shows a modem ISA bus card that takes advantage of the 16550 UART built into the MC68356. An upcoming version of the MC68356 will include a complete modem datapump in on-chip ROM so that no external DSP memory will be needed. Note that the need for a separate PCMCIA interface chip as well as a controller to DSP interface chip is eliminated.



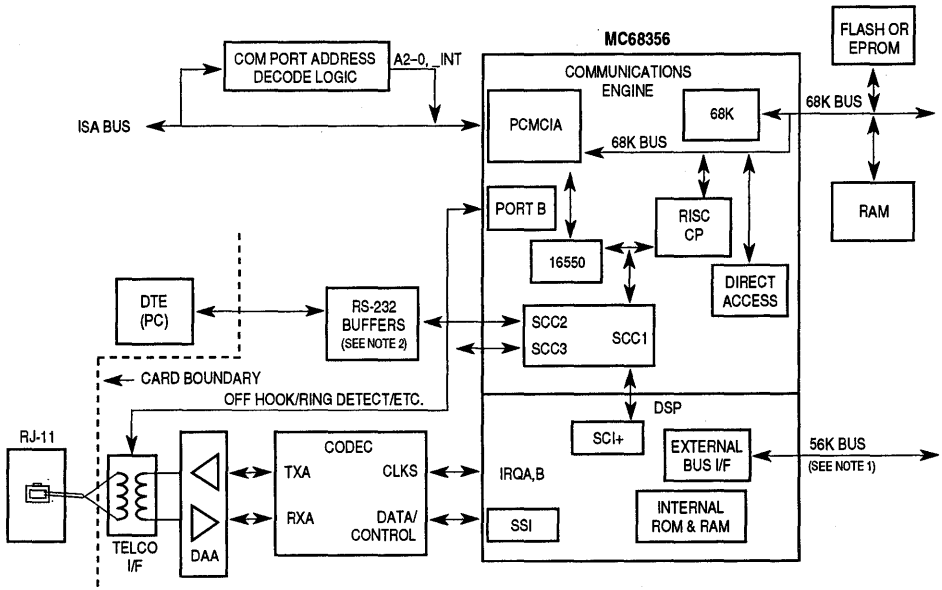
NOTE : NO EXTERNAL DSP MEMORY NEEDED

Figure 1-4. PCMCIA Modem Application



NOTES:
1. NO EXTERNAL DSP MEMORY NEEDED
2. STAND ALONE MODEMS

Figure 1-5. Serial RS-232 Modem Application



NOTES:
1. NO EXTERNAL DSP MEMORY NEEDED
2. STAND ALONE MODEMS

Figure 1-6. ISA Bus Modem Application

1.4.4 Fax/Modem Software Availability

To meet this application need, DSP data pump functions of a V.34 modem can be obtained directly from Motorola. The following industry standards and functions will be supported by the DSP datapump software:

- V.34
- V.33
- V.32bis with rate re-negotiation
- V.32
- V.29
- V.27ter
- V.23
- V.22bis
- V.22
- V.21
- V.17
- Bell 212A
- Bell 103
- Test Mode Support

Check with Motorola to obtain release schedules of the various components of this software. This software will be made available for purchase in a special mask ROM part number. Motorola will continue to develop and supply DSP software in MC68356 internal DSP ROM that will support future dial line modem standards.

SECTION 2
SIGNAL DESCRIPTION

This section defines the MC68356 pinout (Figure 2-1). The input and output signals of the MC68356 are organized into two main groups, the IMP pins and the DSP pins. Each group is then organized into functional groups and described in the following sections. For more detail on each signal, refer to the paragraph named for the signal.

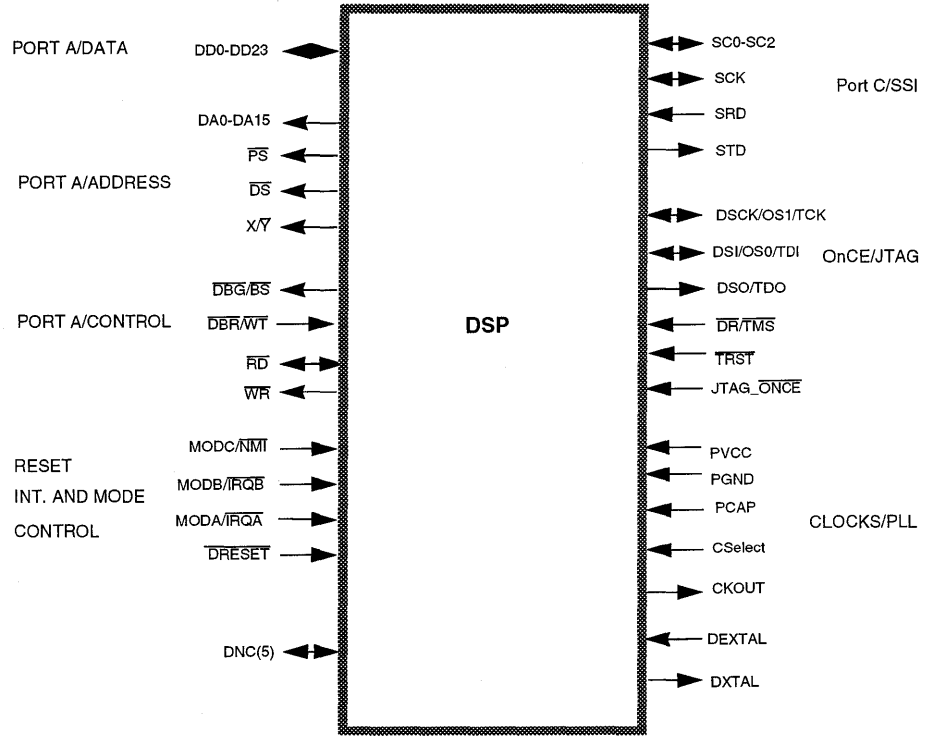


Figure 2-1. Functional Signal Groups

2

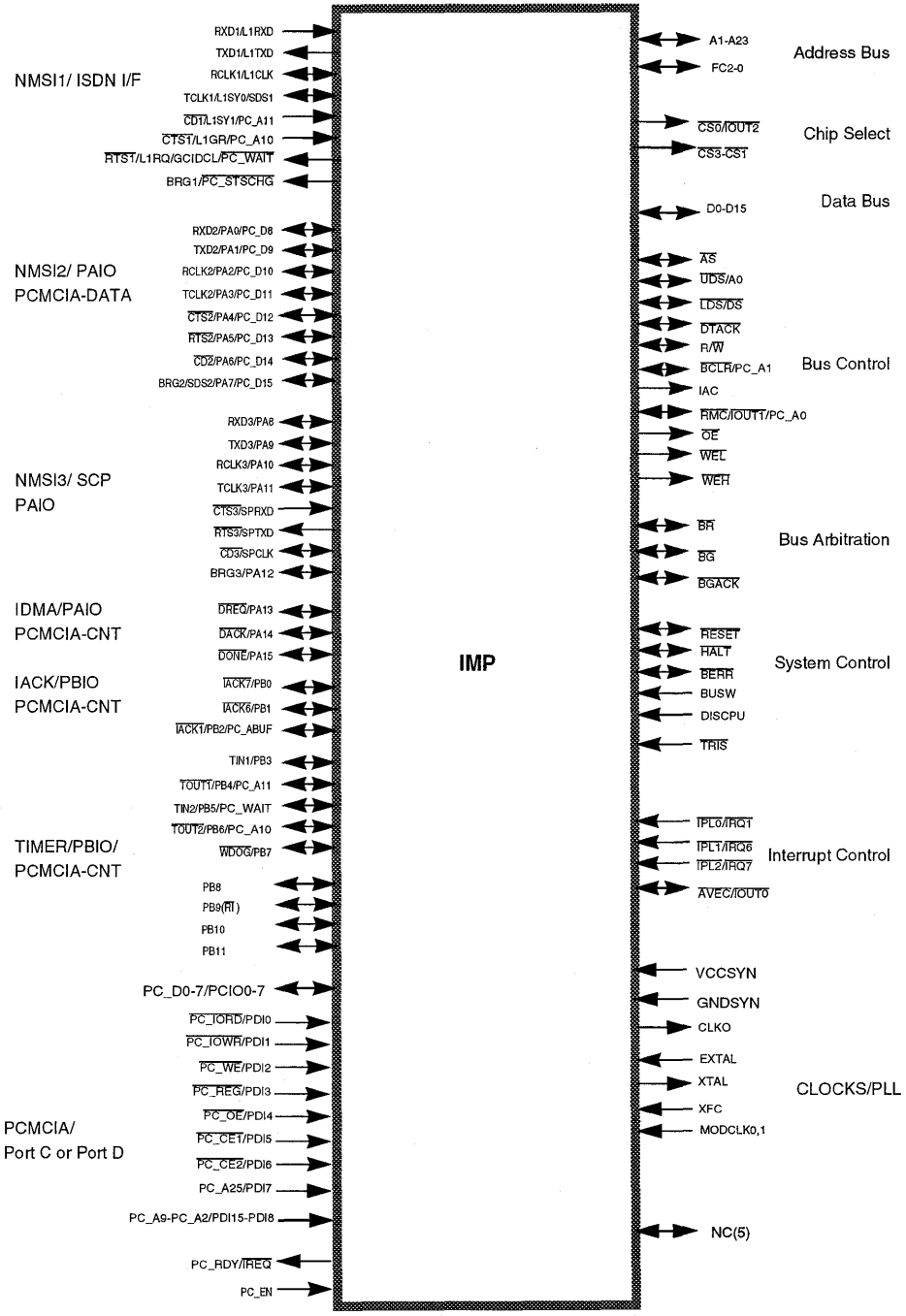


Figure 2-2. Functional Signal Groups

2.1 IMP PINS

Some of the pins allotted to the IMP portion of the MC68356 are multiplexed between the normal IMP signals and the PCMCIA signals.The pin multiplexing is organized so the user has all the corresponding IMP pins available when not using PCMCIA. In fact, when not using the PCMCIA interface, additional PIO ports are available. **The following paragraphs describe the IMP signals, assuming that the PCMCIA interface is *not* enabled.** The PCMCIA signals are described following those paragraphs. The PCMCIA interface is enabled by strapping the PC_EN pin high.

The IMP 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. 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 $\overline{\text{DTACK}}$.

The input and output signals of the IMP are organized into functional groups as shown in Table 2-1 and Table 2-2.

Table 2-1. IMP System Bus Pins

Group	Signal Name	Mnemonic	I/O	Section
Clock	Crystal Oscillator	EXTAL, XTAL	I O	2.3.6 IMP and DSP Clock and PLL Pins
	External Filter Capacitor	XFC	I	
	Clock Mode Select 1 and 2	MODCLK1, 2	I	
	Clock Out	CLKO	O	
System Control	Reset	$\overline{\text{RESET}}$	I/O	2.1.1 System Control
	Halt	$\overline{\text{HALT}}$	I/O	
	Bus Error	$\overline{\text{BERR}}$	I/O	
	Bus Width Select	BUSW	I	
	Disable CPU	DISCPU	I	
	Three-State	$\overline{\text{TRIS}}$	I	
Address and FC	Address Bus	A23-A1	I/O	2.1.2 IMP Address Bus Pins (A23–A1)
	Function Codes	FC2-FC0	I/O	2.1.6 Interrupt Control Pins
Data	Data Bus 15-0	D15-D0	I/O	2.1.3 IMP Data Bus Pins (D15—D0)

2

Table 2-1. IMP System Bus Pins

Group	Signal Name	Mnemonic	I/O	Section
Bus Control	Address Strobe	$\overline{AS}$	I/O	2.1.4 Bus Control Pins
	Read/Write	R/$\overline{W}$	I/O	
	Upper Data Strobe / Address 0	$\overline{UDS}$/A0	I/O	
	Lower Data Strobe / Data Strobe	$\overline{LDS}$/DS	I/O	
	Data Transfer Acknowledge	$\overline{DTACK}$	I/O	
	Read-Modify-Write Cycle/ Interrupt Output 1/ PCMCIA Address	$\overline{RMC}$ /$\overline{IOUT1}$/ PC_A0	I/O I	
	Internal Access	IAC	O	
	Bus Clear PCMCIA Address	$\overline{BCLR}$/ PC_A1	I/O I	
	Write Enable High	$\overline{WEH}$	O	
	Write Enable Low	$\overline{WEL}$	O	
	Output Enable	$\overline{OE}$	O	
Bus Arbitration	Bus Request	$\overline{BR}$	I/O	2.1.5 Bus Arbitration Pins
	Bus Grant	$\overline{BG}$	I/O	
	Bus Grant Acknowledge	$\overline{BGACK}$	I/O	
Interrupt Control	Interrupt Priority Level 0 / Interrupt Request 1	$\overline{IPL0}$/$\overline{IRQ1}$	I	2.1.6 Interrupt Control Pins
	Interrupt Priority Level 1 / Interrupt Request 6	$\overline{IPL1}$/$\overline{IRQ6}$	I	
	Interrupt Priority Level 2 / Interrupt Request 7	$\overline{IPL2}$/$\overline{IRQ7}$	I	
	Autovector / Interrupt Output 0	$\overline{AVEC}$/$\overline{IOUT0}$	I/O	
Chip Select	Chip Select 0 or Interrupt Output 2	$\overline{CS0}$/$\overline{IOUT2}$	O	2.1.17 Chip-Select Pins
	Chip Select 1-3	$\overline{CS1}$-$\overline{CS3}$	O	
Spare	Not Connected	NC1-5		2.1.18 No-Connect Pins
Power	Clock Synthesizer Ground	GNDSYN	I	2.3.6 IMP and DSP Clock and PLL Pins
	Clock Synthesizer Power	VCCSYN	I	
	System Power Supply and Return	VCC, GND	I	2.4 Power and Ground Pins

Table 2-2. IMP Peripheral Pins

Group	Signal Name	Mnemonic	I/O	Section
NMSI1 or ISDN	Receive Data / Layer 1 Receive data	RXD1/L1RXD	I	2.1.10 NMSI1 or ISDN Interface Pins
	Transmit Data / Layer 1 Transmit data	TXD1/L1TXD	O	
	Receive Clock / Layer 1 clock	RCLK1/L1CLK	I/O	
	Transmit Clock / PCM Sync / Serial Data Strobe 1	TCLK1/L1SY0/SDS1	I/O	
	Carrier Detect / Layer 1 Sync / PCMCIA Address	$\overline{CD1}$/L1SY1/ PC_A11	I	
	Clear To Send / Layer 1 Grant / PCMCIA Address	$\overline{CTS1}$/L1GR PC_A10	I	
	Request To Send / Layer 1 Request / GCI Clock Out / PCMCIA WAIT	$\overline{RTS1}$/L1RQ/GCIDCL/ PC_$\overline{WAIT}$	O	
	Baud Rate Generator 1 Output / PCMCIA Status Changed	BRG1/$\overline{PC\_STSCHG}$	O	
NMSI2 or PORT A or PCMCIA Data	Receive Data / Port A / PCMCIA Data	RXD2/PA0/ PC_D8	I/O	2.1.11 NMSI2 Port or Port A Pins
	Transmit Data / Port A / PCMCIA Data	TXD2/PA1/ PC_D9	I/O	
	Receive Clock / Port A / PCMCIA Data	RCLK2/PA2/ PC_D10	I/O	
	Transmit Clock / Port A / PCMCIA Data	TCLK2/PA3/ PC_D11	I/O	
	Clear To Send / Port A / PCMCIA Data	$\overline{CTS2}$/PA4/ PC_D12	I/O	
	Request To Send / Port A / PCMCIA Data	$\overline{RTS2}$/PA5/ PC_D13	I/O	
	Carrier Detect / Port A / PCMCIA Data	$\overline{CD2}$/PA6/ PC_D14	I/O	
	Baud Rate Generator 2 Output / Port A / PCMCIA Data	BRG2/PA7/ PC_D15	I/O	
NMSI3 or PORT A or SCP Pins	Receive Data / Port A	RXD3/PA8	I/O	2.1.12 NMSI3 Port or Port A Pins or SCP Pins
	Transmit Data / Port A	TXD3/PA9	I/O	
	Receive Clock / Port A	RCLK3/PA10	I/O	
	Transmit Clock / Port A	TCLK3/PA11	I/O	
	SCP Receive Serial Data / Clear To Send	SPRXD/$\overline{CTS3}$	I	
	SCP Transmit Serial Data / Request To Send	SPTXD/$\overline{RTS3}$	O	
	SCP Clock / Carrier Detect	SPCLK/$\overline{CD3}$	I/O	
	Baud Rate Generator 3 Output / Port A	BRG3/PA12/	I/O	
IDMA Pins or PORT A	DMA Request / Port A	$\overline{DREQ}$/PA13	I/O	2.1.13 IDMA or Port A Pins
	DMA Acknowledge / Port A	$\overline{DACK}$/PA14	I/O	
	DMA Done / Port A	$\overline{DONE}$/PA15	I/O	
IACK or PORT B or PCMCIA control pin	Interrupt Acknowledge / Port B	$\overline{IACK7}$/PB0	I/O	2.1.14 IACK or PIO Port B Pins
	Interrupt Acknowledge / Port B	$\overline{IACK6}$/PB1	I/O	
	Interrupt Acknowledge / Port B / PCMCIA Address Buffer Enable	$\overline{IACK1}$/PB2/ PC_ABUF	I/O	

2

Table 2-2. IMP Peripheral Pins

Group	Signal Name	Mnemonic	I/O	Section
TIMERs pins or PCMCIA control pins (PCMCIA pins are active only when ISDN interface is needed)	Timer 1 Input / Port B	TIN1/PB3	I/O	2.1.15 Timer Pins
	Timer 1 Output / Port B / PCMCIA Address	$\overline{\text{TOUT1}}$/PB4/ PC_A11	I/O	
	Timer 2 Input / Port B / PCMCIA WAIT	TIN2/PB5/ PC_WAIT	I/O	
	Timer 2 Output / Port B / PCMCIA Address	$\overline{\text{TOUT2}}$/PB6/ PC_A10	I/O	
	Watchdog / Port B	$\overline{\text{WDOG}}$/PB7	I/O	
Port B I/O	Port B	PB11-8	I/O	2.1.16 Parallel I/O Pins with Interrupt Capability
PCMCIA or PORT C OR PORT D	Port C PCMCIA Low Data Bus	PCIO0-7 PC_D0-7	I/O	2.2 PCMCIA Pins
	PCMCIA Enable	PC_EN	I	
	PCMCIA Ready_Busy / Interrupt Request	PC_RDY_BSY/$\overline{\text{IREQ}}$	O	
	Port D / PCMCIA IO Read	PDI0/$\overline{\text{PC\_IORD}}$	I	
	Port D / PCMCIA IO Write	PDI1/$\overline{\text{PC\_IOWR}}$	I	
	Port D / PCMCIA Write Enable	PDI2/$\overline{\text{PC\_WE}}$	I	
	Port D / PCMCIA REG	PDI3/$\overline{\text{PC\_REG}}$	I	
	Port D / PCMCIA Output Enable	PDI4/$\overline{\text{PC\_OE}}$	I	
	Port D / PCMCIA Card Enable 1	PDI5/$\overline{\text{PC\_CE1}}$	I	
	Port D / PCMCIA Card Enable 2	PDI6/$\overline{\text{PC\_CE2}}$	I	
	Port D / PCMCIA Address	PDI7/PC_A25	I	
	Port D / PCMCIA Address	PDI8-15/PC_A2-9	I	

Signals in bold italics are enabled when the PCMCIA pins are enabled.

All pins except EXTAL and CLKO support TTL levels. EXTAL, when used as an input clock, requires CMOS levels. CLKO supplies a CMOS level output.

All outputs (except CLKO) can drive up to 100 pF. CLKO is designed to drive up to 50 pF.

2.1.1 System Control

The system control pins are shown in Figure 2-2.

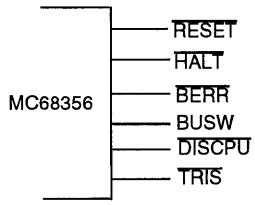


Figure 2-2. System Control Pins

RESET — 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 IMP. 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 Section 5 Memory Map 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 a total system reset, the address, data, and bus control pins are all three-stated, except for CS3–CS0, WEH, WEL, and OE, 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 is 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 IMP peripherals, the RESET instruction may be used.

Besides the total system reset and the RESET instruction, some of the IMP 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).

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 RESET signal to cause a total IMP system reset. If BERR is asserted with the HALT signal, a retry cycle is performed.

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 DTACK), by the chip-select logic (address conflict or write-protect violation), or by external circuitry. If BERR is asserted with the 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 IMP 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 on 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 6.7.4 Disable CPU Logic (M68000) for more details.

$\overline{\text{TRIS}}$ - TRI State

This input is sampled during total system reset ($\overline{\text{RESET}}$ and $\overline{\text{HALT}}$ asserted), and when asserted, three-states all of the MC68356 pins.

2.1.2 IMP Address Bus Pins (A23–A1)

The address bus pins are shown in Figure 2-3.

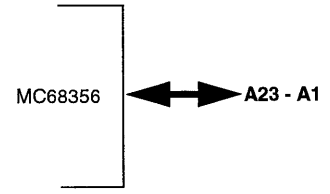


Figure 2-3. Address Bus Pins

A23—A1 form a 24-bit address bus when combined with $\overline{\text{UDS}}$/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.

2.1.3 IMP Data Bus Pins (D15—D0)

The data bus pins are shown in Figure 2-4.

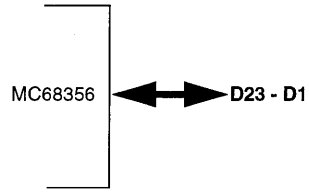


Figure 2-4. Data Bus Pins

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.

2.1.4 Bus Control Pins

The bus control pins are shown in Figure 2-5.

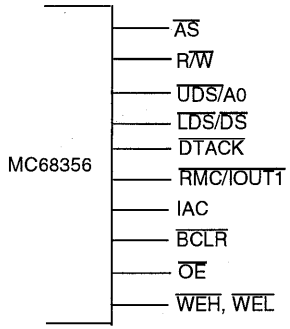


Figure 2-5. Bus Control Pins

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/$\overline{W}$—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.

$\overline{UDS}$/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 ($\overline{UDS}$). 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.

$\overline{LDS}$/$\overline{DS}$—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 ($\overline{LDS}$). When using an 8-bit data bus, this pin functions as $\overline{DS}$. This line is an output when the IMP (M68000 core, SDMA or IDMA) is the bus master and is an input otherwise.

$\overline{DTACK}$—Data Transfer Acknowledge

This bidirectional signal indicates that the data transfer has been completed. $\overline{DTACK}$ 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 registers. If $\overline{DTACK}$ 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.

$\overline{RMC}$/$\overline{IOUT1}$—Read-Modify-Write Cycle Indication/Interrupt Output 1

This signal functions as $\overline{RMC}$ in normal operation. $\overline{RMC}$ 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 $\overline{IOUT1}$. $\overline{IOUT2}$–$\overline{IOUT0}$ provide the interrupt request output signals from the IMP interrupt controller to an external CPU when the M68000 core is disabled.

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 $\overline{CS3}$–$\overline{CS0}$ pins.

IAC can be used to disable an external address/data buffer when the on-chip dual-port RAM and registers are accessed, to prevent 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 2-6. The IAC signal saves the propagation delay and logic required to OR all the system chip-select lines together to determine when to enable the external buffers.

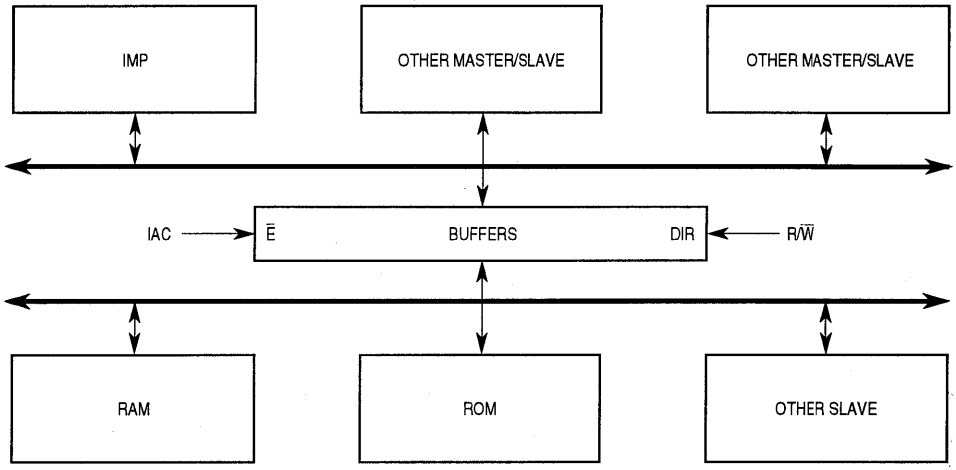


Figure 2-6. External Address/Data Buffer

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.

WEH—Write Enable High

This output is the write enable for high byte accesses to external memory or peripherals (see Section 14 Electrical Characteristics).

WEL—Write Enable Low

This output is the write enable for low byte accesses to external memory or peripherals (see Section 14 Electrical Characteristics).

$\overline{OE}$—Output Enable

This output is active during read cycle and indicates that an external device should place valid data on the bus.

2.1.5 Bus Arbitration Pins

The bus arbitration pins are shown in Figure 2-7.

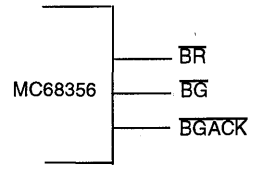


Figure 2-7. Bus Arbitration Pins

$\overline{BR}$—Bus Request

This input signal indicates to the on-chip bus arbiter that an external device desires to become the bus master. See 6.7.5 Bus Arbitration Logic for details. This signal is an open-drain output request signal from the IDMA and SDMA when the internal M68000 core is disabled.

$\overline{BG}$—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, $\overline{BG}$ = $\overline{BR}$.

$\overline{BGACK}$—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 $\overline{BGACK}$ pin will remain asserted continuously.

NOTE

$\overline{BGACK}$ should always be used in the external bus arbitration process. See 6.7.5 Bus Arbitration Logic for more details.

2.1.6 Interrupt Control Pins

The interrupt control pins are shown in Figure 2-8.

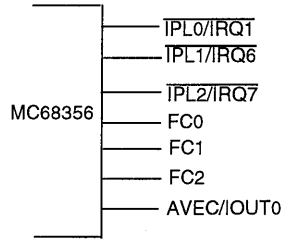


Figure 2-8. Interrupt Control Pins

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 IMP 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 automatic

vectoring only. $\overline{AVEC}$ instead of $\overline{DTACK}$ should be asserted during autovectoring and should be high otherwise.

When the M68000 core is disabled, this pin operates as $\overline{IOUT0}$. $\overline{IOUT2}$–$\overline{IOUT0}$ provide the interrupt request output signals from the IMP interrupt controller to an external CPU when the M68000 core is disabled.

2.1.7 IMP Bus Interface Signal Summary

Table 2-3 and Table 2-4 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 $\overline{BG}$ change their direction and $\overline{BCLR}$ becomes bidirectional.

Table 2-3. Bus Signal Summary—Core and External Master

Signal Name	Pin Type	M68000 Core Master Access To		External Master Access To	
		Internal Memory Space	External Memory Space	Internal Memory Space	External Memory Space
A23–A1, FC2–FC0, $\overline{AS}$, $\overline{UDS}$, $\overline{LDS}$, R/$\overline{W}$, $\overline{RMC}$	I/O	O	O	I	I
$\overline{BCLR}$	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
$\overline{DTACK}$	I/O	O	**	O	**
$\overline{BR}$	I/O Open Drain	I	I	I	I
$\overline{BG}$	I/O	O	O	O	O
$\overline{BGACK}$	I/O	I	I	I	I
$\overline{HALT}$	I/O Open Drain	I/O	I/O	I	I
$\overline{RESET}$	I/O Open Drain	I/O	I/O	I	I
$\overline{BERR}$	I/O Open Drain	I/O***	I/O***	I/O***	I/O***
$\overline{IPL2}$–$\overline{IPL0}$	I	I	I	I	I
$\overline{AVEC}$	I	I	I	I	I
$\overline{IOUT2}$–$\overline{IOUT0}$	O	O	O	O	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.

Table 2-4. Bus Signal Summary—IDMA and SDMA

Signal Name	Pin Type	IDMA Master Access To		SDMA Master Access To		PCMCIA or DSP to IMP Master Access To	
		Internal Memory Space	External Memory Space	Internal Memory Space	External Memory Space	Internal Memory Space	External Memory Space
A23–A1, FC–FC0, AS, UDS, LDS, R/W, RMC	I/O	O	O	N/A	O	O	O
BCLR	I/O Open Drain	I #	I #	N/A	O	I #	I #
IAC	O	O	O	N/A	O	O	O
D15—D0 Read	I/O	O	I	N/A	I	O	I
D15—D0 Write	I/O	O	O	N/A	O	O	O
DTACK	I/O	O	**	N/A	**	O	**
BR	I/O	O ##	O ##	N/A	O ##	O ##	O ##
BG	I/O	I ##	I ##	N/A	I ##	I ##	I ##
BGACK	I/O	O	O	N/A	O	O	O
HALT	I/O Open Drain	I	I	N/A	I	I	I
RESET	I/O Open Drain	I	I	N/A	I	I	I
BERR	I/O Open Drain	I/O***	I/O***	N/A	I/O***	I/O***	I/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.
Applies to disable CPU mode only. The internal signal IBCLR is used otherwise.
Applies to disable CPU mode only, otherwise N/A.

2.1.8 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 2-5 shows the functionality of each group of pins and their internal connection to the three SCC and one SCP controller. 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 2-5. 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.

2.1.9 Typical Serial Interface Pin Configurations

Table 2-6 shows typical configurations of the physical layer interface pins for an ISDN environment. Table 2-7 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 2-6. 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.

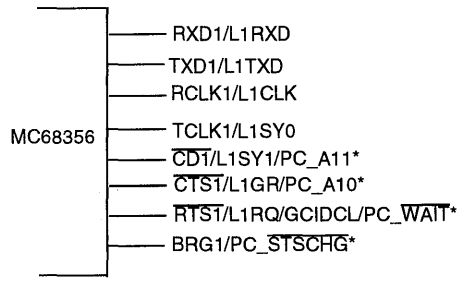
Table 2-7. Typical Generic Configurations

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.

2.1.10 NMSI1 or ISDN Interface Pins

The NMSI1 or ISDN interface pins are shown in Figure 2-9.



NOTE: * If IPINS = 0

Figure 2-9. 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 2-8 shows the functionality of each pin in NMSI, GCI, IDL, and PCM highway modes.

Table 2-8. 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
$\overline{CD1}$/L1SY1	I	$\overline{CD1}$	I	L1SYNC	I	L1SYNC	I	L1SY1
$\overline{CTS1}$/L1GR	I	$\overline{CTS1}$	I	L1GR	I	L1GR		
$\overline{RTS1}$/L1RQ	O	$\overline{RTS1}$	O	GCIDCL	O	L1RQ	O	$\overline{RTS}$
BRG1	O	BRG1	O	BRG1	O	BRG1	O	BRG1

NOTES:
1. In IDL and GCI mode, SDS2 is output on the PA7 pin.
2. $\overline{CD1}$ may be used as an external sync in NMSI mode.
3. $\overline{RTS}$ is the $\overline{RTS1}$, $\overline{RTS2}$, or $\overline{RTS3}$ 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 14 in the DISC register (see 7.5.3 DSP Interconnection and Serial Connections Register–DISC).

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 15 in the DISC register (see 7.5.3 DSP Interconnection and Serial Connections Register–DISC).

2

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 2-9. 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. $\overline{\text{CD1}}$ may also be used as an external sync in NMSI mode. $\overline{\text{CD1}}$ will be grounded internally when this pin is used for an alternate function.

$\overline{\text{CTS1}}$/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_{CC}.

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.

$\overline{\text{RTS1}}$/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 7.5.1 SCC Features). When BRG1 is disabled the pin is driven high.

2.1.11 NMSI2 Port or Port A Pins

The NMSI2 port or port A pins are shown in Figure 2-10. The PCMCIA function of these pins is shown in italics. Refer to 2.2 PCMCIA Pins for PCMCIA pin description.

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.

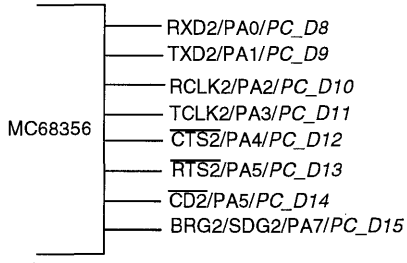


Figure 2-10. NMSI2 Port or Port A Pins

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 SI-MODE 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

2

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 2-10. 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.

2.1.12 NMSI3 Port or Port A Pins or SCP Pins

The NMSI3 port or port A pins or SCP pins are shown in Figure 2-11.

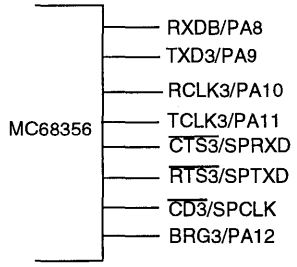


Figure 2-11. NMSI3 Port or Port A Pins or SCP Pins

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—Port A Bit 12/SCC3 Baud Rate Generator

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.

2.1.13 IDMA or Port A Pins

The IDMA or port A pins are shown in Figure 2-12.

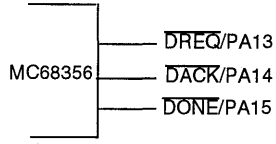


Figure 2-12. IDMA or Port A Pins

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

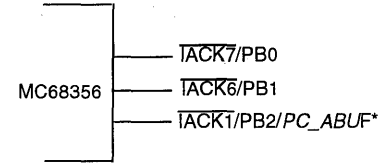
This output, asserted by the IDMA, signals to the peripheral that an operand is being transferred in response to a previous transfer request.

$\overline{\text{DONE}}$/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.

2.1.14 IACK or PIO Port B Pins

The IACK or PIO port B pins are shown in Figure 2-13.



NOTE: * If *ABUF* = 1

Figure 2-13. 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

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.

IACK7/PB0, IACK6/PB1, IACK1/PB2—Interrupt Acknowledge/Port B I/O

As IACK1, IACK6, and IACK7, these active low output signals indicate to the external device that the IMP 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).

2.1.15 Timer Pins

The timer pins are shown in Figure 2-14.

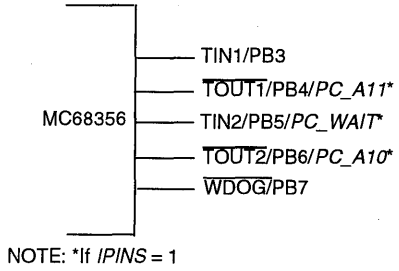


Figure 2-14. Timer Pins

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.

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.

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.

WDOG/PB7—Watchdog Output

This active-low, open-drain output indicates expiration of the watchdog timer. WDOG may be externally connected to the RESET and HALT pins to reset the IMP. When WDOG is asserted, it will remain asserted for 16 IMP system clock cycles (if the IMP PLL is not enabled by the MODCLK pins), or 16 clock cycles after the PLL locks (if the PLL is enabled by the MODCLK pins). WDOG is never asserted by the on-chip hardware watchdog (see the BERR signal description). The WDOG pin function is enabled after a total system reset. It may be reassigned as the PB7 I/O pin in the PBCNT register.

2.1.16 Parallel I/O Pins with Interrupt Capability

The four parallel I/O pins with interrupt are shown in Figure 2-15.

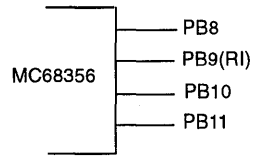


Figure 2-15. Port B Parallel I/O Pins with Interrupt

PB11 and 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.

PB9(RI)

This pin may be configured as a general-purpose parallel I/O port with interrupt capability and wake-up capability.This pin can be configured either as an input or an output. When configured as an input, this pin can generate a separate, maskable interrupt on a high-to-low transition.

This pin can also wake-up the IMP from low power STOP, DOZE, or STAND-BY mode when a high to low transition occurs on this pin when configured as an input. See 3.5.2.3 IMP Wake-Up from Low Power STOP Modes for more information.

In addition, when the MC68356 is configured for the PCMCIA interface, this pin can connected internally to the PCMCIA I/O Event Indication Register or directly to the PC_$\overline{\text{STSCHG}}$ pin. For more details on this feature refer to 8.2.1 PCMCIA Ring Indication.

This pin's input buffer has a Schmitt trigger.

PB10

This pin may be configured as a general-purpose parallel I/O port with interrupt capability and wake-up capability.This pin can be configured either as an input or an output. When configured as an input, this pin can generate a separate, maskable interrupt on a high-to-low transition.

This pin can also wake-up the IMP from low power STOP, DOZE, or STAND-BY mode when a high to low transition occurs on this pin when configured as an input. See 3.5.2.3 IMP Wake-Up from Low Power STOP Modes for more information.

This pin's input buffer has a Schmitt trigger.

2.1.17 Chip-Select Pins

The chip-select pins are shown in Figure 2-16.

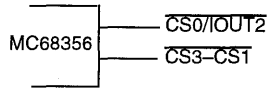


Figure 2-16. Chip-Select Pins

CS0/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 IOUT2. IOUT2—IOUT0 provide the interrupt request output signals from the IMP interrupt controller to an external CPU when the M68000 core is disabled.

CS3–CS1—Chip Selects 3–1

These three active-low output pins function as chip selects for external devices or memory. CS3—CS0 do not activate on accesses to the internal RAM or registers (including the BAR SCR, or CKCR registers).

2.1.18 No-Connect Pins

NC1–NC5 and DNC1–2 are reserved for future use and should not be connected.

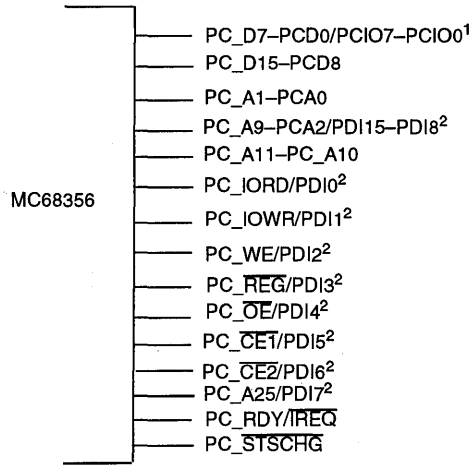
2.2 PCMCIA PINS

The following pins are valid when the PC_EN pin is connected to V_{CC}. These pins are muxed and therefore replace other pins described in the previous paragraphs. The replaced signal is indicated in each description.

When using the PCMCIA pins, the $\overline{\text{BCLR}}$ and $\overline{\text{RMC}}$ pins will not be available externally to the chip. When ISDN functionality is not needed, the $\overline{\text{CD1}}$, $\overline{\text{CTS1}}$ and $\overline{\text{RTS1}}$ will be used for the PCMCIA interface signals and will not be available externally. When ISDN functionality is needed, the $\overline{\text{TOUT1}}$, TIN2 and $\overline{\text{TOUT2}}$ will not be available externally and will be used for the PCMCIA control pins. ISDN pin functionality with PCMCIA pins can be selected by setting the IPINS bit in the PCMR (See Section 8 PCMCIA Controller).

NOTE

The PCMCIA pins are denoted in the pinout diagrams as PC_*SIGNALNAME*.



NOTES:
1.Extra parallel I/O when not in PCMCIA mode.
2.Extra parallel inputs when not in PCMCIA mode.

Figure 2-17. PCMCIA Signals

PC_D15–PC_D0—PCMCIA Data Bus

The bidirectional PCMCIA card interface data bus. When the MC68356 is not enabled for PCMCIA, these pins can be used for Port C general purpose input/output pins.

Non-PCMCIA functionality: PCIO0-7 for PC_D0-7 and NMSI2 for PC_D8-15.

PC_A0–PC_A10—PCMCIA Address Bus

Address bus input lines from the PCMCIA card interface. These lines are used as the lower address lines when the cycle is transferred into the 68000 bus. The high address lines and FC are taken from the base registers (See Section 8 PCMCIA Controller).

When the MC68356 is not enabled for PCMCIA, the PC_A2-9 pins can be used for port D general purpose input only pins.

Non-PCMCIA functionality: RMC/IOUT for PC_A0, BCLR for PC_A1, PDI15-8 for PC_A2-9, $\overline{\text{CTS}}$/L1GR for PC_A10, $\overline{\text{CD1}}$/L1SY1 for PC_A11. With IPINS set: $\overline{\text{TOUT2}}$/PB6 for PC_A10 and $\overline{\text{TOUT1}}$/PB4 for PC_A11.

The next several pins described can be used for port D general purpose input only pins when the PCMCIA interface is not enabled (See non-PCMCIA functionality in each description).

PC_A25—PCMCIA Address Bus

This PCMCIA card interface input address line is used to identify 68000 bus PCMCIA cycles. A25 is low during 68000 bus PCMCIA cycles.

Non-PCMCIA functionality: PDI7.

PC_$\overline{CE1}$ & PC_$\overline{CE2}$—PCMCIA Card Enable

Active low, card enable PCMCIA card interface input signals.

Non-PCMCIA functionality: PDI5 for PC_CE1, PDI6 for PC_CE2.

PC_$\overline{OE}$—PCMCIA Output Enable

The PC_$\overline{OE}$ line is the active low, input signal used to gate memory read data from the memory card.

Non-PCMCIA functionality: PDI4

PC_$\overline{WE}$—PCMCIA Write Enable/Program

The $\overline{WE/PGM}$ input used for strobing memory write data into the memory card.

Non-PCMCIA functionality: PDI2

PC_$\overline{REG}$—PCMCIA Attribute Memory Select

This input is kept inactive (high) for all common memory accesses. When this signal is active, access is limited to attribute memory ($\overline{OE}$ or $\overline{WE}$ active) and to the I/O space ($\overline{IORD}$ or $\overline{IOWR}$ active).

Non-PCMCIA functionality: PDI3

PC_$\overline{IORD}$—PCMCIA I/O Read

The $\overline{IORD}$ signal is made active to read data from the card's I/O space. The $\overline{REG}$ signal and at least one of $\overline{CE1}$ or $\overline{CE2}$ must also be active. A PC card will not respond to the $\overline{IORD}$ signal until it has been configured for I/O operation by the system.

Non-PCMCIA functionality: PDI0

PC_$\overline{IOWR}$—PCMCIA I/O Write

The $\overline{IOWR}$ signal is made active to write data to the card's I/O space. The $\overline{REG}$ signal and at least one of $\overline{CE1}$ or $\overline{CE2}$ must also be active. A PC card will not respond to the $\overline{IORD}$ signal until it has been configured for I/O operation by the system.

Non-PCMCIA functionality: PDI1

PC_$\overline{\text{WAIT}}$—PCMCIA Extend Bus Cycle

The $\overline{\text{WAIT}}$ signal is asserted by the MC68356 to delay the memory space accesses or I/O space accesses.

Non-PCMCIA functionality: $\overline{\text{RTS}}$/L1RQ/GCIDCL or TIN2/PB5 in ISDN mode (IPINS bit set in PCMR register).

PC_$\overline{\text{STSCHG}}$—PCMCIA Status Changed [replace BVD1]

Status changed is an optional signal used to alert the system to changes in the ready/busy (RDY/$\overline{\text{BSY}}$), write protect (WP), or battery voltage (BV) conditions of the card while the I/O interface is configured.

This signal is inactive (high) when this function is not supported by the card or when the SigChg bit in the card status register is false (logic 0). When the SigChg bit is true (logic 1) this signal is active when the changed bit in the card status register is true (logic 1). The changed bit is logical OR of the individual changed bits - CBVD1, CBVD2, CWP, and CRDB-SY - in the pin replacement register.

Non-PCMCIA functionality: BRG1.

PC_RDY/$\overline{\text{IREQ}}$—PCMCIA Ready/Busy/Interrupt Request

The Ready/Busy (RDY/$\overline{\text{BSY}}$) function is provided by this signal only when the card and the host are configured for the memory-only interface. When a host socket and the card inserted into it are both configured for the I/O interface, this function is provided by the RDY/$\overline{\text{BSY}}$ status bit in the card's pin replacement register.

This signal is available as Interrupt Request ($\overline{\text{IREQ}}$) only when the card and the interface are configured for the I/O and memory interface. The interrupt request is level only. This pin serves as RDY/$\overline{\text{BSY}}$ in memory-only cards, and as $\overline{\text{IREQ}}$ for I/O cards. This pin is used as RDY/$\overline{\text{BSY}}$ while an I/O capable card is configured for the memory-only interface.

Non-PCMCIA functionality: None (No connect)

2.3 DSP PINS

This section introduces pins associated with the DSP. It divides the pins into their functional groups and explains the role each pin plays in the operation of the chip. It acts as a reference for following chapters which explain the chip's peripherals in detail.

Table 2-11. DSP System Bus Pins

Group	Signal Name	Mnemonic	I/O	Section
Clock	Crystal Oscillator	DEXTAL, DXTAL	IO	2.3.6 IMP and DSP Clock and PLL Pins
	External Filter Capacitor	PCAP	I	
	Clock Mode Select	CSelect	I	
	Clock Out	CKOUT	O	

Table 2-11. DSP System Bus Pins

Group	Signal Name	Mnemonic	I/O	Section
Port A Address	Address Bus	DA15-DA0	I/O	2.3.1 Port A Address and Data Bus
Port A Data	Data Bus	DD23-DD0	I/O	2.3.1 Port A Address and Data Bus
Port A Bus Control	Program Memory Select	$\overline{PS}$	O	2.3.1 Port A Address and Data Bus
	Data Memory Select	$\overline{DS}$	O	
	X/$\overline{Y}$ Select	X/$\overline{Y}$	O	
	Read Enable	$\overline{RD}$	O	
	Write Enable	$\overline{WR}$	O	
	DSP Bus Strobe/DSP Bus Grant	$\overline{BS}$/DBG	O	
	Bus Wait/DSP Bus Request	$\overline{WT}$/$\overline{DBR}$	O	
Interrupt and Mode Control	Mode Select A/External Interrupt Request A/STOP Recovery	MODA/$\overline{IRQA}$	I	2.3.3 Interrupt and Mode Control
	Mode Select B/External Interrupt Request B	MODB/$\overline{IRQB}$	I	
	Mode Select C/Non-Maskable Interrupt Request	MODC/$\overline{NMI}$	I	
	Reset	$\overline{RESET}$	I/O	
SSI	Serial Clock Zero	SC0	I/O	2.3.4 Synchronous Serial Interface (SSI)
	Serial Clock One	SC1	I/O	
	Serial Clock Two	SC2	I/O	
	SSI Serial Clock	SCK	I/O	
	SSI Receive Data	SRD	I	
	SSI Transmit Data	STD	O	
On-Chip Emulation and JTAG	Debug Serial Input/Chip Status 0/Test Data In	DSI/OS0/TDI	I/O	2.3.5 On-Chip Emulation (OnCE) and JTAG Pins
	Debug Serial Clock/Chip Status 1/Test Clock	DSCK/OS1/TCK	I/O	
	Debug Serial Output/Test Data Out	DSO/TDO	O	
	Debug Request Input/Test Mode Select	$\overline{DR}$/TMS	I	
	$\overline{ONCE}$ Reset	$\overline{TRST}$	/I	
	JTAG /$\overline{ONCE}$ Mode Select	JTAG_$\overline{ONCE}$	I	
Spare	Not Connected	DNC1–2		2.1.18 No-Connect Pins
Power	DSP Clock Synthesizer Power	PVCC	I	2.3.6 IMP and DSP Clock and PLL Pins
	DSP Clock Synthesizer Ground	PGND	I	
	System Power Supply and Return	VCC, GND	I	2.4 Power and Ground Pins

2.3.1 Port A Address and Data Bus

The port A address and data bus signals control the access to external memory. They are three-stated during reset unless noted otherwise, and may require pull-up resistors to minimize power consumption and to prevent erroneous operation.

NOTE

All unused inputs should have pull-up resistors for two reasons: 1) floating inputs draw excessive power, and 2) a floating input

can cause erroneous operation. For example, during reset, all signals are three-stated. Without pull-up resistors, the $\overline{BR}$ and $\overline{WT}$ signals may become active, causing two or more memory chips to try to simultaneously drive the external bus, which can damage the memory chips. A pull-up resistor in the 50K-ohm range should be sufficient.

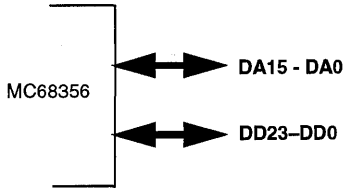


Figure 2-18. Port A Address and Data Bus Signals

DA15–DA0—Address

These three-state output pins specify the address for external program and data memory accesses. To minimize power dissipation, DA15–DA0 do not change state when external memory spaces are not being accessed.

DD23–DD0—Data Bus

These pins provide the bidirectional data bus for external program and data memory accesses. DD23–DD0 are in the high-impedance state when the bus grant signal is asserted.

2.3.2 Port A Bus Control

The port A bus control signals are discussed in the following paragraphs. The bus control signals provide a means to connect additional bus masters (which may be additional DSPs, microprocessors, direct memory access (DMA) controllers, etc.) through port A to the DSP. They are three-stated during reset and may require pull-up resistors to prevent erroneous operation.

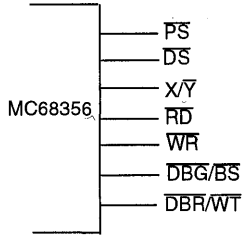


Figure 2-19. Port A Bus Control Signals

$\overline{PS}$—Program Memory Select

This three-state output is asserted only when external program memory is referenced (see Table 2-12).

Table 2-12. Program and Data Memory Select Encoding

PS	DS	X/$\overline{Y}$	External Memory Reference
1	1	1	No Activity
1	0	1	X Data Memory on Data Bus
1	0	0	Y Data Memory on Data Bus
0	1	1	Program Memory on Data Bus (Not Exception)
0	1	0	External Exception Fetch: Vector or Vector +1 (Development Mode Only)
0	0	X	Reserved
1	1	0	Reserved

$\overline{DS}$—Data Memory Select

This three-state output is asserted only when external data memory is referenced (see Table 2-12).

X/$\overline{Y}$—X/$\overline{Y}$ Select

This three-state output selects which external data memory space (X or Y) is referenced by $\overline{DS}$ (see Table 2-12).

$\overline{RD}$—Read Enable

This three-state output is asserted to read external memory on the data bus (DD23–DD0).

$\overline{WR}$—Write Enable

This three-state output is asserted to write external memory on the data bus (DD23–DD0).

The following two signal pairs share common pins, therefore the user must make a choice between having functionality of the Bus Request/Bus Grant pins (allows external bus masters) or the Wait/Bus Strobe pins (allows external peripherals to generate wait states).

$\overline{DBG}$/$\overline{BS}$—DSP Bus Grant/Bus Strobe

If the BR/BG Enable (BR/BGen) bit is clear, this output is DSP bus strobe ($\overline{BS}$):

The $\overline{BS}$ output is asserted when the DSP accesses port A. It acts as an early indication of the state of the external bus access by the DSP. It may also be used with the bus wait input, $\overline{WT}$, to generate wait states, a feature which provides capabilities such as connecting asyn-

chronous devices to the DSP, allowing devices with differing timing requirements to reside in the same memory space, allowing a bus arbiter to provide a fast multiprocessor bus access, and providing an alternative to the WAIT and STOP instructions to halt the DSP at a known program location and have a fast restart. This output is deasserted during hardware reset.

If the BR/BG Enable (BR/BGen) bit is set, this output is the DSP Bus Grant ($\overline{BG}$) signal: When the Bus Grant ($\overline{BG}$) output is asserted, it signals to the external device that it has been granted the external bus (i.e. port A has been three-stated). This output is deasserted during hardware reset.

$\overline{DBR}$/$\overline{WT}$—DSP Bus Request/Bus Wait

If the BR/BG Enable (BR/BGen) bit is clear, this input becomes the Wait ($\overline{WT}$) signal.

For as long as it is asserted by an external device, this input allows that device to force the DSP to generate wait states. If $\overline{WT}$ is asserted when $\overline{BS}$ is asserted, wait states will be inserted into the current cycle (see Section 14 Electrical Characteristics) for timing details.

If the BR/BG Enable (BR/BGen) bit is set, this input becomes the DSP Bus Request ($\overline{DBR}$) signal.

When the DSP Bus Request input ($\overline{DBR}$) is asserted, the DSP will always relinquish the bus to an external device such as a processor or DMA controller. The external device will become the new master of the external address and data buses while the DSP continues internal operations using internal memory spaces. When $\overline{BR}$ is deasserted, the DSP will again assume bus mastership.

When $\overline{BR}$ is asserted, the DSP will always release port A, including A15–A0, D23–D0, and the bus control pins ($\overline{PS}$, $\overline{DS}$, X/$\overline{Y}$, $\overline{RD}$, $\overline{WR}$, and $\overline{BS}$) by placing them in the high-impedance state after the execution of the current instruction has been completed.

NOTE

To prevent erroneous operation, the $\overline{BR}$ pin should be pulled up when it is not in use.

2.3.3 Interrupt and Mode Control

The interrupt and mode control pins select the chip's operating mode as it comes out of hardware reset, and they receive interrupt requests from external sources.

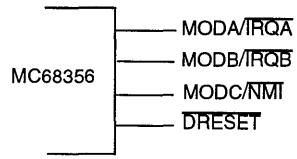


Figure 2-20. Interrupt and Mode Control

MODA/IRQA—Mode Select A/External Interrupt Request A/STOP Recovery

This input pin has three functions. It works with the MODB and MODC pins to select the chip's operating mode, it receives an interrupt request from an external source, and it turns on the internal clock generator, causing the chip to recover from the stop processing state. Reset causes this input to act as MODA.

During reset, this pin should be forced to the desired state, because as the chip comes out of reset, it reads the states of MODA, MODB, and MODC and writes the information to the operating mode register to set the chip's operating mode. (Operating modes are discussed in Section 5 Memory Map.) After the chip has left the reset state, the MODA pin automatically changes to external interrupt request ($\overline{\text{IRQA}}$).

$\overline{\text{IRQA}}$ receives external interrupt requests. It can be programmed to be level sensitive or negative-edge-triggered. When the signal is edge-triggered, triggering occurs at a voltage level and is not directly related to the fall time of the interrupt signal. However, as the fall time of the interrupt signal increases, the probability that noise on $\overline{\text{IRQA}}$ will generate multiple interrupts also increases.

NOTE

If the SELA bit is set in the DISC register, then this pin's functionality will be replaced by the MODA/$\overline{\text{IRQA}}$ bit in the DISC register and this pin will not have any effect on DSP operation until the DISC bits are cleared either by software or RESET or the IMP.

MODB/IRQB—Mode Select B/External Interrupt Request B

This input pin works with the MODA and MODC pins to select the chip's operating mode, and it receives an interrupt request from an external source. Reset causes this input to act as MODB.

During reset, this pin should be forced to the desired state, because as the chip comes out of reset, it reads the states of the mode pins and writes the information to the operating mode register, which sets the chip's operating mode. After the chip has left the reset state, the MODB pin automatically changes to external interrupt request $\overline{\text{IRQB}}$.

$\overline{\text{IRQB}}$ receives external interrupt requests. It can be programmed to be level sensitive or negative-edge-triggered. When the signal is edge-triggered, triggering occurs at a voltage

level and is not directly related to the fall time of the interrupt signal. However, as the fall time of the interrupt signal increases, the probability that noise on $\overline{\text{IRQB}}$ will generate multiple interrupts also increases.

NOTE

If the SELB bit is set in the DISC register, then this pin's functionality will be replaced by the MODA/$\overline{\text{IRQB}}$ bit in the DISC register and this pin will not have any effect on DSP operation until the DISC bits are cleared either by software or RESET or the IMP.

MODC/$\overline{\text{NMI}}$—Mode Select C/Non-Maskable Interrupt Request

This input pin works with the MODA and MODB pins to select the chip's operating mode, and it receives an interrupt request from an external source. Reset causes this input to act as MODC.

During reset, this pin should be forced to the desired state, because as the chip comes out of reset, it reads the states of the mode pins and writes the information to the operating mode register, which sets the chip's operating mode. After the chip has left the reset state, the MODC pin automatically changes to a nonmaskable interrupt request ($\overline{\text{NMI}}$) input.

The negative-edge-triggered $\overline{\text{NMI}}$ receives nonmaskable interrupt requests. Triggering occurs at a voltage level and is not directly related to the fall time of the interrupt signal. However, as the fall time of the interrupt signal increases, the probability that noise on $\overline{\text{NMI}}$ will generate multiple interrupts also increases.

NOTE

If the SELC bit is set in the DISC register, then this pin's functionality will be replaced by the MODC/$\overline{\text{NMI}}$ bit in the DISC register and this pin will not have any effect on DSP operation until the DISC bits are cleared either by software or RESET or the IMP.

$\overline{\text{DRESET}}$—DSP Reset

This Schmitt trigger input pin is used to reset the DSP. When $\overline{\text{RESET}}$ is asserted, the DSP is initialized and placed in the reset state. When $\overline{\text{RESET}}$ is deasserted, the chip writes the mode pin (MODA, MODB, MODC) information to the operating mode register, setting the chip's operating mode. The CSEL pin is sampled and the PEN bit of the PLL control register is written accordingly. When the chip comes out of the reset state, deassertion occurs at a voltage level and is not directly related to the rise time of the $\overline{\text{RESET}}$ signal. However, the probability that noise on $\overline{\text{RESET}}$ will generate multiple resets increases with increasing rise time of the $\overline{\text{RESET}}$ signal.

NOTE

If the SELR bit is set in the DISC register, then this pin's functionality will be replaced by the RST bit in the DISC register and

this pin will not have any effect on DSP operation until the DISC bits are cleared either by software or RESET or the IMP.

2.3.4 Synchronous Serial Interface (SSI)

The SSI signals are presented in the following paragraphs. The SSI operating mode affects the definition and function of SSI control pins SC0, SC1 and SC2. They are introduced briefly here and are described in more detail in Section 12 DSP Serial Ports.

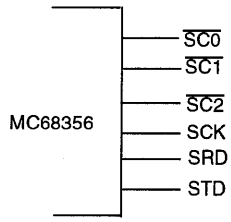


Figure 2-21. Synchronous Serial Interface Signals

SC0—Serial Clock Zero

This bidirectional pin's function is determined by whether the SCLK is in synchronous or asynchronous mode. In synchronous mode, this pin is used for serial flag I/O. In asynchronous mode, this pin receives clock I/O. SC0 can be programmed as a general-purpose I/O pin (PC3) when the SSI SC0 function is not being used, and it is configured as a GPIO input pin during hardware reset.

SC1—Serial Control One

The SSI uses this bidirectional pin to control flag or frame synchronization. This pin's function is determined by whether the SCLK is in synchronous or asynchronous mode. In asynchronous mode, this pin is frame sync I/O. For synchronous mode with continuous clock, this pin is serial flag SC1 and operates like the SC0. SC0 and SC1 are independent serial I/O flags but may be used together for multiple serial device selection. SC1 can be programmed as a general-purpose I/O pin (PC4) when the SSI SC1 function is not being used, and it is configured as a GPIO input pin during hardware reset.

SC2—Serial Control Two

The SSI uses this bidirectional pin to control frame synchronization only. As with SC0 and SC1, its function is defined by the SSI operating mode. SC2 can be programmed as a general-purpose I/O pin (PC5) when the SSI SC2 function is not being used, and it is configured as a GPIO input pin during hardware reset.

SCK—SSI Serial Clock

This bidirectional pin provides the serial bit rate clock for the SSI when only one clock is being used. SCK can be programmed as a general-purpose I/O pin (PC6) when it is not needed as an SSI pin, and it is configured as a GPIO input pin during hardware reset.

SRD—SSI Receive Data

This input pin receives serial data into the SSI receive shift register. SRD can be programmed as a general-purpose I/O pin (PC7) when it is not needed as an SSI pin, and it is configured as a GPIO input pin during hardware reset.

STD—SSI Transmit Data

This output pin transmits serial data from the SSI transmit shift register. STD can be programmed as a general-purpose I/O pin (PC8) when it is not needed as an SSI pin, and it is configured as a GPIO input pin during hardware reset.

2.3.5 On-Chip Emulation (OnCE) and JTAG Pins

Since the OnCE port is multiplexed with the JTAG pins, the following paragraphs will describe the OnCE port pin functionality first, followed by those same pin's JTAG functionality.

For more information on the OnCE port operation please refer to the *DSP56000 Digital Signal Processor Family Manual* (DSP56KFAMUM/AD).

JTAG_$\overline{\text{ONCE}}$—JTAG/$\overline{\text{ONCE}}$ Mode Select

When this pin is strapped to V_{CC}, the JTAG/$\overline{\text{ONCE}}$ pin group function as JTAG pins; when this pin is grounded the ONCE/JTAG pins group functions as ONCE pins.

2.3.5.1 On-Chip Emulation Pins

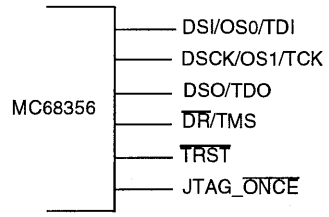


Figure 2-22. On-Chip Emulation and JTAG Signals

DSI/OS0—Debug Serial Input/Chip Status 0

Serial data or commands are provided to the OnCE controller through the DSI/OS0 pin when it is an input. The data received on the DSI pin will be recognized only when the DSP has entered the debug mode of operation. Data is latched on the falling edge of the DSCK serial clock. Data is always shifted into the OnCE serial port most significant bit (MSB) first. When the DSI/OS0 pin is an output, it works in conjunction with the OS1 pin to provide chip status information. The DSI/OS0 pin is an output when the processor is not in debug mode. When switching from output to input, the pin is three-stated. During hardware reset, this pin is defined as an output and it is driven low.

NOTE

To avoid possible glitches, an external pull-down resistor should be attached to this pin.

DSCK/OS1—Debug Serial Clock/Chip Status 1

The DSCK/OS1 pin supplies the serial clock to the OnCE when it is an input. The serial clock provides pulses required to shift data into and out of the OnCE serial port. (Data is clocked into the OnCE on the falling edge and is clocked out of the OnCE serial port on the rising edge.) The debug serial clock frequency must be no greater than 1/8 of the processor clock frequency. When an output, this pin, in conjunction with the OS0 pin, provides information about the chip status. The DSCK/OS1 pin is an output when the chip is not in debug mode. When switching from output to input, the pin is three-stated. During hardware reset, this pin is defined as an output and it is driven low.

NOTE

To avoid possible glitches, an external pull-down resistor should be attached to this pin.

DSO—Debug Serial Output

Serial data is read from the OnCE through the DSO output pin, as specified by the last command received from the external command controller. Data is always shifted out the OnCE serial port most significant bit (MSB) first. Data is clocked out of the OnCE serial port on the rising edge of DSCK.

The DSO pin also provides acknowledge pulses to the external command controller. When the chip enters the debug mode, the DSO pin will be pulsed low to indicate (acknowledge) that the OnCE is waiting for commands. After receiving a read command, the DSO pin will be pulsed low to indicate that the requested data is available and the OnCE serial port is ready to receive clocks in order to deliver the data. After receiving a write command, the DSO pin will be pulsed low to indicate that the OnCE serial port is ready to receive the data to be written; after the data is written, another acknowledge pulse will be provided.

During hardware reset and when the processor is idle, the DSO pin is held high.

$\overline{\text{DR}}$—Debug Request Input

The debug request input ($\overline{\text{DR}}$) allows the user to enter the debug mode of operation from the external command controller. When $\overline{\text{DR}}$ is asserted, it causes the DSP to finish the current instruction being executed, save the instruction pipeline information, enter the debug mode, and wait for commands to be entered from the DSI line. While in debug mode, the $\overline{\text{DR}}$ pin lets the user reset the OnCE controller by asserting it and deasserting it after receiving acknowledge. It may be necessary to reset the OnCE controller in cases where synchronization between the OnCE controller and external circuitry is lost. $\overline{\text{DR}}$ must be deasserted after the OnCE responds with an acknowledge on the DSO pin and before sending the first OnCE command. Asserting $\overline{\text{DR}}$ will cause the chip to exit the STOP or WAIT state.

2

2.3.5.2 JTAG Pins

The following signals are used with the onboard test logic defined by the IEEE 1149.1 JTAG standard. See Section 13 IEEE 1149.1 Test Access Port for more information on the use of these signals. These pins are active when the JTAG_$\overline{\text{ONCE}}$ pin is strapped high.

$\overline{\text{TRST}}$—Test Reset

This active low input provides an asynchronous reset to the test logic.

TCK—Test Clock

This input provides a clock for onboard test logic defined by the IEEE 1149.1 standard.

$\overline{\text{TMS}}$—Test Mode Select

This input controls test mode operations for onboard test logic defined by the IEEE 1149.1 standard.

TDI—Test Data In

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

TDO—Test Data Out

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

2.3.6 IMP and DSP Clock and PLL Pins

The clock and PLL pins are shown in Figure 2-23.

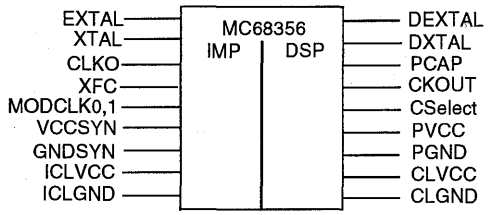


Figure 2-23. IMP and DSP Clock and PLL Pins

2.3.6.1 IMP PLL and Clock Pins

EXTAL—IMP 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 frequency of EXTAL may range from 0 MHz to 25

MHz (or the maximum rated frequency, whichever is higher). 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 Section 14 Electrical Characteristics. A valid clock signal oscillates between a low voltage of between GND – 0.3 and .6 volts and a high voltage of between 4.0 and V_{CC} volts.

XTAL—IMP 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.

CLKO—IMP 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 IPLCR register (see 3.4.3.4 IMP PLL and Clock Control Register (IPLCR)) to one-third, two-thirds, full strength, or this output can be disabled.

XFC—IMP External Filter Capacitor

This pin is a connection for an external capacitor to filter the PLL. See 3.4.5 IMP PLL Pins for more details.

MODCLK1-0—Clock Mode Select

The state of these input signals during reset selects the type of external clock that is used by the phase locked loop (PLL) in the clock synthesizer to generate the system clocks. Table 2-13 shows the default values of the PLL. See 3.3.1 Default System Clock Generation for more details of the MODCK0-1 options.

Table 2-13. Default Operation Mode of the IMP PLL

MODCK 1-0	PLL	Mult. Factor (MF+1)	EXTAL Freq. (Examples)	CLKIN to the PLL	IMP System Clock
00	Disabled	x	-	=EXTAL	=EXTAL
01	Enabled	1	>10 MHz	=EXTAL	=EXTAL
10	Enabled	4	4.192MHz	4.192MHz	16.768 MHz
11	Enabled	401	32.768KHz	32.768KHz	13.14 MHz

VCCSYN—IMP Analog PLL Circuit Power

This pin is dedicated to the IMP analog PLL circuits. The voltage should be well regulated and the pin should be provided with an extremely low impedance path to the V_{CC} power rail.

2

V_{CCSYN} should be bypassed to GND by a 0.1µF capacitor located as close as possible to the chip package.

GNDSYN—IMP Analog PLL Circuits' Ground

This pin is dedicated to the IMP analog PLL circuits. The pin should be provided with an extremely low impedance path to ground. GNDSYN should be bypassed to VCCSYN by a 0.1µF capacitor located as close as possible to the chip package.

2.3.6.2 DSP PLL and Clock Pins

DEXTAL—DSP External Clock/Crystal Input

The DEXTAL input interfaces the internal crystal oscillator input to an external crystal or an external clock. If the IMP EXTAL clock source is used to drive the DSP clock, this pin should be grounded.

DXTAL—DSP Crystal Output

This output connects the internal crystal oscillator output to an external crystal. If an external clock or the IMP clock source is used, XTAL should not be connected. It may be disabled through software control using the XTLD bit in the PLL control register.

PCAP—DSP PLL Filter Off-Chip Capacitor

This input acts as an off-chip capacitor for the PLL filter. One terminal of the capacitor is connected to PCAP while the other terminal is connected to PVCC. The capacitor value is specified in Section 14 Electrical Characteristics.

CKOUT—DSP PLL Output Clock

This output pin provides a 50% duty cycle output clock synchronized to the internal processor clock when the PLL is enabled and locked. When the PLL is disabled, the output clock at CKOUT is derived from, and has the same frequency and duty cycle as DEXTAL.

NOTE

If the PLL is enabled and the multiplication factor is less than or equal to 4, then CKOUT is synchronized to DEXTAL.

CSelect - DSP Clock Select

During the assertion of hardware reset, the value at the CSelect input pin determines the DSP PLL input clock source and together with the MODCLK1-0 pins the DSP PLL operational mode. The DSP PLL enable/disable mode is written into the PEN bit of the PLL control register. The PEN bit enables the PLL by causing it to derive the internal clocks from the PLL VCO output. When the bit is clear, the PLL is disabled and the chip's internal clocks are derived from either the clock connected to the DEXTAL pin or the IMP's CLKIN. Table 2-14

shows the default values of the PLL after hardware reset is deasserted - the CSelect pin is ignored. See the clock synthesizer description of the DSP for more details.

Table 2-14. Default Operation Mode of the DSP PLL

CSelect	MODCK 1-0	PLL	Mult. Factor (MF+1)	DEXTAL Freq. (Examples)	CLKIN to the PLL	DSP System Clock
0	00	Disabled	x	-	=IMP EXTAL	=IMP EXTAL
0	01	Disabled	x	-	=IMP EXTAL	=IMP EXTAL
0	10	Disabled	x	-	=IMP EXTAL	=IMP EXTAL
0	11	Enabled	401	32.768KHz	32.768 kHz	13.14 MHz
1	xx	Disabled	x	-	=DSP EXTAL	=DSP EXTAL

PV_{CC}—DSP Analog PLL Circuit Power

The V_{CC} input is dedicated to the DSP analog PLL circuits. The voltage should be well regulated and the pin should be provided with an extremely low impedance path to the V_{CC} power rail. PV_{CC} should be bypassed to PGND by a 0.1µF capacitor located as close as possible to the chip package.

PGND—DSP Analog PLL Circuits' Ground

This GND input is dedicated to the DSP analog PLL circuits. The pin should be provided with an extremely low impedance path to ground. PV_{CC} should be bypassed to PGND by a 0.1µF capacitor located as close as possible to the chip package.

2.4 POWER AND GROUND PINS

The MC68356 has 16 power supply pins throughout the outer four rows/columns and 40 V_{CC} pins on the V_{CC} ring located on the perimeter of the inner ground pins. There are 3 grounds throughout the four row/columns and 81 ground pins located in the inner 9 x 9 ground matrix. The inner ground matrix should be connected directly to an internal ground plane with individual vias for every pad connecting to the ground plane. These pins are especially important for efficient heat dissipation. Similarly the outer ring of V_{CC} pins should be connected to the power plan with individual vias for every pad.Special attention should be paid to connecting the DSP and IMP PLL power pins designated by PVCC, PGND and VCCSYN and GNDSYN respectively (see 2.3.6 IMP and DSP Clock and PLL Pins). Careful attention has been paid to reducing IMP noise, potential cross-talk, and RF radiation from the output drivers. Inputs may be +5 V when V_{CC} is 0 V without damaging the device.

2.5 WHEN TO USE PULLUP RESISTORS

Pins that are input-only or output-only do not require external pullups. The bidirectional bus control signals require pullups since they are three-stated by the processor 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 IMP 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_{CC} 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 RESET pin description for the condition of all pins during reset.

SECTION 3
CLOCK GENERATION AND LOW POWER CONTROL

Since the MC68356 consists of two separate main functional blocks: an Integrated Multiprotocol Processor (IMP) and a DSP56002 with expanded memory, two separate clocking generation blocks are needed to assure that both processors can be clocked independently to achieve maximum flexibility in this dual processor architecture. The MC68356 contains two clock circuits, one for the IMP section and one for the DSP section of the device. Each clock circuit consists of crystal oscillator drive circuit capable of driving either an external crystal or accepting an oscillator clock, a PLL clock synthesizer capable of multiplying a low frequency clock or crystal such as a 32-KHz watch crystal up to the maximum clock rate of each processor, and a low power divider which allows dynamic gear down and gear up of the system clock for each processor on the fly. While the dual-clock-generation architecture allows each section to run completely independently, both clock generation blocks can rely on the same crystal or oscillator clock source to save external oscillator components and board space.

3.1 CLOCK GENERATION AND LOW POWER CONTROL

- **Dual On-Chip Clock Synthesizers** (each with output system clocks)
  - **—Two Independent Oscillator Drive Circuits and Pins**
  - **—Two Independent PLL Clock Synthesizer Circuits with Low Power Output Clock Divider Blocks.**
  - **—One External Crystal Can Generate Clocks for both IMP and DSP**
- **Low Power Control Of IMP**
  - **—Slow-Go Modes using PLL Clock Divider Blocks**
  - **—Varied Low Power STOP Modes for Optimizing Wake-Up Time to Low Power Mode Power Consumption: Stand-By, Doze and STOP.**
- **Low Power Control DSP**
  - **—Wait and Stop modes**
  - **—Wake-Up by IMP through Direct Internal Connection of $\overline{\text{RESET}}$ and $\overline{\text{IRQ}}$ signals**
  - **—Minimum Current Drain in μA Range**

3.2 PLL AND OSCILLATOR CHANGES TO IMP AND DSP

The oscillator that was on the MC68302 has been replaced by the new clock synthesizer described in this section.The registers related to the oscillator have been either removed or changed according to the description below. Several control bits are still available but have new locations.

The low power modes on the MC68302 have changed completely and will be discussed later in 3.5.1 IMP Low Power Modes

The DSP PLL is the same as described in the DSP family manual. One addition is that the input clock source may now be optionally routed from the IMP EXTAL pin.

The PINIT pin has been removed; DSP PLL initialization after reset is determined by the combination of CSelect and MODCK1–0 pins as shown in Figure 3-1. The PLOCK and CKP pins are not implemented on the MC68356.

3.2.1 Clock Control Register

The clock control register address $FA is not implemented on the MC68356. This register location has been reassigned to the IOMCR and ICKCR registers. The clock control register bits have been reassigned as follows:

CLKO Drive Options (CLKOMOD1–2)

These bits are now in the IMP clock control register (IPLCR) on the MC68356, see 3.4.3.4 IMP PLL and Clock Control Register (IPLCR).

Tri-State TCLK1 (TSTCLK1)

This bit is now in the DISC register on the MC68356, see 6.9.2 IMP-DSP Reset and Mode Interconnections Register.

Tri-State RCLK1 (TSRCLK1)

This bit is now in the internal connection register (ICR) on the MC68356, see 6.9.2 IMP-DSP Reset and Mode Interconnections Register.

Disable BRG1 (DISBRG1)

This bit is now in the internal connection register (ICR) on the MC68356, see 6.9.2 IMP-DSP Reset and Mode Interconnections Register.

3.3 MC68356 SYSTEM CLOCK GENERATION

Figure 3-1 shows the MC68356 system clock schematic which includes both the DSP and IMP clock synthesizers. The MC68356 has two sets of clock generation circuit blocks for both the DSP and IMP sections. Each block includes an on-chip oscillator, a clock synthesizer, and a low-power divider, which allows a comprehensive set of options for generating system clocks. The choices offer many opportunities to save power and system cost, without sacrificing flexibility and control. In addition to performing frequency multiplication, the PLL block can also provide EXTAL to CLKO skew elimination, and dynamic low power divides of the output PLL system clock.

To allow maximal flexibility for various applications, the DSP clock may be generated by the DSP clock oscillator, or to reduce power, noise, and component count, the IMP external crystal can generate clocks for both IMP and the DSP. Clock source and default settings are determined during independent resets of the IMP and DSP sections. The MC68356 decodes the CSelect and MODCK1–0 pins and the value of these pins determines the initial

clocking for the part. Further changes to the clocking scheme can be made by software. After reset, the 68000 core can control the IMP clocking through the following registers:

1. IMP Operation Mode Control Register, IOMCR (3.5.1.6 IMP Operation Mode Control Register (IOMCR)).
2. IMP PLL and Clock Control Register, IPLCR (3.4.3.1 Frequency Multiplication).
3. IMP Interrupt Wake-Up Control Register, IWUCR (3.5.2.4 IMP Wake-Up Control Register – IWUCR).
4. Periodic Interrupt Timer Register, PITR (6.5.4 Periodic Interrupt Timer).

The DSP can control its clocking through the following register:

DSP PLL Control Register, (3.7.6 DSP PLL Control Register (PCTL)).

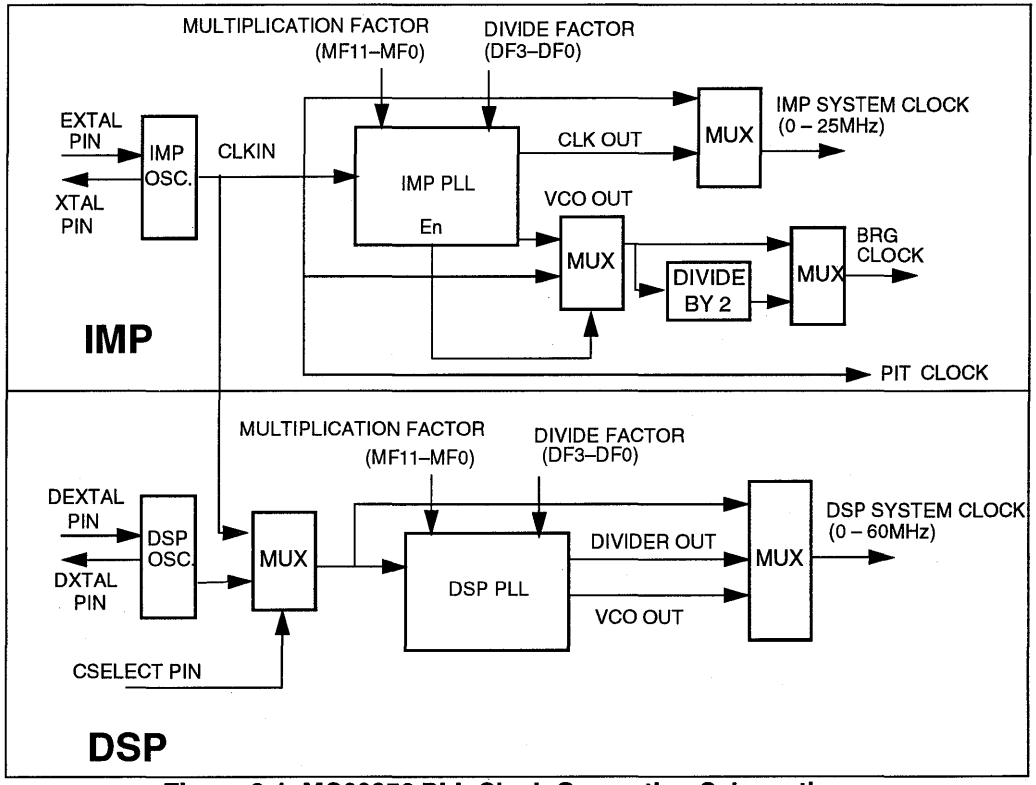


Figure 3-1. MC68356 PLL Clock Generation Schematic

3.3.1 Default System Clock Generation

During the assertion of hardware reset, the value at the CSelect and MODCLK1–0 input pins determine the input clock source for the DSP and IMP as well as which PLLs will be enabled according to Table 3-1. After the deassertion of reset, these pins are ignored.

If the CSelect pin is low, the DSP will use the IMP EXTAL pin as its clock source. The DSP PLL can be enabled or disabled after reset by writing to the PEN bit of the DSP PLL control register (PCTL). When the CSelect bit is set, the DSP will come out of reset with its PLL disabled and the DSP's internal clocks connected to the DSP EXTAL pin (See Figure 3-1).

NOTE

The DSP DEXTAL pin should be grounded if the DSP is using the IMP EXTAL pin as its clock source (CSelect=0).

The MODCK1–0 pins control the IMP clock selection at hardware reset and the DSP PLL multiplication factor if CSELECT=0 and MODCLK = 11. The IMP PLL can be enabled or disabled and the multiplication factor can be preset to support different industry standard crystals. After reset the multiplication factor can be changed in the IPLCR register, and the IMP PLL divide factor can be set in the IOMCR register.

NOTE

The IMP and DSP PLL input frequency ranges are limited to between 25 kHz and the maximum operating frequency, and the PLL output frequency range before the low power divider is limited to between 10 MHz and the maximum system clock frequency (25 MHz for a 25 MHz IMP and 60 MHz for a 60 MHz DSP).

Table 3-1 IMP and DSP Default System Clock Generation

CSelect	MODCK 1–0	DSP PLL	DSP MF+1	DSP System Clock	Example IMP EXTAL Freq.	IMP PLL	IMP MF+1	IMP System Clock
0	00	Disabled	x	IMP EXTAL	25 MHz	Disabled	x	IMP EXTAL
0	01	Disabled	x	IMP EXTAL	25 MHz	Enabled	1	IMP EXTALx1
0	10	Disabled	x	IMP EXTAL	4.192 MHz	Enabled	4	IMP EXTALx4
0	11	Enabled	401	IMP EXTALx 401	32.768 kHz	Enabled	401	IMP EXTALx401
1	00	Disabled	x	DSP EXTAL	25 MHz	Disabled	x	IMP EXTAL
1	01	Disabled	x	DSP EXTAL	25 MHz	Enabled	1	IMP EXTALx1
1	10	Disabled	x	DSP EXTAL	4.192 MHz	Enabled	4	IMP EXTALx4
1	11	Disabled	x	DSP EXTAL	32.768 kHz	Enabled	401	IMP EXTALx401

Note:
By loading the IPLCR register the user can change the multiplication factor of the PLL after RESET.
By loading the IOMCR register, the user can change the power saving divide factor of the IMP PLL.

3.4 IMP SYSTEM CLOCK GENERATION

3.4.1 System Clock Configuration

The IMP has an on-chip oscillator and phased locked loop (Figure 3-2). These features provide flexible ways to save power and reduce system cost. The operation of the clock generation circuitry is determined by the following registers.

The IMP Operation Mode Control Register, IOMCR in 3.5.1.6 IMP Operation Mode Control Register (IOMCR).

The IMP PLL and Clock Control Register, IPLCR in 3.4.3 Phase-Locked Loop (PLL).

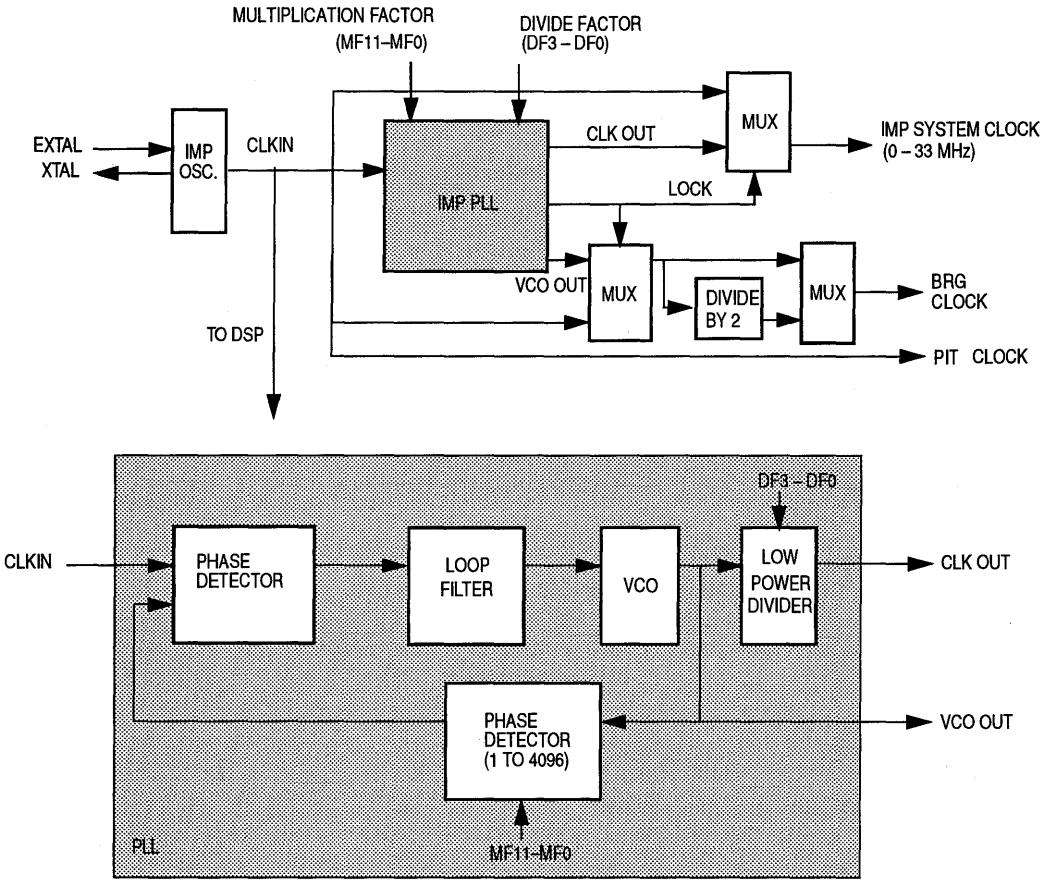


Figure 3-2. IMP System Clocks Schematic - PLL Enabled

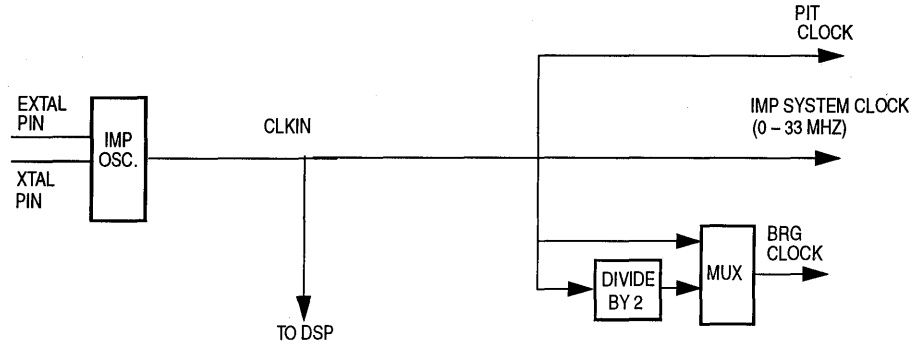


Figure 3-3. IMP System Clocks Schematic - PLL disabled

Figure 3-2 shows the IMP system clocks schematic with the IMP PLL enabled. Figure 3-3 shows the IMP system clocks schematic with the IMP PLL disabled.

The clock generation features of the IMP are discussed in the following paragraphs.

3.4.2 On-Chip Oscillator

A 32.768-kHz watch crystal provides an inexpensive reference, but the EXTAL reference crystal frequency can be any frequency from 25 kHz to 6.0 MHz. Additionally, the system clock frequency can be driven directly onto the EXTAL pin. In this case, the EXTAL frequency should be the exact system frequency desired (0 to 25 MHz for the IMP), and the XTAL pin should be left floating. Figure 3-4 shows all the external connections required for the on-chip oscillator (as well as the PLL VCC and GND connections).

3.4.3 Phase-Locked Loop (PLL)

The IMP PLL's main functions are frequency multiplication, and EXTAL pin to CLKO pin clock skew elimination. The phase-locked loop takes the CLKIN frequency and outputs a high-frequency source used to derive the general system frequency of the IMP. The IMP PLL is comprised of a phase detector, loop filter, voltage-controlled oscillator (VCO), and multiplication block.

3.4.3.1 Frequency Multiplication

The IMP PLL can multiply the CLKIN input frequency by any integer between 1 and 4096. The multiplication factor may be changed to the desired value by writing the MF11–MF0 bits in the IPLCR. When the IMP PLL multiplier is modified in software, the IMP PLL will lose lock, and the clocking of the IMP will stop until lock is regained (worst case is 2500 EXTAL clocks). If an alteration in the system clock rate is desired without losing IMP PLL lock, the value in the low-power clock divider can be to modified to change the system clock rate dynamically. The low power clock divider bits are located in the IOMCR register.

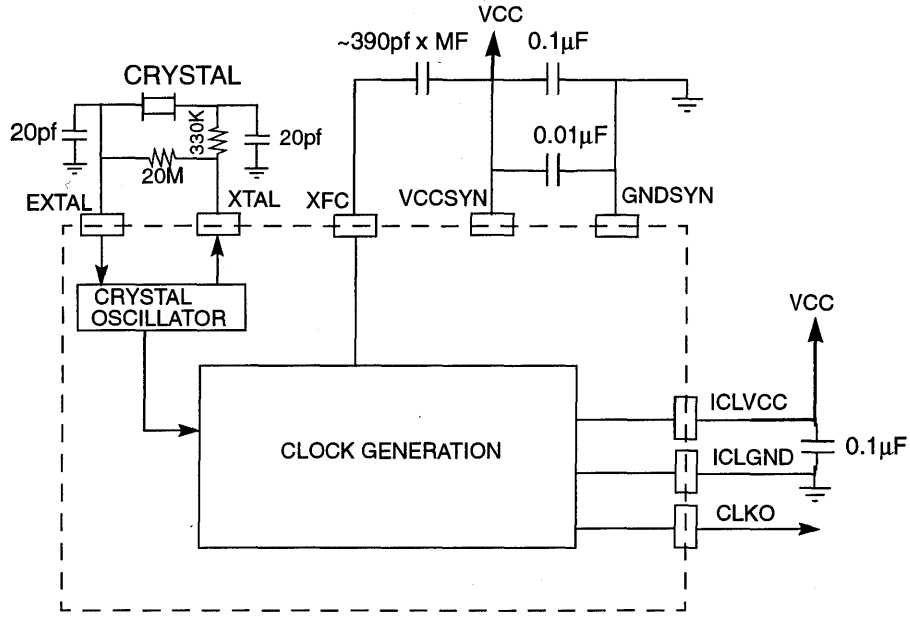


Figure 3-4. PLL External Components

NOTE

If IMP PLL is enabled, the multiplication value must be large enough to result in the VCO clock being greater than 10 MHz.

3.4.3.2 Skew Elimination

The IMP PLL is also capable of eliminating the skew between the EXTAL and IMP system clock (CLKO). When the IMP PLL is enabled with a MF<5, the EXTAL to IMP clock skew is reduced to less than 4 ns. Without the IMP PLL enabled, the clock skew could be much larger. This significant reduction of the clock skew is useful for synchronous clocking of multiple external system components. The IMP PLL also can compensate for duty cycle mismatch on the EXTAL input and will create a 50% duty cycle output

NOTE

When PLL is active, skew elimination is guaranteed only if the multiplication factor is less than 5.

3.4.3.3 Low Power PLL Clock Divider

The output of the IMP VCO is sent to a low power divider block. The clock divider can divide the output frequency of the VCO before it generates the system clock. The clock for the baud rate generators (BRGs) bypasses this clock divider.

The purpose of the clock divider is to allow the user to reduce and restore the operating frequency of the IMP without losing the IMP's PLL lock. Using the clock divider, the user can still obtain full IMP operation, but at a slower frequency. The BRG is not affected by the low

power divider circuitry so previous BRG divider settings will not have to be changed when the divide factors are changed.

When the PLL low power divider bits (DF0–3) are programmed to a non-zero value, the IMP is in SLOW_GO mode. The selection and speed of the SLOW_GO mode may be changed at any time, with changes occurring immediately.

NOTE

The IMP low power clock divider is active only if the IMP PLL is active.

The low-power divider block is controlled in the IOMCR. The default state of the low-power divider is to divide all clocks by 1.

If the low-power divider block is not used and the user is concerned that errant software could accidentally write the IOMCR, the user may set a write protection bit in IOMCR to prevent further writes to the register.

3.4.3.4 IMP PLL and Clock Control Register (IPLCR)

IPLCR is a 16-bit read/write register used to control the IMP's PLL, multiplication factor and CLKO drive strength. This register is mapped in the 68000 bus space at address $0F8. If the 68000 bus is set to 8 bits (BUSW grounded at reset), during 8-bit accesses, changes to the IPLCR will take effect in the IMP PLL after loading the high byte of IPLCR (the low byte is written first). The WP bit in IPLCR is used as a protect mechanism to prevent erroneous writing. When this bit is set further accesses to the IPLCR will be blocked.

IMP PLL and Clock Control Register (IPLCR) $0F8

15	14	13	12	11	10	9	8
IPLWP	CLKOMOD0–1		PEN	MF11	MF10	MF9	MF8
RESET							
0	0	0	MODCLK	0	0	0	MODCLK0,1

7	6	5	4	3	2	1	0
MF7	MF6	MF5	MF4	MF3	MF2	MF1	MF0
RESET							
MODCLK0,1	0	0	MODCLK0,1	0	0	MODCLK0	MODCLK0

Read/Write

MF 11–0—Multiplication Factor

These bits define the multiplication factor that will be applied to the IMP PLL input frequency. The multiplication factor can be any integer from 1 to 4096. The system frequency is ((MF bits + 1) x EXTAL). The multiplication factor must be chosen to ensure that the resulting VCO output frequency will be in the range from 10 MHz to the maximum allowed clock input frequency (e.g. 25 MHz for a 25 MHz IMP).

The value 000 results in a multiplier value of 1. The value $FFF results in a multiplier value of 4096.

Any time a new value is written into the MF11–MF0 bits, the IMP PLL will lose the lock condition, and after a delay, will relock. When the IMP PLL loses its lock condition, all the clocks that are generated by the IMP PLL are disabled. After hardware reset, the MF11–MF0 bits default to either 0, 3 or 400 ($190 hex) depending on the MODCK1–0 pins (giving a multiplication factor of 1, 4 or 401). If the multiplication factor is 401, then a standard 32.768 kHz crystal generates an initial general system clock of 13.14 MHz. If the multiplication factor is 4, then a standard 4.192 MHz crystal generates an initial general system clock of 16.768 MHz. The user would then write the MF bits or adjust the output frequency to the desired frequency.

NOTE

Since the clock source for the periodic interrupt timer is CLKIN (see Figure 3-1), the PIT timer is not disturbed when the IMP PLL is in the process of acquiring lock.

PEN—PLL Enable Bit

The PEN bit enables the IMP PLL operation. When the IMP PLL is disabled, the VCO is not operating in order to minimize power consumption. The PEN bit may be set by software but it cannot be reset by software. During hardware reset this bit is set if the MODCK1–0 pins specify that the IMP PLL is enabled. The only way to clear PEN is to hold the MODCK1–0 pins low during a hardware reset.

0 = The IMP PLL is disabled. Clocks are derived directly from the EXTAL pin.
1 = The IMP PLL is enabled. Clocks are derived from the CLKOUT output of the PLL.

CLKODM0–1—CLKO Drive Mode 0–1

These bits control the output buffer strength of the CLKO pin. Those bits can be dynamically changed without generating spikes on the CLKO pin. Disabling CLKO will save power and reduce noise.

00 = Clock Out Enabled, Full-Strength Output Buffer.
01 = Clock Out Enabled, 2/3-Strength Output Buffer
10 = Clock Out Enabled, 1/3-Strength Output Buffer
11 = Clock Out Disabled (CLKO is driven high by internal pullup)

NOTE

These IMP bits are in a different address location than in the MC68302, where they are located at address $FA (bits 15, 14).

IPLWP—IMP PLL Control Write Protect Bit

This bit prevents accidental writing into the IPLCR. After reset, this bit defaults to zero to enable writing. Setting this bit prevents further writing (excluding the first write that sets this bit).

3.4.4 IMP Internal Clock Signals

The following paragraphs describe the IMP internal clock signals.

3.4.4.1 IMP System Clock

The IMP system clock is supplied to all modules on the IMP (with the exception of the BRG clocks which are connected directly to the VCO output with the PLL enabled). The IMP can be programmed to operate with or without IMP PLL. If IMP PLL is active, the system clock will be driven by PLL clock divider output. If IMP PLL is not active, the system clock will be driven by the PLL input clock (CLKIN).

3.4.4.2 BRG Clock

The clock to the BRGs can be supplied from the IMP PLL input (CLKIN) when the IMP PLL is disabled, or from the IMP PLL VCO output (when the PLL is enabled). The BRG prescaler input clock may be optionally programmed to be divided by 2 to allow very low baud rates to be generated from the system clock by setting the BCD bit in the IOMCR.

3.4.4.3 PIT Clock

CLKIN is supplied to the periodic interrupt timer (PIT) submodule which allows the PIT clock to run independently of the system clock (refer to Figure 3-1 and 6.5.4 Periodic Interrupt Timer).

3.4.5 IMP PLL Pins

The following pins are dedicated to the IMP PLL operation.

3.4.5.1 VCCSYN

This pin is the V_{CC} dedicated to the analog IMP PLL circuits. The voltage should be well regulated, and the pin should be provided with an extremely low-impedance path to the V_{CC} power rail. VCCSYN should be bypassed to GNDSYN by a 0.1-µF capacitor located as close as possible to the chip package.

3.4.5.2 GNDSYN

This pin is the GND dedicated to the analog IMP PLL circuits. The pin should be provided with an extremely low-impedance path to ground. GDNSYN should be bypassed to VCCSYN by a 0.1 µF capacitor located as close as possible to the chip package. The user should also bypass GNDSYN to VCCSYN with a 0.01 µF capacitor as close as possible to the chip package.

3.4.5.3 XFC

This pin connects to the off-chip capacitor for the PLL filter. One terminal of the capacitor is connected to XFC; the other terminal is connected to IQVCC.

3.4.5.4 MODCLK1–MODCLK0

MODCLK1–MODCLK0 specifies whether the IMP PLL is enabled and what the initial VCO frequency is after a hardware reset. During the assertion of RESET, the value of the MODCLK1–MODCLK0 input pins causes the PEN bit and the MF11–0 bits of the IMP PLL and Clock Control Register (IPLCR) $0F8 to be appropriately written. MODCLK1–MODCLK0 also determines if the oscillator's prescaler is used. After RESET is negated, the MODCLK1–MODCLK0 pins are ignored. These pins have an internal pullup during a hard-

ware reset. Table 3-1 shows the combinations of MODCLK1–0 pins with the corresponding default settings.

3.5 IMP POWER MANAGEMENT

The IMP portion of the MC68356 has several low power modes from which to choose. In addition, the IMP can control the DSP power management functions.

3.5.1 IMP Low Power Modes

The MC68356 provides a number of low power modes for the IMP section. Each of the operation modes has different current consumption, wake-up time, and functionality characteristics. The state of the IMP's 68000 data and address bus lines can be either driven high, low or tristated during low power stop mode by programming the low power drive control register (LPDCR).

NOTE

Current consumption for all operating modes is specified in Section 14 Electrical Characteristics.

Table 3-2 IMP Low Power Modes - IMP PLL Enabled

Operation Mode	Oscillator	PLL	Clock	Wake_Up (Osc. Clock Cycles)	Current Consumption (Approximate)	Method of Entry/LPM bits	IMP Functionality
STOP	Not Active	Not active	Not active	70000	<0.1mA	Stop instruction/ LPM1–0=11	No
DOZE	Active	Not active	Not active	2500	About 5mA	Stop instruction/ LPM1–0=10	No
STAND_BY	Active	Active (if enabled	Not active	2–5 cycles	About 5mA	Stop instruction/ LPM1–0=01	Partial (PCM-CIA)
SLOW_GO/ NORMAL	Active	Active (if enabled)	Active		Low, depends on CLK freq.	Write to DF3–0	Full

3.5.1.1 STOP Mode

In STOP mode, all parts of IMP are inactive and the current consumption is less than 0.1mA. Both the crystal oscillator and the IMP PLL are shut down. Because both the oscillator and the PLL must start up, the wake-up time takes 70000 EXTAL clocks (for example, 70000 cycles of 32.768KHz crystal will take about 2.2 sec).

The STOP mode is entered by executing the STOP instruction with the LPM0–1 bits in the IOMCR register set to 11. Refer to 3.5.2.2 Entering the STOP/ DOZE/ STAND_BY Mode for an example instruction sequence for use with the STOP instruction.

3.5.1.2 DOZE Mode

In DOZE mode, the oscillator is active in the IMP but the IMP PLL is shut down. The current consumption depends on the frequency of the external crystal but is on the order of a few mA. In DOZE mode, the IMP is shut down. The wake-up time is 2500 cycles of the external crystal (for example, 2500 cycles of 32.768 kHz crystal will take about 80 msec). Doze mode has faster wake-up time than the STOP mode, at the price of higher current consumption.

The DOZE mode is entered by executing the STOP instruction with the LPM1–0 bits in the IOMCR register set to 10. Refer to 3.5.2.2 Entering the STOP/ DOZE/ STAND_BY Mode for an example instruction sequence for use with the STOP instruction.

3.5.1.3 STAND_BY Mode

In STAND_BY mode, the oscillator is active, and the IMP PLL, if enabled, is active but the IMP clock is not active and the IMP is shut down. Current consumption in STAND-BY mode is less than 10mA (less than 5mA if the IMP PLL is not being used). The wake up time is a few IMP clock cycles.

The STAND_BY mode is entered by executing the STOP instruction with the LPM1–0 bits in the IOMCR register set to 01. Refer to 3.5.2.2 Entering the STOP/ DOZE/ STAND_BY Mode for an example instruction sequence for use with the STOP instruction.

In STAND-BY mode, the PCMCIA interface is active and any access on the PCMCIA interface will cause the IMP to come out of STAND-BY mode. The STAND_BY mode is useful in applications which have time slots with no activity, and require a minimal wake-up time before IMP returns to NORMAL or SLOW-GO mode.

3.5.1.4 SLOW_GO Mode

In the SLOW-GO mode, the IMP is fully operational but the IMP PLL divider has been programmed with a value that is dividing the IMP PLL VCO output to the system clock in order to save power. The PLL output divider can only be used with the IMP PLL enabled. The divider value is programmed in the DF3–0 bits in the IOMCR. The clock may be divided by a power of 2 ($2^0 - 2^{15}$). No functionality is lost in SLOW-GO mode.

3.5.1.5 NORMAL Mode

In NORMAL mode the IMP part is fully operational and the system clock from the PLL is not being divided down.

3.5.1.6 IMP Operation Mode Control Register (IOMCR)

IOMCR is a 8-bit read/ write register used to control the operation modes of the IMP. The WP bit in IOMCR is used as a protect mechanism to prevent erroneous writing of IOMCR.

IOMCR $0FA

7	6	5	4	3	2	1	0
IOMWP	DF3	DF2	DF1	DF0	BCD	LPM1	LPM0
RESET: 0	0	0	0	0	0	0	0

Read/Write

IOMWP—IMP Operation Mode Control Write Protect Bit

This bit prevents accidental writing into the IOMCR. After reset, this bit defaults to zero to enable writing. Setting this bit prevents further writing (excluding the first write that sets this bit).

DF 3–0—Divide Factor

The Divide Factor Bits define the divide factor of the low power divider of the PLL. These bits specify a divide range between 2^0 and 2^{15}. Changing the value of these bits will not cause a loss of lock condition to the IMP PLL.

BCD—BRG Clock Divide Control

This bit controls whether the divide-by-two block shown in Figure 3-1 is enabled.

0 = The BRG clock is divided by 2.
1 = The BRG clock is divided by 1.

LPM—Low Power Modes

When the 68000 core executes the STOP instruction, the IMP will enter the specified mode.

LPM1–0:

00 = Normal - the IMP PLL and clock oscillator will continue to operate normally.
01 = Stand_by Mode
10 = DOZE Mode
11 = Stop Mode

3.5.1.7 Low Power Drive Control Register (LPDCR)

This register controls the state of the IMP's 68000 address and data buses during the Stand-by, Doze, and Stop modes. By programming this register it is possible to minimize power consumption due to external pullups or pull downs, or floating inputs.

LPDCR BAR+$82A

7	6	5	4	3	2	1	0
						LPDL	LPDEN
RESET: 0	0	0	0	0	0	0	0

Read/Write

LPDEN—Low Power Drive Enable

0 - The IMP 68000 data and address buses will be high impedance.
1 - The IMP 68000 data and address buses to be driven according to the LPDL bit.

LPDL—Low Power Drive Data Low

0 - The data bus will be driven high when the LPDEN bit is set.
1 - The data bus will be driven low when the LPDEN bit is set.

3.5.1.8 Default Operation Modes, See 3.3.1 Default System Clock Generation.

3.5.2 Low Power Support

3.5.2.1 Enter the SLOW_GO mode

3

When the required IMP performance can be achieved with a lower clock rate, the user can reduce power consumption by dividing IMP PLL output clock that provides the IMP system clock. Switching between the NORMAL and SLOW_GO modes is achieved by changing the DF3–0 field in the IOMCR register to a non-zero value. The IMP PLL will not lose lock when the DF3–0 field in the IOMCR register is changed.

3.5.2.2 Entering the STOP/ DOZE/ STAND_BY Mode

Entering the STOP/ DOZE/ STAND_BY mode is achieved by the 68000 core executing the following code:

```
        nop
        move.b    *+6(PC),$000000FB        ;copy STOP operand high byte to addr 000000fb
        stop      #$xxxx                   ;xxxx -> SR
        nop
```

This code is position independent. The core must be in the supervisor state to execute the STOP instruction, therefore the write to $000000FB must be done in the supervisor state (function code 5, supervisor data). The core trace exception should be disabled, otherwise the low power control will not enter the STOP mode.

To guarantee supervisor state and trace exceptions disabled, this code should be part of a TRAP routine. Upon entering the trap routine, examine the stacked status register. If it indicates the supervisor state, then execute this code to enter STOP mode. If not supervisor, do NOT execute this code (could perform some application-specific error):

```
TRAP_x    btst.b    #5,(SP)              ; supervisor?
          beq.s     NO_STOP
          nop                            ; flush execution, bus pipes
          move.b    *+6(PC),$000000FB    ;copy STOP operand high byte to addr 000000fb
          stop      #$xxxx               ; xxxx -> SR
          nop
          rte
NO_STOP   ...                            ; error routine?
```

NOTE

The RI, DTE and PIT conditions will generate a level 4 interrupt. The PCMCIA will generate a level 1, 6, or 7 interrupt. The user should set the 68000 interrupt mask register to the appropriate level before executing this code.

IMP's low power control logic will:

1. Detect the write cycle.
2. Check if bit 5 = 1 (supervisor space) (if it is 0, the low power request will be ignored).
3. Sample the interrupt mask bits (bits 0–2). If during this process of stopping the clocks an interrupt of higher level than the mask is asserted to the core, this process will abort.
4. Wait for 16 clocks to guarantee the execution of the STOP command by the core. $\overline{BG}$ and $\overline{BGACK}$ will reset the 16-clock counter and it will restart its count.
5. Assert bus request signal to the core.
6. Wait for Bus Grant
7. Force the IMP to the selected power-down mode, as defined in Table 3-1.

3.5.2.3 IMP Wake-Up from Low Power STOP Modes

The IMP can wake up from STOP/DOZE/STAND_BY mode to NORMAL/SLOW_GO mode in response to inputs from the following sources:

1. Asserting both $\overline{RESET}$ and $\overline{HALT}$ (hard reset) pins.
2. Asserting (high to low transition) either $\overline{RI}$ (PB9) or PB10 pins (if these interrupts are enabled).
3. A timeout of the periodic interrupt timer (if the PIT interrupt is enabled).
4. Reset of the PwrDwn bit in the PCMCIA card configuration and status register when the PCMCIA controller is enabled.

When one of these events occur (and the corresponding event bit is set), the IMP low power controller will asynchronously restart the IMP clocks. Then IMP low power control logic will release the 68000 bus and the IMP will return to normal operation. If one of the above wake-up events occurs during the execution of the STOP command, the low power control logic will abort the power down sequence and return to normal operation.

If the IMP is in STAND_BY mode, any PCMCIA access will initiate a wake-up from STAND_BY mode (See 8.2 Power Down Options).

NOTE

The RI (PB9), DTE (PB10) and periodic interrupt timer timeout interrupts conditions will generate level 4 interrupts. The PCMCIA controller can be programmed to generate either level 1, 6, or 7 interrupts. The user should also set the 68000 interrupt mask in the status register (SR) to the appropriate level before executing the STOP command to ensure that the IMP will wake up to the desired events.

3.5.2.4 IMP Wake-Up Control Register – IWUCR

The IWUCR contains control for the wake-up options. This register can be read and written by the 68000 core.

IWUCR $0F7

7	6	5	4	3	2	1	0
0	PITE	PB10E	PB9Ev	0	PITEn	PB10En	PB9En
RESET: 0	0	0	0	0	0	0	0

Read/Write

PB9Ev—PB9 Event

This bit will be set to one when there is a high to low transition on the PB9 pin. When PB9En is set and PB9Ev is set, the IMP will wake-up from the selected power down state, and a PB9 Interrupt will be generated. The IMP cannot enter the power-down mode if PB9Ev and PB9En are both set to one. PB9Ev is cleared by writing a one (writing a zero has no effect).

In modem applications $\overline{RI}$ should be connected to the PB9 pin.

PB10Ev—PB10 Event

This bit will be set to one when there is a high to low transition on the PB10 pin. When PB10En is set and PB10Ev is set, the IMP will wake-up from the selected power down state, and the PB10 Interrupt will be generated. The IMP cannot enter the power-down mode when PB10Ev and PB10En are both set to one. PB10Ev is cleared by writing a one (writing a zero has no effect).

In modem applications the DTE TxD line may be connected to the PB10 pin.

PITEv—PIT Event

This bit will be set to one when there is a time-out on the periodic interrupt timer (PIT). When PITEn bit is set and a time-out occurs (PITEv is set), the IMP will wake-up from the selected power down, and a PIT Interrupt will be generated. The IMP cannot enter the power-down mode if PITEv and PITEn are both set to one. PITEv is cleared by writing a one (writing a zero has no effect).

PB9En—PB9 Enable

This bit, when set, enables the IMP to wake up from power down mode and generate an interrupt when the PB9 Event bit becomes set.

PB10En—PB10 Enable

This bit, when set, enables the IMP to wake up from power down mode and generate an interrupt when the PB10 Event bit becomes set.

PITEn—PIT Enable

This bit, when set, enables the IMP to wake up from power down mode and generate an interrupt when the PIT event bit becomes set, see 6.5.4 Periodic Interrupt Timer.

3.5.2.5 IMP Control of DSP Low Power Modes

The 68000 core can communicate with the DSP through either the host port connection or the DISC register to interrupt the DSP and have the DSP execute a routine to place itself in low power mode. See Section 11 DSP Host Port and 6.9.2 IMP-DSP Reset and Mode Interconnections Register.

3.6 DSP PLL CLOCK OSCILLATOR INTRODUCTION

The DSP56K family of processors features a PLL (phase-locked loop) clock oscillator in its central processing module. The DSP PLL allows the processor to operate at a high internal clock frequency using a low frequency clock input, a feature which offers two immediate benefits. Lower frequency clock inputs reduce the overall electromagnetic interference generated by a system, and the ability to oscillate at different frequencies reduces costs by eliminating the need to add additional oscillators to a system.

The DSP PLL performs frequency multiplication to allow the processor to use almost any available external system clock for full speed operation, while also supplying an output clock synchronized to a synthesized internal core clock. It also improves the synchronous timing of the processor's external memory port, significantly reducing the timing skew between DEXTAL and the internal chip phases. The DSP PLL provides a low power divider on its output, which can reduce or restore the chip operating frequency without losing the DSP PLL lock.

A DSP processor uses a four-phase clock for instruction execution which runs at the instruction execution rate. It can accept an external clock through the DEXTAL input, or it can run on an internal oscillator when the user connects an external crystal between DXTAL and DEXTAL. In addition to this, the DSP PLL in the MC68356 can receive an input clock signal from the IMP clock circuitry. During $\overline{\text{DRESET}}$ the clock source is selected by the CSelect pin. Refer to Table 3-1 which shows the strapping options on reset for initializing the DSP PLL.

3.7 PLL COMPONENTS

The DSP PLL block diagram is shown below in Figure 3-5. The components of the DSP PLL are described in the following sections.

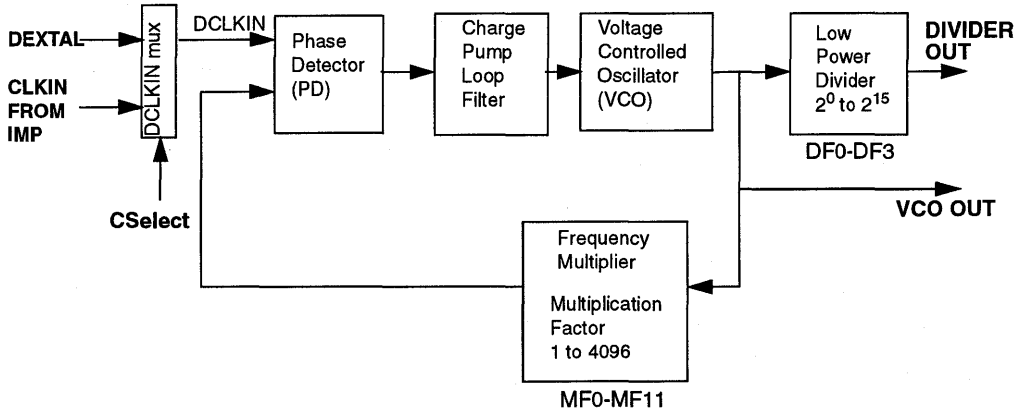


Figure 3-5. DSP PLL Block Diagram

3.7.1 DCLKIN Mux

The input clock to the DSP PLL or (the DSP itself if the PLL is disabled) is selected by the CLKIN mux which is controlled by the Cselect pin. The Cselect pin is sampled during the DSP reset. The output of the DCLKIN mux is the DCLKIN signal and will be referenced further in the remainder of this chapter.

3.7.2 Phase Detector and Charge Pump Loop Filter

The phase detector (PD) detects any phase difference between the external clock (DCLKIN) and an internal clock phase from the frequency multiplier. At the point where there is negligible phase difference and the frequency of the two inputs is identical, the DSP PLL is in the locked state. The phase detector on the IMP PLL operates in a similar way.

The charge pump loop filter receives signals from the PD, and either increases or decreases the phase based on the PD signals. An external capacitor is connected to the PCAP pin (described in Section 3.8) and determines the DSP PLL operation. (See Section 14 Electrical Characteristics for more detailed information on the DSP's phase and frequency.)

After the DSP PLL locks on to the proper phase/frequency, it reverts to the narrow bandwidth mode, which is useful for tracking small changes due to frequency drift of the DCLKIN clock.

3.7.3 Voltage Controlled Oscillator (VCO)

The VCO can oscillate at frequencies from the minimum speed specified in Section 14 Electrical Characteristics (typically10 MHz) up to the device's maximum allowed clock input frequency. The VCO on the IMP PLL operates in a similar way.

3.7.4 Frequency Multiplier

Inside the DSP PLL, the frequency multiplier divides the VCO output frequency by its division factor (n). If the frequency multiplier's output frequency is different from the DCLKIN frequency, the charge pump loop filter generates an error signal. The error signal causes the VCO to adjust its frequency until the two input signals to the phase detector have the same phase and frequency. At this point (phase lock) the VCO will be running at n times the DCLKIN frequency, where n is the multiplication factor for the frequency multiplier. The programmable multiplication factor ranges from 1 to 4096. The VCO on the IMP PLL operates in a similar way.

NOTE

If DSP PLL is enabled, the multiplication value must be large enough to result in the VCO clock being greater than 10MHz.

3.7.5 Low Power Divider (LPD)

The Low Power Divider (LPD) divides the output frequency of the VCO by any power of 2 from 2^0 to 2^{15}. Since the LPD is not in the closed loop of the DSP PLL, changes in the divide factor will not cause a loss of lock condition. This fact is particularly useful for utilizing the LPD in low power consumption modes when the chip is not involved in intensive calcula-

tions. This can result in significant power saving. When the chip is required to exit the low power mode, it can immediately do so with no time needed for clock recovery or DSP PLL lock.

3.7.6 DSP PLL Control Register (PCTL)

The DSP PLL control register (PCTL) is a 24-bit read/write register which directs the operation of the on-chip DSP PLL. It is mapped into the processor's internal X memory at X:$FFFD. The PCTL control bits are described in the following sections.

11	10	9	8	7	6	5	4	3	2	1	0
MF11	MF10	MF9	MF8	MF7	MF6	MF5	MF4	MF3	MF2	MF1	MF0

23	22	21	20	19	18	17	16	15	14	13	12
**	CKOS	CSRC	COD1	COD0	PEN	PSTP	XTLD	DF3	DF2	DF1	DF0

** Reserved bits, read as zero, should be written with zero for future compatibility.

Figure 3-6. DSP PLL Control Register (PCTL)

3.7.6.1 PCTL Multiplication Factor Bits (MF0–MF11) - Bits 0–11

The multiplication factor bits MF0–MF11 define the multiplication factor (MF) that will be applied to the DSP PLL input frequency. The MF can be any integer from 1 to 4096. Table 3-3 shows how to program the MF0–MF11 bits. The VCO will oscillate at a frequency of MF x F_{ext}, where F_{ext} is the DCLKIN clock frequency. The multiplication factor must be chosen to ensure that the resulting VCO output frequency will be between 10Mhz and the maximum allowable clock frequency (see Section 14 Electrical Characteristics). Any time a new value is written into the MF0–MF11 bits, the DSP PLL will lose the lock condition. After a time delay, the DSP PLL will relock.

Table 3-3 Multiplication Factor Bits MF0–MF1

MF11–MF0	Multiplication Factor MF
$000	1
$001	2
$002	3
Σ	Σ
Σ	Σ
$FFE	4095
$FFF	4096

3.7.6.2 PCTL Division Factor Bits (DF0–DF3) - Bits 12–15

The division factor bits DF0–DF3 define the divide factor (DF) of the low power divider. These bits specify any power of two divide factor in the range from 2^0 to 2^{15}. Table 3-4 shows the programming of the DF0–DF3 bits. Changing the value of the DF0–DF3 bits will

not cause a loss of lock condition. Whenever possible, changes of the operating frequency of the chip (for example, to enter a low power mode) should be made by changing the value of the DF0–DF3 bits rather than changing the MF0–MF11 bits. For MF≤4, changing DF0–DF3 may lengthen the instruction cycle following the PLL control register update; this is done in order to keep synchronization between DCLKIN and the internal chip clock. For MF>4 such synchronization is not guaranteed and the instruction cycle is not lengthened. Note that CKOUT is synchronized with the internal clock in all cases. The DF bits are cleared (division by one) by hardware reset.

Table 3-4 Division Factor Bits DF0–DF3

DF3–DF0	Division Factor DF
$0	2^0
$1	2^1
$2	2^2
Σ	Σ
Σ	Σ
$E	2^{14}
$F	2^{15}

3.7.6.3 PCTL DXTAL Disable Bit (XTLD) - Bit 16

The DXTAL disable (XTLD) bit controls the on-chip crystal oscillator DXTAL output. When XTLD is cleared, the XTAL output pin is active permitting normal operation of the crystal oscillator. Note that the Cselect pin must be strapped high during DSP reset to select the DSP clock source as the DEXTAL pin. When XTLD is set, the DXTAL output pin is held in the high (1) state, disabling the on-chip crystal oscillator. If the on-chip crystal oscillator is not used (EXTAL is driven from an external clock source), it is recommended that XTLD be set (disabling XTAL) to minimize RFI noise and power dissipation. The XTLD bit is cleared by hardware reset.

3.7.6.4 PCTL STOP Processing State Bit (PSTP) - Bit 17

The PSTP bit controls the behavior of the DSP PLL and of the on-chip crystal oscillator during the STOP processing state. When PSTP is set, the DSP PLL and the on-chip crystal oscillator will remain operating while the chip is in the STOP processing state, enabling rapid recovery from the STOP state. When PSTP is cleared, the DSP PLL and the on-chip crystal oscillator will be disabled when the chip enters the STOP processing. For minimal power consumption during the STOP state, at the cost of longer recovery time, PSTP should be cleared. To enable rapid recovery when exiting the STOP state, at the cost of higher power consumption in the STOP state, PSTP should be set. PSTP is cleared by hardware reset.

3.7.6.5 PCTL PLL Enable Bit (PEN) - Bit 18

The PEN bit enables the DSP PLL operation. When this bit is set, the DSP PLL is enabled and the internal clocks will be derived from the PLL VCO output. When this bit is cleared, the DSP PLL is disabled and the internal clocks are derived directly from the DCLKIN clock. When the DSP PLL is disabled, the VCO does not operate in order to minimize power consumption. The PEN bit may be set by software but it cannot be reset by software. During

hardware reset, this bit receives the value of the combination of the CSelect and MODCK1–0 pins as described in Table 3-1. The only way to clear PEN is to hold these pins in the right combination during hardware reset.

A relationship exists between PSTP and PEN where PEN adjusts PSTP's control of the DSP PLL operation. When PSTP is set and PEN (see Table 3-5) is cleared, the on-chip crystal oscillator remains operating in the STOP state, but the DSP PLL is disabled. This power saving feature enables rapid recovery from the STOP state when the user operates the chip with an on-chip oscillator and with the DSP PLL disabled.

Table 3-5 PSTP and PEN Relationship

		Operation During STOP			
PSTP	PEN	PLL	Oscillator	Recovery	Power Consumption
0	x	Disabled	Disabled	Long	Minimal
1	0	Disabled	Enabled	Rapid	Lower
1	1	Enabled	Enabled	Rapid	Higher

3.7.6.6 PCTL Clock Output Disable Bits (COD0–COD1) - Bits 19–20

The COD0-COD1 bits control the output buffer of the clock at the CKOUT pin. Table 3-6 specifies the effect of COD0-COD1 on the CKOUT pin. When both COD0 and COD1 are set, the CKOUT pin is held in the high (1) state. If the CKOUT pin is not connected to external circuits, it is recommended that both COD1 and COD0 be set (disabling clock output) to minimize RFI noise and power dissipation. If the CKOUT output is low at the moment the COD0–COD1 bits are set, it will complete the low cycle and then be disabled high. If the programmer re-enables the CKOUT output before it reaches the high logic level during the disabling process, the CKOUT operation will be unaffected. The COD0–COD1 bits are cleared by hardware reset.

Table 3-6 Clock Output Disable Bits COD0–COD1

COD1	COD0	CKOUT Pin
0	0	Clock Out Enabled, Full Strength Output Buffer
0	1	Clock Out Enabled, 2/3 Strength Output Buffer
1	0	Clock Out Enabled, 1/3 Strength Output Buffer
1	1	Clock Out Disabled

3.7.6.7 PCTL Chip Clock Source Bit (CSRC) - Bit 21

The CSRC bit specifies whether the clock for the chip is taken from the output of the VCO or is taken from the output of the low power divider (LPD). When CSRC is set, the clock for the chip is taken from the VCO. When CSRC is cleared, the clock for the chip is taken from the output of the LPD. See 3.9.8 CKOUT Considerations for restrictions. CSRC is cleared by hardware reset.

3.7.6.8 PCTL CKOUT Clock Source Bit (CKOS) - Bit 22

The CKOS bit specifies whether the CKOUT clock output is taken from the output of the VCO or is taken from the output of the LPD. When CKOS is set, the CKOUT clock output is

3.7.6.8 PCTL CKOUT Clock Source Bit (CKOS) - Bit 22

The CKOS bit specifies whether the CKOUT clock output is taken from the output of the VCO or is taken from the output of the LPD. When CKOS is set, the CKOUT clock output is taken from the VCO. When CKOS is cleared, the CKOUT clock output is taken from the output of the LPD. If the DSP PLL is disabled (PEN=0), CKOUT is taken from DCLKIN. See 3.9.8 CKOUT Considerations. CKOS is cleared by hardware reset.

3.7.6.9 PCTL Reserved Bit - Bit 23

This bit is reserved for future expansion. It reads as zero and should be written with zero for future compatibility.

3.8 DSP PLL PINS

The following pins are dedicated to the DSP PLL operation:

PVCC—VCC Dedicated to the Analog DSP PLL Circuits.

The voltage should be well regulated and the pin should be provided with an extremely low impedance path to the VCC power rail. PVCC should be bypassed to PGND by a 0.1µF capacitor located as close as possible to the chip package.

PGND—GND Dedicated to the Analog PLL Circuits.

The pin should be provided with an extremely low impedance path to ground. PVCC should be bypassed to PGND by a 0.1µF capacitor located as close as possible to the chip package.

PCAP—Off-Chip Capacitor for the PLL Filter.

One terminal of the capacitor is connected to PCAP while the other terminal is connected to PVCC. The capacitor value is specified in Section 14 Electrical Characteristics.

CKOUT—Clock Out.

This output pin provides a 50% duty cycle output clock synchronized to the internal processor clock when the DSP PLL is enabled and locked. When the DSP PLL is disabled, the output clock at CKOUT is derived from, and has the same frequency and duty cycle as DCLKIN.

NOTE

If the DSP PLL is enabled and the multiplication factor is less than or equal to 4, then CKOUT is synchronized to DCLKIN.

3.9 DSP PLL OPERATION CONSIDERATIONS

The following paragraphs discuss DSP PLL operation considerations.

3.9.1 Operating Frequency

The operating frequency of the chip is governed by the frequency control bits in the DSP PLL control register as follows:

$$'F_{CHIP} = \frac{'F_{EXT} \times MF}{DF} = \frac{Fvco}{DF}$$

where: DF is the division factor defined by the DF0-DF3 bits

F_{CHIP} is the chip operating frequency

F_{EXT} is the external input frequency to the chip at the DCLKIN pin

F_{VCO} is the output frequency of the VCO

MF is the multiplication factor defined by the MF0-MF11 bits

The chip frequency is derived from the output of the low power divider. If the low power divider is bypassed, the equation is the same but the division factor should be assumed to be equal to one.

3.9.2 Hardware Reset

Hardware reset causes the initialization of the DSP PLL. The following considerations apply:

1. The MF0–MF11 bits in the PCTL register are set to their pre-determined hardware reset value (400) if Cselect=0 and MODCLK1–0=11. The DF0–DF3 bits and the chip clock source bit in the PCTL register are cleared. The DCLKIN mux will be set to route the IMP CLKIN source to the DSP PLL. This causes the chip clock frequency to be equal to the external input frequency (DCLKIN) multiplied by the multiplication factor defined by MF0–MF11(401). All other combinations of Cselect and MODCLK1–0 will result in the PLL being disabled on DSP reset.
2. During hardware reset assertion, the PEN bit in the PCTL register is set only if Cselect=0 and MODCLK1–0=11. If the PEN is set, the DSP PLL acquires the proper phase/frequency. While hardware reset is asserted, the internal chip clock will be driven by the DCLKIN source until the DSP PLL achieves lock (if enabled). If PEN is cleared, the DSP PLL is deactivated and the internal chip clock is driven by the DCLKIN source.
3. Proper operation of the DSP PLL cannot be guaranteed for VCO out frequencies of less than 10 MHz. If the resulting VCO clock frequency would be less than the minimum and the user wishes to operate with the DSP PLL enabled, the user should set the MODCLK pins and the Cselect pins to an acceptable combination that will cause the DSP PLL to be disabled initially after DSP reset (see Table 3-1). Then, after reset, the user must issue an instruction which loads the PCTL control register with a multiplication factor that would bring the VCO frequency above 10 MHz and would enable the DSP PLL operation. Until this instruction is executed, the DSP PLL is disabled, which may cause a large skew (<15nsec) between the external input clock and the internal processor clock. If internal low frequency of operation is desired with the DSP PLL enabled, the VCO output frequency may be divided down by using the internal

low power divider.

3.9.3 Operation with DSP PLL Disabled

If the DSP PLL is disabled, the internal chip clock and CKOUT are driven from the DCLKIN input.

3.9.4 Changing the MF0–MF11 Bits

Changes to the MF0–MF11 bits cause the following to occur:

1. The DSP PLL will lose the lock condition.
2. The DSP PLL acquires the proper phase/frequency. Until this occurs, the internal chip clock phases will be frozen. This ensures that the clock used by the chip is a clock that has reached a stable frequency.
3. When lock occurs, the DSP PLL drives the internal chip clock and CKOUT.
4. While DSP PLL has not locked, CKOUT is held low.

3.9.5 Change of DF0–DF3 Bits

Changes to the DF0–DF3 bits do not cause a loss of lock condition. The internal clocks will immediately revert to the frequency prescribed by the new divide factor. For MF≤4, changing DF0–DF3 may lengthen the instruction cycle or CKOUT pulse following the DSP PLL control register update in order to keep synchronization between DCLKIN and the internal chip clock. (Here, T3 is equal to the phase described by the new divide factor plus the time required to wait for a synchronizing pulse, which is less than 1.5ETc.) For MF>4, such synchronization is not guaranteed and the instruction cycle is not lengthened.

If the DF0–DF3 bits are changed by the same instruction that changes the MF0–MF11 bits, the LPD divider factor changes before the detection of the change in the multiplication factor. This means that the detection of loss of lock will occur after the LPD has started dividing by the new division factor.

3.9.6 Loss of Lock

The DSP PLL distinguishes between cases where MF>4 and cases where MF≤4. If MF≤4, the DSP PLL will detect loss of lock if a skew of 2.5 to 4.5 ns develops between the two clock inputs to the phase detector.

If MF>4, the DSP PLL will detect loss of lock when there is a discrepancy of one clock cycle between the two clock inputs to the phase detector.

3.9.7 STOP Processing State

If the PSTP bit is cleared, executing the STOP instruction will disable the on-chip crystal oscillator and the DSP PLL. In this state the chip consumes the least possible power. When recovering from the STOP state, the recovery time will be 16 or 64k external clock cycles (according to bit 6 in the operating mode register) plus the time needed for the DSP PLL to achieve lock.

programmer re-enables the CKOUT clock output before it reaches the high logic level during the disabling process, the CKOUT operation will be unaffected.

While the DSP PLL is regaining lock, the CKOUT clock output remains at the same logic level it held when the DSP PLL lost lock, which is when the clocks were frozen in the DSP.

When the chip enters the WAIT processing state, the core phases are disabled but CKOUT continues to operate. When DSP PLL is disabled, CKOUT will be fed from DCLKIN.

If DF>1 and CKOS≠CSRC, then the programmer must change either CKOS or CSRC before taking any action that causes the DSP PLL to lose and subsequently regain lock, such as changing the multiplication factor, enabling DSP PLL operation, or recovering from the STOP state with PSTP=0.

Any change of the CKOS or CSRC bits must be done while DF=1.

3.9.9 Synchronization Among DCLKIN, CKOUT, and the Internal Clock

Low clock skew between DCLKIN and CKOUT is guaranteed only if MF≤4. The synchronization between CKOUT and the internal chip activity and Port A timing is guaranteed in all cases where CKOS=CSRC and the bits have never differed from one another.

3.9.10 DSP Low Power Modes

The DSP has two different low power modes, the WAIT mode and the STOP mode.To bring the DSP into low power operating states, the WAIT instruction or the STOP instruction is executed. The difference between the WAIT and STOP states is that in the wait state the oscillator is still running. In the stop state the oscillator is gated off, reducing power even more.

In the WAIT mode the DSP system clock is disconnected to all internal circuitry except the internal peripherals. All internal processing is halted until one of the following occurs:

- An unmasked interrupt is detected from an internal peripheral or IRQ pin.
- The Debug Request pin of the OnCE is asserted
- The DSP is reset.

In the STOP mode all the internal peripherals are kept in their reset states and the clock oscillator is gated off. One of the following actions will wake up the DSP:

- A low level on the IRQA pin.
- A low level on the $\overline{\text{RESET}}$ pin.
- A low level on the DR pin.

Either of these actions will activate the oscillator. After a clock stabilization delay the processor and peripherals will be reneabled.

More information about stabilization delay is given in Section 14 Electrical Characteristics.

More information on wake up timing, both from wait and stop mode is discussed in the DSP family manual referring to processing states.

- A low level on the $\overline{\text{RESET}}$ pin.
- A low level on the DR pin.

Either of these actions will activate the oscillator. After a clock stabilization delay the processor and peripherals will be reneabled.

More information about stabilization delay is given in Section 14 Electrical Characteristics.

More information on wake up timing, both from wait and stop mode is discussed in the DSP family manual referring to processing states.

SECTION 4
MC68000/MC68008 CORE

The IMP 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 IMP 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 IMP, 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.

4.1 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 4-1, 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.

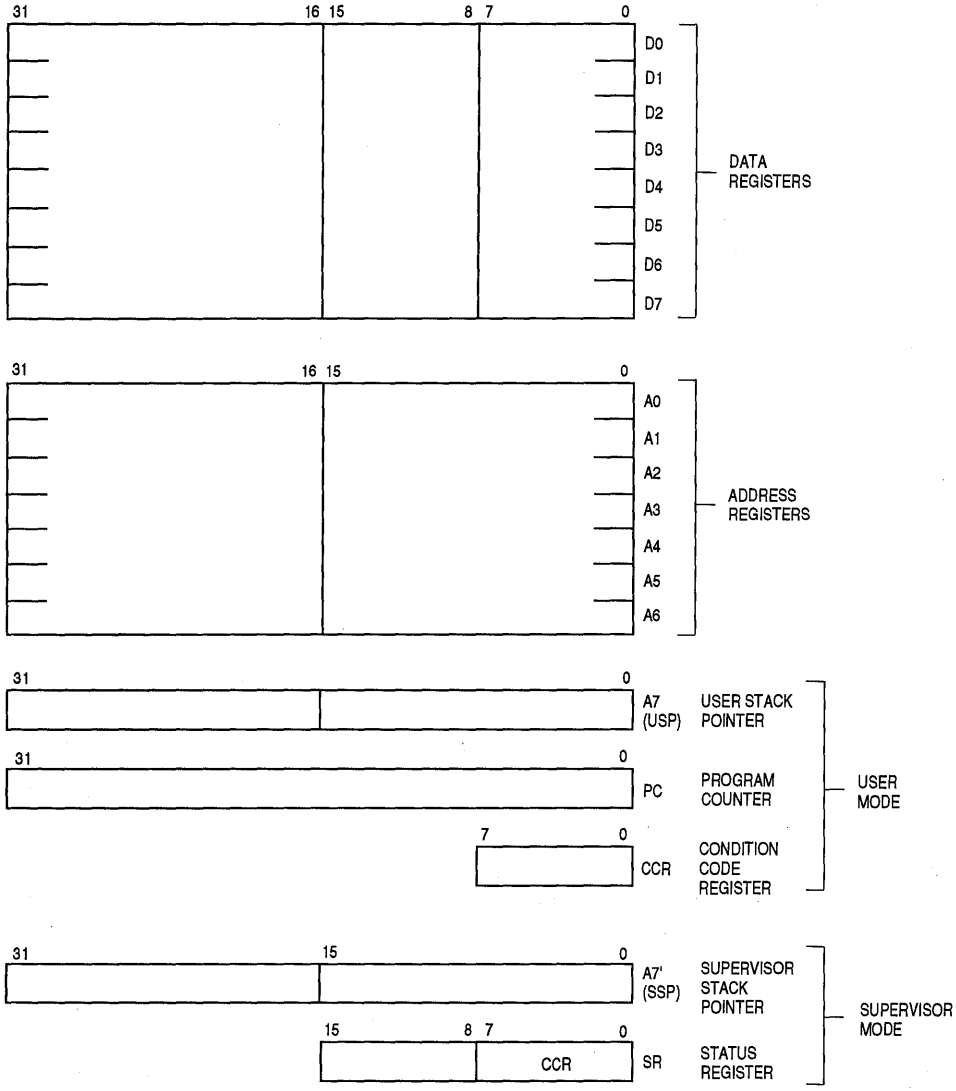


Figure 4-1. M68000 Programming Model

The supervisor's programming model includes supplementary registers, including the supervisor stack pointer (SSP) and the status register (SR) as shown in Figure 4-2. 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.

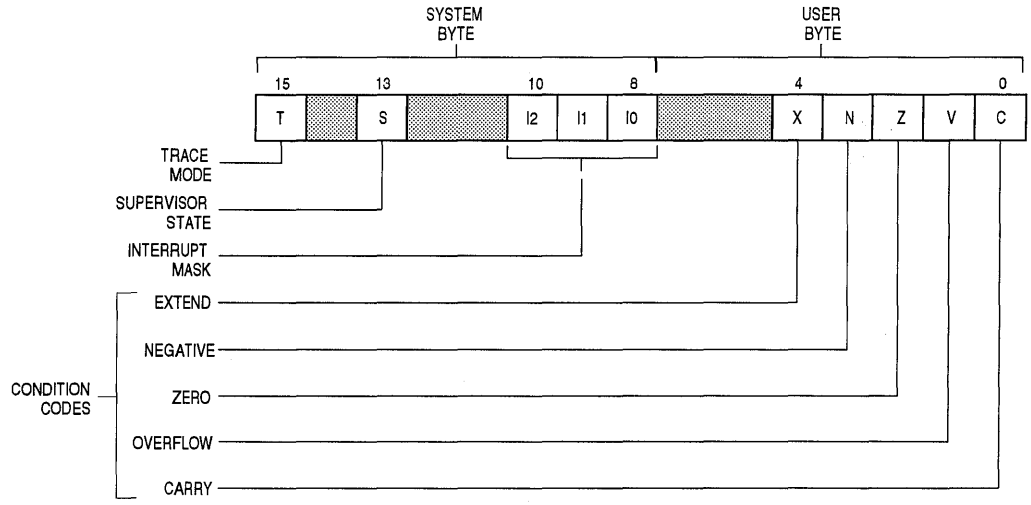


Figure 4-2. M68000 Status Register

4.2 INSTRUCTION SET SUMMARY

The five data types supported by the M68000 on the IMP 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 4-1, 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 4-2.

Some basic instructions also have variations as shown in Table 4-3.

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 4-1. 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 4-2. 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 4-3. M68000 Instruction Type Variations

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

4.3 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 4-4.

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 IMP, this capability is provided for each potential internal bus master (i.e., the IDMA, SDMA, and DRAM refresh units). Also on the IMP, 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 4-4. 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).

4.4 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.

4.4.1 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 4-5.

Table 4-5. 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 4-5. M68000 Exception Vector Assignment

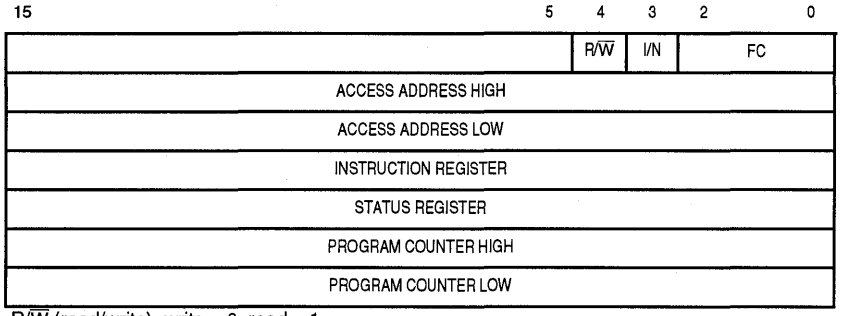
11	44	02C	SD	Line 1111 Emulator
12[1]	48	030	SD	(Unassigned, Reserved)
13[1]	52	034	SD	(Unassigned, Reserved)
14[1]	56	038	SD	(Unassigned, Reserved)
15	60	03C	SD	Uninitialized Interrupt Vector
16–23[1]	64	040	SD	(Unassigned, Reserved)
	92	05C	SD	
24	96	060	SD	Spurious Interrupt[3]
25	100	064	SD	Level 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 IMP (see 5.1 IMP Configuration 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.

4.4.2 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 4-5. A bus error or address error causes the M68000 to store the information shown in Figure 4-3. The interrupts, traps, illegal instructions, and trace stack frames are shown in Figure 4-4.



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

Figure 4-3. M68000 Bus/Address Error Exception Stack Frame

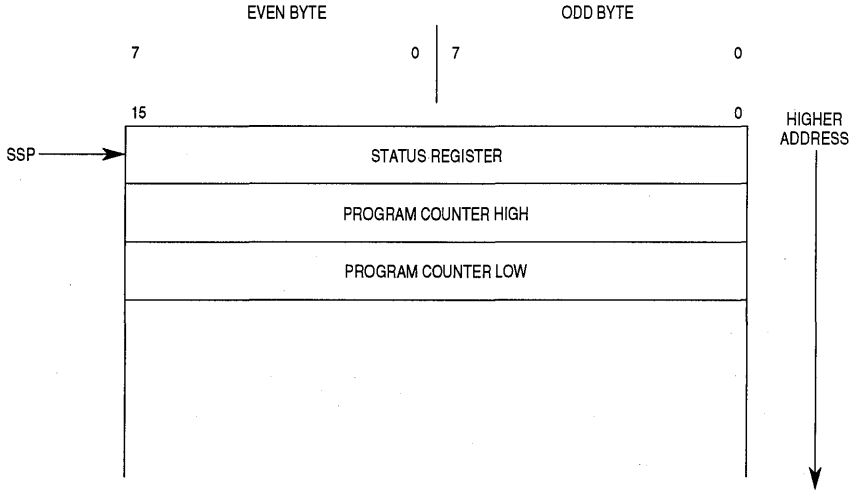


Figure 4-4. M68000 Short-Form Exception Stack Frame

NOTE

The IMP 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.*

4.5 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 4-2).

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.

4.6 M68000 SIGNAL DIFFERENCES

The MC68000 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 IMP 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 IMP 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 6.7.3 System Control Bits).

Two M68000 signals are omitted from the IMP: 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 IMP as $\overline{\text{AVEC}}$ to direct the core to use an autovector during interrupt acknowledge cycles.

SECTION 5
MEMORY MAP

This section includes tables that show the registers of the IMP portion of the MC68356. All of the registers are memory mapped into the 68000 space.

5.1 IMP CONFIGURATION CONTROL

Four reserved entries in the external M68000 exception vector table (see Table 4-5) are used as addresses for internal system configuration registers. These entries are at locations $0F0, $0F4, $0F8, and $0FC. See Table 5-1, System Configuration Registers.

The BAR entry contains the BAR described in this section. The SCR entry contains the SCR described in Section 6 System Integration Block (SIB).

Figure 5-1 shows all the IMP on-chip addressable locations and how they are mapped into system memory.

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).

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 CCR 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.

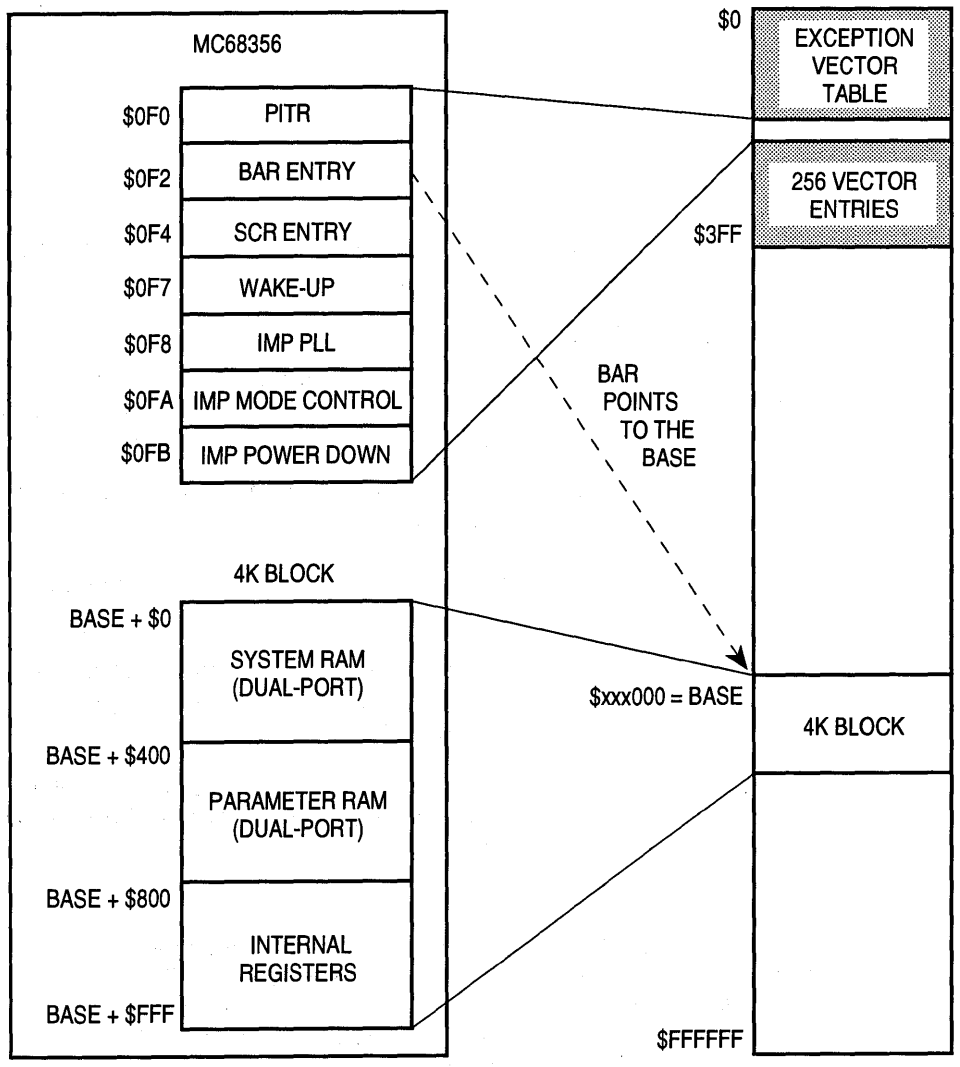
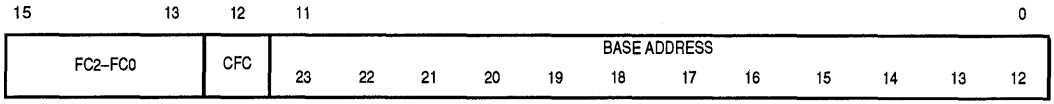


Figure 5-1. IMP Configuration Control

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.



Bits 15–13—FC2–FC0

Bits 15–13 of the BAR contain the FC2–FC0 field. 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.

5.2 SYSTEM CONFIGURATION REGISTERS

Four entries in the M68000 exception vectors table (located in low RAM) are reserved for the addresses of system configuration registers (see Table 5-1). These registers have seven addresses within $0F0-$0FF. The MC68356 uses one of the IMP 32-bit reserved spaces for 3 registers added for the MC68356. These registers are used to control the PLL, clock generation and low power modes. See Section 3 Clock Generation and Low Power Control for more details.

Table 5-1. System Configuration Registers

Address	Name	Width	Description	Reset Value
$0F0	PITR	16	Periodic Interrupt Timer Register	0000
$0F2	BAR	16	Base Address Register	BFFF
$0F4	SCR	24	System Control Register	0000 0F
$0F7	IWUCR	8	IMP Wake-Up Control Register	00
$0F8	IPLCR	16	IMP PLL Control Register	
$0FA	IOMCR	8	IMP Operations Mode Control Register	00
$0FB	IPDR	8	IMP Power Down Register	00
$0FC	RES	32	Reserved	

The internal 1176-byte dual-port RAM has 576 bytes of system RAM (see Table 5-2) and 576 bytes of parameter RAM (see Table 5-3).

Table 5-2. 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, the16550 channel, 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 Section 7 Communications Processor (CP). Base + 67E contains the MC68356 revision number.

Table 5-3. 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

Table 5-3. Parameter RAM

Base + 4C0 • • • Base + 4FF			Reserved (Not Implemented)
Base + 500	4 Word	SCC2, 16550	Rx BD 0
Base + 508	4 Word	SCC2, 16550	Rx BD 1
Base + 510	4 Word	SCC2, 16550	Rx BD 2
Base + 518	4 Word	SCC2, 16550	Rx BD 3
Base + 520	4 Word	SCC2, 16550	Rx BD 4
Base + 528	4 Word	SCC2, 16550	Rx BD 5
Base + 530	4 Word	SCC2, 16550	Rx BD 6
Base + 538	4 Word	SCC2, 16550	Rx BD 7
Base + 540	4 Word	SCC2, 16550	Tx BD 0
Base + 548	4 Word	SCC2, 16550	Tx BD 1
Base + 550	4 Word	SCC2, 16550	Tx BD 2
Base + 558	4 Word	SCC2, 16550	Tx BD 3
Base + 560	4 Word	SCC2, 16550	Tx BD 4
Base + 568	4 Word	SCC2, 16550	Tx BD 5
Base + 570	4 Word	SCC2, 16550	Tx BD 6/DRAM Refresh
Base + 578	4 Word	SCC2, 16550	Tx BD 7/DRAM Refresh
Base + 580 • • • Base + 5BF		SCC2 SCC2	Specific Protocol Parameters
Base + 5C0 • • • Base + 5FF			Reserved (Not Implemented)
Base + 600	4 Word	SCC3	Rx BD 0
Base + 608	4 Word	SCC3	Rx BD 1
Base + 610	4 Word	SCC3	Rx BD 2
Base + 618	4 Word	SCC3	Rx BD 3
Base + 620	4 Word	SCC3	Rx BD 4
Base + 628	4 Word	SCC3	Rx BD 5
Base + 630	4 Word	SCC3	Rx BD 6
Base + 638	4 Word	SCC3	Rx BD 7
Base + 640	4 Word	SCC3	Tx BD 0
Base + 648	4 Word	SCC3	Tx BD 1
Base + 650	4 Word	SCC3	Tx BD 2
Base + 658	4 Word	SCC3	Tx BD 3 ##
Base + 660	3 Word	SMC	Reserved
Base + 666	Word	SMC1	Rx BD
Base + 668	Word	SMC1	Tx BD
Base + 66A	Word	SMC2	Rx BD
Base + 66C	Word	SMC2	Tx BD
Base + 66E #	6 Word	SMC1–SMC2	Internal Use
Base + 67A	Word	SCP	Rx/Tx BD
Base + 67C	Word	SCC1–SCC3	BERR Channel Number
Base + 67E #	Word	CP	MC68356 Revision Number

Table 5-3. Parameter RAM

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. (7.5.6Buffer 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 5-4) are memory-mapped registers offset from the BAR pointer 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.

5.3 INTERNAL REGISTERS MAP

Table 5-4. Internal Registers Map

Address	Name	Width	Block	Description	Reset Value
68356 Base + 800	RES	16	IDMA	Reserved	0000
68356 Base + 802	CMR	16	IDMA	Channel Mode Register	XXXX XXXX
68356 Base + 804	SAPR	32	IDMA	Source Address Pointer	XXXX XXXX
68356 Base + 808	DAPR	32	IDMA	Destination Address Pointer	XXXX
68356 Base + 80C	BCR	16	IDMA	Byte Count Register	00
! 68356 Base + 80E	CSR	8	IDMA	Channel Status Register	
68356 Base + 80F	RES	8	IDMA	Reserved	
68356 Base + 810	FCR	8	IDMA	Function Code Register	XX
68356 Base + 811	RES	8	IDMA	Reserved	
68356 Base + 812 #	GIMR	16	Int Cont	Global Interrupt Mode Register	0000
! 68356 Base + 814	IPR	16	Int Cont	Interrupt Pending Register	0000
68356 Base + 816	IMR	16	Int Cont	Interrupt Mask Register	0000
! 68356 Base + 818	ISR	16	Int Cont	In-Service Register	0000
68356 Base + 81A	RES	16	Int Cont	Reserved	
68356 Base + 81C	RES	16	Int Cont	Reserved	
68356 Base + 81E #	PACNT	16	PIO	Port A Control Register	0000
68356 Base + 820 #	PADDR	16	PIO	Port A Data Direction Register	0000
68356 Base + 822 #	PADAT	16	PIO	Port A Data Register	XXXX ##
68356 Base + 824 #	PBCNT	16	PIO	Port B Control Register	0080
68356 Base + 826 #	PBDDR	16	PIO	Port B Data Direction Register	0000
68356 Base + 828 #	PBDAT	16	PIO	Port B Data Register	XXXX ##
68356 Base + 82A	LPDCR	8	PIO	Low Power Drive Control Register	00
68356 Base + 82C	PPR	8	CS	PCMCIA protection register	0000
68356 Base + 82E	RES	16	CS	Reserved	
68356 Base + 830 #	BR0	16	CS0	Base Register 0	C001
68356 Base + 832 #	OR0	16	CS0	Option Register 0	DFFD
68356 Base + 834 #	BR1	16	CS1	Base Register 1	C000
68356 Base + 836 #	OR1	16	CS1	Option Register 1	DFFD

Table 5-4. Internal Registers Map

Address	Name	Width	Block	Description	Reset Value
68356 Base + 838 #	BR2	16	CS2	Base Register 2	C000
68356 Base + 83A #	OR2	16	CS2	Option Register 2	DFFD
68356 Base + 83C #	BR3	16	CS3	Base Register 3	C000
68356 Base + 83E #	OR3	16	CS3	Option Register 3	DFFD
68356 Base + 840	TMR1	16	Timer	Timer Unit 1 Mode Register	0000
68356 Base + 842	TRR1	16	Timer	Timer Unit 1 Reference Register	FFFF
68356 Base + 844	TCR1	16	Timer	Timer Unit 1 Capture Register	0000
68356 Base + 846	TCN1	16	Timer	Timer Unit 1 Counter	0000
68356 Base + 848	RES	8	Timer	Reserved	
! 68356 Base + 849	TER1	8	Timer	Timer Unit 1 Event Register	00
68356 Base + 84A	WRR	16	WD	Watchdog Reference Register	FFFF
68356 Base + 84C	WCN	16	WD	Watchdog Counter	0000
68356 Base + 84E	RES	16	Timer	Reserved	
68356 Base + 850	TMR2	16	Timer	Timer Unit 2 Mode Register	0000
68356 Base + 852	TRR2	16	Timer	Timer Unit 2 Reference Register	FFFF
68356 Base + 854	TCR2	16	Timer	Timer Unit 2 Capture Register	0000
68356 Base + 856	TCN2	16	Timer	Timer Unit 2 Counter	0000
68356 Base + 858	RES		Timer	Reserved	
! 68356 Base + 859	TER2	8	Timer	Timer Unit 2 Event Register	00
68356 Base + 85A	RES	16	Timer	Reserved	
68356 Base + 85C	RES	16	Timer	Reserved	
68356 Base + 85E	RES	16	Timer	Reserved	
68356 Base + 860	CR	8	CP	Command Register	00
68356 Base + 861 68356 Base + 86F	Reserved				
68356 Base + 870	COR	8	Int Cont	Configuration Option Register	00
68356 Base + 871	CCSR	8	Int Cont	Card Configuration and Status Register	00
68356 Base + 872	PRR	8	Int Cont	Pin Replacement Register	00
68356 Base + 873	PSCR	8	Int Cont	Socket and Copy Register	XX

Table 5-4. Internal Registers Map

Address	Name	Width	Block	Description	Reset Value
68356 Base + 874	IOEIR	8	Int Cont	I/O Event Indication register	00
68356 Base + 875	RES1	8		Reserved1	00
68356 Base + 876	RES2	8		Reserved2	00
68356 Base + 877	RES3	8		Reserved3	00
68356 Base + 878 68356 Base + 87F	Reserved				
68356 Base + 880	RES	16	SCC1	Reserved	
68356 Base + 882	SCON1	16	SCC1	SCC1 Configuration Register	0004
68356 Base + 884	SCM1	16	SCC1	SCC1 Mode Register	0000
68356 Base + 886	DSR1	16	SCC1	SCC1 Data Sync. Register	7E7E
! 68356 Base + 888	SCCE1	8	SCC1	SCC1 Event Register	00
68356 Base + 889	RES	8	SCC1	Reserved	
68356 Base + 88A	SCCM1	8	SCC1	SCC1 Mask Register	00
68356 Base + 88B	RES	8	SCC1	Reserved	
68356 Base + 88C	SCCS1	8	SCC1	SCC1 Status Register	00
68356 Base + 88D	RES	16	SCC1	Reserved	
68356 Base + 88E	RES	16	SCC1	Reserved	
68356 Base + 890	RES	16	SCC2	Reserved	
68356 Base + 892	SCON2	16	SCC2	SCC2 Configuration Register	0004
68356 Base + 894	SCM2, EMR	16	SCC2 16550	SCC2,16550 Mode Register	0000
68356 Base + 896	DSR2	16	SCC2 16550	SCC2 Data Sync. Register	7E7E
! 68356 Base + 898	SCCE2, E550	16	SCC2 16550	SCC2,16550 Emulation Event Register	0000
68356 Base + 899	RES	8	SCC2	Reserved	
68356 Base + 89A	SCCM2, M550	8	SCC2 16550	SCC2,16550 Mask Register	00
68356 Base + 89B	RES	8	SCC2	Reserved	
68356 Base + 89C	SCCS2, S550	16	SCC2 16550	SCC2,16550 Emulation Status Register	0000
68356 Base + 89D	RES	16	SCC2	Reserved	

Table 5-4. Internal Registers Map

Address	Name	Width	Block	Description	Reset Value
68356 Base + 89E	RES	16	SCC2	Reserved	
68356 Base + 89A	M550	16	SCC2 16550	16550 Emulation Mask Register	0000
68356 Base + 8A0	RES	16	SCC3	Reserved	
68356 Base + 8A2	SCON3	16	SCC3	SCC3 Configuration Register	0004
68356 Base + 8A4	SCM3	16	SCC3	SCC3 Mode Register	0000
68356 Base + 8A6	DSR3	16	SCC3	SCC3 Data Sync. Register	7E7E
! 68356 Base + 8A8	SCCE3	8	SCC3	SCC3 Event Register	00
68356 Base + 8A9	RES	8	SCC3	Reserved	
68356 Base + 8AA	SCCM3	8	SCC3	SCC3 Mask Register	00
68356 Base + 8AB	RES	8	SCC3	Reserved	
68356 Base + 8AC	SCCS3	8	SCC3	SCC3 Status Register	00
68356 Base + 8AD	RES	8	SCC3	Reserved	
68356 Base + 8AE	RES	16	SCC3	Reserved	
68356 Base + 8B0	SPMODE	16	SCM	SCP, SMC Mode and Clock Control Register	0000
68356 Base + 8B2 #	SIMASK	16	SI	Serial Interface Mask Register	FFFF
68356 Base + 8B4 #	SIMODE	16	SI	Serial Interface Mode Register	0000
68356 Base + 8B6 68356 Base + 8BF	Reserved				
68356 Base + 8C0	PCMR	24	PCMCIA	PCMCIA Mode Register	000000
68356 Base + 8C4!	PCRWER	8	PCMCIA	PCMCIA Configuration Registers Write Event Register	00
! 68356 Base + 8C5!	PCAWER	8	PCMCIA	PCMCIA Access Wake-up Event Register	00
68356 Base + 8C6	PCRWMR	8	PCMCIA	PCMCIA Configuration Registers Write Mask Register	00
68356 Base + 8C7	PCAWMR	8	PCMCIA	PCMCIA Access Wake-up Mask Register	00
! 68356 Base + 8C8!	PCHER	8	PCMCIA	PCMCIA Host (PC) Event Register	00
68356 Base + 8CC	CISBAR	16	PCMCIA	CIS Base Address Register	0000
68356 Base + 8D0	CMBAR1	16	PCMCIA	Common Memory Space Base Address Register1	0000

Table 5-4. Internal Registers Map

Address	Name	Width	Block	Description	Reset Value
68356 Base + 8D4	CMBAR2	16	PCMCIA	Common Memory Space Base Address Register2	0000
68356 Base + 8D8 68356 Base + 8DF	Reserved				
68356 Base + 8E0	PUCR	8		Pull-Up Control Register	00
68356 Base + 8E2	PCDDR	8		Port C Data Direction Register	00
68356 Base + 8E4	PCDAT	8		Port C Data Register	XX##
68356 Base + 8E6	PDDAT	16		Port D Data Register	XXXX*
68356 Base + 8E8	HBAR	16		DSP Host Port Base Address Register	0000
68356 Base + 8EA	HCOR	8		DSP Host Port Configuration Option Register	00
68356 Base + 8EC	Reserved				
68356 Base + 8EE	DISC	16		DSP Interconnection and Serial Connections Register	0000
68356 Base + 8F0	EMR	16	16550	16550 Emulation Mode Register	0000
68356 Base + 8F2	LCR	8	16550	16550 Emulation Line Control Register	00
68356 Base + 8F3	MSR	8	16550	16550 Emulation Modem Status Register	00
68356 Base + 8F4	DLM	8	16550	16550 Emulation Divisor Latch (MSB)	XX
68356 Base + 8F5	DLL	8	16550	16550 Emulation Divisor Latch (LSB)	XX
68356 Base + 8F6	LSR	8	16550	16550 Emulation Line Status Register	00
68356 Base + 8F7 68356 Base + FFF				Reserved (Not implemented)	

Reset only upon total system reset. (RESET and 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 5.4 Event Registers).

5.4 EVENT REGISTERS

The IMP contains a few special registers that are 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, SCCE3 in SCC3, the PCRWER, PCAWER, PCHER in the PCMCIA controller and the E550 in the 16550 controller. 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 6
SYSTEM INTEGRATION BLOCK (SIB)

The MC68356 contains an extensive SIB that simplifies the job of both the hardware and software designer. Most of the features are taken from the MC68302 without change, features that have been added are highlighted in **bold** text.

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, their associated wait-state logic and the addition of the new **WE and OE signals** allow a glueless interface to SRAM, EPROM, flash EPROM, and EEPROM. The **four** 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: DREQ, DACK, and DONE
- Interrupt Controller with Two Modes of Operation
- **More** Parallel Input/Output (I/O) Ports, Some with Interrupt Capability
- **Pullup Resistors on Parallel Input and PCMCIA Input Pins**
- On-Chip 1152-Byte Dual-Port RAM
- Four Timers Including a Watchdog Timer and **Periodic Interrupt Timer**
- Four Programmable Chip-Select Lines with Wait-State Generator Logic
- **Glueless Interface to SRAM, EPROM, Flash EPROM, and EEPROM**
- **IMP Internal Control of DSP Resets, Interrupts, and Modes (Disk)**
- 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
  - —DRAM Refresh Controller

6.1 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 7.2 SDMA Channels. The IDMA is discussed in the following paragraphs.

6.1.1 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 IMP 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 6-1.

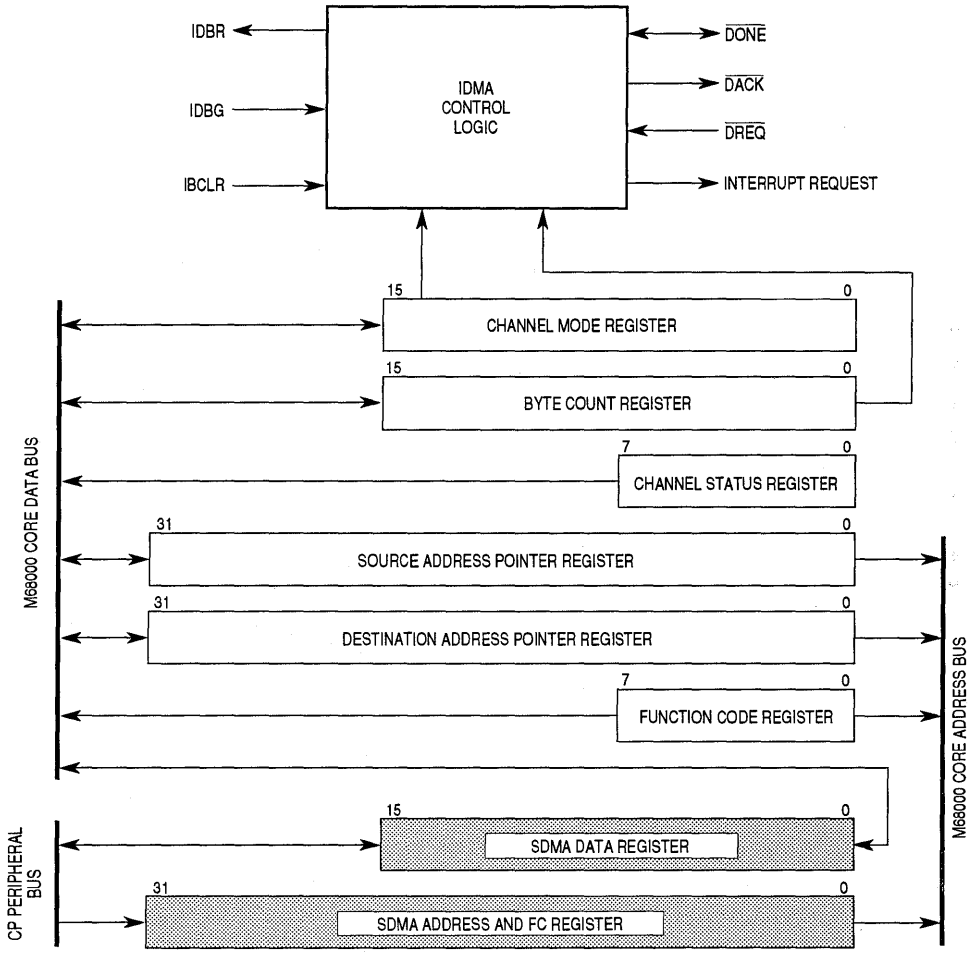


Figure 6-1. IDMA Controller Block Diagram

6.1.2 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.

6.1.2.1 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

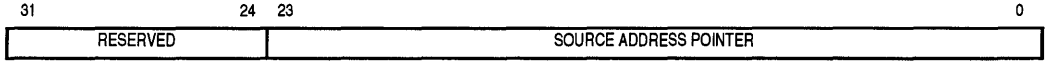
This bit starts the IDMA transfer if the REQG bits are programmed for an internal request. (The IDMA begins requesting the M68000 bus one clock after STR is set.) If the REQG bits are programmed for an external request, this bit must be set before the IDMA will recognize the first request on the $\overline{\text{DREQ}}$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

6.1.2.2 Source Address Pointer Register (SAPR)



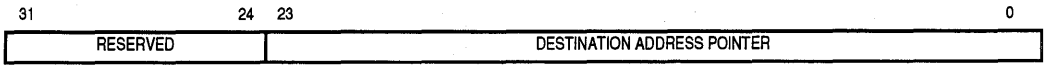
The SAPR is a 32-bit register.

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.

6.1.2.3 Destination Address Pointer Register (DAPR)

The DAPR is a 32-bit register.



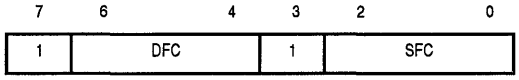
The DAPR contains 24 (A23–A0) address bits of the destination operand used by the IDMA to access memory or memory-mapped peripheral controller registers. During the IDMA write cycle, the address on the master address bus is driven from this register. The DAPR may

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.

6.1.2.4 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

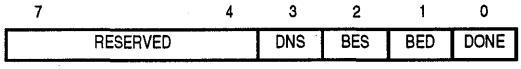
This register is undefined following power-on reset. The user should always initialize it and should not use the function code value "111" in this register.

6.1.2.5 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.

6.1.2.6 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.

6.1.3 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 6.1.6 DMA Bus Arbitration. The peripheral used with these signals may be either a source or a destination of the transfers.

6.1.3.1 $\overline{\text{DREQ}}$ 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.

6.1.3.2 $\overline{\text{DONE}}$

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.

6.1.4 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.

6.1.4.1 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 6.1.2 IDMA Registers (Independent DMA Controller) and 6.1.5 IDMA Programming for further details.

6.1.4.2 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 6.1.2.2 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 6.1.2.3 Destination Address Pointer Register (DAPR) and 6.1.2.5 Byte Count Register (BCR) for more details.

6.1.4.3 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 6-1 to see how the SAPR and DAPR will be incremented in all combinations.

Table 6-1. 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 —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 6-2 for more details on the IDMA bus cycles.

Table 6-2. IDMA Bus Cycles

Source Size	Source Address	Destination Size	Destination Size	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.

6.1.4.4 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 6.1.6 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 per-

centage 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.

6.1.4.5 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.

6.1.5 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.

6.1.6 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 6-11). 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 $\overline{\text{BGACK}}$.

NOTE

The PCMCIA Controller asserts internal Bus Clear to the IDMA if enabled in PCMR register (See section 6).

Internally generated IDMA requests are affected by a mechanism supported to reduce the M68000 core interrupt latency and external bus master arbitration latency (see 6.7.5 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.

6.1.7 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 provi-

sions 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 7.5.9.4 Bus Error on SDMA Access for further details.

6.1.7.1 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.

6.1.7.2 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.

6.1.7.3 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.

6.1.7.4 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.

6.2 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 6-2.

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

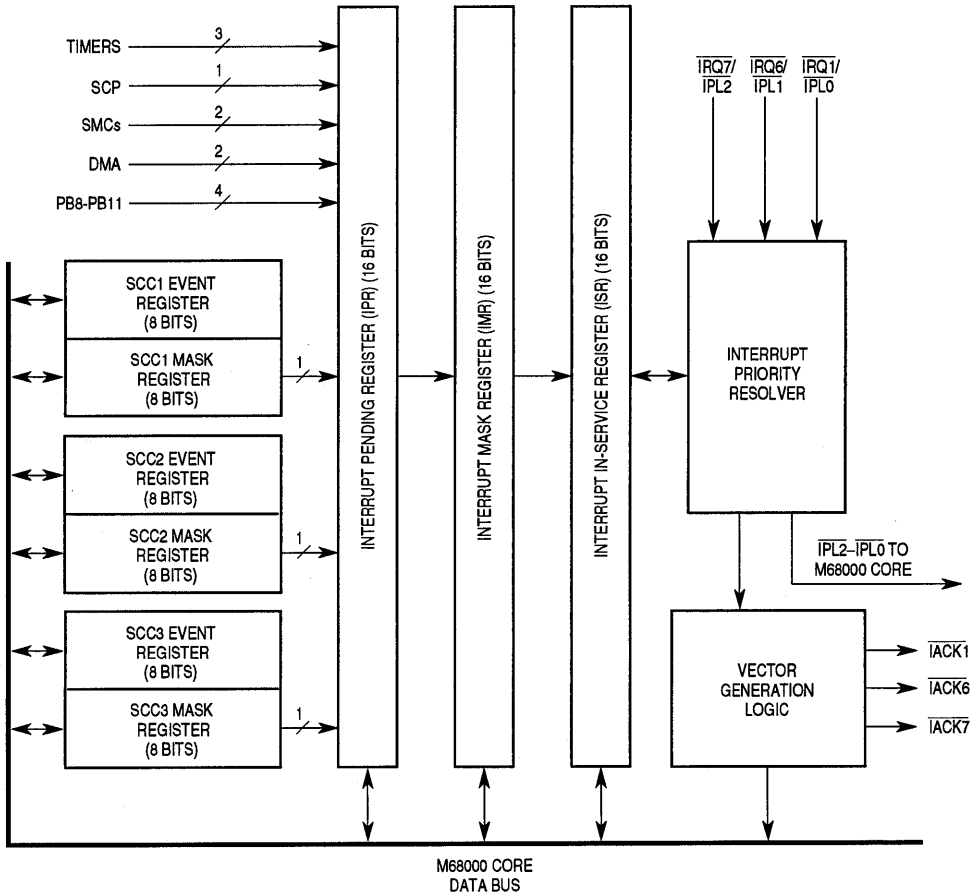


Figure 6-2. Interrupt Controller Block Diagram

6.2.1 Overview

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

6.2.1.1 IMP Interrupt Processing Overview

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

1. The M68000 responds to the interrupt request by executing an interrupt acknowledge bus cycle after the completion of the current instruction.
2. The interrupt controller recognizes the interrupt acknowledge cycle and places the interrupt vector for that interrupt request onto the M68000 bus.
3. 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 4.5 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. $\overline{UDS}$ and $\overline{LDS}$ 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 4-5), 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.

6.2.1.2 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 above mentioned internal sources, the interrupt controller receives interrupts from the PCMCIA Controller and from the DSP host port connection. These interrupts are additional sources to $\overline{IRQ7}$, $\overline{IRQ6}$, and $\overline{IRQ1}$. (programmable in PCMR and DCOR registers). With PCMCIA and DSP Host Interrupts it is possible to work in Normal or Dedicated mode for the three external interrupt requests pins. The chip ORs all sources at the same

level and asserts the interrupt to the 68000 core. Interrupts at the same level (1,6 or 7) are not prioritized.

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 $\overline{IPL2}$–$\overline{IPL0}$ lines. In the dedicated mode, EXRQ interrupts are presented directly as $\overline{IRQ7}$, $\overline{IRQ6}$, and $\overline{IRQ1}$.

Normal Mode

In this mode, the three external interrupt request pins are configured as $\overline{IPL2}$–$\overline{IPL0}$ 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 or the DSP host port connection 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 $\overline{IACK1}$). In this mode, if DSP host port interrupt acknowledge 1,6 or 7 is enabled, the DSP host port provides the vector and asserts $\overline{DTACK}$, and no $\overline{IACK7}$, $\overline{IACK6}$ or $\overline{IACK1}$ is generated. If DSP host port interrupt is not enabled for the relevant level external, $\overline{IACK7}$, $\overline{IACK6}$ or $\overline{IACK1}$ outputs are provided. It is recommended to have only one interrupt source at each level.

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.

If the interrupt source is the PCMCIA controller, only the first and third options are available since the controller cannot provide a vector.

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 6.7 System Control).

6.2.2 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.

6.2.2.1 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 6-3 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 6-3. 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	IRQ7	EXRQ
6	001	IRQ6	EXRQ
5	010	*	EXRQ
4	*	*	INRQ
3	100	*	EXRQ
2	101	*	EXRQ
1 (Lowest)	110	IRQ1	EXRQ

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

6.2.2.2 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 6-4. 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.

6.2.2.3 Nested Interrupts

Table 6-4. 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	—

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.

6.2.3 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 6.2.5.2 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 6-3). Table 6-4 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.

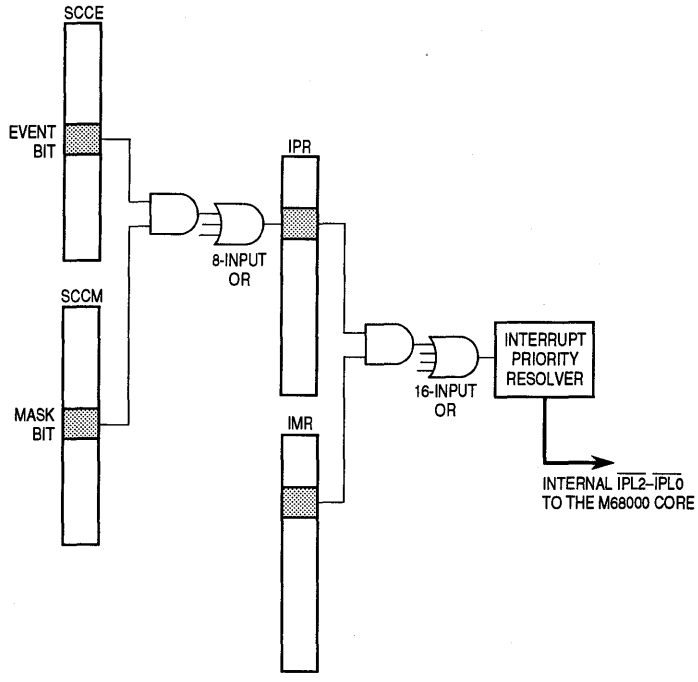


Figure 6-3. Interrupt Request Logic Diagram for SCCs

6.2.4 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 6-5. An example vector calculation is shown in Figure 6-4. 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 6-5. 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

6.2.5 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.

6.2.5.1 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

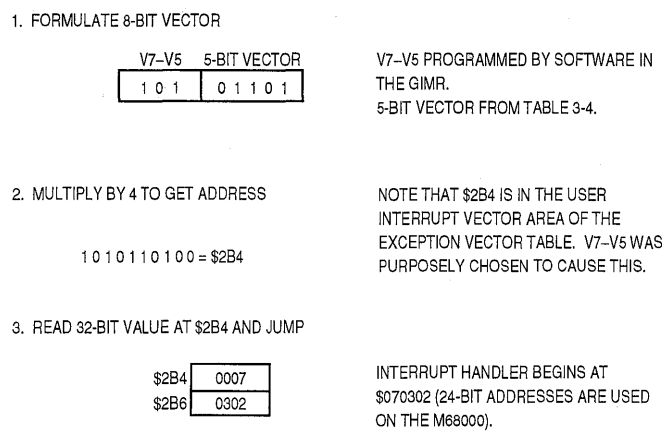


Figure 6-4. SCC1 Vector Calculation Example

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. For the host port, if the interrupt acknowledge for level 7 is enabled in the HCOR register, the external vector will be provided by DSP host port internally (see 11.2.3.4 Interrupt Vector Register (IVR)).

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. For the host port, if the interrupt acknowledge for level 6 is enabled in the HCOR register, the external vector will be provided by DSP host port internally (see 11.2.3.4 Interrupt Vector Register (IVR)).

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. For the host port, if the interrupt acknowledge for level 1is enabled in the HCOR register, the external vector will be provided by DSP host port internally (see 11.2.3.4 Interrupt Vector Register (IVR)).

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 IMP is used in the disable CPU mode. This situation is as follows:

For a slave-mode IMP, 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 IMP 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 $\overline{DTACK}$) according to the IV6 bit. The $\overline{IACK6}$ falling edge may be used externally to negate $\overline{IRQ6}$.

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

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

This bit is valid only in the dedicated mode.

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

6.2.5.2 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

6.2.5.3 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	—

6.2.5.4 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 6.2.2.3 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

6.2.6 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.

6.3 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.

6.3.1 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, and the PCMCIA interface is not enabled or the PCMCIA data bus is only 8 bits wide. Port A pins are configured as dedicated on-chip peripheral pins if the corresponding PACNT bit is set. An example block diagram of PA8 is given in Figure 6-5.

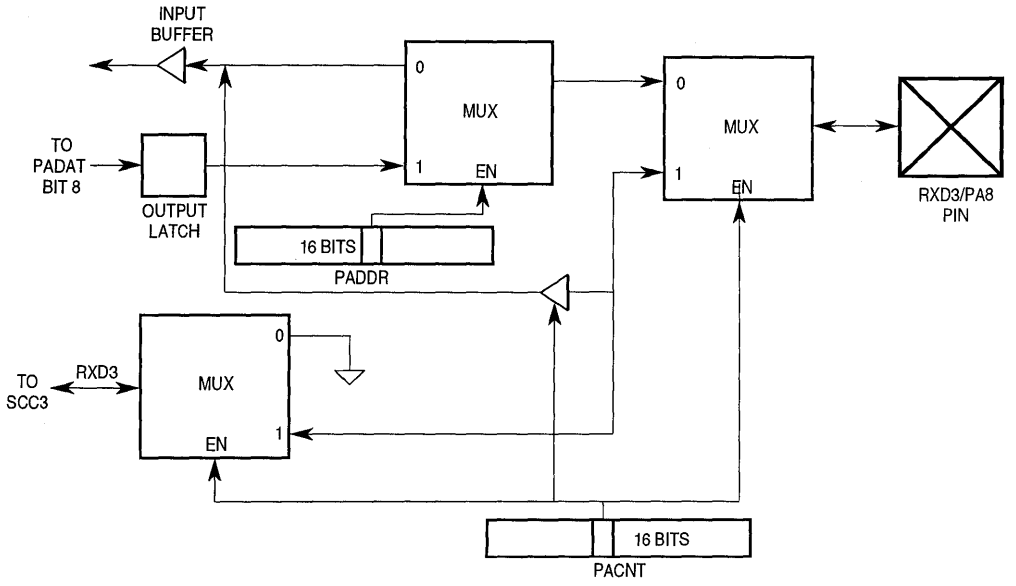


Figure 6-5. Parallel I/O Block Diagram for PA8

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 for non-PCMCIA mode).

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, $\overline{CD2}$, $\overline{CTS2}$, and $\overline{RTS2}$) 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 6-6. 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 6-6. Port A Pin Functions

PACNT Bit = 1 Pin Function	PACNT Bit = 0 Pin Function	Input to SCC2/SCC3/IDMA
RXD2	PA0	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.

6.3.2 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.

6.3.2.1 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 ($\overline{\text{WDOG}}$) upon total system reset.

Table 6-7 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 6-7. 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	—

6.3.2.2 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. 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. Internal pull-up resistors are optionally available for PB0-3, PB5, PB6 and PB8-PB11 pins (see 8.3.3 Pullup Control Register – PUCR).

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 6-4.) 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 Section 14 Electrical Characteristics.

6.3.3 Port C

When the PCMCIA interface is disabled (PC_EN tied to GND), 8 general purpose I/O pins are available.

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 C data direction register (PCDDR). The port I/O pin is configured as an input if the corresponding PCDDR bit is cleared; it is configured as an output if the corresponding PCDDR bit is set. All PCDDR bits are cleared on total system reset, configuring all port C pins as general-purpose input pins.

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

6.3.4 Port D

When the PCMCIA interface is disabled (PC_EN tied to GND), 16 general purpose input only pins are available as port D.These pins also have optional internal pull-up resistors (see 8.3.3 Pullup Control Register – PUCR).

Port D may be accessed through the Port D data register (PCDAT). When PCDAT is read, the state of the input port pin is read.

6.3.5 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 6-6. Parallel I/O Port Registers 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

Port C Data Direction Register(PCDDR)

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

0 = Input 1 = Output

Port C Data Register(PCDAT)

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

Port D Data Register(PDDAT)

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

Figure 6-6. Parallel I/O Port Registers

6.4 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, the PCMCIA controller, the DSP to IMP direct access block or an external master. The M68000 core and the IDMA access the RAM synchronously 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 6-7. 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 $\overline{\text{DTACK}}$ signal output when the RAM and on-chip registers are accessed by any M68000 bus master.

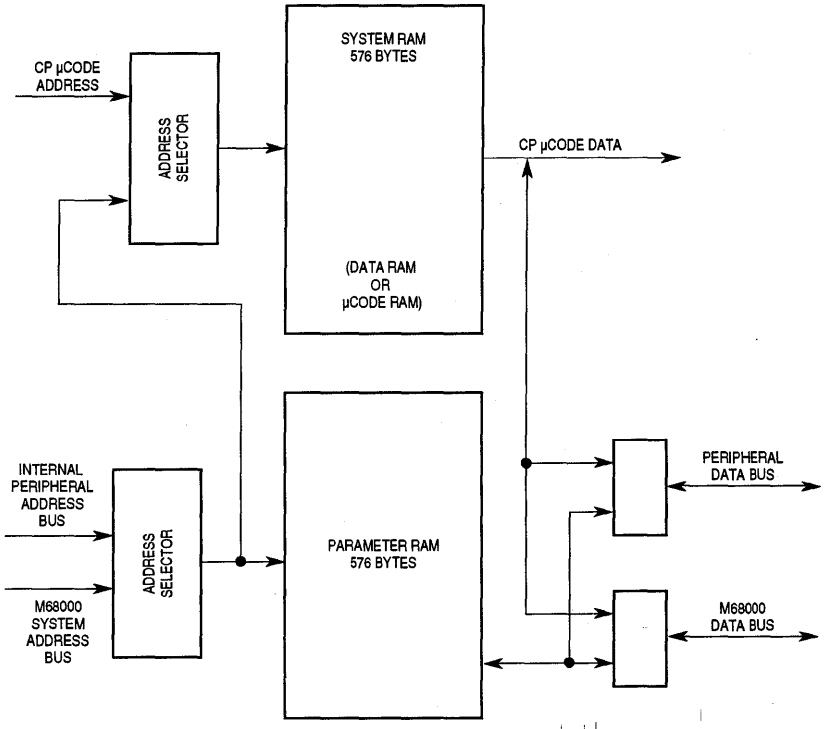


Figure 6-7. RAM Block Diagram

6.5 TIMERS

The IMP includes four timer units: a periodic interrupt timer (PIT), 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 6-8.

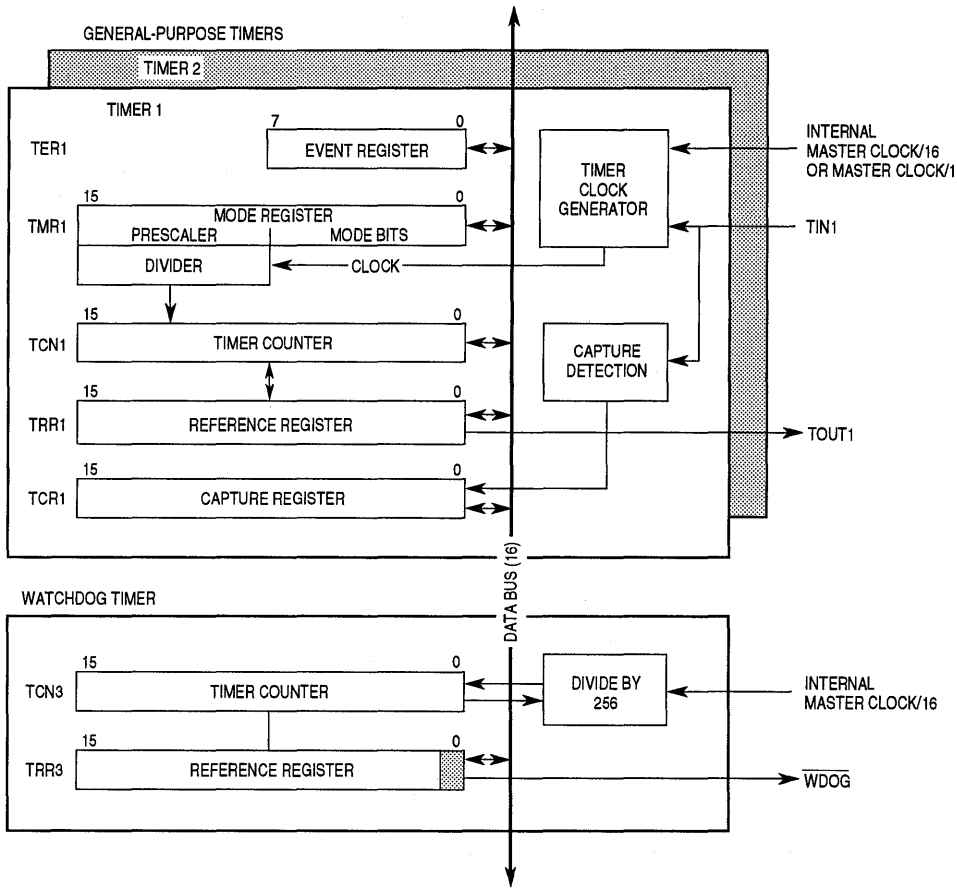


Figure 6-8. Timer Block Diagram

6.5.1 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

6.5.2 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 reset on the next clock or continue 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.

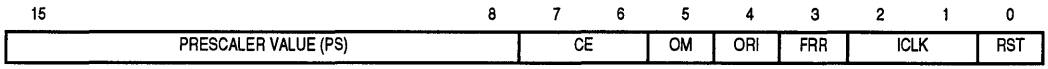
Each timer may output a signal on the timer output ($\overline{\text{TOUT1}}$ or $\overline{\text{TOUT2}}$) pin when the reference value is reached, as selected by the output mode (OM) bit of the corresponding TMR. This signal can be an active-low pulse or a toggle of the current output. The output can 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.

6.5.2.1 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.



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 res-

olution 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 6.5.2.6 General Purpose Timer Example).

6.5.2.2 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.

6.5.2.3 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.

6.5.2.4 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.

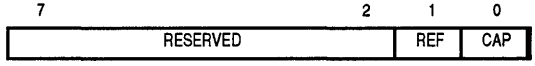
When working in the MC68008 mode (BUSW is low), reading the high byte of TCN1 and TCN2 will latch the low byte into a temporary register; a subsequent read cycle on the low byte yields the value of the temporary register.

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.

6.5.2.5 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.

6.5.2.6 General Purpose Timer Example

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

6.5.2.6.1 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.

6.5.2.6.2 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. Since when OM is set, the TOUT pin only toggles when the reference value is reached, 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 (= 5000/2).
3. Program the TMR to $321B (prescaler = 3, OM =1 to toggle TOUTt, 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.

6.5.3 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.

6.5.3.1 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.

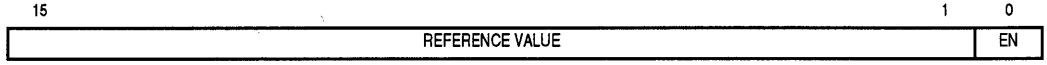
The software watchdog timer has an 8-bit prescaler that is not accessible to the user, a read-only 16-bit counter, and a reference register (WRR).

6.5.3.2 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.



6.5.3.3 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.

6.5.4 Periodic Interrupt Timer

The 68356 IMP provides a timer to generate periodic interrupts for use with a real-time operating system or the application software. The periodic interrupt time period can vary from 122 μs to 128 s (assuming a 32.768-kHz crystal is used to generate the general system clock). This function can be disabled.

6.5.4.1 Overview

The periodic interrupt timer consists of an 11-bit modulus counter that is loaded with the value contained in the PITR. The modulus counter is clocked by the CLKIN signal derived from the IMP EXTAL pin. See Figure 3-1.

The clock source is divided by four before driving the modulus counter (PITCLK). When the modulus counter value reaches zero, an interrupt request signal is generated to the IMP interrupt controller.

The value of bits 11–1 in the PITR is then loaded again into the modulus counter, and the counting process starts over. A new value can be written to the PITR only when the PIT is disabled.

The PIT interrupt replaces the IMP PB8 interrupt and is mapped to the PB8 interrupt priority level 4. The PIT interrupt is not maskable —the only way to disable the PIT interrupt is to disable the PIT by resetting the PTEN bit in the PITR register to zero.

NOTE

When the PIT is enabled, PB8 can still be used as parallel I/O pin or as DRAM refresh controller request pin, but PB8 will not be capable of generating interrupts.

6.5.4.2 Periodic Timer Period Calculation

The period of the periodic timer can be calculated using the following equation:

$$\text{periodic interrupt timer period} = \frac{\text{PITR count value+1}}{\frac{((EXTAL)/1 or 512)}{(4)}}$$

Solving the equation using a crystal frequency of 32.768 kHz with the prescaler disabled gives:

$$\text{periodic interrupt timer period} = \frac{\text{PITR count value+1}}{\frac{32768/1}{2^2}}$$

$$\text{periodic interrupt timer period} = \frac{\text{PITR counvalue}}{8192}$$

This gives a range from 122 µs, with a PITR value of $0, to 250 ms, with a PITR value of $7FF (assuming a 32.768Khz at the EXTAL pin.

Solving the equation with the prescaler enabled (PTP=1) gives the following values:

$$\text{periodic interrupt timer period} = \frac{\text{PITR count value}}{\frac{32768/512}{2^2}}$$

$$\text{periodic interrupt timer period} = \frac{\text{PITR count value}}{16}$$

This gives a range from 62.5 ms, with a PITR value of $0 to 128 s, with a PITR value of $7FF.

For a fast calculation of periodic timer period using a 32.768-kHz crystal, the following equations can be used:

With prescaler disabled:

programmable interrupt timer period = PITR (122 µs)

With prescaler enabled:

programmable interrupt timer period = PITR (62.5 ms)

6.5.4.3 Using the Periodic Timer As a Real-Time Clock

The periodic interrupt timer can be used as a real-time clock interrupt by setting it up to generate an interrupt with a one-second period. When using a 32.768-kHz crystal, the PITR should be loaded with a value of $0F with the prescaler enabled to generate interrupts at a one-second rate. The interrupt is generated, in this case, at a precise 1 second rate, even if the interrupt is not serviced immediately. A true real time clock is obtained if the current interrupt is serviced completely before the next one occurs.

6.5.4.4 Periodic Interrupt Timer Register (PITR)

The PITR contains control for prescaling the periodic timer as well as the count value for the periodic timer. This register can be read or written only during normal operational mode. Bits 14–13 are not implemented and always return a zero when read. A write does not affect these bits.

PITR $0F0

15	14	13	12	11	10	9	8
PTEN	0	0	PTP	PITR10	PITR9	PITR8	PITR7
RESET							
0	0	0	0	0	0	0	0

7	6	5	4	3	2	1	0
PITR6	PITR5	PITR4	PITR3	PITR2	PITR1	PITR0	RES
RESET							
0	0	0	0	0	0	0	0

Read/Write

PTEN—Periodic Timer Enable

This bit contains the enable control for the periodic timer.

0 = Periodic timer is disabled
1 = Periodic timer is enabled

PTP—Periodic Timer Prescaler Control

This bit contains the prescaler control for the periodic timer.

0 = Periodic timer clock is not prescaled
1 = Periodic timer clock is prescaled by a value of 512

PITR10–0—Periodic Interrupt Timer Register Bits

These bits of the PITR contain the remaining bits of the PITR count value for the periodic timer. **These bits may be written only when the PIT is disabled (PTEN=0) to modify the PIT count value.**

NOTE

If the PIT is enabled with the PTP bit is set, the first interrupt can be up to 512 clocks early, depending on the prescaler counter value when the PIT is enabled.

6.6 EXTERNAL CHIP-SELECT SIGNALS AND WAIT-STATE LOGIC

The IMP 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 6-9.

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 14-15) 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 (6.7.3 System Control Bits).

Each chip select can be protected to block PCMCIA accesses to that chip select's block of memory by the external PCMCIA host. The protect for each chip select block can be enabled by setting the PCS(3-0) bits in the PCMCIA protection register (PPR). In addition, internal IMP registers and dual port RAM access from the PCMCIA bus can be disabled by using the PIR bit in the PPR.

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

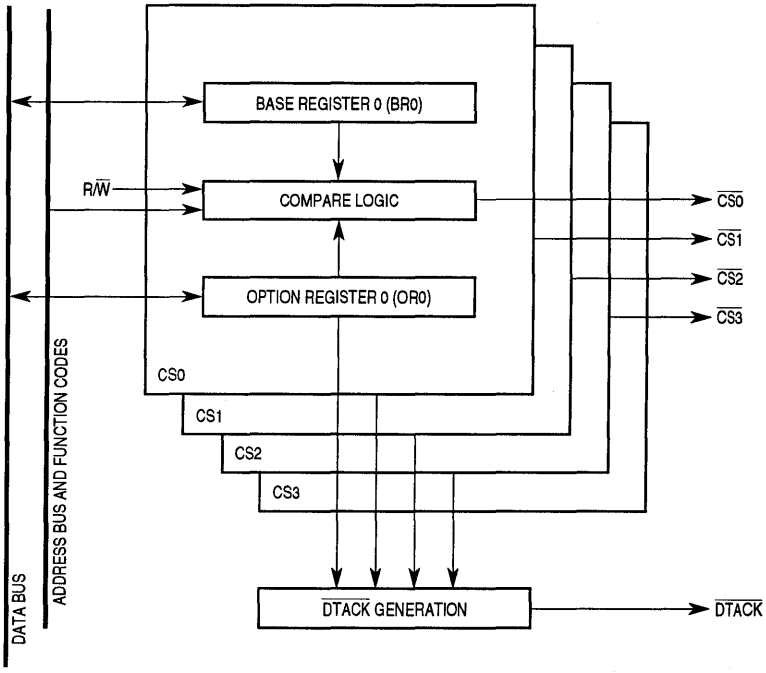


Figure 6-9. Chip-Select Block Diagram

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 $\overline{BERR}$ if address decode conflict enable (ADCE) is set. Only one chip-select line will be driven because of internal line priorities. $\overline{CS0}$ has the highest priority, and $\overline{CS3}$ the lowest. $\overline{BERR}$ 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 $\overline{BERR}$ if write protect violation enable (WPVE) is set. The $\overline{CS}$ line will not be asserted.

NOTE

The chip-select logic is reset only on total system reset (assertion of $\overline{RESET}$ and $\overline{HALT}$). 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 6.7.3 System Control Bits for more details.

6.6.1 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

6.6.2 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.

6.6.2.1 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.

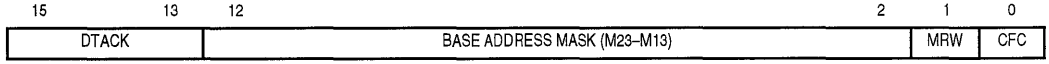
EN—Enable

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.

6.6.2.2 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)).

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.

Table 6-8. 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

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.

6.6.2.3 PCMCIA Protection Register (PPR)

This register controls the PCMCIA protection mechanism that blocks the chip select mechanism for external PCMCIA accesses made to the block of memory covered by the protected chip select. It also includes the Protect Internal Registers (PIR) bit that controls the

protection mechanism to block PCMCIA accesses to the internal registers and dual port RAM. This register is located at BAR+82C.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	PIR	PCS3	PCS2	PCS1	PCS0

PCS(3-0)—Protect Chip Select

Each of these bits when set will disable the corresponding chip select for PCMCIA accesses. When a PCMCIA controller access is made to a protected memory space, the chip select line will not assert and the DTACK line will also not be asserted. This will result in a 68000 bus error and a bus error on the PCMCIA bus (if the BERR bit is set in the PCMR register).

PIR—Protect Internal Registers

This bit when set will block PCMCIA accesses to internal registers and the internal dual port ram. When a PCMCIA access is made to an internal register or memory location while the PIR bit is set, the register will not be addressed and the DTACK line will not be asserted. This will result in a 68000 bus error and a bus error on the PCMCIA bus (if the BERR bit is set in the PCMR register).

6.6.3 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 M 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 M block beginning at $200000 and a 1 M block at $A00000.

6.7 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

6.7.1 System Control Register (SCR)

The SCR is a 32-bit register that consists of system status and control bits, a bus arbiter control bit, and hardware watchdog control bits. Refer to Figure 6-10 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).

$F4

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

$F5

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

$F6

15	14	13	12	11	10 9 8
RESERVED	RESERVED	RESERVED	SAM	HWDEN	HWDCN2–HWDCN0

Figure 6-10. System Control Register

Table 6-9. 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
SAM	Synchronous Access Mode	3.8.3, 3.8.4
HWDEN	Hardware Watchdog Enable	3.8.6
HWDCN	Hardware Watchdog Count	3.8.6

6.7.2 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 and PCMCIA controller 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 $\overline{\text{BCLR}}$ 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 6.7.6 Hardware Watchdog) reaches the end of its time interval; $\overline{\text{BERR}}$ 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, $\overline{\text{BERR}}$ will be asserted on the bus cycle that sets this bit. If WPV and WPVE are both set when a write protect violation occurs, $\overline{\text{BERR}}$ 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 $\overline{\text{BERR}}$ to be asserted. If this bit is already set when another address decode conflict occurs, $\overline{\text{BERR}}$ 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 $\overline{\text{BERR}}$ 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.

NOTE

In MC68356, there is no logic that decodes a conflict between chip-select user-allocated memory and DSP host port address allocation. Therefore, the DSP host port memory space programmed in the host port base address register (HBAR) must not overlap any of the chip select memory blocks.

6.7.3 System Control Bits

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

WPVE—Write Protect Violation Enable

0 = $\overline{\text{BERR}}$ is not asserted when a write protect violation occurs.
1 = $\overline{\text{BERR}}$ 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 IMP 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 every memory cycle to external memory and peripherals and also in every cycle to 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.

BCLM bit affects PCMCIA operation, if enabled in PCMR register (see 8.4.2 PCMCIA Mode Register - PCMR).

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 IMP peripheral area. This bit is not relevant for applications that do not have external bus masters that access the IMP. 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 IMP, 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 IMP internal RAM and registers (including BAR and SCR) by an external master are asynchronous to the IMP clock. Read and write accesses are with three wait states, and $\overline{\text{DTACK}}$ is asserted by the IMP assuming three wait-state accesses. This is the default value.

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

6.7.4 Disable CPU Logic (M68000)

The IMP 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 IMP 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 IMP.
2. $\overline{\text{BG}}$ will be an input to the IDMA and SDMA from the external M68000 bus, rather than being an output from the IMP. When BG is sampled as low by the IMP, it waits for $\overline{\text{AS}}$,

$\overline{BERR}$, $\overline{HALT}$, and $\overline{BGACK}$ to be negated, and then asserts $\overline{BGACK}$ and performs one or more bus cycles.

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 IMP RAM and registers may be asynchronous or synchronous to the IMP 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 IMP devices share a single M68000 bus, vector generation at level 4 should be prevented on all but one IMP. When using disable CPU mode to implement an interface, such as between the MC68020 and a single IMP, vector generation can be enabled. For this purpose, the VGE bit is defined.

VGE—Vector Generation Enable

0 = In disable CPU mode, the IMP will not output interrupt vectors during interrupt acknowledge cycles.
1 = In disable CPU mode, the IMP 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 IMP 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 requesting the M68000 bus with $\overline{BR}$. See 6.7.6 Hardware Watchdog for further details.

RME—Ram Microcode Enable

This bit is used to initiate the execution of Communication Processor microcode that has been loaded into the dual port RAM. See B.5 RISC Microcode from RAM.

6.7.5 Bus Arbitration Logic

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

6.7.5.1 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)
4. PCMCIA Controller

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 6-11). 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 IMP 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 6-10), and by enabling BCLR functions in PCMR Register.

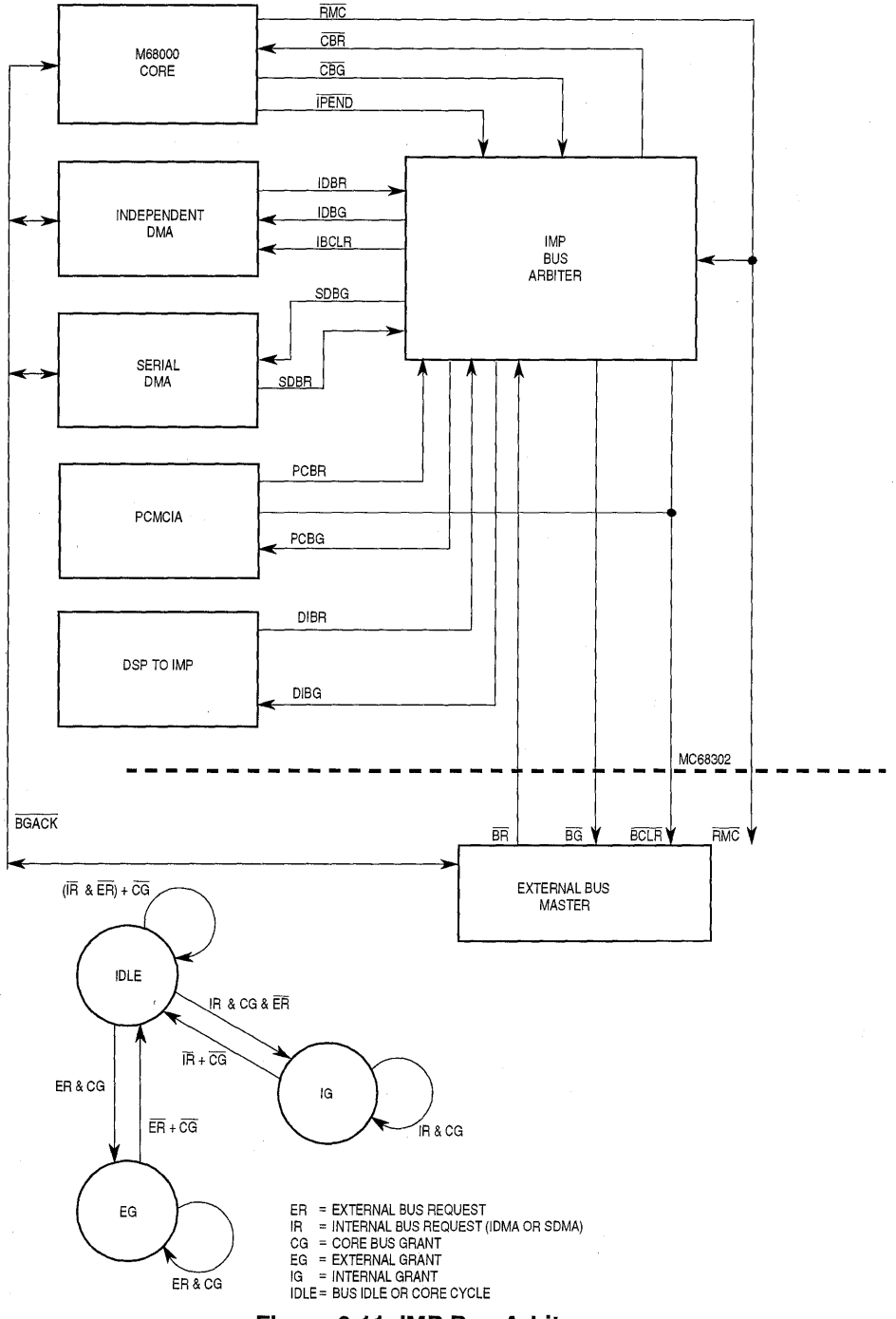


Figure 6-11. IMP Bus Arbiter

Table 6-10. Bus Arbitration Priority Table

$\overline{BCLR}$ Ignored BCLM = 0	$\overline{BCLR}$ Used BCLM = 0	$\overline{BCLR}$ Ignored BCLM = 1	$\overline{BCLR}$ Used BCLM = 1
$\overline{BR}$ Pin SDMA IDMA PCMCIA/DSPIMP M68000 Interrupts M68000	SDMA IDMA PCMCIA/DSPIMP $\overline{BR}$ Pin M68000 Interrupts M68000	$\overline{BR}$ Pin SDMA M68000 Interrupts IDMA[4] PCMCIA/DSPIMP M68000	SDMA M68000 Interrupts IDMA[4] PCMCIA/DSPIMP 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 ($\overline{BCLR}$) 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 $\overline{BCLR}$ 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, $\overline{BCLR}$ will be asserted until the interrupt priority active (IPA) bit in the SCR is cleared. To implement this feature, $\overline{BCLR}$ 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 ($\overline{BR}$ 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 $\overline{BCLR}$ could be used to negate $\overline{DREQ}$ 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.

6.7.5.2 External Bus Arbitration

An external bus master may gain ownership of the M68000 bus by asserting the bus request ($\overline{BR}$) 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.

6.7.6 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.

6.8 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 IMP 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.

6.8.1 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 IMP DRAM system is shown in Figure 6-12. The IMP 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, $\overline{UDS}$ or $\overline{LDS}$ will be asserted from the IMP, but not both. A refresh to an odd address will assert $\overline{LDS}$; whereas, a refresh to an even address will assert $\overline{UDS}$.

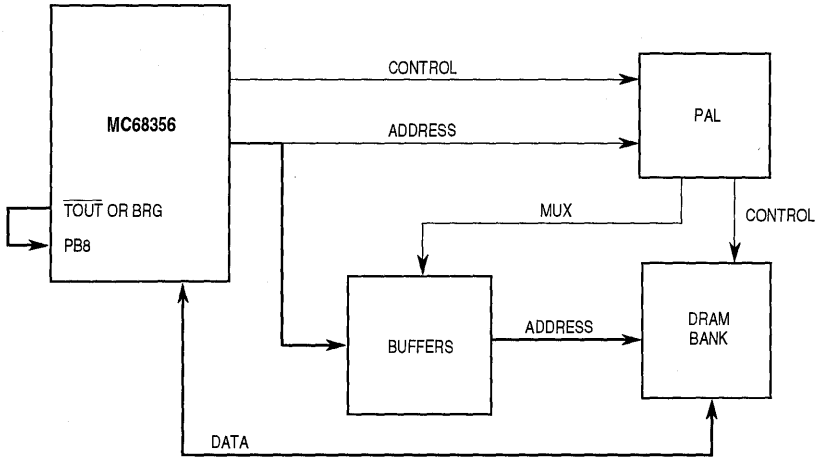


Figure 6-12. DRAM Control Block Diagram

6.8.2 DRAM Refresh Controller Bus Timing

The DRAM refresh controller bus cycles are actually SDMA byte read accesses (see 7.2 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 $\overline{BCLR}$ 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 $\overline{BERR}$ 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.

6.8.3 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.

6.8.4 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.

6.8.5 DRAM Refresh Memory Map

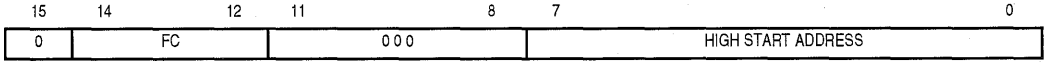
The DRAM refresh memory map replaces the SCC2 Tx BD 6 and Tx BD 7 structures in the parameter RAM. The wrap bit must therefore be set in SCC2 Tx BD 5 so that only six Tx BDs are used for SCC2. These parameters should be written before the DRAM refresh controller receives its first request, but may be read at any time. They are undefined at reset.

Table 6-11. 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.

6.8.6 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 × 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 IMP 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 IMP is a byte read. The COUNT value is the number of word refreshes required in the entire DRAM bank.

6.9 IMP Control of the DSP Reset, Modes and Interrupts

The DISC register contains the bits that control the following functions:

1. The IMP control of the DSP reset, modes and interrupts.
2. DSP SCI serial connections
3. TIN1 divide-by-two, and BRG1 and RCLK1/TCLK1 pin options.

Item #1, IMP control of the DSP reset, modes and interrupts, is described in the following section. Items #2 and 3 are covered in 7.5.3 DSP Interconnection and Serial Connections Register–DISC.

Through the DISC register, DSP reset and interrupt functions can be generated from the IMP instead of using the external DSP pins. The IMP can control the following DSP functions:

1. DSP hardware reset: The RESET signal normally connected to the DRESET pin can be rerouted to the RST bit in the DISC register.
2. Similarly to the reset function, the MODA/IRQA, MODB/IRQB, and MODC/NMI pin functions can be rerouted to the DISC register bits. The DSP mode can be configured internally during power-on reset by setting the MODA/IRQA, MODB/IRQB, and MODC/NMI bits in the DISC register.
3. The IMP can assert the NMI, IRQ1 and IRQ2 interrupt lines of the DSP internally - as mentioned above, the IRQA, B and/or NMI pins can be rerouted and controlled by the IMP.
4. The IMP can wake-up the DSP from STOP mode by asserting the IRQ bits or resetting the DSP using the RST bit.

The 68000 can read and write to the DISC register. This register contains 4 control bits that select between the external pins and the bits 0–3 in this register. The following multiplexer outputs are connected to the DSP: reset, MODA/IRQA, MODB/IRQB and MODC/NMI pins

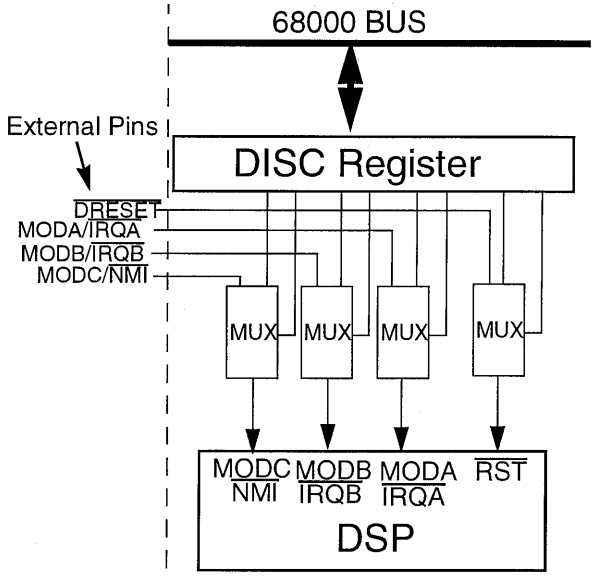


Figure 6-13. IMP to DSP Reset, Mode and Interrupts

6.9.1 IMP Control of DSP Reset at Power-On Reset

The IMP can reset the DSP and set its initial operating mode using the following procedure:

1. After the IMP has completed its reset initialization routine it should, in a single access to the DISC register, 1) reset the RST bit to zero, 2) set the MODA, B, C bits to the desired DSP bootstrap mode, and 3) set all the SEL bits to one. This access switches control of the DSP reset line to the IMP and holds DSP reset still asserted. The MOD signals are also applied to the DSP.

Although not required, the DRESET pin can be grounded to hold the DSP in reset until the IMP is ready to release it from reset. This eliminates the need for separate reset circuitry for the $\overline{\text{DRESET}}$ pin.

2. To release the DSP from reset, the IMP should, in a single access, 1) set the RST bit to one, 2) program MODA, B, C to the corresponding value desired for the IRQA,B and NMI signals (usually ones for the MOD bits to avoid an immediate DSP interrupt), and 3) set the SELA, B, C bits to the desired signal source (the corresponding SEL bits should be reset to zero if off-chip interrupt sources are required).

The IMP can interrupt the DSP by setting either the IRQA or IRQB bits in the DSP interconnection and serial connections register (DISC).

6.9.2 IMP-DSP Reset and Mode Interconnections Register

The DSP interconnection and serial connections register (DISC) is a 16-bit read/write register. The lower 8 bits allow the 68000 to control the DSP reset, modes, and interrupt lines directly through internal connections. This register, for example, allows the 68000 core to toggle the DSP IRQ line to wake the DSP up from STOP mode. The upper 8 bits control the DSP SCI serial connections, the TIN1 divide-by-two option, and the BRG1 and RCLK1/TCLK1 pin options. Refer to 7.5.3 DSP Interconnection and Serial Connections Register–DISC for more information on these upper bits.

DISC 68356 Base+$8EE

15	14	13	12	11	10	9	8
TSTCLK1	TSRCLK1	DISBRG1	BRGDIV	IC3	IC2	IC1	IC0
RESET: 0	0	0	0	0	0	0	0

7	6	5	4	3	2	1	0
SELC	SELB	SELA	SELR	MODC/NMI	MODB/IRQB	MODA/IRQA	RST
RESET: 0	0	0	0	0	0	0	0

NOTE

All bits default to 'zeros' (no-connect state) after the IMP system reset.

SELC—SELect MODC/NMI Pin

0 = The DSP MODC/NMI line is driven from the MODC/NMI external pin.
1 = The DSP MODC/NMI line is driven from the MODC/NMI bit in this register (bit 3).

SELB—SELect MODB/IRQBI Pin

0 = The DSP MODB/IRQB line is driven from the MODB/IRQB external pin.
1 = The DSP MODB/IRQB line is driven from the MODB/IRQB bit in this register (bit 2).

SELA—SELect MODA/IRQA Pin

0 = The DSP MODA/IRQA line is driven from the MODA/IRQA external pin.
1 = The DSP MODA/IRQA line is driven from the MODA/IRQA bit in this register (bit 1).

SELR—SELect Reset Pin

0 = The DSP RST line is driven from the RST external pin.
1 = The DSP RST line is driven from the RST bit in this register (bit 0).

MODC—MODC/NMI Signal

0 = The DSP MODC/NMI line is zero if SELC is set.
1 = The DSP MODC/NMI line is one if SELC is set.

MODB—MODB/IRQBI Signal

0 = The DSP MODB/IRQB line is zero if SELB is set.
1 = The DSP MODB/IRQB line is one if SELB is set.

MODC—MODA/IRQA Signal

0 = The DSP MODB/IRQB line is zero if SELC is set.
1 = The DSP MODB/IRQB line is one if SELC is set.

RST—Reset Signal

0 = The DSP RESET line is asserted if SELR is set.
1 = The DSP RESET line is unasserted if SELR is set.

The TSTCLK1, TSRCLK1, DISBRG1, BRGDIV, IC3, IC2, IC1, and IC0 bits are covered in 7.5.3 DSP Interconnection and Serial Connections Register–DISC.

SECTION 7
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
  —Direct Connection of SCC1 to the DSP SCI+ port
- Three Independent Full Duplex Serial Communication Controllers (SCCs) Supporting the Following Protocols:

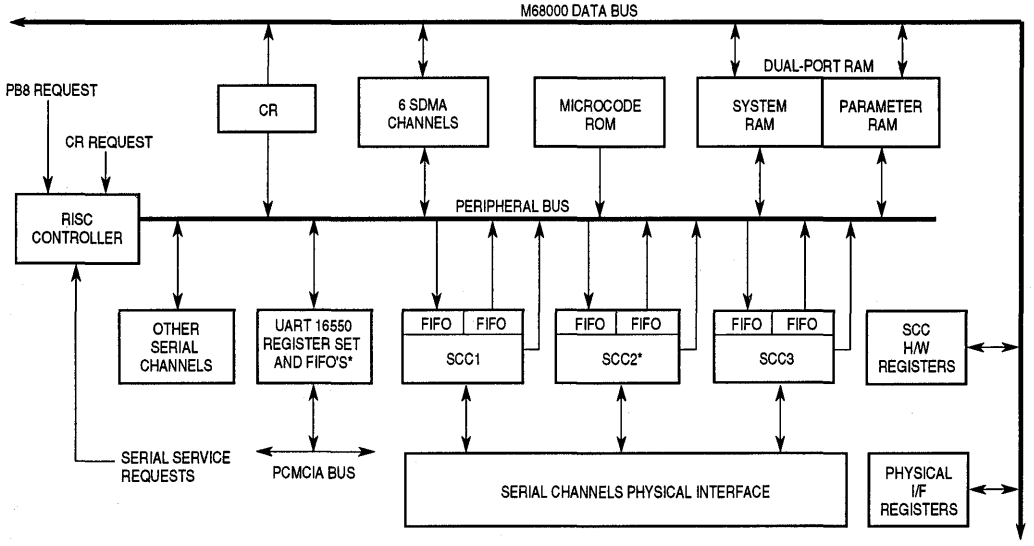
  —High-Level/Synchronous Data Link Control (HDLC/SDLC)
  —Universal Asynchronous Receiver Transmitter (UART)
  —Binary Synchronous Communication (BISYNC)
  —Transparent Modes
  —Uart 16550 hardware and software emulation
- Serial Communication Port (SCP) for Synchronous Communication
- Two Serial Management Controllers (SMCs) to Support the IDL and GCI Management Channels

7.1 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 7-1). 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.



NOTE: If UART 16550 mode is enabled, the UART 16550 block replaces SCC2.

Figure 7-1. 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 (or 16550 receive channel)
8. SCC2 Transmit Channel (or 16550 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 6.8 Dynamic RAM Refresh Controller.

7.2 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 7-2 shows the paths of the data flow. Data from the SCCs (as well as the UART 16550 FIFOs) 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.

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 6.7.5 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

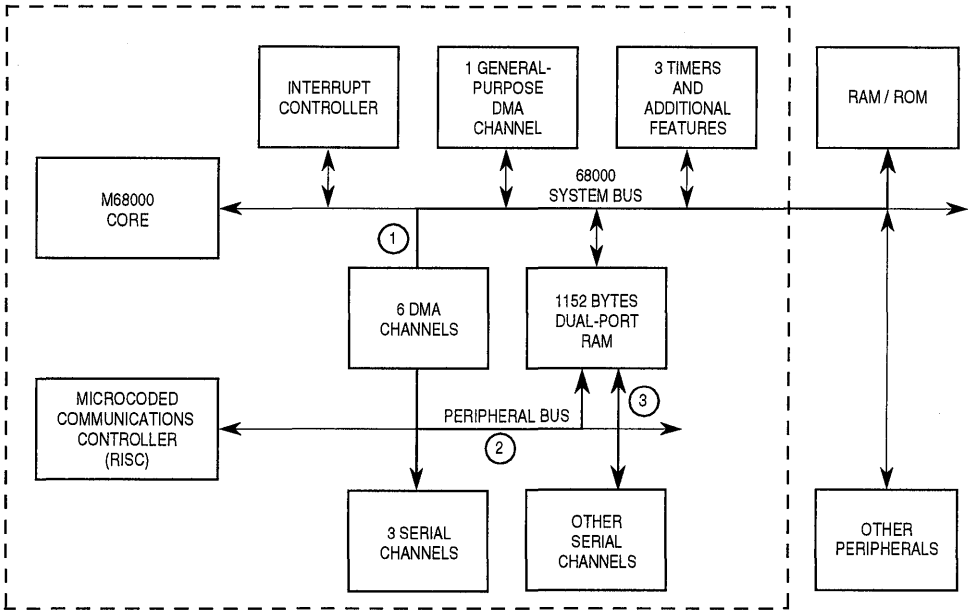


Figure 7-2. Three Serial Data Flow Paths

time. Each bus cycle is a standard M68000-type bus cycle. The chip select and wait state generation logic on the IMP 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 7.5 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 IMP is negated by the SDMA during its access to memory.

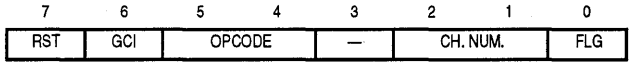
The SDMA keeps the M68000 bus for only one operand (8 or 16 bit) transfer before giving it up. If the SDMA begins a word operation on an 8-bit bus, it will complete that operation before giving up the bus, unless a bus exception such as reset, halt, retry, or bus error occurs. Reset suspends and resets all SDMA activity. Halt suspends activity after the current bus cycle. For information on a bus error during an SDMA access, see 7.5.9.4 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.

7.3 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 IMP. 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 7.5.12 UART Controller.

The detailed command description for the HDLC protocol is presented in 7.5.14 HDLC Controller.

The detailed command description for the BISYNC protocol is presented in 7.5.15 BISYNC Controller.

The detailed command description for the transparent protocol is presented in 7.5.16 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 7.8 Serial Management Controllers (SMCs) for more details.

00 = TRANSMIT ABORT REQUEST; the GCI receiver sends an abort request on the A 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.

7.3.1 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 UART16550; 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.

7.4 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 7.7 Serial Communication Port (SCP).)

The IMP supports five different external physical interfaces from the SCCs:

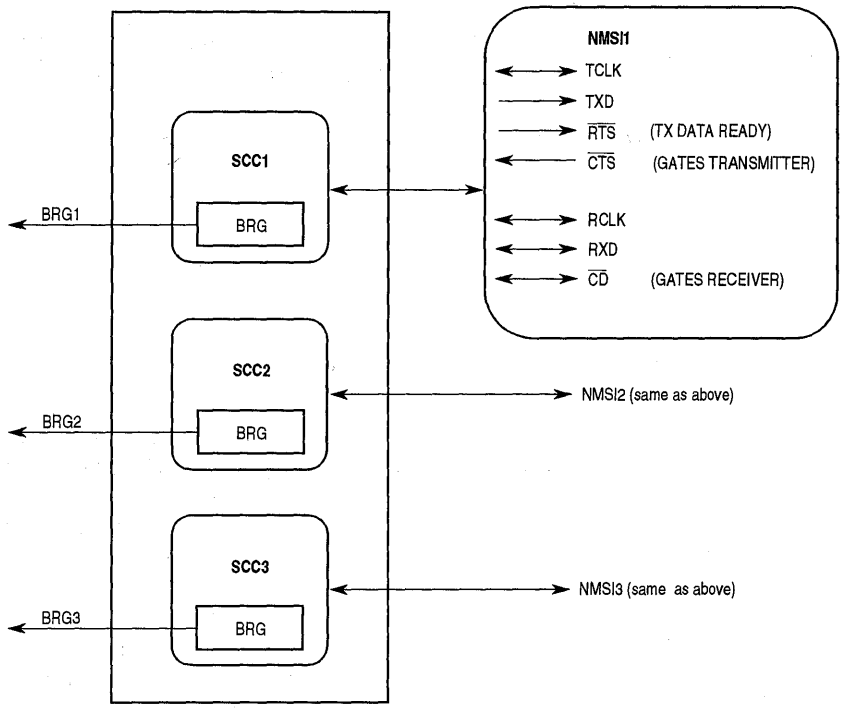
1. NMSI—Nonmultiplexed Serial Interface
2. PCM—Pulse Code Modulation Highway
3. IDL—Interchip Digital Link
4. GCI—General Circuit Interface
5. UART 16550 physical register set through the PCMCIA interface

The most generic physical interface on the IMP 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 7-3. 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 assuming that the PCMCIA interface is not enabled for a 16 bit data bus which in that case these pins will be dedicated to the PCMCIA interface upper data bits and NMSI2 will not be accessible. 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; $\overline{\text{RTS}}$, $\overline{\text{CTS}}$, and $\overline{\text{CD}}$ are multiplexed with the SCP. See 7.7 Serial Communication Port (SCP) for more details on the SCP.

When the PCMCIA interface is enabled NMSI1 signals CD1, CTS1, RTS1 and BRG1 will not be accessible unless the IPINS bit in the PCMR register is set. This bit should be set when it is desired to have a ISDN interface and a PCMCIA interface simultaneously.

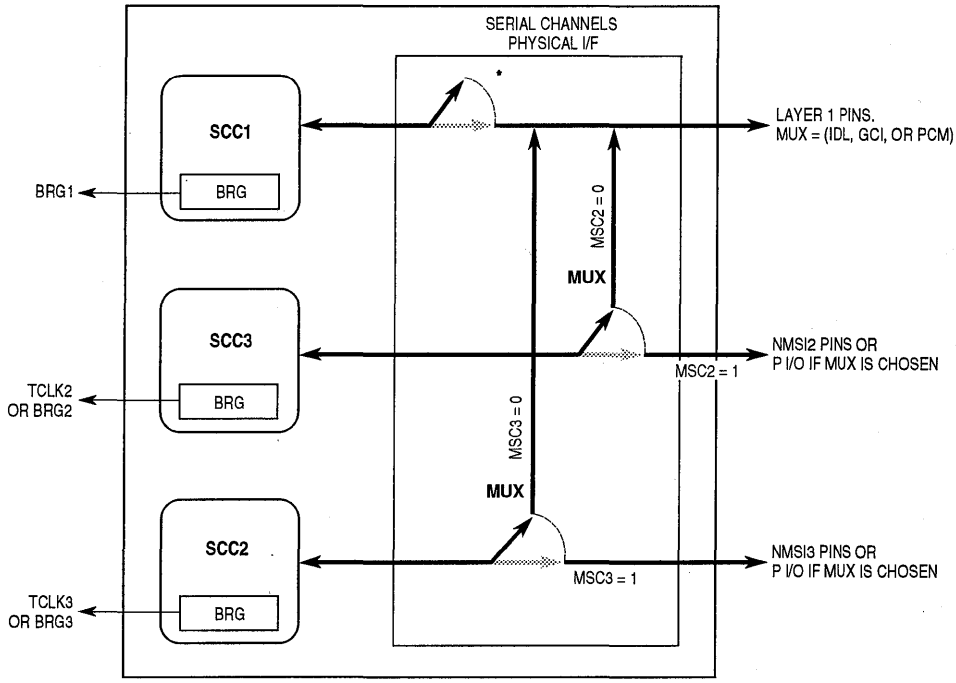
The other three physical interfaces, PCM, IDL, and GCI here are called multiplexed interfaces since they allow data from one, two, or all three SCCs to be time multiplexed together on the same pins. Note that if a multiplexed mode is chosen, the first SCC to use that mode must be SCC1 (unless the IC3-1 in the DISC register is set to 1001) since the three multiplexed modes share pins with SCC1. If the IC3-1 bits in the DISC register=1001, then SCC1 will not be connected to the multiplexed interface as shown in Figure 7-4.

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 7-4). 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.



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

Figure 7-3. NMSI Physical Interface



* If IC = 1001 in DISC register

Figure 7-4. Multiplexed Mode on SCC1 Opens Additional Configuration Possibilities

There are five serial channel physical interface combinations for the three SCCs (see Table 7-1).

Table 7-1. 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.

If IC=1001 in the DISC register, then SCC1 will not be connected to the muxed interface.

The PCM highway interface is a flexible time-division multiplexed interface. It allows the IMP 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 IMP 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 IMP through two SMC channels: SMC1 and SMC2. (For other matters relating to the SMCs, refer to 7.8 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 7-5 for the serial channels physical interface block diagram.

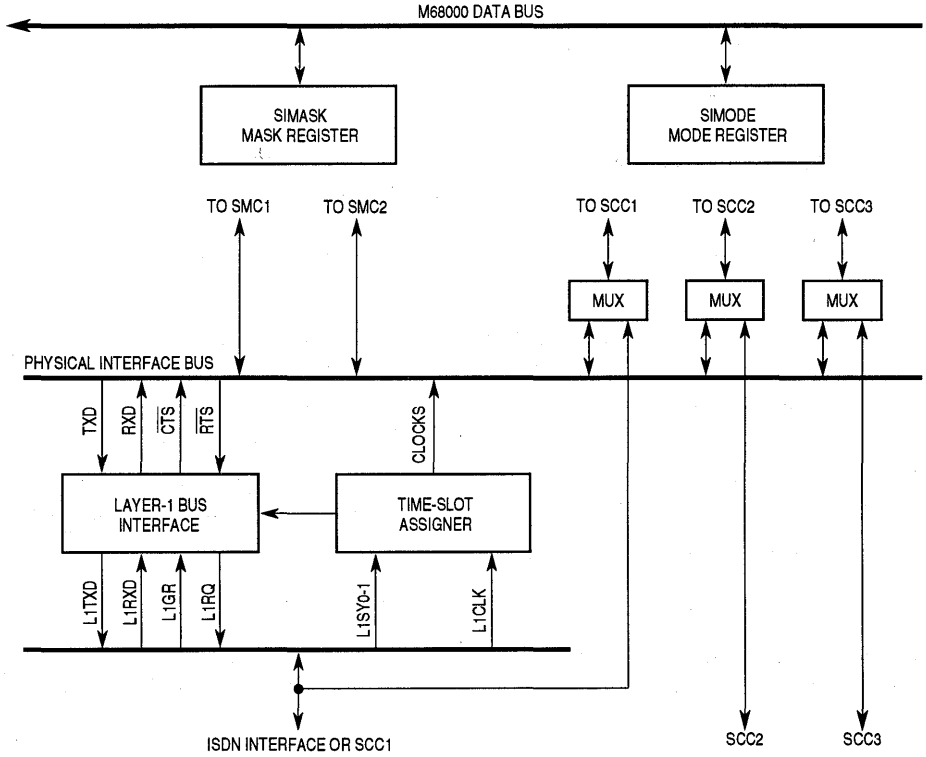


Figure 7-5. Serial Channels Physical Interface Block Diagram

7.4.1 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 7-6 for the IDL bus signals.

The IMP supports 10-bit IDL as shown in Figure 7-6; it does not support 8-bit IDL.

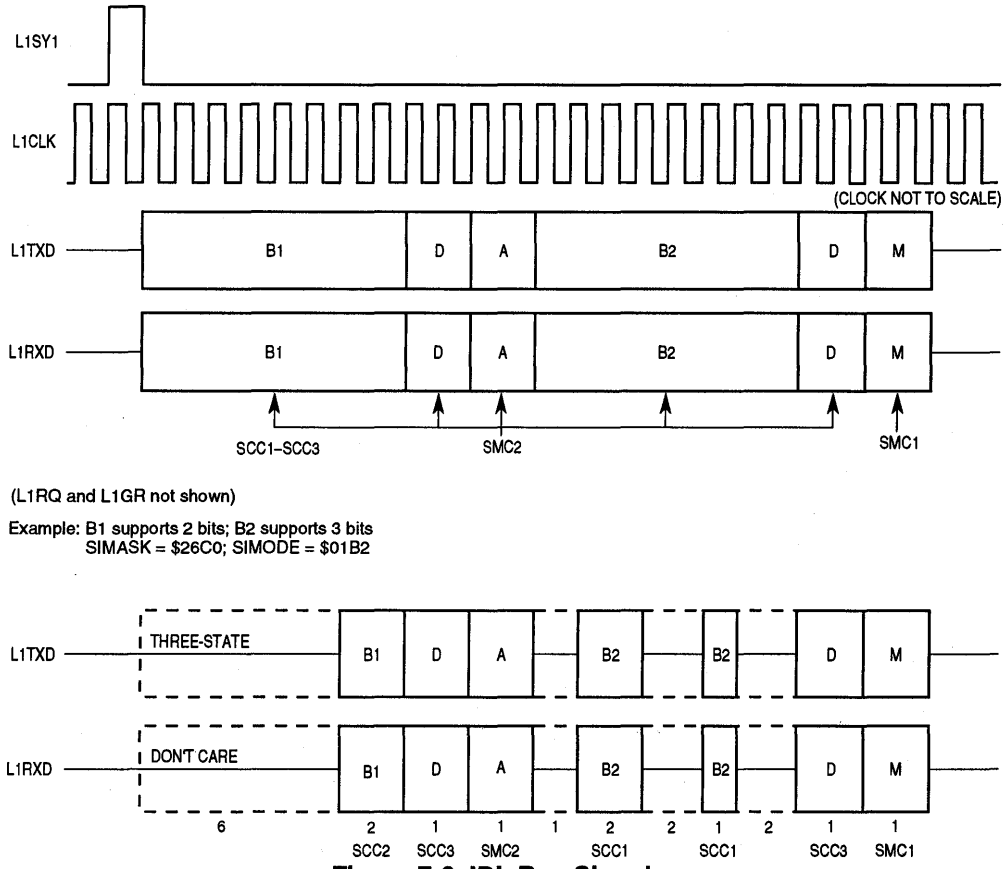


Figure 7-6. IDL Bus Signals

An application of the IDL interface is to build a basic rate ISDN terminal adaptor (see Figure 7-7). 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. 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 or BISYNC mode depending on the terminal protocol used.

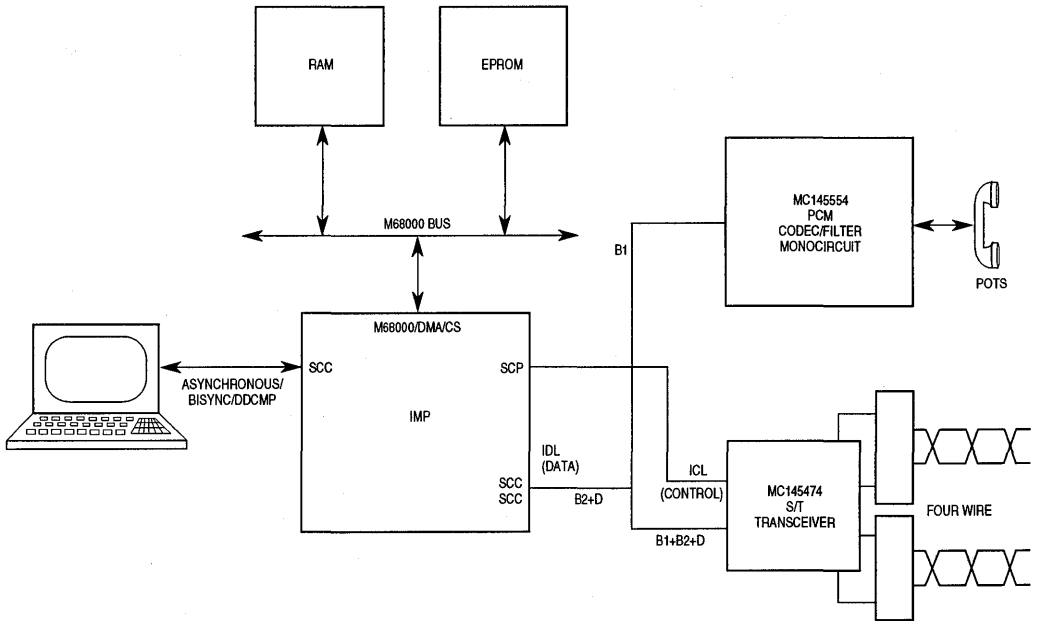


Figure 7-7. IDL Terminal Adaptor

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 insure 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 7-6 for an example of supporting two bits in the B1 channel and three bits in the B2 channel.

7.4.2 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 con-

trol 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 7-8. 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 25-MHz system clock, the serial clock rate must not exceed 8.333MHz.

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.

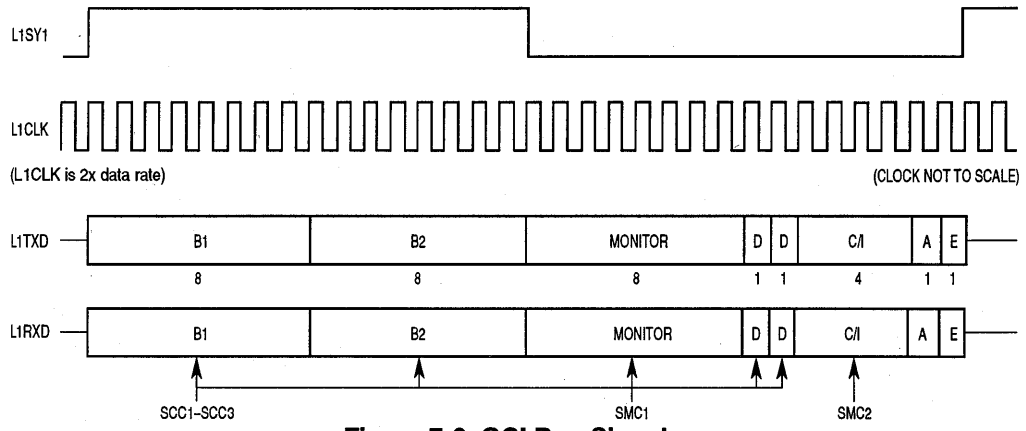


Figure 7-8. GCI Bus Signals

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 insure 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.

7.4.3 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 7-2).

Table 7-2. 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 IMP 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 IMP 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 IMP. They select one of three PCM channels to which data is routed or select no channel (see Table 7-3).

Table 7-3. 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 7-9). 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 7-10. 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.

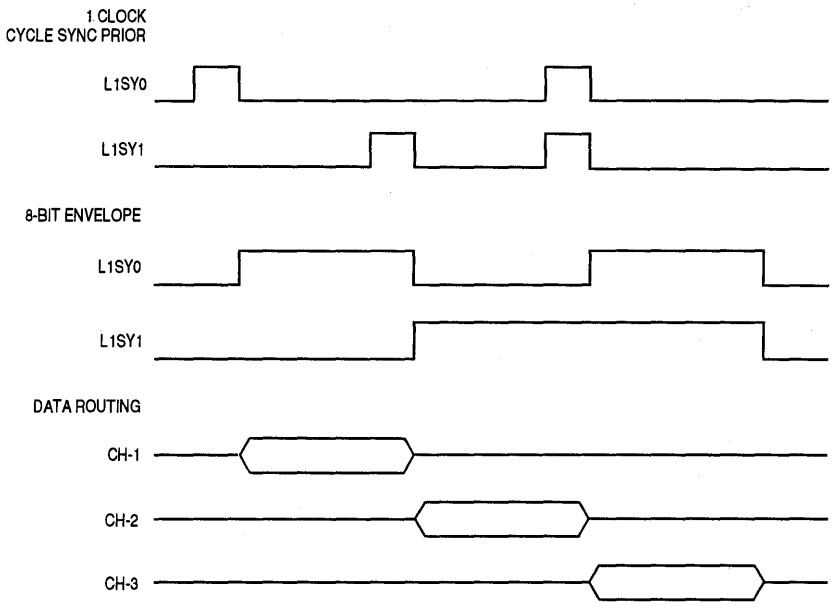
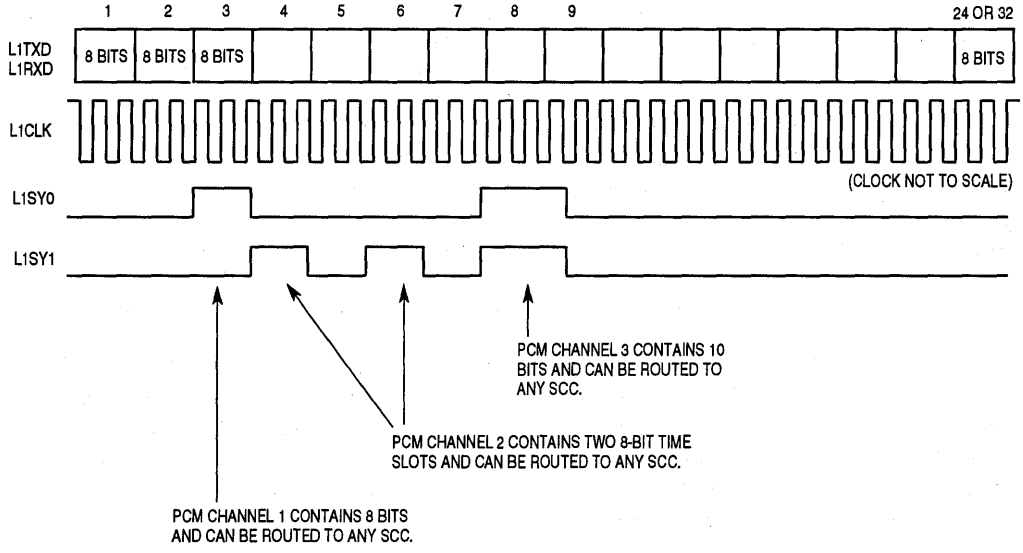


Figure 7-9. Two PCM Sync Methods



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

Figure 7-10. PCM Channel Assignment on a T1/CEPT Line

7.4.4 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, $\overline{CD1}$, $\overline{CTS1}$, and $\overline{RTS1}$) 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 6.3 Parallel I/O Ports for an example. $\overline{RTS}$ is an output of the transmitter, while $\overline{CTS}$ and $\overline{CD}$ are inputs to the transmitter and receiver, respectively. See 7.5.4 SCC Mode Register (SCM) and 7.4 Serial Channels Physical Interface for additional information.

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

7.4.5 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.

7.4.5.1 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 7.4.2 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.

7

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)

NOTE

If IC3-0=1001 in the DISC register, SCC1 will not be connected to the multiplexed bus regardless of the state of the B1, B2 and D channel route bits above.

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 6.3 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 7.7 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.

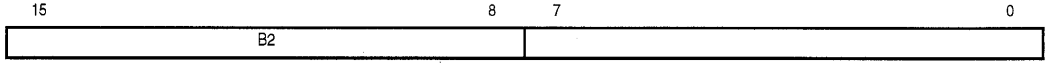
10 = IDL Mode
When working in IDL/GCI mode, each ISDN channel (D, B1, and B2) can be routed independently to each of the three SCCs. This connection is determined by the DRB, DRA, B1RB, B1RA, B2RB, and B2RA bits. SCC2 and SCC3 can be connected directly to their respective NMSI pins (if they are not needed 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.

7.4.5.2 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.



7

NOTE

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

7.5 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, UART, and several transparent modes. Each protocol can be implemented with IDL, GCI, PCM, or NMSI physical layer interfaces (see 7.4 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 7.4 Serial Channels Physical Interface).

The receive and transmit section of each SCC is supported with one of the six dedicated SDMA channels (see 7.2 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 25-MHz system clock frequency, the serial clock must not exceed 10MHz. 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 7-11.

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 6.3 Parallel I/O Ports for related details.)

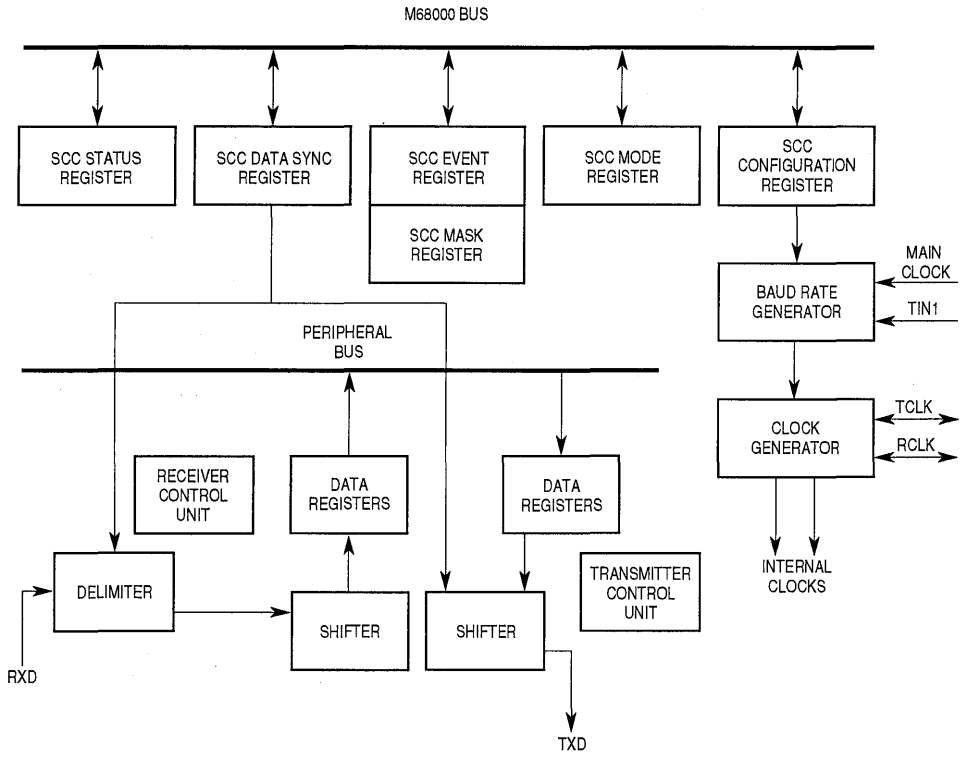


Figure 7-11. SCC Block Diagram

7.5.1 SCC Features

Each SCC channel has the following features:

- HDLC/SDLC, BISYNC, UART, or Transparent 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

7.5.2 SCC Configuration Register (SCON)

Each SCC controller has a configuration register that controls its operation and selects its clock source and baud rate. Figure 7-12 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 2-10.

NOTE

The BRG output is 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 6-6.

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.

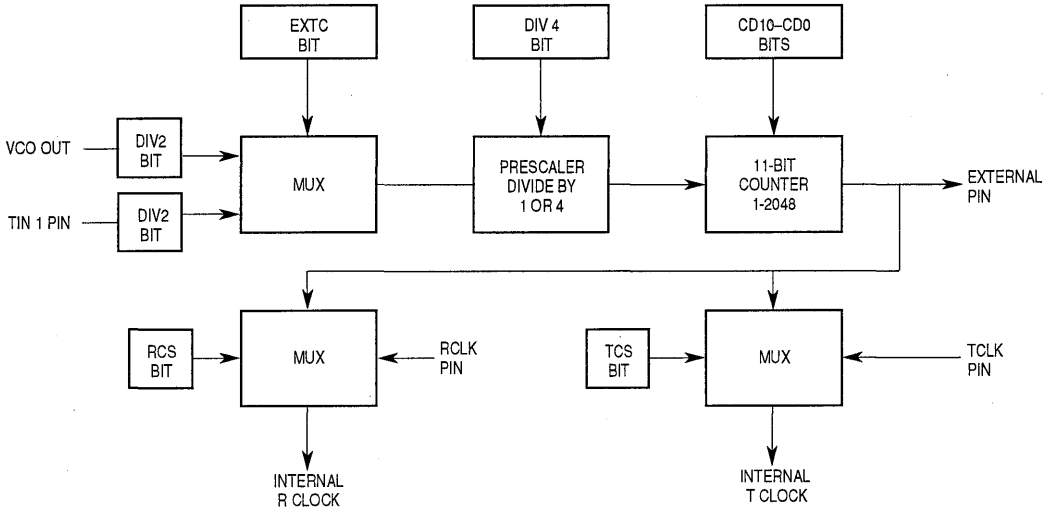


Figure 7-12. SCC Baud Rate Generator

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.

7.5.2.1 Divide by 2 Input Blocks

The divide by two blocks shown in Figure 7-12 can be enabled by setting the BCD bit in the IOMCR register if the BRG clock source is derived from the IMP system clock, or by setting the BRGDIV bit in the DISC register if the BRG clock source is derived from the TIN pin.

7.5.2.2 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 25-MHz crystal gives a UART clock rate of 8.33MHz and a baud rate of 520 kbaud. Assuming again a 25-MHz crystal, an externally supplied UART clock on the TCLK or RCLK pins can be as high as 10MHz, giving a maximum baud rate of 625kbaud.

The baud rate using the baud rate generator is (System Clock or TIN1 clock)/ (1or 2)/(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 7-4 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 7-4. Typical Bit Rates of Asynchronous Communication

Baud Rates	18.432				22.12				25			
	DIV2	DIV4	DIV	Actual Frequency	DIV2	DIV4	DIV	Actual Frequency	DIV2	DIV4	DIV	Actual Frequency
110	1	1	1308	110	1	1	1570	110	1	1	1570	110
150	1	1	959	150	1	1	1151	150.01	1	1	1151	150.01
300	1	1	479	300	0	1	1151	300.02	0	1	1151	300.02
600	1	1	239	600	0	1	575	600.04	0	1	575	600.04
1200	1	1	119	1200	0	1	287	1200.09	0	1	287	1200.09
2400	1	1	59	2400	0	1	143	2400.17	0	1	143	2400.17
4800	1	1	29	4800	0	1	71	4800.35	0	1	71	4800.35
9600	1	1	14	9600	0	1	35	9600.69	0	1	35	9600.69
19200	1	1	29	19200	0	0	71	19201.39	0	0	71	19201.39
28800	1	0	4	28800	0	0	47	28802.08	0	0	47	28802.08
38400	0	0	29	38400	0	0	35	38402.78	0	0	35	38402.78
115200	0	0	9	115200	0	0	11	115208.33	0	0	11	115208.33

7.5.2.3 Synchronous Baud Rate Generator Examples

- For synchronous communication (HDLC/SDLC, BISYNC, and Transparent), 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 2)/(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,DIV2=0 and the Clock Divider = 239. Of course, a 64 kbps rate provided externally on the TCLK or RCLK pins could also be used.

7.5.3 DSP Interconnection and Serial Connections Register–DISC

The DSP interconnection and serial connections register (DISC) is an 16-bit read/write register. The lower 8-bits are used to allow the 68000 to control the DSP reset, modes, and interrupt lines directly through internal connections and are discussed in 6.9.2 IMP-DSP Reset and Mode Interconnections Register. The upper 8 bits control: (1) the direct connection of the DSP SCI+ port to the IMP SCC1 port and IMP serial pins, (2) enabling the divide by 2 prescaler for the baud rate generator from the TIN1 pin, and (3) options for three stating the BRG1, TCLK1, and RCLK1 pins.

DISC Base+$8EE

15	14	13	12	11	10	9	8
TSTCLK1	TSRCLK1	TSTBRG1	BRGDIV	IC3	IC2	IIC1	IC0
RESET: 0	0	0	0	0	0	0	0

7	6	5	4	3	2	1	0
SELC	SELB	SELA	SELR	MODC/NMI	MODB/IRQB	MODA/IRQA	RST
RESET: 0	0	0	0	0	0	0	0

7.5.3.1 BRG1 and RCLK1/TCLK1 Pin Options

TSTBRG1—Disable BRG1

0 = When the PCMCIA interface is not enabled, the BRG1 clock is driven on the BRG1 pin.

1 = When the PCMCIA interface is not enabled, the BRG1 output is three-stated.

In order to allow the baud rate generator output clocks (which will be driven on TCLK1 and RCLK1 when an internal baud rate generator is used for SCC1) to be disabled, the following bits are provided.

TSRCLK1

0 = RCLK1 is driven on its pin when SCC1 RCLK is the baud rate generator output.
1 = RCLK1 is three-state.

TSTCLK1

0 = TCLK1 is driven on its pin when SCC1 RCLK is the baud rate generator output.
1 = TCLK1 is three-state.

BRGDIV

Enables and disables the divide by two block between the TIN1 pin and the BRG1 prescaler input (See Figure 7-12).

0 = The divide by two block is disabled.
1 = The divide by two block is enabled.

7.5.3.2 SCI+ Serial Connections

Figure 7-13 shows the suggested serial port assignments for a modem application.

The typical modem application using the MC68356 would use the IMP to handle the upper level data compression, error correction and serial bit stream formatting (such as HDLC). Data processed by the IMP would then be transferred to the DSP from SCC1 to the SCI+ port. A Codec typically is connected to the synchronous serial interface (SSI) of the DSP and optionally provides the bit and baud clocks to the SCC1 to SCI+ connections to clock the data transfers. The analog side of the Codec is connected to the Data Access Arrangement (DAA) which interfaces to the telephone line.

The MC68356 has some extra options for connection of the SCI+ signals to the serial channels and serial pins of the IMP. SCC1 must be the serial channel that connects to the DSP SCI+. SCC2 should be used for the connection to the DTE (typically the PC or host computer). SCC2 is also the serial channel used for UART 16550 emulation (see 7.6.116550 Emulation Controller Features).

Given the serial port assignments in Figure 7-13, there are three different options for connecting the SCI+ clock and data signals to either the SCC1 directly, or to external pins. These optional serial connections can be enabled by programming the IC3–0 bits in the DSP interconnection and serial connections register (DISC) (bits 8–15).

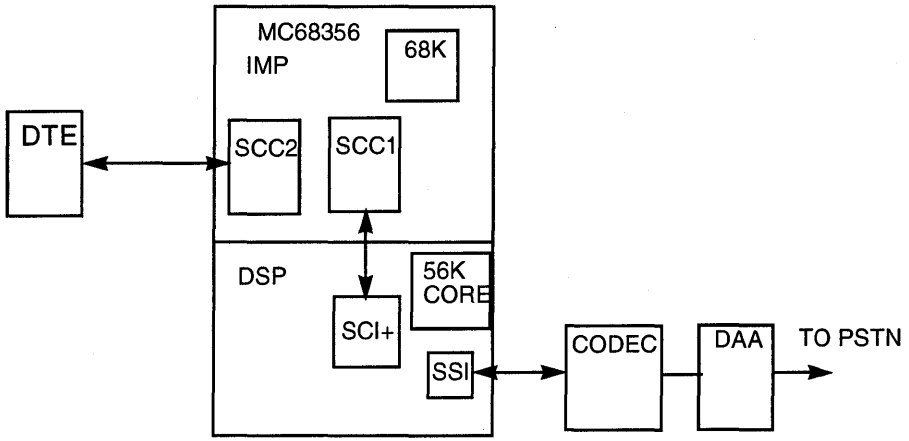


Figure 7-13. Suggested Modem Serial Port Assignments

IC3 – IC0 —IMP and DSP Interconnection

The following modes are supported.

IC3 – IC0 = 0000 Normal IMP mode
IC3 – IC0 = 0001 SCI+ to SCC1
IC3 – IC0 = 0111 SCI+ to DTE (SCC2 pins)
IC3 – IC0 = 0101 Clocks:NMSI1 driving the SCI+ and NMSI2 (DTE), Data: SCC1 to SCI+ and SCC2 to NMSI2
IC3 – IC0 = 1001 SCI+ Clocks with ISDN

7.5.3.2.1 Normal Mode (IC3 – IC0 = 0000)

The SCI+ to SCC connection is not enabled. The SCC1 functions in normal mode with its clock and data lines connected to its pins (NMSI1). This mode of operation supports the normal IMP interconnection in non-multiplexed or multiplexed modes of operation. This is the default mode.

7.5.3.2.2 SCI+ to SCC1 (IC3 – IC0 = 0001)

The SCI+ port is internally connected to the IMP SCC1 port as shown in Figure 7-14. The SCI+ receiver input can be monitored on the TXD1 pin of NMSI1. However, the RXD1 pin cannot be used to monitor the SCI+ TXD signal since this pin is input only in this mode. The SCI+ port can be set up in either synchronous or asynchronous modes to communicate with SCC1. In SCI synchronous mode, the SCI+ must have both the transmit and receive clocks provided to it either by the SCC1 through the internal connection or from the RCLK1 and TCLK1 pins. A common clock should be input on both the SCI+ TCLK and RCLK pins if it is desired to use an asynchronous mode with and external (to the SCI+) baud rate generator.

The clocking of SCC1 and SCI+ is done either externally or from baud rate generator BRG1.

NOTE

The SCKP bit in the DSP SCI control register (SCR) must be set to one for SCI+ to SCC1 internal data transfers in this mode to prevent erroneous data transfers.

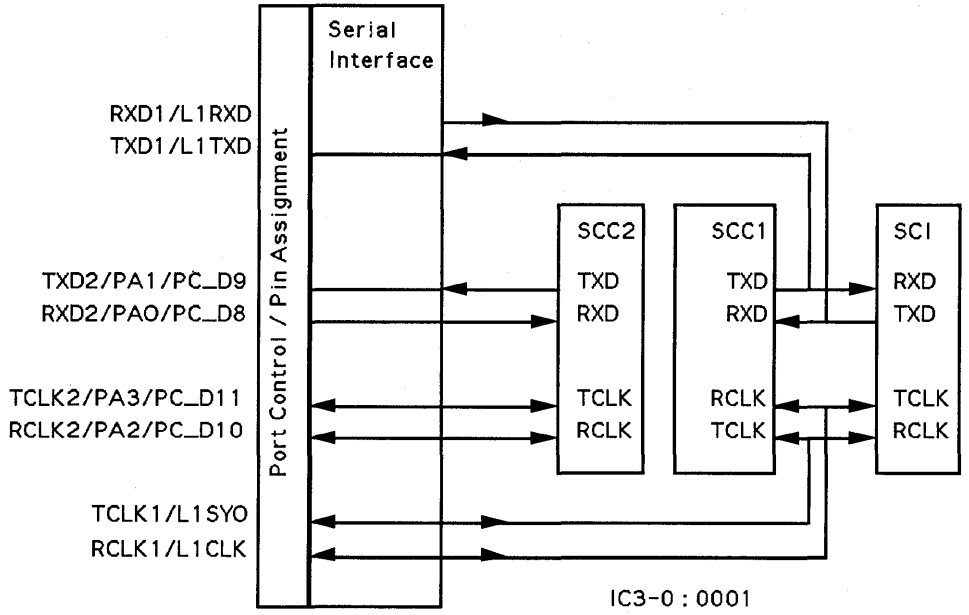


Figure 7-14. SCI+ to SCC1 (IC3 – IC0 = 0001)

7.5.3.2.3 SCI+ to DTE: (IC3 – IC0 = 0111)

This mode is used to connect the SCI+ port directly to external pins to allow asynchronous or synchronous-slave communication with external devices.This mode is used when it is desired to bypass the IMP processing of the serial data stream from the DTE and connect the DTE directly to the SCI+ port of the DSP. This mode is also useful when it is desired to interface the SCI+ port clock and data pins to external devices in either synchronous or asynchronous mode. The signals are routed as shown in Figure 7-15.

When the clock source is the TCLK1 and/or RCLK1 pins, the TCLK2 and/or RCLK2 pins will be driven with that clock. If either clock source is the BRG1 from the SCC1, the NMSI2 TCLK2 and/or RCLK2 pins will be driven with that clock. If SCC2 is set to drive clocks (TCS and/or RCS in SCON2= 0), then the special clock connections will be overridden and the NMSI2 clock will be driven from SCC2.

NOTE

The SCKP bit in the DSP SCI Control Register (SCR) must be set to one for SCI+ to SCC1 internal data transfers in this mode to prevent erroneous data transfers.

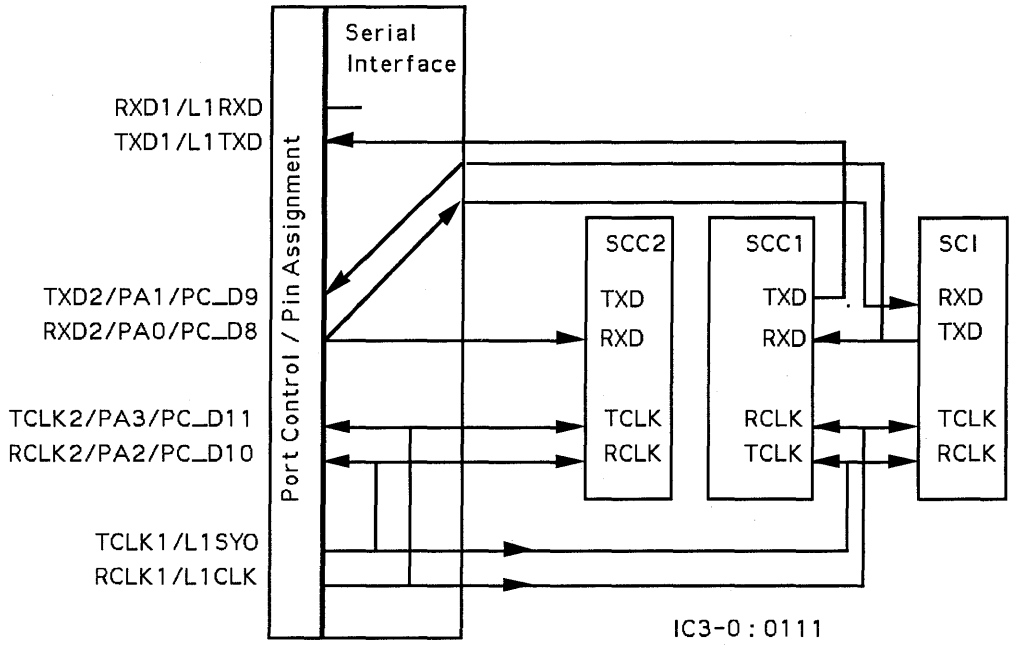


Figure 7-15. SCI+ to DTE (IC3 – IC0 = 0111)

7.5.3.2.4 Clocks:NMSI1 Driving the SCI+ and NMSI2 (DTE), Data: SCC1 to SCI+ and SCC2 to NMSI2 (IC3 – IC0 = 0101)

This mode connects the SCI+ port data lines directly to the SCC1 lines and connects the NMSI2 clocks to the SCC1, SCI+, and SCC2 clocks (as well as theNMSI2 clock pins). This mode is useful when it is desired to have an external clocking source on the NMSI1 pins drive all serial channels shown as well as the DTE.

When the clock source is the TCLK1 and/or RCLK1 pins, the TCLK2 and/or RCLK2 pins will be driven with that clock. If either clock source is the BRG1 from the SCC1, the NMSI2 TCLK2 and/or RCLK2 pins will be driven with that clock. If SCC2 is set to drive clocks (TCS and/or RCS in SCON2= 0), then the special clock connections will be overridden and the NMSI2 clock will be driven from SCC2.

NOTE

The SCKP bit in the DSP SCI Control Register (SCR) must be set to one for SCI+ to SCC1 internal data transfers in this mode to prevent erroneous data transfers.

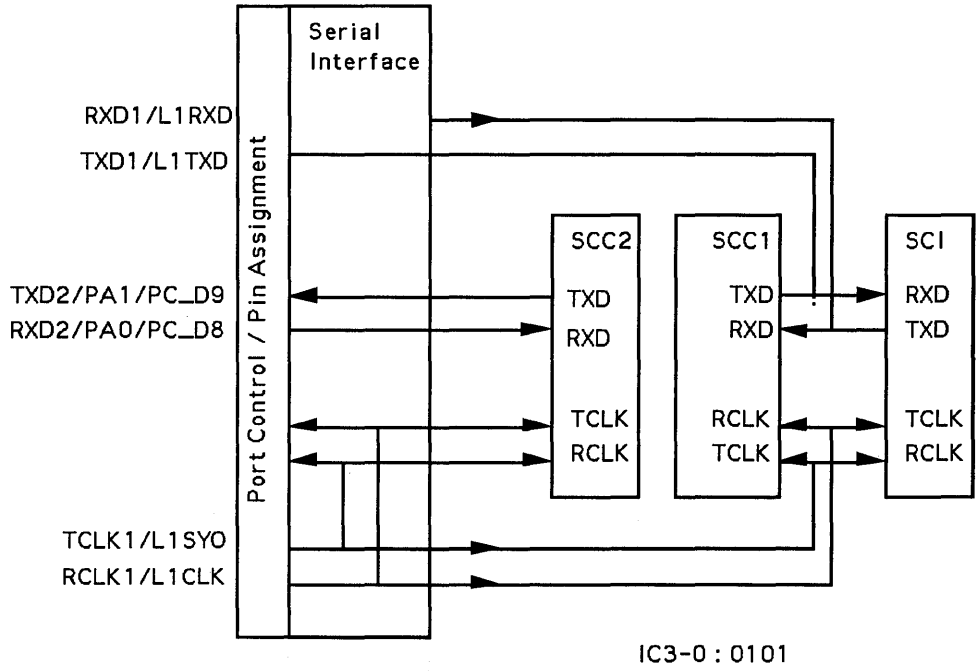


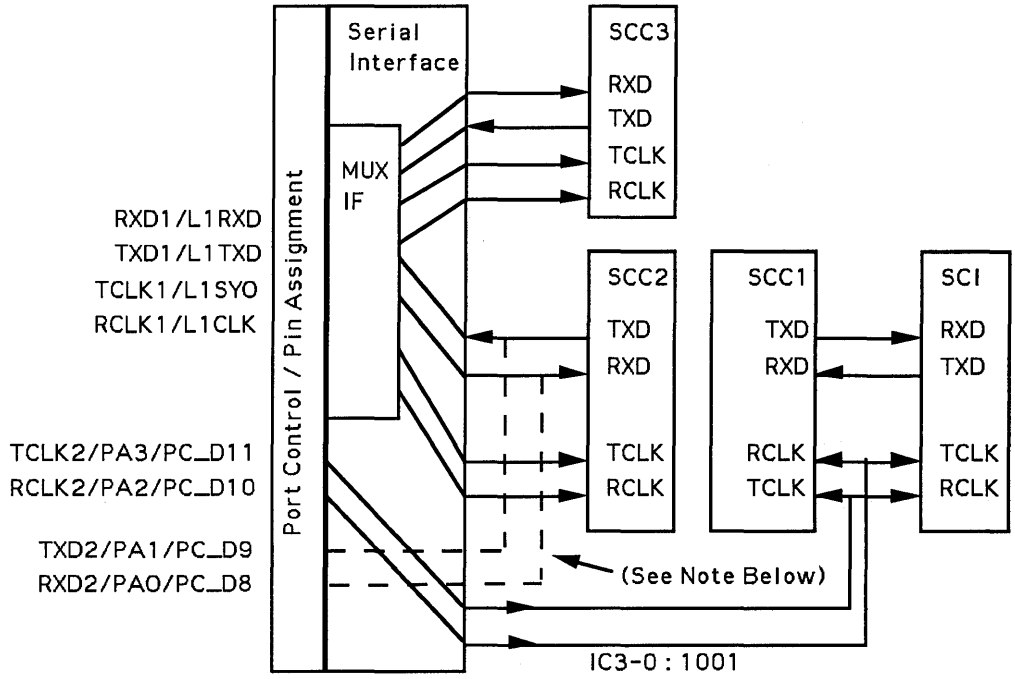
Figure 7-16. Data: SCI+ to SCC1, Clocks: NMSI1 Driving NMSI2 and SCI+

7.5.3.2.5 SCI+ Clocks with ISDN (IC3 - IC0 = 1001)

This mode is used when it is desired to clock the SCC to SCI+ direct data transfer with external clocks while still having a fully functional ISDN muxed interface. In this mode, SCC2 and SCC3 can still be used in the muxed interface and SCC1 clocks are moved to the SCC RCLK2 and TCLK2 pins to allow external sources to generate the input clock to SCC1 and the SCI+.

NOTE

The SCKP bit in the DSP SCI Control Register (SCR) must be set to one for SCI+ to SCC1 internal data transfers in this mode to prevent erroneous data transfers.



Note: If the TDM or ISDN interface is not enabled the NMSI2 data pins will remain connected to SCC2.

Figure 7-17. SCI+ Clocks with ISDN

7.5.4 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.

15 … 6	5	4	3	2	1	0
SPECIFIC MODE BITS	DIAG1	DIAG0	ENR	ENT	MODE1	MODE0

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 7-5. Transmit Data Delay (TCLK Periods)

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

NOTES:
1. RTS low values assume CTS is already asserted when RTS is asserted.
2. CTS low values assume 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 7-18 shows a diagram of synchronous mode timing from $\overline{RTS}$ low. Figure 7-19 shows a diagram of synchronous mode timing delays from $\overline{CTS}$ low.

NOTE

At least two rising edges of the receive clock are necessary after the last bit of the frame to completely transfer the last bit of receive data into the receiver. Therefore, the clock should not be gated directly with $\overline{RTS}$; at least two clocks should be added to complete the frame reception and allow the receive interrupt to be generated.

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.

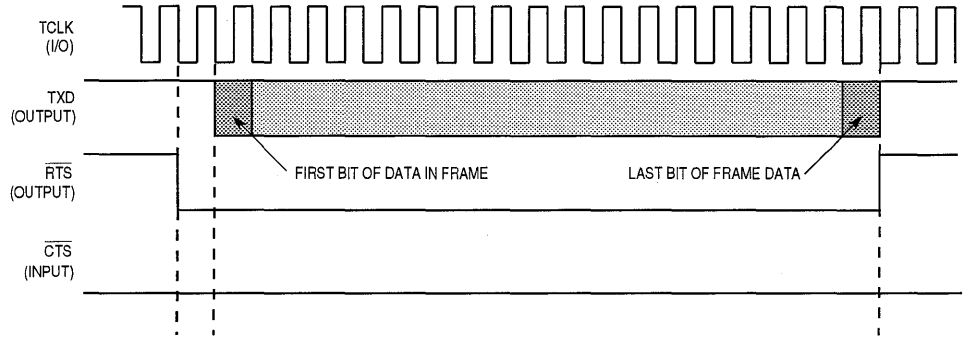
NOTE

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 de-

sired, 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.

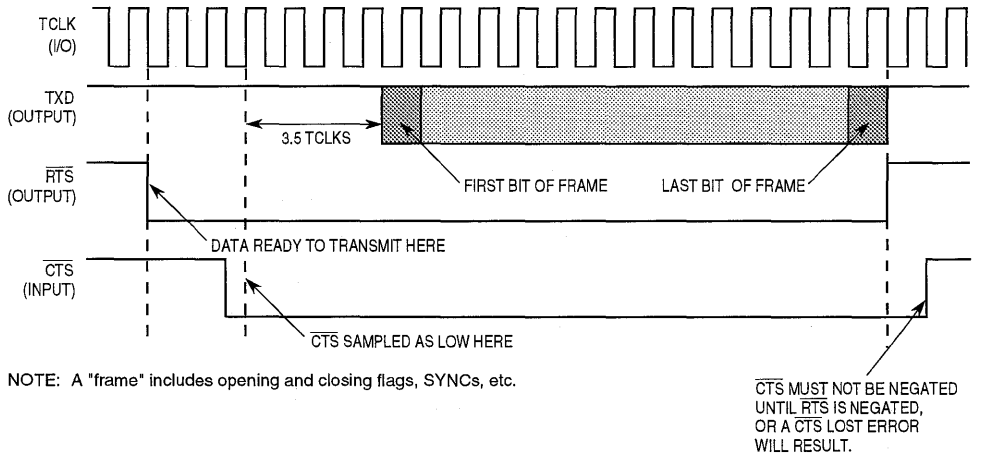
01 = Loopback mode

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 parallel I/O lines in the PACNT register. To cause the TXD and $\overline{\text{RTS}}$ 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 7-18. Output Delays from RTS Low, Synchronous Protocol



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

Figure 7-19. Output Delays from 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 IMP), 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 IMP 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 7.3 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)
10 = UART 16550 (SCC2 only)
11 = BISYNC and Promiscuous Transparent

7.5.5 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 the contents of the DSR should be written before the channel is enabled.

15	8	7	0
SYN2		SYN1	

NOTE

The DSR register has no relationship to the RS-232 signal "data set ready," which is also abbreviated DSR.

7.5.6 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 7-21). Each BD table forms a circular queue.

The format of the BDs is the same for each SCC mode of operation (HDLC, UART, UART 16550, BISYNC, 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 7-20.

	15	0
OFFSET + 0	STATUS AND CONTROL	
OFFSET + 2	DATA LENGTH	
OFFSET + 4	HIGH-ORDER DATA BUFFER POINTER (only lower 8 bits use, upper 8 bits must be 0)	
OFFSET + 6	LOW-ORDER DATA BUFFER POINTER	

Figure 7-20. SCC Buffer Descriptor Format

For frame-oriented protocols (HDLC, BISYNC), 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 an 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.

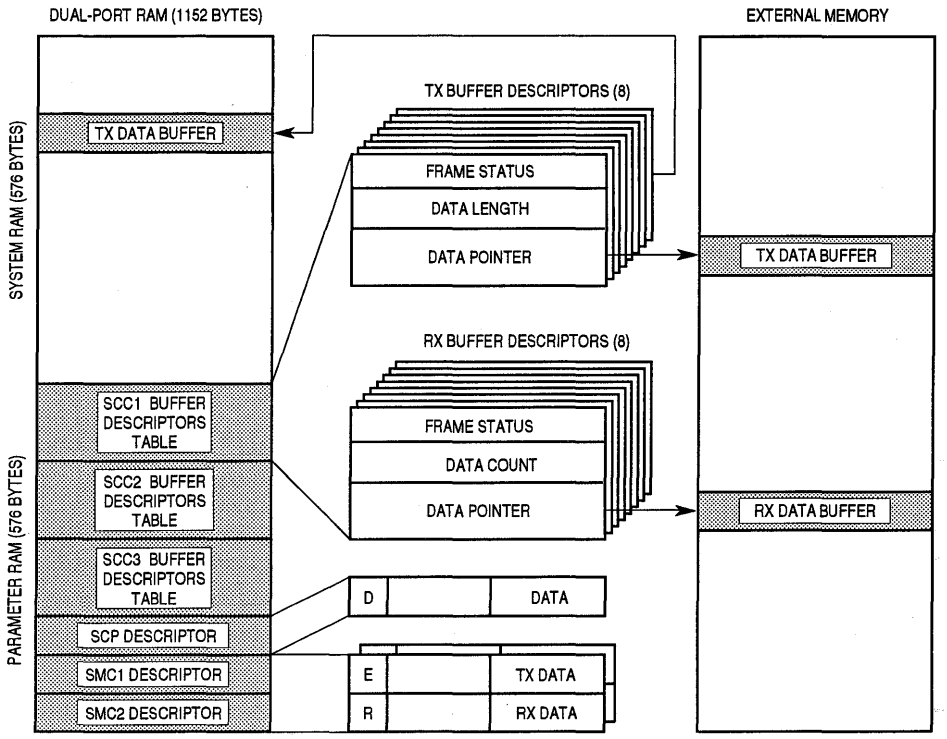


Figure 7-21. Memory Structure

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 7.5.9.4 Bus Error on SDMA Access), but this is not a major concern since the CP must be reset after any SDMA bus error.

7.5.7 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 5-3 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 7-6. Offsets $9C–$BF comprise the protocol-specific portion of the SCC parameter RAM and are discussed relative to the particular protocol chosen.

Table 7-6. 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			First Word of Protocol-Specific Area
SCC Base + BF			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 7.5.11 Disabling the SCCs for a discussion of when the SCC registers may be changed.

The registers (see Table 7-5) that typically need to be accessed by the user are described in the following paragraphs.

7.5.7.1 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

7.5.7.2 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 7.5.7 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.

7.5.7.3 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.

7.5.7.4 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.

7.5.7.5 Other General Parameters

Additional parameters are listed in Table 7-2. 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.

7.5.8 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 6.3 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 7.4.5 Serial Interface Registers).
3. Write SCON (see 7.5.2 SCC Configuration Register (SCON)).
4. Write SCM (SCC Mode) but do not set the ENT or ENR bits yet (see 7.5.4 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 7.5.7 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 6.2.5.3 Interrupt Mask Register (IMR)).
11. 1Set the ENR and/or ENT bits in SCM (see 7.4.3 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 7.5.11 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.

7.5.9 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.

7.5.9.1 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).

7.5.9.2 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.

7.5.9.3 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.

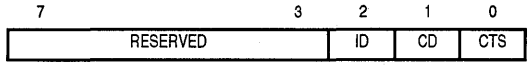
NOTE

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, regardless of what happens externally. This signifies that the corresponding SCCS bit is now valid.



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.

7.5.9.4 Bus Error on SDMA Access

- When a bus error occurs on an access by the SDMA channel, the CP generates a unique interrupt (see 6.2 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.

7.5.10 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 7.5.16 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 IMP can both receive and transmit the entire serial bit stream transparently. See 7.5.16 Transparent Controller for details.

7.5.11 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 7.5.7 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 6.7 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.

7.5.12 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 7-22.

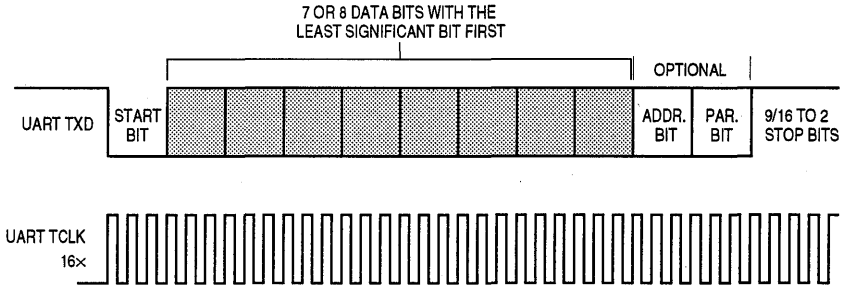


Figure 7-22. UART Frame Format

7

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 are sometimes defined to run over asynchronous links. Other protocols like PROFIBUS (see B.5 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 ($\overline{RTS}$). 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
- Flow Control Character Transmission Supported

7.5.12.1 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 any time 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 7-22. 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.

7.5.12.2 UART Memory Map

When configured to operate in UART mode, the IMP overlays the structure (see Table 7-6) onto the protocol-specific area of that SCC's parameter RAM. Refer to 5.2 System Configuration Registers for the placement of the three SCC parameter RAM areas and to Table 7-5 for the other parameter RAM values.

Table 7-7. UART Specific Parameter RAM

Address	Name	Width	Description
SCC Base + 9C #	MAX_IDL	Word	Maximum IDLE Characters (Receive)
SCC Base + 9E	IDLC	Word	Temporary Receive IDLE Counter
SCC Base + A0 #	BRKCR	Word	Break Count Register (Transmit)
SCC Base + A2 #	PAREC	Word	Receive Parity Error Counter
SCC Base + A4 #	FRMEC	Word	Receive Framing Error Counter
SCC Base + A6 #	NOSEC	Word	Receive Noise Counter
SCC Base + A8 #	BRKEC	Word	Receive Break Condition Counter
SCC Base + AA #	UADDR1	Word	UART ADDRESS Character 1
SCC Base + AC #	UADDR2	Word	UART ADDRESS Character 2
SCC Base + AE	RCCR	Word	Receive Control Character Register
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).

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 7.5.12.10 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 7.5.12.7 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 7.5.12.10 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 7.5.12.5 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 7.5.12.6 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.

7.5.12.3 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.

7.5.12.4 UART Command Set

These commands are issued to the command register described in 7.3 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 (the receive character FIFO is not affected), 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.

7.5.12.5 UART Address Recognition

In multidrop systems, more than two stations may be present on a network, with each having a specific address. Figure 7-23 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.

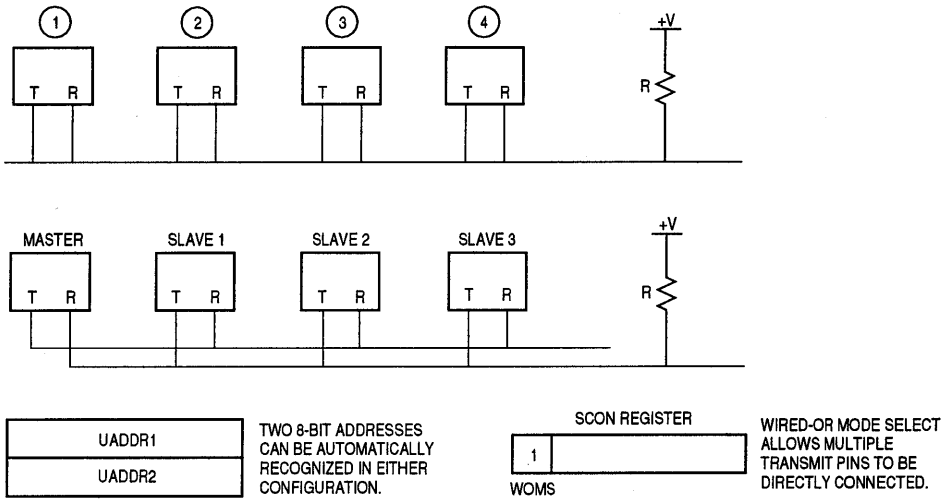


Figure 7-23. Two Configurations of UART Multidrop Operation

7.5.12.6 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 7-24.

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.

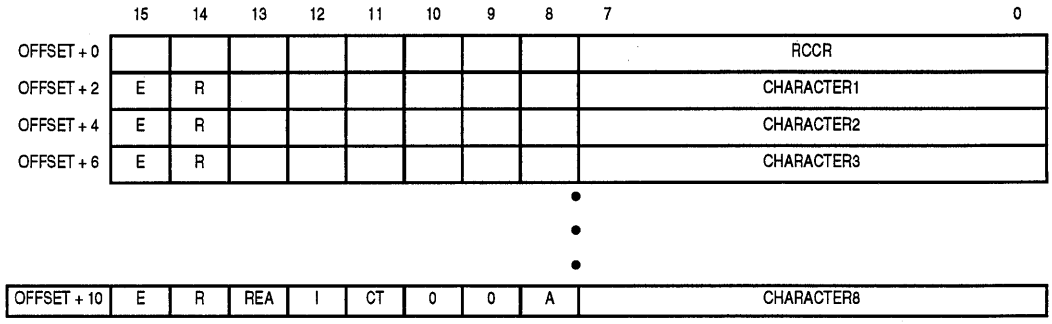


Figure 7-24. UART Control Characters Table

7

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 $\overline{\text{CTS}}$ 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 $\overline{\text{CTS}}$ signal was negated during transmission, and the CP transmits this character in the middle of buffer transmission, the $\overline{\text{CTS}}$ 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.

7.5.12.7 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 (0 through 65535) 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 char-

acters, the transmitter sends at least one high bit before transmitting any data to guarantee recognition of a valid start bit.

7.5.12.8 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.

7.5.12.9 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).

7.5.12.10 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 char-

acter 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

7.5.12.11 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:

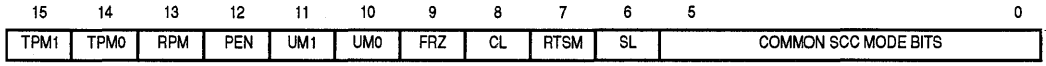
14–12 of DSR		
111	Last Transmit Stop Bit	16/16 (the default value after reset)
110	Last Transmit Stop Bit	15/16
. . .		
001	Last Transmit Stop Bit	10/16
000	Last Transmit Stop Bit	9/16

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).

7.5.12.12 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.



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 receives the address character and writes it to a new buffer. No address recognition is performed.
10 = Reserved
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 7.5.12.11 Fractional Stop Bits).

0 = One stop bit
1 = Two stop bits

COMMON SCC MODE BITS—see 7.5.4 SCC Mode Register (SCM) for a description of the DIAG1, DIAG0, ENR, ENT, MODE1, and MODE0 bits.

7.5.12.13 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 7-25. 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

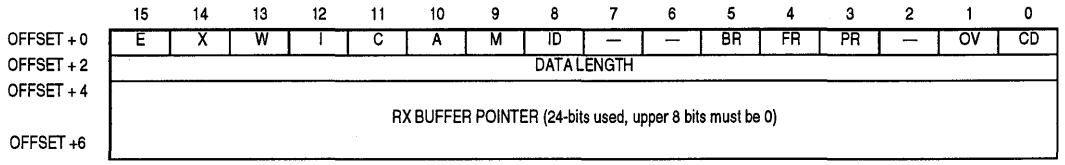


Figure 7-25. UART Receive Buffer Descriptor

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 7-26. 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.

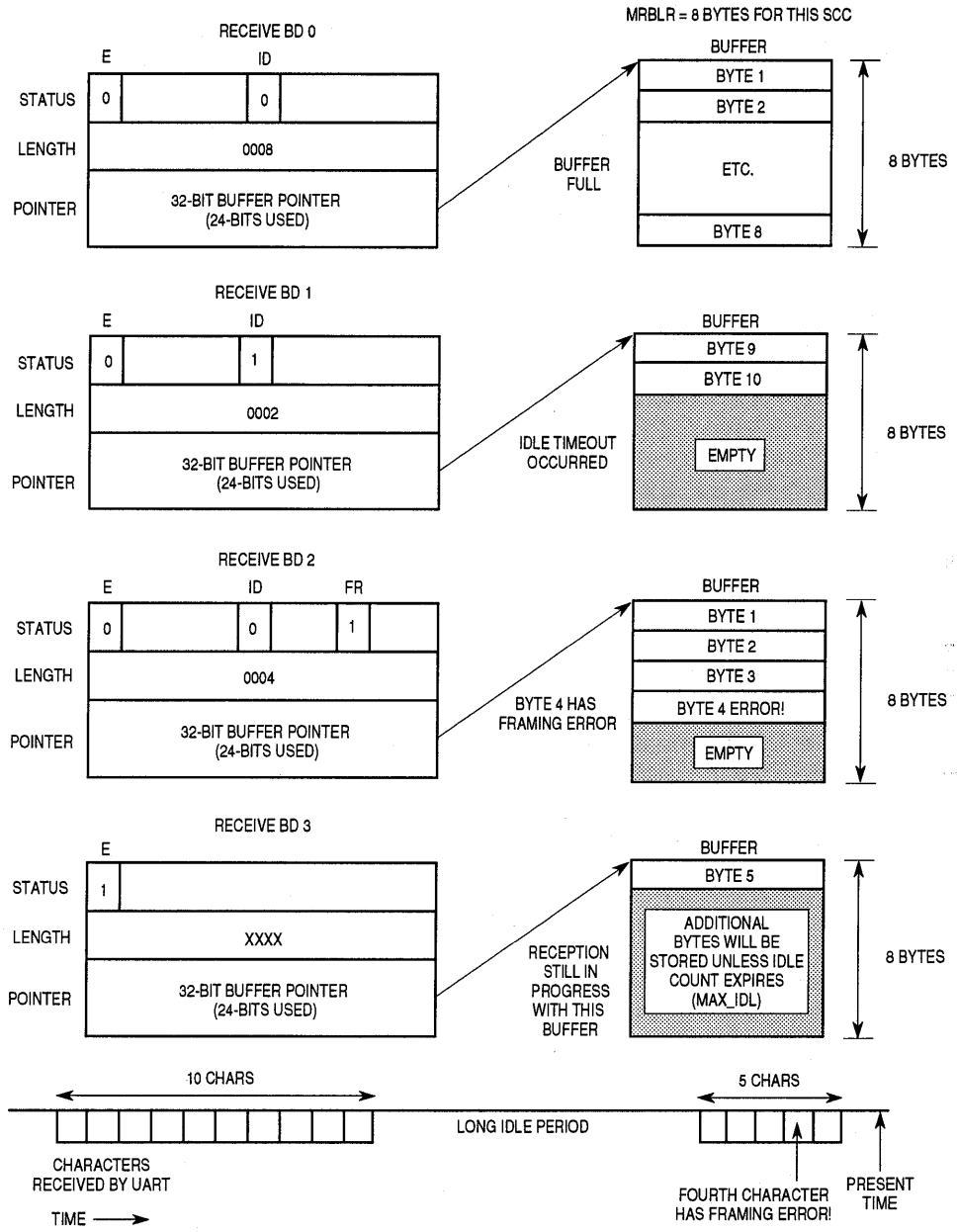


Figure 7-26. UART Rx BD Example

The first word of the Rx BD contains the 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 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.

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.

7.5.12.14 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 7-27.

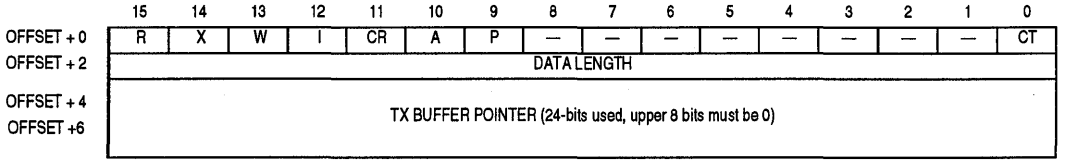


Figure 7-27. UART Transmit Buffer Descriptor

7

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

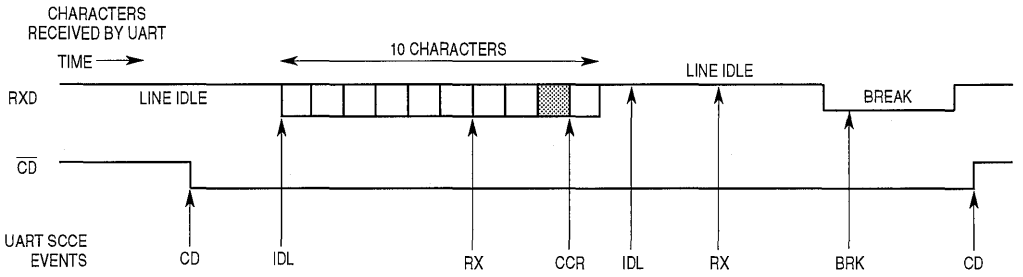
For correct operation of the function codes, the upper 8 bits of the pointer must be initialized to zero.

7.5.12.15 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.

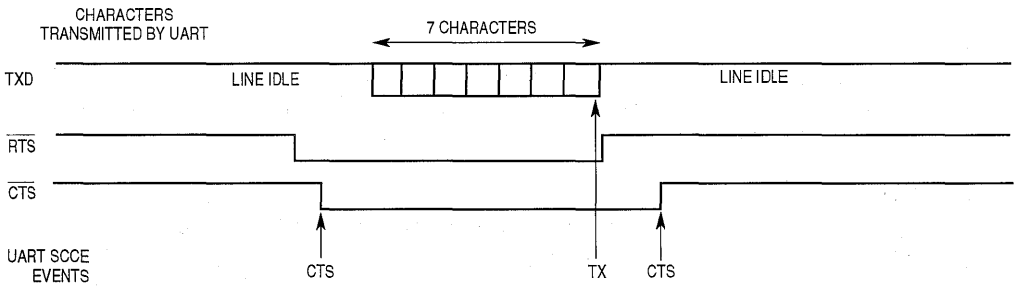
An example of the timing of various events in the UART event register is shown in Figure 7-28.



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 7-28. 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 $\overline{CTS}$ 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 $\overline{CD}$ 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.

7.5.12.16 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.

7.5.12.17 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.

7.5.13 Autobaud Controller

The autobaud function determines the baud rate and format of an asynchronous data stream starting with a known character. This controller may be used to implement the standard AT command set or other characters.

In order to use the autobaud mode, the serial communication controller (SCC) is initially programmed to BISYNC mode. The SCC receiver then 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 baud rate and compares the length to values in a user supplied lookup table. After the baud rate 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 baud rate, 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, 115.2K and 230K. To estimate the performance of the autobaud mode, the performance table in Appendix A can be used. The maximum full-duplex rate for a BISYNC channel is one-tenth of the system clock rate. So a 25 MHz IMP can support 230K autobaud rate with another low-speed channel (<50 kbps) and a 20 MHz IMP can support 115.2K autobaud rate with 2 low-speed channels. The performance can vary depending on system loading, configuration, and echoing mode.

It is important that the highest priority SCC be used for the autobaud function, since it is running at a very high rate. Any SCC that is guaranteed to be idle during the search operation of the autobaud process will not impact the performance of autobaud in an application. Idle is defined as not having any transmit or receive requests to/from the SCC FIFOs.

7.5.13.1 Autobaud Channel Reception Process

The interface between the autobaud controller and the host processor is implemented with shared data structures in the SCC parameter RAM and in external memory and through the use of a special command to the SCC.

The autobaud controller uses receive buffer descriptor number 7 (Rx BD7) for the autobaud command descriptor. This Rx BD is initialized by the host to contain a pointer to a lookup table residing in the external RAM (contains the maximum and nominal START bit length for each baud rate). The host also prepares two characters against which the autobaud controller will compare the received character (usually these characters are 'a' and 'A') and the host initializes a pointer to a buffer in external memory where the assembled characters will be stored until the host stops the autobaud process. Finally, the host initializes the SCC data synchronization register (DSR) to $7FFF in order to synchronize on the falling edge of the START bit.

Once the data structures are initialized, the host programs the SCON register to provide a sampling clock that is 16X the maximum supported baud rate. The host then issues the Enter_Baud_Hunt command and enables the SCC in the BISYNC mode.

The autobaud controller reception process begins when the START bit arrives. The autobaud controller then begins to measure the START bit length. With each byte received from the SCC that "belongs" to the START bit, the autobaud controller increments the start length

counter and compares it to the current lookup table entry. If the start length counter passed the maximum bit length defined by the current table entry, the autobaud controller switches to the next lookup table entry (the next slower baud rate). This process goes on until the autobaud controller recognizes the end of the START bit. Then, the autobaud controller starts the character assembly process.

The character assembly process uses the nominal bit length, taken from the current lookup table entry, to sample each incoming bit in it's center. Each bit received is stored to form an 8-bit character. When the assembly process is completed (a STOP bit is received), the character is compared against two user-defined characters.

If the received character does not match any of the two user defined characters, the autobaud controller re-enters the Enter_Baud_Hunt process. The host is not notified until a match is encountered.

If a match is found, the character is written to the received control character register (RCCR) with the corresponding status bit set in Rx BD7. The channel will generate the control character received (CCR) interrupt (bit 3 in the SCCE), if enabled. If the character matched, but a framing error was detected on the STOP bit, the autobaud controller will also set the framing error status bit in Rx BD7.

The autobaud controller then continues to assemble the incoming characters and to store them in the external data buffer. The host receives a CCR interrupt after each character is received.The host is responsible for determining the end of the incoming message (for example, a carriage return), stopping the autobaud process, and reprogramming the SCC to UART mode. The autobaud controller returns the nominal START bit length value for the detected baud rate from the lookup table and a pointer to the last character received that was written to the external data buffer. The host must be able to handle each character interrupt in order to determine parity and character length (this information may be overwritten when the next character interrupt is presented to the host). The host uses the two received characters to determine 1) whether a properly formed "at" or "AT" was received, and 2) the proper character format (character length, parity).

Once this is decided, three possible actions can result. First, the host may decide that the data received was not a proper "at" or "AT", and issue the Enter_Baud_Hunt command to cause the autobaud controller to resume the search process. Second, the host may decide the "at" or "AT" is proper and simply continue to receive characters in BISYNC mode. Third, the M68000 core may decide that the "at" or "AT" is proper, but a change in character length or parity is required.

7.5.13.2 Autobaud Channel Transmit Process

The autobaud microcode package supports two methods for transmission. The first method is automatic echo which is supported directly in the SCC hardware, and the second method is a smart echo or software transmit which is supported with an additional clock and software.

Automatic echo is enabled by setting the DIAG bits in the SCC mode register (SCM) to '10' and asserting the $\overline{CD}$ pin (externally on SCC1 and on SCC2 and SCC3, either externally or

by leaving the pin as a general purpose input). The ENT bit of the SCC should remain cleared. The transmitter is not used, so this echoing method does not impact performance.

The smart echo or software transmit requires use of an additional clock and the transmitter, so the overall performance could be affected if other SCCs are running. This method requires an additional clock for sampling the incoming bit stream since the baud rate generator (BRG) must be used to provide the correct frequency for transmission. The user needs to provide the sampling clock that will be used for the autobaud function on the RCLK pin (for example, a 1.8432 MHz clock for 115.2K). The clock that will be used for the SCC transmission can be provided to the BRG from the system clock or on TIN1.The TIN1 and RCLK1 pins can be tied together externally. After the first two characters have been received and character length and parity determined, the host programs the DSR to $FFFF, enables the transmitter (by setting ENT), and programs the transmit character descriptor (overlays CONTROL Character 8). The host is interrupted after each character is transmitted.

For modem applications with the 68356, SCC2 will be used as the DTE interface and autobauding to the DTE baud rate will often be required. If use of the smart echo feature is desired, the receive clock can be provided by the baud rate generator 2 (BRG2) internally by resetting the RCS bit in the SCON2 register to zero. The separate transmit clock can be provided externally to the TCLK2 pin through a hardwire connection to either of the other baud rate generator pins (BRG1 or BRG3). The TCS bit in the SCON2 register should be set to one to enable the external clock source. After autobauding is complete, both the transmit and receive clock sources can be derived internally from BRG2 and the external pin connected to TCLK2 should be three stated to assure that it does not contend with the TCLK2 pin. This can be done as follows:

1. When hardwiring BRG1, setting the TRISBRG1 bit to one in the DISC register to three state the BRG1 pin.
2. By programming the BRG3 pin to a general purpose input by resetting bit 12 in the PACNT register to zero and resetting bit 12 in the PADDR register.

7.5.13.3 Autobaud Parameter RAM

When configured to operate in the autobaud mode, the IMP overlays some entries of the UART-specific parameter RAM as illustrated in Table 7-8.

Table 7-8. Autobaud Specific Parameter

Address	Name	Width	Description
SCC Base + 9C *	MAX_IDL	Word	Maximum IDLE Characters
SCC Base + 9E	MAX_BIT	Word	Current Maximum START Bit Length
SCC Base + A0	NOM_START	Word	Current Nom. START Bit (used to determine baud rate)
SCC Base + A2 *	PAREC	Word	Receive Parity Error Counter
SCC Base + A4 *	FRMEC	Word	Receive Framing Error Counter
SCC Base + A6 *	NOSEC	Word	Receive Noise Counter
SCC Base + A8 *	BRKEC	Word	Receive Break Error Counter

Table 7-8. Autobaud Specific Parameter

Address	Name	Width	Description
SCC Base + AA *	ABCHR1	Word	User Defined Character1
SCC Base + AC *	ABCHR2	Word	User Defined Character2
SCC Base + AE	RCCR	Word	Receive Control Character Register
SCC Base + B0 *	CHARACTER1	Word	CONTROL Character1
SCC Base + B2 *	CHARACTER2	Word	CONTROL Character2
SCC Base + B4 *	CHARACTER3	Word	CONTROL Character3
SCC Base + B6 *	CHARACTER4	Word	CONTROL Character4
SCC Base + B8 *	CHARACTER5	Word	CONTROL Character5
SCC Base + BA *	CHR6/RxPTR	Word	CONTRChar6/MSW of pointer to external Rx Buffer
SCC Base + BC *	CHR7RxPTR	Word	CONTRChar7/LSW of pointer to external Rx Buffer
SCC Base + BE *	CHR8/TxBD	Word	CONTROL Character8/Transmit BD

* These values should be initialized by the user (M68000 core).

Note the new parameters that have been added to the table. They are MAX_BIT, NOM_START, ABCHR1, ABCHR2, RxPTR (2 words), and TxBD. These parameters are of special importance to the autobaud controller. They must be written prior to issuing the Enter_Baud_Hunt command.

When the channel is operating in the autobaud hunt mode, the MAX_BIT parameter is used to hold the current maximum START bit length. The NOM_START location contains the current nominal start from the lookup table. After the autobaud is successful and the first character is matched, the user should use the NOM_START value from the autobaud specific parameter RAM to determine which baud rate from the lookup table was detected. Also the Tx internal data pointer (at offset SCC Base + 94) will point to the last character received into external data buffer.

NOTE

When the channel is operating in the UART mode, the NOM_START_/BRKCR is used as the break count register and must be initialized before a STOP_TRANSMIT command is issued.

The characters ABCHR1 and ABCHR2 are the autobaud characters that should be searched for by the autobaud controller. Typically these are 'a' and 'A' (i.e. $0061 and $0041) if using the Hayes command set. These characters must be odd in order for the autobaud controller to correctly determine the length of the START bit. Characters are transmitted and received least significant bit first, so the autobaud controller detects the end of the START bit by the least significant bit of the character being a '1'.

The RxPTR is a 2 word location that contains a 32-bit pointer to a buffer in external memory used for assembling the received characters and must be initialized before the Enter_Baud_Hunt command is issued.

NOTE

Since a length for this external buffer is not given, the user must provide enough space in memory for characters to be assembled and written until the autobaud process is to avoid overwriting other data in memory. This location is not used as the CHARACTER7 value in the control character table until the channel operates in normal UART mode. After reception begins in normal UART mode (i.e. the "a" or "A" is found), this entry is available again as a control character table entry.

The TxBD entry is used as the transmit character descriptor for smart echo or software transmit. This location is not used as the CHARACTER8 value in the control character table until the channel operates in normal UART mode. After reception begins in normal UART mode (i.e. the "a" or "A" is found), this entry is available again as a control character table entry.

7.5.13.4 Autobaud Programming Model

The following sections describe the details of initializing the autobaud microcode, preparing for the autobaud process, and the memory structures used.

7.5.13.4.6 Preparing for the Autobaud Process

The host begins preparation for the autobaud process with the following steps. Steps 1 and 2 are required if the SCC has been used after reset or after UART mode in order to re-enable the process.

1. Disable the SCC by clearing the ENR and ENT bits. (The host may wish to precede this action with the STOP_TRANSMIT commands to abort transmission in an orderly way).
2. Issue the ENTER_HUNT_MODE command to the SCC (This ensures that an open buffer descriptor is closed).
3. Set up all the autobaud parameters in the autobaud specific parameter RAM shown in Table 7-8, the autobaud command descriptor shown in Table 7-9, and the lookup table shown in Table 7-10. Of these three areas, the autobaud controller only modifies the autobaud specific parameter RAM and the first word of the autobaud command descriptor during its operation.
4. Write the SCON to configure the SCC to use the baud rate generator clock of 16x the maximum supported baud rate. A typical value is $4000 assuming a 1.8432 MHz clock rate on TIN1 and a maximum baud rate of 115.2K, but this can change depending on the maximum baud rate and the EXTAL frequency.
5. Write the DSR of the SCC with the value $7FFF in order to detect the START bit.
6. The host initiates the autobaud search process by issuing the Enter_Baud_Hunt command

7. Write the SCM of the SCC with $1133 to configure it for BISYNC mode, with the REVD and RBCS bits set, software operation mode, and the transmitter disabled. After a few characters have been received, the transmitter can be enabled, and the software echo function may be performed after issuing the RESTART TRANSMIT command.

In general, the autobaud controller uses the same data structure as that of the UART controller. The first character (if matched) is stored in the receiver control character register and the external data buffer, and the status of that character is reported in the autobaud command descriptor. After the first character, each incoming character is then stored in the buffer pointed to by RxPTR, and the status is updated in the autobaud command autobaud descriptor. The Tx internal data pointer (at offset SCC Base + 94) is updated to point to the last character stored in the external data buffer.

7.5.13.4.7 Enter_Baud_Hunt Command

This command instructs the autobaud controller to begin searching for the baud rate of a user predefined character. Prior to issuing the command the M68000 prepares the autobaud command descriptor to contain the lookup table size and pointer.

The Enter_Baud_Hunt uses the GCI command with opcode = 11, and the channel number set for the corresponding SCC. For example, with SCC1, the value written to the command register would be $61.

7.5.13.4.8 Autobaud Command Descriptor

The autobaud controller uses the receive buffer descriptor number 7 (Rx BD7) as an autobaud command descriptor. The autobaud command descriptor is used by the M68000 core to transfer command parameters to the autobaud controller, and by the autobaud controller to report information concerning the received character.

The structure of the autobaud command descriptor for the autobaud process is shown in Table 7-9. The first word of the descriptor or the status word is updated after every character is received.

Offset	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0						FE	M2	M1					EOT		OV	CD
2	Lookup Table Size															
4	**Function Code**															
8	Lookup Table Pointer															

Table 7-9. Autobaud Command Descriptor

FE – Framing Error (Bit 10)

If this bit is set, a character with a framing error was received. A framing error is detected by the autobaud controller when no STOP bit is detected in the received data. FE will be set for a 9-bit character (8 bits + parity) if the parity bit is '0'.

NOTE

The user must clear this bit when it is set.

M2 – Match Character2 (Bit 9)

When this bit is set, the character received matched the User Defined Character 2. The received character is written into the receive control character register (RCCR).

M1 – Match Character1 (Bit 8)

When this bit is set, the character received matched the User Defined Character 1. The received character is written into the receive control character register (RCCR).

EOT – End Of Table (bit 3)

When this bit is set, the autobaud controller measured start length exceeded the maximum start length of the last entry in the lookup table (lowest baud rate).

NOTE

The user must clear this bit when it is set.

OV – Overrun (bit 1)

If this bit is set, a receiver overrun occurred during autobaud reception.

NOTE

The user must clear this bit when it is set.

CD – Carrier Detect Lost (bit 0)

If this bit is set, the carrier detect signal was negated during autobaud reception.

NOTE

The user must clear this bit when it is set.

Lookup Table Size - Lookup table size is the number of baud rate entries in the external lookup table.

Lookup Table Pointer - The lookup table pointer is the address in the external RAM where the lookup table begins.

NOTE

The lookup table cannot cross a 64k memory block boundary.

7.5.13.4.9 Autobaud LookUp Table

The autobaud controller uses an external lookup table to determine the baud rate while in the process of receiving a character. The lookup table contains two entries for each supported baud rate. The first entry is the maximum start length for the particular baud rate, and the second entry is the nominal length for a 1/2 START bit.

To determine the two values for each table entry, first calculate the autobaud sampling rate (EQ 2). To do this EQ 1 must be used until EQ 2 is satisfied. The sampling rate is the lowest speed baud rate that can be generated by the SCC baud rate generator that is over a threshold defined in EQ 2.

BRG Clk Rate = System Clock or TIN1 / ((Clock Divider bits in SCON) + 1) (EQ 1)

assuming that the DIV bit in SCON is set to 0, (otherwise an additional "divide-by-4" must be included).

Sampling Rate = BRG Clk Rate, where BRG Clk rate >= (Max Desired UART Baud Rate) x 16 (EQ 2)

For instance, if a 115.2K baud rate is desired, with a 16.67 MHz system clock, the minimum sampling rate possible is 1.843 MHz = 115.2K x 16. This exact frequency can be input to RCLK1 or TIN1 as the sample clock. If the system clock is to be used, a 16.67 MHz system clock cannot produce an exact baud rate clock of 1.843 MHz. The lowest one that can be used is Baud Rate = 16.67 MHz / (7+1) = 2.083 MHz. Thus, 2.083 MHz is the sampling rate, and the SCON should be set to $000E to produce this.

Once the sampling rate is known, the other two equations follow easily. The maximum START bit length is calculated by the following equation:

Maximum start length = (Sampling Rate/Recognized baud rate) x 1.05 (EQ 3)

Thus, for the first entry in the table, the maximum start length is 1.8432 Mhz/115200 x 1.05 = 17 for an external sample clock. The value 1.05 is a suggested margin that allows characters 5% larger than the nominal character rate to be accepted. In effect, the margin determines the "split point" between what is considered to be a 56.7K character rate and what is a 38.4K character rate. The margin should not normally be less than 1.03 due to clocking differences between UARTs.

The nominal START bit length is calculated by:

Nominal start length = (Sampling Rate/Recognized baud rate) / 2 (EQ 4)

For the 115.2K example in the first table entry, this would be 1.8432 MHz/115.2K/2 = 8.

The structure of the lookup table is shown in Table 7-10. The table starts with the maximum UART baud rate supported and ends with the minimum UART baud rate supported.

Table 7-10. Autobaud Lookup Table Format

OFFSET from Lookup Table Pointer	DESCRIPTION
0	Maximum Start Length
2	Nominal Start Length
4	Maximum Start Length
6	Nominal Start Length
•	Maximum Start Length
•	Nominal Start Length
(Lookup Table Size - 1) * 4	Maximum Start Length
[(Lookup Table Size - 1) * 4] + 2	Nominal Start Length

NOTE

If less margin is used in the calculation of the maximum start length above, it is possible to distinguish between close UART rates such as 64K and 57.6K. However variations in RS232 drivers of up to 4%, plus nominal clocking rate variations of 3%, plus the fact that the sampling rate may not perfectly divide into the desired UART rate, can make this distinction difficult to achieve in some scenarios.

LOOKUP TABLE EXAMPLE.

Table 7-11 is an example autobaud lookup table. The maximum start and nominal start values are derived assuming a 1.8432 MHz sampling clock on TIN1 or RCLK and a shift factor of 5%.

Table 7-11. Lookup Table Example

Desired Baud Rate	Maximum Start	Nominal Start
115200	17	8
57600	34	16
38400	50	24
28800	67	32
19200	101	48
14400	134	64
12000	161	77
9600	202	96
7200	269	128
4800	403	192
2400	806	384
1200	1613	768
600	3226	1536
300	6451	3072
110	17594	8378

7.5.13.5 DETERMINING CHARACTER LENGTH AND PARITY. Table 7-12 shows the different possible character lengths and parity that will be discussed. The following paragraphs will discuss for each case how to determine the parity.

Table 7-12. Character Lengths and Parity Cases

Case #	Character Length	Parity	Notes
1	7-bit	No parity, 1 STOP bit	Not Supported
2	7-bit	Even parity Odd parity Parity=1 Parity=0	Parity is indicated by the most significant bit of the byte
3	8-bit	No parity	Same as 7-bit, parity=0
4	8-bit	Even parity Odd parity Parity=0	Parity is indicated by which characters generate a FE interrupt
5	8-bit	Parity=1	Not Supported

- Case 1– This case cannot be supported because the autobaud can not separate the first character from the second character.
- Case 2– As each character is assembled, it is stored into a complete byte. Assuming that the characters are ASCII characters with 7-bit codes, the 8th bit of the byte will contain the parity bit. If the parity is either even or odd, then after receiving an odd character and an even character, the 8th bit should be different for the odd and even characters. The parity can be determined by the setting of the parity bit for one of the two characters. If the 8th bit is always a 1, this is the same as a 7-bit character, no parity and at least 2 STOP bits or a 7-bit character with force 1 parity. If the 8th bit is always a zero, then either the character is a 7-bit character with force 0 parity, or the character is a 8-bit character with no parity.
- Case 3– This case is the same as 7-bit character with force 0 parity. The 8th bit of the byte will always be zero.
- Case 4–This case assumes a 8-bit character with the 8th bit of the character equal to a 0 (ASCII character codes define the 8th bit as zero). If the parity is either even or odd, then after receiving an odd and an even character, a framing error (FE) interrupt should have been generated for one of them (the interrupt is generated when the parity bit is zero). The user can determine the parity by which character generated a FE interrupt (if the odd character did, then the parity is odd). If a framing error occurs on every character, then the character is 8-bits with force 0 parity. If no framing error occurs, than this is the same as Case 5.
- Case 5– This case is not supported, because it can not be differentiated from 7-bit force 0 parity and 8-bit no parity. If the 9th bit is a 1, then it will be interpreted as a STOP bit.

7.5.13.6 AUTOBAUD RECEPTION ERROR HANDLING PROCEDURE. The autobaud controller reports reception error conditions using the autobaud command descriptor. Three types of errors are supported:

- Carrier Detect Lost during reception

When this error occurs and the channel is not programmed to control this line with software, the channel terminates reception, sets the carrier detect lost (CD) bit in the command descriptor, and generates the CCR interrupt, if enabled. CCR is bit 3 of the SCCE register.

- Overrun Error

When this error occurs, the channel terminates reception, sets the overrun (OV) bit in the command descriptor, and generates the CCR interrupt, if enabled.

- End Of Table Error

When this error occurs, the channel terminates reception, sets the end of table (EOT) bit in the command descriptor, and generates the CCR interrupt, if enabled.

Any of these errors will cause the channel to abort reception. In order to resume autobaud operation after an error condition, the M68000 should clear the status bits and issue the Enter_Baud_Hunt command again.

7.5.13.7 AUTOBAUD TRANSMISSION. The autobaud package supports two methods for echoing characters or transmitting characters. The two methods are automatic echo and smart echo.

7.5.13.7.1 Automatic Echo. This method uses the SCC hardware to automatically echo the characters back on the TxD pin. The automatic echo is enabled by setting the DIAG bits in the SCM to '10'. The transmitter should not be enabled. The hardware echo is done automatically. The $\overline{CD}$ pin needs to be asserted in order for the characters to be transmitted back. On SCC1, the external $\overline{CD}$ pin must be tied low. On SCC2 and SCC3, either the external $\overline{CD}$ pin must be tied low or the $\overline{CD}$ pins should be left configured as general purpose input pins (the $\overline{CD}$ signal to the SCC is then connected to ground internally).

Using the automatic echo, the receiver still autobauds correctly and performance is not affected. The SCC echoes the received data with a few nanoseconds delay.

7.5.13.7.2 Smart Echo. This method requires addition hardware and software to implement. The user must provide two clock sources. One clock source is the sample clock which is input on RCLK and cannot be divided down. The BRG is used to divide the second clock down to provide the clock used for transmit. The second clock can be either the system clock or a clock connected to TIN1. The TIN1 and RCLK pins can be connected to each other externally.

After the first character is received, the user must take the following steps:

1. Determine the baud rate from the returned NOM_START value and program SCON to (input frequency/baud rate)-1, where the input frequency is either the system clock or the clock on TIN1.
2. Program the DSR to $FFFF. The DSR will need to be programmed back to $7FFF before the Enter_Baud_Hunt command is issued again.
3. Set the ENT bit in the mode register.

4. Program the transmit character BD as show in Table 7-13.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		CL		PE	PM			CHAR							

Table 7-13. Transmit Character BD

R (ready bit)
- 0 = Character is not ready
- 1 = Character is ready to transmit

CL (character len)
- 0 = 7 bits + parity or 8 bits with no parity
- 1 = 8 bits + parity

PE (parity enable)
- 0 = No parity
- 1 = Parity

PM (parity mode)
- 0 = Even parity
- 1 = Odd parity

The autobaud controller issues a Tx interrupt after each character is transmitted.

7.5.13.8 REPROGRAMMING TO UART MODE OR ANOTHER PROTOCOL. The following steps should be followed in order to switch the SCC from autobaud to UART mode or to another protocol.

- Disable the SCC by clearing ENR and ENT.
- Issue the Enter_Hunt_Mode command.
- Initialize the SCC parameter RAM (specifically, the Rx and Tx internal states and the words containing the Rx and Tx BD#s) to the state immediately after reset and initialize the protocol specific parameter area for the new protocol.
- Re-enable the SCC with the new mode.

7.5.14 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 7-29.

OPENING FLAG	ADDRESS	CONTROL	INFORMATION (OPTIONAL)	CRC	CLOSING FLAG
8 BITS	16 BITS	8 BITS	8N BITS	16 BITS	8 BITS

Figure 7-29. 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 7.5.2 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

7.5.14.1 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.

7.5.14.2 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 exceeds 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.

7.5.14.3 HDLC Memory Map

When configured to operate in HDLC mode, the IMP overlays the structure shown in Table 7-7 onto the protocol-specific area of that SCC parameter RAM. Refer to 5.2 System Configuration Registers for the placement of the three SCC parameter RAM areas and to Table 7-2 for the other parameter RAM values.

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.

Table 7-14. HDLC-Specific Parameter RAM

Address	Name	Width	Description
SCC Base + 9C	RCRC_L	Word	Temp Receive CRC Low
SCC Base + 9E	RCRC_H	Word	Temp Receive CRC High
SCC Base + A0 #	C_MASK_L	Word	Constant ($F0B8 16-Bit CRC, $DEBB 32-Bit CRC)
SCC Base + A2 #	C_MASK_H	Word	Constant ($XXXX 16-Bit CRC, $20E3 32-Bit CRC)
SCC Base + A4	TCRC_L	Word	Temp Transmit CRC Low
SCC Base + A6	TCRC_H	Word	Temp Transmit CRC High
SCC Base + A8 #	DISFC	Word	Discard Frame Counter
SCC Base + AA #	CRCEC	Word	CRC Error Counter
SCC Base + AC #	ABTSC	Word	Abort Sequence Counter
SCC Base + AE #	NMARC	Word	Nonmatching Address Received Counter
SCC Base + B0 #	RETRC	Word	Frame Retransmission Counter
SCC Base + B2 #	MFLR	Word	Max Frame Length Register
SCC Base + B4	MAX_cnt	Word	Max_Length Counter
SCC Base + B6 #	HMASK	Word	User-Defined Frame Address Mask
SCC Base + B8 #	HADDR1	Word	User-Defined Frame Address
SCC Base + BA #	HADDR2	Word	User-Defined Frame Address
SCC Base + BC #	HADDR3	Word	User-Defined Frame Address
SCC Base + BE #	HADDR4	Word	User-Defined Frame Address

Initialized by the user (M68000 core).

7

7.5.14.4 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 and BISYNC 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 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.

7.5.14.5 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 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.

7.5.14.6 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 7-30.

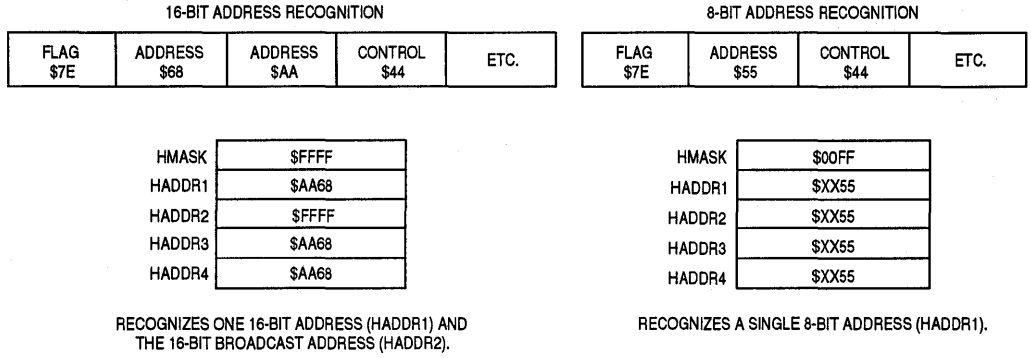


Figure 7-30. HDLC Address Recognition Examples

7.5.14.7 HDLC Maximum Frame Length Register (MFLR)

7

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.

7.5.14.8 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.

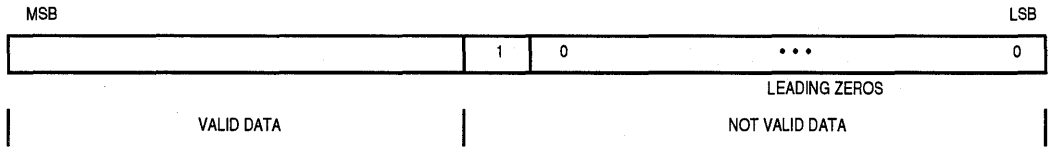
NOTE

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.

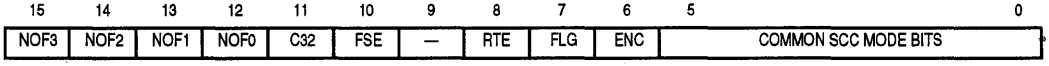
Error Counters

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)

7.5.14.9 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.



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 $\overline{CTS}$ lost condition happens on the first or second buffer of the frame. See 7.5.14.8 HDLC Error-Handling Procedure.

FLG—Transmit Flags/Idles between Frames and Control the RTS Pin

0 = Send ones between frames; $\overline{RTS}$ is negated between frames. If NOF–NOF0 is greater than zero, $\overline{RTS}$ 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. $\overline{RTS}$ 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 7.5.4 SCC Mode Register (SCM) for a description of the DIAG1, DIAG0, ENR, ENT, MODE1, and MODE0 bits.

7.5.14.10 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 7-31.

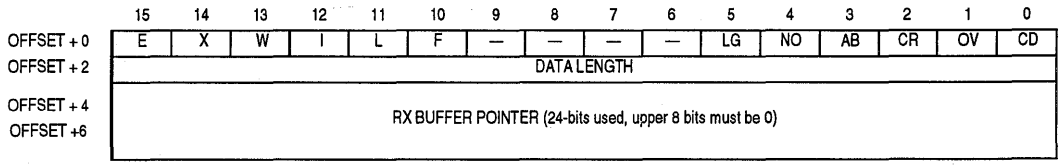


Figure 7-31. HDLC Receive Buffer Descriptor

An example of the HDLC receive process is shown in Figure 7-32. 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.

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.

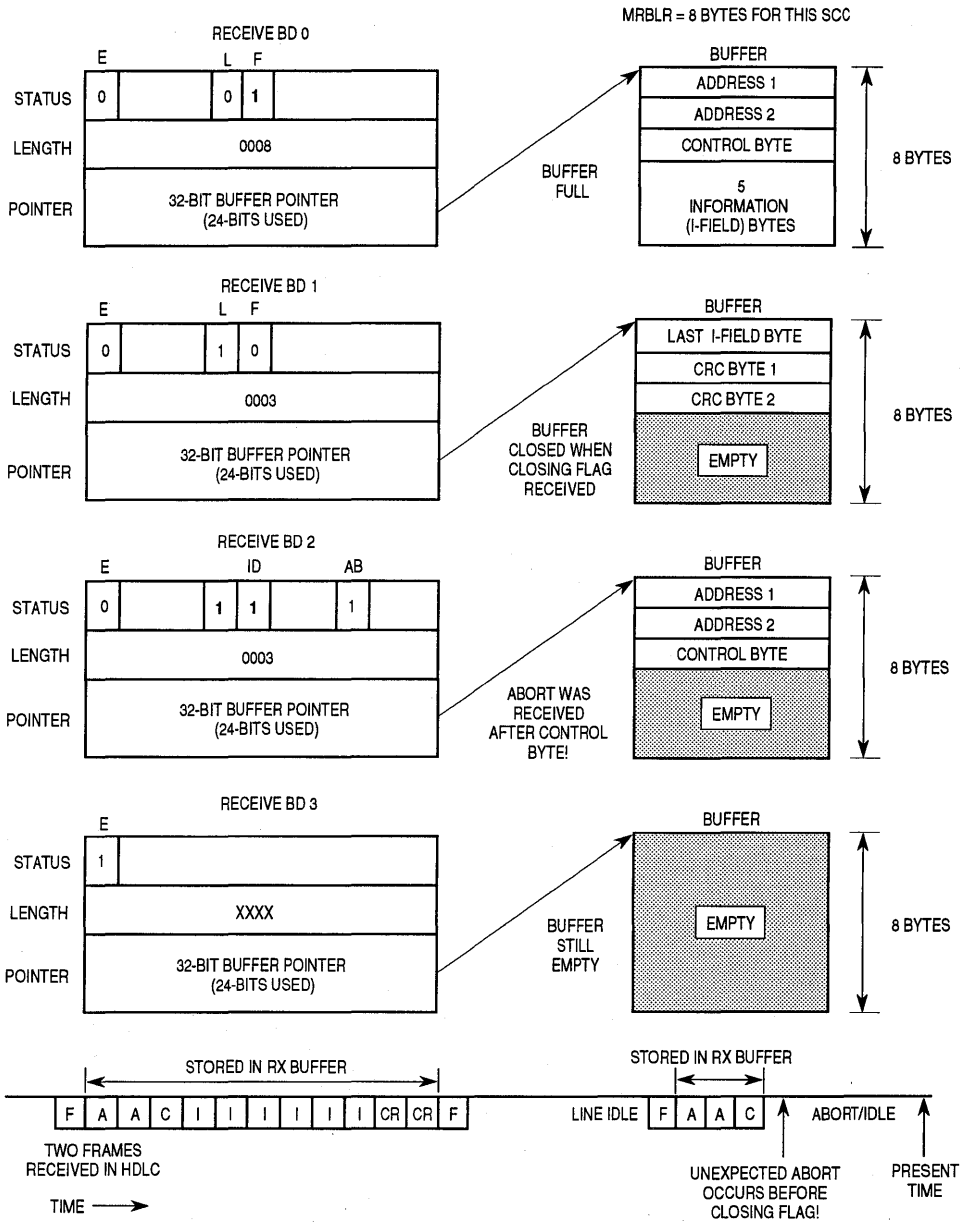


Figure 7-32. HDLC Receive BD Example

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.

7.5.14.11 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 7-33.

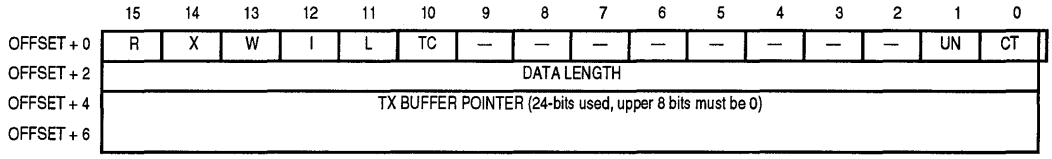


Figure 7-33. HDLC Transmit Buffer Descriptor

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

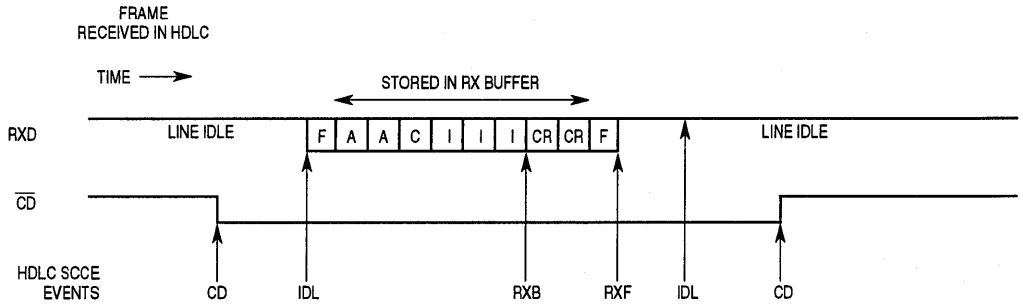
For correct operation of the function codes, the upper 8 bits of the pointer must be initialized to zero.

7.5.14.12 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.

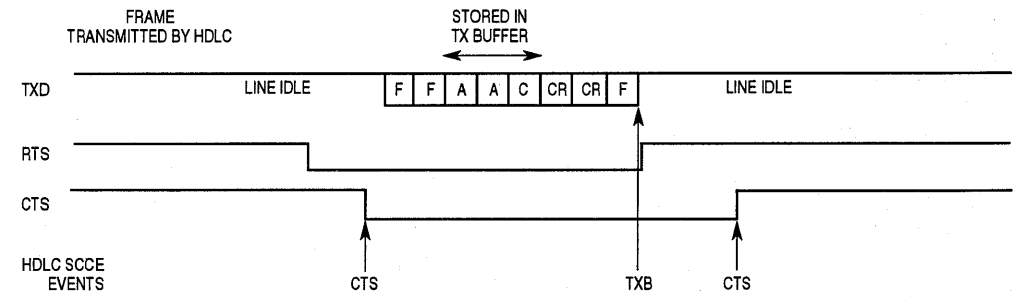
The HDLC event register is a memory-mapped register that may be read at any time. A bit is cleared by writing a one; writing a zero does not affect a bit's value. More than one bit may be cleared at a time. All unmasked bits must be cleared before the CP will clear the internal interrupt request. This register is cleared at reset.

An example of the timing of various events in the HDLC event register is shown in Figure 7-34.



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 7-34. 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 $\overline{\text{CTS}}$ 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 (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.

7.5.14.13 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.

7.5.15 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 7-35). The transparent mode in BISYNC allows full binary data to be transmitted; with any possible character pattern is 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	TRANSPARENTTEXT	DLE	ETX	BCC

Figure 7-35. 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 IMP.)

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 ($\overline{CD}$), clear to send ($\overline{CTS}$), and request to send ($\overline{RTS}$). 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 7.5.2 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

7.5.15.1 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 7.5.15.8 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 7.5.16 Transparent Controller.

7.5.15.2 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 7-36.

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 7.5.16 Transparent Controller.

7.5.15.3 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 5.2 System Con-

figuration Registers for the placement of the three SCC parameter RAM areas and Table 7-5 for the other parameter RAM values.

Table 7-15. 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 7.5.5 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 7.5.15.14 Programming the BISYNC Controllers for more information.

7.5.15.4 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 command 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.

7.5.15.5 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 7.5.15.14 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 7-36. 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 7-36. 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.

7.5.15.6 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 (the V bit must be set).

When using 7-bit characters with parity, the parity bit should be included in the SYNC register value.

15	14						8	7	0
V	0	0	0	0	0	0	0	SYNC	

7.5.15.7 BDLE-BISYNC DLE Register

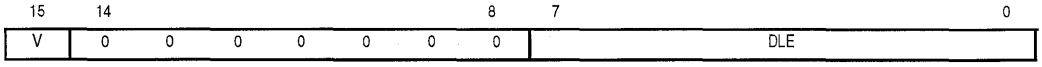
The 16-bit, memory-mapped, read-write BDLE register is used to define the BISYNC stripping and insertion of the DLE character. When the BISYNC controller is in transparent mode

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 excludes 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.



7.5.15.8 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 mes-

sage 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)

7.5.15.9 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 $\overline{CD1}$/L1SY1 pin, if SCC1 is used, and the $\overline{CD2}$ and $\overline{CD3}$ 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 $\overline{CD}$ pins (and the $\overline{CD}$ timing) are used to synchronize the data. $\overline{CD}$ 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 7.5.16 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.

7

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 7.5.4 SCC Mode Register (SCM) for a description of the DIAG1, DIAG0, ENR, ENT, MODE1, and MODE0 bits.

7.5.15.10 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 7-37. 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 LENG															
OFFSET +4	RX BUFFER POINTER (24-bits used, upper 8 bits must be 0)															
OFFSET + 6																

Figure 7-37. 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.

7.5.15.11 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 7-38.

	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 7-38. 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 DLE and inserts an extra DLE if the V-bit is set.

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.

7.5.15.12 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 recognized 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.

RX—Rx Buffer

A complete buffer has been received on the BISYNC channel.

7.5.15.13 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.

7.5.15.14 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.

7.5.16 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 IMP 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 7.5.15 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 IMP 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 7.5.2 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

7.5.16.1 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 7.5.16.5 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. $\overline{\text{RTS}}$ 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.

7.5.16.2 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 7.5.16.5 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.

7.5.16.3 Transparent Memory Map

When configured to operate in transparent mode, the IMP overlays the structure illustrated in Table 7-16 onto the protocol specific area of that SCC parameter RAM. Refer to 5.2 System Configuration Registers for the placement of the three SCC parameter RAM areas and Table 7-5 for the other parameter RAM values.

Table 7-16. 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

Table 7-16. Transparent-Specific Parameter RAM

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 7-5).

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 7.5.5 SCC Data Synchronization Register (DSR)) must be initialized with the SYN1–SYN2 synchronization characters. See 7.5.16.5 Transparent Synchronization for details.

7.5.16.4 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 7.5.16.5 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.

7.5.16.5 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.

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 $\overline{\text{CTS}}$ pin is ignored, and the $\overline{\text{CD}}$ pin is used to control both transmission and reception. For the transmitter, once $\overline{\text{RTS}}$ is asserted and the transmitter samples $\overline{\text{CD}}$ 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 $\overline{\text{CD}}$ as low. Note that $\overline{\text{CD}}$ is sampled on a rising TCLK for the transmitter and a rising RCLK for the receiver.
   Once $\overline{\text{CD}}$ is sampled as low by the receiver and transmitter, further toggling of $\overline{\text{CD}}$ 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.

7.5.16.6 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.

7.5.16.7 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 7.5.4 SCC Mode Register (SCM) for a description of the DIAG1, DIAG0, ENR, ENT, MODE1, and MODE0 bits.

7.5.16.8 Transparent Receive Buffer Descriptor (RxBD)

The CP reports information about the received data for each buffer using BD. The RxBD is shown in Figure 7-39. 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

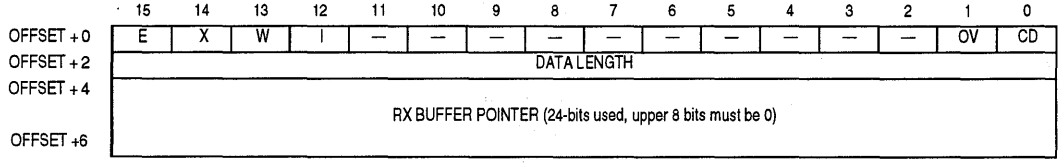


Figure 7-39. Transparent Receive Buffer Descriptor

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 MR-BLR.

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.

7.5.16.9 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 7-40.

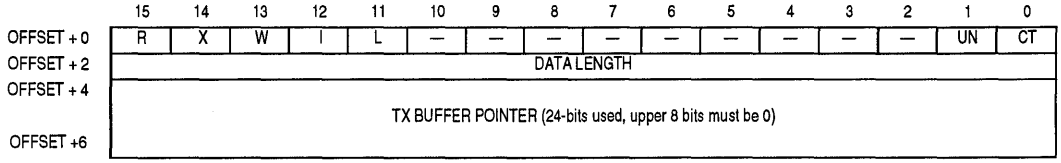


Figure 7-40. Transparent Transmit Buffer Descriptor

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.

7

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.

7.5.16.10 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.

7.5.16.11 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.

7.6 16550 EMULATION CONTROLLER

7.6.1 16550 Emulation Controller Features

- Contains PC port side that implements the full register set, external pins, external pin timing, and interrupt structure of the 16550.
- The 16550 UART registers are accessible from the PCMCIA port in any page of I/O space.
- The emulation controller can optionally scale the transfer rates to and from the PC to match the speeds of a real 16550.

- The interface to the 68000 core maintains the standard 68302 programming model of a normal SCC in UART mode.
- Data is transferred between the 68000 bus and the 16550 emulation controller using SDMA transfers under RISC control.
- Data transfers are never serialized in the emulation controller.

7.6.2 16550 Emulation Controller Overview

The 16550 emulation controller has the complete register set of the 16550 with the same FIFO and interrupt structure. The data transfer between the PC and the 68000 core consists of parallel transfers of data. There is no parallel to serial (shift-register) transformation of the data.

The 16550 emulation controller transfers modem status written by the PC via hardware registers visible to the 68000 core. The 68000 core transfers data and modem status bits to the 16550 emulation controller though register visible to both the 68000 core and the PC. Data transfers to and from the 16550 FIFO or holding registers are routed to IMP memory using the RISC controller and the SDMA channels. Data transfers to and from the PC take place through the PCMCIA interface. The 16550 emulation controller uses SCC2 parameter RAM space and its baud rate generator (BRG) when enable and therefore the SCC2 serial channel cannot be used simultaneously with the 16550 emulation. The 16550 controller uses the same data structures as a typical IMP SCC in UART mode and supports multi-buffer operation.

Two 4-bit counters and a BRG control the data transfer rate of the 16550 emulation controller, the FIFO emulation time-out, and the 302 idle emulation.

7.6.2.1 16550 Emulation Controller FIFOs Overview

The 16550 controller has separate transmit and receive FIFO's for implementing 16550 mode which is enabled by setting the FIFO enable bit in the FIFO control register. Figure 7-42 shows the receiver FIFO which transfers data from the CP 16550 transmit buffer descriptors via the communication processor's peripheral bus to the PC via the PCMCIA bus. The FIFO is 16 bytes deep and includes three error bits that are attached to each byte and are decoded in the line status register as each byte reaches the top of the FIFO. The data is never serialized in the process. The baud rate clock is provided by BRG2 and can be divided down further by a 4 bit counter. The 4 bit counter is useful to slow the baud rate clock down to a rate that emulates the serial transfer rates of a true 16550 while allowing the baud rate generator itself to be programmed with traditional values used for serial UART rates. This will keep the receive interrupt rate to the PC at the level it is used to handling and prevent the PC from being inundated with bursts of fast interrupts.

7.6.3 PC Accesses

The PC accesses the 16550 emulation registers through the PCMCIA interface using I/O space accesses. These read and write cycles are synchronized with the IMP clock. The IMP clock rate affects the synchronization time and the cycle length. When using low rate clocks, $\overline{\text{WAIT}}$ should be asserted on the PCMCIA cycle to delay the completion of the cycle. This can be done by making sure that the NWAIT bit in the 16550 emulation mode register is reset.

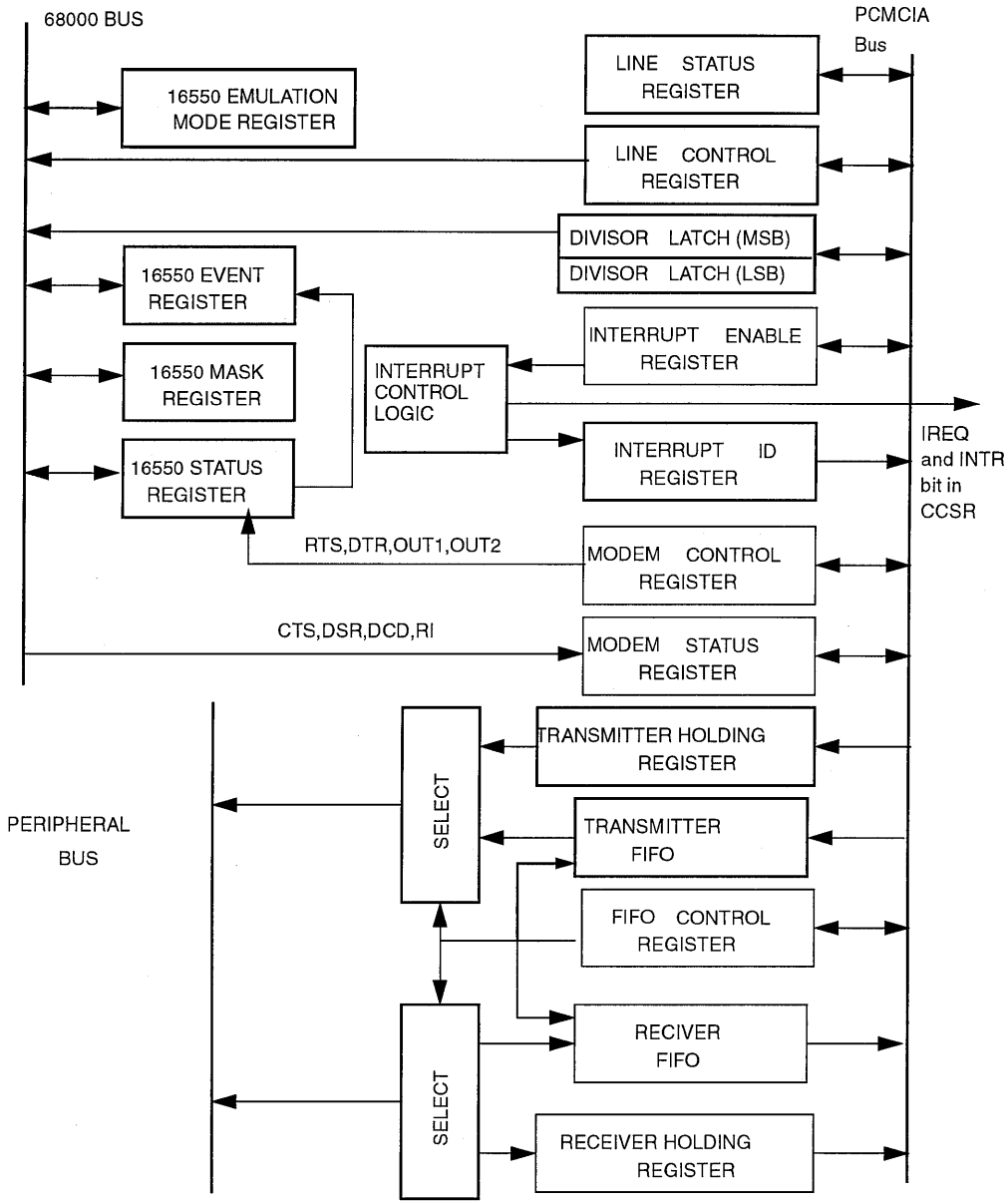


Figure 7-41. 16550 Emulation Controller Block Diagram

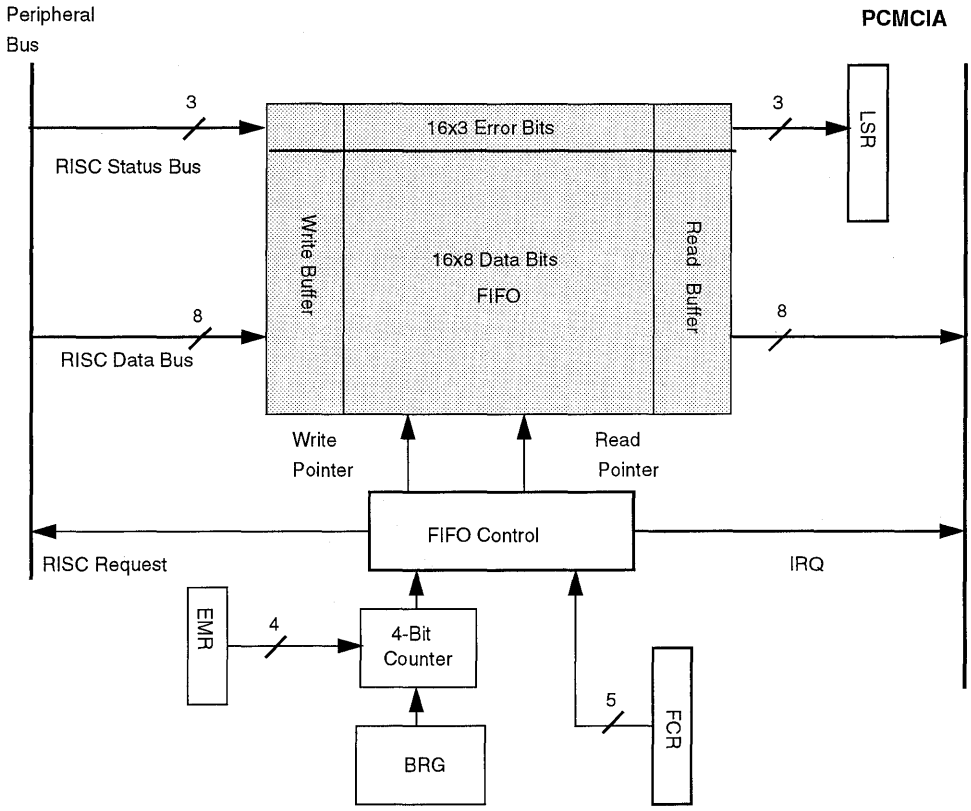


Figure 7-42. 16550 Emulation Controller Receiver FIFO

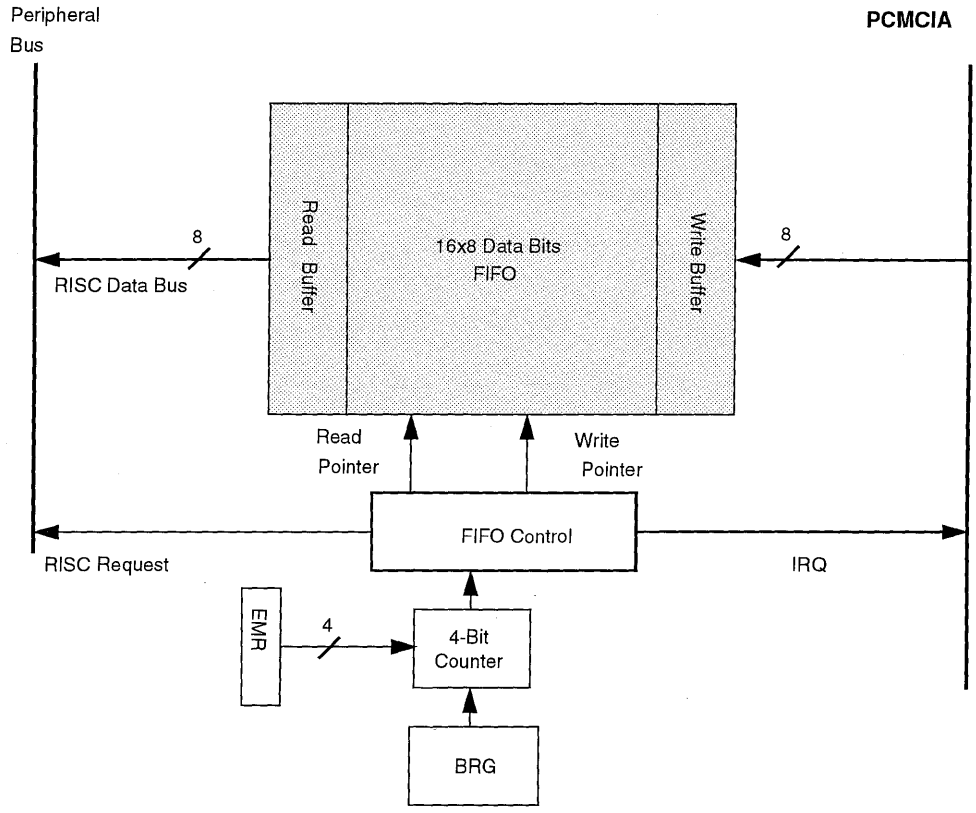


Figure 7-43. 16550 Emulation Controller Transmitter FIFO

7.6.4 PC Programmer Model

The PC programmer model is identical to the standard 16550. All software packages that support the 16650 can be executed without any modification.

Table 7-17. 16550 Emulation Registers Addresses

$\overline{REG}$	$\overline{CE1}$	IORD	IOWR	DLAB	A25	A2	A1	A0	Register
L	L	L	H	0	0	0	0	0	Receiver buffer (read)
L	L	H	L	0	0	0	0	0	Transmitter holding register (write)
L	L	L	H	0	0	0	0	1	Interrupt enable (read)
L	L	H	L	0	0	0	0	1	Interrupt enable (write)
L	L	L	H	x	0	0	1	0	Interrupt identification (read)
L	L	H	L	x	0	0	1	0	FIFO control (write)
L	L	L	H	x	0	0	1	1	Line control (read)
L	L	H	L	x	0	0	1	1	Line control (write)
L	L	L	H	x	0	1	0	0	MODEM control (read)

Table 7-17. 16550 Emulation Registers Addresses

$\overline{REG}$	$\overline{CE1}$	IORD	IOWR	DLAB	A25	A2	A1	A0	Register
L	L	H	L	x	0	1	0	0	MODEM control (write)
L	L	L	H	x	0	1	0	1	Line status (read)
L	L	L	H	x	0	1	1	0	MODEM status (read)
L	L	L	H	x	0	1	1	1	Scratch (read)
L	L	H	L	x	0	1	1	1	Scratch (write)
L	L	L	H	1	0	0	0	0	Divisor latch (LS) (read)
L	L	H	L	1	0	0	0	0	Divisor latch (LS) (write)
L	L	L	H	1	0	0	0	1	Divisor latch (MS) (read)
L	L	H	L	1	0	0	0	1	Divisor latch (MS) (write)

7.6.4.1 16550 Emulation Registers Description

7

7.6.4.1.1 Line Control Register (LCR)

This register is used to specify the format of the asynchronous serial data. The 68000 core receives an interrupt when the PC writes to this register (See 7.6.5.12 16550 Event Register for more details). The 68000 may read the register bits and emulate the UART functions.

The PC can write and read the contents of this register. The 68000 can only read this register.

7.6.4.1.2 Line Control Register (LSR)

LCR Base + $8F2

7	6	5	4	3	2	1	0
DLAB	Set Break	Stick Parity	Even Parity	Parity Enable	Stop Bit Length	Character Length	
RESET							
0	0	0	0	0	0	0	0

Read/Write—PC
Read-Only—68000

Bits 0-5

These bits do not affect the 16550 emulation controller. The 68000 core may read them and use them in the emulation process. The 16550 emulation controller generates the LCR interrupt (if enabled) when the PC writes these bits.

NOTE

The 16550 emulation logic will write a zero to the bit 8 location when the character length is programmed to 7-bits.

Set Break—Break Control Bit

When the PC sets this bit to one, the 16550 emulation controller will transfer the receive break indication to the 68000 core. It is done by incrementing the BRKEC counter, closing the receive buffer, setting the BR bit (if a buffer is currently open) and generating the BRK interrupt (if enabled). In addition, the 16550 emulation controller generates the LCR inter-

rupt (if enabled). When the PC clears this bit, the 16550 emulation controller generates the LCR interrupt (if enabled). This enables the 68000 core to measure the BREAK length.

DLAB—Divisor Latch Access Bit

The PC must set this bit to one to access the divisor latches during a read or write operation. The PC must set this bit to zero to access the receiver buffer, the transmitter holding register, or the interrupt enable register.

7.6.4.1.3 Line Status Register (LSR) (Read Only)

LSR Base + $8F6

7	6	5	4	3	2	1	0
Rx FIFO Error	Transmitter Empty	Transmitter Holding Register	Break Interrupt	Framing Error	Parity Error	Overrun Error	Data Ready
RESET							
0	0	0	0	0	0	0	0

Read-Only by the PC—68000 can set the Overrun bit

Data Ready

0 = The receiver data register or FIFO is empty
1 = A character has been transferred by the RISC controller into the receiver data register or FIFO.

Overrun

This bit is set by the 68000 core when the transmit buffer has not been transmitted for a long period of time. It is done to emulate the 16550 Rx FIFO overrun.

This bit is reset when the PC reads the LSR.

Parity Error

This bit is set by the 68000 core, by setting the P bit in the transmit buffer descriptor. This bit is reset when the PC reads the LSR. In FIFO mode, the parity error bit will be set when the character associated with the parity error moves to the top of the FIFO.

Framing Error

This bit is set by the 68000 core indirectly when the 68000 core sets the FR bit in the transmit buffer descriptor. This bit is reset when the PC reads the LSR register. In FIFO mode, the framing error bit is set when the character associated with the framing error moves to the top of the FIFO.

Break Interrupt (BI)

This bit is set by the 68000 core by issuing the STOP TRANSMIT command. This bit is reset when the PC reads the LSR register. In FIFO mode, this error bit will be set when the character associated with the error moves to the top of the FIFO. When break occurs, only one character is written into the FIFO.

NOTE

Bits 1-4 are error bits that produce a receiver line status interrupt whenever one of these bits is set and the interrupt is enabled.

Transmitter Holding Register Empty (THRE)

0 = The transmitter holding register is written by the PC. In the FIFO mode at least one byte is written to the FIFO.
1 = The transmitter is ready to accept a new character. A character is transferred from the transmitter holding register to the transmitter shift register emulation. In the FIFO mode, the Tx FIFO is empty.

Transmitter Empty (TEMT)

0 = The transmitter holding register contains a character. In the FIFO mode at least one byte is written to the FIFO.
1 = The transmitter holding register and the transmitter shift register emulation are both empty. In the FIFO mode, the Tx FIFO is empty.

LSR7

In the NS16450 mode this bit is 0. In the FIFO mode, this bit is set when there is at least one parity error, framing error, or break indication in the FIFO. It is cleared when the pc reads the LSR, if there are no subsequent errors in the FIFO.

7.6.4.1.4 FIFO Control Register FCR (Write Only)

FCR PC Access Only

7	6	5	4	3	2	1	0
Rx FIFO Trigger		Res	Res	DMA MODE	XMIT FIFO RST	RCVR FIFO RST	FIFO Enable
RESET							
0	0	0	0	0	0	0	0

Write-Only—PC

FIFO Enable

0 = Clear all bytes in XMIT and RCVR FIFOs.
1 = Enables both FIFOs.

NOTE

When changing from FIFO mode to NS16450 mode and vice versa, data is automatically cleared from the FIFOs. This bit must be one when other FCR bits are written.

Receiver FIFO Reset

When the PC writes a one to this bit, all bytes are cleared in the RCVR FIFO, and the RCVR FIFO counter is reset. The one that is written to this bit is cleared immediately (i.e. it is not latched).

Transmitter FIFO Reset

When the PC writes a one to this bit, all bytes in the XMIT FIFO are cleared and the XMIT FIFO counter is reset.The one that is written to this bit is cleared immediately (i.e. it is not latched).

DMA Mode Select

This bit is not implemented because the DMA mode is not supported.

Receiver FIFO trigger

These bits are used to set the trigger level for the receiver FIFO interrupt.

Table 7-18. Receiver FIFO Trigger Level (Bytes)

FCR7	FCR6	RCVR FIFO Trigger Level (Bytes)
0	0	01
0	1	04
1	0	08
1	1	14

7.6.4.1.5 Interrupt Identification Register -IIR (Read Only)

The interrupts are prioritized into four levels and recorded in the interrupt identification register (IIR). The four levels of interrupt conditions in order of priority are: receiver line status, received data ready, transmitter holding register empty, and MODEM status. When the PC accesses the IIR, the contents of the register and all pending interrupts are frozen. Any new interrupts will be recorded and updated after the current access is terminated.

IIR PC Access Only

7	6	5	4	3	2	1	0
FIFO Enables		Res	Res	INTERRUPT ID			Interrupt Pending
RESET							
0	0	0	0	0	0	0	0

Read-Only–PC

Interrupt Pending

0 = An interrupt is pending and the IIR contents may be used as a pointer to the appropriate interrupt routine.
1 = No interrupt is pending

Interrupt ID

These bits are used to identify the highest priority interrupt pending, as indicated in Table 7-19.

Bits 4 and 5—These bits are always 0.

Bits 6 and 7—These bits are set when FCR0=1. In the 16450 mode they are always 0.

Table 7-19. Interrupt Control Functions

FIFO Mode Only	Interrupt Identification Reg			Interrupt Set and Reset Functions			
Bit 3	Bit 2	Bit 1	Bit 0	Priority Level	Interrupt Type	Interrupt Source	Interrupt Reset Control
0	0	0	1	-	None	None	-
0	1	1	0	Highest	Rx line status	Overrun or parity error or framing error or break interrupt	Reading the line status register
0	1	0	0	Second	Rx data available	Rx Data available or trigger level reached	Reading the Rx buffer register or the FIFO drops below the trigger level.
1	1	0	0	Second	Character timeout indication	No characters have been removed or input to the Rx FIFO during the last 4 character times and there was at least 1 character in it during this time.	Reading the receiver buffer register.
0	0	1	0	Third	Tx holding register empty	Tx Holding register empty	Reading the IIR register (if source of interrupt) or writing into the transmitter holding register.
0	0	0	0	Fourth	MODEM status	CTS, DSR, ring indication, or CD	Reading the MODEM status register

7

7.6.4.1.6 Interrupt Enable Register - IER

IER PC Access Only

7	6	5	4	3	2	1	0
0	0	0	0	EDSSI	ELSI	ETBEI	ERBFI
RESET 0	0	0	0	0	0	0	0

Read/Write–PC

ERBFI—Enable Received Data Available Interrupt

This bit enables the received data available interrupt to the PC (and timeout interrupts in the FIFO mode), when set to one.

ETBEI—Enable Transmitter Holding Register Empty Interrupt

This bit enables the transmitter holding register empty interrupt to the PC, when set to one.

ELSI —Enable Receiver Line Status Interrupt

This bit enables the receiver line status interrupt to the PC, when set to one.

EDSSI—Enable MODEM Status Interrupt

This bit enables the MODEM status interrupt to the PC, when set to one.

Bits 4 to 7—These bits are always 0.

7.6.4.1.7 MODEM Control Register - MCR

MCR PC Access Only

7	6	5	4	3	2	1	0
0	0	0	Loop	Out2	Out1	RTS	DTR
RESET							
0	0	0	0	0	0	0	0

Read/Write–PC

DTR—Data Terminal Ready

When this bit is set, the DTR status bit in the 16550 status register is forced to 1. When this bit is reset, the DTR status bit in the 16550 status register is forced to 0. DTR status change interrupt may be generated. See 7.6.5.12 16550 Event Register for more details.

RTS—Request To Send

When this bit is set, the RTS status bit in the 16550 status register is forced to 1. When this bit is reset, the RTS status bit in the 16550 status registers forced to 0. RTS status change interrupt may be generated. See 7.6.5.12 16550 Event Register for more details.

Out 1

When this bit is set, the Out 1 status bit in the 16550 status register is forced to 1. When this bit is reset, the Out 1 status bit in the 16550 status register is forced to 0. An Out 1 status change interrupt may be generated. See 7.6.5.12 16550 Event Register for more details.

Out 2

When this bit is set, the Out 2 status bit in the 16550 status register is forced to 1. When this bit is reset, the Out 2 status bit in the 16550 status register is forced to 0. Out 2 status change interrupt may be generated. See 7.6.5.12 16550 Event Register for more details.

Loop—Loopback Mode

When this pin is set to one the 16550 emulation controller is forced into the local loopback diagnostic feature. The following occurs:

The four MODEM control outputs (DTR, RTS, OUT 1, OUT 2) are internally connected to the four MODEM control inputs (DSR, CTS, RI and DCD).

The 68000 core will receive the loop interrupt (See Loop—Loopback Mode on page 154) and will have to transfer data between the TxBDs and the RxBDs to emulate the loopback feature. The buffer length should only be one byte.

NOTE

To emulate the local loopback diagnostic, the slowdown mechanism should be enabled. It should be enabled by the user, otherwise the loopback will be done too quickly.

Bits 5-7 —Always 0.

7.6.4.1.8 MODEM Status Register

MSR — Base + $8F3

7	6	5	4	3	2	1	0
DCD	RI	DSR	CTS	DDCD	TERI	DDSR	DCTS
RESET							
0	0	0	0	0	0	0	0

Read-Only–PC

Read/Write-68000

DCTS—Delta Clear to Send

This bit indicates that the CTS bit has changed state since the last time it was read by the PC. This bit is reset to logic 0 whenever the PC reads the MSR register.

DDSR—Delta Data Set Ready

This bit indicates that the DSR bit has changed state since the last time it was read by the PC. This bit is reset to logic 0 whenever the PC reads the MSR register.

TERI—Trailing Edge of Ring Indicator

This bit indicates that the RI bit has changed from high to low state. This bit is reset to logic 0 whenever the PC reads the MSR register.

DDCD—Delta Data Carrier Detect

This bit indicates that the DCD bit has changed state. This bit is reset to logic 0 whenever the PC reads the MSR register.

NOTE

Whenever bits 0, 1, 2 or 3 are set to one, a MODEM status interrupt is generated to the PC.

CTS—Clear To Send

This bit emulates the complement of the CTS input. The 68000 core writes this bit. If bit 4 (loop) of the MCR is set to 1, this bit is equivalent to RTS in the MCR.

DSR—Data Set Ready

This bit emulates the complement of the DSR input. The 68000 core writes this bit. If bit 4 (loop) of the MCR is set to 1, this bit is equivalent to DTR in the MCR.

RI—Ring Indicator

This bit emulates the complement of the RI input. The 68000 core writes this bit. If bit 4 (loop) of the MCR is set to 1, this bit is equivalent to OUT 1 in the MCR.

DCD—Data Carrier Detect

This bit emulates the complement of the DCD input. The 68000 core writes this bit. If bit 4 (loop) of the MCR is set to 1, this bit is equivalent to OUT 2 in the MCR.

7.6.4.1.9 Divisor Latch (LS) - DLL

This register contains the low byte of the baud rate generator divisor latch. When the PC writes this register, the DLL interrupt (if enabled) will be generated to the 68000 core. See 7.6.5.12 16550 Event Register for more details. The 68000 core can read this register value and program SCC2 BRG according to this value.

NOTE

The 16550 requires a baud rate multiplied by one clock instead of the baud rate multiplied by sixteen clock used in the standard 68302 SCC in UART mode (i.e. to generate a 16550 9600 baud clock at 16.67MHZ, the clock divider bits in the SCON register should be programmed to 1735 instead of 108). The 4 bit dividers for the TX and RX FIFOs should also be programmed to match the number of bits per character being transferred to slow down the PC interrupt rate to a level to emulate a serialized UART.

7.6.4.1.10 Divisor Latch (LM) - DLM

This register contains the high byte of the baud rate generator divisor latch. When the PC writes this register, the DLM interrupt (if enabled) will be generated to the 68000 core. See 7.6.5.12 16550 Event Register for more details. The 68000 core can read the register value at program SCC2 BRG.

7.6.4.1.11 Receive Buffer Register - RBR

When the 68000 core has data to deliver to the PC, it prepares a buffer descriptor with associated data buffer. The RISC controller will deliver the data buffer characters into the receive buffer register. An interrupt will be generated to the PC if the receive data available interrupt is enabled and the ready bit is set. Also, the 68000 core may set the FE and P error bits in the transmit buffer descriptor (TxBD) and the RISC will transfer them with the data to the FIFO or the line status register. To slow down the transfer rate to the PC and emulate the same rate as a serial 16550, the PC interrupt request rate may be slowed by programming the EMR RX divider bits (see 7.6.5.2 16550 Emulation Mode Register - EMR).

7.6.4.1.12 Transmit Holding Register - THR

When the PC writes to the transmit holding register, the RISC controller receives a request from the 16550 emulation logic. The RISC controller, when servicing the request, will read the register and will transfer the data into the Rx data buffer. When the RISC reads the register, the transmit holding register empty flag is set. To slow down the transfer rate and emulate the same rate as the serial 16550, the RISC controller request may be delayed by programming the EMR TX divider bits (see 7.6.5.2 16550 Emulation Mode Register - EMR).

7.6.4.1.13 Scratchpad Register - SCR

This 8-bit read/write register does not control the 16550 in any way. It is intended as a scratchpad register to be used by the PC to hold data temporarily.

7.6.5 68000 Programming Model

SCC2 can be configured as a 16550 emulation controller by programming its mode register (SCM) into UART16550 mode (MODE1-MODE0 = 10). BRG2 should be programmed to the appropriate rate after reading the 16550 emulation divisor latch. It uses the same data structure as the SCCs in the other modes. The 16550 data structure supports multi-buffer operation. The user can program the 16550 emulation controller to accept or reject control characters. If a control character is rejected, an interrupt may be generated. The 16550 emulation controller enables the user to transfer break sequences, parity and framing errors to the PC via the buffer descriptor table. An indication of the status of the 16550 emulation transmit FIFO (IDLE) is reported through the status register, and a maskable interrupt is generated upon a status change. In its simplest form, the 16550 can function in a character-oriented environment. Each character is transmitted with accompanied stop bits and parity (as configured by the user), and received into separate one byte buffers. Reception of each buffer may generate a maskable interrupt.

7

7.6.5.1 16550 Memory Map

When the 16550 emulation controller is enabled, the 68356 overlays the structure illustrated in Table 7-7, with the 16550 emulation controller specific parameters, described in Table 7-20.

Table 7-20. 16550 Specific Parameter RAM

Address	Name	Width	Description
SCC2 Base+9C	**MAX_IDL**	Word	Maximum IDLE characters
SCC2 Base+9E	IDLC	Word	Temporary IDLE counter
SCC2 Base+A0	**BRKCR**	Word	Break count register (transmit)
SCC2 Base+A2	res	Word	-
SCC2 Base+A4	res	Word	-
SCC2 Base+A6	res	Word	-
SCC2 Base+A8	BRKEC	Word	Receive break condition counter
SCC2 Base+AA	res	Word	-
SCC2 Base+AC	res	Word	-
SCC2 Base+AE	**RCCR**	Word	Receive control character register
SCC2 Base+B0	**CHARCTER1**	Word	CONTROL character 1
SCC2 Base+B2	**CHARCTER2**	Word	CONTROL character 2
SCC2 Base+B4	**CHARCTER3**	Word	CONTROL character 3
SCC2 Base+B6	**CHARCTER4**	Word	CONTROL character 4
SCC2 Base+B8	**CHARCTER5**	Word	CONTROL character 5
SCC2 Base+BA	**CHARCTER6**	Word	CONTROL character 6
SCC2 Base+BC	**CHARCTER7**	Word	CONTROL character 7
SCC2 Base+BE	**CHARCTER8**	Word	CONTROL character 8

The items above in **bold face** should be initialized by the user.

MAX_IDL.

The 16550 emulation TX FIFO is polled by the RISC controller every programmable amount of time. When the Tx FIFO is empty, the RISC controller assumes reception of an IDLE character. Once a character of data is received, the RISC controller counts any idle characters received. If a MAX_IDL number of idle characters are received before the next data character is received, and idle timeout occurs, the buffer is closed. This, in turn, can produce an interrupt request to the 68000 core to receive the data from the buffer. Thus, MAX_IDL provides a convenient way to demarcate frames in the 16550 mode.

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 emulation 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 16550 emulation controller will send an a break character sequence towards the PC whenever a STOP TRANSMIT command is given. The Break Interrupt bit in the emulation line status register will be set. (See 7.6.4.1.3 Line Status Register (LSR) (Read Only) for more details). The data sent towards the 16550 emulation receiver is the data written in character8 in the control characters table. The number of break characters sent by the 16550 emulation controller is determined by the value in BRKCR.

RCCR,CHARACTER8-1.

These characters define the receive control characters, on which interrupts may be generated.

7.6.5.2 16550 Emulation Mode Register - EMR

This register programmed by the 68000 core is used to enable the RISC controller reception and transmission processes. In addition, this register is used to program the counters used for reception, to slow down transmissions, for timeout in the 16550 emulation Rx FIFO, and the Rx buffer descriptor IDLE count.

EMR Base + $8F0

15	14	13	12	11	10	9	8
RES	RES	NWAIT	FRZ	ENT	ENR	TX Divider	
RESET				0			
0	0	0	0		0	0	0

0

7	6	5	4	3	2	1	0
Tx Divider		Rx Divider				TSDE	RSDE
RESET							
0	0	0	0	0	0	0	0

Read/Write–68000

The 16550 controller uses the SCC2 interrupt event and mask registers and the SCC2 parameter RAM. The 16550 controller BRG uses the SCC2 SCON register to set the BRG clock frequency which can be further divided by the two 4-bit counters. A divide by two block can be enabled from either the system clock source or the TIN pin depending on where the baud rate clock is supplied.

NOTE

IMP clock to 16550 BRG rate should not exceed a ratio of 3:1.

NWAIT—PCMCIA No WAIT Cycle

The PC accesses the 16550 emulation registers through the PCMCIA interface on I/O space. These read and write cycles are synchronized with the IMP clock. The IMP clock rate affects the synchronization time and the cycle length. When using lower rate IMP clocks, $\overline{\text{WAIT}}$ should be asserted on the PCMCIA cycle to delay the completion of the cycle.

0 = $\overline{\text{WAIT}}$ will be asserted.
1 = $\overline{\text{WAIT}}$ will not be asserted. The 302 clock rate should be equal or higher than 25 Mhz, otherwise the data will not be read or written correctly.

FRZ—Freeze Transmission

This bit allows the user to halt the 68000 data transfer into the 16550 emulation receive FIFO. Data transfer will continue from the next character in the buffer when this bit is reset to zero.

0 = Normal operation (or resume transmission after FRZ is set)
1 = The RISC controller completes the current transfer into the 16550 emulation receive FIFO and then stops transferring data.

ENT—Enable Transmitter

0 = The RISC controller transmission process is disabled.
1 = The RISC controller transmission process is enabled

ENR—Enable Receiver

0 = The RISC controller reception process is disabled.
1 = The RISC controller reception process is enabled

Tx Divider

This 4-bit value is loaded into a 4-bit countdown counter. This counter divides BRG2 input clock. The 16550 emulation transmit FIFO will request the RISC controller to transfer data into memory when the counter reaches zero and the FIFO is not empty. This mechanism will slow down the transfer rate and emulate the PC interrupts rate as if the data transfer was done serially. When the FIFO is empty, the RISC controller will identify the request as an IDLE reception for use by the close BD mechanism (See 7.6.5.10 16550 Rx Buffer Descriptor (Rx BD))

NOTE

In this case, the PC is transmitting to the 68000 core.

Rx Divider

This 4-bit value is loaded into a 4-bit countdown counter. This counter divides the BRG2 input clock. The 16550 emulation receive FIFO requests the RISC controller to poll its Tx BD. When the BD is not empty, it will transfer data from memory into the FIFO, when the counter reaches zero and the FIFO is not full. This mechanism slows down the transfer rate and emulates the PC interrupt rate as if the data transfer was done serially. This counter is also used for the 16550 Rx FIFO time-out interrupt emulation.

NOTE

In this case, the 68000 is transmitting to the PC core.

TSDE—Transmitter Slowdown Enable

When set, this bit enables the transmitter slow-down mechanism. When cleared, the transmit counter is used only for IDLE emulation and the transfer rate is as fast as the RISC controller and the PC can operate.

NOTE

In this case, the PC is transmitting to the 68000 core.

RSDE—Receiver Slowdown Enable

When set, this bit enables the receiver slow-down mechanism. When cleared, the receive counter is used only for FIFO time-out interrupt emulation and the transfer rate is as fast as the RISC controller and the PC can transfer. In this case, transfers will probably be limited by PC software, rather than the RISC and SDMA channels on the 68K bus. The IMP transmitter will poll the transmit BD ready bit every Tx counter time-out. When the buffer is ready, data will be transferred at the maximal rate.

NOTE

In this case, the 68000 is transmitting to the PC core.

7.6.5.3 16550 Command Set

The following commands can be issued to the command register (CR).

7.6.5.4 Transmit Commands

7.6.5.4.14 *STOP TRANSMIT* Command

After a hardware or software reset, and after the channel has been enabled by setting the transmit enable bit in the EMR register, the channel will be in the transmit enable mode and will start polling the first buffer in the table every programmable number of clocks (as programmed by the Tx Divider bits in the EMR).

The channel *STOP TRANSMIT* command disables the transmission of characters to the 16550 emulation receive FIFO.

The 16550 transmitter will transmit a programmable number of break sequences to the 16550 emulation receive FIFO and then stop transmission. The number of break sequences (which may be zero) should be written to the break count register (BRKCR) before this command is issued to the 16550 Controller.

7.6.5.4.15 *RESTART TRANSMIT* Command

The channel *RESTART TRANSMIT* command enables the transmission of characters on the 16550 emulation receive FIFO. This command is expected by the 16550 controller after disabling the channel in the mode register (EMR), and after a STOP TRANSMIT command. The 16550 controller will resume transmission from the current transmitter buffer in the Tx BD Table.

7.6.5.5 Receive Commands

7.6.5.5.16 *ENTER HUNT MODE* Command

After a hardware or software reset, and after the channel has been enabled by setting the receive enable bit in the EMR register, the channel will be in the transmit enable mode and will use the first buffer in the table.

The *ENTER HUNT MODE* command is used to force the 16550 controller to close the current Rx BD (if data is being received using it). The command generates an RX interrupt (if enabled) as the buffer is closed. The 16550 controller will resume reception using the next BD once a single IDLE character is received.

7.6.5.6 16550 Control Characters (Receiver)

The 16550 controller has the capability to recognize special control characters transferred from the PC to the card. These characters may be used when the 16550 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 written to the receive buffer (upon which the 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 16550 uses a table of 16-bit entries to support control character recognition. Each entry consists of the control character, a valid bit, and a reject character bit. The control characters table is described in Table 7-21.

RCCR Received Control Character Register

Upon a control character match for which the Reject bit is set, the 16550 controller will write the control character into the RCCR and generate a maskable interrupt. The core must process the interrupt and read the RCCR before a second control character arrives. Failure to do so will result in the 16550 controller overwriting the first control character.

CHARACTER8-1 =T Control Character Values

These fields define control characters that should be compared to the incoming character.

Table 7-21. 16550 Control Character Table

	15							8	7	0
Offset+0									RCCR	
Offset+2	E	R							CHARACTER1	
Offset+4	E	R							CHARACTER2	
Offset+6										
Offset+10	E	R	REA	I	CT	0	0	0	CHARACTER8	

E—End of table

0 = This entry is valid. The lower 8 bits will be checked against the incoming character.
1 = The entry is not valid. This must be the last entry in the control characters table.

NOTE

In tables with 8 control characters, this bit is always 0.

R—Reject character

0 = The character is not rejected but 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 (I-bit in the receive BD) interrupt is generated.
1 = If this character is recognized it will not be written to the receive buffer. Instead, it is written to the received control characters register (RCCR) and a maskable interrupt is generated. The current buffer is not closed when a control character is received with R set.

7.6.5.6.17 Transmission of Out-of-Sequence Characters (Transmitter)

Transmission of out-of-sequence characters to the PC is also supported by the 16550 controller, and is normally used for the transmission of flow control characters such as XON or XOFF. This is performed using the last (eight) entry in the control characters table. The 16550 controller will poll this character whenever the transmitter is enabled for 16550 controller operation. This includes 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 preempt characters already in the 16550 receiver emulation FIFO.

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 flow control transmission character. For receive control characters entry, it maintains its functionality as previously defined.

REA—Ready

This bit is set by the 68000 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 core will be interrupted when this character has been transmitted. (The TX bit will be set in the 16550 event register.)

CHARACTER8—Flow Control Character Value

This value contains the character to be transmitted. This value may be modified only while the REA bit is cleared.

7.6.5.7 BREAK Support (Receiver)

The 16550 controller offers very flexible BREAK emulation support for the receiver. See 7.6.5.9.19 BREAK Sequence for more details.

7.6.5.8 Send Break (Transmitter)

A break is an all-zeros character without a stop bit(s). A break emulation is transferred to the PC by issuing the STOP TRANSMIT command. The 16550 controller sends a programmable number of break characters according to the break count register (BRKCR), and then reverts to idle or sends data if the RESTART TRANSMIT command was given before completion.

The break characters do not pre-empt characters already in the 16550 receiver emulation FIFO.

7.6.5.9 16550 Error Handling

The 16550 controller reports character reception and transmission error conditions via the channel buffer descriptors, the error counters, and the 16550 event register. The modem interface lines can be monitored by the port C pins.

7.6.5.9.18 IDLE Sequence Receive

An IDLE is detected when one character consisting of all ones is received. The 16550 controller emulates IDLE reception. The 16550 emulation TX FIFO is polled by the RISC controller every programmable amount of time. When the Tx FIFO is empty, the RISC controller assumes reception of an IDLE character. When the 16550 is receiving data into a receive buffer, and an IDLE is received, the channel counts the number of consecutive IDLE characters received. If the count reaches the value programmed into MAX_IDL, the buffer is closed and an RX interrupt is generated. If no receive buffer is open, this event does not generate an interrupt or any status information. The internal idle counter (IDLC) is reset every time a character is received.

7.6.5.9.19 BREAK Sequence

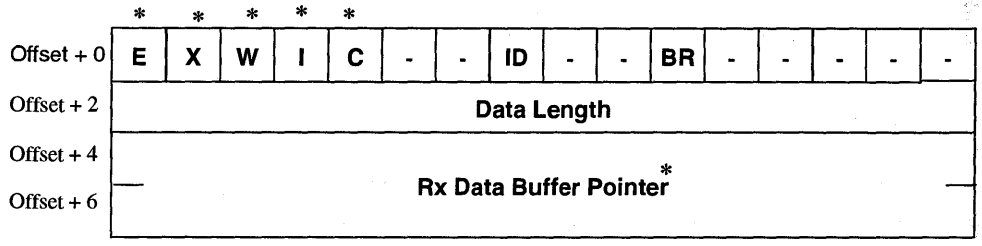
The 16550 controller offers very flexible BREAK support for the receiver. When the PC sets the Set Break bit in the LCR register or a BREAK sequence is received, the 16550 controller increments the break error counter (BRKEC), and issues the break (BRK) event in the 16550 event register, which can generate an interrupt if enabled. If the 16550 controller was currently in the process of receiving characters when the BREAK was received, it will also close the receive buffer and set the BR bit in the Rx BD, and write the RX bit in the event register, which can generate an interrupt if enabled. A long break sequence only increments the counter once.

Error Counter

BRKEC - Break Error Counter

7.6.5.10 16550 Rx Buffer Descriptor (Rx BD)

The CP reports information concerning the received data on a per buffer basis via buffer descriptors. The receive buffer descriptor (Rx BD) is described in Figure 7-44. The CP closes



*** Initialized by the user.**

Figure 7-44. 16550 Emulation Receive Buffer Descriptor (Rx BD)

the current buffer, generates a maskable interrupt, and starts to receive data to the next buffer due to these events:

1. Reception of a user-defined control character (when the Reject bit = 0 in the control character table entry).
2. Detection of the receive buffer being full.
3. Reception of a MAX_IDL number of consecutive IDLE characters.
4. Issuing the ENTER HUNT MODE command

The first word of the Rx BD contains control and status bits. Its format is detailed below.

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 core is free to examine

or write to any fields of this Rx BD. The CP will not use this BD again while the empty bit remains zero.

1 = The data buffer associated with this BD is empty, or reception is currently in progress. This Rx BD and its associated receive buffer are owned by the CP. Once the E bit is set, the 68000 core should not write any fields of this Rx BD.

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 buffer descriptor in the Rx BD table.
1 = This is the last buffer descriptor 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, erratic behavior may occur.

I—Interrupt

0 = No interrupt is generated after this buffer has been filled.
1 = The RX bit in the 16550 event register will be set when this buffer has been completely filled by the CP, indicating the need for the 68000 core to process the buffer. The RX bit can cause an interrupt.

C—Control Character

0 = This buffer does not contain a control character.
1 = This buffer contains a control character. The last byte in the buffer is one of the user defined control characters.

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).

BR—Break Received

A break sequence was received while receiving data into this buffer.

Data Length

Data length is 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

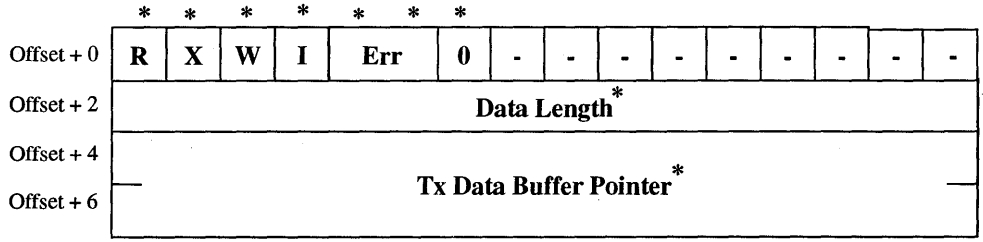
The actual amount of memory allocated for this buffer should be greater than or equal to the contents of the maximum receive buffer length register (MRBLR).

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.

7.6.5.11 16550 Tx Buffer Descriptor (Tx BD)

Data is presented to the CP for transmission on the 16550 channel by arranging it in buffers referenced by the channel's transmit buffer descriptor table. The CP confirms transmission



*** Initialized by the user.**

Figure 7-45. 16550 Transmit Buffer Descriptor (Tx BD)

via the buffer descriptors to inform the processor that the buffers have been serviced. The Tx BD is described in Figure 7-45. The first word contains status and control bits.

R—Ready

0 = The data buffer associated with this BD is not currently ready for transmission. The user is free to manipulate this BD or its associated data buffer. The CP clears this bit after the buffer has been transmitted or after an error condition is encountered.

1 = The data buffer, which has been prepared for transmission by the user, has not been transmitted or is currently being 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 buffer descriptor in the Tx BD table.

1 = This is the last buffer descriptor in the Tx BD table. After this buffer has been used, the CP will transmit data from 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, erratic behavior may occur.

I—Interrupt

0 = No interrupt is generated after this buffer has been serviced.
1 = The TX bit in the 16550 event register will be set when this buffer has been serviced by the CP, which can cause an interrupt.

Err—Parity and Framing Error

00 = No error is written into the 16550 emulation FIFO for the PC.
01 = Framing error is associated with the buffer's characters. The framing error will be set in the 16550 emulation line status register when the PC reads this character.
10 = When the transmit Slow Down is disabled (See TSDE—Transmitter Slowdown Enable on page 145) the next buffer transmission will be delayed by the counter time-out.
11 = A parity error is associated with the buffer's character. The parity error will be set in the 16550 emulation line status register when the PC reads this character.

NOTE

When parity error is transmitted to the PC, the buffer will only contain a single character.

NOTE

When error is transmitted to the PC, and the transmit Slow Down is disabled (See TSDE—Transmitter Slowdown Enable on page 145) the next buffer transmission will be delayed by the counter time-out.

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.

Tx Buffer Pointer

The Tx 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.

7.6.5.12 16550 Event Register

The 16550 event register (SCCE) is called the 16550 event register when the 16550 is enabled. It is a 16-bit register used to report events to the 68000 core recognized by the 16550 emulation controller and is also used generate interrupts. On recognition of an event, the 16550 controller will set the corresponding bit in the 16550 event register. Interrupts to the 68000 core generated by this register may be masked in the 16550 mask register.

The 16550 event register is a memory-mapped register that may be read by the 68000 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.

E550 Base + $898

15	14	13	12	11	10	9	8
RES	RES	RES	BRK	CCR	BSY	TX	RX
RESET 0	0	0	0	0	0	0	0

7	6	5	4	3	2	1	0
IDL	LCR	Loop	DL	OUT2	OUT1	RTS	DTR
RESET 0	0	0	0	0	0	0	0

Read/Write

Bits 15-13—Reserved. Should be written with zeros.

BRK—Break

This bit is set when a break character is received. This is the first break of a break sequence. Multiple break events will not be generated if a long break sequence is received.

CCR—Control Character Received

This bit is set when a control character was received (with reject character bit in the control character table (R bit = 1) and stored in the receive control character register (RCCR).

BSY—Busy Condition

This bit is set when a character was received and discarded due to lack of buffers. Reception continues as soon as an empty buffer is provided.

TX—Tx Buffer

This bit is set when a buffer has been transmitted over the 16550 channel. If CR = 1 in the Tx BD, this bit is set no sooner than when the last stop 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

This bit is set when a buffer has been received over the 16550 channel. This event occurs no sooner than the middle of the first stop bit of the character that causes the buffer to be closed.

IDL—IDLE Sequence Status Changed

This bit is set when a change in the status of the serial line is detected on the 16550 channel. The real-time status of the line may be read in SCCS. Idle is entered when 16550 emulation FIFO has been empty for at least one full character time. It is exited when a character is received.

LCR—Line Control Register Write

This bit is set when the PC writes to the 16550 LCR emulation register

DL—Divisor Latch

This bit is set when the PC writes to the 16550 divisor latch (either DLL or DLM) emulation registers

Loop—Loopback Mode

This bit is set when the PC changes the loop signal level by writing to the 16550 MODEM control emulation register. The status register may be read to determine loop current status.

Out2—Out2 Status Changed

This bit is set when the PC changes the Out2 signal level by writing to the 16550 MODEM control emulation register. The status register may be read to determine Out2 current status.

Out1—Out1 Status Changed

This bit is set when the PC changes the Out1 signal level by writing to the 16550 MODEM control emulation register. The status register may be read to determine Out1 current status.

RTS—RTS Status Changed

This bit is set when the PC changes the RTS signal level by writing to the 16550 MODEM control emulation register. The status register may be read to determine RTS current status.

DTR—DTR Status Changed

This bit is set when the PC changes the DTR signal level by writing to the 16550 MODEM control emulation register. The status register may be read to determine DTR current status.

7.6.5.13 16550 Mask Register

The 16550 mask register is a 16-bit read-write register with the same bit formats as the 16550 event register. If a bit in the 16550 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.

7.6.5.14 16550 Status Register

The 16550 status register is a 16-bit read-only register which allows the user to monitor real-time status conditions set by the PC.

S550 Base + $8AC

15	14	13	12	11	10	9	8
RES	RES	RES	RES	RES	RES	RES	RES
RESET				0			
0	0	0	0		0	0	0

7	6	5	4	3	2	1	0
RES	RES	Loop	ID	OUT2	OUT1	RTS	DTR
RESET							
0	0	0	0	0	0	0	0

Read/Write

Out2—Out1 Status

This bit reflects the current status of the Out2 bit in the 16550 MODEM control emulation register.

Out1—Out1 Status

This bit reflects the current status of the Out1 bit in the 16550 MODEM control emulation register.

RTS—RTS Status

This bit reflects the current status of the RTS bit in the 16550 MODEM control emulation register.

DTR—DTR Status

This bit reflects the current status of the DTR bit in the 16550 MODEM control emulation register.

ID—Idle Status

The ID bit is set when the 16550 emulation FIFO has been empty for at least one full character time.

0 = The line is not currently idle.
1 = The line is currently idle.

Loop—Loopback Mode

This bit reflects the current status of the Loop bit in the 16550 MODEM control emulation register.

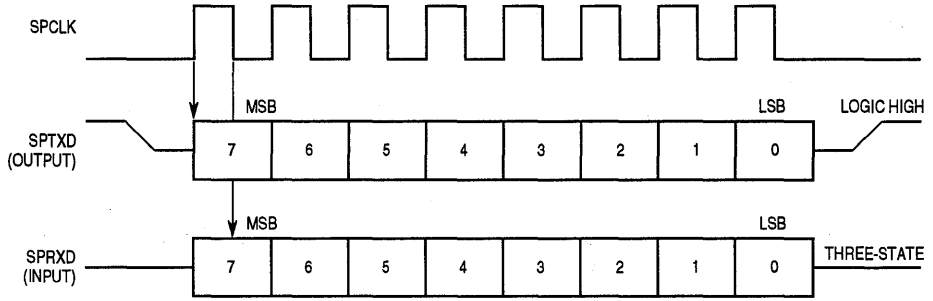
7.7 SERIAL COMMUNICATION PORT (SCP)

The SCP (see Figure 7-46) 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 IMP is an SCP master for this serial channel, it generates both the enable and the clock signals.

The SCP allows the IMP 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.

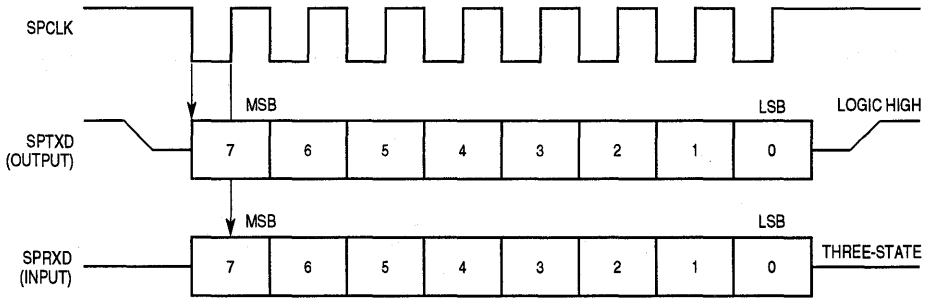
The SCP enable signals, which can be implemented using the general-purpose I/O pins, are used to enable one of several potential SCP slave devices. The clock signal (SPCLK) shifts the received data (SPRXD) in and shifts the transmitted data (SPTXD) out. The clock is gated; it operates only while data 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 7-46. 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 IMP 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

7.7.1 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 SC (see Figure 7-47). 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 $\overline{CTS}$ and $\overline{CD}$ lines and the ENT or ENR bits of SCC3 are set, the user may not modify the EN bit.

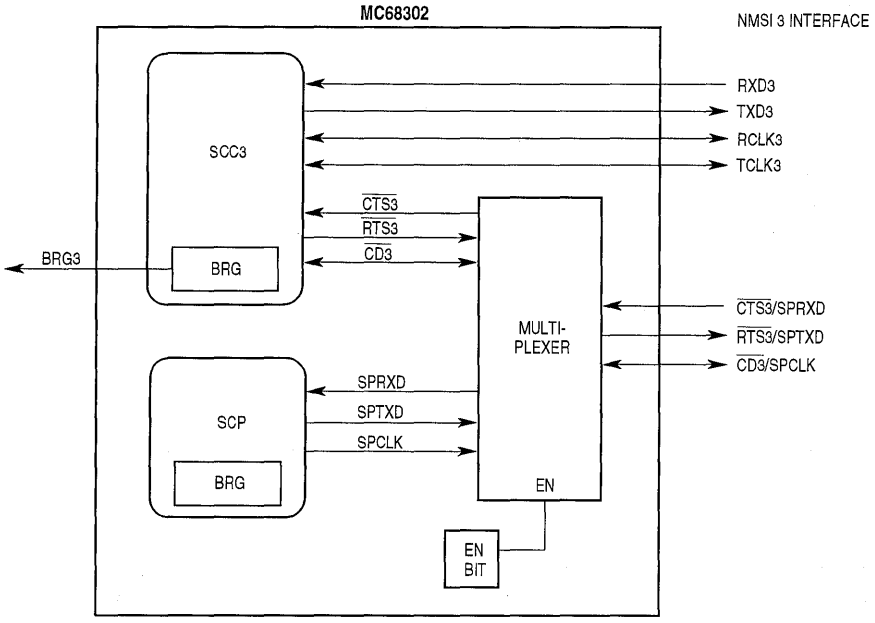
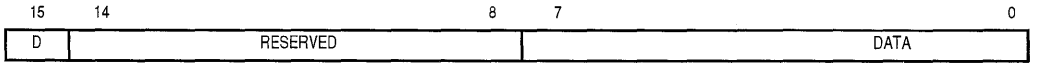


Figure 7-47. SCP vs. SCC Pin Multiplexing

7.7.2 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.



7.7.3 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.

7.8 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

7.8.1 SMC 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.

7.8.1.1 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.

7.8.1.2 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 7.4 Serial Channels Physical Interface).

7.8.2 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

7.8.3 SMC Commands

The following commands issued to the CP command register (see 7.3 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.

7.8.4 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 6-4). 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.

7.8.4.1 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

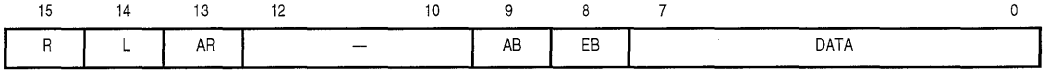
This bit is valid only in GCI mode when the monitor channel is in transparent mode.

Data—Data Field

The data field contains the byte of data received by SMC1.

7.8.4.2 SMC1 Transmit Buffer Descriptor

The CP reports information about this transmit byte through the BD.



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

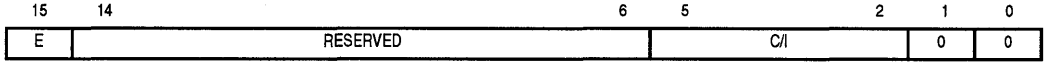
This bit is valid only in GCI mode when the monitor channel is in transparent mode.

Data—Data Field

The data field contains the data to be transmitted by SMC1.

7.8.4.3 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.



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.

7

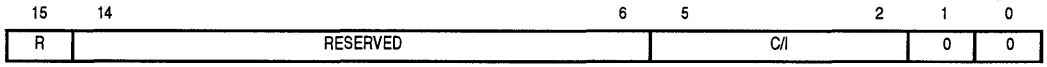
Bits 14–6—These bits are reserved and should be set to zero by the M68000 core.

C/I—Command/Indication Channel Data

Bits 1–0—The CP always writes these bits with zeros.

7.8.4.4 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.



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.

7.8.5 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 6.2 Interrupt Controller for more details.

SECTION 8
PCMCIA CONTROLLER

The MC68356 includes a PCMCIA interface, implemented including the registers and signal pins necessary to provide a fully functional PCMCIA release 2.1 slave interface. In addition, there is a mechanism which allows the PC to directly access resources on the 68000 bus.

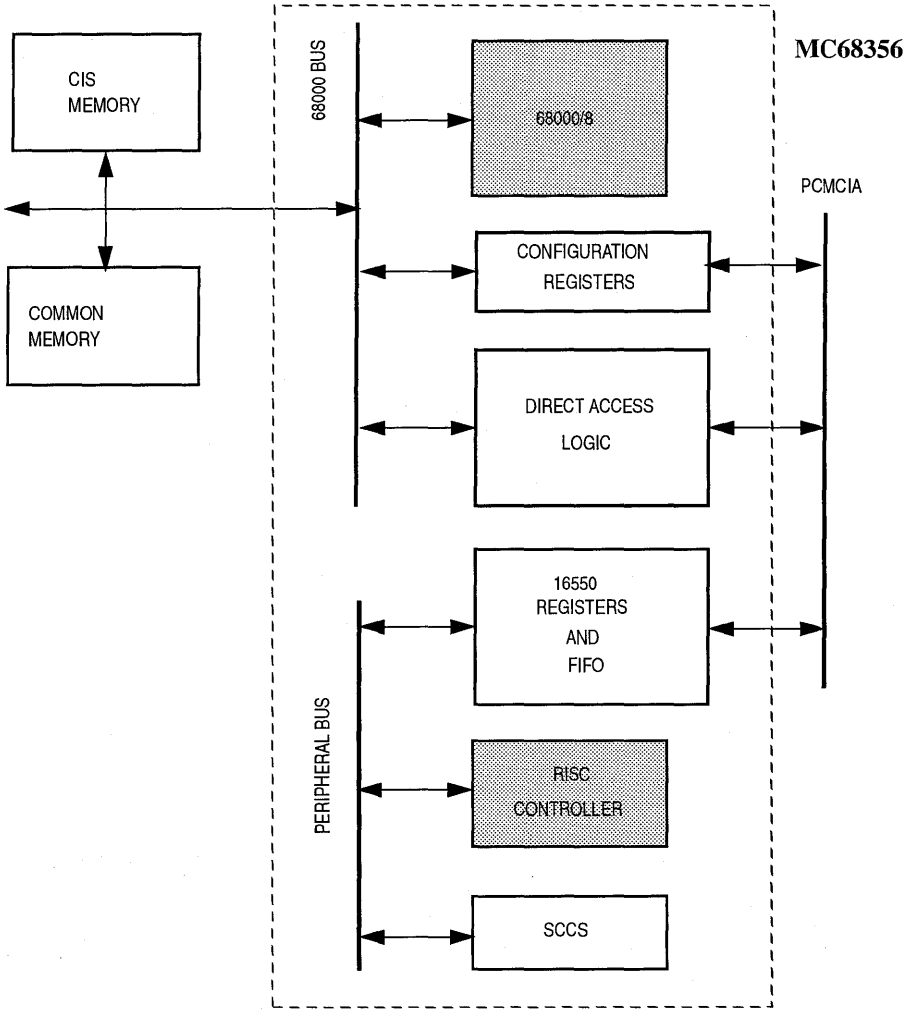


Figure 8-1. PCMCIA Architecture

The following are PCMCIA controller key features:

- Fully Supports PCMCIA Standard 2.1
- Supports Five Configuration Registers
  —Configuration Option Register
  —Card Configuration and Status Register
  —Pin Replacement Register
  —Socket and Copy Register
  —I/O Event Indication Register
- Supports Attribute, Common Memory and I/O Space Access with Optional $\overline{\text{WAIT}}$ Handshake Mechanism
- Common Memory Spaces Are Mapped into 68000 Space Using either of Two Common Memory Base Address Registers
  —Base Address Registers Are Programmable by either the PC or 68000 Core
  —Base Register Can Be Used as Transmit and Receive Buffer Pointers for Ultra-Fast Communication Data Transfers
  —Supports Cycle Steal Transfers and Burst Transfers Arbitration Scheme for Common Memory Accesses
- Supports Full 64 Megabytes of Attribute Memory Space Addressing
  —Card Information Structure Mapped into 68000 Space with Special Base Address Register
- UART 16550 Registers and FIFO Mapped into I/O Space for x86 PC Compatibility
  —I/O Pin Definitions Supported
- Supports Pull-Up/Pull-Down Resistors on PCMCIA Pins (PUCR Register)
- Supports Two Ring Indication Handling Methods:
  —I/O Event Indication Register RI Enable and Event Bits
  —Direct Connection to $\overline{\text{STSCHG}}$ Signal
- Supports PC power management of the card
- Internal Register and Chip Select Lockout of PCMCIA Accesses
- ExCA Compatible

8.1 PCMCIA CONTROLLER FUNCTIONAL OVERVIEW

The PCMCIA controller fully supports PCMCIA standard 2.1. The block diagram of the MC68356 PCMCIA controller is shown in Figure 8-2. The PCMCIA controller interfaces between the PCMCIA bus, and both the 68000 bus and the UART16550 registers. Depending on the memory access mode, the controller will handle the transfer of data from or to the appropriate source or destination. Attribute memory accesses are supported for both the card configuration registers and the card information structure (CIS) memory. The CIS is located in external 68000 memory space. I/O accesses are relayed directly to the UART

16550 registers (Section 7 Communications Processor (CP)). Common memory accesses are mapped into 68000 memory. There is a unique common memory burst access mode which allows a much higher data transfer rate between the PC and the card because it eliminates the WAIT signal handshaking.

NOTE

In this section, the PCMCIA master, or the computer to which the MC68356 slave interface will be plugged into and will communicate with, is referred to as the "PC".

The PCMCIA controller also supports low power modes. The PC can place the card into one of the three supported STOP modes through the use of the PwrDwn bit in the CCSR. The PCMCIA controller is capable of waking up the MC68356 from the stand-by STOP mode when any access is made on the PCMCIA bus. Setting the power down bit can also cause a wake-up from any of the three stop modes.

The PCMCIA controller also supports the ring and packet indication for modem and other I/O cards. In addition to being able to wake up from low power modes when ring indicate (RI) is asserted, the MC68356 can optionally directly notify the PC without waking-up through a direct connection of $\overline{RI}$ to the $\overline{STSCHG}$ pin.

This chapter is structured to cover, first, the three memory space accesses specified by the PCMCIA committee: the attribute, I/O, and common memory spaces. The low power modes and wake up options are covered next, followed by the descriptions of all of the PCMCIA controller registers. Finally the PCMCIA card software initialization procedures are covered.

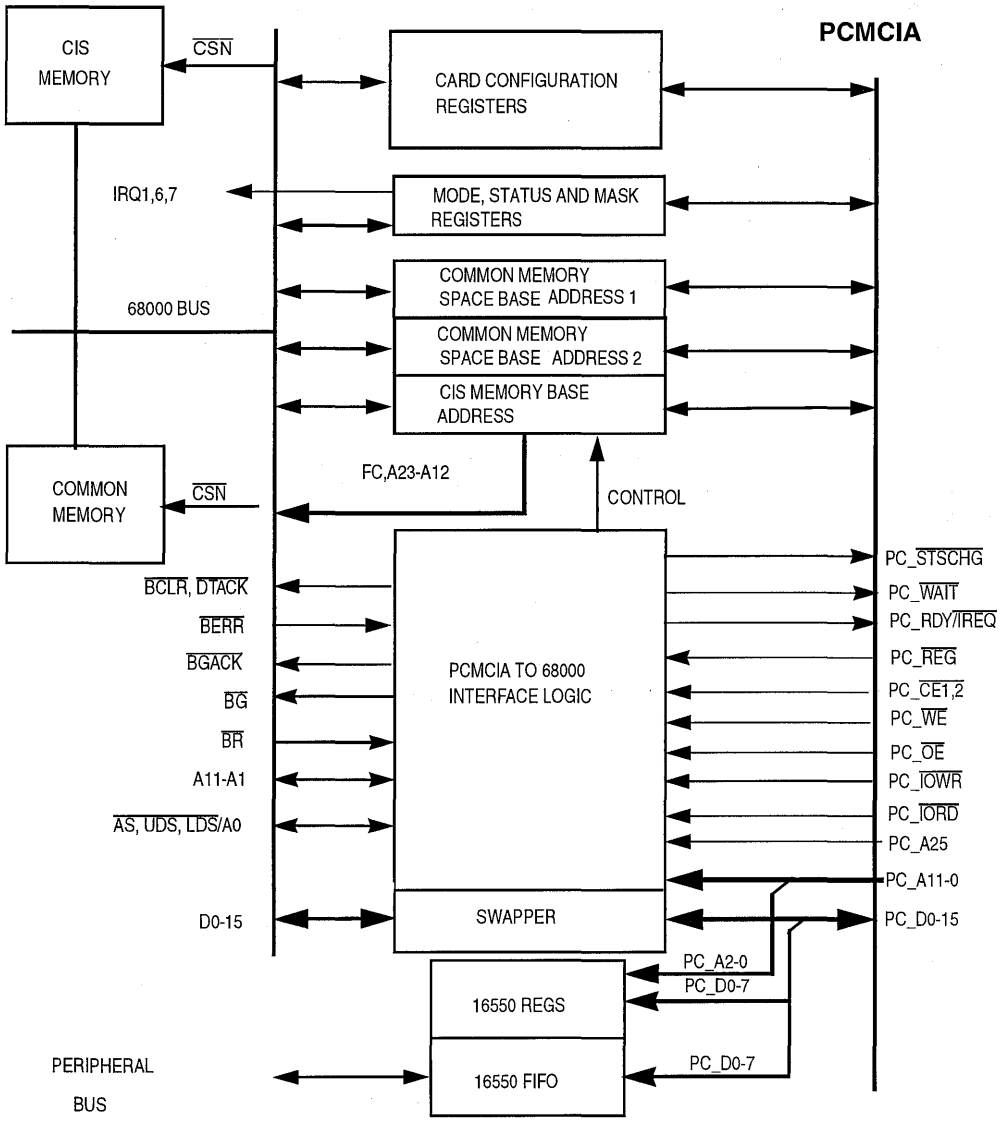


Figure 8-2. PCMCIA Controller Block Diagram

8.1.1 Attribute Memory Accesses

Attribute memory space accesses are used to access the configuration registers and the card information structure (CIS). Attribute memory accesses are generated when the PC asserts the PC_$\overline{\text{REG}}$ pin as shown in Table 8-1 and Table 8-2. As shown in Table 8-3, A25 selects between accessing the CIS with the CIS base address register (CISBAR), and direct

asynchronous accesses to the card configuration registers and MC68356 PCMCIA controller registers. Attribute accesses can only be byte wide and memory locations and accesses must be at even addresses only. See Table 8-1, Table 8-2, and Table 8-3 for signal states and bus validity for the attribute memory read and write function.

8.1.2 Configuration Registers

The card configuration registers are built into the PCMCIA controller hardware. These registers, along with additional 68356 PCMCIA controller specific registers, are visible to the PCMCIA bus and have PCMCIA addresses in the attribute space as shown in Table 8-4. Accesses to these registers are asynchronous with respect to the system clock.

Table 8-1 Attribute Memory Read

Function Mode	PC_REG	PC_CE2	PC_CE1	PC_A0	PC_OE	PC_WE	PC_D15-PC_D8	PC_D7-PC_D0
Standby Mode	x	H	H	x	x	x	High-Z	High-Z
Byte Access	L L	H H	L L	L H	L L	H H	High-Z High-Z	Even- Byte Not Valid
Word Access	L	L	L	x	L	H	Not Valid	Even-Byte
Odd Byte Only Access	L	L	H	x	L	H	Not Valid	High-Z

Table 8-2 Attribute Memory Write

Function Mode	PC_REG	PC_CE2	PC_CE1	PC_A0	PC_OE	PC_WE	PC_D15-PC_D8	PC_D7-PC_D0
Standby Mode	x	H	H	x	x	x	xxx	High-Z
Byte Access	L L	H H	L L	L H	L L	H H	xxx xxx	Even- Byte xxx
Word Access	L	L	L	x	L	H	xxx	Even-Byte
Odd Byte Only Access	L	L	H	x	L	H	xxx	xxx

Table 8-3 Attribute Memory Space Map

PC_CE1	PC_REG	PC_OE	PC_WE	PC_A25	PC_A0	68000 Address	Selected Register or Space
L	L	L	H	L	L	CIS Base Reg (FC,A23-A11) II PCMCIA Address (A0-A10)	CIS Memory Read
L	L	H	L	L	L	CIS Base Reg (FC,A23-A11) II PCMCIA Address (A0-A10)	CIS Memory Write
L	L	L	H	H	L	PCMCIA Address (A0-A7)	Configuration Registers Read
L	L	H	L	H	L	PCMCIA Address (A0-A7)	Configuration Registers Write

8.1.3 Card Information Structure

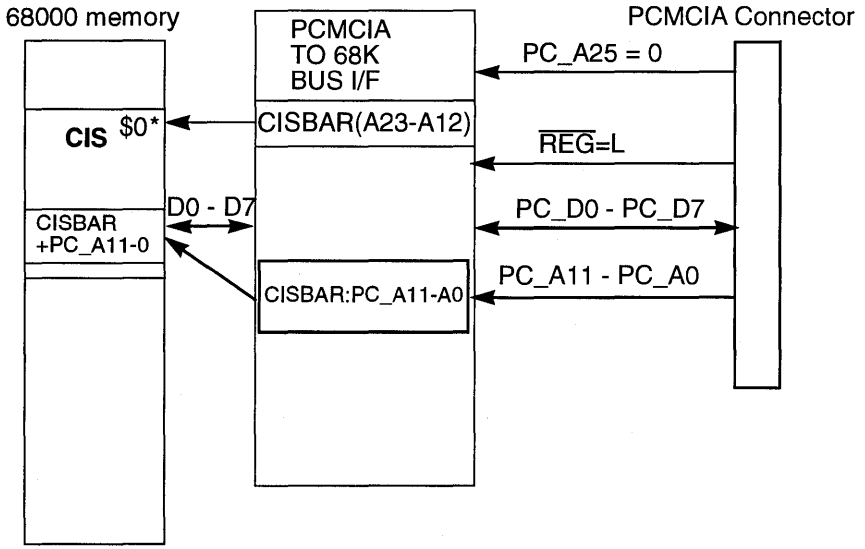
The card information structure is a data structure at address $0 of attribute space in the card that contains information about the card and its capabilities. The CIS will be located in 68000 bus memory space. Location $0 of the CIS is pointed to by CISBAR. Referring to Table 8-3, accesses to the CIS in attribute space are mapped into the 68000 address space with the concatenation of the CIS base address register (CISBAR=> A23-A12) and the low address bits taken from the PCMCIA address lines (PC_A11 - PC_A0). A25 must be equal to zero for CIS accesses to occur.

When the PC accesses the CIS by performing an attribute space read cycle to address 0 through $1FFFFFF (A25=0) the PCMCIA controller detects the cycle, asserts the $\overline{\text{WAIT}}$ signal on the PCMCIA lines and arbitrates for the 68000 bus. When granted, the controller generates the cycle on the 68000 bus. The high address bits and FC are taken from the CIS base address register, the low address bits are taken from the PCMCIA address lines. When the cycle is terminated with $\overline{\text{DTACK}}$, the 68000 bus will be released, data will be transferred to the PCMCIA data lines, and the $\overline{\text{WAIT}}$ signal will be negated.

8

NOTE

The PCMCIA attribute space accesses are only 8-bit. The external CIS memory must be at 68000 even addresses only



* This maps to location zero in attribute space.

Figure 8-3. CIS Mapped into 68000 Space

Table 8-4 PCMCIA Controller Registers Access Map

PC_CE 1	PC_REG	PC_OE	PC_WE	PC_A 25	Offset(PC_A7-PC_A0)	Selected Register or Space
L	L	L	H	H	00	Configuration Option Register
L	L	H	L	H	00	Configuration Option Register
L	L	L	H	H	02	Card Configuration and Status Register
L	L	H	L	H	02	Card Configuration and Status Register
L	L	L	H	H	04	Pin Replacement Register
L	L	H	L	H	04	Pin Replacement Register
L	L	L	H	H	06	Socket and Copy Register
L	L	H	L	H	06	Socket and Copy Register
L	L	L	H	H	08	Reserved1 (for future PCMCIA Definition)
L	L	H	L	H	08	Reserved1 (for future PCMCIA Definition)
L	L	L	H	H	0A	Reserved2 (for future PCMCIA Definition)
L	L	H	L	H	0A	Reserved2 (for future PCMCIA Definition)
L	L	L	H	H	0C	Reserved3 (for future PCMCIA Definition)
L	L	H	L	H	0C	Reserved3 (for future PCMCIA Definition)
L	L	L	H	H	0E	Reserved4 (for future PCMCIA Definition)
L	L	H	L	H	0E	Reserved4 (for future PCMCIA Definition)
L	L	L	H	H	20	PCMCIA Mode Register Read
L	L	H	L	H	20	PCMCIA Mode Register Write
L	L	L	H	H	2A	PCMCIA Event Register Read
L	L	H	L	H	2A	PCMCIA Event Register Write
L	L	L	H	H	30	CIS Base Address Register Read
L	L	H	L	H	30	CIS Base Address Register Write
L	L	L	H	H	38	Common Memory Space Base Address Register 1 Read
L	L	H	L	H	38	Common Memory Space Base Address Register 1 Write
L	L	L	H	H	40	Common Memory Space Base Address Register 2 Read
L	L	H	L	H	40	Common Memory Space Base Address Register 2 Write

8.1.4 I/O Space Accesses

The MC68356 fully emulates the 16550 UART (Section 7 - Communications Processor (CP)). The host can transfer data to the card by writing or reading the 16550 FIFO which is mapped into the PCMCIA address space. I/O accesses are initiated by the assertion of either the PC_IORD or the PC_$\overline{\text{IOWR}}$ signal and the PC_$\overline{\text{REG}}$ signal. All I/O accesses are connected directly to the 16550 emulation logic registers without using the 68000 bus.

The 16550 emulation registers addresses are located at addresses $0 through $7 in the PCMCIA I/O address space, therefore PCMCIA address lines 3 to 25 are ignored for I/O accesses. (See Table 8-7). Refer to Table 8-5 and Table 8-6 for signal states and bus validity for the I/O read and write function.

Table 8-5 I/O Input Accesses

Function Mode	PC_REG	PC_CE2	PC_CE1	PC_A0	PC_IORD	PC_IOWR	PC_D15-PC_D8	PC_D7-PC_D0
Standby Mode	x	H	H	x	x	x	High-Z	High-Z
Byte Access	L L	H H	L L	L H	L L	H H	High-Z High-Z	Even- Byte Odd-Byte
Word Access	L	L	L	x	L	H	Odd-Byte	Even-Byte
I/O inhibit (during DMA)	H	X	X	X	L	H	High-Z	High-Z
High Byte Only	L	L	H	x	L	H	Odd-Byte	High-Z

Table 8-6 I/O Output Accesses

Function Mode	PC_REG	PC_CE2	PC_CE1	PC_A0	PC_IORD	PC_IOWR	PC_D15-PC_D8	PC_D7-PC_D0
Standby Mode	x	H	H	x	x	x	xxx	xxx
Byte Access	L L	H H	L L	L H	H H	L L	xxx xxx	Even- Byte Odd-Byte
Word Access	L	L	L	x	H	L	Odd-Byte	Even-Byte
I/O Inhibit (during DMA)	H	X	X	X	H	L	xxx	xxx
High Byte Only	L	L	H	x	H	L	Odd-Byte	XXX

Table 8-7 Card I/O Space Address Map

PC_REG	PC_CE1	PC_CE2	PC_IORD	PC_IOWR	PC_A0 - PC_A2	Selected Register or Space
L	L	H	L	H	0-7*	16550 - 8 bytes read
L	L	H	H	L	0-7*	16550 - 8 bytes write

*See Table 5-4 for specific 16550 register address locations.

8.1.5 Common Memory and Direct Access Mode Accesses

Common memory reads and writes are initiated by the PC when it does a normal access by *not* asserting the PC_$\overline{\text{IORD}}$, PC_$\overline{\text{IOWR}}$, or the PC_$\overline{\text{REG}}$ pins. See Table 8-8 and Table 8-9 for signal states for reads and writes.

Common memory reads and writes are mapped directly onto the 68000 bus using a concatenation of the lower 12 address lines from the PCMCIA address bus and upper address lines from either of the two common memory base address registers (CMBAR1,2). The base address register to be used is selected by A25 (CMBAR1 is used when A25=1, and CMBAR2 is used when A25=0). Both the PC and the 68000 can program the CMBARs. See Table 8-10.

In addition to the being able to perform single common memory accesses between the PCMCIA bus and the 68000 bus there is also a feature that allows the PCMCIA controller to hold onto the 68000 bus and perform multiple reads or writes without having to rearbitrate for the 68000 bus. This allows the PCMCIA side access cycles to be carried out without having to assert the $\overline{\text{WAIT}}$ signal for each access. This mechanism works as follows:

When the PC initiates a common memory space cycle on the PCMCIA interface, the PCMCIA controller will assert the $\overline{\text{WAIT}}$ signal on the PCMCIA lines and arbitrate for the 68000 bus. When granted, the PCMCIA controller will generate the cycle on the 68000 bus. When the cycle is terminated with $\overline{\text{DTACK}}$, the PCMCIA controller may be programmed to maintain the bus ownership for the next cycle or release the bus. For burst access cycles (FAST=1 in PCMR register) bus ownership will be retained. Data is transferred to the PCMCIA data lines and the $\overline{\text{WAIT}}$ signal will be negated and the next cycle can take place without $\overline{\text{WAIT}}$ being reasserted. Subsequent accesses can occur until the PC has completed the burst and the PCMCIA bus is idle for a programmable number of 68000 bus clocks (CLKO) as specified by the ArbIDL bit in the PCMR.

The PCMCIA controller may be programmed to automatically increment the CMBAR when the PC reaches the current 2K page boundary defined by CMBAR (i.e. A11-A0=FFF). This mechanism is useful when the PC is using a DMA cycle to transfer data to or from the card because it eliminates the need to reprogram the base address register each time a page boundary is reached.

The two common memory base address registers (CMBARs) can be used to read and write to separate spaces in 68K memory, forming pointers to transmit and receive buffers. This can be especially useful in high speed communication cards where it is possible to bypass the use of the UART 16550 and simply transfer data directly to and from 68000 memory. Figure 8-4 shows how the CMBARs can be used as transmit and receive data buffer pointers. Once the data transfer begins, the PC can burst transfer data into 68K memory, incrementing the lower addresses for each transfer and using the A25 line to select between transmit and receive buffers. The CMBAR auto increment feature is also useful in this case if the buffer size needs to be larger than 2 Kbytes. By using this method, data transfer rates are limited only by the 68000 bus cycle rate or the PC bus transfer rate.

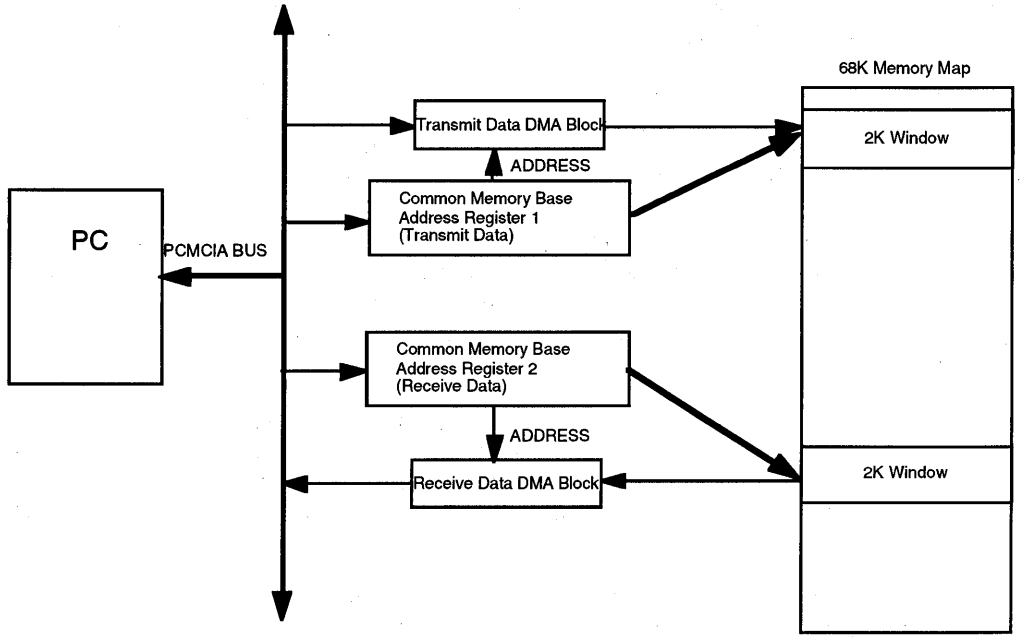


Figure 8-4. Use of the CMBARs for Fast Mode Transmit and Receive Data Transfers.

To eliminate paging entirely, the PCMCIA controller may be programmed not to drive the high address lines from the base register, but rather drive the ABUF pin which has the appropriate timing to control external address buffers. This address buffer control line is connected to external address buffer circuitry that interfaces the upper PCMCIA address bus lines to the 68000 bus. This feature is enabled by setting the ABUF bit in the PCMR register (See Figure 8-5)

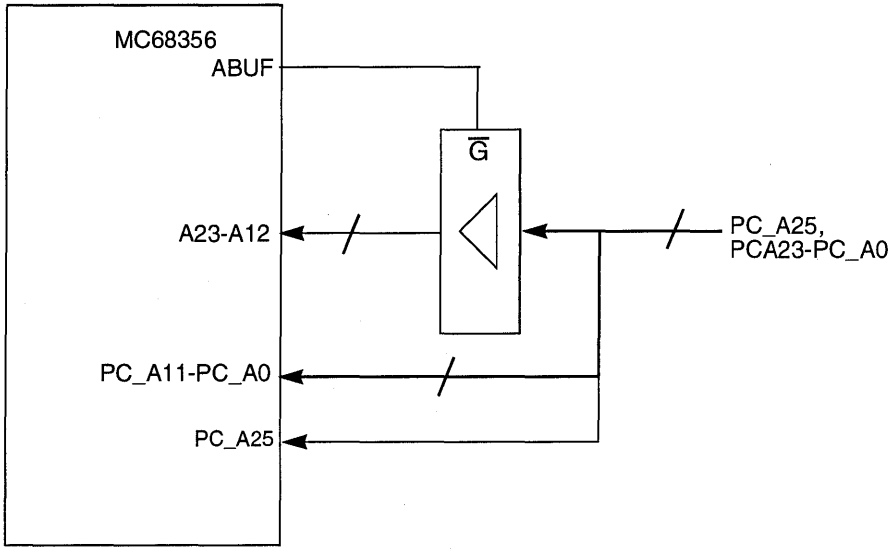


Figure 8-5. PCMCIA Direct Addressing Using ABUF

To summarize, common memory accesses can occur using two different modes:

- Normal mode (FAST =0): The PCMCIA controller will assert the $\overline{\text{WAIT}}$ signal on the PCMCIA lines and generate a 68000 cycle. When the cycle is terminated with $\overline{\text{DTACK}}$, data will be transferred to the PCMCIA data lines and the $\overline{\text{WAIT}}$ signal will be negated. The next normal mode cycle can then be initiated.
- Fast mode(FAST=1): This mode is used to eliminate the $\overline{\text{WAIT}}$ on the PCMCIA interface and perform fast direct access transfers. The first cycle (with arbitration) is as described above. From the second cycle until the burst is terminated, the PCMCIA controller **will not** assert the $\overline{\text{WAIT}}$ signal on the PCMCIA line. The cycle on the 68000 bus will use the memory interface signals $\overline{\text{WE}}$ and $\overline{\text{OE}}$ but not $\overline{\text{DS}}$. The cycle will be terminated when the PCMCIA cycle is terminated ($\overline{\text{DTACK}}$ and $\overline{\text{BERR}}$ are ignored in this mode). The PCMCIA controller will release the 68000 bus when the (PC) does not access the card for a programmable number of clocks.

NOTE

FAST mode accesses are not allowed for PC accesses to the IMP internal space. Erratic operation will result.

Table 8-8 Common Memory Read

Function Mode	PC_REG	PC_CE2	PC_CE1	PC_A0	PC_OE	PC_WE	PC_D15-PC_D8	PC_D7-PC_D0
Standby Mode	x	H	H	x	x	x	High-Z	High-Z
Byte Access	H H	H H	L L	L H	L L	H H	High-Z High-Z	Even- Byte Odd-Byte
Word Access	H	L	L	x	L	H	Odd-Byte	Even-Byte
Odd Byte Only Access	H	L	H	x	L	H	Odd-Byte	High-Z

Table 8-9 Common Memory Write

Function Mode	PC_REG	PC_CE2	PC_CE1	PC_A0	PC_OE	PC_WE	PC_D15-PC_D8	PC_D7-PC_D0
Standby Mode	x	H	H	x	x	x	xxx	xxx
Byte Access	H H	H H	L L	L H	H H	L L	xxx xxx	Even- Byte Odd-Byte
Word Access	H	L	L	x	H	L	Odd-Byte	Even-Byte
Odd Byte Only Access	H	L	H	x	H	L	Odd-Byte	XXX

8

Table 8-10 Card Common Memory Space Address Map

PC_REG	PC_CE1	PC_CE2	PC_OE	PC_WE	PCMCIA Offset Address	68000 Base Address	Selected Register or Space
H	L	H/L*	L	H	A25=1;	Common memory base reg1 FC,A23-11	Common memory read accesses into 68000 bus
H	L	H/L*	H	L	A25=1;	Common memory base reg1 FC,A23-11	Common memory write accesses into 68000 bus
H	L	H/L*	L	H	A25=0;	Common memory base reg2 FC,A23-11	Common memory read accesses into 68000 bus
H	L	H/L*	H	L	A25=0;	Common memory base reg2 FC,A23-11	Common memory write accesses into 68000 bus

*PC_CE2 will remain unasserted during byte accesses.

8.1.6 Protecting Memory and Internal Space from PCMCIA Accesses

Each chip select can be set to block PCMCIA accesses to its memory space. When the chip select protection bit is set, the external PCMCIA host cannot access the memory block. The protection for each chip select block can be enabled by setting the PCS(3-0) bits in the PCMCIA protection register (PPR). In addition, internal IMP registers and dual port RAM access from the PCMCIA bus can also be disabled using the PIR bit in the PPR. For more information see 6.6.2.3 PCMCIA Protection Register (PPR).

8.1.7 PCMCIA Controller Initialization

After hardware reset the PCMCIA controller assumes the default "memory only" card configuration, with the RDY/$\overline{\text{BSY}}$ signal being low to indicate the busy condition. The 68000 core should initialize the CIS base address register with the high address of the external CIS memory address. The CIS memory may be located in external IMP RAM/ROM which is pointed to by the CIS base address register (CISBAR). The MC68356 has five configuration registers, according to the PCMCIA standard, plus the three registers reserved for future use. These registers are located on the PCMCIA attribute address space and can be accessed by the PC when A25 is high (PCMCIA attribute addresses $2000000 or higher). The CIS's configuration tuple TPCC_RADR should have the base address of these registers (PCMCIA attribute addresses $2000000 or higher).

The 68000 core software may then signal that the card is ready by setting the RDY/$\overline{\text{BSY}}$ line in the pin replacement register (PRR).

The host reads the CIS from attribute memory. This is done by the host performing an attribute space read cycle to address 0. The PCMCIA controller detects the cycle, asserts the $\overline{\text{WAIT}}$ signal on the PCMCIA lines and arbitrates for the 68000 bus. When granted, the controller generates the cycle on the 68000 bus. The high address bits and FC are taken from the CIS base address register, the low address bits are taken from the PCMCIA address lines. When the cycle is terminated with $\overline{\text{DTACK}}$, the 68000 bus will be released, data will be transferred to the PCMCIA data lines, and the $\overline{\text{WAIT}}$ signal will be negated. The host reads the CIS by this mechanism.

The host can then initialize the configuration registers. The host can access these registers directly without using the 68000 bus. When A25 is high on an attribute space cycle, the PCMCIA controller can asynchronously access the configuration registers. Those registers can also be accessed by the 68000 core. The 68000 core can read and write these registers like any other register in the 68000 address space. The PCMCIA controller may also generate an interrupt to the 68000 core when one of the registers is written by the PC.

After the initialization is completed, the MC68356 is configured for either PCMCIA common memory or I/O transfers depending on the value written in the configuration index of the configuration option register (COR). If the configuration index is non-zero, the card will respond to I/O cycles and the I/O pin functionality replaces the memory -card-only pin configuration. I/O data transfers are used to read and write the 16550 registers.

8.1.8 PCMCIA to 68000 Bus Access and Monitoring Options

The PCMCIA controller can be programmed to allow the SDMA, the DSP2IMP mechanism, the core interrupt acknowledge routine, and external peripherals to use the 68000 bus between PCMCIA transfers by setting the BCLR bit in the PCMR register. The PCMCIA controller can also be programmed to generate a bus clear signal and gain mastership of the bus when it is not immediately granted upon request. This feature is enabled by setting the BCLROE bit in the PCMR.

If a PCMCIA to 68000 bus cycle is terminated with $\overline{\text{BERR}}$, the PCMCIA controller can be programmed to assert an interrupt to the PC through the INTRL bit in the PCMR. The $\overline{\text{BERR}}$ status bit can be read by the PC through the BERR bit in the PCHER register.

There is also a PCMCIA bus watchdog timer which can issue an interrupt to the PC as well as the 68000 core. The counter will count the $\overline{\text{WAIT}}$ length and, if expired, the PCMCIA cycle will be terminated and the PTIE interrupt to the PC will be generated (if enabled). The interrupt will be generated to the PC through the PTIE bit in the PCHER if the TIEn bit is set in the PCMR. The five-bit counter-preset is programmed through the TValue bits in the PCMR. The prescaler value is programmed in the TPres bits in the PCMR. The PCMCIA bus watchdog timer signal source is the IMP system clock.

NOTE

The features described in the above three paragraphs also apply for CIS accesses in attribute space.

NOTE

All PCMCIA locations (registers, CIS memory, etc.) that can be accessed by the host may also be accessed by the 68000 core. See Section 5 Memory Map for 16550 register accessibility.

8.2 POWER DOWN OPTIONS

The PC can place the MC68356 into low power mode by setting the power down (PwrDwn) bit in the CCSR. When the PC sets the PwrDwn bit, the PCMCIA controller will generate an interrupt to the 68000 core (if enabled through the INTRL bits in the PCMR). The IMP interrupt routine should place the devices on the card (including the DSP) in a low power mode and then place itself into one of the low power modes listed in Table 3-2.

NOTE

The host should not change the PwrDwn bit in the CCSR while the card is busy. (i.e. The RDY/$\overline{\text{Bsy}}$ bit in the PRR is zero).

If the 68000 core is in a stand-by mode when the PwrDwn bit is set, the 68000 core will come out of stand-by mode to process the PwrDwn interrupt as above. Setting the PwrDwn bit will reset the RDY/$\overline{\text{Bsy}}$ bit in the pin replacement register (PRR) to zero. This bit will be set to one (RDY) when the IMP completes its transition to the power-down (STOP) mode.

Of the three different power down modes, the STAND_BY mode is the only one in which a normal host PCMCIA interface access will cause the IMP to resume operation (see 8.2.2.1 Wake Up on PCMCIA Access in STAND-BY mode).

The 68000 core can also place the card into one of four different low power modes when the card is idle for long periods of time and it is desired to conserve power (Table 3-2) even if the PC does not initiate the power down transition.

The PC can access the configuration registers while in power-down.

Table 8-11 IMP Low Power Modes

Low Power Mode	Method of Entry	Comments
Slow Go	Write to DF3 -DF0 in IOMCR	PCMCIA I/F functional, no wake up time, higher power consumption than other STOP mode.
Stand-by	STOP, LPM = 01	The imp can detect any PCMCIA access and wake-up. 2-5 cycle wake-up time.
Doze	STOP, LPM = 10	The IMP wakes up on setting of PwrDwn bit in CCSR, 2500 cycle wake-up time.
Stop	STOP, LPM = 11	The IMP wakes up on setting of PwrDwn bit in CCSR, 70000 max. cycle wake-up time.

In addition to placing itself in a STOP mode, the 68000 core can also lower the system clock frequency to save power by writing a higher value divide factor to the PLL clock divider. In this mode, the IMP and PCMCIA interface is fully functional. For more information see 3.5 IMP Power Management.

Before placing the IMP into low power STOP mode it is usually desirable to place the DSP into a low-power mode first. For information on how to place the DSP into low power mode refer to DSP56KFAMUM/AD the *DSP56000 Digital Signal Processor Family Manual.*

8.2.1 PCMCIA Ring Indication

The MC68356 PCMCIA interface has several options for handling the ring indicate $\overline{RI}$ (or PB9) signal either by automatically passing the indication to the PC or by first allowing the 68000 core to process it. Figure 8-6 shows a diagram of possible routings of the ring indicate signal. Note that the IMP can be programmed to wake up from any of the three stop modes upon assertion of the RI signal.

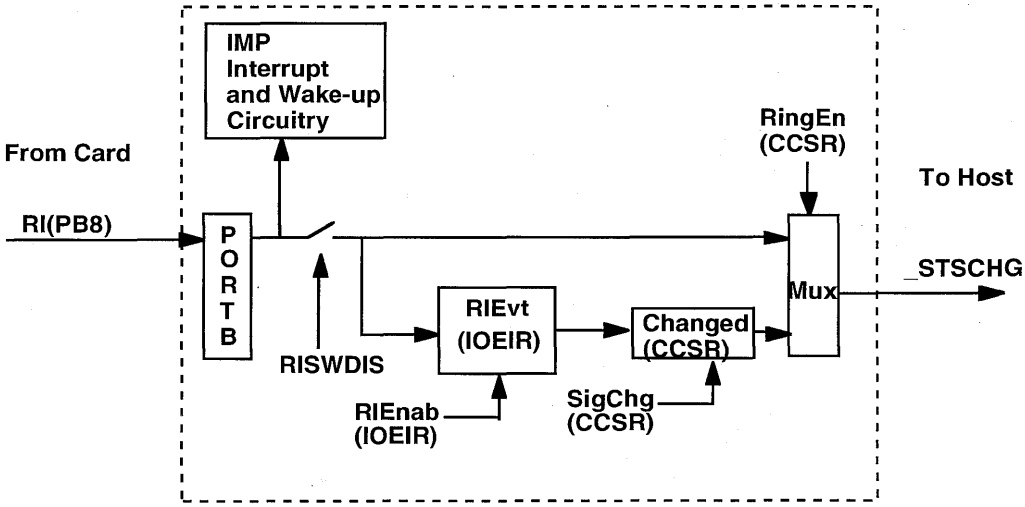


Figure 8-6. Ring Indicate (PB9) Connection Options

The MC68356 can transfer the $\overline{RI}$ (PB9) to the host using either of three programmable methods:

1. Pass the $\overline{RI}$ signal directly to the PC through the RIEvt bit in the I/O event indication register (IOEIR) which also will set the changed bit in the card configuration and status register (CCSR) and can also optionally assert the $\overline{STSCHG}$ pin (if the SigChg bit is set in the CCSR). When the $\overline{RI}$ (PB9) is asserted, the RIEvt bit in the I/O event indication register(IOEIR) will be set to one. If the RIEnab has been set by the PC, the changed bit in the CCSR will be set. If the SigChg bit in the CCSR has been set to one also, the $\overline{STSCHG}$ pin will be driven low. In addition, the PC may poll the RIEvt bit in the IOEIR register, or the changed bit in the CCSR at all times. When the host detects the $\overline{RI}$, it should clear out the RIEvt bit.
2. The 68000 may poll or take an interrupt off of the $\overline{RI}$ in the port B data register (PB9) and update the RIEvt bit in the IOEIR. This allows the 68000 software to debounce or otherwise process the RI signal to detect a valid ring signal before passing it on to the PC. The RISWDIS bit in the PCMR register should be set which will break the direct connection of the $\overline{RI}$ signal to the RIEvt bit in the IOEIR register (see Figure 8-6).
3. Connect the $\overline{RI}$ directly to the $\overline{STSCHG}$ pin by setting RingEn bit in the CCSR. This mode complies with Intel ExCA specification for host $\overline{RI}$ using $\overline{STSCHG}$. When the RingEn bit in the CCSR register is set to one by the PC, the $\overline{STSCHG}$ line reflects the state of the $\overline{RI}$ (PB9) pin, all other status change events are disabled. When the RingEn bit is zero this function is disabled.

8.2.1.1 Wake Up Using the PwrDwn Bit

The host can wake up the card from any of the three STOP modes by resetting the PwrDwn bit in the CCSR. The IMP system clocks will start up, the 68000 core will receive an interrupt, and may optionally wake-up the DSP. Resetting the PwrDwn bit will set the RDY/$\overline{Bsy}$ bit to zero in the pin replacement register. The 68000 core should set the RDY/$\overline{Bsy}$ bit to one upon completing its wake-up routine.

NOTE

The user should reset enable and event bits in the IMP wake-up control register – IWUCR after it has recovered from STOP mode. (See 3.5.2.3 IMP Wake-Up from Low Power STOP Modes.)

8.2.2 Wake Up Options

There are several methods that allow the PC to wake up the IMP and subsequently the DSP. The IMP will wake-up when:

1. The PC accesses the card and the IMP is in STAND-by mode.
2. The PwrDwn bit is reset.
3. The modem $\overline{RI}$($\overline{PB9}$) or the PB10 signal is asserted, if enabled (See 3.5.2.3 IMP Wake-Up from Low Power STOP Modes.)
4. The (PIT) timer is enabled and expires, if enabled (See 3.5.2.3 IMP Wake-Up from Low Power STOP Modes.)

5. The IMP RESET and HALT is asserted.

8.2.2.1 Wake Up on PCMCIA Access in STAND-BY mode

The IMP will wake up on any of five different types of accesses to the PCMCIA interface if it is in the STAND-BY low power mode. The PCMCIA write event register (PCAWER) indicates the type of access that has taken place to wake-up the IMP. The PCMCIA write event mask registers (PCAWMR) is used to enable the wake-up mechanism for a particular type of PCMCIA access. The following accesses can be programmed as described above to wake up the IMP from the STANDY-BY mode:

- I/O space access.
- Attribute space Access
- Common memory space access (1 or 2)
- Mode registers write access (when one of the PCMCIA registers accessible by the PC is written to).

8.2.2.2 Power Down and Wake Up Using the PwrDwn Bit

The Rdy/$\overline{\text{Bsy}}$ bit is reset to zero (busy) by the PCMCIA logic if the host writes a one to the PwrDwn bit in the CCSR, and will stay in the busy state until the IMP has completed its transition to one of the low power STOP modes or can be set back to one by software (if it is not desired by the card to enter STOP mode). Once the IMP has entered low power mode, the Rdy/$\overline{\text{Bsy}}$ bit will be set to ready. When the host writes a zero to the PwrDwn bit while the IMP is in low power mode, the Rdy/$\overline{\text{Bsy}}$ bit will again be reset to busy while the IMP wakes up. IMP software should set the Rdy/$\overline{\text{Bsy}}$ bit to one when it has finished recovering from low power mode. Refer to Figure 8-7 for a more detailed description of the transition to, and wake up from low power stop mode using the PwrDwn bit.

8.2.2.3 PCMCIA Host Interrupts

The host can be interrupted by the PCMCIA controller in I/O mode through the $\overline{\text{IREQ}}$ pin which is the Rdy/$\overline{\text{Bsy}}$ pin in memory-only mode. The $\overline{\text{IREQ}}$ pin's status is always reflected by the INTR bit in the CCSR for PCMCIA interrupts:

1. The PCMCIA controller will set the INTR bit and the $\overline{\text{IREQ}}$ pin in I/O mode if the PTIE or BERR bit in the PCHER register is set AND the PTIEn and BERRIE bits in the PCMR register are set respectively.
2. The 68000 core can set the INTR bit in the CCSR by setting the SW bit in the PCHER which will cause the $\overline{\text{IREQ}}$ pin to assert.
3. When an unmasked 16550 interrupt is issued from the 16550 controller, the INTR bit in the CCSR will be set and the $\overline{\text{IREQ}}$ pin will be asserted.

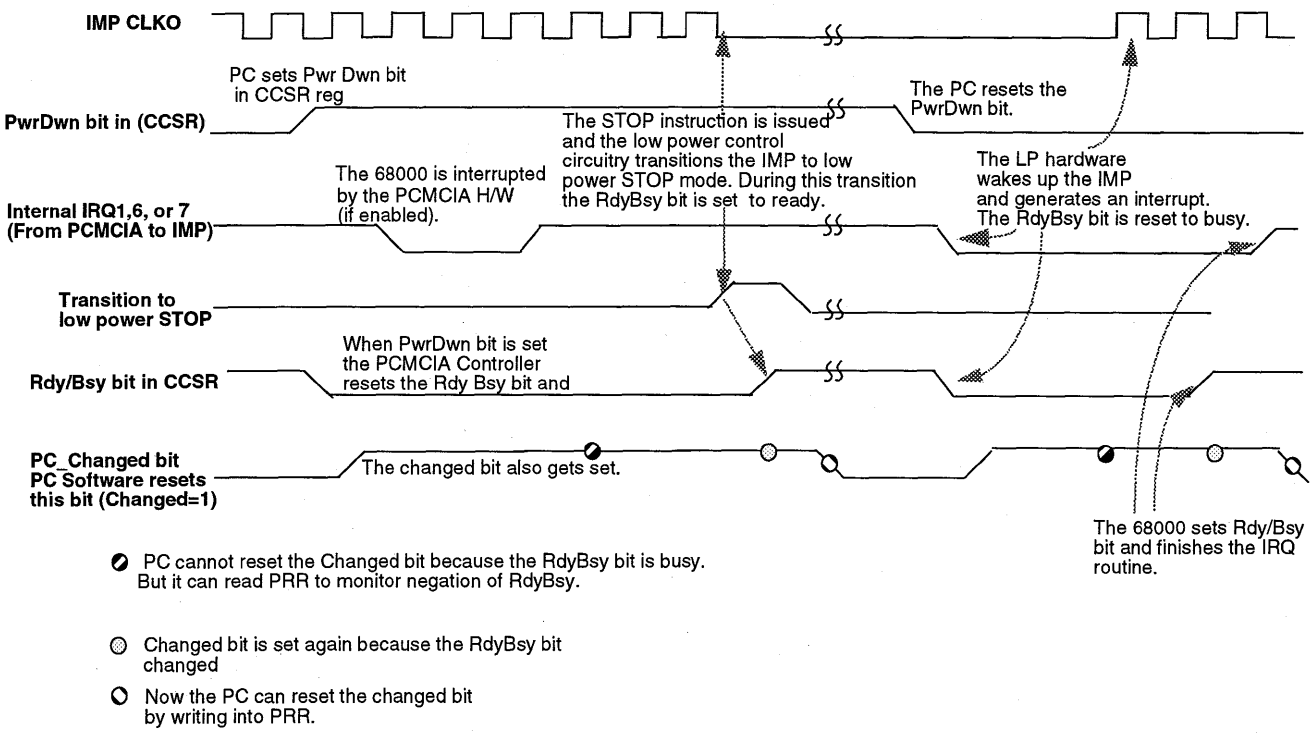


Figure 8-7. Using the PwrDwn bit to Enter and Exit Low Power Mode

8.2.2.4 The Ready Busy Signal (Rdy/$\overline{\text{Bsy}}$)

The ready busy signal can be passed to the host through the Rdy/$\overline{\text{Bsy}}$ ($\overline{\text{IREQ}}$ in I/O mode) pin in memory only mode, and the Rdy/Bsy bit in the pin replacement register (PRR) in I/O mode.The RDY/$\overline{\text{BSY}}$ pin can be controlled by the 68000 by writing to the Rdy/$\overline{\text{Bsy}}$ bit in the PRR. The state of the Rdy/$\overline{\text{Bsy}}$ pin can be monitored in the PRR (even though the PCMCIA interface is configured as a memory only interface).

The RdyBsy bit is also controlled during the transitions to and from the three low power STOP modes. See 8.2.2.2 Power Down and Wake Up Using the PwrDwn Bit for more details.

8.3 PCMCIA PINS

8.3.1 PCMCIA Pins Supported by the MC68356

The pin descriptions for the MC68356 PCMCIA pins are given in Section 2 Signal Description. The MC68356 will read the PC_EN pin at IMP reset and if this pin is strapped to Vcc it will configure the pins with PCMCIA functionality as PCMCIA pins. If it is strapped to ground, the pins will revert to their non-PCMCIA functions.

Most PCMCIA pins have internal pullup or pulldown resistors (see 8.3.3 Pullup Control Register – PUCR).

NOTE

The PCMCIA pins are denoted in the signal description (Section 2) as PC_*SIGNALNAME.*

8.3.2 PCMCIA Pins Not Supported

The following pins are not implemented on the MC68356 and should be supported with external circuitry.

Write Protect (WP)

This output signal reflects the status of the card's write protect switch. When a card or socket is configured for the I/O interface, the write protect status may be available in the pin replacement register, and the signal is replaced in the interface by the I/O Port IS 16 bits ($\overline{\text{IOIS16}}$) signal.

I/O IS 16 Bit Port ($\overline{\text{IOIS16}}$)

This output signal is asserted when the address at the socket corresponds to an I/O address to which the card responds, and the I/O port being accessed is capable of 16-bit access.

When this signal is not asserted during a 16-bit I/O access, the system will generate 8-bit references to the even and odd bytes of the 16-bit port being accessed.

NOTE

The MC68356 PCMCIA controller does not support 16-bit I/O accesses. Therefore this pin should be tied high (not asserted) on the card.

Input Acknowledge ($\overline{\text{INPACK}}$) [I/O Operation]

This output signal is asserted when the card is selected and the card can respond to an I/O read cycle at the address on the address bus.

NOTE

If enabled for I/O mode accesses, the MC68356 PCMCIA controller responds to I/O read cycles at all addresses. All I/O accesses are routed to the 16550 registers. This signal may thus be asserted whenever the card enable ($\overline{\text{CE1}}$ and $\overline{\text{CE2}}$) and the $\overline{\text{REG}}$ inputs are true.

Audio Digital Waveform ($\overline{\text{SPKR}}$) [replace BVD2]

8

Binary audio output signal.

Card Reset (RESET)

This signal clears the card configuration option register thus placing the card in an unconfigured (memory only interface) state. A card remains in the unconfigured state until the card configuration register has been written.

In most cases this signal should trigger a reset sequence in the IMP and DSP of the MC68356. External circuitry such as a one shot circuit will typically be required to assure that the IMP and DSP resets are asserted long enough to meet the 68000 and 5600 minimum reset specifications (see Section 14 Electrical Characteristics) when the card is reset.The card reset signal should be an input to the external reset circuitry in order to guarantee that the reset polarity and pulse times meet the requirements of the IMP and DSP (if applicable).

8.3.3 Pullup Control Register – PUCR

The PUCR contains control for the optional pullup resistors on pins with PCMCIA functionality. This register can be read and written by the 68000 core.

A set bit disables the pullup for a particular block of pins. A cleared bit enables the pullups in a particular block. Some pins have pull-ups available for non-PCMCIA functionality and some do not as noted in the bit descriptions.

PUCR 68356 Base+$8E0

7	6	5	4	3	2	1	0
3.3VM	PCADDC	PDI0–7C	PCIOC	PB8–11C	PB0–7C	PA8–15C	PA0–7C
RESET: 0	0	0	0	0	0	0	0

Read/Write

3.3VM—3.3V Mode

This bit should be set when the operating the MC68356 at 3.3V to ensure proper pullup resistance values.

PCADDC - PCMCIA Address Control (CD1_/ PC_A11, CTS1_/ PC_A10, PC_A9 - PC_A2, BCLR_/ PC_A1, RMC_/ PC_A0)

This bit, when cleared, in PCMCIA mode only (PC_EN = 1), will enable a 100K ohm pulldown resistor for each of these PCMCIA address pins. Setting this bit will disable the pulldowns. In non PCMCIA mode (PC_EN = 0) these pulldowns are disabled.

NOTE:

PC_A11 and PC_A10 pins may be placed on the $\overline{CD1}$, $\overline{CTS1}$ pins or on the $\overline{TOUT1}$ and $\overline{TOUT2}$ pins. The IPins bit in the PCMCIA mode register (PCMR) controls the selection. If the PC_A11 and PC_A10 pins are placed on the $\overline{TOUT1}$ and $\overline{TOUT2}$ pins, internal pulldowns are not available and must be implemented externally.

PDI0–7C—Port D 0–7 Control (PCMCIA Control)

This bit, when cleared, will enable a 10K ohm pullup resistor for each of the PCMCIA control pins, or the PDI0–7 pins (if configured as an input). Setting this bit will disable the pullups.

PCIOC—Port C 0–7 Control (PCMCIA D0–7)

This bit, when cleared, in PCMCIA mode only (PC_EN = 1), will enable a 100K ohm pulldown resistor for each of the PCMCIA data pins 0-7. Setting this bit will disable the pulldowns.

In non-PCMCIA mode (PC_EN = 0) these pulldowns are disabled.

PB8–11C—Port B 8–11 Control

This bit, when cleared, will enable a 10K ohm pullup resistor for each of the PCMCIA control pins, or the PB8–11 pins (if configured as an input). Setting this bit will disable the pullups.

PB0–7C—Port B 0–7 Control

This bit, when cleared, will enable a 10K ohm pullup resistor for each of the PCMCIA control pins or the PB0–3, PB5 and PB7 pins (if configured as an input). Setting this bit will disable the pullups. Note that PB4 and PB6 do not have pullups or pulldowns in the chip.

PA8–15C—Port A 8–15 Control

This bit, when cleared, will enable a 10K ohm pullup resistor for each of the PA8-15 pins (if configured as an input). Setting this bit will disable the pullups.

PA0–7C—Port A 0–7Control (PCMCIA D8–15)

This bit, when cleared, in PCMCIA mode only (PC_EN = 1), will enable a 100K ohm pull-down resistor for each of the PCMCIA data pins 8-15. Setting this bit will disable the pull-downs.

In non-PCMCIA mode (PC_EN = 0) these pulldowns are disabled.

8.4 PROGRAMMER'S MODEL

The PCMCIA controller contains a number of registers, described in the following paragraphs. Those registers can be accessed by the host and the 68000 core.

8.4.1 PCMCIA Controller Accesses

8.4.2 PCMCIA Mode Register - PCMR

8

PCMR

PCMCIA Address:A25=1;$20
68000 Address: BAR + $8C0

31	30	29	28	27	26	25	24
TIEn	BERRIE	BCLROE	CLRIE	SWAP	FAST	ArbIDL	
RESET							
0	0	0	0	0	0	0	0

PCMCIA Address:A25=1;$22
68000 Address: BAR + $8C1

23	22	21	20	19	18	17	16
TValue					TPres		Ten
RESET							
0	0	0	0	0	0	0	0

PCMCIA Address:A25=1;$24
68000 Address: BAR + $8C2

15	14	13	12	11	10	9	8
RISDIS	AINC2	AINC1	IPins	INTRL		16bit	ABUF
RESET							
0	0	0	0	0	0	0	0

Read/Write

ArbIDL—Arbitration IDLE Time

These bits are used to indicate when to release the 68000 bus after a common memory space transfer has terminated.

NOTE

This mechanism is implemented only for PCMCIA common memory space accesses that are transferred into the 68000 bus.

Attribute space cycles are always made on a cycle steal basis.
I/O space cycles are mode to the 16550 registers only.

00 = Cycle steal mode. The 68000 bus is released after the cycle has been terminated. The arbitration logic will arbitrate for the 68000 bus when next cycle starts (CE active)
01 = Burst mode. The arbitration logic waits for 4 idle 68000 bus clocks on the PCMCIA bus before releasing the 68000 bus.
10 = Burst mode. The arbitration logic waits for 8 idle 68000 bus clocks on the PCMCIA bus before releasing the 68000 bus.
11 = Burst mode. The arbitration logic waits for 16 idle 68000 bus clocks on the PCMCIA bus before releasing the 68000 bus.

FAST—FAST burst mode

Setting this bit enables fast, burst mode PCMCIA common memory accesses without the WAIT signal (except for the first access). When the PCMCIA controller is programmed to burst mode by setting the ArbIDL1-0 bits to non-zero, the burst can be done in two different modes. The first cycle with arbitration is identical in both modes.

0 = Normal mode

When the PCMCIA controller is granted control of the 68000 bus and the PC initiates a common memory space cycle, the PCMCIA controller asserts the $\overline{\text{WAIT}}$ signal on the PCMCIA line. It then generates a 68000 cycle. When the cycle is terminated with $\overline{\text{DTACK}}$, data is transferred to the PCMCIA data lines and the $\overline{\text{WAIT}}$ signal is negated.

1 = Fast mode

This mode is used to minimize the usage of the $\overline{\text{WAIT}}$ signal on the PCMCIA interface, allowing fast direct access transfers to be performed. In this mode, when the PCMCIA controller is granted control of the 68000 bus and the PC initiates a common memory space cycle, the PCMCIA controller will not assert the $\overline{\text{WAIT}}$ signal on the PCMCIA line. The cycle on the 68000 bus will use the memory interface signals $\overline{\text{WE}}$ and $\overline{\text{OE}}$, but not the $\overline{\text{DS}}$. The cycle will be terminated when the PCMCIA cycle is terminated. The PCMCIA controller timing ignores $\overline{\text{DTACK}}$ and $\overline{\text{BERR}}$ so if 68000 bus cycles are not complete when the PCMCIA bus cycle is terminated garbage data will result (68000 Bus Errors can be reported to the host by setting the BERRIE bit in the PCMR). Fast mode accesses are only allowed for external memory space accesses. Accesses to internal memory space will result in erratic operation.

INTRL—INTeRrupt Level

The PCMCIA control logic will generate an interrupt to the 68000 core when the host writes one of the configuration registers if its interrupt mask bit is set in the PCMCIA Configuration Registers Write Mask Register - PCRWMR. The interrupt level generated as a result of one of the accesses can be programmed to the following 68000 interrupt levels:

00 = No interrupt is generated.
01 = The interrupt is generated on level 1.
10 = The interrupt is generated on level 6.
11 = The interrupt is generated on level 7.

NOTE

The interrupt will be generated as level triggered. The user should program the GIMR register to generate the interrupt vector.

BCLROE—Bus CLeaR Output Enable

When this bit is set the PCMCIA controller will assert the BCLR signal when requesting the bus and it is not granted. The IDMA, when programed to work under software control, will release the bus at the end of its current transfer.

BCLRIE—Bus CLeaR Input Enable

When this bit is set, the PCMCIA controller will release the 68000 bus when BCLR is asserted and the current cycle is terminated, regardless of the ArbIDL value. BCLR can be asserted for the following reasons:

1. The SDMA requests the 68000 bus.
2. An external master requests the 68000 bus.
3. A 68000 interrupt is pending and the BCLM bit is set in the SCR.

BERRIE—Bus ERRor Interrupt Enable

When this bit is set, the PCMCIA control logic will generate an interrupt to the PC when a PCMCIA bus cycle results in a 68000 bus error.

SWAP—SWAPPER ENABLE

This bit, when set, enables the swapping function for direct read and write PCMCIA accesses. The PCMCIA logic swaps the high and low bytes of 16 bit words when doing direct read or write accesses. Both the PCMCIA interface and the 68000 bus must be set to 16-bit wide for the swapper to be enabled.

TEn—Timer Enable

The PCMCIA direct access mechanism has a bus monitor (hardware watchdog) timer to prevent excessively long cycles, or hung state conditions on the PCMCIA interface. The timer is activated when a PCMCIA interface cycle begins. The WAIT signal is negated and a maskable interrupt is generated when the timer expires before the cycle is terminated.

0 = The PCMCIA interface bus monitor timer is disabled.
1 = The PCMCIA interface bus monitor timer is enabled.

TIEn—PCMCIA Watchdog Timer Interrupt Enable

This bit, when set, enables interrupt generation to the PC from the PCMCIA bus monitor timer. A PTIE interrupt is generated to the PC when this bit is set and the PCMCIA bus monitor timer expires.

TPres—PCMCIA Watchdog Timer Prescaler

The PCMCIA watchdog timer uses this prescaler value. The PCMCIA watchdog timer uses the prescaler output clock as an input to its 5-bit down counter.

00 = Divide by one.
01 = Divide by 4.
10 = Divide by 8.
11 = Divide by 16.

TValue—Timer Value

The PCMCIA watchdog timer is a 5-bit down counter with the programmable TValue preset. The counter has a programmable prescaler. The counter will count the $\overline{\text{WAIT}}$ length and, if expired, the PCMCIA cycle will terminated and the PTIE interrupt to the PC will be generated (if enabled).

ABUF—Address Buffer Control

This bit, when set, enables the address buffer control output function. The PCMCIA controller will not drive the high address lines (A12-A23) as programmed in the base registers but instead will drive an address buffer control line with the appropriate timing. This mechanism is useful when the paging is not sufficient for the system design requirements. External address buffers must be implemented on the card.

16Bit—PCMCIA Interface is 16-bit

0 = The PCMCIA interface is 8 data bits wide. The D8-D15 and CE2 pins can be muxed for their alternative functions.
1 = The PCMCIA interface is 16 data bits wide.

IPins—ISDN Pins

0 = The ISDN interface is not implemented. The $\overline{\text{CD1}}$/L1SY1/PC_A11 pin will be set for PC_A11 with the PCMCIA interface enabled. The $\overline{\text{CTS}}$/L1GR/PC_A10 will be set for PC_A10. The $\overline{\text{RTS}}$/L1RQ/GCIDCL/$\overline{\text{PC\_WAIT}}$ pin will be set for $\overline{\text{PC\_WAIT}}$.
1 = The ISDN interface is implemented.The TOUT1,TIN2, and TOUT2 pins are instead set for PC_A11, PC_WAIT, PC_A10 pins respectively.

AINC1,2—Auto INCrement

When set, the corresponding common memory base address register will auto-increment when the last entry in the page (A0-A10 = 3FFx) is accessed. This mechanism is useful during burst mode accesses because it eliminates the need to reprogram the base address register when crossing a page boundary.

RISWDIS—Ring Indicate Software Disable

0 = The RIEvt bit in the IOEIR register will be set when $\overline{\text{RI}}$ is asserted.
1 = The RIEvt bit in the IOEIR is not set when $\overline{\text{RI}}$ is asserted. This mechanism is useful to allow the 68000 core to either integrate or debounce the $\overline{\text{RI}}$ signal by software before updating the RIEvt bit (the $\overline{\text{RI}}$(PB9) pin can be read by the 68000 core through the port B data register). Refer to Figure 8-6. Ring Indicate (PB9) Connection Options.

8.4.3 PCMCIA Configuration Registers Write Event Register - PCRWER

PCRWER

PCMCIA Address: Not accessible
68000 Address: BAR + $8C4

7	6	5	4	3	2	1	0
RES3	RES2	ES1	IOIEIR	SCR	PRR	CCSR	COR
RESET							
0	0	0	0	0	0	0	0

Read/Write

The PCMCIA configuration registers write event register is an 8-bit register that reports PC writes to one of the PCMCIA configuration registers. The appropriate event bit is set depending on the location of host write, regardless of the corresponding mask bit in the PCRWMR. If the corresponding mask bit is set and the INTRL0-1 bits in the PCMR register are non-zero, an interrupt will be generated. In order to generate an interrupt, the INTRL0-1 bits in the PCMR register must be set to a non-zero value. The PCRWER is a memory mapped register that may be read at any time. A bit can be reset by writing a one and is left unchanged by writing a zero. More than one bit may be reset at a time. All bits are cleared by reset.

8

CCR—Card Configuration Register

This bit is set when the PC writes to the card configuration option register. The 68000 core may monitor writes to this register to detect SRESET and changes to the configuration index field (to determine if the host has enabled or disable the I/O card). This bit can be cleared by writing a one to it.

CCSR—Card Configuration and Status Register

This bit is set when the PC writes to the card configuration and status register. The 68000 core may monitor writes to this register to detect if the PC has requested to place the card in the power-down state by setting the PwrDwn bit. This bit can be cleared by writing a one to it.

PRR—Pin Replacement Register

This bit is set when the PC writes to the pin replacement register. This bit can be cleared by writing a one to it.

SCR—Socket and Copy Register

This bit is set when the PC writes to the socket and copy register. This bit can be cleared by writing a one to it.

IOEIR—IO Event Indication Register

This bit is set when the PC writes to the IO event indication register. This bit can be cleared by writing a one to it.

Res1—Reserved1 Register

This bit is set when the PC writes to the reserved1 register. This bit can be cleared by writing a one to it.

Res2—Reserved2 Register

This bit is set when the PC writes to the reserved2 register. This bit can be cleared by writing a one to it.

Res3—Reserved3 Register

This bit is set when the PC writes to the reserved3 register. This bit can be cleared by writing a one to it.

8.4.4 PCMCIA Configuration Registers Write Mask Register - PCRWMR

The PCRWMR is 8-bit memory-mapped, read-write register that has the same bit format as the PCRWER. If a bit in the PCRWMR is a one, the corresponding interrupt in the PCRWER will be enabled. If the bit is zero, the corresponding interrupt in the PCRWMR will be disabled. PCRWMR is cleared at reset.

8.4.5 PCMCIA Access Wake-Up Event Register - PCAWER

PCAWER

PCMCIA Address: Not accessible
68000 Address: BAR + $8C5

7	6	5	4	3	2	1	0
–	–	–	MRW	CMS2	CMS1	ATS	IOS
RESET							
0	0	0	0	0	0	0	0

Read/Write

The 68000 core may program the IMP into stand-by mode when there is no activity on the card for long periods of time. PC accesses to the card will wake-up the IMP.

The PCAWER is an 8-bit register used to report which type of PC write access was made to the card to wake the IMP up from STAND-BY mode. The 68000 core will receive an interrupt if:

1. It is waked up by a PCMCIA access, and

2. The INTRL bits in the PCMR are non-zero (PCMCIA interrupts enabled), and

3. The corresponding mask bit for the access type is set in the PCAWMR.

NOTE

Read accesses by the PC to one of the PCMCIA configuration registers will **not** wake up the IMP.

The appropriate event bit will be set, regardless of the corresponding mask bit in the PCRWMR. If the corresponding mask bit is set, an interrupt will be generated. The PCAWER is a memory mapped register that may be read at any time. A bit is reset by writing a one and is left unchanged by writing a zero. More than one bit may be reset at a time, and the register is cleared by reset.

IOS—IO Space Access

This bit is set when the PC accesses the 16550 emulation module (an I/O space access is made). When the IMP is in stand-by mode, the IMP will resume its operation. An interrupt may be generated if the corresponding mask bit is set. This bit is cleared by writing one.

ATS—ATtribute Space Access

This bit is set when the PC accesses the CIS memory. When the IMP is in stand-by mode, the IMP will resume its operation. An interrupt may be generated if the corresponding mask bit is set. This bit is cleared by writing one.

NOTE

PC writes to one of the PCMCIA configuration registers while the IMP is in stand-by mode will also resume operation. The 68000 will be able to detect it by the 8.4.5 PCMCIA Access Wake-Up Event Register - PCAWER.

CMSI1—Common Memory Space Access

This bit is set when the PC accessing common memory space with A25=1 (CMBAR1 is selected). When the IMP is in stand-by mode, the IMP will resume its operation. An interrupt may be generated if the corresponding mask bit is set. This bit is cleared by writing one.

CMSI2—Common Memory Space Access

This bit is set when the PC accessing common memory space with A25=0 (CMBAR2 is selected). When the IMP is in stand-by mode, the IMP will resume its operation and an interrupt may be generated if the corresponding mask bit is set. This bit is cleared by writing one.

MRW—Mode Registers Write.

This bit is set when the PC writes to one of the PCMCIA mode or base registers in the attribute space. When the IMP is in stand-by mode, the IMP will resume its operation. An interrupt may be generated if the corresponding mask bit is set. This bit is cleared by writing one.

8.4.6 PCMCIA Access Wake-up Mask Register - PCAWMR

The PCAWMR is an 8-bit memory-mapped, read-write register that has the same bit format as the PCAWER. If a bit in the PCAWMR is a one, the corresponding interrupt in the PCAWER will be enabled. If the bit is zero, the corresponding interrupt in the PCRWMR will be disabled. PCAWMR is cleared at reset.

8.4.7 PCMCIA Host (PC) Event Register - PCHER

PCHER

PMCIA Address:A25 = 1; $2A
68000 Address: BAR + $8C8

7	6	5	4	3	2	1	0
–	–	–	–	–	SW	PTIE	BERR
RESET							
0	0	0	0	0	0	0	0

Read/Write

BERR—Bus ERRor

This bit is set when a cycle is terminated by bus error. An interrupt to the PC will be generated if the BERRIE bit is set. This bit is cleared by writing one.

PTIE—PCMCIA Timer Expired

This bit is set when a cycle is terminated by a PCMCIA watchdog timer expiration. An interrupt to the PC will be generated if the TIEn bit is set. This bit is cleared by writing a one.

SW—68000 Software Interrupt

The 68000 S/W may set this bit to generate an interrupt to the PC. This bit is cleared by the PC writing a one.

8

8.4.8 CIS Base Address Register - CISBAR

CISBAR

PCMCIA Address: A25 = 1; $32
68000 Address: BAR + $8CC

15	14	13	12	11	10	9	8
BA23	BA22	BA21	BA20	BA19	BA18	BA17	BA16
RESET				0			
0	0	0	0		0	0	0

PMCIA Address: A25 = 1; $30

7	6	5	4	3	2	1	0
BA15	BA14	BA13	BA12	FC2	FC1	FC0	V
RESET							
0	0	0	0	0	0	0	0

Read/Write

This register is used to locate PCMCIA CIS accesses to the 68000 bus. The CIS must be located in memories (RAM, EEPROM etc.) on the 68000 bus. The 68000 core should program the starting address of the CIS memory structure into this register.

NOTE

Since the PCMCIA attribute space accesses are only 8-bits and A0 should be always 0, the CIS memory should be on the 68000 even addresses only.

NOTE

This register is in the PCMCIA attribute space and this 24-bit register address will be 3 bytes on **even** addresses only.

V—Valid Bit

This bit indicates that the contents of the base register is valid. The PCMCIA attribute space cycle ($\overline{REG}$ is asserted and A25=0) will not be transferred to the 68000 bus until the V-bit is set.

0 = This CIS base address register is invalid.
1 = This CIS base address register is valid.

BA12-23—Base address

The base address field should contain the upper address bits of the CIS data structure address on the 68000 bus. The lower address bits will be taken from the PCMCIA address lines.

FC0-2—Function Code

The function code field should contain the function code of the CIS data structure address on the 68000 bus.

8.4.9 Common Memory Space Base Address Register - CMBAR1,2

CMBAR1, CIMBAR2

PCMCIA Address: A25 = 1; $3A, $42
68000 Address: BAR + $8D0, $8D4

15	14	13	12	11	10	9	8
BA23	BA22	BA21	BA20	BA19	BA18	BA17	BA16
RESET							
0	0	0	0	0	0	0	0

PCMCIA Address: A25 = 1; $38, $40

7	6	5	4	3	2	1	0
BA15	BA14	BA13	BA12	FC2	FC1	FC0	V
RESET							
0	0	0	0	0	0	0	0

Read/Write

There are two identical registers to map common memory accesses into the 68000 bus. This mechanism will enable the card designer to use memories (SRAM, DRAM etc.) or peripherals on the 68000 bus and access them by the host. The 68000 core or the PC should program the starting 68000 address of the memory or peripheral into these registers. A25 will distinguish between the two base registers. When A25=1 CMBAR1 will be selected and when A25=0 CMBAR2.

NOTE

Since the PCMCIA attribute space accesses are only 8-bits, this 16-bit register address will be 2 bytes on even addresses only.

V—Valid Bit

This bit indicates that the contents of the base register is valid. The PCMCIA common memory space cycle will not be transferred into the 68000 bus until the V-bit is set.

0 = This register base address is invalid.
1 = This register base address is valid.

BA12-23—Base Address

The base address field should contain the upper address bits of the I/O memory address on the 68000 bus. The lower address bits will be taken from the PCMCIA address lines.

NOTE

The 2Kbyte block defined by this base address should not contain any internal addresses (internal dual-port RAM, internal registers, etc.) if it is desired to use the fast mode accesses. Fast mode accesses (Fast Mode=1) to internal memory locations will result in erratic operation.

FC0-2—Function Code

The function code field should contain the function code of the common memory address on the 68000 bus.

8.4.10 Card Configuration Registers

8.4.10.1 Configuration Option Register - COR

COR

PMCIA Address: A25 = 1; $0
68000 Address: BAR + $870

7	6	5	4	3	2	1	0
SRESET	LevIREQ	Configuration Index					
0	0	0	0	0	0	0	0

Read/Write

The 68000 core will receive an interrupt when this register is written by the PC. See 8.4.5 PCMCIA Access Wake-Up Event Register - PCAWER.

SRESET—Reset Card

When this bit is set to one, the IMP is put in a hardware reset state. The reset pin will be asserted, and the 68000 core and the rest of the IMP are reset.

LevIREQ—Level Mode Interrupts

The MC68356 PCMCIA controller supports the interrupt level mode only. (Regardless of this bit value)

NOTE

The pulse mode may be supported by the 68000 software.

Configuration Index

This field is written with the index number of the entry in the card's configuration table which corresponds to the configuration which the system chooses for the card. When the configuration index is 0, the card's I/O is disabled and will not respond to any I/O cycles, and will use the memory only interface.

8.4.10.2 Card Configuration and Status Register - CCSR

CCSR

PMCIA Address: A25 = 1; $2
68000 Address: BAR + $871

7	6	5	4	3	2	1	0
Changed	SigChg	IOis8	RingEn	Audio	PwrDwn	Intr	Res(0)

Read/Write

The 68000 core will receive an interrupt when this register is written by the PC. See 8.4.5 PCMCIA Access Wake-Up Event Register - PCAWER.

8

Changed

This bit indicates that one or more of the pin replacement register bits CBVD1, CBVD2, CRdy$\overline{BSY}$, CWProt is set to one or when one or more of the I/O event indication register event bits are set and its corresponding enable bit is also set. When this bit is set, the $\overline{STSCHG}$(BVD1) pin is held low if the SigChg bit is 1 and the card is configured as I/O.

SigChg

1 = When the card is configured for I/O interface, the changed bit controls the $\overline{STSCHG}$ signal and is called the changed status signal.
0 = The BVD1 ($\overline{STSCHG}$) line will be held high while the card is configured as I/O.

RingEn

0 = The state of the $\overline{STSCHG}$ is controlled by the SigChg bit as described above.
1 = The $\overline{STSCHG}$ signal is used to indicate the state of the ring indicate signal. $\overline{STSCHG}$ is asserted (low) whenever ringing is present on the RI (PB9) pin. $\overline{STSCHG}$ is negated (high) when there is no ringing.

IOis8

1 = The host can provide I/O cycles only with 8-bit D0-D7 data path. Accesses to 16-bit registers will occur as two byte accesses.
0 = The host can provide I/O cycles with 16-bit data path.

Audio

This bit is set to one to enable audio information on BVD2 pin while the card is configured.

PwrDwn

This bit is set to one to request the card enter a powerdown state. The 68000 core will receive an interrupt (if enabled) when this register is written (see 8.4.4 PCMCIA Configuration Registers Write Mask Register - PCRWMR) and should check the state of this bit to determine if the PC has placed the card into power down mode. If it has, the 68000

should place the card into power down mode. When this bit is reset while the IMP is in any of the three STOP modes, the 68000 core will be waked up (the PC can access the configuration registers while in powerdown mode).

Intr

This bit represents the state of the interrupt request. This value is available whether or not interrupts have been configured. This signal should remain true until the condition which caused the interrupt request has been serviced. See 8.2.2.3 PCMCIA Host Interrupts for more information on the functionality of this bit.

8.4.10.3 Pin Replacement Register Organization - PRR

PRR

PMCIA Address: A25 = 1; $4
68000 Address: BAR +$872

7	6	5	4	3	2	1	0
CBVD1	CBVD2	CRdy/Bsy	CWProt	RBVD1	RBVD2	RRdy/Bsy	RWProt

Read/Write

The 68000 core will receive an interrupt when this register is written by the PC. See 8.4.5 PCMCIA Access Wake-Up Event Register - PCAWER.

CBVD1,CBVD2

These bits are set to one when the corresponding bits change state. These bits may also be written by the host.

CRdy/Bsy

This bit is set to one when the corresponding bit changes state. These bits may also be written by the host.

CWProt

This bit is set to one when the corresponding bit changes state. These bits may also be written by the host.

RBVD1, RBVD2

When read these bits represent the internal state of the corresponding signals. When this bit is written as one, the corresponding changed bit is also written.Since the MC68356 does not support this pin directly, external circuitry should be used to implement this pin.

NOTE

The 68000 should write the state of this signal into this location. The corresponding changed bit will be updated according to the written data.

RRdy/Bsy

When read, this bit represents the internal state of the corresponding signal. When this bit is written as one, the corresponding changed bit is also written. See 8.2.2.4 The Ready Busy Signal (Rdy/Bsy) for more information on the functionality of this bit.

NOTE

The 68000 core should write the state of this signal into this location. The corresponding changed bit will be updated according to the written data.The 68000 core should set this bit to one after initializing the PCMCIA controller to signal ready (high) to the host. When set to zero, the PCMCIA controller will signal busy. After reset, the PCMCIA controller will set the RRdy/$\overline{\text{Bsy}}$ pin low and clear this bit.

NOTE

In I/O mode the RDY/$\overline{\text{BSY}}$ pin becomes the $\overline{\text{IREQ}}$ pin.

RWProt

When read, this bit represents the internal state of the corresponding external signal. When this bit is written as one, the corresponding changed bit is also written. Since the MC68356 does not support this pin directly, external circuitry should be used to monitor the state of this signal.

NOTE

The 68000 should write the state of this signal into this location. The corresponding changed bit will be updated according to the written data.

8.4.10.4 Socket and Copy Register - SCR

SCR

PMCIA Address: A25 = 1; $6
68000 Address: BAR + 873

7	6	5	4	3	2	1	0
Res	Copy Number			Socket Number			

Read/Write

This register can be R/$\overline{\text{W}}$ accessed by the PC and the 68000 core. The 68000 core will receive an interrupt when this register is written by the PC. See 8.4.5 PCMCIA Access Wake-Up Event Register - PCAWER.

NOTE

This is an optional register which the card may use to distinguish between similar cards installed in a system.

8.4.10.5 I/O Event Indication Register - IOEIR

IOEIR

PMCIA Address: A25 = 1; $8
68000 Address: BAR + $874

7	6	5	4	3	2	1	0
ResEvt3	ResEvt2	Pievt	RIEvt	ResEnab3	ResEnab2	PIEnab	RIEnab

Read/Write

The 68000 core will receive an interrupt when this register is written by the PC. See 8.4.5 PCMCIA Access Wake-Up Event Register - PCAWER.

This is an optional register that contains information about the changes in the card status. Its bit assignments are defined below. This register may be read or written. The upper four bits are latched to a one when the corresponding IO event occurs on the PC card. When one of these upper four bits is latched and the corresponding enable bit in the lower nibble has also previously been set, the changed bit in the card configuration and status register (CCSR) is set to a one, and the $\overline{\text{STSCHG}}$ pin will be driven low (if enabled by the SigChg bit in the CCSR).

The host can clear any one of the upper four bits by writing a one to that bit. Writing a zero to these bits will have no effect. All bits of this register are cleared by an IMP reset.

ResEvt3

Reserved for future expansion/definition. Set to zero.

ResEvt3

Reserved for future expansion/definition. Set to zero.

PIEvt

This bit is latched to a one by the card (the 68000 core can set this bit) after the receipt of a valid incoming packet over a modem channel. When this bit is set to a one and the PI-Enab bit is set to a one, the changed bit in the card configuration and status register will also be set to a one. If the SigChg bit in the card configuration and status register has also been set by the host, then the $\overline{\text{STSCHG}}$ pin will be driven low. Writing one to this bit will clear it to zero. Writing a zero to this bit has no effect.

RIEvt

This bit is latched to a one by the MC68356 on the start of each cycle of the ring frequency on the phone line. When this bit is set to a one, and the RIEnab bit will be set to a one the changed bit in the card configuration and status register will also be set to a one. If the SigChg bit in the card configuration and status register (CCSR) has also been set by the PC, then the $\overline{\text{STSCHG}}$ pin will be driven low. Writing one to this bit will clear it to zero. Writing a zero to this bit has no effect.

NOTE

If it is desired to have the 68000 core debounce or otherwise process the RI signal before passing it to the host, the RISDIS

bit can be set in the PCMR register which will disconnect the direct connection to the RIEvt bit (See Figure 8-6).

ResEnab3

Reserved for future expansion/definition. Set to zero.

ResEnab2

Reserved for future expansion/definition. Set to zero.

PIEnab

Setting this bit to a one enables setting the changed bit in the card configuration and status register when the PIEvt bit is set. When this bit is cleared to a zero, this feature is disabled. The state of PIEvt bit is not affected by this bit.

RIEnab

Setting this bit to a one enables setting the changed bit in the card configuration and status register when the RIEvt bit is set. When this bit is cleared to a zero, this feature is disabled. The state of RIEvt bit is not affected by this bit.

8.4.10.6 Reserved Registers

The 68356 implements additional 3 card configuration registers. Those 8-bits registers can be read/written by both the 68000 core and the PC. They are reserved for future use.

SECTION 9
DSP MEMORY MODULES AND OPERATING MODES

9.1 MEMORY MODULES AND OPERATING MODES

The memory of the DSP56002 can be partitioned in several ways to provide high-speed parallel operation and additional off-chip memory expansion. Program and data memory are separate, and the data memory is, in turn, divided into two separate memory spaces, X and Y. Both the program and data memories can be expanded off-chip. There are also two on-chip data read-only memories (ROMs) that can overlay a portion of the X and Y data memories, and a bootstrap ROM that can overlay part of the program random-access memory (RAM). The data memories are divided into two independent spaces to work with the two address arithmetic logic units (ALUs) to feed two operands simultaneously to the data ALU.

The DSP operating modes determine the memory maps for program and data memories and the start-up procedure when the DSP leaves the reset state. This section describes the DSP56002 Operating Mode Register (OMR), its operating modes and their associated memory maps, and discusses how to set and reset operating modes.

This section also includes details of the interrupt vectors and priorities and describes the effect of a hardware reset on the PLL multiplication factor.

9.2 DSP56002 DATA AND PROGRAM MEMORY

The DSP on the MC68356 has 5.25kwords of program RAM, 64 words of bootstrap ROM, 3K words of X data RAM, 256 words of X data ROM, 2.5K words of Y data RAM, 256 words of Y Data ROM. The memory maps are shown in Figure 9-1.

9.2.1 Program Memory

The DSP56002 has 5.25K words of program RAM and 64 words of factory-programmed bootstrap ROM.

The bootstrap ROM is programmed to perform the bootstrap operation from the memory expansion port (port A), from the host interface, or from the SCI. It provides a convenient, low cost method of loading the program RAM with a user program after power-on reset. The bootstrap ROM activity is controlled by the MA, MB, and MC bits in the OMR (see 9.3 DSP56002 Operating Mode Register (OMR) for a complete explanation of the OMR and the DSP56002's operating modes and memory maps).

Addresses are received from the program control logic (usually the program counter) over the PAB. Program memory may be written using the program memory (MOVEM) instruc-

tions. The interrupt vectors are located in the bottom 128 locations ($0000-$007F) of program memory. Program memory may be expanded to 64K off-chip.

9.2.2 X Data Memory

The on-chip X data RAM is a 24-bit-wide, static internal memory occupying the lowest 3072 locations (0-3071) in X memory space. The on-chip X data ROM occupies locations 3072–3327 in the X data memory space and is controlled by the DE bit in the OMR. (See the explanation of the DE bit in 9.3.2 Data ROM Enable (Bit 2). Also, see Figure 9-1.)The on-chip peripheral registers occupy the top 64 locations of the X data memory ($FFC0–$FFFF). The 16-bit addresses are received from the XAB, and 24-bit data transfers to the data ALU occur on the XDB. The X memory may be expanded to 64K off-chip.

9.2.3 Y Data Memory

The on-chip Y data RAM is a 24-bit-wide internal static memory occupying the lowest 2560 locations (0–2559) in the Y memory space. The on-chip Y data ROM occupies locations 2559–2815 in Y data memory space and is controlled by the DE and YD bits in the OMR. (See the explanations of the DE and YD bits in 9.3.2 Data ROM Enable (Bit 2) and 9.3.3 Internal Y Memory Disable Bit (Bit 3), respectively. Also, see Figure 9-1). The 16-bit addresses are received from the YAB, and 24-bit data transfers to the data ALU occur on the YDB. Y memory may be expanded to 64K off-chip.

NOTE

The off-chip peripheral registers should be mapped into the top 64 locations ($FFC0–$FFFF) to take advantage of the move peripheral data (MOVEP) instruction.

9.3 DSP56002 OPERATING MODE REGISTER (OMR)

Operating modes determine the memory maps for program and data memories, and the start-up procedure when the DSP leaves the reset state. The processor samples the MODA, MODB, and MODC signals as it leaves the reset state, establishes the initial operating mode, and writes the operating mode information to the operating mode register. The MODA, MODB, and MODC signals can either be sampled from the external MODA/$\overline{\text{IRQA}}$, MODB/$\overline{\text{IRQB}}$, and MODC/$\overline{\text{NMI}}$ pins, or the MODA/$\overline{\text{IRQA}}$, MODB/$\overline{\text{IRQB}}$, and MODC/$\overline{\text{NMI}}$ bits in the DISC register on the IMP side if the SELA, SELB and SELC bits in the DISC have been set. For more information on the DISC register see Section 6 System Integration Block (SIB). When the processor leaves the reset state, the MODA and MODB pins become general-purpose interrupt pins, $\overline{\text{IRQA}}$ and $\overline{\text{IRQB}}$, respectively, and the MODC pin becomes the nonmaskable interrupt pin $\overline{\text{NMI}}$.

The OMR is a 24-bit register (only six bits are defined) that controls the current operating mode of the processor. It is located in the DSP56002's Program Control Unit (described in Section 5 of DSP56KFAMUM/AD the *DSP56000 Digital Signal Processor Family Manual*). The OMR bits are only affected by processor reset and by the ANDI, ORI, MOVEC, BSET, BCLR, and BCHG instructions, which directly reference the OMR. The OMR format for the DSP56002 is shown in Figure 9-2.

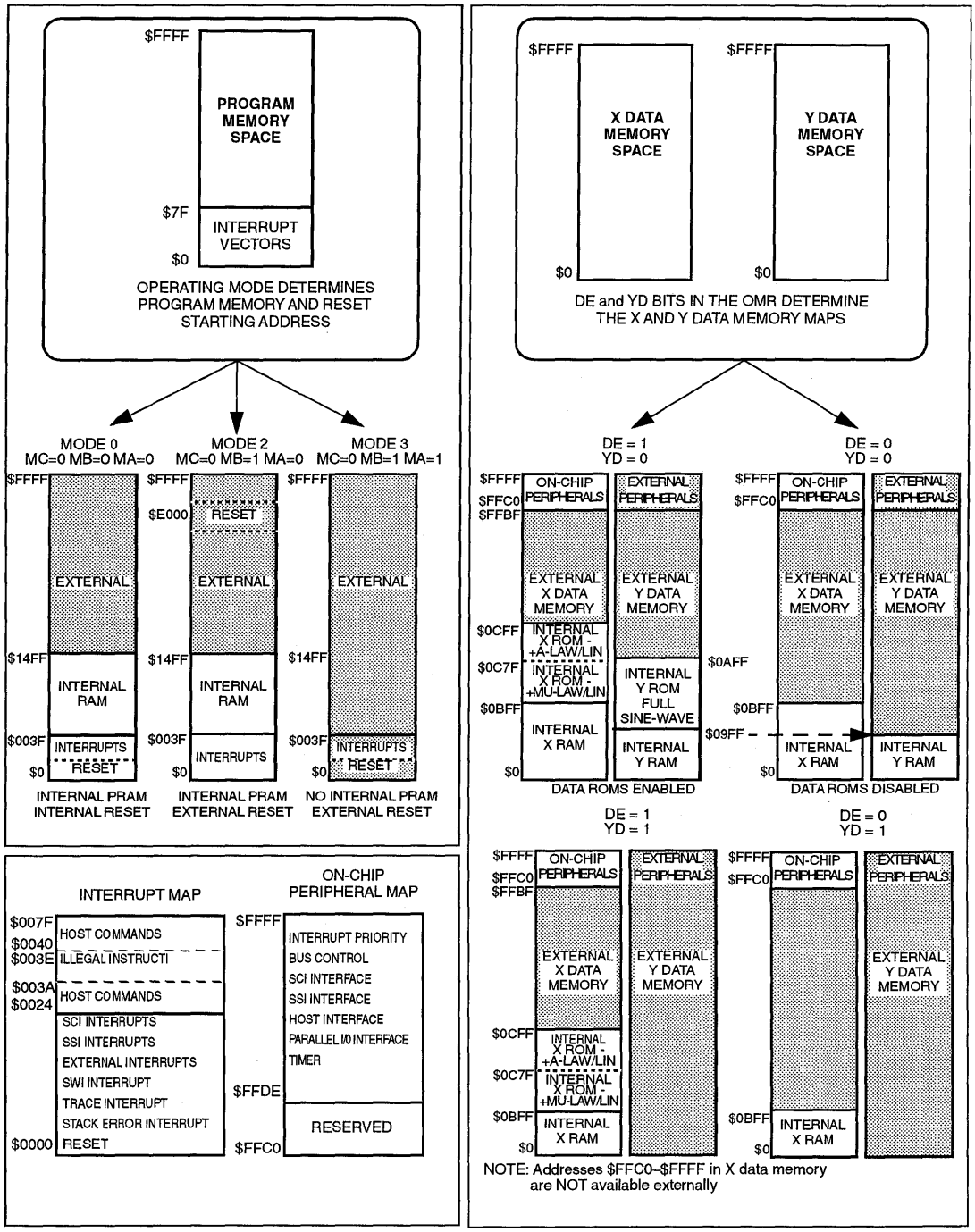
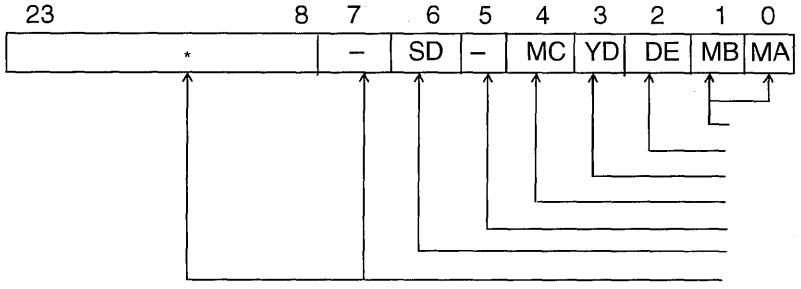


Figure 9-1. MC68356 DSP56002 Memory Maps



Bits 5 and 7 to 23 are reserved, read as zero, and should be written with zero for future compatibility.

Figure 9-2. OMR Format

9.3.1 Chip Operating Mode (Bits 0 and 1)

The chip operating mode bits, MB and MA, together with MC, define the program memory maps and the operating mode of the DSP56002. On processor reset, MB and MA are loaded from the external mode select pins, MODB and MODA, respectively. After the DSP leaves the reset state, MB and MA can be changed under software control.

9

9.3.2 Data ROM Enable (Bit 2)

The DE bit enables the two, on-chip, 256X24 data ROMs located between addresses $0C00–$0CFF in the X and Y memory spaces. When DE is cleared, the $0A00–$0AFF address space is part of the external X and Y data spaces, and the on-chip data ROMs are disabled. Hardware reset clears the DE bit.

9.3.3 Internal Y Memory Disable Bit (Bit 3)

Bit 3 is defined as Internal Y Memory Disable (YD). When set, all Y Data Memory addresses are considered to be external, disabling access to internal Y Data Memory. When cleared, internal Y Data Memory may be accessed according to the state of the DE control bit. The content of the internal Y Data Memory is not affected by the state of the YD bit. The YD bit is cleared during hardware reset.

Figure 9-1 shows a graphic representation of the DE and YD bit effects on the X and Y data memory maps. Table 9-1 also compares the DE and YD effects on the memory maps.

Table 9-1. Memory Mode Bits

DE	YD	Data Memory
0	0	Internal ROMs Disabled and their addresses are part of External Memory
0	1	Internal X Data ROM is Disabled and is part of External Memory. Internal Y Data RAM and ROM are Disabled and are part of External Memory
1	0	X and Y Data ROMs Enabled
1	1	Internal Y Data RAM and ROM are Disabled and are part of External Memory. Internal X Data ROM Enabled.

9.3.4 Chip Operating Mode (Bit 4)

The MC bit, together with bits MA and MB, define the program memory map and the operating mode of the chip. Upon reset, the processor loads this bit from the MODC external mode select pin. After the DSP leaves the reset state, MC can be changed under software control.

9.3.5 Reserved (Bit 5)

This bit is reserved for future expansion and will be read as zero during read operations.

9.3.6 Stop Delay (Bit 6)

The SD bit determines the length of the clock stabilization delay that occurs when the processor leaves the stop processing state. If the stop delay bit is zero when the chip leaves the stop state, a 64K clock cycle delay is selected before continuing the stop instruction cycle. However, if the stop delay bit is one, the delay before continuing the instruction cycle is long enough to allow a clock stabilization period for the internal clock to begin oscillating and to stabilize. (See Section 14 Electrical Characteristics) for the actual timing values.) When a stable external clock is used, the shorter delay allows faster start-up of the DSP.

9.3.7 Reserved OMR Bits (Bits 7–23)

These bits are reserved for future expansion and will be read as zero during read operations.

9.4 DSP56002 OPERATING MODES

The user can set the chip operating mode through hardware by pulling high the MODC, MODB, and MODA pins appropriately, and then assert the RESET pin. When the DSP leaves the reset state, it samples the mode signals and writes to the OMR to set the initial operating mode.

Chip operating modes can also be changed using software to write the operating mode bits (MC, MB, MA) in the OMR. Changing operating modes does not reset the DSP.

NOTE

The user should disable interrupts immediately before changing the OMR to prevent an interrupt from going to the wrong memory location. Also, one no-operation (NOP) instruction should be included after changing the OMR to allow for remapping to occur.

Table 9-2. DSP56002 Operating Mode Summary

Operating Mode	M C	M B	M A	Description
0	0	0	0	Single-Chip Mode - PRAM enabled, reset @ $0000
1	0	0	1	Bootstrap from EPROM, exit in Mode 0
2	0	1	0	Normal Expanded Mode - PRAM enabled, reset @ $E000
3	0	1	1	Development Mode - PRAM disabled, reset @ $0000
4	1	0	0	Reserved for bootstrap
5	1	0	1	Bootstrap from Host, exit in Mode 0
6	1	1	0	Bootstrap from SCI (external clock), exit in Mode 0
7	1	1	1	Reserved for bootstrap

9.4.1 Single Chip Mode (Mode 0)

In the single-chip mode, all internal program and data RAM memories are enabled (see Figure 9-1). A hardware reset causes the DSP to jump to internal program memory location $0000 and resume execution. The memory maps for mode 0 and mode 2 (see Figure 9-1) are identical. The difference between the two modes is that reset vectors to program memory location $0000 in mode 0 and vectors to location $E000 in mode 2.

9.4.2 Bootstrap From EPROM (Mode 1)

The bootstrap modes allow the DSP to load a program from an inexpensive byte-wide ROM into internal program memory during a power-on reset. On power-up, the wait-state generator adds 15 wait states to all external memory accesses so that slow memory can be used. The bootstrap program uses the bytes in three consecutive memory locations in the external ROM to build a single word in internal program memory.

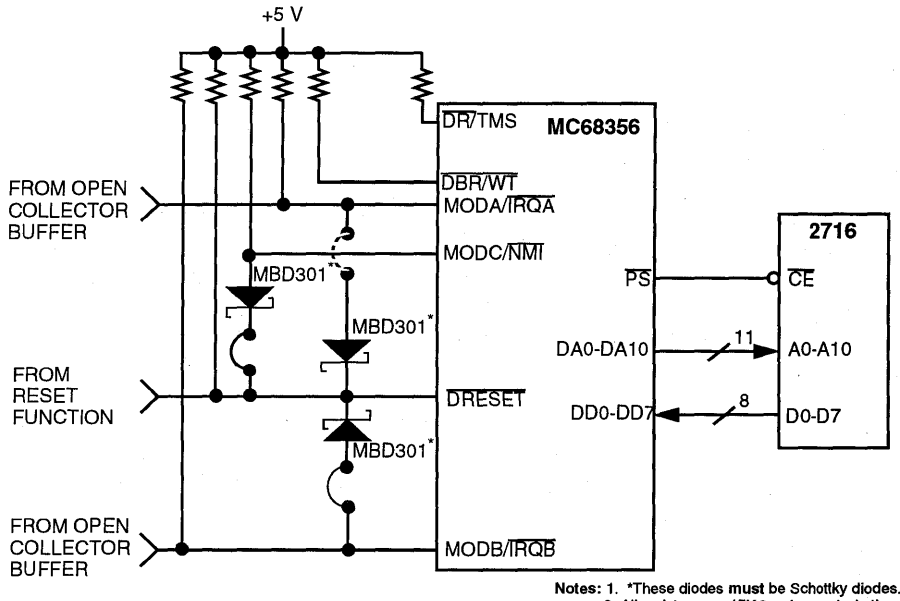
In the bootstrap mode, the chip enables the bootstrap ROM and executes the bootstrap program. (The bootstrap program code is shown in Appendix C DSP Bootstrap Program.) The bootstrap ROM contains the bootstrap firmware program that performs initial loading of the DSP program RAM. Written in DSP56002 assembly language, the program initializes the program RAM by loading from an external byte-wide EPROM starting at location P:$C000.

The EPROM is typically connected to the chip's address and data bus.The data contents of the EPROM must be organized as shown in Table 9-3:

After loading the internal memory, the DSP switches to the single-chip mode (Mode 0) and begins program execution at on-chip program memory location $0000.

If the user selects Mode 1 through hardware (MODA, MODB, MODC pins), the following actions occur once the processor comes out of the reset state.

1. The control logic maps the bootstrap ROM into the internal DSP program memory space starting at location $0000.
2. The control logic causes program reads to come from the bootstrap ROM (only address bits 5–0 are significant) and all writes go to the program RAM (all address bits are significant). This condition allows the bootstrap program to load the user program from $0000–$01FF.



Notes: 1. *These diodes must be Schottky diodes.
2. All resistors are 15KΩ unless noted otherwise.
3. When in RESET, IRQA, IRQB and NMI must be deasserted by external peripherals.

ADDRESS OF EXTERNAL BYTE-WIDE P MEMORY	CONTENTS LOADED TO INTERNAL PRAM AT:
P:$C000	P:$0000 LOW BYTE
P:$C001	P:$0000 MID BYTE
P:$C002	P:$0000 HIGH BYTE
.	.
.	.
.	.
P:$C5FD	P:$01FF LOW BYTE
P:$C5FE	P:$01FF MID BYTE
P:$C5FF	P:$01FF HIGH BYTE

Figure 9-3. Port A Bootstrap Circuit

Table 9-3. Organization of EPROM Data Contents

Address of External Byte-Wide Memory:	Contents Loaded to Internal Program RAM at:	
P:$C000	P:$0000	low byte
P:$C001	P:$0000	mid byte
P:$C002	P:$0000	high byte
•	•	
•	•	
•	•	
P:$C5FD	P:$01FF	low byte
P:$C5FE	P:$01FF	mid byte
P:$C5FF	P:$01FF	high byte

3. Program execution begins at location $0000 in the bootstrap ROM. The bootstrap ROM program loads program RAM from the external byte-wide EPROM starting at P:$C000.
4. The bootstrap ROM program ends the bootstrap operation and begins executing the user program. The processor enters Mode 0 by writing to the OMR. This action is timed to remove the bootstrap ROM from the program memory map and re-enable read/write access to the program RAM. The change to Mode 0 is timed to allow the bootstrap program to execute a single-cycle instruction (clear status register), then a JMP #<00, and begin execution of the user program at location $0000.

The user can also get into the bootstrap mode (Mode 1) through software by writing zero to MC and MB, and one to MA in the OMR. This selection initiates a timed operation to map the bootstrap ROM into the program address space (after a delay to allow execution of a single-cycle instruction), and then a JMP #<00 to begin the bootstrap process described previously in steps 1 through 4. This technique allows the user to reboot the system (with a different program, if desired).

The code to enter the bootstrap mode is as follows:

```
MOVEP     #0,X:$FFFF    ;Disable interrupts.
MOVEC     #1,OMR        ;The bootstrap ROM is mapped
                        ;into the lowest 64 locations
                        ;in program memory.
NOP                     ;Allow one cycle delay for the
                        ;remapping.
JMP       <$0           ;Begin bootstrap.
```

The code disables interrupts before executing the bootstrap code. Otherwise, an interrupt could cause the DSP to execute the bootstrap code out of sequence because the bootstrap program overlays the interrupt vectors.

9.4.3 Normal Expanded Mode (Mode 2)

In this mode, the internal program RAM is enabled and the hardware reset vectors to location $E000. (The memory maps for Mode 0 and Mode 2 are identical. The difference for Mode 0 is that, after reset, the instruction at location $E000 is executed instead of the instruction at $0000 — see Figure 9-1 and Table 9-2.).

9.4.4 Development Mode (Mode 3)

In this mode, the internal program RAM is disabled and the hardware reset vectors to location $0000. All references to program memory space are directed to external program memory. The reset vector points to location $0000. The memory map for this mode is shown in Figure 9-1 and Table 9-2.

9.4.5 Reserved (Mode 4)

This mode is reserved for future definition. If selected, it defaults to Mode 5.

9.4.6 Bootstrap From Host (Mode 5)

In this mode, the bootstrap ROM is enabled and the bootstrap program is executed. This is similar to Mode 1 except that the bootstrap program loads internal PRAM from the host port.

NOTE

The difference between Modes 1 and 5 in the DSP56002 and Mode 1 in the DSP56001 may be considered software incompatibility. A DSP56001 program that reloads the internal PRAM from the host port by setting MB-MA = 01 (assuming external pull-up resistor on bit 23 of P:$C000) will not work correctly in the DSP56002. In the DSP56002, the program would trigger a bootstrap from the external EPROM. The solution is to modify the DSP56001 program to set MC-MA = 101

9.4.7 Bootstrap From SCI (Mode 6)

In this mode, the bootstrap ROM is enabled and the bootstrap program is executed. The internal and/or external program RAM is loaded from the SCI serial interface. The number of program words to load and the starting address must be specified. The SCI bootstrap code expects to receive three bytes specifying the number of program words, three bytes specifying the address from which to start loading the program words, and then three bytes for each program word to be loaded. The number of words, the starting address and the program words are received least significant byte first, followed by the mid-, and then by the most significant byte. After receiving the program words, program execution starts at the address where the first instruction was loaded. The SCI is programmed to work in asynchronous mode with 8 data bits, 1 stop bit, and no parity. The clock source is external and the clock frequency must be 16x the baud rate. After each byte is received, it is echoed back

through the SCI transmitter. Refer to Section 12 DSP Serial Ports for more information on the SCI port boot mode.

9.4.8 Reserved (Mode 7)

This mode is reserved for future definition. If selected, the processor defaults to Mode 6.

9.5 DSP56002 INTERRUPT PRIORITY REGISTER

Section 7 of DSP56KFAMUM/AD the *DSP56000 Digital Signal Processor Family Manual* describes interrupt (exception) processing in detail. It discusses interrupt sources, interrupt types, and interrupt priority levels (IPL).

Interrupt priority levels for each on-chip peripheral device and for each external interrupt source can be programmed under software control by writing to the interrupt priority register. Level 3 interrupts are nonmaskable, and interrupts of levels 0-2 are maskable.

The DSP56002 Interrupt Priority Register (IPR) configuration is shown in Figure 9-4. The starting addresses of interrupt vectors in the DSP56002 are defined as shown in Table 9-4, while the relative priorities of exceptions within the same IPL are defined as shown in Table 9-5).

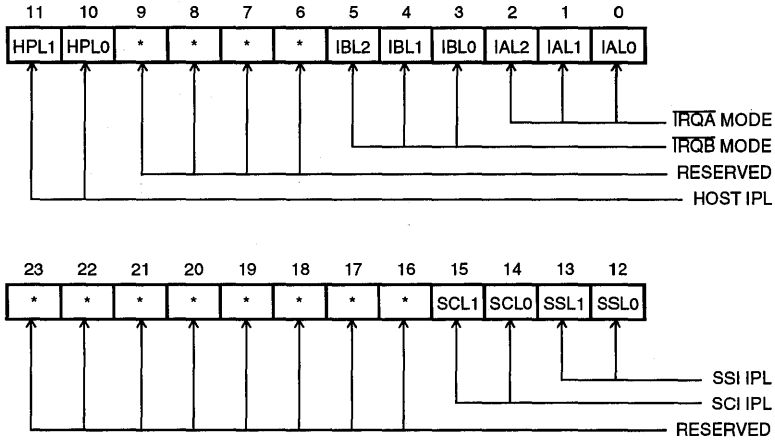
9.6 DSP56002 PHASE-LOCKED LOOP (PLL) MULTIPLICATION FACTOR

Section 9 of DSP56KFAMUM/AD the *DSP56000 Digital Signal Processor Family Manual* discusses the details of the PLL. The multiplication factor determines the frequency at which the voltage controlled oscillator (VCO) will oscillate. The user sets the multiplication factor by writing to the MF0-MF11 bits in the PLL control register.

See Section 3 Clock Generation and Low Power Control for default multiplication factor settings during hardware reset.

Table 9-4. Interrupt Vectors

Interrupt Starting Address	IPL	Interrupt Source
P:$0000	3	Hardware $\overline{\text{RESET}}$
P:$0002	3	Stack Error
P:$0004	3	Trace
P:$0006	3	SWI
P:$0008	0 - 2	IRQA
P:$000A	0 - 2	IRQB
P:$000C	0 - 2	SSI Receive Data
P:$000E	0 - 2	SSI Receive Data With Exception Status
P:$0010	0 - 2	SSI Transmit Data
P:$0012	0 - 2	SSI Transmit Data with Exception Status
P:$0014	0 - 2	SCI Receive Data
P:$0016	0 - 2	SCI Receive Data with Exception Status
P:$0018	0 - 2	SCI Transmit Data
P:$001A	0 - 2	SCI Idle Line
P:$001C	0 - 2	SCI Timer
P:$001E	3	NMI
P:$0020	0 - 2	Host Receive Data
P:$0022	0 - 2	Host Transmit Data
P:$0024	0 - 2	Host Command (Default)
P:$0026	0 - 2	Available for Host Command
P:$003A	0 - 2	Available for Host Command
P:$003C	0 - 2	Available for Host Command
P:$003E	3	Illegal Instruction
P:$0040	0 - 2	Available for Host Command
P:$007E	0 - 2	Available for Host Command



Bits 6 to 9 and 16 to 23 are reserved, read as zero, and should be written with zero for future compatibility.

9

Figure 9-4. DSP56002 Interrupt Priority Register (IPR)

Table 9-5. Exception Priorities within an IPL

Priority	Exception
Level 3 (Nonmaskable)	
Highest	Hardware $\overline{\text{RESET}}$
	Illegal Instruction
	NMI
	Stack Error
	Trace
Lowest	SWI
Levels 0, 1, 2 (Maskable)	
Highest	IRQA (External Interrupt)
	$\overline{\text{IRQB}}$ (External Interrupt)
	Host Command Interrupt
	Host Receive Data Interrupt
	Host Transmit Data Interrupt
	SSI RX Data with Exception Interrupt
	SSI RX Data Interrupt
	SSI TX Data with Exception Interrupt
	SSI TX Data Interrupt
	SCI RX Data with Exception Interrupt
	SCI RX Data Interrupt
	SCI TX Data with Exception Interrupt
	SCI TX Data Interrupt
	SCI Idle Line Interrupt
Lowest	SCI Timer Interrupt

SECTION 10
DSP PORT A

10.1 INTRODUCTION

The DSP port A provides a versatile interface to external memory, allowing economical connection with fast memories/devices, slow memories/devices, and multiple bus master systems.

Port A on the MC68356 is identical to the port A interface of the DSP56002 with two exceptions: the bus strobe and bus grant pins have been multiplexed with the wait and bus strobe pins. The functionality of these pins can be selected using the DBR/DBG Enable - BR/BGen bit in the DSPACR, and the DSP to IMP direct access interface hardware has been added.

Port A has two power-reduction features. It can access internal memory spaces, toggling only the external memory signals that need to change, thereby eliminating unneeded switching current. Also, if conditions allow the processor to operate at a lower memory speed, wait states can be added to the external memory access to significantly reduce power while the processor accesses those memories.

10.2 DSP PORT A INTERFACE

The 56002 processor can access one or more of its memory sources (X data memory, Y data memory, and program memory) while it executes an instruction. The memory sources may be either internal or external to the DSP. Three address buses (XAB, YAB, and PAB) and four data buses (XDB, YDB, PDB, and GDB) are available for internal memory accesses during one instruction cycle. Port A's one address bus and one data bus are available for external memory accesses.

If all memory sources are internal to the DSP, one or more of the three memory sources may be accessed in one instruction cycle (i.e., program memory access or program memory access plus an X, Y, XY, or L memory reference). However, when one or more of the memories are external to the chip, memory references may require additional instruction cycles because only one external memory access can occur per instruction cycle.

If an instruction cycle requires more than one external access, the processor will make the accesses in the following priority: X memory, Y memory, and program memory. It takes one instruction cycle for each external memory access – i.e., one access can be executed in one instruction cycle, two accesses take two instruction cycles, etc. Since the external data bus is only 24 bits wide, one XY or long external access will take two instruction cycles. The 16-bit address bus can sustain a rate of one memory access per instruction cycle (using no-wait-state memory which is discussed in 10.4 Port A Wait States).

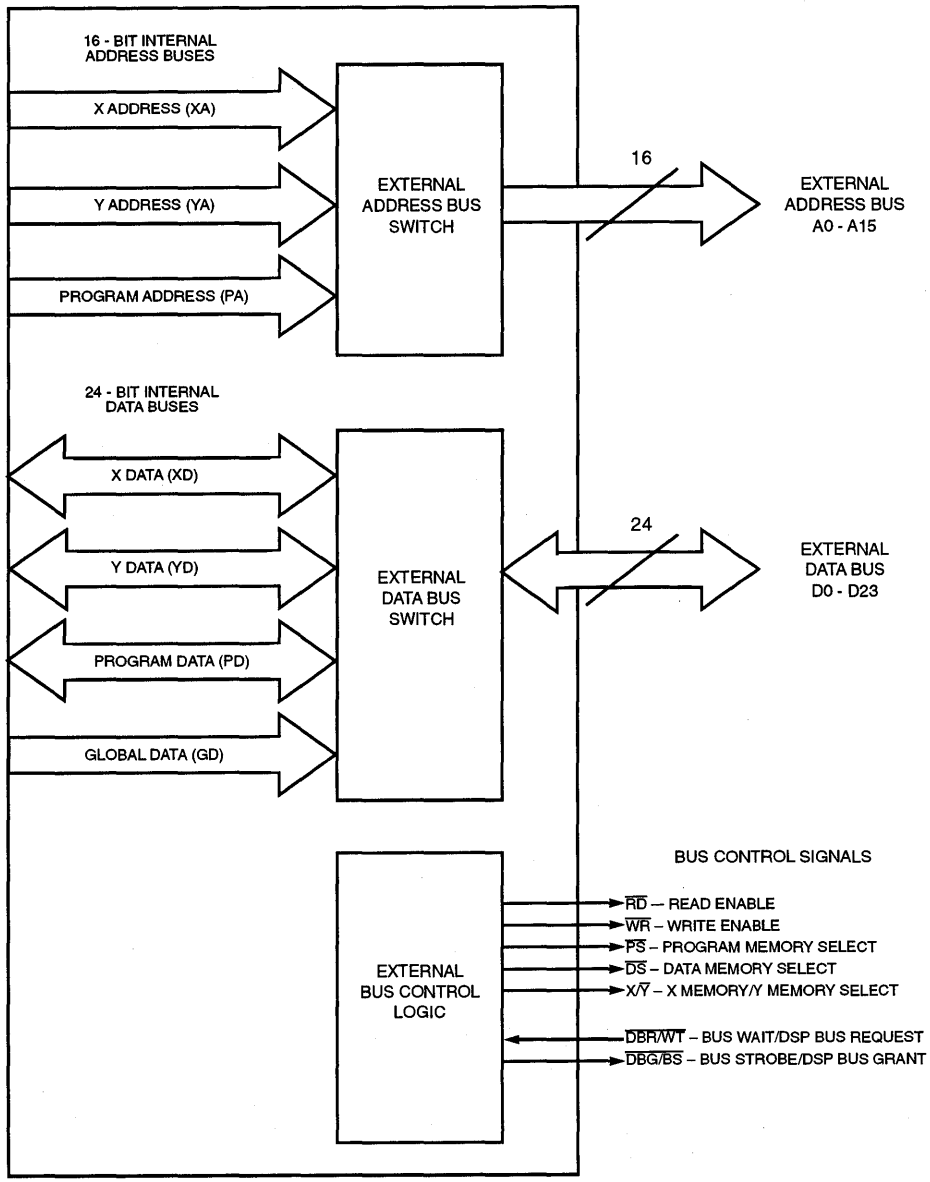


Figure 10-1. Port A Signals

NOTE

The 56002 BR and WT pins, and the BG and BS pins share the same pin on the MC68356. Either bus request/bus grant or external wait state generation functionality can be enabled but not both.

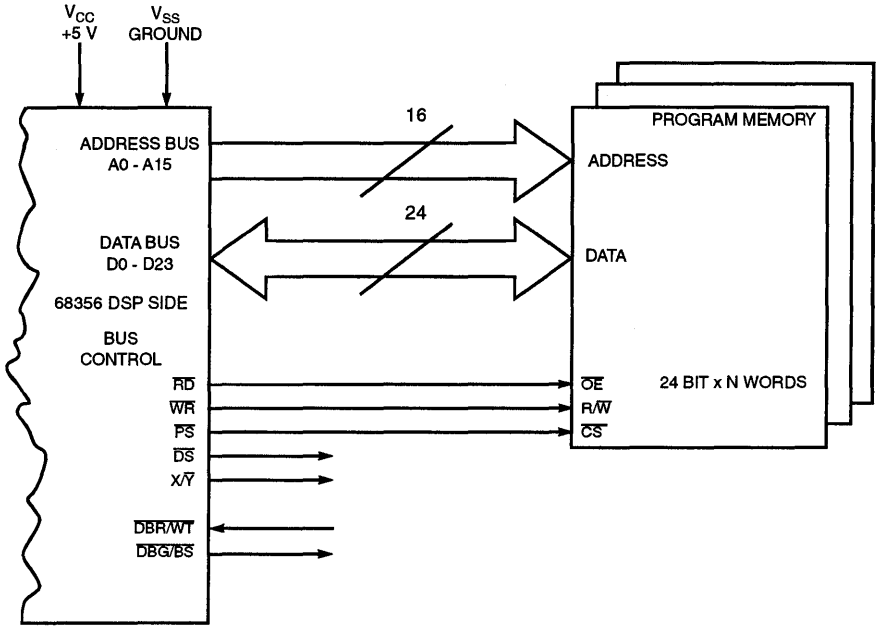


Figure 10-2. External Program Space

10

Figure 10-1 shows the port A signals divided into their three functional groups: address bus signals (A0-A15), data bus signals (D0-D15), and bus control. The bus control signals can be subdivided into three additional groups: read/write control ($\overline{RD}$ and $\overline{WR}$), address space selection (including program memory select ($\overline{PS}$), data memory select ($\overline{DS}$), and X/$\overline{Y}$ select) and bus access control ($\overline{DBR/WT}$, $\overline{BS/BG}$).

The read/write controls can act as decoded read and write controls, or, as seen in Figure 10-2, Figure 10-3, and Figure 10-4, the write signal can be used as the read/write control, and the read signal can be used as an output enable (or data enable) control for the memory. Decoding in such a way simplifies connection to high-speed random-access memories (RAMs). The program memory select, data memory select, and X/Y select can be considered additional address signals, which extend the directly addressable memory from 64K words to 192K words total.

Since external logic delay is large relative to RAM timing margins, timing becomes more difficult as faster DSPs are introduced. The separate read and write strobes used by the 56002 are mutually exclusive, with a guard time between them to avoid an instance where two data buffers are enabled simultaneously. Other methods using external logic gates to generate the RAM control inputs require either faster RAM chips or external data buffers to avoid data bus buffer conflicts.

Figure 10-2 shows an example of external program memory. A typical implementation of this circuit would use three-byte-wide static memories and would not require any additional logic.

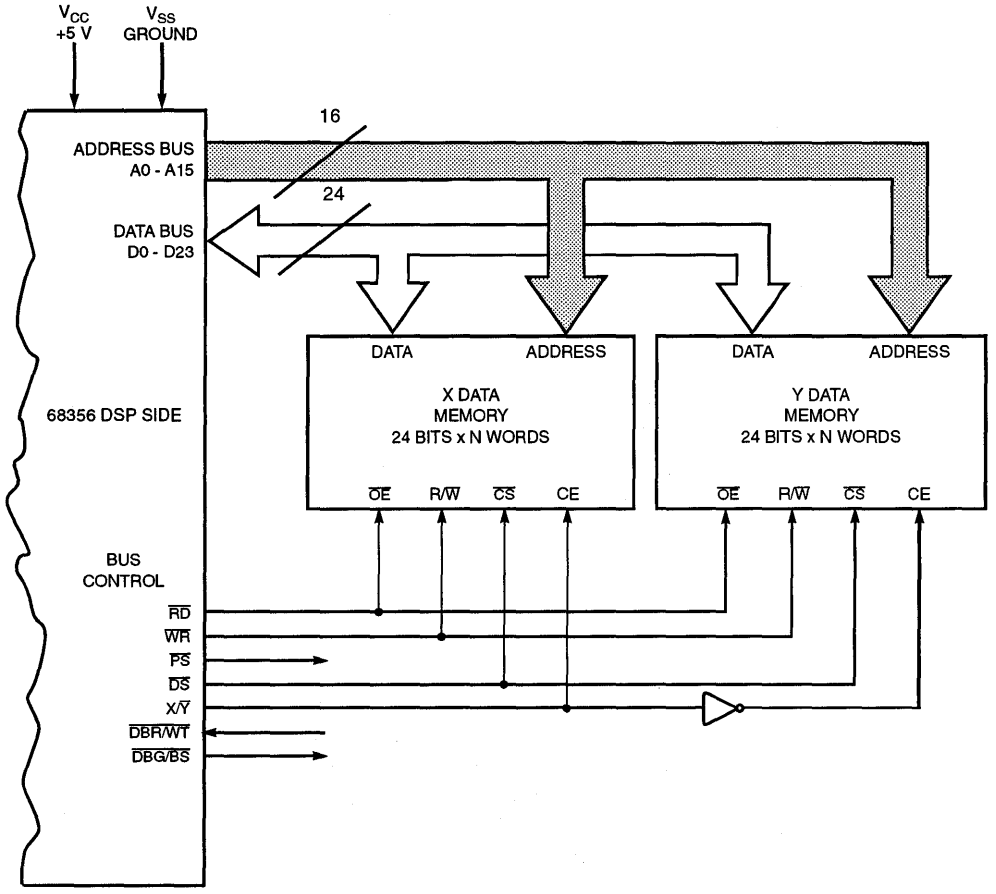


Figure 10-3. External X and Y Data Space

The $\overline{PS}$ signal is used as the program-memory chip-select signal to enable the program memory at the appropriate time.

Figure 10-3 shows a similar circuit using the $\overline{DS}$ signal to enable two data memories and using the X/$\overline{Y}$ signal to select between them. The three external memory spaces (program, X data, and Y data) do not have to reside in separate physical memories; a single memory can be employed by using the $\overline{PS}$, $\overline{DS}$, and X/$\overline{Y}$ signals as additional address lines to segment the memory into three spaces (see Figure 10-4). Table 10-1 shows how the $\overline{PS}$, $\overline{DS}$, and X/$\overline{Y}$ signals are decoded.

If the DSP is in the development mode, an exception fetch to any interrupt vector location will cause the X/$\overline{Y}$ signal to go low when $\overline{PS}$ is asserted. This procedure is useful for debugging and for allowing external circuitry to track interrupt servicing.

Figure 10-5 shows a system that uses internal program memory loaded from an external ROM during power-up and splits the data memory space of a single memory bank into X:

Table 10-1. Program and Data Memory Select Encoding

$\overline{PS}$	$\overline{DS}$	X/$\overline{Y}$	External Memory Reference
1	1	1	No Activity
1	0	1	X Data Memory on Data Bus
1	0	0	Y Data Memory on Data Bus
0	1	1	Program Memory on Data Bus (Not an Exception)
0	1	0	External Exception Fetch: Vector or Vector +1 (Development Mode Only)
0	0	X	Reserved
1	1	0	Reserved

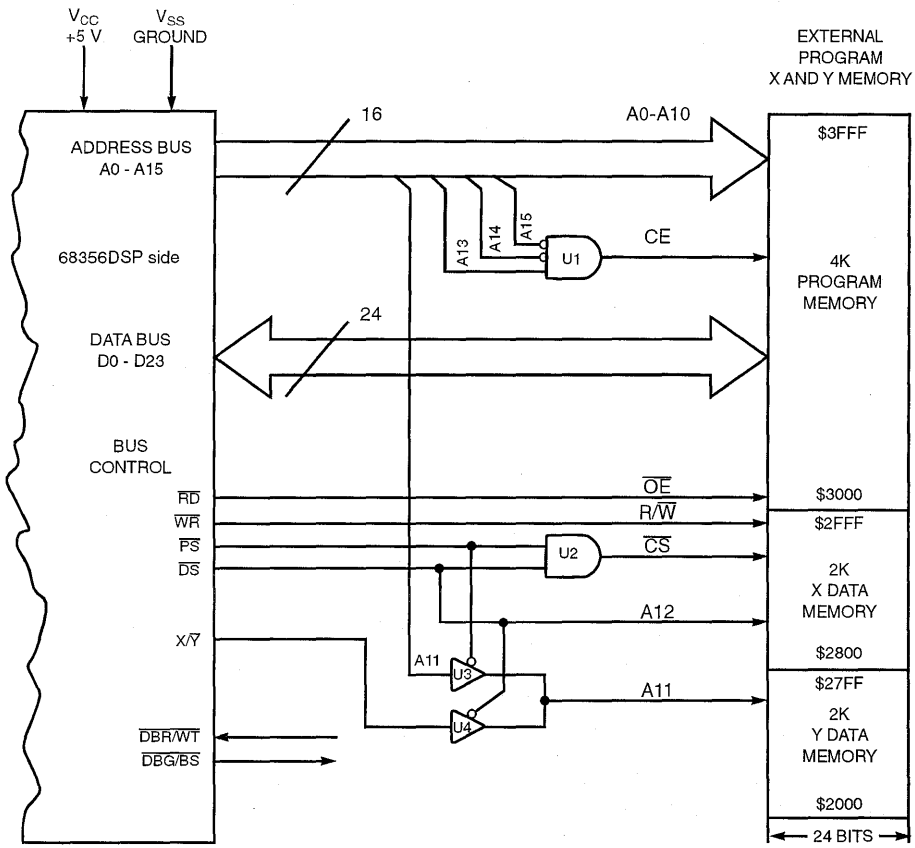


Figure 10-4. Memory Segmentation

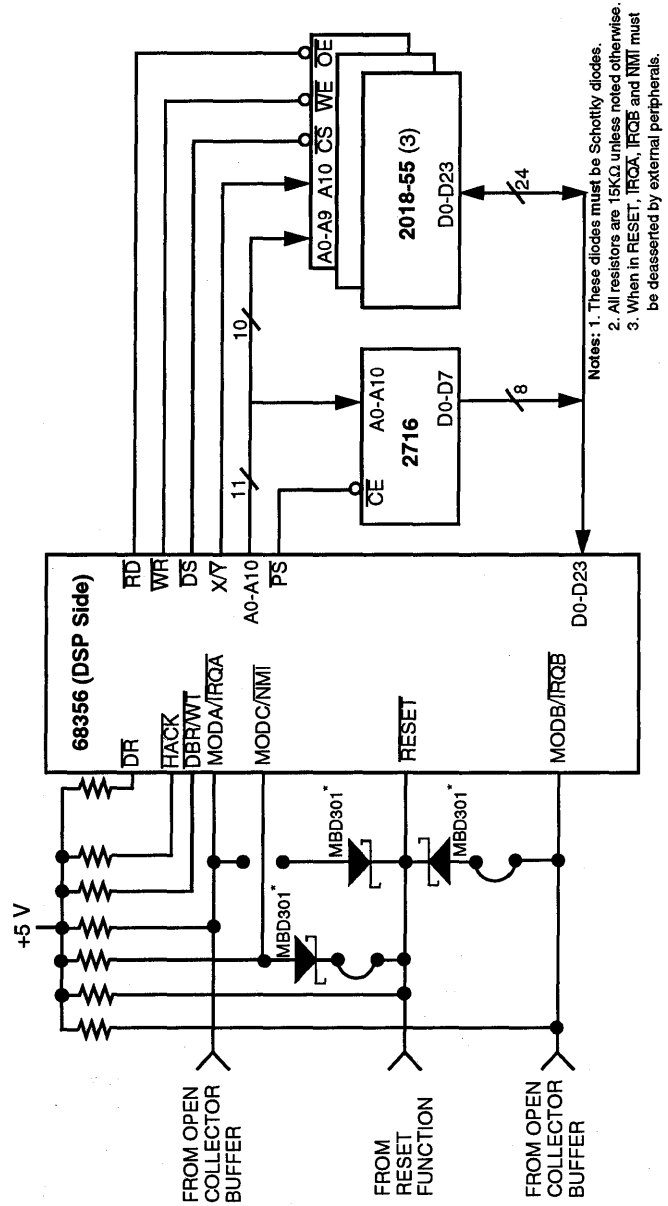


Figure 10-5. Port A Bootstrap ROM with X and Y RAM

and Y: memory spaces. Although external program memory must be 24 bits, external data memory does not. Of course, this is application specific. Many systems use 16 or fewer bits for A/D and D/A conversion and, therefore, they may only need to store 16, 12, or even 8 bits of data. The 24/56 bits of internal precision is usually sufficient for intermediate results. This is a cost saving feature which can reduce the number of external memory chips.

10.3 PORT A TIMING

The external bus timing is defined by the operation of the address bus, data bus, and bus control pins. The transfer of data over the external data bus is synchronous with the clock. The timing A, B, and C relative to the edges of an external clock (see Figure 10-6 and Figure 10-7) are provided in Section 14 Electrical Characteristics. This timing is essential for designing synchronous multiprocessor systems. Figure 10-6 shows the port A timing with no wait states (wait-state control is discussed in 10.4 Port A Wait States). One instruction cycle equals two clock cycles or four clock phases. The clock phases, which are numbered T0 – T3, are used for timing on the DSP. Figure 10-7 shows the same timing with two wait states added to the external X: memory access. Four TW clock phases have been added because one wait state adds two T phases and is equivalent to repeating the T2 and $\overline{T2}$ clock phases. The write signal is also delayed from the T1 to the T2 state when one or more wait states are added to ease interfacing to the port. Each external memory access requires the following procedure:

1. The external memory address is defined by the address bus (A0–A15) and the memory reference selects ($\overline{PS}$, $\overline{DS}$, and X/$\overline{Y}$). These signals change in the first phase (T0) of the bus cycle. Since the memory reference select signals have the same timing as the address bus, they may be used as additional address lines. The address and memory reference signals are also used to generate chip-select signals for the appropriate memory chips. These chip-select signals change the memory chips from low-power standby mode to active mode and begin the read access time. This mode change allows slower memories to be used since the chip-select signals can be address based rather than read or write enable based. Read and write enable do not become active until after the address is valid. See the timing diagrams in Section 14

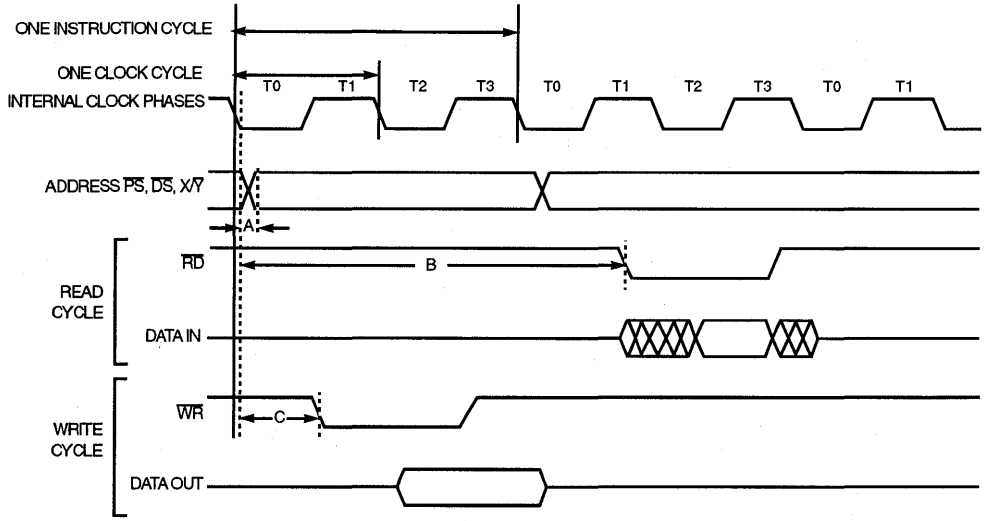


Figure 10-6. Port A Bus Operation with No Wait States

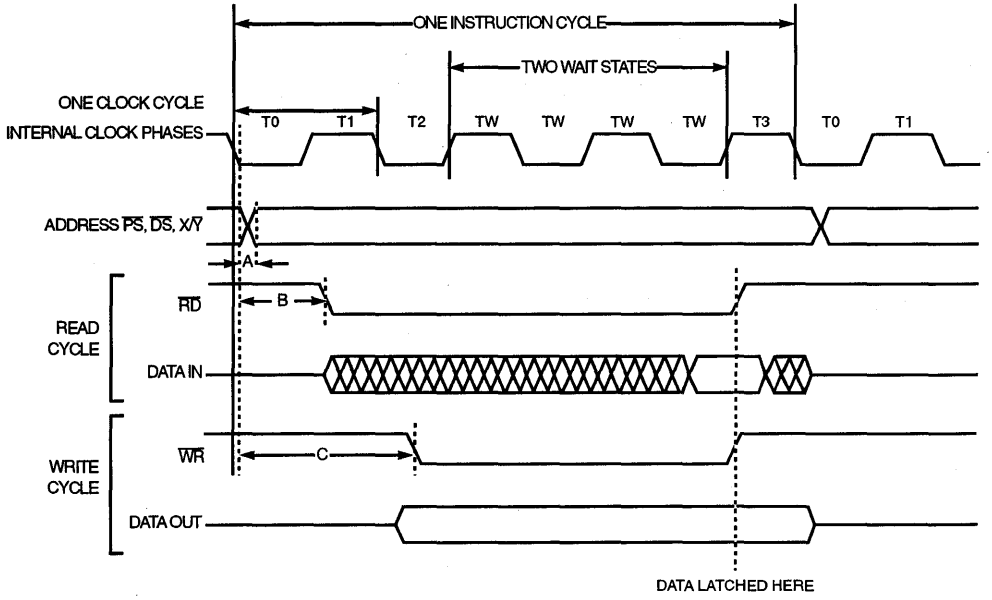


Figure 10-7. Port A Bus Operation with Two Wait States

10

Electrical Characteristics for detailed timing information.

2. When the address and memory reference signals are stable, the data transfer is enabled by read enable ($\overline{RD}$) or write enable ($\overline{WR}$). $\overline{RD}$ or $\overline{WR}$ is asserted to qualify the address and memory reference signals as stable and to perform the read or write data transfer. $\overline{RD}$ and $\overline{WR}$ are asserted in the second phase of the bus cycle (if there are no wait states). Read enable is typically connected to the output enable ($\overline{OE}$) of the memory chips and simply controls the output buffers of the chip-selected memory. Write enable is connected to the write enable ($\overline{WE}$) or write strobe ($\overline{WS}$) of the memory chips and is the pulse that strobes data into the selected memory. For a read operation, $\overline{RD}$ is asserted and $\overline{WR}$ remains deasserted. Since write enable remains negated, a memory read operation is performed. The DSP data bus becomes an input, and the memory data bus becomes an output. For a write operation, $\overline{WR}$ is asserted and $\overline{RD}$ remains deasserted. Since read enable remains deasserted, the memory chip outputs remain in the high-impedance state even before write strobe is asserted. This state assures that the DSP and the chip-selected memory chips are not enabled onto the bus at the same time. The DSP data bus becomes an output, and the memory data bus becomes an input.
3. Wait states are inserted into the bus cycle by a wait-state counter or by asserting the $\overline{WT}$ signal ($\overline{DBR/WT}$ with the BR/BGen bit in the DSPACR register equal to zero). The wait-state counter is loaded from the bus control register. If the value loaded into the wait-state counter is zero, no wait states are inserted into the bus cycle, and $\overline{RD}$ and $\overline{WR}$ are asserted as shown in Figure 10-6. If a value W≠0 is loaded into the wait state counter, W wait states are inserted into the bus cycle. When wait states are inserted

into an external write cycle, $\overline{WR}$ is delayed from T1 to T2. The timing for the case of two wait states (W=2) is shown in Figure 10-7.

4. When $\overline{RD}$ or $\overline{WR}$ are deasserted at the start of T3 in a bus cycle, the data is latched in the destination device – i.e., when $\overline{RD}$ is deasserted, the DSP latches the data internally; when $\overline{WR}$ is deasserted, the external memory latches the data on the positive-going edge. The address signals remain stable until the first phase of the next external bus cycle to minimize power dissipation. The memory reference signals ($\overline{PS}$, $\overline{DS}$, and X/$\overline{Y}$) are deasserted (held high) during periods of no bus activity, and the data signals are three-stated. For read-modify-write instructions such as BSET, the address and memory reference signals remain active for the complete composite (i.e., two I_{cyc}) instruction cycle.

Figure 10-8 shows an example of mixing different memory speeds and memory-mapped peripherals in different address spaces. The internal memory uses no wait states, X: memory uses two wait states, Y: memory uses four wait states, P: memory uses five wait states, and the analog converters use 14 wait states. Controlling five different devices at five different speeds requires only one additional logic package. Half the gates in that package are used to map the analog converters to the top 64 memory locations in Y: memory.

10.4 PORT A WAIT STATES

The 56002 features two methods to allow the user to accommodate slow memory by changing the port A bus timing. The first method uses the bus control register (BCR), which allows a fixed number of wait states to be inserted in a given memory access to all locations in each of the four memory spaces: X, Y, P, and I/O. The second method uses the bus strobe ($\overline{BS}$) and bus wait ($\overline{WT}$) facility, which allows an external device to insert an arbitrary number of wait states when accessing either a single location or multiple locations of external memory or I/O space. Wait states are executed until the external device releases the DSP to finish the external memory cycle.

Table 10-2. Wait State Control

BCR Contents	WT	Number of Wait States Generated
0	Deasserted	0
0	Asserted	2 (minimum)
> 0	Deasserted	Equals value in BCR
> 0	Asserted	Minimum equals 2 or value in BCR. Maximum is determined by BCR or $\overline{WT}$, whichever is larger.

10.5 BUS CONTROL REGISTER (BCR)

The BCR determines the expansion bus timing by controlling the timing of the bus interface signals, $\overline{RD}$ and $\overline{WR}$, and the data output lines. It is a memory mapped register located at X:$FFFE. Each of the memory spaces in Figure 10-9 (X data, Y data, program data, and I/

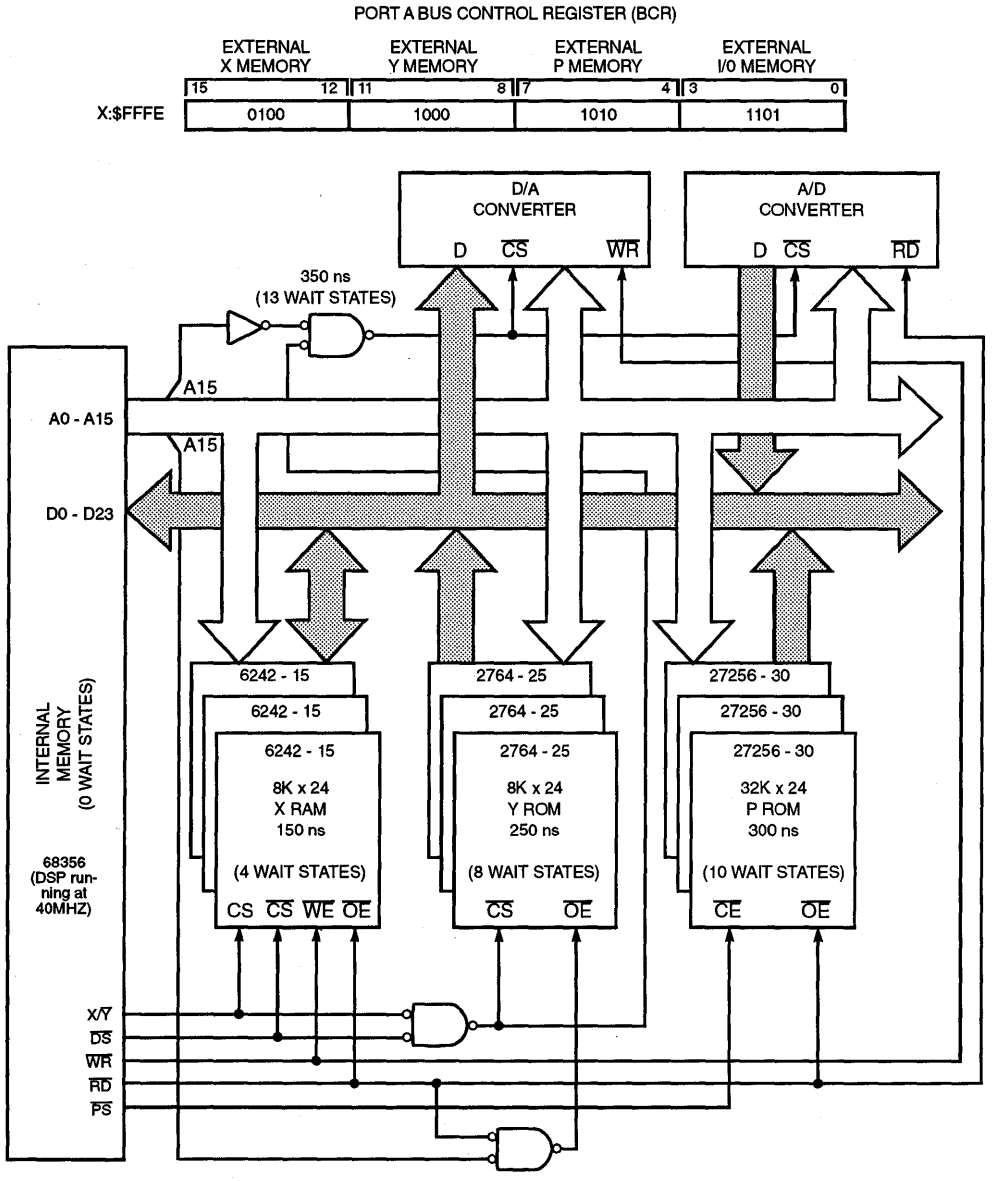
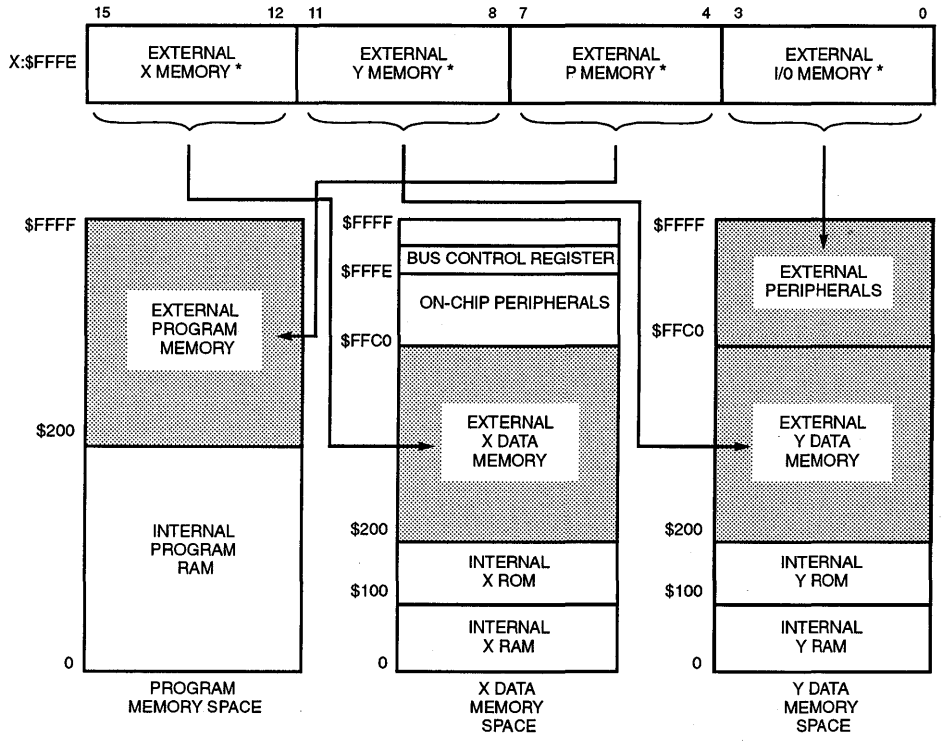


Figure 10-8. Mixed-Speed Expanded System

O) has its own 4-bit BCR, which can be programmed for inserting up to 15 wait states (each wait state adds one-half instruction cycle to each memory access – i.e., 50 ns for a 20-Mhz clock). In this way, external bus timing can be tailored to match the speed requirements of the different memory spaces. **On processor reset, the BCR is preset to all ones (15 wait states).** This allows slow memory to be used for bootstrapping. The BCR needs to be set



* Zero to 15 wait states can be inserted into each external memory access.

Figure 10-9. Bus Control Register

appropriately for the memory being used or the processor will insert 15 wait states between each memory fetch and cause the DSP to run slow.

Figure 10-9 illustrates which of the four BCR subregisters affect which external memory space. All the internal peripheral devices are memory mapped, and their control registers reside between X:$FFC0 and X:$FFFF.

To load the BCR the way it is shown in Figure 10-8, execute a "MOVEP #$48AD, X:$FFFE" instruction. Or, change the individual bits in one of the four subregisters by using the BSET and BCLR instructions which are detailed in the *DSP56000 Digital Signal Processor Family Manual*, Section 6 and Appendix A.

Figure 10-8 shows an example of mixing different memory speeds and memory-mapped peripherals in different address spaces. The internal memory uses no wait states, X: memory uses two wait states, Y: memory uses four wait states, P: memory uses five wait states, and the analog converters use 14 wait states. Controlling five different devices at five different speeds requires only one additional logic package. Half the gates in that package are used to map the analog converters to the top 64 memory locations in Y: memory.

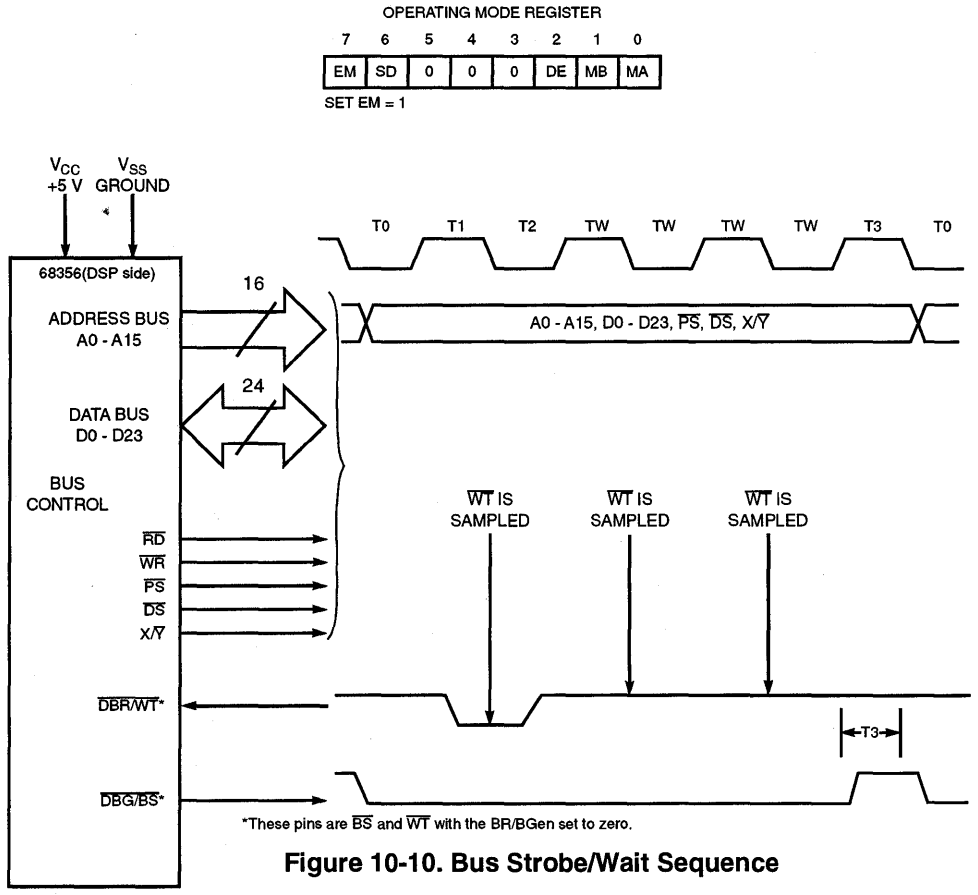


Figure 10-10. Bus Strobe/Wait Sequence

Adding wait states to external memory accesses can substantially reduce power requirements. Consult Section 14 Electrical Characteristics for specific power consumption requirements.

10.6 BUS STROBE AND WAIT PINS

The ability to insert wait states using $\overline{BS}$ and $\overline{WT}$ provides a means to connect asynchronous devices to the DSP, allows devices with differing timing requirements to reside in the same memory space, allows a bus arbiter to provide a fast multiprocessor bus access, and provides another means of halting the DSP at a known program location with a fast restart. In order to use the $\overline{BS}$ and $\overline{WT}$ pins, the DBR/DBG Enable - BRBGen bit in the DSPACR register must be set to zero.

The timing of the $\overline{BS}$ and $\overline{WT}$ pins is illustrated in Figure 10-10. Every external access, $\overline{BS}$ is asserted concurrently with the address lines in T0. $\overline{BS}$ can be used by external wait-state logic to establish the start of an external access. $\overline{BS}$ is deasserted in T3 of each external bus cycle, signaling that the current bus cycle will complete. Since the $\overline{WT}$ signal is internally

synchronized, it can be asserted asynchronously with respect to the system clock. The $\overline{WT}$ signal should only be asserted while $\overline{BS}$ is asserted. Asserting $\overline{WT}$ while $\overline{BS}$ is deasserted will give indeterminate results. However, for the number of inserted wait states to be deterministic, $\overline{WT}$ timing must satisfy setup and hold timing with respect to the negative-going edge of EXTAL. The setup and hold times are provided in Section 14 Electrical Characteristics. The timing of $\overline{WR}$ is controlled by the BCR and is independent of $\overline{WT}$. The minimum number of wait states that can be inserted using the $\overline{WT}$ pin is two. The BCR is still operative when using $\overline{BS}$ and $\overline{WT}$ and defines the minimum number of wait states that are inserted. Table 10-2 summarizes the effect of the BCR and $\overline{WT}$ pin on the number of wait states generated.

10.7 BUS ARBITRATION AND SHARED MEMORY

The 56002 has four signals muxed on two pins that control port A. They are bus request ($\overline{DBR}$), bus grant ($\overline{DBG}$), bus strobe ($\overline{BS}$) and bus wait ($\overline{WT}$) and they are described in Section 2 Signal Description. The four signals are muxed onto two pins: $\overline{DBR}$ and $\overline{WT}$, and $\overline{DBG}$ and $\overline{BS}$. The pin functionality is selected through the DBR/DBG Enable - BRBGEN bit in the DSPACR register.

The bus control signals provide the means to connect additional bus masters (which may be additional DSPs, microprocessors, direct memory access (DMA) controllers, etc.) to the port A bus. They work together to arbitrate and determine what device gets access to the bus.

If an external device has requested the external bus by asserting the $\overline{DBR}$ input, and the DSP has granted the bus by asserting $\overline{BG}$, the DSP will continue to process as long as it requires no external bus accesses itself. If the DSP **does** require an external access but is not the bus master, it will stop processing and remain in wait states until it regains bus ownership.

Three examples of bus arbitration will be described later in this section: 1) bus arbitration using only $\overline{DBR}$ and $\overline{DBG}$ with internal control, 2) bus arbitration using $\overline{DBR}$, $\overline{DBG}$ and $\overline{WT}$, $\overline{BS}$ with no overhead, and 3) signaling using semaphores.

The $\overline{DBR}$ input allows an external device to request and be given control of the external bus while the DSP continues internal operations using internal memory spaces. This allows a bus controller to arbitrate a multiple bus-master system. (A bus master can issue addresses on the bus; a bus slave can respond to addresses on the bus. A single device can be both a master and a slave, but can only be one or the other at any given time.)

Before $\overline{DBR}$ is asserted, all port A signals are driven. When $\overline{DBR}$ is asserted (see Figure 10-11), the DSP will assert $\overline{DBG}$ after the current external access cycle completes and will simultaneously three-state (high-impedance) the port A signals (see Section 14 Electrical Characteristics) for exact timing of $\overline{DBR}$ and $\overline{BG}$). The bus is then available to whatever external device has bus mastership. The external device will return bus mastership to the DSP by deasserting $\overline{BR}$. After the DSP completes the current cycle (an internally executed instruction with or without wait states), $\overline{DBG}$ will be deasserted. When $\overline{DBG}$ is deasserted, the A0-A15, $\overline{PS}$, $\overline{DS}$, X/$\overline{Y}$, and $\overline{RD}$, $\overline{WR}$ lines will be driven. However, the data lines will remain in three-state. All signals are now ready for a normal external access.

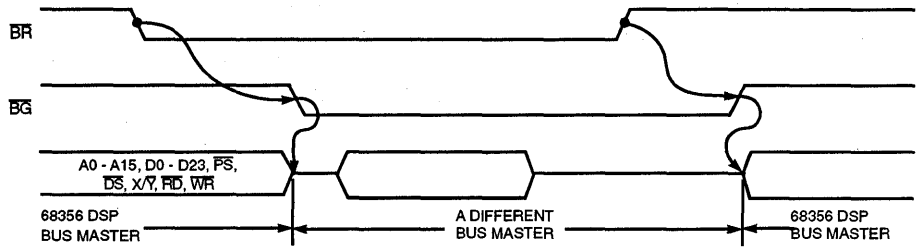


Figure 10-11. Bus Request/Bus Grant Sequence

During the wait state (see Section 7 in the *DSP56000 Digital Signal Processor Family Manual*), the DBR and DBG circuits remain active. However, the port is inactive - the control signals are deasserted, the data signals are inputs, and the address signals remain as the last address read or written. When DBR is asserted, all signals are three-stated (high impedance). Table 10-3 shows the status of DBR and DBG during the wait state.

10.7.1 Bus Arbitration Using Only DBR and DBG With Internal Control

Perhaps the simplest example of a shared memory system using a DSP56002 is shown in Figure 10-12. The bus arbitration is performed within the DSP#2 by using software. DSP#2 controls all bus operations by using I/O pin OUT2 to three-state its own port A and by never accessing port A without first calling the subroutine that arbitrates the bus. When the DSP#2 needs to use external memory, it uses I/O pin OUT1 to request bus access and I/O pin IN1 to read bus grant. DSP#1 does not need any extra code for bus arbitration since the DBR and DBG hardware handles its bus arbitration automatically. The protocol for bus arbitration is as follows:

Table 10-3. DBR and DBG During WAIT

Signal	Before BR Asserted	While BG Asserted	After BR Deasserted	After Return to Normal State (DBG Deasserted)	After First External Access
PS	Driven	Three-state	Three-state	Driven	Driven
DS	Driven	Three-state	Three-state	Driven	Driven
X/Y	Driven	Three-state	Three-state	Driven	Driven
RD	Driven	Three-state	Three-state	Driven	Driven
WR	Driven	Three-state	Three-state	Driven	Driven
Data	Driven	Three-state	Three-state	Three-state	Driven
Address	Driven	Three-state	Three-state	Driven	Driven

At reset: DSP#2 sets OUT2=0 ($\overline{BR}$#2=0) and OUT1=1 ($\overline{BR}$#1=1), which gives DSP#1 access to the bus and suspends DSP#2 bus access.

When DSP#2 wants control of the memory, the following steps are performed (see Figure 10-13):

1. DSP# 2 sets OUT1=0 ($\overline{BR}$#1=0).
2. DSP# 2 waits for IN1=0 ($\overline{BG}$#1=0 and DSP#1 off the bus).
3. DSP#2 sets OUT2=1 ($\overline{BR}$#2=1 to let DSP#2 control the bus).
4. DSP#2 accesses the bus for block transfers, etc. at full speed.
5. To release the bus, DSP#2 sets OUT2=0 ($\overline{BR}$#2=0) after the last external access.
6. DSP#2 then sets OUT1=1 ($\overline{BR}$#1=1) to return control of the bus to DSP#1.
7. DSP#1 then acknowledges mastership by deasserting $\overline{BG}$#1.

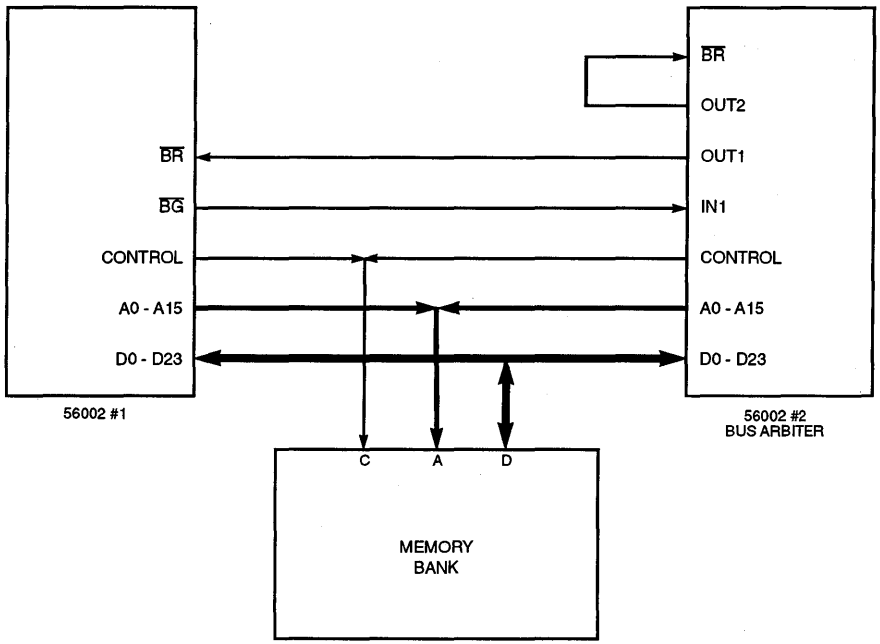


Figure 10-12. Bus Arbitration Using Only $\overline{DBR}$ and $\overline{DBG}$ with Internal Control

10.7.2 Bus Arbitration Using $\overline{DBR}$ and $\overline{BG}$, and $\overline{WT}$ and $\overline{BS}$ With No Overhead

By using the circuit shown in Figure 10-15, two DSPs can share memory with hardware arbitration that requires no software on the part of the DSPs. The protocol for bus arbitration in Figure 10-15 is as follows:

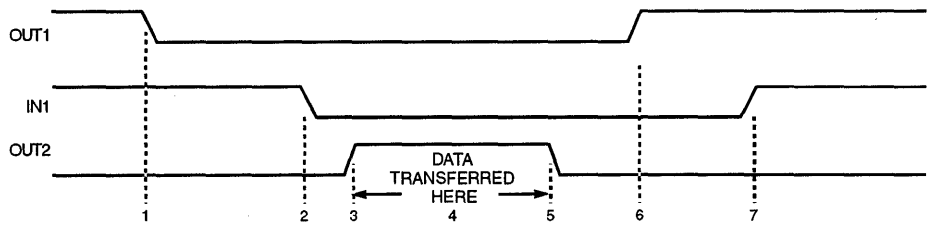


Figure 10-13. Two DSPs with External Bus Arbitration Timing

At RESET assume DSP#1 is not making external accesses so that $\overline{BR}$#2 is deasserted. Hence, $\overline{BG}$ of DSP#2 is deasserted, which three-states the buffers, giving DSP#2 control of the memory.

When DSP#1 wants control of the memory, the following steps are performed (see Figure 10-14):

1. DSP#1 makes an external access, thereby asserting $\overline{BS}$, which asserts $\overline{WT}$ (causing DSP#1 to execute wait states in the current cycle) and asserts DSP#2 $\overline{DBR}$ (requesting that DSP#2 release the bus).
2. When DSP#2 finishes its present bus cycle, it three-states its bus drivers and asserts $\overline{BG}$. Asserting $\overline{BG}$ enables the three-state buffers, placing the DSP#1 signals on the memory bus. Asserting $\overline{BG}$ also deasserts $\overline{WT}$, which allows DSP#1 to finish its bus cycle.

When DSP#1's memory cycle is complete, it releases $\overline{BS}$, which deasserts $\overline{BR}$. DSP#2 then deasserts $\overline{BG}$, three-stating the buffers and allowing DSP#2 to access the memory bus

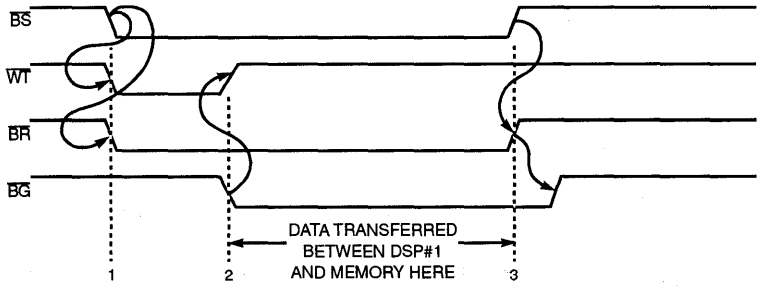


Figure 10-14. Two DSPs with External Bus Arbitration Timing

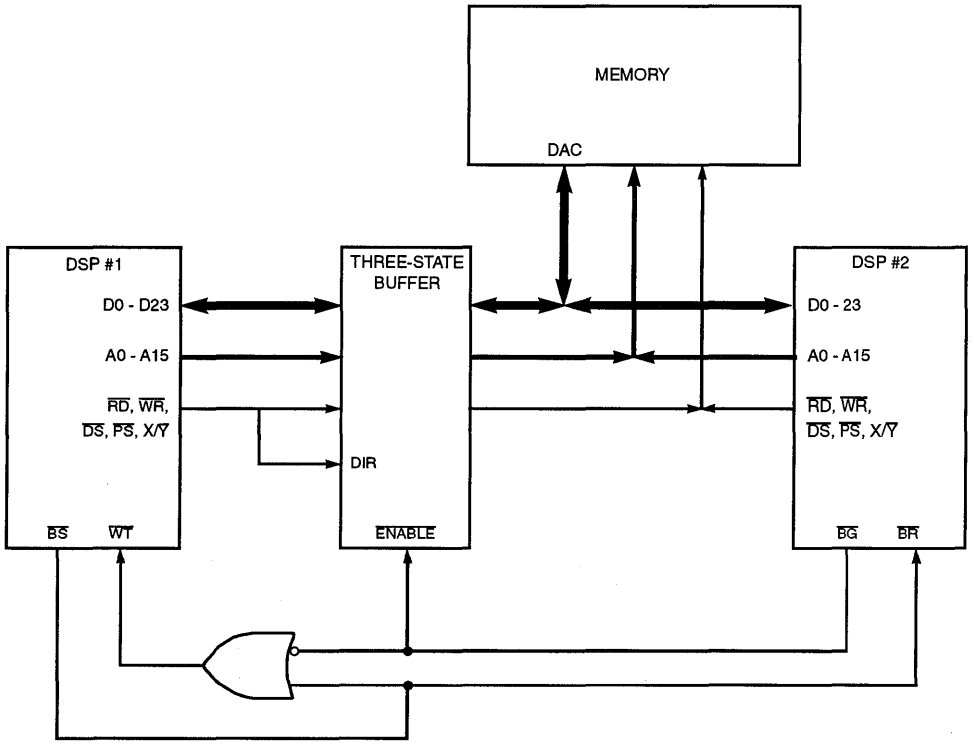


Figure 10-15. Bus Arbitration Using $\overline{DBR}$ and $\overline{BG}$, and $\overline{WT}$ and $\overline{BS}$ with No Overhead

10.7.3 Signaling Using Semaphores

Figure 10-16 shows a more sophisticated shared memory system that uses external arbitration with both local external memory and shared memory. The four semaphores are bits in one of the words in each shared memory bank used by software to arbitrate memory use. Semaphores are commonly used to indicate that the contents of the semaphore's memory blocks are being used by one processor and are not available for use by another processor. Typically, if the semaphore is cleared, the block is not allocated to a processor; if the semaphore is set, the block is allocated to a processor.

Without semaphores, one processor may try to use data while it is being changed by another processor, which may cause errors. This problem can occur in a shared memory system when separate test and set instructions are used to "lock" a data block for use by a single processor.

The **correct procedure** is to test the semaphore and then set the semaphore if it was clear to lock and gain exclusive use of the data block. The problem occurs when the second processor acquires the bus and tests the semaphore after the first processor tests the semaphore but before the first processor can lock the data block. The **incorrect sequence** is:

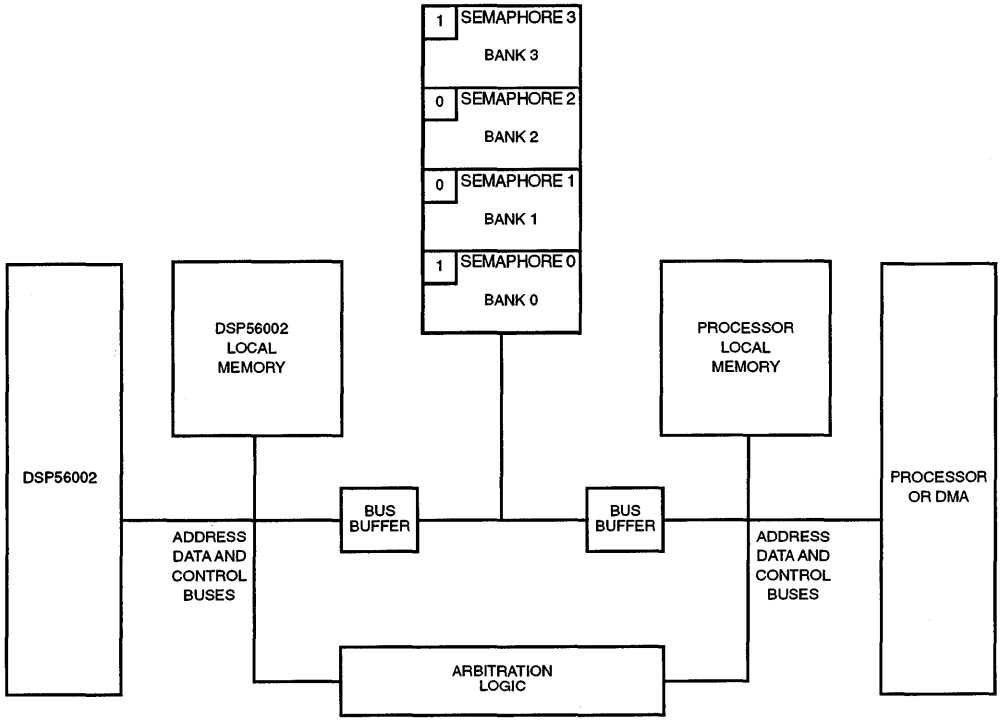


Figure 10-16. Signaling Using Semaphores

1. The first processor tests the semaphore and sees that the block is available
2. The second processor then tests the bit and also sees that the block is available
3. Both processors then set the bit to lock the data
4. Both proceed to use the data on the assumption that the data cannot be changed by another processor.

The DSP56K processor series has a group of instructions designed to prevent this problem. They perform an indivisible read-modify-write operation and do not release the bus between the read and write (specifically, A0–A15, $\overline{DS}$, $\overline{PS}$, and X/$\overline{Y}$ do not change state). **Using a read-modify-write operation allows these instructions to test the semaphore and then to set, clear, or change the semaphore without the possibility of another processor testing the semaphore before it is changed.** The instructions are bit test and change (BCHG), bit test and clear (BCLR), and bit test and set (BSET). (They are discussed in detail in the *DSP56000 Digital Signal Processor Family Manual*) The proper way to set the semaphore to gain exclusive access to a memory block is to use BSET to test the semaphore and to set it to one. After the bit is set, the result of the test operation will reveal if the semaphore was clear before it was set by BSET and if the memory block is available. If the bit was already set and the block is in use by another processor, the DSP must wait to access the memory block.

10.8 DSP TO IMP DIRECT ACCESS MECHANISM

The MC68356 supports a direct access mechanism that allows the DSP to access the 68000 bus locations. The direct access mechanism allows the DSP to create a window in the IMP 68000 memory space and read and write data with no servicing from the 68000 core. This mechanism allows the DSP to store less time critical data in the slower, more economical 68000 bus memory with no wait states required for port A bus cycles.

This gives the system designer three options for implementing DSP memory:

1. The large internal DSP memory space already on board the MC68356.
2. External memory on the DSP port A bus.
3. External memory on the 68000 bus via the direct access mechanism.

The system designer now can use these three options to allocate the DSP memory spaces based on the relative performance and cost tradeoffs of each type of memory.

The DSP to IMP direct access mechanism performs cycle steal 68000 accesses to read and write data from and to the 68000 memory locations. A base address register DSPBAR defines the block or window of memory that the DSP can access in 68000 memory and another base address register defines the block of memory in the 56000 memory map that when accessed will cause a DSP to IMP data access (Figure 10-18).

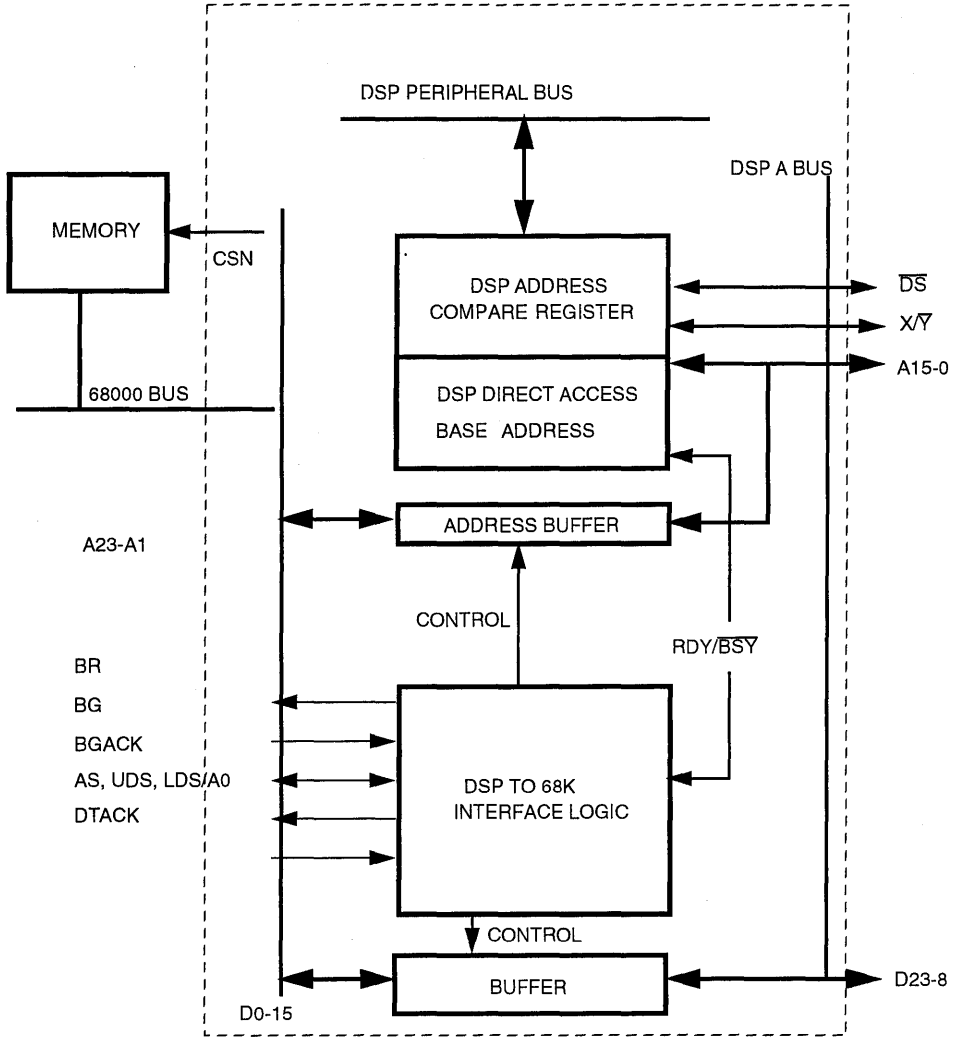
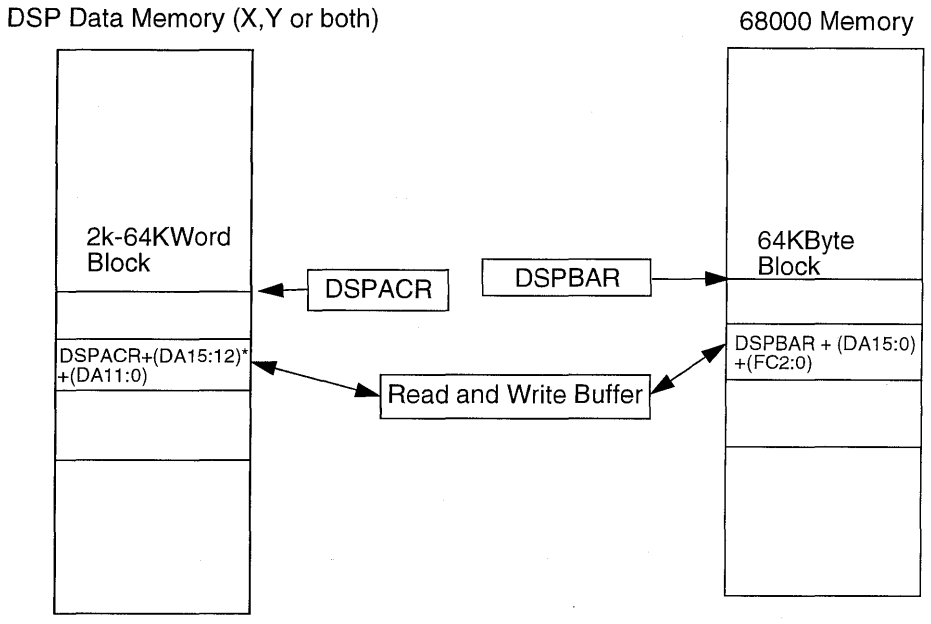


Figure 10-17. DSP to 68000 Direct Access Block Diagram



10

Figure 10-18. DSP to IMP Memory Remapping

10.8.1 DSP to IMP Write Accesses

The block diagram for the DSP to IMP direct access mechanism is shown in Figure 10-17. When the DSP initiates a data write access on its external bus, the MC68356 will compare the A15-A12 address lines space to a previously programmed value located in the DSP address compare register (DSPACR). This register points to the block in the DSP memory map that when written to, will trigger the DSP to IMP direct access controller to initiate a 68000 bus access. If matched, the RDY/$\overline{\text{BSY}}$ bit in the DSPACR will be reset, the data and address bits sampled, and then the direct access logic will begin arbitrating for the 68000 bus. Once the 68000 bus is granted, a standard 68000 write cycle will be generated on the 68000 bus. The high address bits and FC are taken from the DSP base address register, the low address bits are taken from the sampled DSP address lines. When the cycle is terminated with $\overline{\text{DTACK}}$, the 68000 bus will be released and the RDY/$\overline{\text{BSY}}$ bit in the DSPACR will be set to ready. The DSP to IMP direct access mechanism is now ready for the next access.

Because DSP to IMP direct access mechanism write accesses are normal writes to port A, care must be taken to assure that there are no overlapping memory devices in the memory window defined in the DSPACR. Duplicate memory writes will occur if external memory is defined for the same address space.

10.8.2 DSP to IMP Read Accesses

When the DSP initiates a data read access on the port A external bus, the MC68356 will compare the A15-A12 address lines space to the programmable value. If matched, the RDY/$\overline{\text{BSY}}$ bit in the DSP address compare register (DSPACR) will be reset, the data lines DD23-DD8 will be driven with the previous read cycle data, the address bits will be sampled, and then the direct access logic will begin arbitrating for the 68000 bus. Once the 68000 bus is granted, a standard 68000 read cycle will be generated on the 68000 bus. The high address bits and function codes are taken from the DSP base address register, and the low address bits (DA01-DA15) are taken from the sampled DSP address lines.

Because DSP-to-IMP direct access mechanism write accesses are normal writes on port A, care must be taken to assure that there are no overlapping memory devices in the corresponding memory locations on port A. Overlapping memory locations will be written in parallel with the intended write to the DSP-to-IMP window. Read accesses to the DSP-to-IMP window will not appear on external port A pins, and will **not** cause bus contention.

NOTE

If the 68000 bus is configured for 16-bit wide bus (normal 68000 mode), DA0 does not form part of the address on the 68000 side, so odd or even DSP bus accesses will result in even word accesses on the 68000 bus. Thus if it is desired to access successive 16-bit (word) locations on the 16-bit 68000 bus, it will be necessary to increment the DSP address by two.

If the 68000 bus is configured for 8-bits wide (68008 mode), then DA0 will be used as part of the address and the DSP can address odd locations.

When the cycle is terminated with $\overline{\text{DTACK}}$, the data lines are sampled and ready for the next DSP read cycle, the 68000 bus will be released, and the RDY/$\overline{\text{BSY}}$ bit in the DSPACR will be set to ready. The DSP to IMP direct access mechanism is now ready for the next access.

NOTE

When the IMP is configured for 8-bit-bus-mode (BUSW pin low on reset) the direct access controller will not pack the data to and from the 68000 bus. Therefore, in 8-bit mode for an individual DSP to IMP read access, data from D15–D8 of the internal 68000 bus will be invalid and will appear on DD23–DD16 of the DSP PORTA data bus and should be disregarded. Only DD15–DD8 on the DSP bus will represent valid data.

For the case of a write, only one byte from the DSP bus (DD15–DD8) will be written per access. The upper and lower bytes will be disregarded.

The DSP to IMP direct access mechanism has priority over the PCMCIA direct access mechanism and it is not affected by BCLR (See Figure 10-17).

10.9 PROGRAMMER'S MODEL

The DSP to 68000 direct access controller contains two registers, described in the following paragraphs, that must be initialized by the DSP core before making the first access.These registers are the DSP address compare register (DSPACR) and the DSP base address register (DSPBAR).

10.9.1 DSP Address Compare Register - DSPACR

DSPACR — DSP Address Space: X:FFE6

15	14	13	12	11	10	9	8
RES	RES	X/$\overline{Y}$	CBA15	CBA14	CBA13	CBA12	RDY/$\overline{BSY}$
RESET				0			
0	0	0	0		0	0	1

7	6	5	4	3	2	1	0
BR/BGEn	RES	MX/$\overline{Y}$	MA15	MA14	MA13	MA12	EN
RESET							
0	0	0	0	0	0	0	0

Read/Write

The DSPACR resides on the DSP's peripheral space, and should be programmed by the DSP core. This register contains the DSP high address bits, data memory space and the enable for the DSP to 68000 bus direct access. When the DSP performs an external memory cycle, the values in this register are compared with the corresponding DSP values. When a match occurs, the 68000 bus access is initiated.

EN—Enable Bit

This bit enables the DSP to 68000 direct access mechanism. The DSPACR register will be set to default values when the EN bit is written with a one.

0 = The DSP to 68000 direct access mechanism is disabled.
1 = The DSP to 68000 direct access mechanism is enabled.

RDY/$\overline{BSY}$—Ready/Busy Bit

This bit indicates when the DSP2IMP mechanism is ready for the next access. This bit is set at reset.

0 = The DSP2IMP mechanism is busy doing the current access
1 = The DSP2IMP mechanism is ready for the next access.

BR/BGEn—DBR/DBG Enable

This bit selects between the $\overline{DBR}/\overline{DBG}$ and $\overline{BS}/\overline{WT}$ pin pairs

0 = The $\overline{BS}/\overline{WT}$ pins are selected
1 = The $\overline{DBR}/\overline{DBG}$ pins are selected.

CBA12–15—Compare Base address

The base address field contains the DSP starting address of a particular address space that needs to be transferred into the 68000 bus. The address compare logic uses only A15–A12 to cause an address match within its block size.

NOTE

Care must be taken not to map the DSP to IMP direct access memory block into an area already mapped in external DSP space. Duplicate writes and bus contention for reads will result.

X/$\overline{Y}$—X or Y data memory select

0 = Select Y data memory space.
1 = Select X data memory space.

NOTE

The DSP to IMP direct access mechanism can only perform DSP data space accesses for transferring data to and from the 68000 bus. Program space transfers are not allowed.

MX/$\overline{Y}$—Mask X/$\overline{Y}$

0 = The X/$\overline{Y}$ line on the DSP bus is masked. When the DSP performs data accesses, both X and Y data memory accesses will be transferred into the 68000 bus.
1 = The X/$\overline{Y}$ line on the DSP bus is not masked. When the DSP performs data accesses, the X/$\overline{Y}$ bit will be compared along with the DSP address lines to determine whether to generate a DSP to 68000 bus access.

MA15-12—Base Address Mask

These bits are used to set the block size of the direct access. The compare logic will compare only the DSP address lines that are not masked to detect an address match.

0 = The address bit in the corresponding CBA bit location is masked; DSP cycles will be transferred into the 68000 bus without comparing this address line.
1 = The address bit in the corresponding CBA bit location is not masked; DSP cycles will be transferred into the 68000 bus only when the corresponding address matches the unmasked CBR address portion.

10.9.2 DSP Base Address Register - DSPBAR

DSPBAR DSP Address Space: X:FFE7

15	14	13	12	11	10	9	8
BA23	BA22	BA21	BA20	BA19	BA18	BA17	BA16
RESET				0			
0	0	0	0		0	0	0

7	6	5	4	3	2	1	0
RES	RES	RES	RES	FC2	FC1	FC0	RES
RESET							
0	0	0	0	0	0	0	0

Read/Write

This register resides on the DSP's peripheral space, and is programmable from the DSP core. This register contains the high address and function code bits which will be combined with the lower 16-bit address provided by the DSP to form the 68000 address. When the compare logic detects that the access should be transferred onto the 68000 bus, the upper address lines on the 68000 bus will be driven according to the values programmed into this register. The lower address lines are driven from the DSP address lines.

BA16-23—Base address

The base address field should contain the upper address bits of the DSP data address on the 68000 bus. The lower address bits will be taken from the DSP address lines.

FC0-2—Function Code

The function code field should contain the function code of the DSP data address on the 68000 bus.

SECTION 11
DSP HOST PORT

11.1 INTRODUCTION

The host port (HI) is an 8-bit bidirectional interface (see Figure 11-1) and provides a convenient connection to the 68000 core in the IMP section of the MC68356 or to another external IMP 68000 bus master. This section describes the host port, including examples on configuration and usage, and the internal connection features to the 68000 bus.

The HI (also referred to as port B) is a byte-wide, full-duplex, double-buffered, parallel port which may be connected directly to the data bus of a host processor. In this section, the 68000 core or other 68000 bus master that is servicing the host interface will be referred to as the host. The host processor may be any 68000 bus master because this interface looks like static memory. The HI is an asynchronous interface from the 56000 to 68000 bus and consists of two banks of registers – one bank accessible to the 68000 bus processor and a second bank accessible to the DSP CPU (see Figure 11-1). A brief description of the HI features is presented in the following list:

Speed

5 Million Word/Sec Interrupt Driven Data Transfer Rate (This is the maximum interrupt rate for the DSP running at 60MHz – i.e., one interrupt every six instruction cycles.)

Signals

(These signals are connected internally to the 68000 bus (see Figure 11-5)).

H0–H7	Host Data Bus
HA0-HA2	Host Address Select
HR/$\overline{W}$	Host Read/Write Control
$\overline{HEN}$	Host Transfer Enable
$\overline{HREQ}$	Host Request
$\overline{HACK}$	Host Acknowledge

Interface – DSP CPU Side

Mapping: Three X: Memory Locations
Data Word: 24 Bits

Transfer Modes:
- DSP to Host
- Host to DSP
- Host Command

Handshaking Protocols:
- Software Polled
- Interrupt Driven (Fast or Long Interrupts)

Direct Memory Access

Instructions:
Memory-mapped registers allow the standard DSP MOVE instruction to be used.
Special MOVEP instruction provides for I/O service capability using fast interrupts.
Bit addressing instructions (BCHG, BCLR, BSET, BTST, JCLR, JSCLR, JSET, JSSET) simplify I/O service routines.
I/O short addressing provides faster execution with fewer instruction words.

Interface – Host Side

Mapping:
Eight Consecutive Memory Locations

Data Word: Eight Bits

Transfer Modes:
DSP to Host
Host to DSP
Host Command
Mixed 8-, 16-, and 24-Bit Data Transfers

Handshaking Protocols:
Software Polled
Interrupt Driven with 68000 core
Cycle Stealing DMA with Initialization

Dedicated Interrupts:
Separate Interrupt Vectors for Each Interrupt Source
Special host commands force DSP CPU interrupts under host processor control, which are useful for:
Real-Time Production Diagnostics
Debugging Window for Program Development
Host Control Protocols and DMA Setup

Figure 11-1 is a block diagram showing the registers in the HI. These registers can be divided vertically down the middle into registers visible to the host processor on the left and registers visible to the DSP on the right. They can also be divided horizontally into control at the top, DSP-to-host data transfer in the middle (HTX, RXH, RXM, and RXL), and host-to-DSP data transfer at the bottom (THX, TXM, TXL, and HRX).

11.1.1 Enabling the Host Interface

Since the external pins dedicated to Port B parallel operation on the discrete version of the DSP56002 are not implemented on the MC68356, the port B data register and the port B data direction register have no function. However, it is still necessary to set the port B control register (PCB) bit 1 to 0 and bit 0 to 1 (write $000001 to $FFE0). This will enable the host interface.

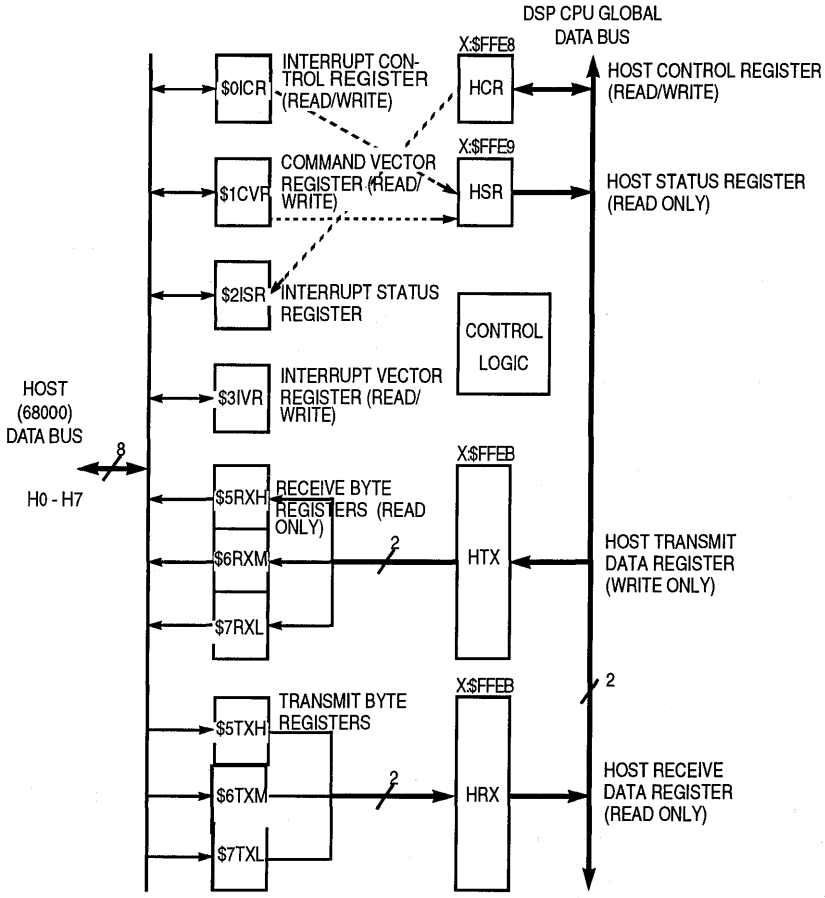


Figure 11-1. HI Block Diagram

11.1.2 Host Interface – DSP CPU Viewpoint

The DSP CPU views the HI as a memory-mapped peripheral occupying three 24-bit words in data memory space. The DSP may use the HI as a normal memory-mapped peripheral, using either standard polled or interrupt programming techniques. Separate transmit and receive data registers are double buffered to allow the DSP and host processor to efficiently transfer data at high speed. Memory mapping allows the DSP CPU communication with the HI registers to be accomplished using standard instructions and addressing modes. In addition, the MOVEP instruction allows HI-to-memory and memory-to-HI data transfers without going through an intermediate register. Both DSP hardware and software reset disable the HI.

11.1.3 Programming Model – DSP CPU Viewpoint

The HI has two programming models: one for the DSP programmer and one for the host processor programmer. In most cases, the notation used reflects the DSP perspective. The HI – DSP programming model is shown in Figure 11-2. There are three registers: a control register (HCR), a status register (HSR), and a data transmit/receive register (HTX/HRX). These registers can only be accessed by the DSP core; they cannot be accessed by the host processor. The HI host processor programming model is shown in Figure 11-7.

The following paragraphs describe the purpose and operation of each bit in each register of the HI visible to the DSP CPU. The effects of the different types of reset on these registers are shown. A brief discussion of interrupts and operation of the DSP side of the HI complete the programming model from the DSP viewpoint. The programming model from the host viewpoint begins at 11.2.3 Programming Model – Host Processor Viewpoint.

11.1.3.1 HOST CONTROL REGISTER (HCR)

The HCR is an 8-bit read/write control register used by the DSP to control the HI interrupts and flags. The HCR cannot be accessed by the host processor. It occupies the low-order byte of the internal data bus; the high-order portion is zero filled. Any reserved bits are read as zeros and should be programmed as zeros for future compatibility. (The bit manipulation instructions are useful for accessing the individual bits in the HCR.) The contents of the HCR are cleared on DSP hardware or software reset. The control bits are described in the following paragraphs.

11.1.3.1.1 HCR Host Receive Interrupt Enable (HRIE) Bit 0

The HRIE bit is used to enable a DSP interrupt when the host receive data full (HRDF) status bit in the host status register (HSR) is set. When HRIE is cleared, HRDF interrupts are disabled. When HRIE is set, a host receive data interrupt request will occur if HRDF is also set. DSP hardware and software resets clear HRIE.

11.1.3.1.2 HCR Host Transmit Interrupt Enable (HTIE) Bit 1

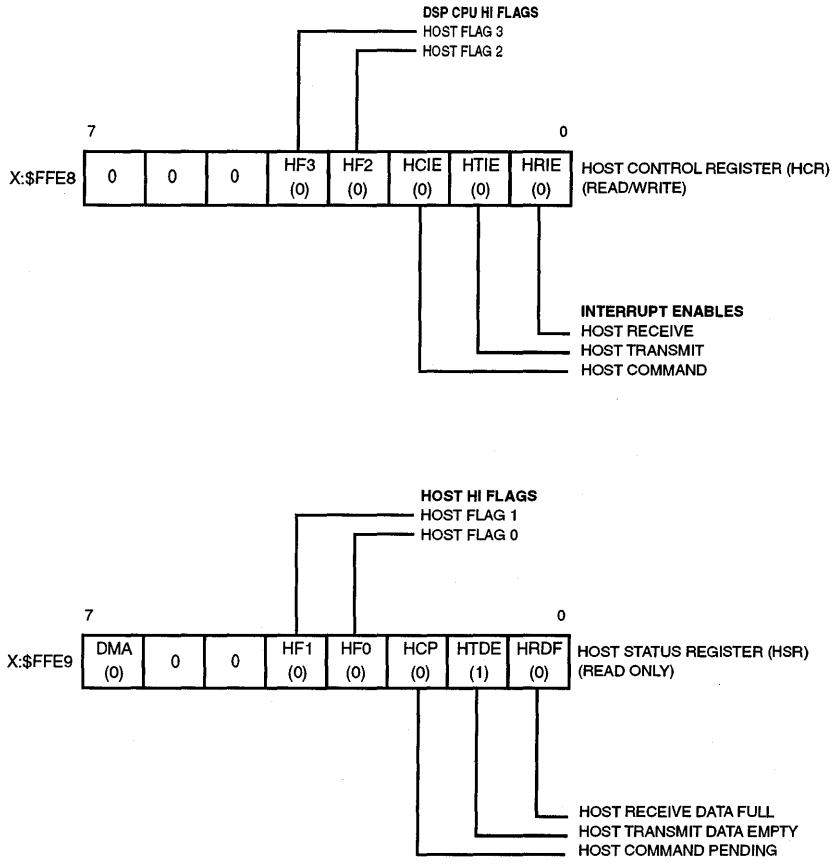
The HTIE bit is used to enable a DSP interrupt when the host transmit data empty (HTDE) status bit in the HSR is set. When HTIE is cleared, HTDE interrupts are disabled. When HTIE is set, a host transmit data interrupt request will occur if HTDE is also set. DSP hardware and software resets clear the HTIE.

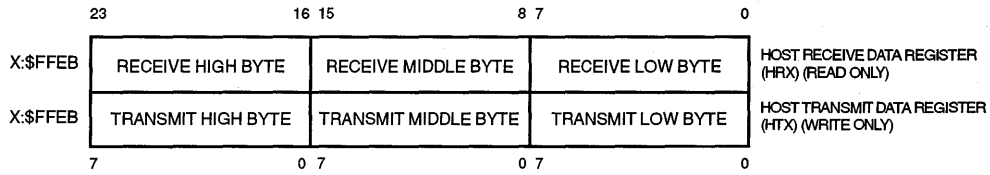
11.1.3.1.3 HCR Host Command Interrupt Enable (HCIE) Bit 2

The HCIE bit is used to enable a vectored DSP interrupt when the host command pending (HCP) status bit in the HSR is set. When HCIE is cleared, HCP interrupts are disabled. When HCIE is set, a host command interrupt request will occur if HCP is also set. The starting address of this interrupt is determined by the host vector (HV). DSP hardware and software resets clear the HCIE.

11.1.3.1.4 HCR Host Flag 2 (HF2) Bit 3

The HF2 bit is used as a general-purpose flag for DSP-to-host communication. HF2 may be set or cleared by the DSP. HF2 is visible in the interrupt status register (ISR) on the host processor side (see Figure 11-3). DSP hardware and software resets clear HF2.





NOTE: The numbers in parentheses are reset values.

Figure 11-2. Host Interface Programming Model – DSP Viewpoint

11.1.3.1.5 HCR Host Flag 3 (HF3) Bit 4

The HF3 bit is used as a general-purpose flag for DSP-to-host communication. HF3 may be set or cleared by the DSP. HF3 is visible in the ISR on the host processor side (see Figure

11-3). DSP hardware and software resets clear HF3.

NOTE

There are four host flags: two used by the host to signal the DSP (HF0 and HF1) and two used by the DSP to signal the host processor (HF2 and HF3). They are general purpose flags and are not designated for any specific purpose. The host flags do not cause interrupts; they must be polled to see if they have changed. These flags can be used individually or as encoded pairs. See 11.1.3.7 Host Port Usage Considerations – DSP Side for additional information. An example of the usage of host flags is the bootstrap loader, which is listed in Appendix C DSP Bootstrap Program. Host flags are used to tell the bootstrap program whether or not to terminate early.

11.1.3.1.6 HCR Reserved Control (Bits 5, 6, and 7)

These unused bits are reserved for future expansion and should be written with zeros for upward compatibility.

11.1.3.2 HOST STATUS REGISTER (HSR)

The HSR is an 8-bit read-only status register used by the DSP to interrogate status and flags of the HI. It cannot be directly accessed by the host processor. When the HSR is read to the internal data bus, the register contents occupy the low-order byte of the data bus; the high-order portion is zero filled. The status bits are described in the following paragraphs.

11.1.3.2.1 HSR Host Receive Data Full (HRDF) Bit 0

The HRDF bit indicates that the host receive data register (HRX) contains data from the host processor. HRDF is set when data is transferred from the TXH:TXM:TXL registers to the HRX register. HRDF is cleared when HRX is read by the DSP. HRDF can also be cleared by the host processor using the initialize function. DSP hardware, software, individual, and STOP resets clear HRDF.

11.1.3.2.2 HSR Host Transmit Data Empty (HTDE) Bit 1

The HTDE bit indicates that the host transmit data register (HTX) is empty and can be written by the DSP. HTDE is set when the HTX register is transferred to the RXH:RXM:RXL registers. HTDE is cleared when HTX is written by the DSP. HTDE can also be set by the host processor using the initialize function. DSP hardware, software, individual, and STOP sets HTDE.

11.1.3.2.3 HSR Host Command Pending (HCP) Bit 2

The HCP bit indicates that the host has set the HC bit and that a host command interrupt is pending. The HCP bit reflects the status of the HC bit in the command vector register (CVR). HC and HCP are cleared by the DSP exception hardware when the exception is taken. The host can clear HC, which also clears HCP. DSP hardware, software, individual, and STOP resets clear HCP.

11.1.3.2.4 HSR Host Flag 0 (HF0) Bit 3

The HF0 bit in the HSR indicates the state of host flag 0 in the ICR on the host processor side. HF0 can only be changed by the host processor (see Figure 11-3). DSP hardware, software, individual, and STOP resets clear HF0.

11.1.3.2.5 HSR Host Flag 1 (HF1) Bit 4

The HF1 bit in the HSR indicates the state of host flag 1 in the ICR on the host processor side. HF1 can only be changed by the host processor (see Figure 11-3). DSP hardware, software, individual, and STOP resets clear HF1.

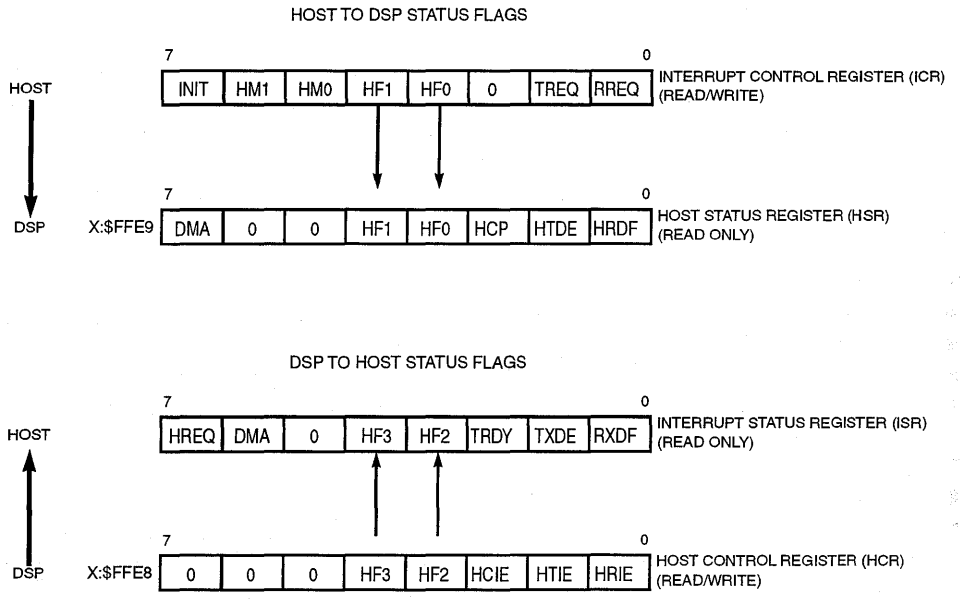


Figure 11-3. Host Flag Operation

11.1.3.2.6 HSR Reserved Status (Bits 5 and 6)

These status bits are reserved for future expansion and read as zero during DSP read operations.

11.1.3.2.7 HSR DMA Status (DMA) Bit 7

The DMA bit indicates that the host processor has enabled the DMA mode of the HI by setting HM1 or HM0 to one. When the DMA bit is zero, it indicates that the DMA mode is disabled by the HM0 and HM1 bits in the ICR and that no DMA operations are pending. When the DMA bit is set, the DMA mode has been enabled if one or more of the host mode bits have been set to one. The channel not in use can be used for polled or interrupt operation by the DSP. DSP hardware, software, individual, and STOP resets clear the DMA bit.

11.1.3.3 HOST RECEIVE DATA REGISTER (HRX)

The HRX register is used for host-to-DSP data transfers. The HRX register is viewed as a 24-bit read-only register by the DSP CPU. The HRX register is loaded with 24-bit data from the transmit data registers (TXH:TXM:TXL) on the host processor side when both the transmit data register empty TXDE (host processor side) and DSP host receive data full (HRDF) bits are cleared. This transfer operation sets TXDE and HRDF. The HRX register contains valid data when the HRDF bit is set. Reading HRX clears HRDF. The DSP may program the HRIE bit to cause a host receive data interrupt when HRDF is set. Resets do not affect HRX.

11.1.3.4 HOST TRANSMIT DATA REGISTER (HTX)

The HTX register is used for DSP-to-host data transfers. The HTX register is viewed as a 24-bit write-only register by the DSP CPU. Writing the HTX register clears HTDE. The DSP may program the HTIE bit to cause a host transmit data interrupt when HTDE is set. The HTX register is transferred as 24-bit data to the receive byte registers (RXH:RXM:RXL) if both the HTDE bit (DSP CPU side) and receive data full (RXDF) status bits (host processor side) are cleared. This transfer operation sets RXDF and HTDE. Data should not be written to the HTX until HTDE is set to prevent the previous data from being overwritten. Resets do not affect HTX.

11.1.3.5 REGISTER CONTENTS AFTER RESET

Table 11-1 shows the results of four reset types on bits in each of the HI registers seen by the DSP CPU. The hardware reset (HW) is caused by the $\overline{\text{DRESET}}$ signal; the software reset (SW) is caused by executing the DSP RESET instruction; the individual reset (IR) is caused by clearing PBC register bits 0 and 1, and the stop reset (ST) is caused by executing the DSP STOP instruction.

11

11.1.3.6 HOST INTERFACE DSP CPU INTERRUPTS

The HI may request interrupt service from either the DSP or the host processor. The DSP CPU interrupts are internal and do not require the use of an external interrupt pin (see Figure 11-4). When the appropriate mask bit in the HCR is set, an interrupt condition caused by the host processor sets the appropriate bit in the HSR, which generates an interrupt request to the DSP CPU. The DSP acknowledges interrupts caused by the host processor by jumping to the appropriate interrupt service routine. The three possible interrupts are 1) receive data register full, 2) transmit data register empty, and 3) host command. The host command can access any interrupt vector in the interrupt vector table although it has a set of vectors reserved for host command use. The DSP interrupt service routine must read or write the appropriate HI register (clearing HRDF or HTDE, for example) to clear the interrupt. In the case of host command interrupts, the interrupt acknowledge from the program controller will clear the pending interrupt condition.

11.1.3.7 HOST PORT USAGE CONSIDERATIONS – DSP SIDE

Synchronization is a common problem when two asynchronous systems are connected, and careful synchronization is required when reading multibit registers that are written by another asynchronous system. The considerations for proper operation on the DSP CPU side are discussed in the following paragraphs, and considerations for the host processor side are discussed in 11.2.5.4 Host Port Usage Considerations–Host Side.

Table 11-1. Host Registers after Reset–DSP CPU Side

Register Name	Register Data	Reset Type			
		HW Reset	SW Reset	IR Reset	ST Reset
HCR	HF(3 - 2)	0	0	—	—
	HCIE	0	0	—	—
	HTIE	0	0	—	—
	HRIE	0	0	—	—
HSR	DMA	0	0	0	0
	HF(1 - 0)	0	0	0	0
	HCP	0	0	0	0
	HTDE	1	1	1	1
	HRDF	0	0	0	0
HRX	HRX (23 - 0)	—	—	—	—
HTX	HTX (23 - 0)	—	—	—	—

DMA, HF1, HF0, HCP, HTDE, and HRDF status bits are set or cleared by the host processor side of the interface. These bits are individually synchronized to the DSP clock.

The only system problem with reading status occurs if HF1 and HF0 are encoded as a pair because each of their four combinations (00, 01, 10, and 11) has significance. There is a small possibility that the DSP will read the status bits during the transition and receive "01" or "10" instead of "11". The solution to this potential problem is to read the bits twice for consensus (See 11.2.5.4 Host Port Usage Considerations–Host Side for additional information).

11.1.4 Host Interface – Host Processor Viewpoint

The HI appears to the host processor as eight words of byte-wide static memory. The host may access the HI asynchronously by using polling techniques or interrupt-based techniques. Separate transmit and receive data registers are double buffered to allow the DSP CPU and host processor to transfer data efficiently at high speed. The HI contains a rudimentary DMA controller, which makes generating addresses (HA0–HA2) for the TX/RX registers in the HI unnecessary.

11.2 HOST PORT TO 68000 BUS CONNECTION

The DSP host port may only be accessed by the 68K bus through internal logic that connects the DSP bus to the 68000 bus. There are no external host port pins. Figure 11-5 shows

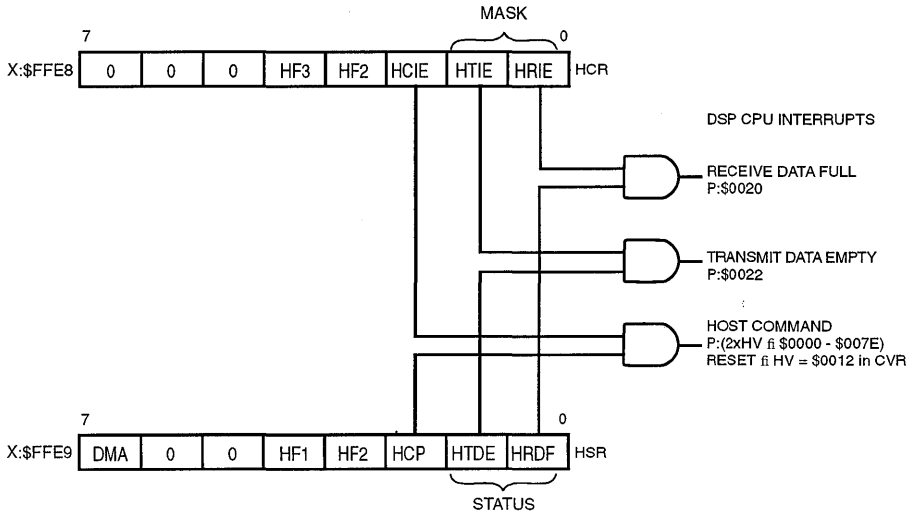


Figure 11-4. HSR–HCR Operation

the IMP to DSP host port connection logic that is built inside the MC68356. An address decoder is connected to the host port (HEN) and the host port 68000 bus address can be programmed in the host port base address register (HBAR). The IMP IDMA can also be used to access the host port registers and the HREQ signal and the HACK signal from the DSP can be optionally connected to the IMP's DREQ and DACK lines (by setting the HRE and HACK bits to one and HRC to 00 in the HCOR). If interrupt driven servicing of the host port is desired, the host port HREQ signal can be optionally connected to either of the IRQ1, IRQ6, or IRQ7 signals of the IMP. This can be done by setting the HRE and HACK bits to one and HRC to non-zero in the HCOR. The DSP host port may be accessed by the 68K, the IMP IDMA, or by any external master that arbitrates for the 68K bus. All external host port accesses conform to standard 68000 bus timings as shown in Section 14 Electrical Characteristics.

NOTE

In order to assure proper 68000 bus to host port access operation, the DSP clock (CKOUT) must be of equal or higher frequency than the IMP clock (CLKO).

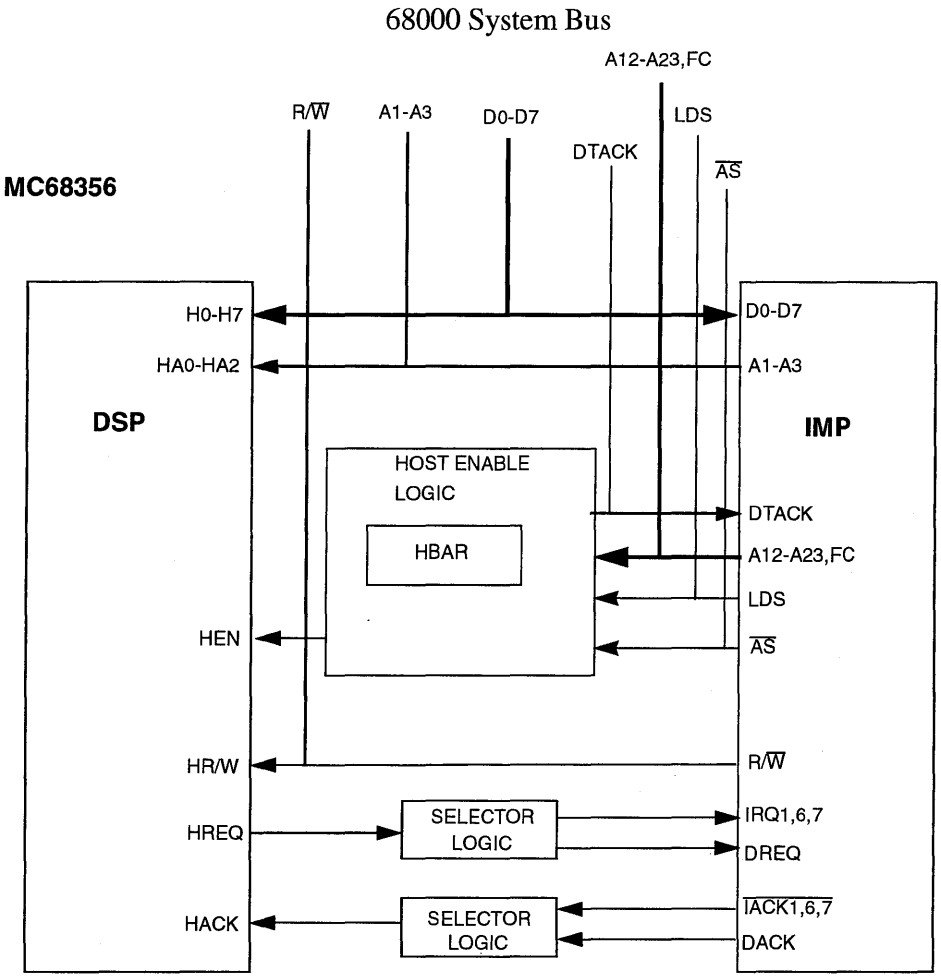


Figure 11-5. DSP Host Port Connection Block Diagram

11.2.1 DSP Host Port Configuration Option Register - HCOR

The DSP host port configuration option register (HCOR) is an 8-bit read/write register visible from the IMP side in 68000 space used to program the IMP to DSP host port access hardware connection options (see Figure 11-5). This register is used to program the host port access mechanism to either interrupt driven operation or DMA driven operation.

HCOR IMP address: BAR+$8EA

7	6	5	4	3	2	1	0
0	0	0	0	HACKE	HRE	HRC	
RESET: 0	0	0	0	0	0	0	0

Read/Write

HRE—Host Request Enable

0 = The host port request line ($\overline{\text{HREQ}}$) is not driven to the IMP.
1 = The host port request line ($\overline{\text{HREQ}}$) is connected to the IMP according to HRC programming.

HACKE—Host Acknowledge Enable

0 = The host port acknowledge line ($\overline{\text{HACK}}$) is connected to a pull-up resistor.
1 = The host port acknowledge line ($\overline{\text{HACK}}$) is connected to the appropriate interrupt acknowledge line or to the IDMA acknowledge line ($\overline{\text{DACK}}$) according to HRC programming.

HRC—Host Request Control

00 = If enabled by HRE, the host port request line ($\overline{\text{HREQ}}$) is connected to the IDMA request line ($\overline{\text{DREQ}}$). If enabled by HACKE, the host port acknowledge line ($\overline{\text{HACK}}$) is connected to the IDMA acknowledge line ($\overline{\text{DACK}}$).
01 = If enabled by HRE, the host port interrupt request line ($\overline{\text{HREQ}}$) is connected to the 68000 interrupt request level one. If enabled by HACKE, the host port acknowledge line ($\overline{\text{HACK}}$) is connected to the interrupt acknowledge line, level one ($\overline{\text{IACK1}}$).

NOTE

To receive the interrupt vector from the DSP'S host port, the user should set bit IV1 in the IMP's GIMR register to one, for external interrupt vector.

10 = If enabled by HRE, the host port interrupt request line ($\overline{\text{HREQ}}$) is connected to the 68000 interrupt request level six. If enabled by HACKE, the host port acknowledge line ($\overline{\text{HACK}}$) is connected to the interrupt acknowledge line level six ($\overline{\text{IACK6}}$).

NOTE

To receive the interrupt vector from the DSP'S host port the user should set bit IV6 in the IMP's GIMR register to one, for external interrupt vector.

11 = If enabled by HRE, the host port interrupt request line $\overline{\text{HREQ}}$ is connected to the 68000 interrupt request level seven. If enabled by HACKE, the host port acknowledge line $\overline{\text{HACK}}$ is connected to the interrupt acknowledge line level seven ($\overline{\text{IACK7}}$).

NOTE

To receive the interrupt vector from the DSP'S host port, the user should set bit IV7 in the IMP's GIMR register to one, for external interrupt vector.

11.2.2 DSP Host Port Base Address Register - HBAR

The DSP host port base address register is a 16-bit read/write register on the IMP side used to map the DSP host port in the 68000 address space. When a 68000 bus access is made into the programmed space, HEN will be asserted. The A3-A1 address lines will be connected directly to the host port address lines (HA2–HA0).

HBAR $IMP address:BAR+$8E8

15	14	13	12	11	10	9	8
BFC2–BFC0			BA23	BA22	BA21	BA20	BA19
RESET							
0	0	0	0	0	0	0	0

7	6	5	4	3	2	1	0
BA18	BA17	BA16	BA15	BA14	BA13	BA12	EN
RESET							
0	0	0	0	0	0	0	0

Read/Write

BFC2-0—Base Function Code Field

111 = The function codes are not compared.
000– These bits are used to set the address space function code for the host port
110 = accesses.

BA23-12—Base Address

These bits are used to set the starting address of the host side, host port registers.

EN—Enable

0 = The host port accesses are disabled. HEN line will not be asserted.
1 = The host port accesses are enabled. HEN line will be asserted when 68000 cycle high address lines match the programmed value.

11.2.3 Programming Model – Host Processor Viewpoint

The HI appears to the host processor as a memory-mapped peripheral occupying eight bytes on even addresses (beginning at the address pointed to by the HBAR register in the IMP) in the host processor address space (see Figure 11-6 and Figure 11-7). These registers can be viewed as one control register (ICR), one status register (ISR), three data registers (RXH/TXH, RXM/TXM, and RXL/TXL), and two vector registers (IVR and CVR). The CVR is a special command register that is used by the host processor to issue commands to the DSP. These registers can be accessed only by the host processor; they cannot be accessed by the DSP CPU. Host processors may use standard host processor instructions

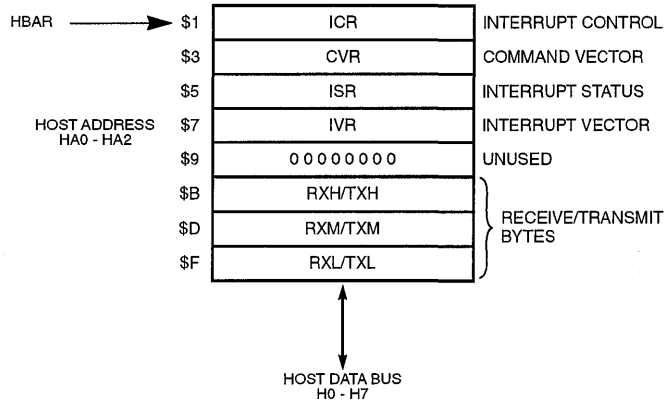


Figure 11-6. HI Register Map

(e.g., byte move) and addressing modes to communicate with the HI registers. The 68000 host processor can address the HI using the special MOVEP instruction for word (16-bit) or long-word (32-bit) transfers. The $\overline{\text{HREQ}}$ and $\overline{\text{HACK}}$ handshake flags are provided for polled or interrupt-driven data transfers with the host processor. Because the DSP interrupt response is sufficiently fast, the 68000 core should be able to load or store data at its maximum programmed I/O (non-DMA) instruction rate without testing the handshake flags for each transfer. If the full handshake is not needed, the host processor can treat the DSP as fast memory, and data can be transferred between the host processor and the DSP at the fastest host processor data rate. The IDMA hardware may be used with the handshake flags to transfer data without host processor intervention.

11

One of the most innovative features of the host interface is the host command feature. With this feature, the host processor can issue vectored exception requests to the DSP. The host may select any one of 64 DSP exception routines to be executed by writing a vector address register in the HI. This flexibility allows the host programmer to execute up to 64 preprogramming functions inside the DSP. For example, host exceptions can allow the host processor to read or write DSP registers (X, Y, or program memory locations), force exception handlers (e.g., SSI, SCI, $\overline{\text{IRQA}}$, $\overline{\text{IRQB}}$ exception routines), and perform control and debugging operations if exception routines are implemented in the DSP to perform these tasks.

11.2.3.1 INTERRUPT CONTROL REGISTER (ICR)

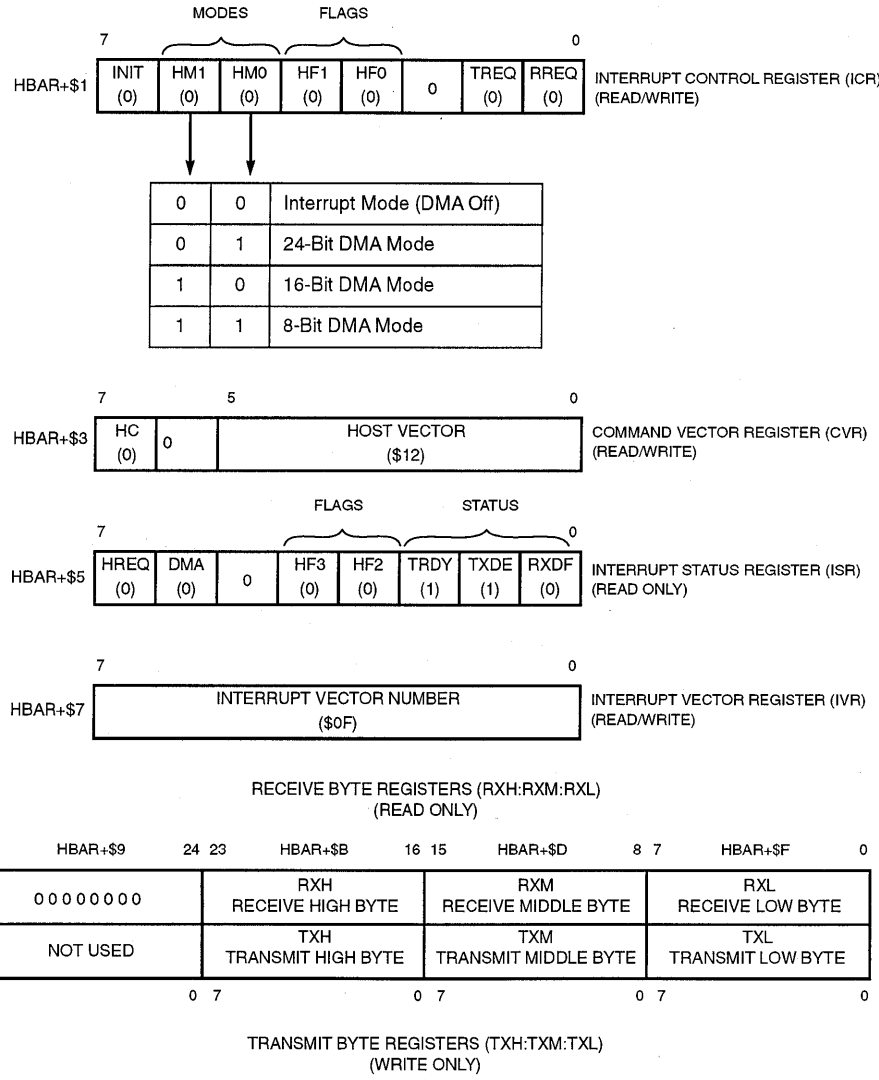
The ICR is an 8-bit read/write control register used by the host processor to control the HI interrupts and flags. ICR cannot be accessed by the DSP CPU. ICR is a read/write register, which allows the use of bit manipulation instructions on control register bits. The control bits are described in the following paragraphs.

11.2.3.1.1 ICR Receive Request Enable (RREQ) Bit 0

The RREQ bit is used to control the $\overline{\text{HREQ}}$ signal for host receive data transfers.

In interrupt mode (DMA off), RREQ is used to enable interrupt requests via the external host request ($\overline{\text{HREQ}}$) signal when the receive data register full (RXDF) status bit in the ISR is set.

When RREQ is cleared, RXDF interrupts are disabled. When RREQ is set, the external $\overline{\text{HREQ}}$ signal will be asserted if RXDF is set.



NOTE: The numbers in parentheses are reset values.

Figure 11-7. Host Processor Programming Model–Host Side

In DMA modes, RREQ must be set or cleared by software to select the direction of DMA transfers. Setting RREQ sets the direction of DMA transfer to be DSP to host and enables the $\overline{\text{HREQ}}$ signal to request data transfer. Hardware, software, individual, and STOP resets clear RREQ.

Table 11-2. $\overline{\text{HREQ}}$ Signal Definition

TREQ	RREQ	$\overline{\text{HREQ}}$ Signal
Interrupt Mode		
0	0	No Interrupts (Polling)
0	1	RXDF Request (Interrupt)
1	0	TXDE Request (Interrupt)
1	1	RXDF and TXDE Request (Interrupts)
DMA Mode		
0	0	No DMA
0	1	DSP to Host Request (RX)
1	0	Host to DSP Request (TX)
1	1	Undefined (Illegal)

11.2.3.1.2 ICR Transmit Request Enable (TREQ) Bit 1

The TREQ bit is used to control the $\overline{\text{HREQ}}$ signal for host transmit data transfers.

In interrupt mode (DMA off), TREQ is used to enable interrupt requests via the external $\overline{\text{HREQ}}$ signal when the transmit data register empty (TXDE) status bit in the ISR is set. When TREQ is cleared, TXDE interrupts are disabled. When TREQ is set, the external $\overline{\text{HREQ}}$ signal will be asserted if TXDE is set.

In DMA modes, TREQ must be set or cleared by software to select the direction of DMA transfers. Setting TREQ sets the direction of DMA transfer to be host to DSP and enables the $\overline{\text{HREQ}}$ signal to request data transfer. DSP hardware, software, individual, and STOP resets clear TREQ.

Table 11-2 summarizes the effect of RREQ and TREQ on the $\overline{\text{HREQ}}$ signal.

11.2.3.1.3 ICR Reserved Bit (Bit 2)

This bit, which is reserved and unused, reads as a logic zero.

11.2.3.1.4 ICR Host Flag 0 (HF0) Bit 3

The HF0 bit is used as a general-purpose flag for host-to-DSP communication. HF0 may be set or cleared by the host processor and cannot be changed by the DSP. HF0 is visible in the HSR on the DSP CPU side of the HI (see Figure 11-3). Hardware, software, individual, and STOP resets clear HF0.

11.2.3.1.5 ICR Host Flag 1 (HF1) Bit 4

The HF1 bit is used as a general-purpose flag for host-to-DSP communication. HF1 may be set or cleared by the host processor and cannot be changed by the DSP. Hardware, software, individual, and STOP resets clear HF1.

11.2.3.1.6 ICR Host Mode Control (HM1 and HM0 bits) Bits 5 and 6

The HM0 and HM1 bits select the transfer mode of the HI (see Table 11-3). HM1 and HM0 enable the DMA mode of operation or interrupt (non-DMA) mode of operation.

Table 11-3. Host Mode Bit Definition

HM1	HM0	Mode
0	0	Interrupt Mode (DMA Off)
0	1	DMA Mode (24 Bit)
1	0	DMA Mode (16 Bit)
1	1	DMA Mode (8 Bit)

When both HM1 and HM0 are cleared, the DMA mode is disabled, and the TREQ and RREQ control bits are used for host processor interrupt control via the external $\overline{\text{HREQ}}$ output signal. Also, in the non-DMA mode, the $\overline{\text{HACK}}$ input signal is used for the 68000 vectored interrupt acknowledge input.

When HM1 or HM0 are set, the DMA mode is enabled, and the $\overline{\text{HREQ}}$ signal is used to request DMA transfers. When the DMA mode is enabled, the TREQ and RREQ bits select the direction of DMA transfers. The $\overline{\text{HACK}}$ input signal is used as a DMA transfer acknowledge input. If the DMA direction is from DSP to host, the contents of the selected register are enabled onto the host data bus when $\overline{\text{HACK}}$ is asserted. If the DMA direction is from host to DSP, the selected register is written from the host data bus when $\overline{\text{HACK}}$ is asserted.

The size of the DMA word to be transferred is determined by the DMA control bits, HM0 and HM1. The HI register selected during a DMA transfer is determined by a 2-bit address counter, which is preloaded with the value in HM1 and HM0. The address counter substitutes for the HA1 and HA0 bits of the HI during a DMA transfer. The host address bit (HA2) is forced to one during each DMA transfer. The address counter can be initialized with the INIT bit feature. After each DMA transfer on the host data bus, the address counter is incremented to the next register. When the address counter reaches the highest register (RXL or TXL), the address counter is not incremented but is loaded with the value in HM1 and HM0. This allows 8-, 16- or 24-bit data to be transferred in a circular fashion and eliminates the need for the DMA controller to supply the HA2, HA1, and HA0 signals. For 16- or 24-bit data transfers, the DSP CPU interrupt rate is reduced by a factor of 2 or 3, respectively, from the host request rate – i.e., for every two or three host processor data transfers of one byte each, there is only one 24-bit DSP CPU interrupt.

Hardware, software, individual, and STOP resets clear HM1 and HM0.

Table 11-4. $\overline{\text{HREQ}}$ Signal Definition

TREQ	RREQ	After INIT Execution	Transfer Direction Initialized
Interrupt Mode (HM1 = 0, HM0 = 0) INIT Execution			
0	0	INIT = 0; Address Counter = 00	None
0	1	INIT = 0; RXDF = 0; HTDE = 1; Address Counter = 00	DSP to Host
1	0	INIT = 0; TXDE = 1; HRDF = 0; Address Counter = 00	Host to DSP
1	1	INIT = 0; RXDF = 0; HTDE = 1; TXDE = 1; HRDF = 0; Address Counter = 00	Host to/from DSP
DMA Mode (HM1 or HM0 = 1) INIT Execution			
0	0	INIT = 0; Address Counter = HM1, HM0	None
0	1	INIT = 0; RXDF = 0; HTDE = 1; Address Counter = HM1, HM0	DSP to Host
1	0	INIT = 0; TXDE = 1; HRDF = 0; Address Counter = HM1, HM0	Host to DSP
1	1	Undefined (Illegal)	Undefined

11.2.3.1.7 ICR Initialize Bit (INIT) Bit 7

The INIT bit is used by the host processor to force initialization of the HI hardware. Initialization consists of configuring the HI transmit and receive control bits and loading HM1 and HM0 into the internal DMA address counter. Loading HM1 and HM0 into the DMA address counter causes the HI to begin transferring data on a word boundary rather than transferring only part of the first data word.

There are two methods of initialization: 1) allowing the DMA address counter to be automatically set after transferring a word, and 2) setting the INIT bit, which sets the DMA address counter. Using the INIT bit to initialize the HI hardware may or may not be necessary, depending on the software design of the interface.

The type of initialization done when the INIT bit is set depends on the state of TREQ and RREQ in the HI. The INIT command, which is local to the HI, is designed to conveniently configure the HI into the desired data transfer mode. The commands are described in the following paragraphs and in Table 11-4. The host sets the INIT bit, which causes the HI hardware to execute the INIT command. The interface hardware clears the INIT bit when the command has been executed. DSP hardware, software, individual, and STOP resets clear INIT.

INIT execution always loads the DMA address counter and clears the channel according to TREQ and RREQ. INIT execution is not affected by HM1 and HM0.

The internal DMA counter is incremented with each DMA transfer (each $\overline{\text{HACK}}$ pulse) until it reaches the last data register (RXL or TXL). When the DMA transfer is completed, the counter is loaded with the value of the HM1 and HM0 bits. When changing the size of the DMA word (changing HM0 and HM1 in the ICR), the DMA counter is not automatically updated, and, as a result, the DMA counter will point to the wrong data register immediately after HM1 and HM0 are changed. The INIT function must be used to preset the internal DMA counter correctly. Always set INIT after changing HM0 and HM1. However, the DMA counter cannot be initialized in the middle of a DMA transfer. Even though the INIT bit is set, the internal DMA controller will wait until after completing the data transfer in progress before executing the initialization.

11.2.3.2 COMMAND VECTOR REGISTER (CVR)

The host processor uses the CVR to cause the DSP to execute a vectored interrupt. The host command feature is independent of the data transfer mechanisms in the HI. It can be used to cause any of the 64 possible interrupt routines in the DSP CPU to be executed. The command vector register is shown in Figure 11-8.

11.2.3.2.1 CVR Host Vector (HV) Bits 0–5

The six HV bits select the host command exception address to be used by the host command exception logic. When the host command exception is recognized by the DSP interrupt control logic, the starting address of the exception taken is 2×HV. The host can write HC and HV in the same write cycle, if desired.

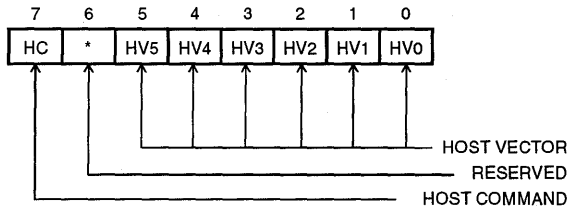


Figure 11-8. Command Vector Register

The host processor can select any of the 64 possible exception routine starting addresses in the DSP by writing the exception routine starting address divided by 2 into HV. This means that the host processor can force any of the existing exception handlers (SSI, SCI, IRQA, IRQB, etc.) and can use any of the reserved or otherwise unused starting addresses provided they have been preprogrammed in the DSP. HV is set to $12 (vector location $0024) by hardware, software, individual, and STOP resets. Vector location $0024 is the first of 45 special host command vectors.

NOTE

The HV should not be used with a value of zero because the reset location is normally programmed with a JMP instruction. Doing so will cause an improper fast interrupt.

11.2.3.2.2 CVR Reserved Bit (Bit 6)

Reserved bit which is unused and read by the host processor as zero.

11.2.3.2.3 CVR Host Command Bit (HC) Bit 7

The HC bit is used by the host processor to handshake the execution of host command exceptions. Normally, the host processor sets HC=1 to request the host command exception from the DSP. When the host command exception is acknowledged by the DSP, the HC bit is cleared by the HI hardware. The host processor can read the state of HC to determine when the host command has been accepted. The host processor may elect to clear the HC bit, canceling the host command exception request at any time before it is accepted by the DSP CPU.

NOTE

The command exception might be recognized by the DSP and executed before it can be canceled by the host, even if the host clears the HC bit.

Setting HC causes host command pending (HCP) to be set in the HSR. The host can write HC and HV in the same write cycle if desired. DSP hardware, software, individual, and STOP resets clear HC.

11.2.3.3 INTERRUPT STATUS REGISTER (ISR)

The ISR is an 8-bit read-only status register used by the host processor to interrogate the status and flags of the HI. The host processor can write this address without affecting the internal state of the HI, which is useful if the user desires to access all of the HI registers by stepping through the HI addresses. The ISR cannot be accessed by the DSP. The status bits are described in the following paragraphs.

11.2.3.3.1 ISR Receive Data Register Full (RXDF) Bit 0

The RXDF bit indicates that the receive byte registers (RXH, RXM, and RXL) contain data from the DSP CPU and may be read by the host processor. RXDF is set when the HTX is transferred to the receive byte registers. RXDF is cleared when the receive data low (RXL) register is read by the host processor. RXL is normally the last byte of the receive byte registers to be read by the host processor. RXDF can be cleared by the host processor using the initialize function. RXDF may be used to assert the $\overline{\text{HREQ}}$ signal if the RREQ bit is set. Regardless of whether the RXDF interrupt is enabled, RXDF provides valid status so that polling techniques may be used by the host processor. DSP hardware, software, individual, and STOP resets clear RXDF.

11.2.3.3.2 ISR Transmit Data Register Empty (TXDE) Bit 1

The TXDE bit indicates that the transmit byte registers (TXH, TXM, and TXL) are empty and can be written by the host processor. TXDE is set when the transmit byte registers are transferred to the HRX register. TXDE is cleared when the transmit byte low (TXL) register is written by the host processor. TXL is normally the last byte of the transmit byte registers to be written by the host processor. TXDE can be set by the host processor using the initialize feature. TXDE may be used to assert the $\overline{\text{HREQ}}$ signal if the TREQ bit is set. Regardless of whether the TXDE interrupt is enabled, TXDE provides valid status so that polling tech-

niques may be used by the host processor. DSP hardware, software, individual, and STOP resets set TXDE.

11.2.3.3.3 ISR Transmitter Ready (TRDY) Bit 2

The TRDY status bit indicates that **both** the TXH, TXM, TXL and the HRX registers are empty.

TRDY=TXDE • $\overline{\text{HRDF}}$

When TRDY is set to one, the data that the host processor writes to TXH,TXM, and TXL will be immediately transferred to the DSP CPU side of the HI. This has many applications. For example, if the host processor issues a host command which causes the DSP CPU to read the HRX, the host processor can be guaranteed that the data it just transferred to the HI is what is being received by the DSP CPU.

DSP hardware, software, individual, and STOP resets set TRDY.

11.2.3.3.4 ISR Host Flag 2 (HF2) Bit 3

The HF2 bit in the ISR indicates the state of host flag 2 in the HCR on the CPU side. HF2 can only be changed by the DSP (see Figure 11-3). HF2 is cleared by a DSP hardware or software reset.

11.2.3.3.5 ISR Host Flag 3 (HF3) Bit 4

The HF3 bit in the ISR indicates the state of host flag 3 in the HCR on the CPU side. HF3 can only be changed by the DSP (see Figure 11-3). HF3 is cleared by a DSP hardware or software reset.

11.2.3.3.6 ISR Reserved Bit (Bit 5)

This status bit is reserved for future expansion and will read as zero during host processor read operations.

11.2.3.3.7 ISR DMA Status (DMA) Bit 6

The DMA status bit indicates that the host processor has enabled the DMA mode of the HI (HM1 or HM0=1). When the DMA status bit is clear, it indicates that the DMA mode is disabled (HM0=HM1=0) and no DMA operations are pending. When DMA is set, it indicates that the DMA mode is enabled and the host processor should not use the active DMA channel (RXH, RXM, RXL or TXH, TXM, TXL depending on DMA direction) to avoid conflicts with the DMA data transfers. The channel not in use can be used for polled operation by the host and operates in the interrupt mode for internal DSP exceptions or polling. DSP hardware, software, individual, and STOP resets clear the DMA status bit.

11.2.3.3.8 ISR Host Request (HREQ) Bit 7

The HREQ bit indicates the status of the host request output signal ($\overline{\text{HREQ}}$). When the HREQ status bit is cleared, it indicates that the $\overline{\text{HREQ}}$ signal is deasserted and no host processor interrupts or DMA transfers are being requested. When the HREQ status bit is set, it indicates that the $\overline{\text{HREQ}}$ signal is asserted, indicating that the DSP is interrupting the host processor or that a DMA transfer request is occurring. The HREQ interrupt request may orig-

inate from either or both of two sources – the receive byte registers are full or the transmit byte registers are empty. These conditions are indicated by the ISR RXDF and TXDE status bits, respectively. If the interrupt source has been enabled by the associated request enable bit in the ICR, HREQ will be set if one or more of the two enabled interrupt sources is set. DSP hardware, software, individual, and STOP resets clear HREQ.

11.2.3.4 INTERRUPT VECTOR REGISTER (IVR)

The IVR is an 8-bit read/write register which typically contains the exception vector number used with 68000 vectored interrupts. Only the host processor can read and write this register. The contents of IVR are placed on the host data bus (H0–H7) when both the $\overline{\text{HREQ}}$ and $\overline{\text{HACK}}$ signals are asserted and the DMA mode is disabled. The contents of this register are initialized to $0F by a DSP hardware or software reset, which corresponds to the uninitialized exception vector in the 68000.

11.2.3.5 RECEIVE BYTE REGISTERS (RXH, RXM, RXL)

The receive byte registers are viewed as three 8-bit read-only registers by the host processor. These registers are called receive high (RXH), receive middle (RXM), and receive low (RXL). These three registers receive data from the high byte, middle byte, and low byte, respectively, of the HTX register and are selected by three external host address inputs (HA2, HA1, and HA0) during a host processor read operation or by an on-chip address counter in DMA operations. The receive byte registers (at least RXL) contain valid data when the receive data register full (RXDF) bit is set. The host processor may program the RREQ bit to assert the $\overline{\text{HREQ}}$ signal when RXDF is set. This informs the host processor or DMA controller that the receive byte registers are full. These registers may be read in any order to transfer 8-, 16-, or 24-bit data. However, reading RXL clears the receive data full RXDF bit. Because reading RXL clears the RXDF status bit, it is normally the last register read during a 16- or 24-bit data transfer. Reset does not affect RXH, RXM, or RXL.

11.2.3.6 TRANSMIT BYTE REGISTERS (TXH, TXM, TXL)

The transmit byte registers are viewed as three 8-bit write-only registers by the host processor. These registers are called transmit high (TXH), transmit middle (TXM), and transmit low (TXL). These three registers send data to the high byte, middle byte, and low byte, respectively, of the HRX register and are selected by three external host address inputs (HA2, HA1, and HA0) during a host processor write operation. Data may be written into the transmit byte registers when the transmit data register empty (TXDE) bit is set. The host processor may program the TREQ bit to assert the HREQ signal when TXDE is set. This informs the host processor or DMA controller that the transmit byte registers are empty. These registers may be written in any order to transfer 8-, 16-, or 24-bit data. However, writing TXL clears the TXDE bit. Because writing the TXL register clears the TXDE status bit, TXL is normally the last register written during a 16- or 24-bit data transfer. The transmit byte registers are transferred as 24-bit data to the HRX register when both TXDE and the HRDF bit are cleared. This transfer operation sets TXDE and HRDF. Reset does not affect TXH, TXM, or TXL.

11.2.3.7 REGISTERS AFTER RESET

Table 11-5 shows the result of four kinds of reset on bits in each of the HI registers seen by the host processor. The hardware reset is caused by asserting the $\overline{\text{DRESET}}$ pin; the software reset is caused by executing the RESET instruction; the individual reset is caused by

clearing the PBC register bit 0; and the stop reset is caused by executing the STOP instruction.

11.2.4 Servicing the Host Interface

The HI can be serviced by using one of the following protocols:

1. Polling, or
2. Interrupts, which can be either
   a. non-DMA or
   b. DMA.

Table 11-5. Host Registers after DSP Reset (Host Side)

Register Name	Register Data	Reset Type			
		HW Reset	SW Reset	IR Reset	ST Reset
ICR	INIT	0	0	0	0
	HM (1 - 0)	0	0	0	0
	TREQ	0	0	0	0
	RREQ	0	0	0	0
	HF (1 - 0)	0	0	0	0
CVR	HC	0	0	0	0
	HV (5 - 0)	$12	$12	$12	$12
ISR	HREQ	0	0	0	0
	DMA	0	0	0	0
	HF (3 - 2)	0	0	—	—
	TRDY	1	1	1	1
	TXDE	1	1	1	1
	RXDF	0	0	0	0
IVR	IV (7 - 0)	$0F	$0F	—	—
RX	RXH (23 - 16)	—	—	—	—
	RXM (15 - 8)	—	—	—	—
	RXL (7 - 0)	—	—	—	—
TX	TXH (23 - 21)	—	—	—	—
	TXM (15 - 8)	—	—	—	—
	TXL (7 - 0)	—	—	—	—

From the host processor viewpoint, the service consists of making a data transfer since this is the only way to reset the appropriate status bits.

11.2.4.1 HI HOST PROCESSOR DATA TRANSFER

The HI looks like a static RAM location on the 68000 bus. Accordingly, in order to transfer data with the HI, the 68000:

1. Asserts the HI address (HA0, HA1, HA2) to select the register to be read or written. HA0-HA2 are connected as follows:

   HA0 is connected to A1 on the 68000 address bus.

   HA1 is connected to A2 on the 68000 address bus.

   HA2 is connected to A3 on the 68000 address bus.

   NOTE

   Host side, host port registers are aligned on even address boundaries in the host (68000) memory space.

2. Asserts R/$\overline{W}$ which is connected to HR/$\overline{W}$, to select the direction of the data transfer;
3. Strobes the data transfer using $\overline{HEN}$ which is connected to the host enable logic which provides the proper timing to access the host interface. All host port accesses are standard IMP internal register accesses.

The specified timing relationships for IMP internal memory location accesses are given in Section 14 Electrical Characteristics.

11.2.4.2 HI INTERRUPTS HOST REQUEST ($\overline{HREQ}$)

The host processor interrupts can be generated to the IMP IRQ inputs using internal selector logic. $\overline{HREQ}$ can optionally be connected to one of the IMP IRQ signals (1, 6, or 7) by setting the HRE bit in the HCOR and setting the HRC bits appropriately. The 68000 acknowledges host interrupts by executing an interrupt service routine. The most significant bit (HREQ) of the host port ISR register may be tested by the host processor to determine if the DSP is the interrupting device and the two least significant bits (RXDF and TXDE) may be tested to determine the interrupt source (see Figure 11-9). The host processor interrupt service routine must read or write the appropriate HI register to clear the interrupt. $\overline{HREQ}$ is deasserted when 1) the enabled request is cleared or masked, 2) DMA $\overline{HACK}$ is asserted, or 3) the DSP is reset.

11.2.4.3 POLLING

In the polling mode of operation, the $\overline{HREQ}$ should not be connected to the host processor and $\overline{HACK}$ must be deasserted to ensure DMA data or IVR data is not being output on H0-H7 when other registers are being polled. This is done by setting both the HRE and HACKE bits in the HCOR to zero.

The host processor first performs a data read transfer to read the ISR (see Figure 11-9) to determine, whether:

1. RXDF=1, signifying the receive data register is full and therefore a data read should

be performed

2. TXDE=1, signifying the transmit data register is empty so that a data write can be performed
3. TRDY=1, signifying the transmit data register is empty and that the receive data register on the DSP CPU side is also empty so that the data written by the host processor will be transferred directly to the DSP side
4. HF2 • HF3 ≠ 0, signifying an application-specific state within the DSP CPU has been reached, which requires action on the part of the host processor

Generally, after the appropriate data transfer has been made, the corresponding status bit will toggle.

If the host processor has issued a command to the DSP by writing the CVR and setting the HC bit, it can read the HC bit in the CVR to determine when the command has been accepted by the interrupt controller in the DSP's central processing module. When the command has been accepted for execution, the interrupt controller will reset the HC bit.

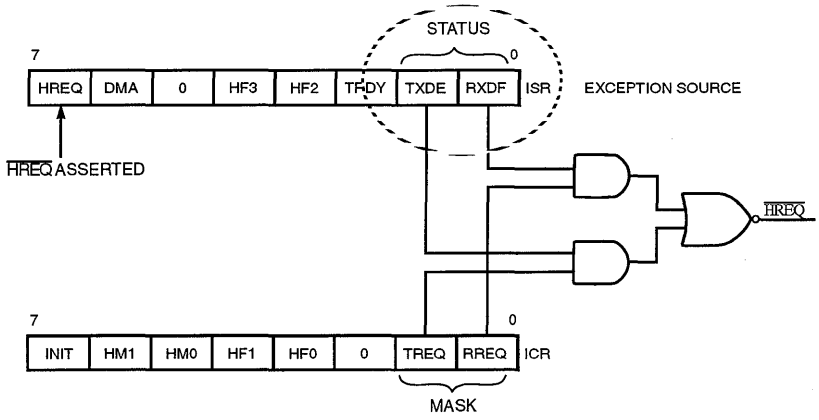


Figure 11-9. HI Interrupt Structure

11.2.4.4 SERVICING NON-DMA INTERRUPTS

When HM0=HM1=0 (non-DMA) and $\overline{\text{HREQ}}$ is connected to the host processor interrupt input, the HI can request service from the host processor by asserting HREQ. HREQ can be connected internally to one of the IMP IRQ inputs (1, 6, or 7) by setting the HRE bit in the HCOR and setting the HRC bits appropriately. In the non-DMA mode, $\overline{\text{HREQ}}$ will be asserted when TXDE=1 and/or RXDF=1 and the corresponding mask bit (TREQ or RREQ, respectively) is set. This is depicted in Figure 11-9.

Generally, servicing the interrupt starts with reading the ISR, as described in the previous section on polling, to determine if the DSP has generated the interrupt and why. When multiple DSPs occur in a system, the HREQ bit in the ISR will normally be read first to determine the interrupting device. The host processor interrupt service routine must read or write the

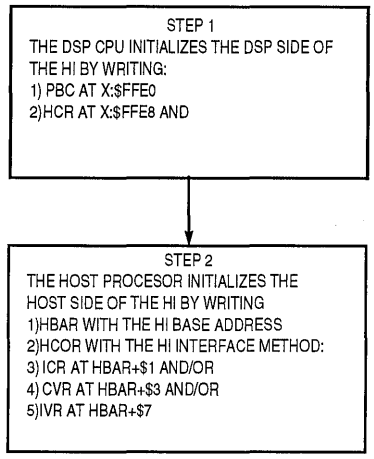


Figure 11-10. HI Initialization Flowchart

appropriate HI register to clear the interrupt. $\overline{\text{HREQ}}$ is deasserted when the enabled request is cleared or masked.

Servicing the interrupt will start by asserting $\overline{\text{HREQ}}$ to interrupt the processor (see Figure 11-9). The host processor then acknowledges the interrupt by asserting $\overline{\text{HACK}}$. While $\overline{\text{HREQ}}$ and $\overline{\text{HACK}}$ are simultaneously asserted, the contents of the IVR are placed on the host data bus. This vector will tell the host processor which routine to use to service the $\overline{\text{HREQ}}$ interrupt.

11

11.2.4.5 SERVICING DMA INTERRUPTS

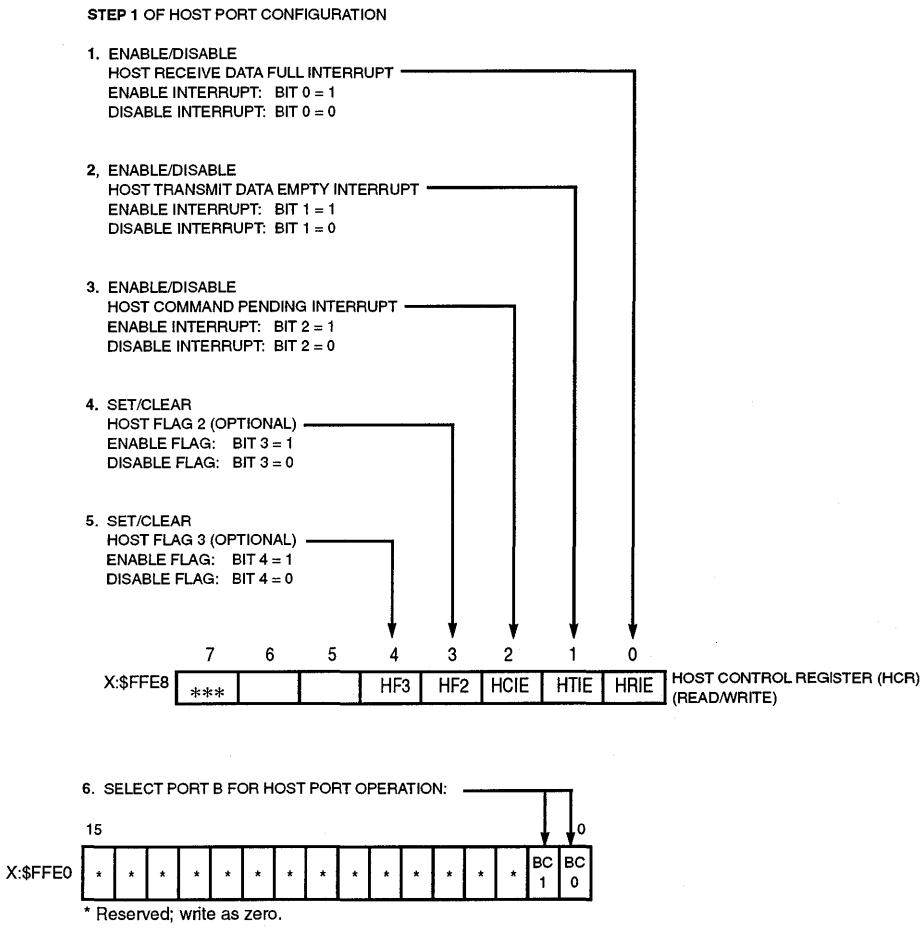
When HM0≠0 and/or HM1≠0, $\overline{\text{HREQ}}$ will be asserted to request a DMA transfer. The $\overline{\text{HREQ}}$ should be connected to the $\overline{\text{DREQ}}$ input of the IMP IDMA controller by setting up the HCOR register appropriately. The HA0-2, $\overline{\text{HEN}}$, and HR/$\overline{\text{W}}$ signals are not used during DMA transfers; DMA transfers only use the $\overline{\text{HREQ}}$ and $\overline{\text{HACK}}$ signals after the DMA channel has been initialized. $\overline{\text{HACK}}$ is used to strobe the data transfer. DMA transfers to and from the HI are considered in more detail in 11.2.5 HI Application Examples.

11.2.5 HI Application Examples

The following paragraphs describe examples of initializing the HI, transferring data with the HI, bootstrapping via the HI, and performing DMA transfers through the HI.

11.2.5.1 HI INITIALIZATION

Initializing the HI takes two steps (see Figure 11-10). The first step is to initialize the DSP side of the HI, which requires that the options for interrupts and flags be selected and then the HI be selected (see Figure 11-11). The second step is for the host processor to clear the HC bit by writing the CVR, select the data transfer method - polling, interrupts, or DMA (see Figure 11-15 and Figure 11-16), and write the IVR if it is desired to have the host port provide the interrupt vector. Figure 11-10 through Figure 11-18 provide a general description of how to initialize the HI. Later paragraphs in this section provide more detailed descriptions for



NOTE: The host flags are general-purpose semaphores. They are not required for host port operation but may be used in some applications.

Figure 11-11. HI Initialization–DSP Side

specific examples.These subsections include some code fragments illustrating how to initialize and transfer data using the HI.

11.2.5.2 POLLING/INTERRUPT CONTROLLED DATA TRANSFER

Handshake flags are provided for polled or interrupt-driven data transfers. Because the DSP interrupt response is sufficiently fast, the host should be able to load and store data at the maximum programmed I/O (non-DMA) instruction rate without testing the handshake flags for each transfer. If the full handshake is not needed, the host processor can treat the DSP as fast memory, and data can be transferred between the host and DSP at the fastest host processor rate. The IDMA may be used to transfer data at the maximum DSP interrupt rate.

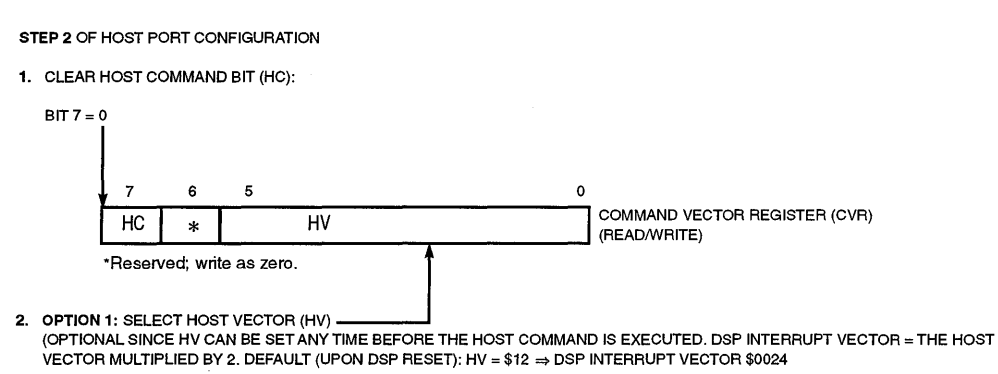


Figure 11-12. HI Configuration–Host Side

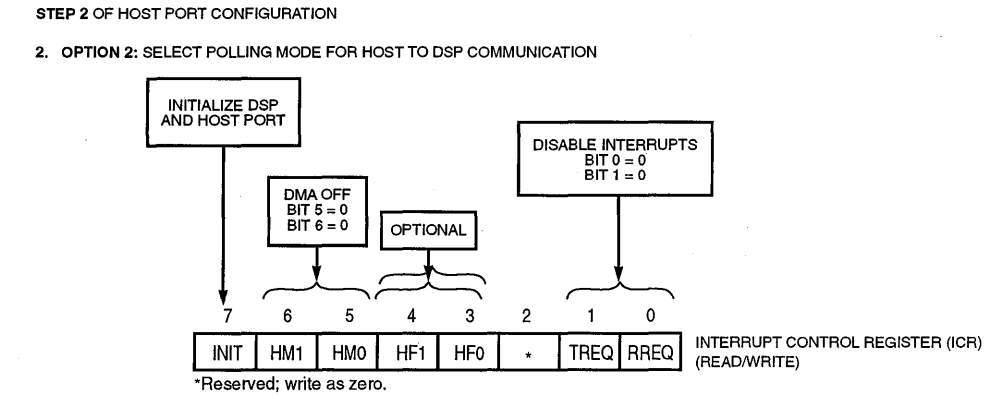
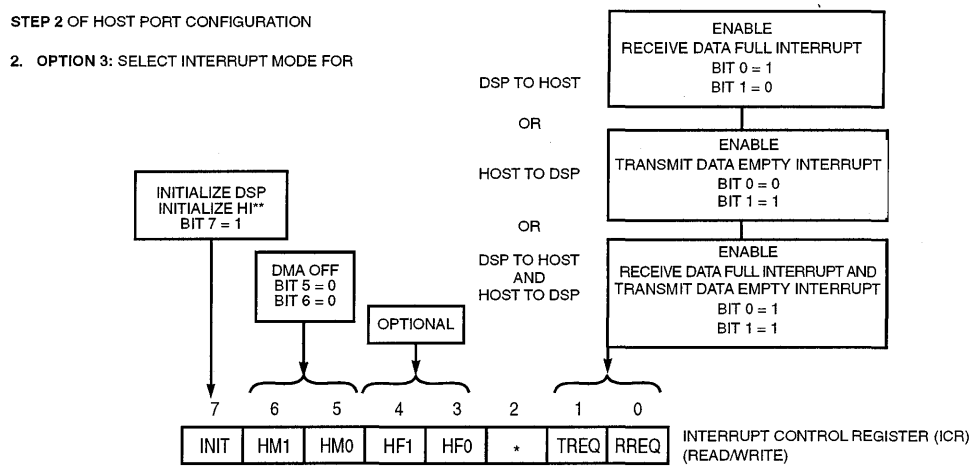


Figure 11-13. HI Initialization–Host Side, Polling

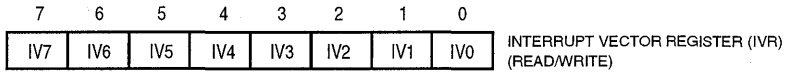
11.2.5.2.1 Host to DSP - Data Transfer

Figure 11-16 shows the bits in the ISR and ICR registers used by the host processor and the bits in the HSR and HCR registers used by the DSP to transfer data from the host processor to the DSP. The registers shown are the status register and control register as they are seen by the host processor, and the status register and control register as they are seen by the DSP. Only the registers used to transmit data from the host processor to the DSP are described. Figure 11-17 illustrates the process of that data transfer. The steps in Figure 11-17 can be summarized as follows:

1. When the TXDE bit in the ISR is set, it indicates that the HI is ready to receive a data byte from the host processor because the transmit byte registers (TXH, TXM, TXL) are empty.
2. The host processor can poll as shown in this step.



2. **OPTION 4:** LOAD HOST INTERRUPT VECTOR IF USING THE INTERRUPT MODE AND THE HOST PROCESSOR REQUIRES AN INTERRUPT VECTOR.



*Reserved; write as zero.

**See Figure 10 - 23.

Figure 11-14. HI Initialization–Host Side, Interrupt Mode

3. Alternatively, the host processor can use interrupts to determine the status of this bit. The HCOR register must be programmed for interrupt driven operation. Setting the TREQ bit in the ICR causes the $\overline{\text{HREQ}}$ to interrupt the host processor when TXDE is set.
4. Once the TXDE bit is set, the host can write data to the HI. It does this by writing three bytes to TXH, TXM, and TXL, respectively, or two bytes to TXM and TXL, respectively, or one byte to TXL.
5. Writing data to TXL clears TXDE in the ISR.
6. From the DSP's viewpoint, the HRDF bit (when set) in the HSR indicates that data is waiting in the HI for the DSP.
7. When the DSP reads the HRX, the HRDF bit is automatically cleared and TXDE in the ISR is set.
8. When TXDE=0 and HRDF=0, data is automatically transferred from TBR to HRX which sets HRDF.
9. The DSP can poll HRDF to see when data has arrived, or it can use interrupts.
10. If HRIE (in the HCR) and HRDF are set, exception processing is started using interrupt vector P:$0020.

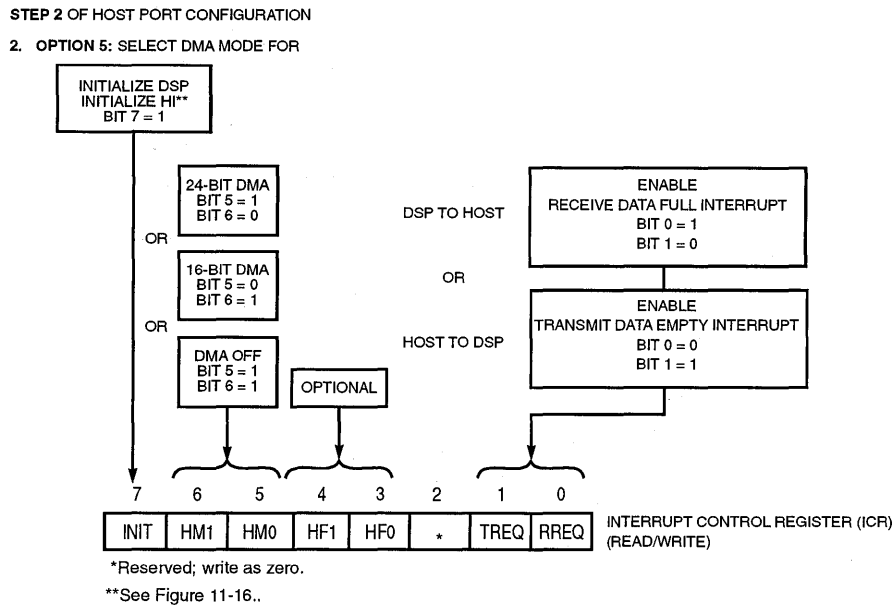


Figure 11-15. (HI Initialization–Host Side, DMA Mode

In the code shown in Figure 11-23, the MAIN PROGRAM initializes the HI and then hangs in a wait loop while it allows interrupts to transfer data from the host processor to the DSP. The first three MOVEP instructions enable the HI and configure the interrupts. The following MOVE enables the interrupts (this should always be done after the interrupt programs and hardware are completely initialized) and prepares the DSP CPU to look for the host flag, HF0=1. The JCLR instruction is a polling loop that looks for HF0=1, which indicates that the host processor is ready. When the host processor is ready to transfer data to the DSP, the DSP enables HRIE in the HCR, which allows the interrupt routine to receive data from the host processor. The jump-to-self instruction that follows is for test purposes only, it can be replaced by any other code in normal operation.

The receive routine in Figure 11-22 was implemented as a long interrupt (the instruction at the interrupt vector location, which is not shown, is a JSR). Since there is only one instruction, this could have been implemented as a fast interrupt. The MOVEP instruction moves data from the HI to a buffer area in memory and increments the buffer pointer so that the next word received will be put in the next sequential location.

11.2.5.2.2 Host to DSP – Command Vector

The host processor can cause three types of interrupts in the DSP (see Figure 11-19). These are host receive data (P:$0020), host transmit data (P:$0022), and host command (P:$0024 - P:$007E). The host command (HC) can be used to control the DSP by forcing it to execute any of 45 subroutines that can be used to run tests, transfer data, process data, etc. In addition, the HC can cause any of the other 19 interrupt routines in the DSP to be executed.

The process to execute a HC (see Figure 11-20) is as follows:

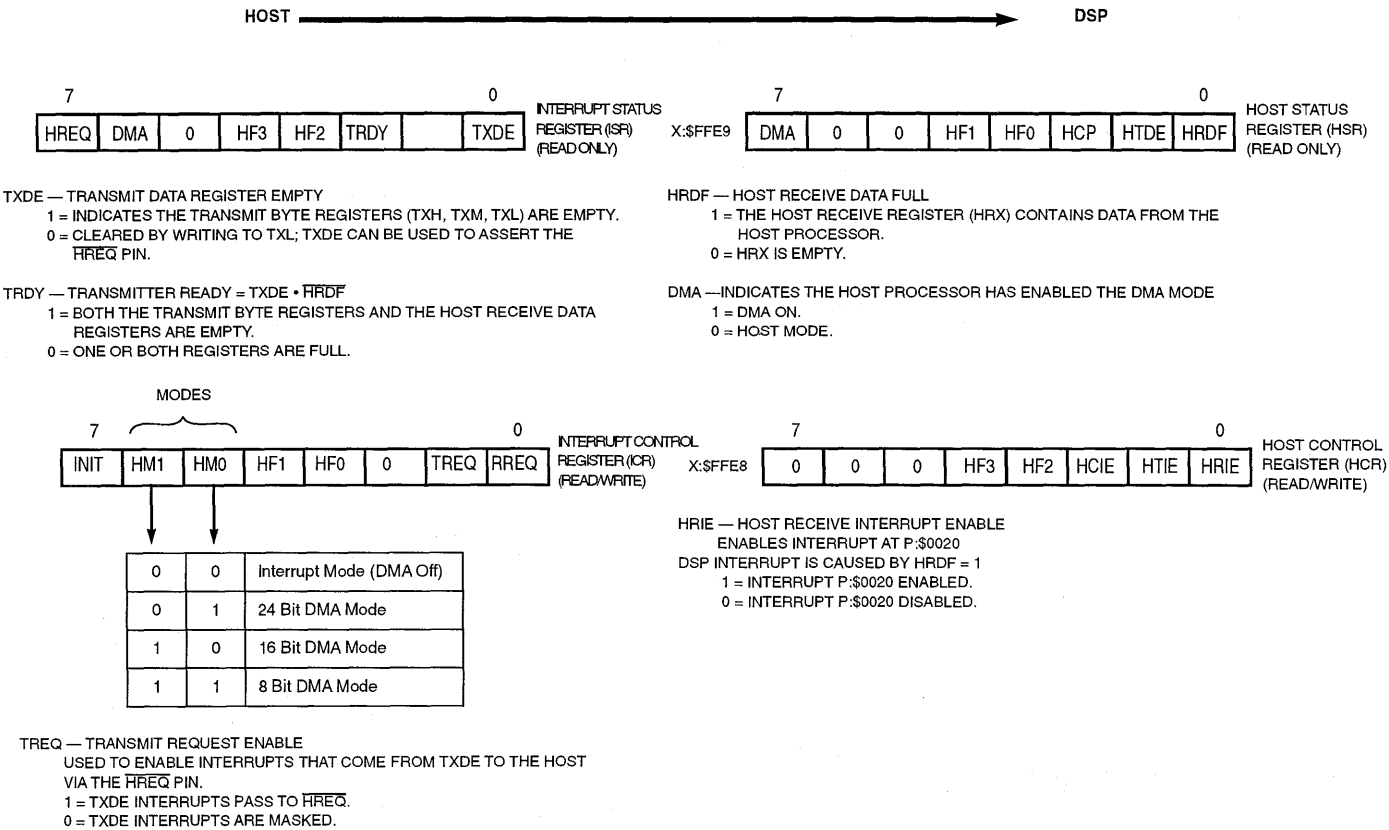


Figure 11-16. Bits Used for Host-to-DSP Transfer

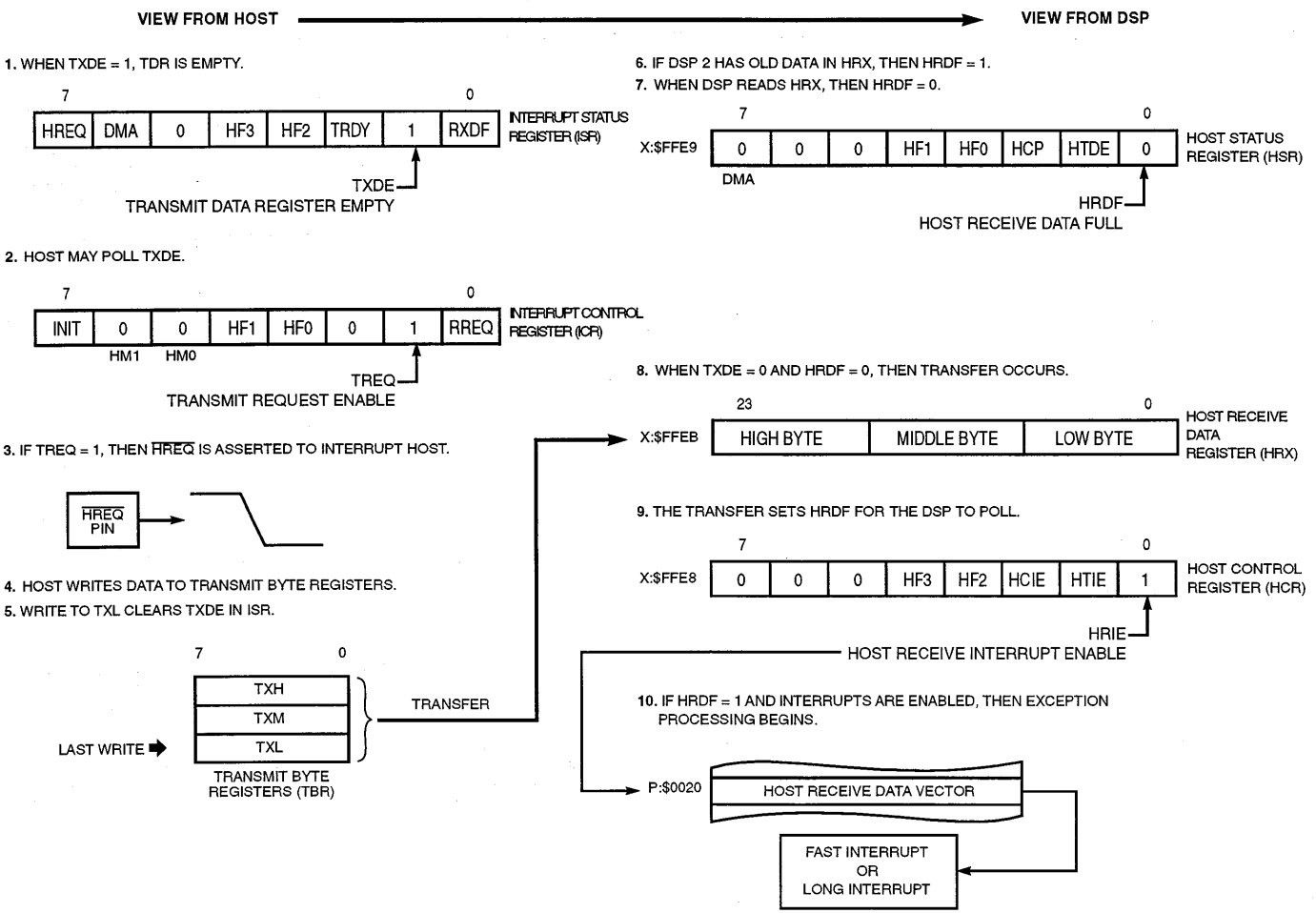
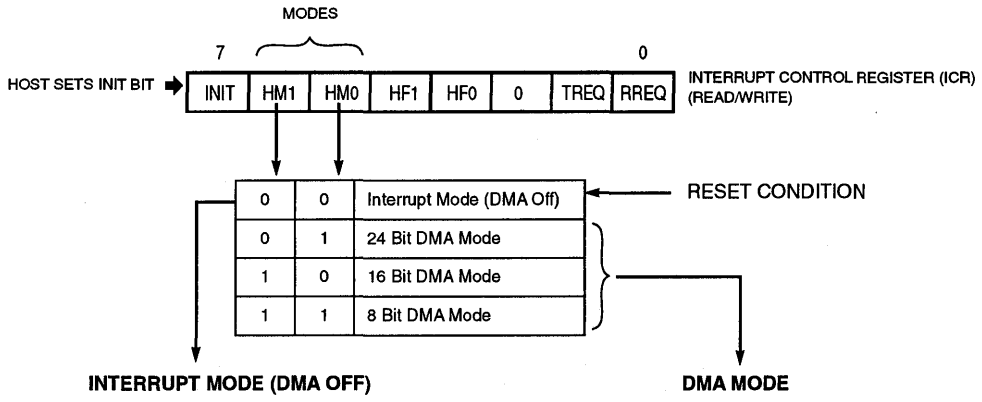


Figure 11-17. Data Transfer from Host to DSP



TREQ	RREQ	INIT Execution
0	0	INIT = 0; Address Counter = 00
0	1	INIT = 0; RXDF = 0; HTDE = 1; Address Counter = 00
1	0	INIT = 0; TXDE = 1; HRDF = 0; Address Counter = 00
1	1	INIT = 0; RXDF = 0; HTDE = 1: TXDE = 1; HRDF = 0; Address Counter = 00

TREQ	RREQ	INIT Execution
0	0	INIT = 0; Address Counter = HM1, HM0
0	1	INIT = 0; RXDF = 0; HTDE = 1; Address Counter = HM1, HM0
1	0	INIT = 0; TXDE = 1; HRDF = 0; Address Counter = HM1, HM0
1	1	Undefined (Illegal)

INIT is used by the HOST to force initialization of the HI hardware.
The HI hardware automatically clears INIT when the command is executed.
INIT is cleared by DSP RESET.

Figure 11-18. Host Mode and INIT Bits

1. The host processor writes the CVR with the desired HV (the HV is the DSP's interrupt vector (IV) location divided by two - i.e. if HV=$12, IV=$24).
2. The HC is then set.
3. The HCP bit in the HSR is set when HC is set.
4. If the HCIE bit in the HCR has been set by the DSP, the HC exception processing will start. The HV is multiplied by 2 and the result is used by the DSP as the interrupt vector.
5. When the HC exception is acknowledged, the HC bit (and therefore the HCP bit) is cleared by the HC logic. HC can be read by the host processor as a status bit to determine when the command is accepted. Similarly, the HCP bit can be read by the DSP CPU to determine if an HC is pending.

To guarantee a stable interrupt vector, write HV only when HC is clear. The HC bit and HV can be written simultaneously. The host processor can clear the HC bit to cancel a host command at any time before the DSP exception is accepted. Although the HV can be programmed to any exception vector, it is **not** recommended that HV=0 (RESET) be used because it does not reset the DSP hardware. DMA must be disabled to use the host exception.

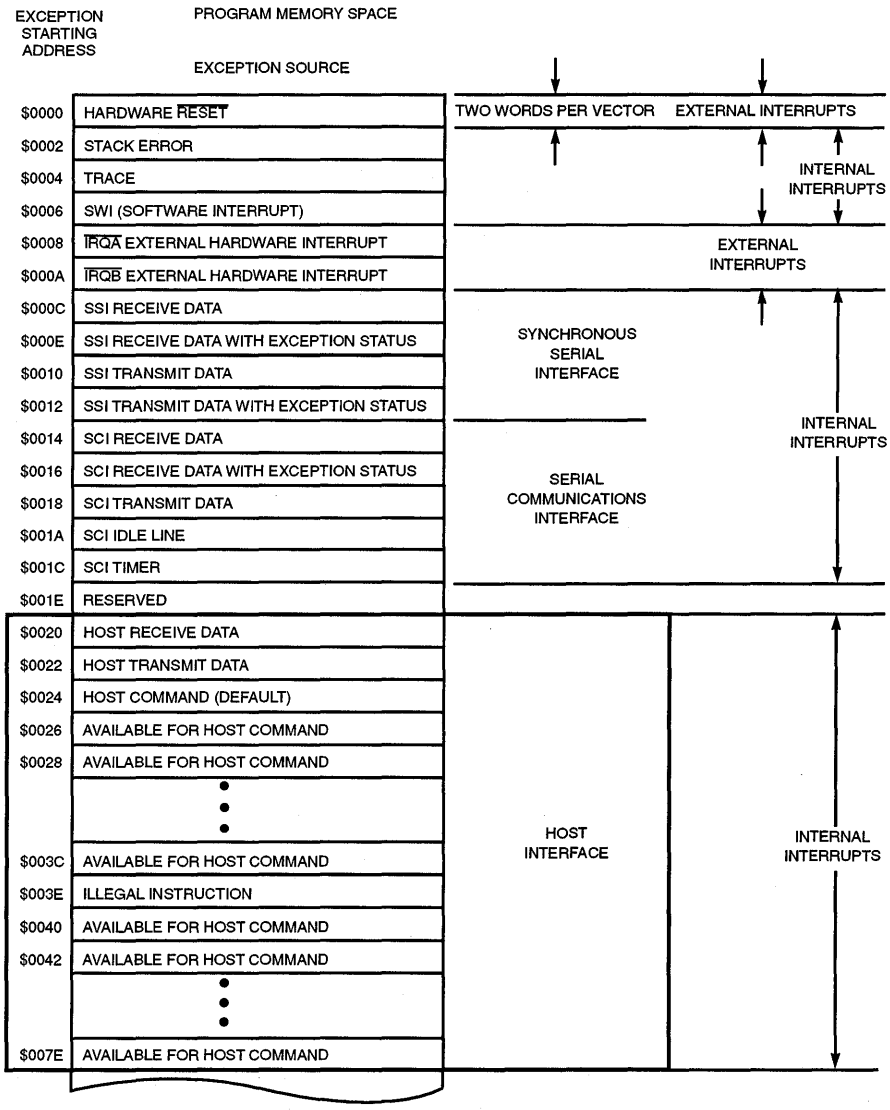


Figure 11-19. HI Exception Vector Locations

11

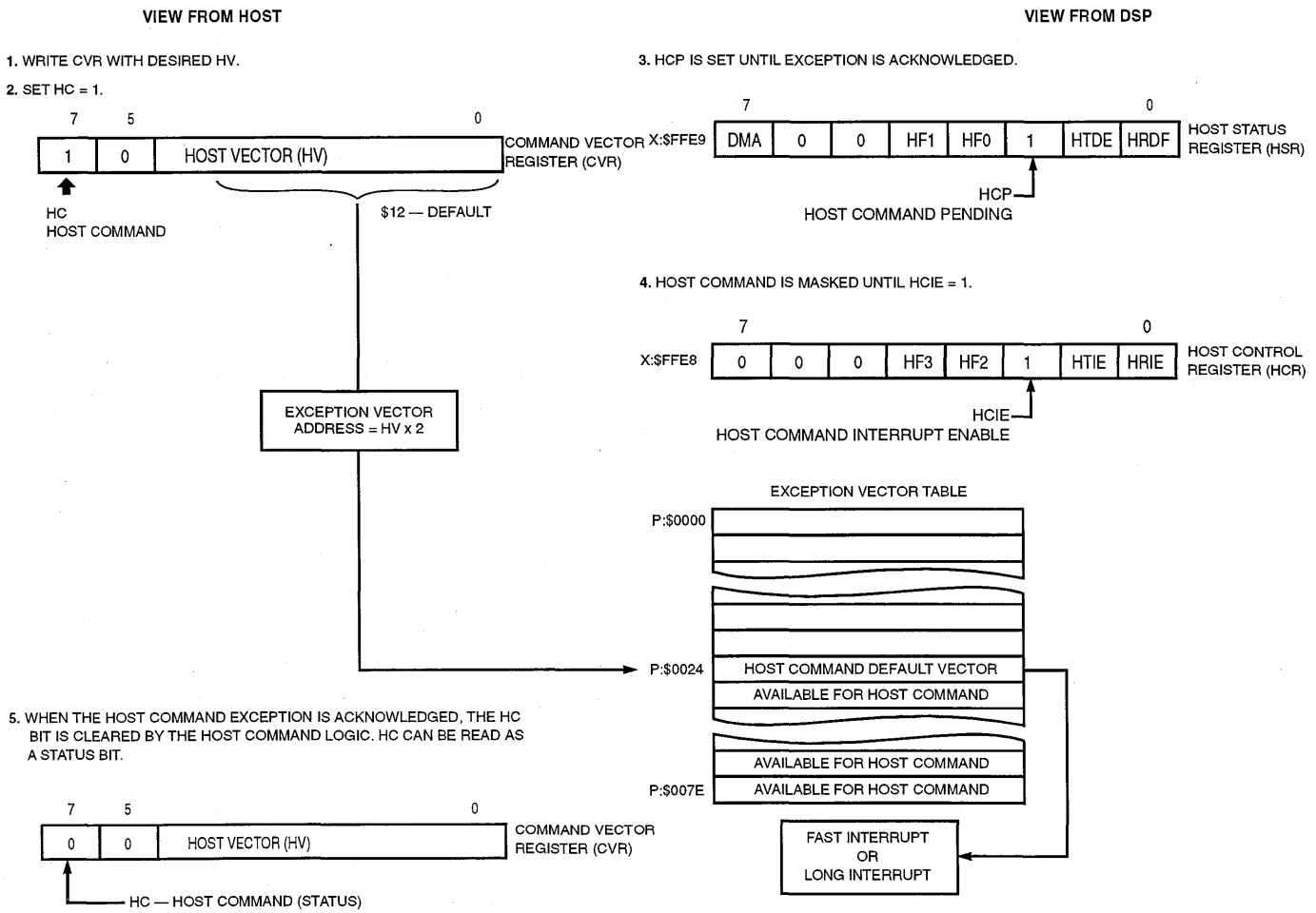
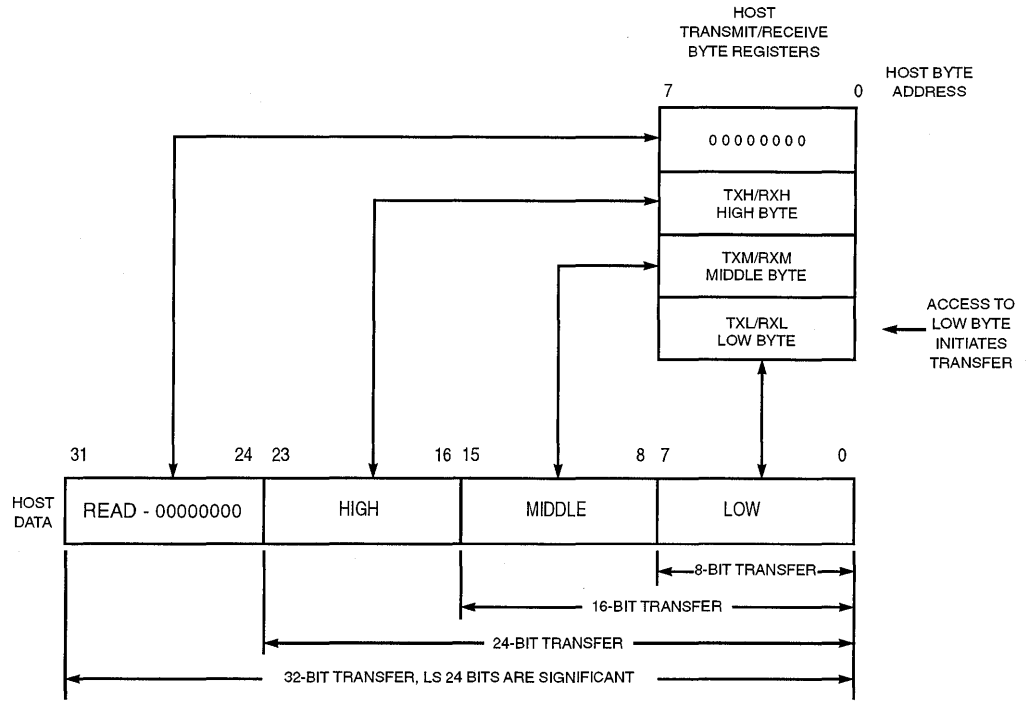


Figure 11-20. Host Command



NOTE: Access low byte last

Figure 11-21. Transmit/Receive Byte Registers

11

11.2.5.2.3 Host to DSP - Bootstrap Loading Using the HI

During the bootstrap program, the DSP looks at the MODC, MODB, and MODA bits. If the bits are set at 101 respectively, the DSP will load from the HI. Data is written by the host processor in a pattern of four bytes, with the high byte being a dummy and the low byte being the low byte of the DSP word (see Figure 11-21 and Figure 11-30). Figure 11-21 shows how an 8-,16-, 24-, or 32-bit word in the host processor maps into the HI registers. The HI register at address $4 is not used and will read as zero. The low order byte (at $7) should always be written last since writing to it causes the HI to initiate the transfer of the word to the HRX. Data is then transferred from the HRX to the DSP program memory. If the host processor needs to terminate the bootstrap loading before 512 words have been downloaded, it can set the HF0 bit in the ICR. The DSP will then terminate the download and start executing at location P:$0000. Since the DSP is typically faster than the host processor, handshaking during the data transfer is normally not required.

The actual code used in the bootstrap program is given in Appendix C DSP Bootstrap Program. The portion of the code that loads from the HI is shown in Figure 11-24. The BSET instruction configures Port B as the HI and the first JCLR looks for a flag (HF0) to indicate an early termination of the download. The second JCLR instruction causes the DSP to wait for a complete word to be received and then a MOVEP moves the data from the HI to memory.

```
;************************************
; Receive from Host Interrupt Routine
;************************************

   RCV        MOVEP     X:HRX,X:(R0)+    ;Receive data.
              RTI
END
```

Figure 11-22. Receive Data from Host Interrupt Routine

```
;****************************************
; MAIN PROGRAM. receive data from host
;****************************************

              ORG       P:$40
              MOVE      #0,R0
              MOVE      #3,M0
              MOVEP     #1,X:PBC         ;Turn on host port
              MOVEP     #0,X:HCR         ;Turn off XMT and RCV interrupts
              MOVEP     #$0C00,X:IPR     ;Turn on host interrupt
              MOVE      #0,SR            ;Unmask interrupts
              JCLR      #3,X:HSR,*       ;Wait for HF0 (from host) set to 1
              MOVEP     #$1,X:HGR        ;Enable host receive interrupt
              JMP       *                ;Now wait for interrupt
```

Figure 11-23. Receive Data from Host–Main Program


```
;************************************************
; This routine loads from the host interface.
; MC:MB:MA=100 - reserved
; MC:MB:MA=101 - host
;************************************************
HOSTLD     BSET      #0,X:PBC          ;Configure Port B as host
           DO        #512,_LOOP3       ;Load 512 instruction words
_LBLA      JCLR      #3,X:HSR,_LBLB    ;If HF0=1, stop loading data.
           ENDDO                       ;Must terminate the DO loop
           JMP       <_LOOP3           ;

_LBLB      JCLR      #0,X:HSR,_LBLA    ;Wait for HRDF to go high
                                       ;(meaning data is present).
           MOVEP     X:HRX,P:(R0)+     ;Store 24-bit data in P memory
_LOOP3                                 ;and go get another 24-bit word.
           JMP       <FINISH           ;finish bootstrap
```

Figure 11-24. Bootstrap Code Fragment

11.2.5.2.4 DSP to Host Data Transfer

Data is transferred from the DSP to the host processor in a similar manner as from the host processor to the DSP. Figure 11-25 shows the bits in the status registers (ISR and HSR) and control registers (ICR and HCR) used by the host processor and DSP CPU, respectively. The DSP CPU (see Figure 11-26) can poll the HTDE bit in the HSR (1) to see when it can send data to the host, or it can use interrupts enabled by the HTIE bit in the HCR (2). If HTIE=1 and interrupts are enabled, exception processing begins at interrupt vector P:$0022 (3). The interrupt routine should write data to the HTX (4), which will clear HTDE in the HSR. From the host's viewpoint, (5) reading the RXL clears RXDF in the ISR. When RXDF=0 and HTDE=0 (6) the contents of the HTX will be transferred to the receive byte registers (RXH:RXM:RXL). This transfer sets RXDF in the ISR (7), which the host processor can poll to see if data is available or, if the RREQ bit in the ICR is set, the HI will interrupt the host processor with $\overline{\text{HREQ}}$.

The code shown in Figure 11-27 is essentially the same as the MAIN PROGRAM in Figure 11-23 except that, since this code will transmit instead of receive data, the HTIE bit is set in the HCR instead of the HRIE bit. The actual code used in the bootstrap program is given in Appendix C DSP Bootstrap Program. The portion of the code that loads from the HI is shown in Figure 11-24. The BSET instruction configures Port B as the HI and the first JCLR looks for a flag (HF0) to indicate an early termination of the download. The second JCLR instruction causes the DSP to wait for a complete word to be received and then a MOVEP moves the data from the HI to memory.

The transmit routine used by the code in Figure 11-27 is shown in Figure 11-28. The interrupt vector contains a JSR, which makes it a long interrupt. The code sends a fixed test pattern ($123456) and then resets the HI for the next interrupt.

11

11.2.5.3 DMA DATA TRANSFER

The DMA mode allows the transfer of 8-, 16-, or 24-bit data through the DSP HI under the control of the IDMA controller. The HI provides the pipeline data registers and the synchronization logic between the two asynchronous processor systems. The DSP host exceptions provide cycle-stealing data transfers with the DSP internal or external memory. This technique allows the DSP memory address to be generated using any of the DSP addressing modes and modifiers. Queues and circular sample buffers are easily created for DMA transfer regions. The host exceptions can be programmed as high priority fast or long exception service routines. The external DMA controller provides the transfers between the DSP HI registers and the external DMA memory. The external DMA controller must provide the address to the external DMA memory; however, the address of the selected HI register is provided by a DMA address counter in the HI.

DMA transfers can only be in one direction at a time; however, the host processor can access any of the registers not in use during the DMA transfer by deasserting $\overline{\text{HACK}}$ and using $\overline{\text{HEN}}$ and HA0-HA2 to transfer data. The host can therefore transfer data in the other direction during the DMA operation using polling techniques.

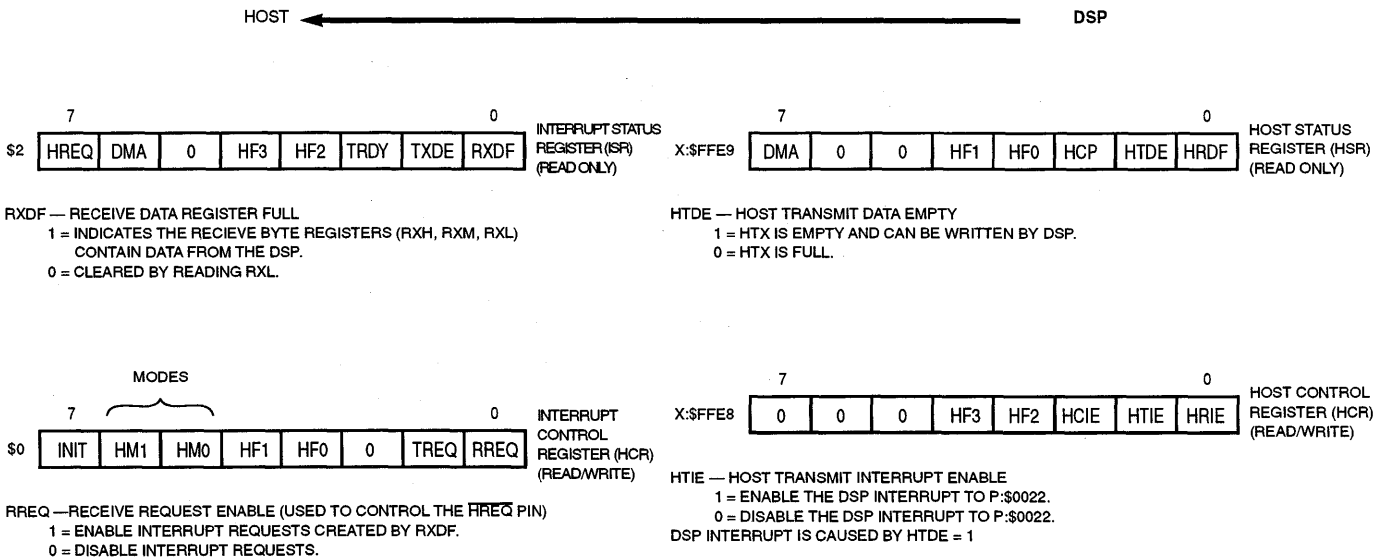


Figure 11-25. Bits Used for DSP to Host Transfer

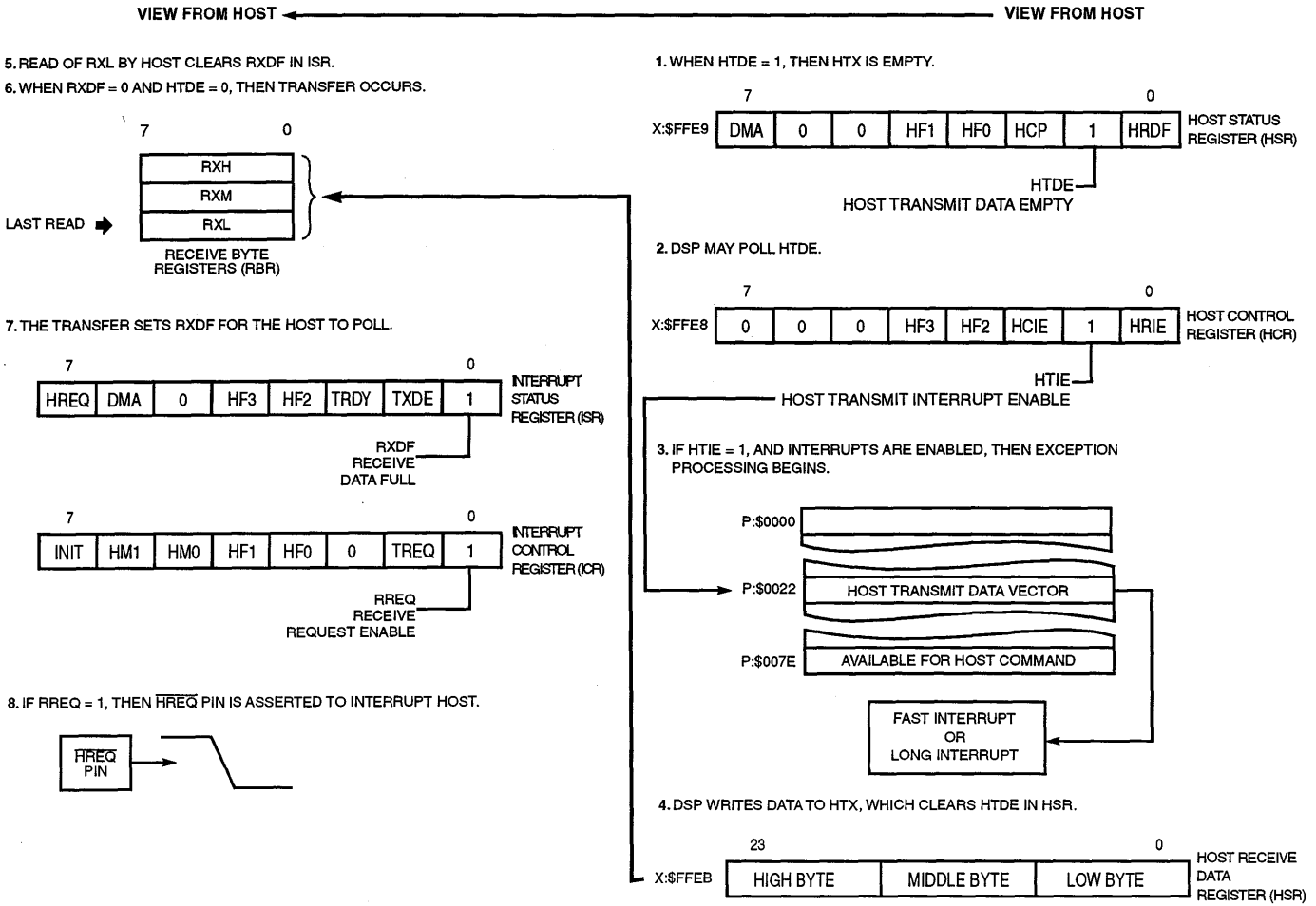


Figure 11-26. Data Transfer from DSP to Host

```
;****************************************
; MAIN PROGRAM... transmit 24-bit data to host
;****************************************
        ORG     P:$40
        MOVEP   #1,X:PBC        ;Turn on host port
        MOVEP   #$0C00,X:IPR    ;Turn on host interrupt
        MOVEP   #0,X:HCR        ;Turn off XMT and RCV interrupts
        MOVE    #0,SR           ;Unmask interrupts
        JCLR    #3,X:HSR,*      ;Wait for HF0 (from host) set to 1
        AND     X0,A
        JEQ     LOOP
        MOVEP   #$2,X:HCR       ;Enable host transmit interrupt
        JMP     *               ;Now wait for interrupt
```

Figure 11-27. Main Program - Transmit 24-Bit Data to Host

```
;************************************
;TRANSMIT to Host Interrupt Routine
;************************************
XMT     MOVEP   #$123456,X:HTX  ;Test value to transmit
        MOVEP   #0,X:HCR        ;Turn off XMT Interrupt
        RTI
```

Figure 11-28. Transmit to HI Routine

11.2.5.3.1 Host To DSP Internal Processing

The following procedure outlines the steps that the HI hardware takes to transfer DMA data from the host data bus to DSP memory (see Figure 11-29).

1. HI asserts the $\overline{\text{HREQ}}$ when TXDE=1.
2. DMA controller enables data on H0-H7 and asserts $\overline{\text{HACK}}$.
3. When $\overline{\text{HACK}}$ is asserted, the HI deasserts $\overline{\text{HREQ}}$.
4. When the DMA controller deasserts $\overline{\text{HACK}}$, the data on H0-H7 is latched into the TXH, TXM, and TXL registers.
5. If the byte register written was not TXL (i.e., not $7) the DMA address counter internal to the HI increments and $\overline{\text{HREQ}}$ is again asserted. Steps 2-5 are then repeated.

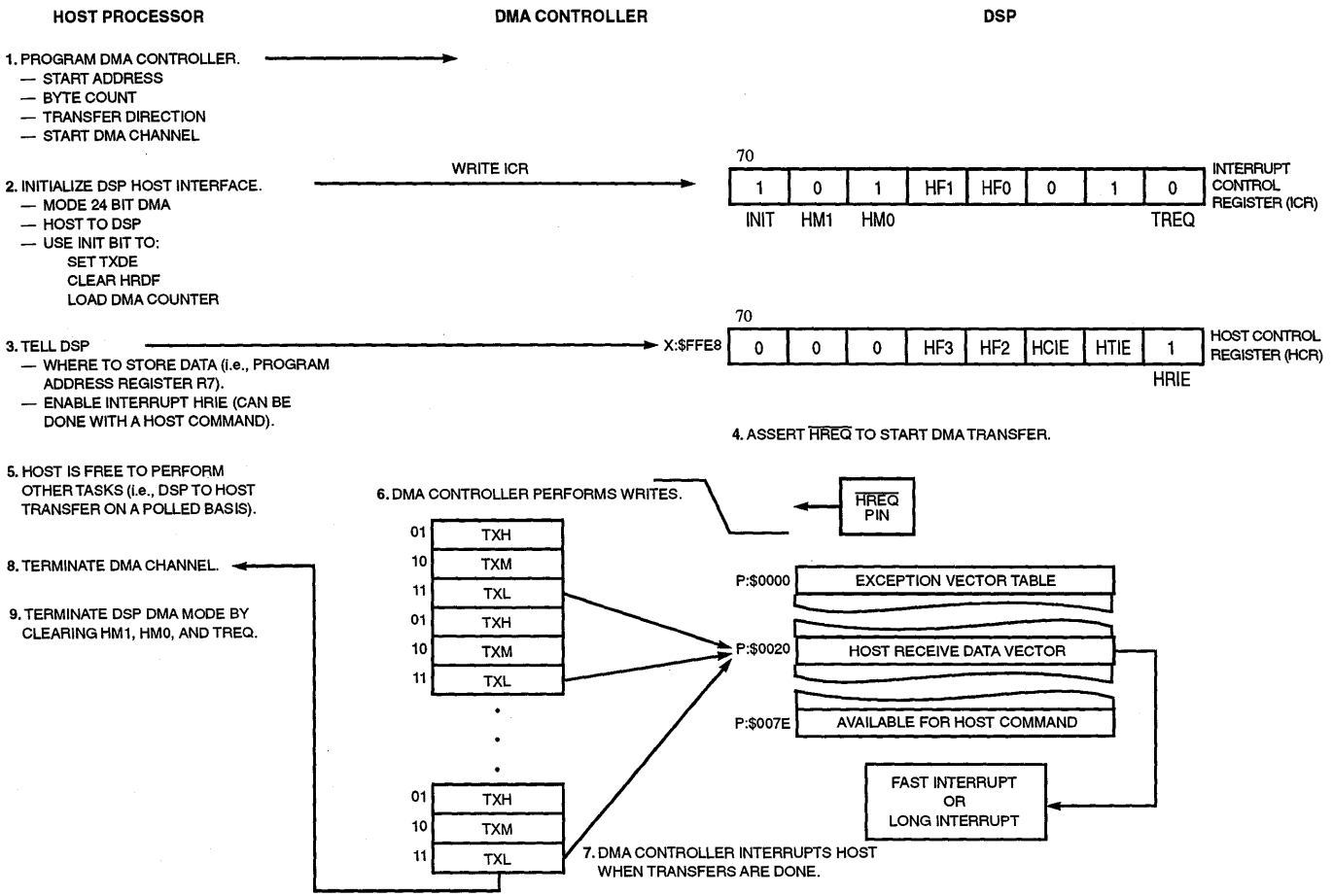
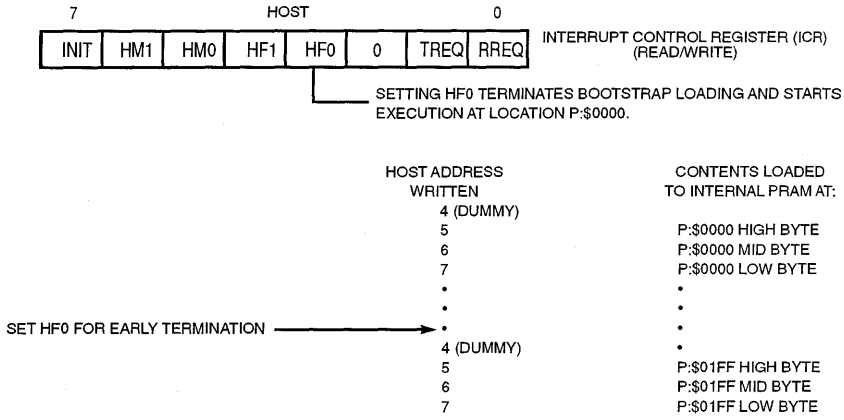


Figure 11-29. Host-to-DSP DMA Procedure

6. If TXL ($7) was written, TXDE will be set to zero and the address counter in the HI will be loaded with the contents of HM1 and HM0. When TXDE=0, the contents of TXH:TXM:TXL are transferred to HRX, provided HRDF=0. After the transfer to HRX, TXDE will be set to one, and $\overline{\text{HREQ}}$ will be asserted to start the transfer of another word from external memory to the HI.
7. When the transfer to HRX occurs within the HI, HRDF is set to one. Assuming HRIE=1, a host receive exception will be generated. The exception routine must read the HRX to clear HRDF.

NOTE

The transfer of data from the TXH, TXM, TXL registers to the HRX register automatically loads the DMA address counter from the HM1 and HM0 bits in the DMA host to DSP mode. This DMA address is used with the HI to place the received byte in the correct register (TXH, TXM, or TXL).



• Because the DSP is so fast, host handshaking is generally not required.

Figure 11-30. Bootstrap Using the HI

The interrupt rate is three times more frequent for 8-bit data transfers than for 24-bit transfers. TXL is always loaded last.

11.2.5.3.2 Host to DSP DMA Procedure

The following procedure outlines the typical steps that the host processor must take to setup and terminate a host-to-DSP DMA transfer (see Figure 11-29).

1. Set up the external DMA controller (1) source address, byte count, direction, and other control registers. Enable the DMA controller channel.
2. Set up the HCOR register setting the HRE and HACK bits to one and setting HRC=00. Also, set up the HBAR register.
3. Initialize the HI (2) by writing the ICR to select the word size (HM0 and HM1), to select the direction (TREQ=1, RREQ=0), and to initialize the channel setting INIT=1 (see Figure 11-31).

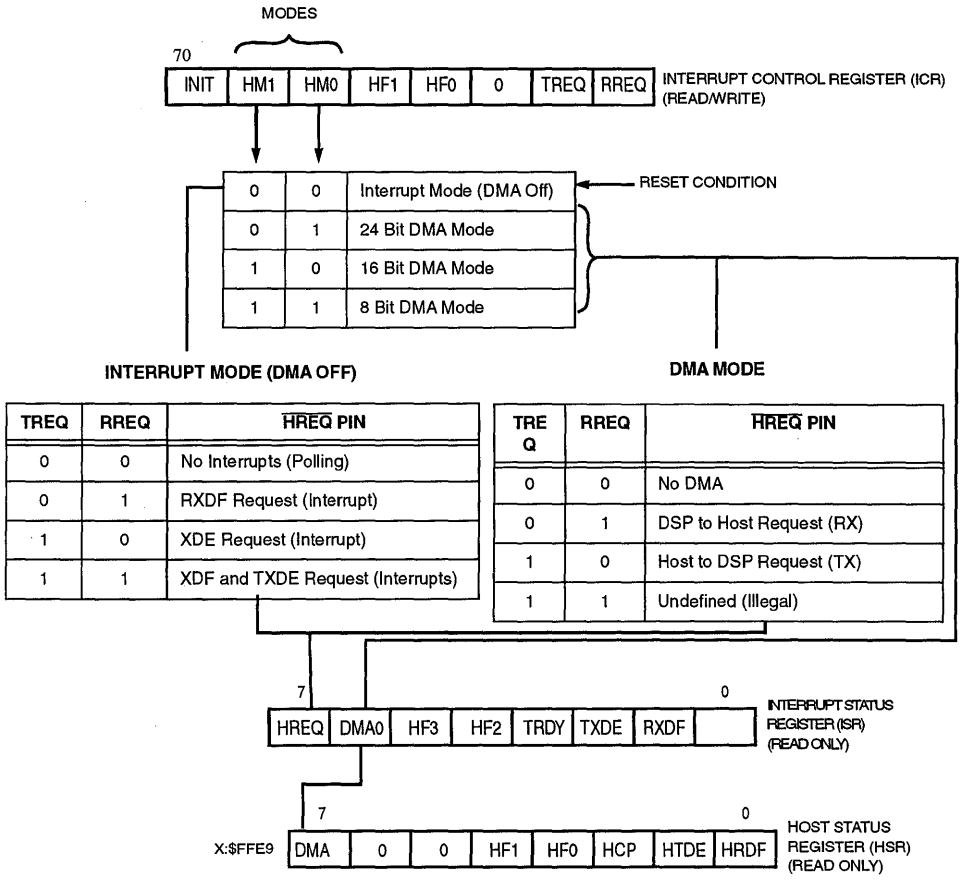


Figure 11-31. Host Bits with TREQ and RREQ

4. The DSP's destination pointer (3) used in the DMA exception handler (an address register, for example) must be initialized and HRIE must be set to enable the HRDF interrupt to the DSP CPU. This procedure can be done with a separate host command exception routine in the DSP. $\overline{\text{HREQ}}$ will be asserted (4) immediately by the HI to begin the DMA transfer.
5. Perform other tasks (5) while the DMA controller transfers data (6) until interrupted by the DMA controller DMA transfer complete interrupt (7). The DSP interrupt control register (ICR), the interrupt status register (ISR), and RXH, RXM, and RXL registers may be accessed at any time by the host processor but the TXH, TXM and TXL registers may not be accessed until the DMA mode is disabled.
6. Terminate the DMA controller channel (8) to disable DMA transfers.
7. Terminate the DSP HI DMA mode (9) in the ICR by clearing the HM1 and HM0 bits and clearing TREQ.

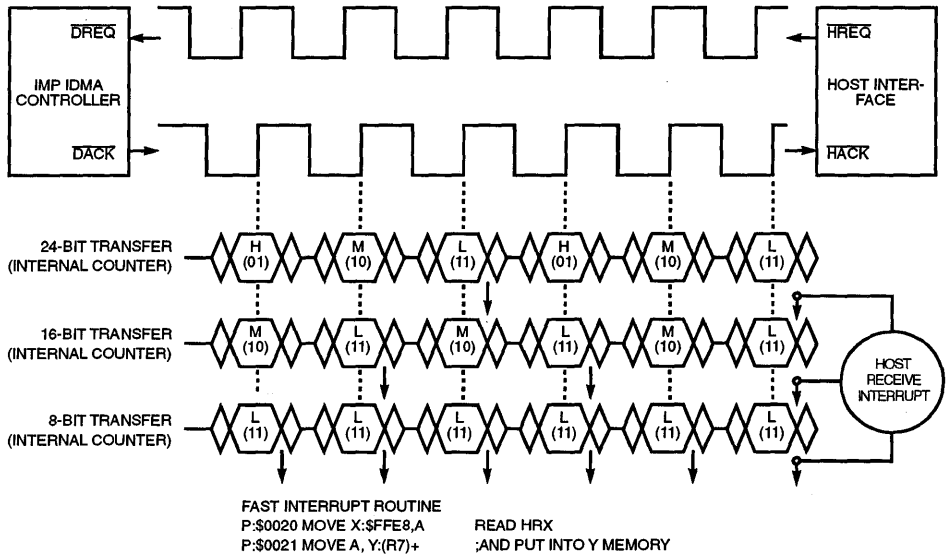


Figure 11-32. DMA Transfer and Host Interrupts

The $\overline{\text{HREQ}}$ will be active immediately after initialization is completed (depending on hardware) because the data direction is host to DSP and TXH, TXM, and TXL registers are empty. When the host writes data to TXH, TXM, and TXL, this data will be immediately transferred to HRX. If the DSP is to work in interrupt mode, HRIE must be enabled.

11

11.2.5.3.3 DSP to Host Internal Processing

The following procedure outlines the steps that the HI hardware takes to transfer DMA data from DSP memory to the host data bus.

1. On the DSP side of the HI, a host transmit exception will be generated when HTDE=1 and HTIE=1. The exception routine must write HTX, thereby setting HTDE=0.
2. If RXDF=0 and HTDE=0, the contents of HTX will be automatically transferred to RXH:RXM:RXL, thereby setting RXDF=1 and HTDE=1. Since HTDE=1 again on the initial transfer, a second host transmit exception will be generated immediately, and HTX will be written, which will clear HTDE again.
3. When RXDF is set to one, the HI's internal DMA address counter is loaded (from HM1 and HM0) and $\overline{\text{HREQ}}$ is asserted.
4. The DMA controller enables the data from the appropriate byte register onto H0-H7 by asserting $\overline{\text{HACK}}$. When $\overline{\text{HACK}}$ is asserted, $\overline{\text{HREQ}}$ is deasserted by the HI.
5. The DMA controller latches the data presented on H0-H7 and deasserts $\overline{\text{HACK}}$. If the byte register read was not RXL (i.e., not $7), the HI's internal DMA counter increments, and $\overline{\text{HREQ}}$ is again asserted. Steps 3, 4, and 5 are repeated until RXL is read.
6. If RXL was read, RXDF will be set to zero and, since HTDE=0, the contents of HTX will be automatically transferred to RXH:RXM:RXL, and RXFD will be set to one. Steps

3, 4, and 5 are repeated until RXL is read again.

NOTE

The transfer of data from the HTX register to the RXH:RXM:RXL registers automatically loads the DMA address counter from the HM1 and HM0 bits when in the DMA DSP–HOST mode. This DMA address is used within the HI to place the appropriate byte on H0-H7.

11.2.5.3.4 DSP to Host DMA Procedure

The following procedure outlines the typical steps that the host processor must take to setup and terminate a DSP-to-host DMA transfer (see Figure 11-33).

1. Set up the DMA controller (1) destination address, byte count, direction, and other control registers. Enable the DMA controller channel.
2. Set up the HCOR register setting the HRE and HACK bits to one and setting HRC=00. Also, set up the HBAR register.
3. Initialize the HI (2) by writing the ICR to select the word size (HM0 and HM1), the direction (TREQ=0, RREQ=1), and setting INIT=1 (see Figure 11-33 for additional information on these bits).
4. The DSP's source pointer (3) used in the DMA exception handler (an address register, for example) must be initialized, and HTIE must be set to enable the DSP host transmit interrupt. This could be done by the host processor with a host command exception routine.
5. The DSP host transmit exception will be activated immediately after HTIE is set. The DSP CPU will move data to HTX. The HI circuitry will transfer the contents of HTX to RXH:RXM:RXL, setting RXDF which asserts $\overline{\text{HREQ}}$. Asserting $\overline{\text{HREQ}}$ (4) starts the DMA transfer from RXH, RXM, and RXL to the host processor.
6. Perform other tasks (5) while the DMA controller transfers data (6) until interrupted by the DMA controller DMA complete interrupt (7). The DSP interrupt control register (ICR), the interrupt status register (ISR), and TXH, TXM, and TXL may be accessed at any time by the host processor but the RXH, RXM and RXL registers may not be accessed until the DMA mode is disabled.
7. Terminate the DMA controller channel (8) to disable DMA transfers.
8. Terminate the DSP HI DMA mode (9) in the Interrupt Control Register (ICR) by clearing the HM1 and HM0 bits and clearing RREQ.

11.2.5.4 HOST PORT USAGE CONSIDERATIONS–HOST SIDE

Synchronization is a common problem when two asynchronous systems are connected, and careful synchronization is required when reading multi-bit registers that are written by another asynchronous system. The considerations for proper operation are discussed below.

1. Unsynchronized Reading of Receive Byte Registers:

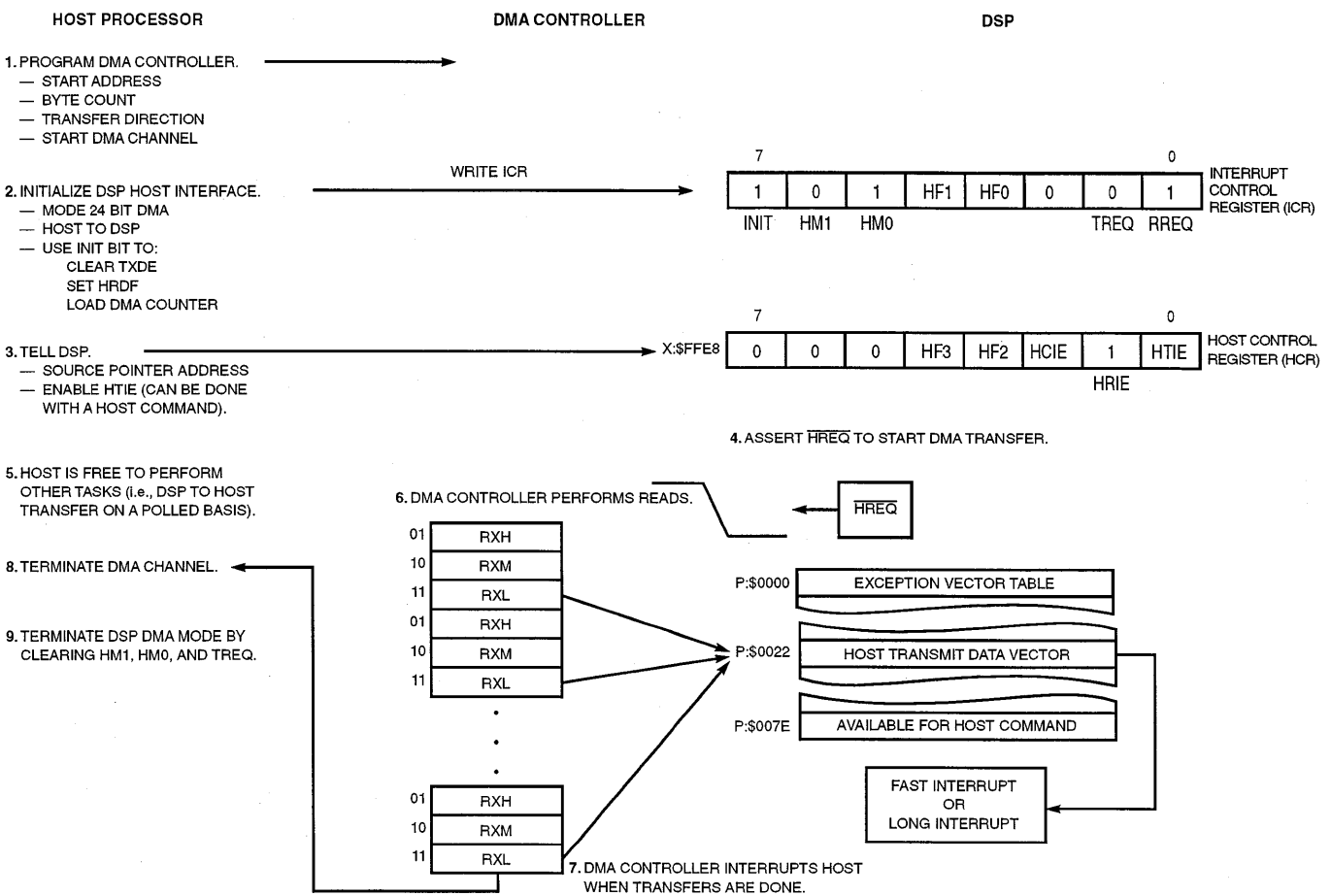


Figure 11-33. DSP to Host DMA Procedure

When reading receive byte registers, RXH, RXM, or RXL, the host programmer should use interrupts or poll the RXDF flag, which indicates that data is available. This guarantees that the data in the receive byte registers will be stable.

2. Overwriting Transmit Byte Registers:

The host programmer should not write to the transmit byte registers, TXH, TXM, or TXL, unless the TXDE bit is set, indicating that the transmit byte registers are empty. This guarantees that the DSP will read stable data when it reads the HRX register.

3. Synchronization of Status Bits from DSP to Host:

HC, HREQ, DMA, HF3, HF2, TRDY, TXDE, and RXDF status bits are set or cleared from inside the HI and read by the host processor. T.he host can read these status bits very quickly without regard to the clock rate used by the DSP, but there is a chance that the state of the bit could be changing during the read operation. This possible change is generally not a system problem, since the bit will be read correctly in the next pass of any host polling routine.

The only potential problem with the host processor's reading of status bits would be its reading HF3 and HF2 as an encoded pair. For example, if the DSP changes HF3 and HF2 from "00" to "11", there is a small possibility that the host could read the bits during the transition and receive "01" or "10" instead of "11". If the combination of HF3 and HF2 has significance, the host processor could potentially read the wrong combination. Two solutions would be to 1) read the bits twice and check for consensus, or 2) hold $\overline{\text{HEN}}$ access for $\overline{\text{HEN}}$ + *x* clock cycles so that status bit transitions are stabilized.

4. Overwriting the Host Vector:

The host programmer should change the host vector register only when the HC bit is clear. This will guarantee that the DSP interrupt control logic will receive a stable vector.

5. Cancelling a Pending Host Command Exception:

The host processor may elect to clear the HC bit to cancel the host command exception request at any time before it is recognized by the DSP. The DSP CPU may execute the host exception after the HC bit is cleared because the host processor does not know exactly when the exception will be recognized. This uncertainty in timing is due to differences in synchronization between the host processor and DSP CPU and the uncertainties of pipelined exception processing. For this reason, the HV should not be changed at the same time the HC bit is cleared. However, the HV can be changed when the HC bit is set.

SECTION 12
DSP SERIAL PORTS

12.1 INTRODUCTION

The DSP has two serial ports, a serial communications interface (SCI+) and a synchronous serial interface (SSI). The serial communications interface does not interface to external pins directly but can be connected internally to the IMP and its pins. The SSI port has six dedicated pins that can either be connected as Port C GPIO, or they can be configured as the synchronous serial interface (SSI) pins.

When configured as general-purpose I/O, port C, which consists of the six SSI pins, can be used for device control. When the pins are configured as SSI pins, port C provides a convenient connection to other DSPs, processors, codecs, digital-to-analog and analog-to-digital converters, and any of several transducers. This section describes the SCI+ port, Port C, and the SSI port as well as examples of how to configure and use each function. Information on connecting the SCI+ port to the IMP is found in Section 7 Communications Processor (CP).

12.2 GENERAL-PURPOSE I/O (PORT C)

When it is configured as GPIO, Port C can be viewed as six I/O pins (see Figure 12-1), which are controlled by three memory-mapped registers. These registers are the Port C control register (PCC), Port C data direction register (PCDDR), and Port C data register (PCD) (see Figure 12-2).

Reset clears PCC and PCDDR to configure port C as general-purpose I/O with all six pins as inputs. (External circuitry connected to these pins may need pullups until the pins are configured for operation.) Each port C pin may be individually programmed as a general-purpose I/O pin or as a dedicated on-chip peripheral pin under software control. Pin selection between general-purpose I/O and SSI is made by setting the appropriate PCC bit (memory location X:$FFE1) to zero for general-purpose I/O or to one for serial interface.

NOTE

Bits 0-2 in the PCC must be set to 1 to enable SCI+ operation even though there are no Port C 0-2 pins.

The PCDDR (memory location X:$FFE3) programs each pin corresponding to a bit in the PCD (memory location X:$FFE5) as an input pin (if PCDDR=0) or as an output pin (if PCDDR=1).

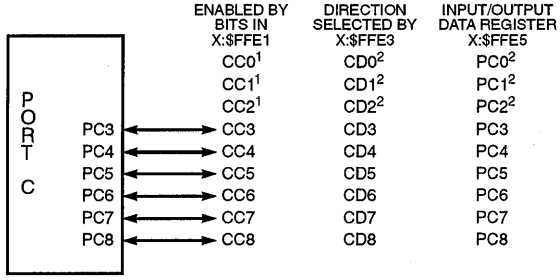


Figure 12-1. Port C GPIO Control

[1] These bits must be set to one to enable SCI+ operation.
2. Even though there are no corresponding pins, Bits 0-2 of the data and data direction registers can still be used to setup and monitor the SCI+ TX and RX data, and transmit clock signals.

If a pin is configured as a GPIO **input** (as shown in Figure 12-3) and the processor reads the PCD, the processor sees the logic level on the pin. If the processor writes to the PCD, the data is latched there, but does not appear on the pin because the buffer is in the high-impedance state.

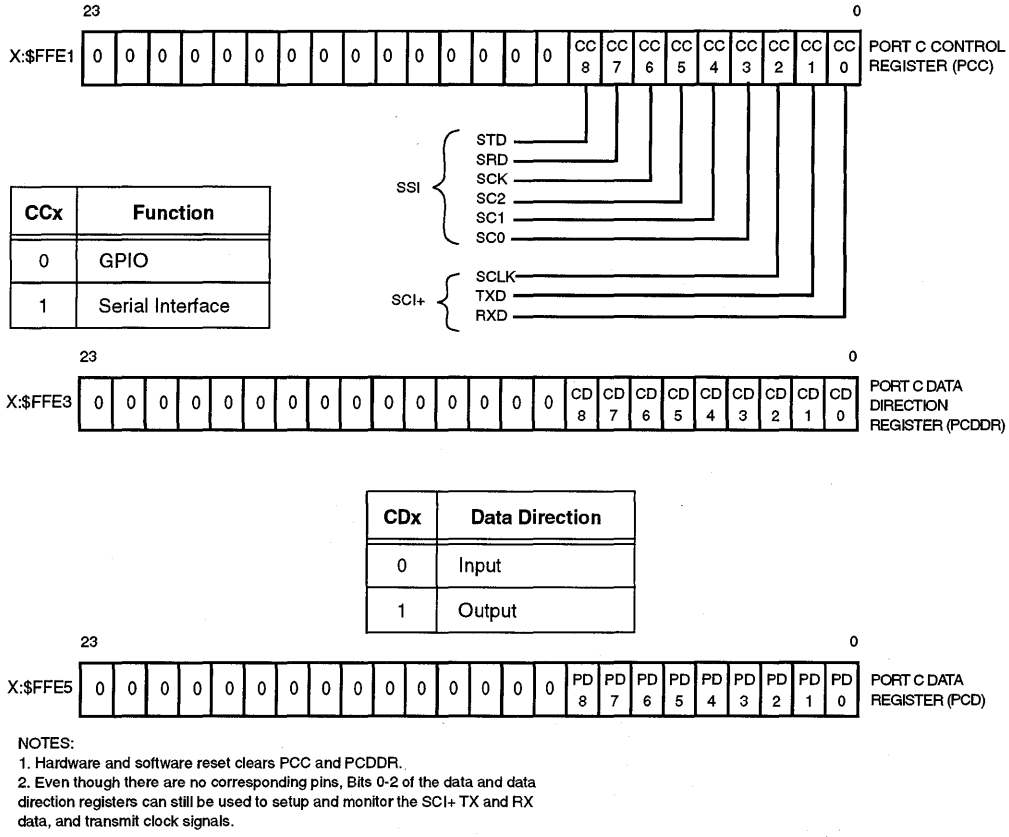
If a pin is configured as a GPIO **output** and the processor reads the PCD, the processor sees the contents of the PCD rather the logic level on the pin, which allows the PCD to be used as a general purpose15-bit register. If the processor writes to the PCD, the data is latched there and appears on the pin during the following instruction cycle (see 12.2.2 Port C General Purpose I/O Timing).

12

If a pin is configured as a synchronous **serial interface** (SSI) pin and/or if the SCI+ port is enabled, the port C GPIO registers can be used to help in debugging the serial interface. If the PCDDR bit for a given pin (or SCI+ signal) is cleared (configured as an input), the PCD will show the logic level on the pin (or signal for the SCI+ port), regardless of whether the serial interface function is using the pin as an input or an output. If the PCDDR is set (configured as an output) for a given serial interface pin, when the processor reads the PCD, it sees the contents of the PCD rather than the logic level on the pin — another case which allows the PCD to act as a general purpose register.

NOTE

The TXCLK cannot be monitored in the PCD.



NOTES:
1. Hardware and software reset clears PCC and PCDDR.
2. Even though there are no corresponding pins, Bits 0-2 of the data and data direction registers can still be used to setup and monitor the SCI+ TX and RX data, and transmit clock signals.

Figure 12-2. Port C GPIO Registers

12.2.1 Programming General Purpose I/O

Port C and all the DSP56002 peripherals are memory mapped (see Figure 12-4). The standard MOVE instruction transfers data between port C and a register; as a result, performing a memory-to-memory data transfer takes two MOVE instructions and a register. The MOVEP instruction is specifically designed for I/O data transfer as shown in Figure 12-6. Although the MOVEP instruction may take twice as long to execute as a MOVE instruction, only one MOVEP is required for a memory-to-memory data transfer, and MOVEP does not use a temporary register. Using the MOVEP instruction allows a fast interrupt to move data to/from a peripheral to memory and execute one other instruction or to move the data to an absolute address. MOVEP is the only memory-to-memory move instruction; however, one of the operands must be in the top 64 locations of either X: or Y: memory. The bit-oriented instructions which use I/O short addressing (BCHG, BCLR, BSET, BTST, JCLR, JSCLR, JSET, and JSSET) can also be used to address individual bits for faster I/O processing.

Port Control Register Bit	Data Direction Register Bit	Pin Function
0	0	Port Input Pin
0	1	Port Output Pin
1	X	Alternate Function

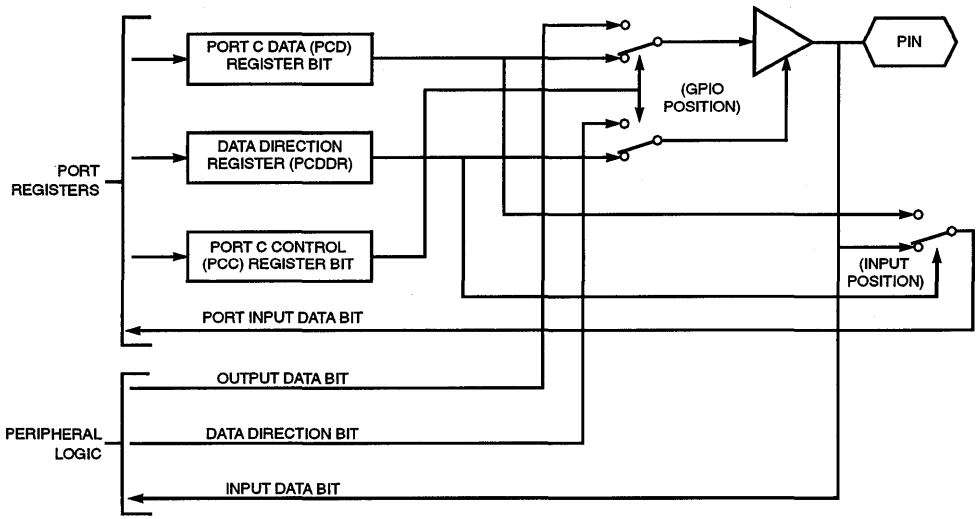


Figure 12-3. Port C I/O Pin Control Logic

12

The DSP does not have a hardware data strobe to strobe data out of the GPIO port. If a data strobe is needed, it can be implemented using software to toggle one of the GPIO pins.

Figure 12-5 shows the process of programming port C as general-purpose I/O. Normally, it is not good programming practice to activate a peripheral before programming it. However, reset activates the port C general-purpose I/O as all inputs, and the alternative is to configure the port as an SCI and/or SSI, which may not be desirable. In this case, it is probably better to ensure that port C is initially configured for general-purpose I/O and then configure the data direction and data registers. It may be better in some situations to program the data direction or the data registers first to prevent two devices from driving one signal. The order of steps 1, 2, and 3 in Figure 12-5 is optional and can be changed as needed.

12.2.2 Port C General Purpose I/O Timing

Parallel data written to port C is delayed by one instruction cycle. For example, the following instruction:

```
MOVEDATA6,X:PORTCDATA24,Y:EXTERN
```

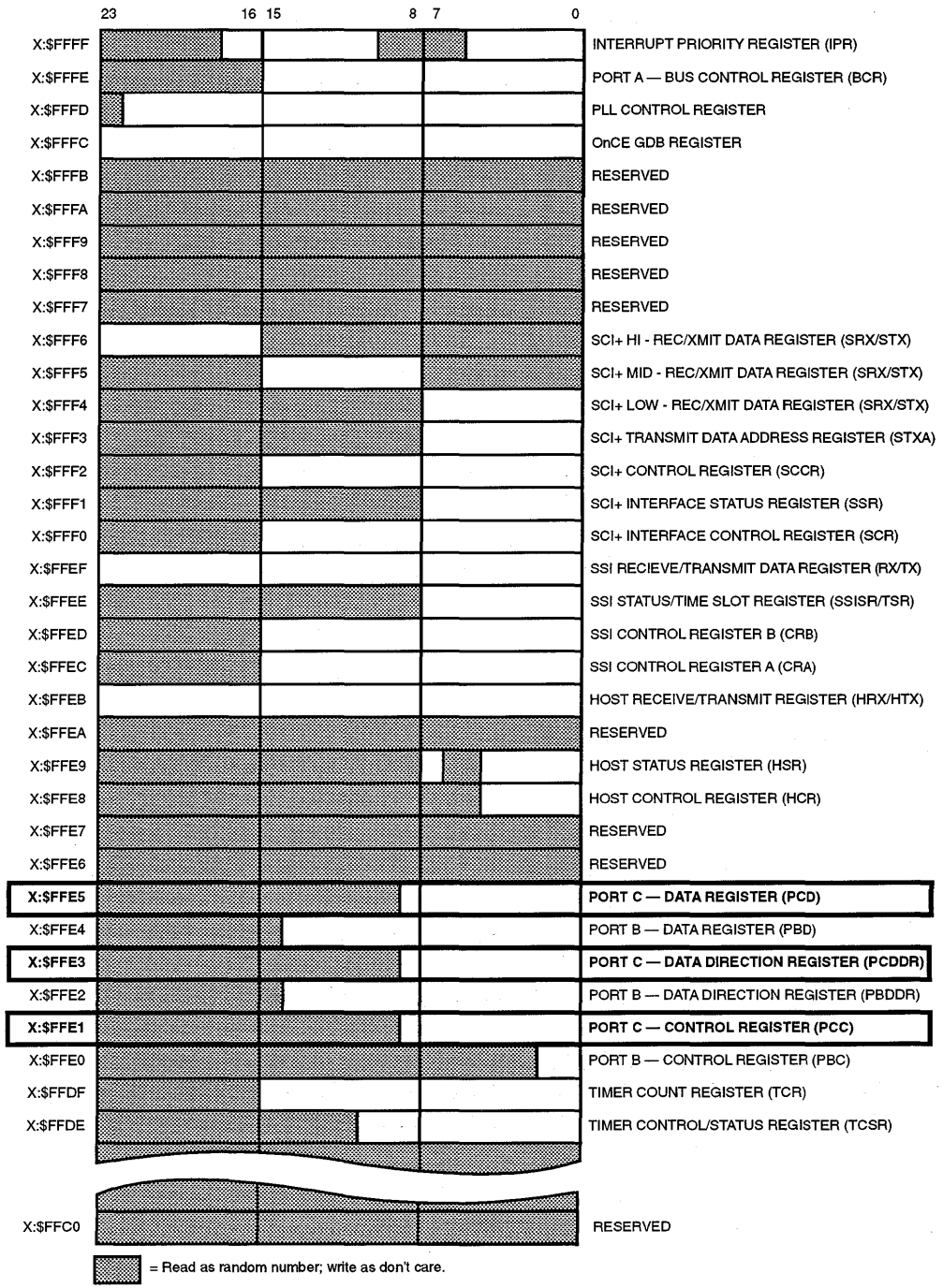


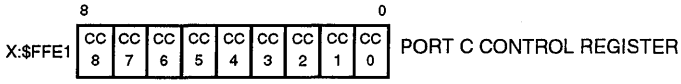
Figure 12-4. On-Chip Peripheral Memory Map

1. Writes six bits of data to the port C register, but the output pins do not change until the following instruction cycle
2. Writes 24 bits of data to the external Y memory, which appears on port A during T2 and T3 of the current instruction

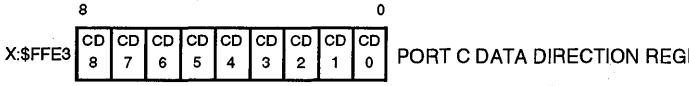
As a result, if it is necessary to synchronize the port A and port C outputs, two instructions must be used:

```
MOVEDATA6,X:PORTC
NOPDATA24,Y:EXTERN
```

STEP 1. SELECT EACH PIN TO BE GENERAL-PURPOSE I/O OR AN ON-CHIP PERIPHERAL PIN:
CCx = 0 GENERAL- PURPOSE I/O (except CC0-2).
CCx = 1 ON-CHIP PERIPHERAL



STEP 2. SET EACH GENERAL-PURPOSE I/O PIN (SELECTED ABOVE) AS INPUT OR OUTPUT:
CDx = 0 INPUT PIN (CD0-2 can be set to inputs to monitor the SCI+ port)
OR
CDx = 1 OUTPUT PIN



STEP 3. READ/WRITE GENERAL-PURPOSE I/O PINS:
PCx = OUTPUT DATA IF SELECTED FOR GENERAL-PURPOSE I/O AND OUTPUT IN STEPS 1 AND 2.
OR
PCx = INPUT DATA IF SELECTED FOR GENERAL-PURPOSE I/O AND INPUT IN STEPS 1 AND 2.
(PC0 = SCI+ RXD, PC1=SCI+ TXD, and PC2 = SCI+ TXCLK, RXCLK not available for monitoring.)

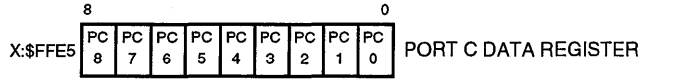


Figure 12-5. I/O Port C Configuration

The NOP can be replaced by any instruction that allows parallel moves. Inserting one or more "MOVE DATA15,X:PORTC DATA24,Y:EXTERN" instructions between the first and second instruction produces an external 33-bit write each instruction cycle with only one instruction cycle lost in setup time:

```
MOVEDATA6,X:PORTC
MOVEDATA6,X:PORTCDATA24,Y:EXTERN
MOVEDATA6,X:PORTCDATA24,Y:EXTERN
:
:
MOVEDATA6,X:PORTCDATA24,Y:EXTERN
NOP     DATA24,Y:EXTERN
```

One application of this technique is to create an extended address for port A by concatenating the port A address bits (instead of data bits) to the port C general-purpose output bits. The port C general-purpose I/O register would then work as a base address register, allowing the address space to be extended from 64K words (16 bits) to 33.5 million words (16 bits + 6 bits = 22 bits).

```
:
:
MOVEP #$0,X:$FFE1;Select port C to be general-purpose I/O
MOVEP #$01F0,X:$FFE3;Select pins PC4-PC8 to be outputs
:            ;(pins PC0-PC2 do not exist on the MC68356)
:
MOVEP #data_out,X:$FFE5;Put bits 4-8 of "data_out" on pins
 ;PB4-PB8, bits 0-3 are ignored.
MOVEP X:$FFE0,#data_in;Put PB0-PB3 in bits 0-3 of "data_in"
```

Figure 12-6. Write/Read Parallel Data with Port C

Port C uses the DSP central processing unit (CPU) four-phase clock for its operation. Therefore, if wait states are inserted in the DSP CPU timing, they also affect port C timing. As a result, port A and port C in the previous synchronization example will always stay synchronized, regardless of how many wait states are used.

12.3 SERIAL COMMUNICATION INTERFACE (SCI+)

The SCI+ provides a full-duplex port for serial communication to other DSPs, microprocessors, or peripherals such as modems. This interface uses four signals: transmit data (TXD), receive data (RXD), transmit clock (TXCLK) and receive clock (RXCLK). These signals must be internally connected to the IMP communications processor (See 7.5.3.2 SCI+ Serial Connections).The SCI+ supports industry-standard asynchronous bit rates and protocols as well as high-speed (up to 7.6 Mbps for a 60-MHz clock) synchronous slave data transmission.

NOTE

Wired-or mode of the transmit data line is not supported.

The SCI+ consists of separate transmit and receive sections whose operations can be asynchronous with respect to each other. A programmable baud-rate generator provides the transmit and receive clocks, or the transmit and receive clocks can be generated externally and can be asynchronous with respect to each other. An enable vector and an interrupt vector have been included so that the baud-rate generator can function as a general-purpose timer when it is not being used by the SCI+ peripheral or when the interrupt timing is the same as that used by the SCI+. The following is a short list of SCI+ features:

- Four-Signal Interface which connects internally to the IMP:
  - TXD – Transmit Data
  - RXD – Receive Data
  - TXCLK – Transmit Serial Clock
  - RXCLK – Receive Serial Clock
- 2625-kbps NRZ Asynchronous Communications Interface (60-MHz System Clock)

- 7.5 Mbps Synchronous Serial Mode (60-MHz System Clock)
- On-Chip or External Baud Rate Generation/Interrupt Timer
- Four Interrupt Priority Levels
- Fast or Long Interrupts

12.3.1 SCI+ to IMP Connection Options

The SCI+ port on the MC68356 can only interface to the IMP or off-chip to other communications devices through IMP pins. There are three options for connecting the SCI+ to other devices. These options are configured through the IC0-3 bits in the IMP's DISC register:

1. Direct connection to the IMP SCC1 of the IMP (SCI+ to SCC1) - This mode is used to establish a direct serial connection between the DSP and the IMP. In this mode, the four SCI+ signals are directly connected to the corresponding SCC1 signals. In this mode the SCI+ can be programmed for synchronous slave operation or asynchronous operation to communicate with SCC1 only. For more information on this serial connections mode refer to 7.5.3.2 SCI+ Serial Connections.

2. Direct connection to NMSI2 pins (SCI+ to DTE) - This mode allows the SCI+ to be connected to the NMSI2 pins normally used by the SCC2. This mode is useful when it is desired to connect the SCI+ directly to the external pins. In this mode, SCC2 cannot be used except to monitor the RXD channel. In this mode, the SCI+ can be programmed for synchronous slave operation or asynchronous operation. For more information on this serial connections mode refer to 7.5.3.2 SCI+ Serial Connections.

3. Direct connection of the SCI+ clock lines to the NMSI2 clock lines, and direct connection of data and clock lines to the IMP SCC1 (SCI+ clocks with ISDN) - This mode is useful when the NMSI1 pins are being used with ISDN and it is still desired to have the SCI+ to SCC1 data transfer clocked by en external source.

All of the above connection modes are described in detail in 7.5.3.2 SCI+ Serial Connections.

12.3.2 The SCI+ in a Modem Application

Figure 12-7 shows the suggested serial port assignments for a modem application.

The typical modem application using the MC68356 would use the IMP to handle the upper level data compression, error correction and serial bit stream formatting (such as HDLC). Data processed by the IMP would then be transferred to the DSP from SCC1 to the SCI+ port. A codec typically is connected to the synchronous serial interface (SSI) of the DSP and optionally provides the bit and baud clocks to the SCC1 to SCI+ connection to clock the data transfers. The analog side of the codec is connected to the Data Access Arrangement (DAA) which interfaces to the telephone line.

The MC68356 has some extra options for connection of the SCI+ signals to the serial channels and serial pins of the IMP. It is suggested that SCC1 be used for the serial port that transmits the data stream to the SCI+ port. SCC2 should be used for the connection to the DTE (typically the PC or host computer). SCC2 is a good choice to connect to the host com-

puter because this serial block and its parameter RAM are also used by the 16550 emulation controller when enabled. The 16550 emulation controller is used for the DTE interface in plug-in modem card applications including PCMCIA (See Section 7 Communications Processor (CP)).

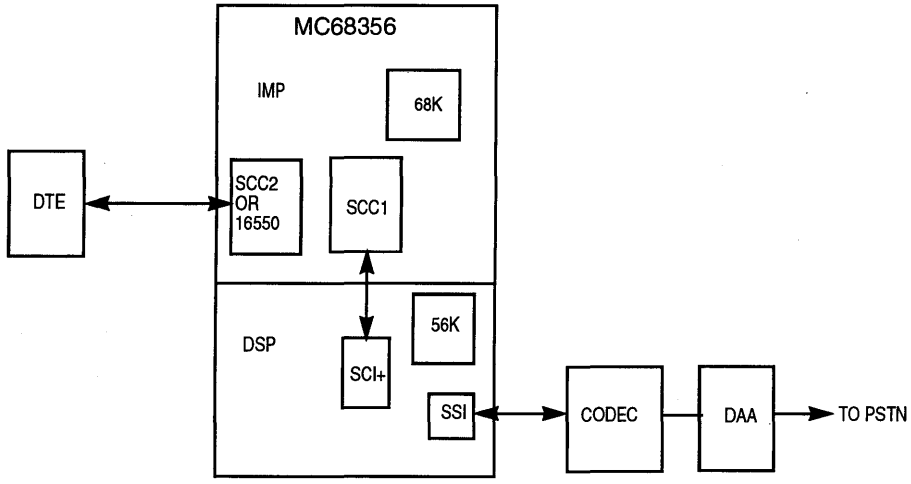


Figure 12-7. Suggested Modem Serial Port Assignments

12.3.3 Programming Port C to Enable the SCI+

At least one of the Port C control register bits CC0-2 must be programmed to a one to release the SCI+ from reset.

SCI+ interrupts may be enabled by programming the SCI+ control registers before any of the Port C control register bits CC0-2 are set. In this case, only one transmit interrupt can be generated because the transmit data register is empty. The timer and timer interrupt will operate as they do when at least one Port C control register bit (CC0-2) is set to one.

12.3.3.1 RECEIVE DATA (RXD)

This input receives byte-oriented serial data and transfers the data to the SCI+ receive shift register. Asynchronous input data is sampled on the positive edge of the receive clock (1 × RXCLK) if SCKP equals zero. See Section 14 Electrical Characteristics for detailed timing information. See 7.5.3.2 SCI+ Serial Connections for internal IMP receive data connection options.

12.3.3.2 TRANSMIT DATA (TXD)

This output transmits serial data from the SCI+ transmit shift register. Data changes on the negative edge of the asynchronous transmit clock (TXCLK) if SCKP equals zero. This output is stable on the positive edge of the transmit clock. See Section 14 Electrical Characteristics for detailed timing information. See 7.5.3.2 SCI+ Serial Connections for internal IMP transmit data connection options.

12.3.3.3 SCI+ SERIAL CLOCKS (TXCLK AND RXCLK)

These input only signals receive clocks from which the transmit and/or receive baud rate can be derived in the asynchronous mode and from which data is transferred in the synchronous mode. In asynchronous mode with external baud rate generation, the external baud rate clock must be input on both the TXCLK and RXCLK sources of the SCI+.

The transmit and receive clocks can be asynchronous to each other in synchronous data mode transfers allowing data transfers to be clocked separately. Refer to 7.5.3.2 SCI+ Serial Connections for internal IMP clock connection options.

12.3.4 SCI+ Programming Model

The resources available in the SCI+ are described before discussing specific examples of how the SCI+ is used. The registers comprising the SCI+ are shown in Figure 12-8 and Figure 12-9. These registers are the SCI+ control register (SCR), SCI+ status register (SSR), SCI+ clock control register (SCCR), SCI+ receive data registers (SRX), SCI+ transmit data registers (STX), and the SCI+ transmit data address register (STXA). The SCI+ programming model can be viewed as three types of registers: 1) control – SCR and SCCR in Figure 12-8; 2) status – SSR in Figure 12-8; and 3) data transfer – SRX, STX, and STXA in Figure 12-9. The following paragraphs describe each bit in the programming model.

12.3.4.1 SCI+ CONTROL REGISTER (SCR)

The SCR is a 16-bit read/write register that controls the serial interface operation. Each bit is described in the following paragraphs.

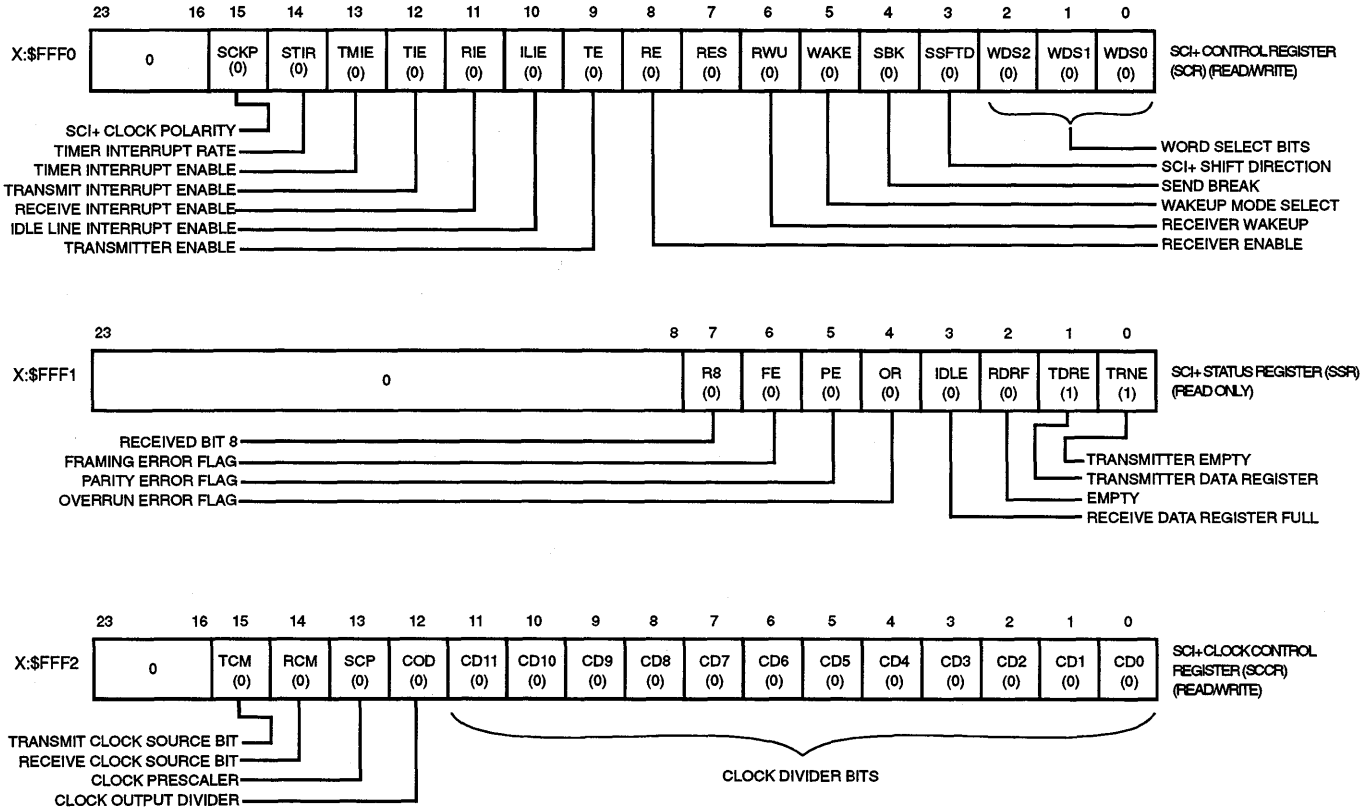
12.3.4.1.1 SCR Word Select (WDS0, WDS1, WDS2) Bits 0, 1, and 2

The three word-select bits (WDS0, WDS1, WDS2) select the format of the transmit and receive data. The formats include three asynchronous, one multidrop asynchronous mode (although wired-or compatible output pins are not available on the MC68356), and an 8-bit synchronous (shift register) mode. The asynchronous modes are compatible with most UART-type serial devices and support standard RS232C communication links when using connection option two described in 12.3.1 SCI+ to IMP Connection Options.

The synchronous data mode is essentially a high-speed shift register used for I/O expansion and stream-mode channel interfaces. The word formats are indicated in Table 12-1 (also see Figure 12-10 and Figure 12-11.

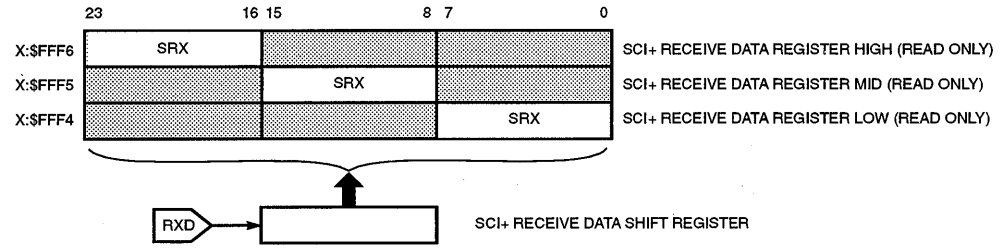
When odd parity is selected, the transmitter will count the number of bits in the data word. If the total is not an odd number, the parity bit is made equal to one and thus produces an odd number. If the receiver counts an even number of ones, an error in transmission has occurred. When even parity is selected, an even number must result from the calculation performed at both ends of the line or an error in transmission has occurred.

The word-select bits are cleared by hardware and software reset.



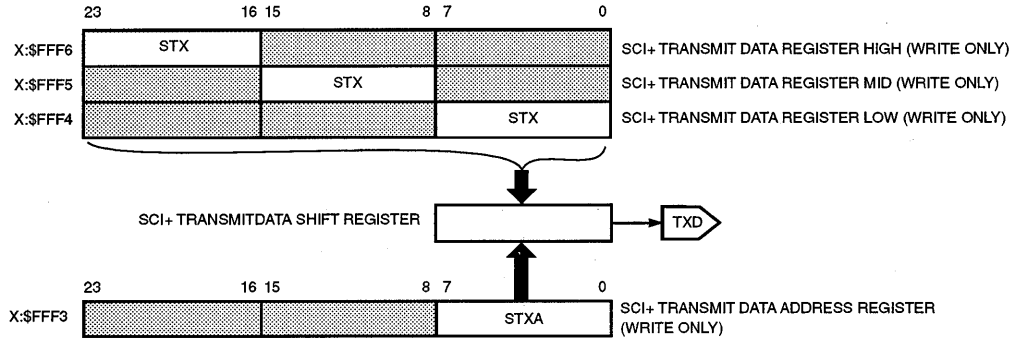
NOTE: The number in parentheses is the condition of the bit after hardware reset.

Figure 12-8. SCI+ Programming Model – Control and Status Registers



NOTE: SRX is the same register decoded at three different addresses.

(a) Receive Data Register



NOTES:
1. Bytes are masked on the fly.
2. STX is the same register decoded at three different addresses.

(b) Transmit Data Register

Figure 12-9. SCI+ Programming Model

Table 12-1. SCI+ Data Word Formats

WDS2	WDS1	WDS0	Word Formats
0	0	0	8-Bit Synchronous Data (shift register mode)
0	0	1	Reserved
0	1	0	10-Bit Asynchronous (1 start, 8 data, 1 stop)
0	1	1	Reserved
1	0	0	11-Bit Asynchronous (1 start, 8 data, 1 even parity, 1 stop)
1	0	1	11-Bit Asynchronous (1 start, 8 data, 1 odd parity, 1 stop)
1	1	0	11-Bit Multidrop (1 start, 8 data, 1 data type, 1 stop)
1	1	1	Reserved

12.3.4.1.2 SCR SCI+ Shift Direction (SSFTD) Bit 3

The SCI+ data shift registers can be programmed to shift data in/out either LSB first if SSFTD equals zero, or MSB first if SSFTD equals one. The parity and data type bits do not change position and remain adjacent to the stop bit. SSFTD is cleared by hardware and software reset.

12.3.4.1.3 SCR Send Break (SBK) Bit 4

A break is an all-zero word frame – a start bit zero, a character of all zeros (including any parity), and a stop bit zero: i.e., 10 or 11 zeros depending on the WDS mode selected. If SBK is set and then cleared, the transmitter completes transmission of any data, sends 10 or 11 zeros, and reverts to idle or sending data. If SBK remains set, the transmitter will continually send whole frames of zeros (10 or 11 bits with no stop bit). At the completion of the break code, the transmitter sends at least one high bit before transmitting any data to guarantee recognition of a valid start bit. Break can be used to signal an unusual condition, message, etc. by forcing a frame error, which is caused by a missing stop bit. Hardware and software reset clear SBK.

12.3.4.1.4 SCR Wakeup Mode Select (WAKE) Bit 5

When WAKE equals zero, an idle line wakeup is selected. In the idle line wakeup mode, the SCI+ receiver is re-enabled by an idle string of at least 10 or 11 (depending on WDS mode) consecutive ones. The transmitter's software must provide this idle string between consecutive messages. The idle string cannot occur within a valid message because each word frame contains a start bit that is a zero.

When WAKE equals one, an address bit wakeup is selected. In the address bit wakeup mode, the SCI+ receiver is re-enabled when the last (eighth or ninth) data bit received in a character (frame) is one. The ninth data bit is the address bit (R8) in the 11-bit multidrop mode; the eighth data bit is the address bit in the 10-bit asynchronous and 11-bit asynchronous with parity modes. Thus, the received character is an address that has to be processed by all sleeping processors – i.e., each processor has to compare the received character with

its own address and decide whether to receive or ignore all following characters. WAKE is cleared by hardware and software reset.

12.3.4.1.5 SCR Receiver Wakeup Enable (RWU) Bit 6

When RWU equals one and the SCI+ is in an asynchronous mode, the wakeup function is enabled – i.e., the SCI+ is put to sleep waiting for a reason (defined by the WAKE bit) to wakeup. In the sleeping state, all receive flags, except IDLE, and interrupts are disabled. When the receiver wakes up, this bit is cleared by the wakeup hardware. The programmer may also clear the RWU bit to wake up the receiver.

RWU can be used by the programmer to ignore messages that are for other devices on a multidrop serial network. Wakeup on idle line (WAKE=0) or wakeup on address bit (WAKE=1) must be chosen.

1. When WAKE equals zero and RWU equals one, the receiver will not respond to data on the data line until an idle line is detected.
2. When WAKE equals one and RWU equals one, the receiver will not respond to data on the data line until a data byte with bit 9 equal to one is detected.

When the receiver wakes up, the RWU bit is cleared, and the first byte of data is received. If interrupts are enabled, the CPU will be interrupted, and the interrupt routine will read the message header to determine if the message is intended for this DSP.

1. If the message is for this DSP, the message will be received, and RWU will again be set to one to wait for the next message.
2. If the message is not for this DSP, the DSP will immediately set RWU to one. Setting RWU to one causes the DSP to ignore the remainder of the message and wait for the next message.

RWU is cleared by hardware and software reset. RWU is a don't care in the synchronous mode.

12.3.4.1.6 Bit 7

Reserved

12.3.4.1.7 SCR Receiver Enable (RE) Bit 8

When RE is set, the receiver is enabled. When RE is cleared, the receiver is disabled, and data transfer is inhibited to the receive data register (SRX) from the receive shift register. If RE is cleared while a character is being received, the reception of the character will be completed before the receiver is disabled. RE does not inhibit RDRF or receive interrupts. RE is cleared by a hardware and software reset.

12.3.4.1.8 SCR Transmitter Enable (TE) Bit 9

When TE is set, the transmitter is enabled. When TE is cleared, the transmitter will complete transmission of data in the SCI+ transmit data shift register; then the serial output is forced high (idle). Data present in the SCI+ transmit data register (STX) will not be transmitted. STX may be written and TDRE will be cleared, but the data will not be transferred into the shift

register. TE does not inhibit TDRE or transmit interrupts. TE is cleared by a hardware and software reset.

Setting TE will cause the transmitter to send a preamble of 10 or 11 consecutive ones (depending on WDS). This procedure gives the programmer a convenient way to ensure that the line goes idle before starting a new message. To force this separation of messages by the minimum idle line time, the following sequence is recommended:

1. Write the last byte of the first message to STX.
2. Wait for TDRE to go high, indicating the last byte has been transferred to the transmit shift register.
3. Clear TE and set TE back to one. This queues an idle line preamble to immediately follow the transmission of the last character of the message (including the stop bit).
4. Write the first byte of the second message to STX.

In this sequence, if the first byte of the second message is not transferred to the STX prior to the finish of the preamble transmission, then the transmit data line will simply mark idle until STX is finally written.

12.3.4.1.9 SCR Idle Line Interrupt Enable (ILIE) Bit 10

When ILIE is set, the SCI+ interrupt occurs when IDLE is set. When ILIE is clear, the IDLE interrupt is disabled. ILIE is cleared by hardware and software reset.

An internal flag, the shift register idle interrupt (SRIINT) flag, is the interrupt request to the interrupt controller. SRIINT is not directly accessible to the user.

When a valid start bit has been received, an idle interrupt will be generated if both IDLE (SCI+ status register bit 3) and ILIE equals one. The idle interrupt acknowledge from the interrupt controller clears this interrupt request. The idle interrupt will not be asserted again until at least one character has been received. The result is as follows:

1. The IDLE bit shows the real status of the receive line at all times.
2. Idle interrupt is generated once for each idle state, no matter how long the idle state lasts.

12.3.4.1.10 SCR SCI+ Receive Interrupt Enable (RIE) Bit 11

The RIE bit is used to enable the SCI+ receive data interrupt. If RIE is cleared, receive interrupts are disabled, and the RDRF bit in the SCI+ status register must be polled to determine if the receive data register is full. If both RIE and RDRF are set, the SCI+ will request an SCI+ receive data interrupt from the interrupt controller.

One of two possible receive data interrupts will be requested:

1. Receive without exception will be requested if PE, FE, and OR are all clear (i.e., a normal received character).
2. Receive with exception will be requested if PE, FE, and OR are not all clear (i.e., a received character with an error condition).

RIE is cleared by hardware and software reset.

12.3.4.1.11 SCR SCI+ Transmit Interrupt Enable (TIE) Bit 12

The TIE bit is used to enable the SCI+ transmit data interrupt. If TIE is cleared, transmit data interrupts are disabled, and the transmit data register empty (TDRE) bit in the SCI+ status register must be polled to determine if the transmit data register is empty. If both TIE and TDRE are set, the SCI+ will request an SCI+ transmit data interrupt from the interrupt controller. TIE is cleared by hardware and software reset.

12.3.4.1.12 SCR Timer Interrupt Enable (TMIE) Bit 13

The TMIE bit is used to enable the SCI+ timer interrupt. If TMIE is set (enabled), the timer interrupt requests will be made to the interrupt controller at the rate set by the SCI+ clock register. The timer interrupt is automatically cleared by the timer interrupt acknowledge from the interrupt controller. This feature allows DSP programmers to use the SCI+ baud clock generator as a simple periodic interrupt generator if the SCI+ is not in use, if external clocks are used for the SCI+, or if periodic interrupts are needed at the SCI+ baud rate. The SCI+ internal clock is divided by 16 (to match the 1 × SCI+ baud rate) for timer interrupt generation. This timer does not require that any SCI+ pins be configured for SCI+ use to operate. TMIE is cleared by hardware and software reset.

12.3.4.1.13 SCR SCI+ Timer Interrupt Rate (STIR) Bit 14

This bit controls a divide by 32 in the SCI+ timer interrupt generator. When this bit is cleared, the divide by 32 is inserted in the chain. When the bit is set, the divide by 32 is bypassed, thereby increasing the timer resolution by 32 times. This bit is cleared by hardware and software reset.

12.3.4.1.14 SCR SCI+ Clock Polarity (SCKP) Bit 15

The clock polarity received on the corresponding clock signal (RXCLK or TXCLK), can be inverted using this bit, eliminating the need for an external inverter. When bit 15 equals zero, the clock polarity is positive; when bit 15 equals one, the clock polarity is negative. In the synchronous mode, positive polarity means that the clock is normally positive and transitions negative during data valid; whereas, negative polarity means that the clock is normally negative and transitions positive during valid data. In the asynchronous mode, positive polarity means that the rising edge of the clock occurs in the center of the period that data is valid; negative polarity means that the falling edge of the clock occurs during the center of the period that data is valid. SCKP is cleared on hardware and software reset.

12.3.4.2 SCI+ STATUS REGISTER (SSR)

The SSR is an 8-bit read-only register used by the DSP CPU to determine the status of the SCI+. When the SSR is read onto the internal data bus, the register contents occupy the low-order byte of the data bus and all high-order portions are zero filled. The status bits are described in the following paragraphs.

12.3.4.2.1 SSR Transmitter Empty (TRNE) Bit 0

The TRNE flag is set when both the transmit shift register and data register are empty to indicate that there is no data in the transmitter. When TRNE is set, data written to one of the three STX locations or to the STXA will be transferred to the transmit shift register and be the first data transmitted. TRNE is cleared when TDRE is cleared by writing data into the transmit data register (STX) or the transmit data address register (STXA), or when an idle,

preamble, or break is transmitted. The purpose of this bit is to indicate that the transmitter is empty; therefore, the data written to STX or STXA will be transmitted next – i.e., there is not a word in the transmit shift register presently being transmitted. This procedure is useful when initiating the transfer of a message (i.e., a string of characters). TRNE is set by the hardware, software, SCI+ individual, and stop reset.

12.3.4.2.2 SSR Transmit Data Register Empty (TDRE) Bit 1

The TDRE bit is set when the SCI+ transmit data register is empty. When TDRE is set, new data may be written to one of the SCI+ transmit data registers (STX) or transmit data address register (STXA). TDRE is cleared when the SCI+ transmit data register is written. TDRE is set by the hardware, software, SCI+ individual, and stop reset.

In the SCI+ synchronous mode, when using the internal SCI+ clock, there is a delay of up to 5.5 serial clock cycles between the time that STX is written until TDRE is set, indicating the data has been transferred from the STX to the transmit shift register. There is a two to four serial clock cycle delay between writing STX and loading the transmit shift register; in addition, TDRE is set in the middle of transmitting the second bit. When using an external serial transmit clock, if the clock stops, the SCI+ transmitter stops. TDRE will not be set until the middle of the second bit transmitted after the external clock starts. Gating the external clock off after the first bit has been transmitted will delay TDRE indefinitely.

In the SCI+ asynchronous mode, the TDRE flag is not set immediately after a word is transferred from the STX or STXA to the transmit shift register nor when the word first begins to be shifted out. TDRE is set two cycles of the 16× clock after the start bit – i.e., two 16× clock cycles into to transmission time of the first data bit.

12.3.4.2.3 SSR Receive Data Register Full (RDRF) Bit 2

The RDRF bit is set when a valid character is transferred to the SCI+ receive data register from the SCI+ receive shift register. RDRF is cleared when the SCI+ receive data register is read or by the hardware, software, SCI+ individual, and stop reset.

12.3.4.2.4 SSR Idle Line Flag (IDLE) Bit 3

IDLE is set when 10 (or 11) consecutive ones are received. IDLE is cleared by a start-bit detection. The IDLE status bit represents the status of the receive line. The transition of IDLE from zero to one can cause an IDLE interrupt (ILIE). IDLE is cleared by the hardware, software, SCI+ individual, and stop reset.

12.3.4.2.5 SSR Overrun Error Flag (OR) Bit 4

The OR flag is set when a byte is ready to be transferred from the receive shift register to the receive data register (SRX) that is already full (RDRF=1). The receive shift register data is not transferred to the SRX. The OR flag indicates that character(s) in the receive data stream may have been lost. The only valid data is located in the SRX. OR is cleared when the SCI+ status register is read, followed by a read of SRX. The OR bit clears the FE and PE bits – i.e., overrun error has higher priority than FE or PE. OR is cleared by the hardware, software, SCI+ individual, and stop reset.

12.3.4.2.6 SSR Parity Error (PE) Bit 5

In the 11-bit asynchronous modes, the PE bit is set when an incorrect parity bit has been detected in the received character. It is set simultaneously with RDRF for the byte which contains the parity error – i.e., when the received word is transferred to the SRX. If PE is set, it does not inhibit further data transfer into the SRX. PE is cleared when the SCI+ status register is read, followed by a read of SRX. PE is also cleared by the hardware, software, SCI+ individual, or stop reset. In the 10-bit asynchronous mode, the 11-bit multidrop mode, and the 8-bit synchronous mode, the PE bit is always cleared since there is no parity bit in these modes. If the byte received causes both parity and overrun errors, the SCI+ receiver will only recognize the overrun error.

12.3.4.2.7 SSR Framing Error Flag (FE) Bit 6

The FE bit is set in the asynchronous modes when no stop bit is detected in the data string received. FE and RDRE are set simultaneously – i.e., when the received word is transferred to the SRX. However, the FE flag inhibits further transfer of data into the SRX until it is cleared. FE is cleared when the SCI+ status register is read followed by reading the SRX. The hardware, software, SCI+ individual, and stop reset also clear FE. In the 8-bit synchronous mode, FE is always cleared. If the byte received causes both framing and overrun errors, the SCI+ receiver will only recognize the overrun error.

12.3.4.2.8 SSR Received Bit 8 Address (R8) Bit 7

In the 11-bit asynchronous multidrop mode, the R8 bit is used to indicate whether the received byte is an address or data. R8 is not affected by reading the SRX or status register. The hardware, software, SCI+ individual, and stop reset clear R8.

12.3.4.3 SCI+ CLOCK CONTROL REGISTER (SCCR)

The SCCR is a 16-bit read/write register which controls the selection of the clock modes and baud rates for the transmit and receive sections of the SCI+ interface. The control bits are described in the following paragraphs. The SCCR is cleared by hardware reset.

The basic points of the clock generator are as follows:

1. The SCI+ core always uses a 16 × internal clock in the asynchronous modes and always uses a 2 × internal clock in the synchronous mode. The maximum internal clock available to the SCI+ peripheral block is the oscillator frequency divided by 4. With a 40-MHz crystal, this gives a maximum data rate of 625 Kbps for asynchronous data and 5 Mbps for synchronous data. These maximum rates are the same for internally or externally supplied clocks.
2. The 16 × clock is necessary for the asynchronous modes to synchronize the SCI+ to the incoming data (see Figure 12-10).
3. For the asynchronous modes, the user must provide a 16 × clock if he wishes to use an external baud rate generator. This clock must be input on both the TXCLK and RXCLK signal input).
4. For the asynchronous modes, the user may select either 1 × or 16 × for the output clock when using internal TX and RX clocks (TCM=0 and RCM=0).
5. The transmit data on the TXD pin changes on the negative edge of the 1 × serial clock

and is stable on the positive edge (SCKP=0). For SCKP equals one, the data changes on the positive edge and is stable on the negative edge.

6. The receive data on the RXD pin is sampled on the positive edge (if SCKP=0) or on the negative edge (if SCKP=1) of the 1 × serial clock.
7. For the asynchronous mode, the output clock is continuous.
8. For the synchronous mode, a 1 × clock is used for the output or input baud rate. The maximum 1 × clock is the crystal frequency divided by 8.
9. For the synchronous mode, the clock is gated.
10. For both the asynchronous and synchronous modes, the transmitter and receiver are synchronous with each other.

12.3.4.3.1 SCCR Clock Divider (CD11–CD0) Bits 11–0

The clock divider bits (CD11–CD0) are used to preset a 12-bit counter, which is decremented at the I_{cyc} rate (crystal frequency divided by 2). The counter is not accessible to the user. When the counter reaches zero, it is reloaded from the clock divider bits. Thus, a value of 0000 0000 0000 in CD11–CD0 produces the maximum rate of I_{cyc}, and a value of 0000 0000 0001 produces a rate of $I_{cyc}/2$. The lowest rate available is $I_{cyc}/4096$. Figure 12-11 and Figure 12-30 show the clock dividers. Bits CD11–CD0 are cleared by hardware and software reset.

12.3.4.3.2 SCCR Clock Out Divider (COD) Bit 12

This bit does not have any effect on SCI+ operation since the SCI+ cannot drive external clocks. This bit can be written with a zero.

The COD bit is cleared by hardware and software reset.

12.3.4.3.3 SCCR SCI+ Clock Prescaler (SCP) Bit 13

The SCI+ SCP bit selects a divide by 1 (SCP=0) or divide by 8 (SCP=1) prescaler for the clock divider. The output of the prescaler is further divided by 2 to form the SCI+ clock. Hardware and software reset clear SCP. Figure 12-11 and Figure 12-30 show the clock divider diagram.

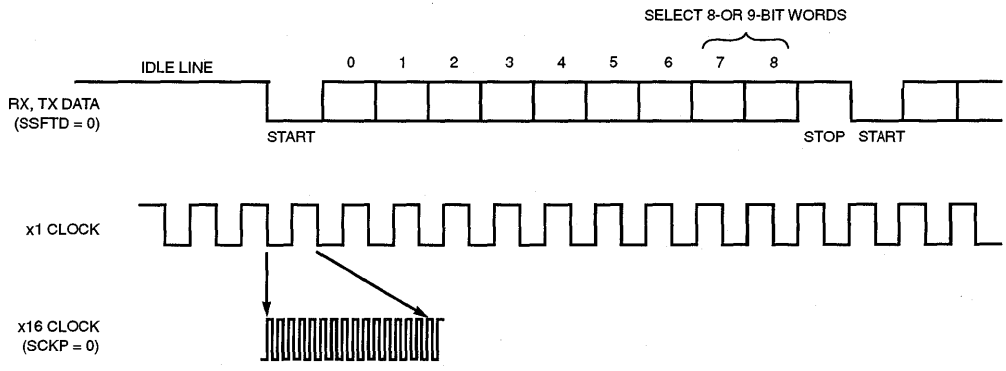
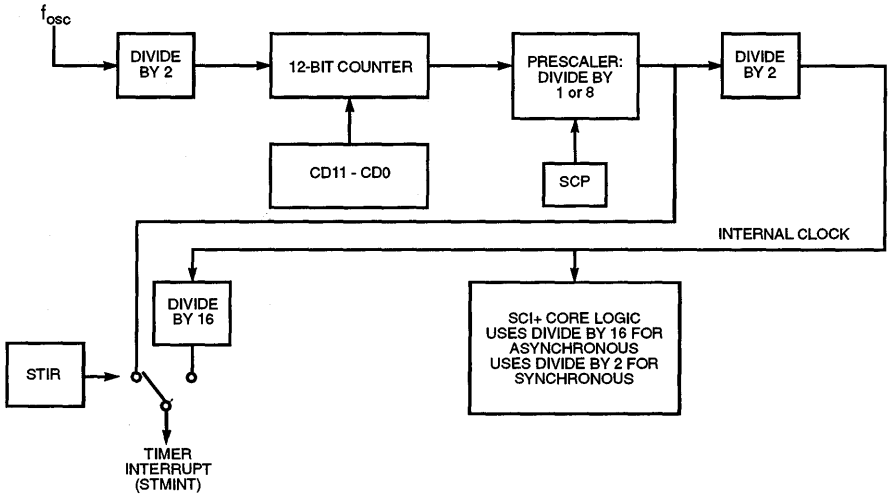


Figure 12-10. 16 x Serial Clock

TCM	RCM	TXCLK	RXCLK	Mode
0	1	Internal	External	Asynchronous Only
1	0	External	Internal	Asynchronous Only
1	1	External	External	Synchronous/Asynchronous



BPS = fo/(64 x (7(SCP) + 1) x CD + 1))

where: SCP = 0 or 1
CD = 0 to $FFF

12

Figure 12-11. SCI+ Baud Rate Generator

12.3.4.3.4 SCCR Receive Clock Mode Source Bit (RCM) Bit 14

RCM selects internal or external clock for the receiver (see Figure 12-30). RCM equals zero selects the internal clock; RCM equals one selects the external clock from the RXCLK signal. Hardware and software reset clear RCM.

12.3.4.3.5 SCCR Transmit Clock Source Bit (TCM) Bit 15

The TCM bit selects internal or external clock for the transmitter (see Figure 12-30). TCM equals zero selects the internal clock; TCM equals one selects the external clock from the TXCLK signal. Hardware and software reset clear TCM.

12.3.4.4 SCI+ DATA REGISTERS

The SCI+ data registers are divided into two groups: receive and transmit. There are two receive registers – a receive data register (SRX) and a serial-to-parallel receive shift regis-

ter. There are also two transmit registers – a transmit data register (called either STX or STXA) and a parallel-to-serial transmit shift register.

12.3.4.4.1 SCI+ Receive Registers

Data words received on the RXD pin are shifted into the SCI+ receive shift register. When the complete word has been received, the data portion of the word is transferred to the byte-wide SRX. This process converts the serial data to parallel data and provides double buffering. Double buffering provides flexibility and increased throughput since the programmer can save the previous word while the current word is being received.

The SRX can be read at three locations: X:$FFF4, X:$FFF5, and X:$FFF6 (see Figure 12-12). When location X:$FFF4 is read, the contents of the SRX are placed in the lower byte of the data bus and the remaining bits on the data bus are written as zeros. Similarly, when X:$FFF5 is read, the contents of SRX are placed in the middle byte of the bus, and when X:$FFF6 is read, the contents of SRX are placed in the high byte with the remaining bits zeroed. Mapping SRX as described allows three bytes to be efficiently packed into one 24-bit word by "OR"-ing three data bytes read from the three addresses. The following code fragment requires that R0 initially points to X:$FFF4, register A is initially cleared, and R3 points to a data buffer. The only programming trick is using BCLR to test bit 1 of the packing pointer to see if it is pointing to X:$FFF6 and clearing bit 1 to point to X:$FFF4 if it had been pointing to X:$FFF6. This procedure resets the packing pointer after receiving three bytes.

```
MOVE    X:(R0),X0     ;Copy received data to temporary register
BCLR    #$1,R0        ;Test for last byte
                      ;reset pointer if it is the last byte
OR      X0,A          ;Pack the data into register A
MOVE    (R0)+         ;and increment the packing pointer
JCS     FLAG          ;Jump to clean up routine if last byte
RTI                   ;Else return until next byte is received
FLAG    MOVE A,(R3)+  ;Move the packed data to memory
CLR     A             ;Prepare A for packing next three bytes
RTI                   ;Return until the next byte is received
```

The length and format of the serial word is defined by the WDS0, WDS1, and WDS2 control bits in the SCI+ control register. In the synchronous modes, the start bit, the eight data bits with LSB first, the address/data indicator bit and/or the parity bit, and the stop bit are received in that order for SSFTD equals zero (see Figure 12-10). For SSFTD equals one, the data bits are transmitted MSB first (see Figure 12-11). The clock source is defined by the receive clock mode (RCM) select bit in the SCR. In the synchronous mode, the synchronization is provided by gating the clock. In either mode, when a complete word has been clocked in, the contents of the shift register can be transferred to the SRX and the flags RDRF, FE, PE, and OR are changed appropriately. Because the operation of the SCI+ receive shift register is transparent to the DSP, the contents of this register are not directly accessible to the programmer.

12.3.4.4.2 SCI+ Transmit Registers

The transmit data register is one byte-wide register mapped into four addresses: X:$FFF3, X:$FFF4, X:$FFF5, and X:$FFF6. In the asynchronous mode, when data is to be transmitted, X:$FFF4, X:$FFF5, and X:$FFF6 are used, and the register is called STX. When

X:$FFF4 is written, the low byte on the data bus is transferred to the STX; when X:$FFF5 is written, the middle byte is transferred to the STX; and when X:$FFF6 is written, the high byte is transferred to the STX. This structure (see Figure 12-9) makes it easy for the programmer to unpack the bytes in a 24-bit word for transmission. Location X:$FFF3 should be written in the 11-bit asynchronous multidrop mode when the data is an address and it is desired that the ninth bit (the address bit) be set. When X:$FFF3 is written, the transmit data register is called STXA, and data from the low byte on the data bus is stored in STXA. The address data bit will be cleared in the 11-bit asynchronous multidrop mode when any of X:$FFF4, X:$FFF5, or X:$FFF6 is written. When either STX or STXA is written, TDRE is cleared.

The transfer from either STX or STXA to the transmit shift register occurs automatically, but not immediately, when the last bit from the previous word has been shifted out – i.e., the transmit shift register is empty. Like the receiver, the transmitter is double buffered. However, there will be a two to four serial clock cycle delay between when the data is transferred from either STX or STXA to the transmit shift register and when the first bit appears on the TXD pin. (A serial clock cycle is the time required to transmit one data bit). The transmit shift register is not directly addressable, and a dedicated flag for this register does not exist. Because of this fact and the two to four cycle delay, two bytes cannot be written consecutively to STX or STXA without polling. The second byte will overwrite the first byte. The TDRE flag should always be polled prior to writing STX or STXA to prevent overruns unless transmit interrupts have been enabled. Either STX or STXA is usually written as part of the interrupt

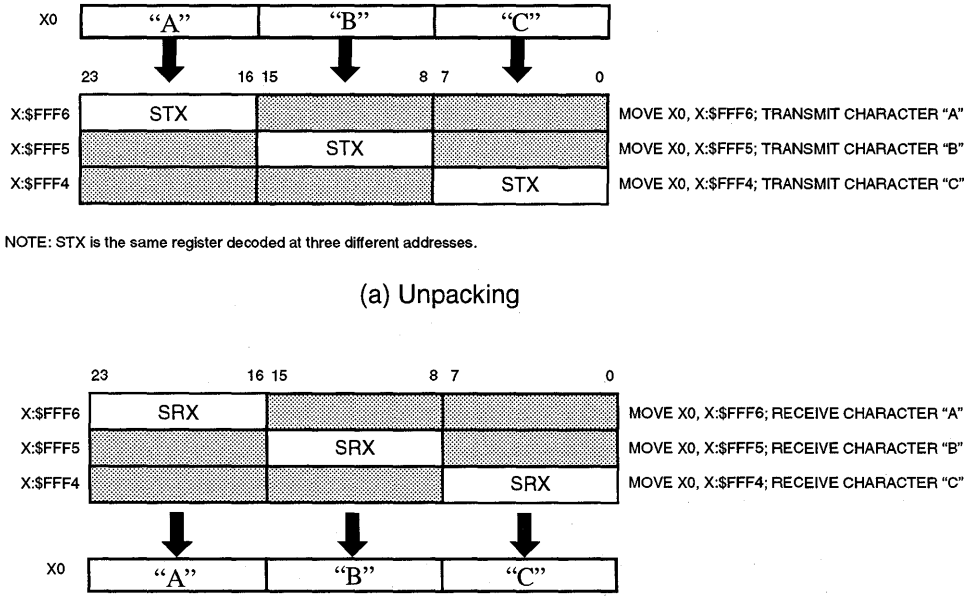


Figure 12-12. Data Packing and Unpacking

service routine. Of course, the interrupt will only be generated if TDRE equals one. The transmit shift register is indirectly visible via the TRNE bit in the SSR.

In the synchronous modes, data is synchronized with the transmit clock, which may have either an internal or external source as defined by the TCM bit in the SCCR. The length and format of the serial word is defined by the WDS0, WDS1, and WDS2 control bits in the SCR. In the asynchronous modes, the start bit, the eight data bits (with the LSB first if SSFTD=0 and the MSB first if SSFTD=1), the address/data indicator bit or parity bit, and the stop bit are transmitted in that order (see Figure 12-10).

The data to be transmitted can be written to any one of the three STX addresses. If SCKP equals one and SSHTD equals one, the SCI+ synchronous mode is equivalent to the SSI operation in the 8-bit data on-demand mode.

12.3.4.5 PREAMBLE, BREAK, AND DATA TRANSMISSION PRIORITY

It is possible that two or three transmission commands are set simultaneously:

1. A preamble (TE was toggled)
2. A break (SBK was set or was toggled)
3. There is data for transmission (TDRE=0)

After the current character transmission, if two or more of these commands are set, the transmitter will execute them in the following priority:

1. Preamble
2. Break
3. Data

12.3.5 Register Contents After Reset

There are four methods to reset the SCI+. Hardware or software reset clears the port control register bits, which will disable the SCI+. The SCI+ will remain in the reset state while CC2, CC1, and CC0=0; the SCI+ will become active only when at least one of the CC2-0 bits is not zero.

During program execution, the CC2, CC1, and CC0 bits may be cleared (individual reset), which will cause the SCI+ to stop serial activity and enter the reset state. All SCI+ status bits will be set to their reset state; however, the contents of the interface control register are not affected, allowing the DSP program to reset the SCI+ separately from the other internal peripherals.

The STOP instruction halts operation of the SCI+ until the DSP is restarted, causing the SSR to be reset. No other SCI+ registers are affected by the STOP instruction. Table 12-2 illustrates how each type of reset affects each register in the SCI+.

12.3.6 SCI+ Initialization

The correct way to initialize the SCI+ is as follows:

1. Hardware or software reset

Table 12-2. SCI+ Registers After Reset

Register Bit	Bit Mnemonic	Bit Number	Reset Type			
			HW Reset	SW Reset	IR Reset	ST Reset
SCR	SCKP	15	0	0	–	–
	STIR	14	0	0	–	–
	TMIE	13	0	0	–	–
	TIE	12	0	0	–	–
	RIE	11	0	0	–	–
	ILIE	10	0	0	–	–
	TE	9	0	0	–	–
	RE	8	0	0	–	–
	RES	7	0	0	–	–
	RWU	6	0	0	–	–
	WAKE	5	0	0	–	–
	SBK	4	0	0	–	–
	SSFTD	3	0	0	–	–
	WDS (2–0)	2–0	0	0	–	–
SSR	R8	7	0	0	0	0
	FE	6	0	0	0	0
	PE	5	0	0	0	0
	OR	4	0	0	0	0
	IDLE	3	0	0	0	0
	RDRF	2	0	0	0	0
	TDRE	1	1	1	1	1
	TRNE	0	1	1	1	1
SCCR	TCM	15	0	0	–	–
	RCM	14	0	0	–	–
	SCP	13	0	0	–	–
	COD	12	0	0	–	–
	CD (11–0)	11–0	0	0	–	–
SRX	SRX (23–0)	23–16, 15–8, 7–0	–	–	–	–
STX	STX (23–0)	23–0	–	–	–	–
SRSH	SRS (8–0)	8–0	–	–	–	–
STSH	STS (8–0)	8–0	–	–	–	–

NOTES:
SRSH – SCI+ receive shift register, STSH – SCI+ transmit shift register
HW – Hardware reset is caused by asserting the external $\overline{\text{DRESET}}$ pin.
SW – Software reset is caused by executing the RESET instruction.
IR – Individual reset is caused by clearing PCC (bits 0–2).
ST – Stop reset is caused by executing the STOP instruction.
1 – The bit is set during the reset.
0 – The bit is cleared during the reset.
– – The bit is not changed during the reset.

2. Program SCI+ control registers
3. Configure at least one Port C control bit 0-2 as SCI+.
4. Configure the IMP-SCI+ connection in the IMP DISC register.

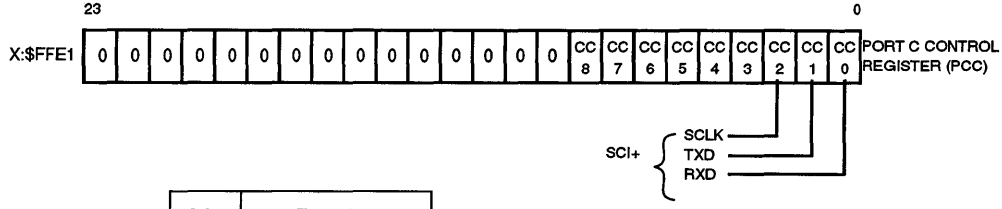
Figure 12-13 and Figure 12-14 show how to configure the bits in the SCI+ registers. Figure 12-13 is the basic initialization procedure showing which registers must be configured. (1) A

1. PERFORM HARDWARE OR SOFTWARE RESET.

2. PROGRAM SCI+ CONTROL REGISTERS:
   a) SCI+ INTERFACE CONTROL REGISTER — X:$FFF0
   b) SCI+ CLOCK CONTROL REGISTER — X:$FFF2

3. CONFIGURE AT LEAST ONE PORT C CONTROL BIT AS SCI+.

4. CONFIGURE THE IMP-SCI+ CONNECTION IN THE IMP DISC REGISTER.



CCx	Function
0	GPIO
1	Serial Interface

5. SCI+ IS NOW ACTIVE.

Figure 12-13. SCI+ Initialization Procedure

hardware or software reset should be used to reset the SCI+ and prevent it from doing anything unexpected while it is being programmed. (2) Both the SCI+ interface control register and the clock control register must be configured for any operation using the SCI+. (3) The pins to be used must then be selected to release the SCI+ from reset, the SCI+ to IMP connection must be established (see 7.5.3 DSP Interconnection and Serial Connections Register–DISC) and (4) Begin operation. If interrupts are to be used, the pins must be selected, and interrupts must be enabled and unmasked before the SCI+ will operate. The order does not matter; any one of these three requirements for interrupts can be used to finally enable the SCI+.

12

Figure 12-14 shows the meaning of the individual bits in the SCR and SCCR. The figures below do not assume that interrupts will be used; they recommend selecting the appropriate pins to enable the SCI+. Programs shown in Figures 12-19, 12-20, 12-27,12-28, and 12-31 control the SCI+ by enabling and disabling interrupts. Either method is acceptable.

Table 12-3 through Table 12-6 provide the settings for common baud rates for the SCI+. The asynchronous SCI+ baud rates show a baud rate error for the fixed oscillator frequency (see Table 12-3). These small-percentage baud rate errors should allow most UARTs to synchronize. The synchronous applications usually require exact frequencies, which require that the crystal frequency be chosen carefully (see Table 12-4 and Table 12-6).

An alternative to selecting the system clock to accommodate the SCI+ requirements is to provide an external clock to the SCI+. For example, a 2.048 MHz bit rate requires a CPU clock of 32.768 MHz. An application may need a 40 MHz CPU clock and an external clock for the SCI+

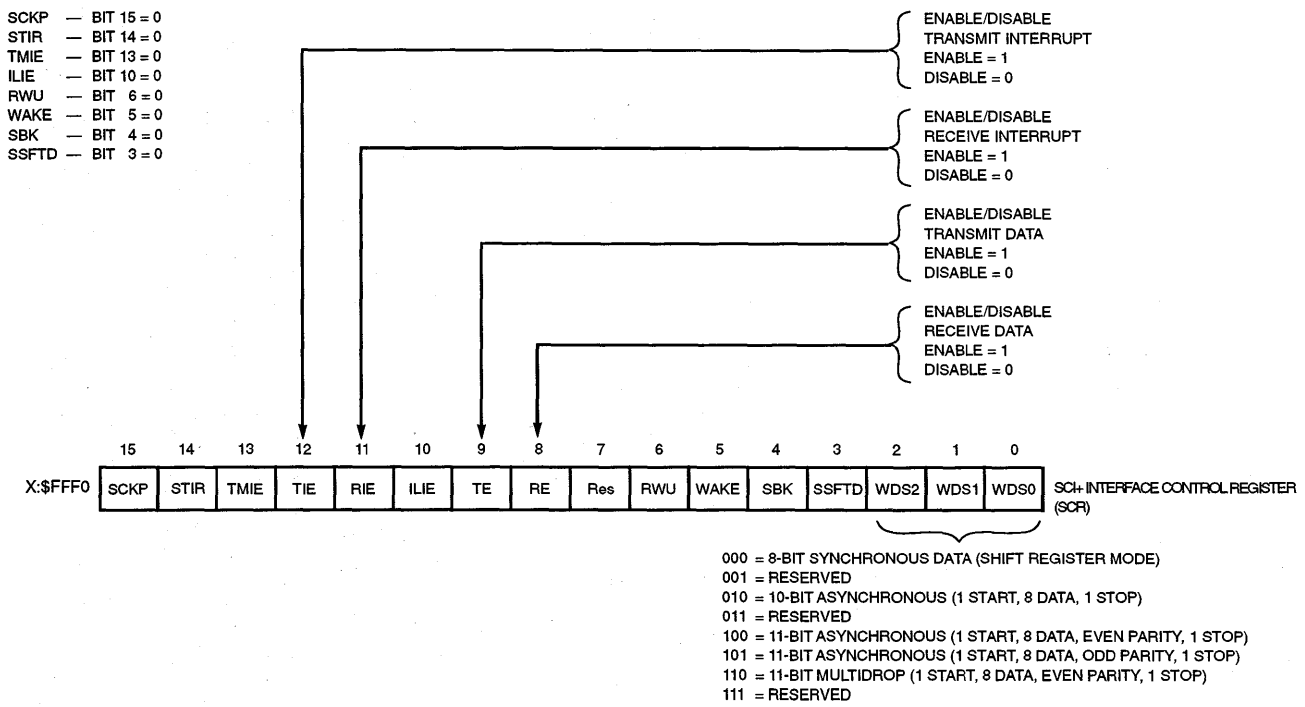


Figure 12-14. SCI+ General Initialization Detail (Step 2a)

STEP 2b. SELECT CLOCK AND DATA RATE:
SET THE CLOCK DIVIDER BITS (CD0 - CD11) ACCORDING TO TABLES 11 - 2 OR 11 - 3.
SET THE SCI+ CLOCK PRESCALER BIT (SCP, BIT 13) ACCORDING TO TABLES 11 - 2 OR 11 - 3.

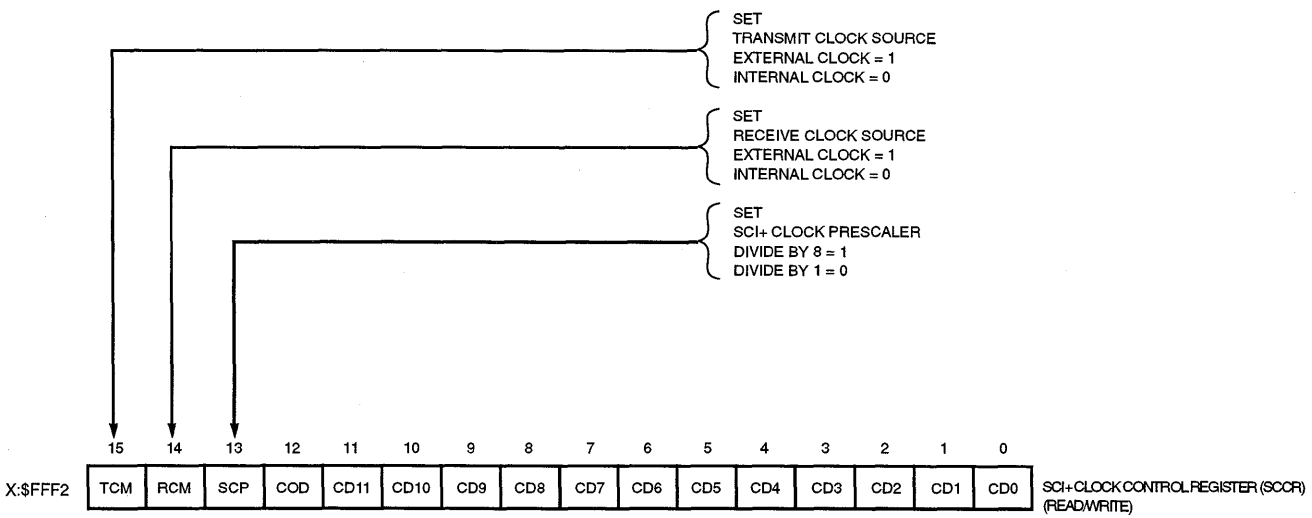


Figure 12-15. SCI+ General Initialization Detail (Step 2b)

Table 12-3. Asynchronous SCI+ Bit Rates for a 40-MHz Crystal

Bit Rate (BPS)	SCP Bit	Divider Bits (CD0–CD11)	Bit Rate Error, Percent
625.0K	0	$000	0
56.0K	0	$00A	+1.46
38.4K	0	$00F	+1.72
19.2K	0	$020	-1.36
9600	0	$040	+0.16
8000	0	$04D	+0.15
4800	0	$081	+0.15
2400	1	$020	-1.38
1200	1	$040	+0.08
600	1	$081	0
300	1	$103	0

f0=BPS X 64X (7(SCP) + 1)X(CD + 1))
SCP=0 or 1
CD=0 to $FFF

Table 12-4. Frequencies for Exact Asynchronous SCI+ Bit Rates

Bit Rate (BPS)	SCP Bit	Divider Bits (CD0–CD11)	Crystal Frequency
9600	0	$040	39,936,000
4800	0	$081	39,936,000
2400	0	$103	39,936,000
1200	0	$207	39,936,000
300	0	$822	39,993,000
9600	1	$007	39,321,600
4800	1	$00F	39,321,600
2400	1	$01F	39,321,600
1200	1	$040	39,360,000
300	1	$103	39,936,000

BPS= $f_0 \div \Pi$ (64X (7(SCP) + 1) X (CD + 1)); f_0=40 MHz
SCP=0 or 1
CD=0 to $FFF

Table 12-5. Synchronous SCI+ Bit Rates for a 32.768-MHz Crystal

Baud Rate (BPS)	SCP Bit	Divider Bits (CD0–CD11)	Baud Rate Error, Percent
4.096M	0	$000	0
128K	0	$01F	0
64K	0	$03F	0
56K	0	$048	-0.195
32K	0	$07F	0
16K	0	$0FF	0
8000	0	$1FF	0
4000	0	$3FF	0
2000	0	$7FF	0
1000	0	$FFF	0

$BPS = f_0 \div (8 \times (7(SCP) + 1) \times (CD + 1))$; f_0=32.768 MHz
SCP=0 or 1
CD=0 to $FFF

Table 12-6. Frequencies for Exact Synchronous SCI+ Bit Rates

Bit Rate (BPS)	SCP Bit	Divider Bits (CD0–CD11)	Clock Frequency, MHz
2.048M	0	$001	32.768
1.544M	0	$002	37.056
1.536M	0	$002	36.864

$f_0 = BPS \times 8 \times (7(SCP) + 1) \times (CD + 1)$
SCP=0 or 1
CD=0 to $FFF

12.3.7 SCI+ Exceptions

The SCI+ can cause five different exceptions in the DSP (see Figure 12-46). These exceptions are as follows:

1. SCI+ Receive Data – caused by receive data register full with no receive error conditions existing. This error-free interrupt may use a fast interrupt service routine for min-

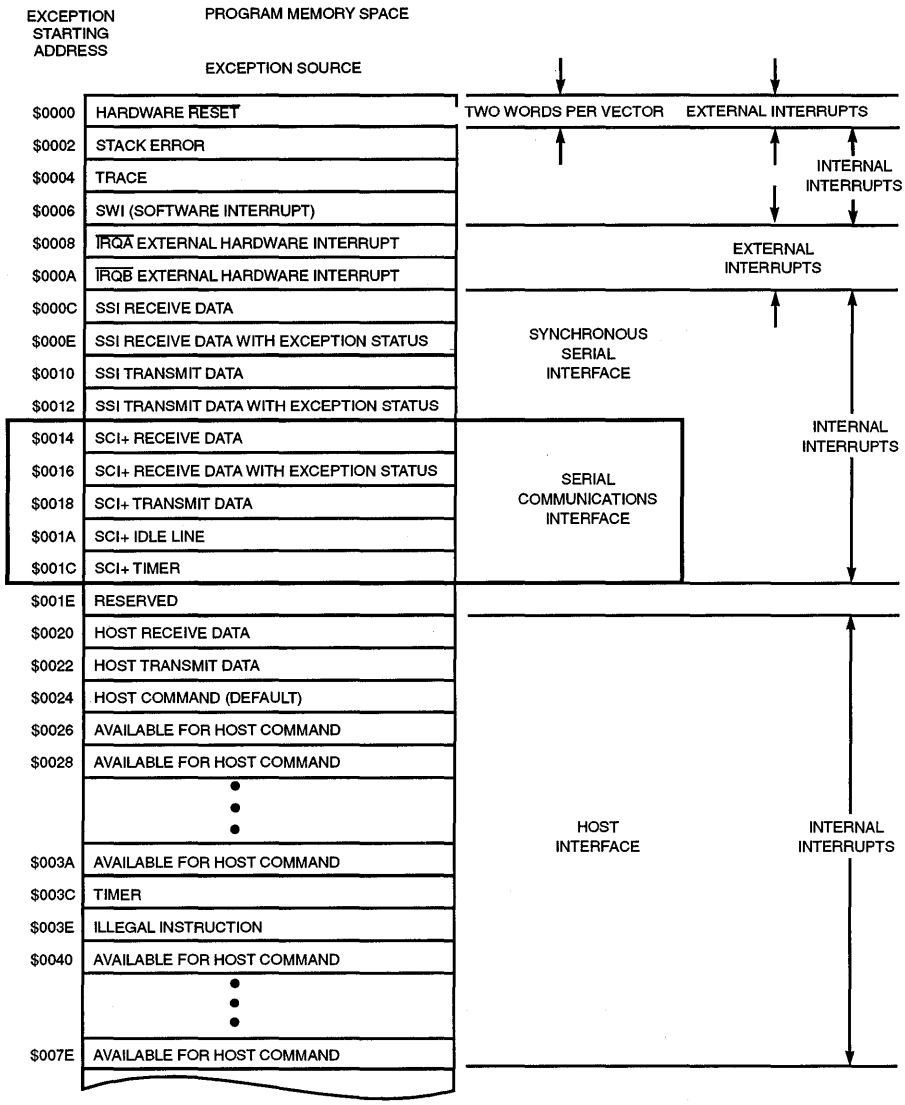


Figure 12-16. SCI+ Exception Vector Locations

imum overhead. This interrupt is enabled by SCR bit 11 (RIE).

2. SCI+ Receive Data with Exception Status – caused by receive data register full with a receiver error (parity, framing, or overrun error). The SCI+ status register must be read to clear the receiver error flag. A long interrupt service routine should be used to handle the error condition. This interrupt is enabled by SCR bit 11 (RIE).
3. SCI+ Transmit Data – caused by transmit data register empty. This error-free interrupt may use a fast interrupt service routine for minimum overhead. This interrupt is enabled by SCR bit 12 (TIE).

4. SCI+ Idle Line – occurs when the receive line enters the idle state (10 or 11 bits of ones). This interrupt is latched and then automatically reset when the interrupt is accepted. This interrupt is enabled by SCR bit 10 (ILIE).
5. SCI+ Timer – caused by the baud rate counter underflowing. This interrupt is automatically reset when the interrupt is accepted. This interrupt is enabled by SCR bit 13 (TMIE).

12.3.8 Synchronous Data

The synchronous mode (WDS=0, shift register mode) is designed to implement serial-to-parallel and parallel-to-serial conversions.

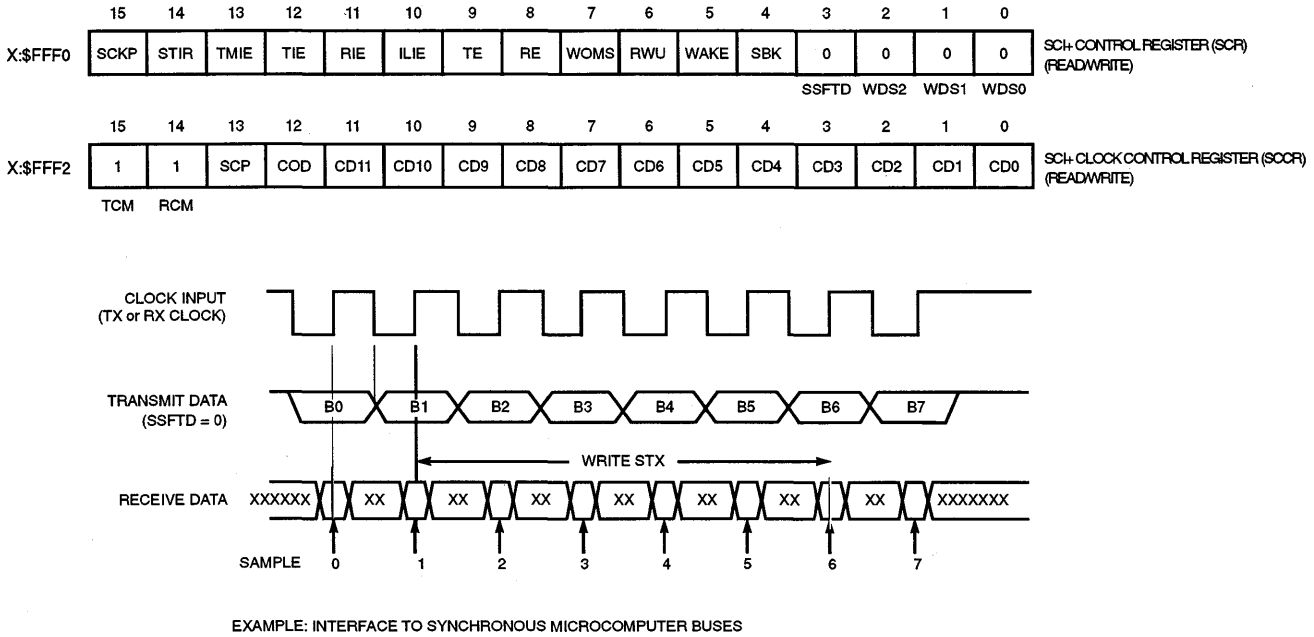
The SCI+ port on the MC68356 must always operate as a peripheral in a synchronous slave mode as shown in Figure 12-17. If SCKP equals zero, data is clocked in on the rising edge of RXCLK, and data is clocked out on the falling edge of TXCLK. If SCKP equals one, data is clocked in on the falling edge of RXCLK, and data is clocked out on the rising edge of TXCLK. The slave mode is selected by choosing external transmit and receive clocks (TCM and RCM=1). Since there is no frame signal, if a clock is missed due to noise or any other reason, the receiver will lose synchronization with the data without any error signal being generated. Detecting an error of this type can be done with an error detecting protocol or with external circuitry such as a watchdog timer. The simplest way to recover synchronization is to reset the SCI+.

The timing diagram in Figure 12-17 shows transmit data in the normal driven mode. There is a window during which STX must be written with the next byte to be transmitted to prevent the current word from being retransmitted. This window is from the time TDRE goes high, which is halfway into the transmission of bit 1, until the middle of bit 6 (see Figure 12-18). Of course, this assumes the clock remains continuous – i.e., there is a second word. If the clock stops, the SCI+ stops.

NOTE

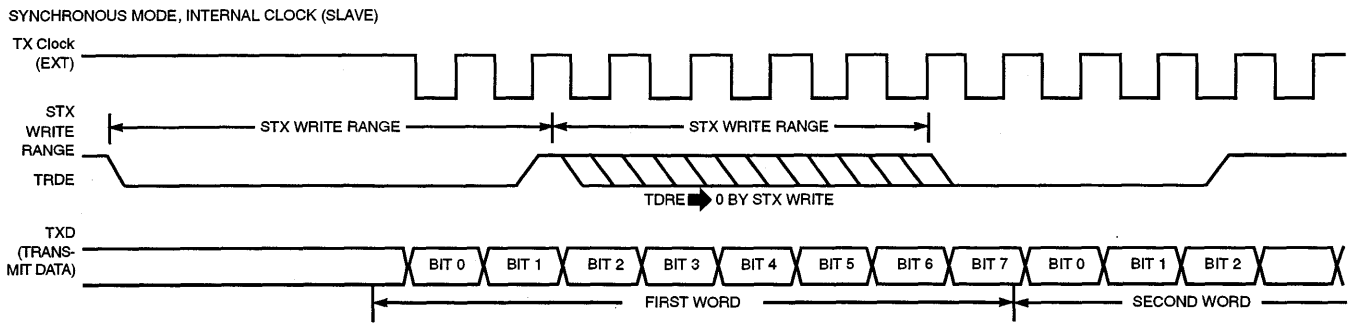
In synchronous mode TCM and RCM must equal one.

The assembly program shown in Figure 12-19 uses the SCI+ synchronous mode to transmit only the low byte of the Y data ROM contents. The program sets the reset vector to run the program after a hardware reset, puts the MOVEP instruction at the SCI+ transmit interrupt vector location, sets the memory wait states to zero, and configures the memory pointers, operating mode register, and the IPR.



EXAMPLE: INTERFACE TO SYNCHRONOUS MICROCOMPUTER BUSES

Figure 12-17. Synchronous Slave



NOTE: In external clock mode, if data 2 is written after the middle of bit 6 of data 1, then the previous data is retransmitted and data 2 is transmitted after the retransmission of data 1.

Figure 12-18. Synchronous Timing (Slave)

The SCI+ is then configured and the interrupts are unmasked, which starts the data transfer. The jump-to-self instruction (LAB0 JMP LAB0) is used to wait while interrupts transfer the data.

```
        ORG   P:0          ;Reset vector
        JMP   $40          ;
        ORG   P:$18        ;SCI+ transmit interrupt vector
      MOVEPY:(R0)+,X:$FFF4;Transmit low byte of data
      ORGP:$40
      MOVEP#0,X:$FFFE      ;Clear BCR
      MOVE#$100,R0         ;Data ROM start address
      MOVE#$FF,M0          ;Size of data ROM - Wraps around at $200
      MOVEC#6,OMR          ;Change operating mode to enable data ROM
      MOVEP#$C000,X:$FFFF;Interrupt priority register
      MOVEP#$1200,X:$FFF0;8-bit synchronous mode
      MOVEP#7,X:$FFE1      ;Port C control register - enable SCI+
      MOVEC#0,SR           ;Unmask interrupts
LAB0  JMPLAB0              ;Wait in loop for interrupts
```

Figure 12-19. SCI+ Synchronous Transmit

The program shown in Figure 12-20 is the program for receiving data from the program presented in Figure 12-19. The program sets the reset vector to run the program after hardware reset, puts the MOVEP instruction to store the data in a circular buffer starting at $100 at the SCI+ receive interrupt vector location, puts another MOVEP instruction at the SCI+ receive interrupt vector location, sets the memory wait states to zero, and configures the memory pointers and IPR. The SCI+ is then configured and the interrupts are unmasked, which starts the data transfer. The jump-to-self instruction (LAB0 JMP LAB0) is used to wait while interrupts transfer the data.


```
      ORGP:0 ;Reset vector
      JMP$40 ;
      ORGP:$14;SCI+ receive data vector
      MOVEPX:$FFF4,Y:(R0)+;Receive low byte of data
      NOP    ;Fast interrupt response
      MOVEPX:$FFF1,X0;Receive with exception. Read status register
      MOVEPX:$FFF4,Y:(R0)+;Receive low byte of data
      ORGP:$40
      MOVEP#0,X:$FFFE;Clear BCR
      MOVE#$100,R0;Data ROM start address
      MOVE#$FF,M0; Size of data ROM - wraps around at $200
      MOVEP#$C000,X:$FFFF;Interrupt priority register
      MOVEP#$900,X:$FFF0; 8-bit synchronous mode receive only
      MOVEP#$C000,X:$FFF2;Clock control register external clock
      MOVEP#7,X:$FFE1;Port C control register - enable SCI+
      MOVEC#0,SR;Unmask interrupts
LAB0  JMPLAB0;Wait in loop for interrupts
```

Figure 12-20. SCI+ Synchronous Receive

12.3.9 Asynchronous Data

Asynchronous data uses a data format with embedded word sync, which allows an unsynchronized data clock to be synchronized with the word if the clock rate and number of bits per word is known. Thus, the clock can be generated by the receiver rather than requiring a separate clock signal. The transmitter and receiver both use an internal clock that is 16 × the data rate to allow the SCI+ to synchronize the data. The data format requires that each data byte have an additional start bit and stop bit. In addition, two of the word formats have a parity bit. The multidrop mode used when SCIs are on a common bus has an additional data type bit. The SCI+ can operate in full-duplex or half-duplex modes since the transmitter and receiver are independent. The SCI+ transmitter and receiver can use either the internal clock (TCM=0 and/or RCM=0) or an external clock (TCM=1 and/or RCM=1) or a combination. If a combination is used, the transmitter and receiver can run at different data rates.

12.3.9.1 ASYNCHRONOUS DATA RECEPTION

Figure 12-21 illustrates initializing the SCI+ data receiver for asynchronous data. The first step (1) sets up the IMP DISC register for a direct connect to the NMSI2 pins on the IMP, step (2) resets the SCI+ to prevent the SCI+ from transmitting or receiving data. Step three(3) selects the desired operation by programming the SCR. As a minimum, the word format (WDS2, WDS1, and WDS0) must be selected, and (4) the receiver must be enabled (RE=1). If (5) interrupts are to be used, set RIE equals one. Use Table 12-3 through Table 12-6 to set (6) the baud rate (SCP and CD0–CD11 in the SCCR). Once the SCI+ is completely configured, it is enabled by (7) setting the RXD bit in the PCC.

The SCI+ can be connected to either SCC1 or the NMSI2 pins for use in asynchronous mode. See 7.5.3.2 SCI+ Serial Connections.

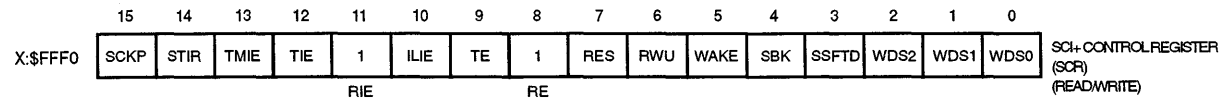
The receiver is continually sampling RDX at the 16 × clock rate to find the idle-start-bit transition edge. When that edge is detected (1) the following eight or nine bits, depending on the mode, are clocked into the receive shift register (see Figure 12-22). Once a complete byte is received, (2) the character is latched into the SRX, and RDRF is set as well as the error flags, OR, PE, and FE. If (3) interrupts are enabled, an interrupt is generated. The interrupt service routine, which can be a fast interrupt or a long interrupt, (4) reads the received character. Reading the SRX (5) automatically clears RDFR in the SSR and makes the SRX ready to receive another byte.

If (1) an FE, PE, or OR occurs while receiving data (see Figure 12-23), (2) RDRF is set because a character has been received; FE, PE, or OR is set in the SSR to indicate that an error was detected. Either (3) the SSR can be polled by software to look for errors, or (4) interrupts can be used to execute an interrupt service routine. This interrupt is different from the normal receive interrupt and is caused only by receive errors. The long interrupt service routine should (5) read the SSR to determine what error was detected and then (6) read the SRX to clear RDRF and all three error flags.

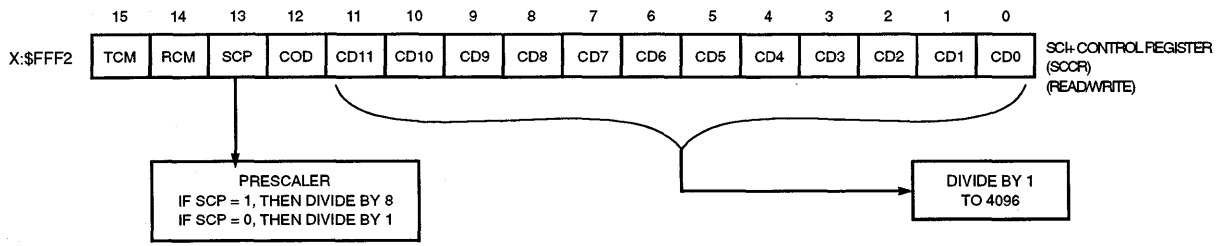
12.3.9.2 ASYNCHRONOUS DATA TRANSMISSION

Figure 12-24 illustrates initializing the SCI+ data transmitter for asynchronous data. The first step (1) sets up the IMP DISC register for desired connection option, (2)resets the SCI+ to prevent the SCI+ from transmitting or receiving data. Step three (3) selects the desired op-

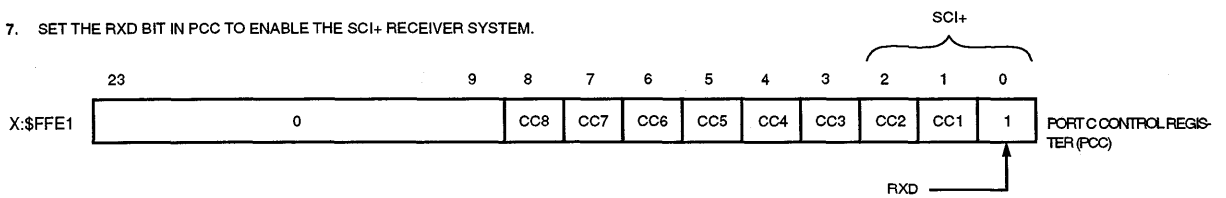
1. SET UP THE IMP DISC REGISTER FOR DESIRED CONNECTION OPTION.
2. HARDWARE OR SOFTWARE RESET
3. PROGRAM SCR WITH DESIRED MODE AND FEATURES.
4. TURN ON RECEIVER (RE = 1).
5. OPTIONALLY ENABLE RECEIVER INTERRUPTS (RIE = 1).



6. SET THE BAUD RATE BY PROGRAMMING THE SCCR.



7. SET THE RXD BIT IN PCC TO ENABLE THE SCI+ RECEIVER SYSTEM.



CCx	Function
0	GPIO
1	Serial Interface

NOTE: If RE is cleared while a valid character is being received, the reception of the character will be completed before the receiver is disabled.

Figure 12-21. Asynchronous SCI+ Receiver Initialization

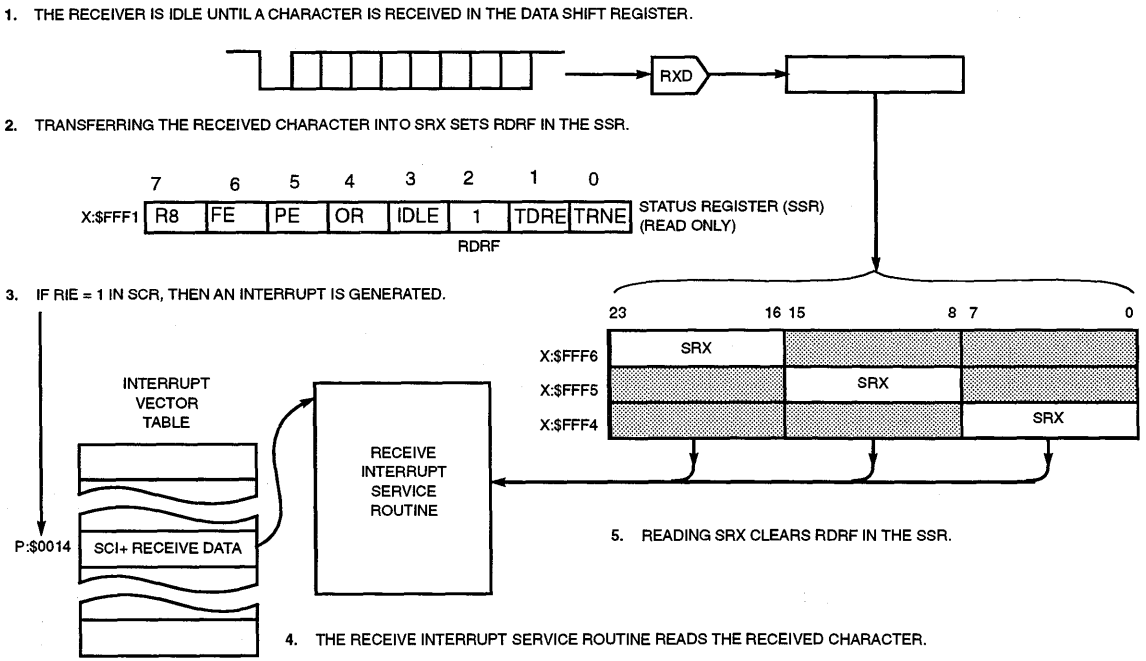


Figure 12-22. SCI+ Character Reception

1. A CHARACTER IS RECEIVED WITH AT LEAST ONE OF THE FOLLOWING ERRORS:
   — FRAMING ERROR (FE = BIT 6 IN SSR
   — PARITY ERROR (PE = BIT 5 IN SSR)
   — OVERRUN ERROR (OR = BIT 4 IN SSR)

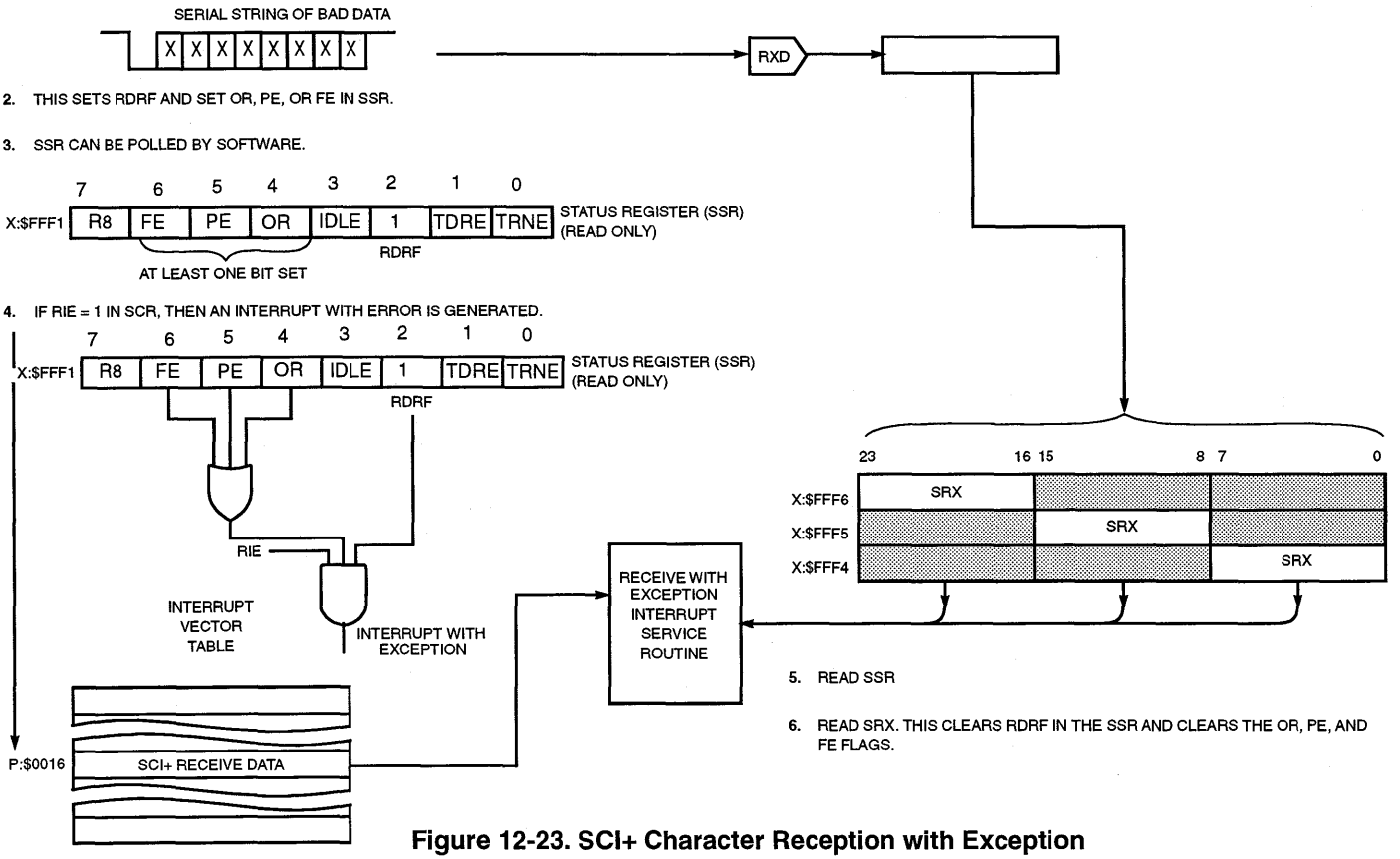
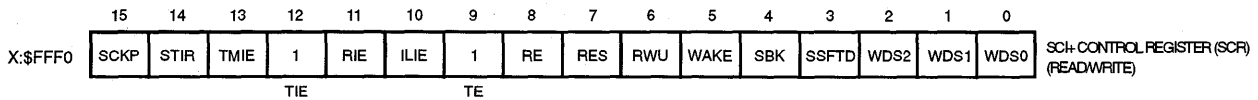
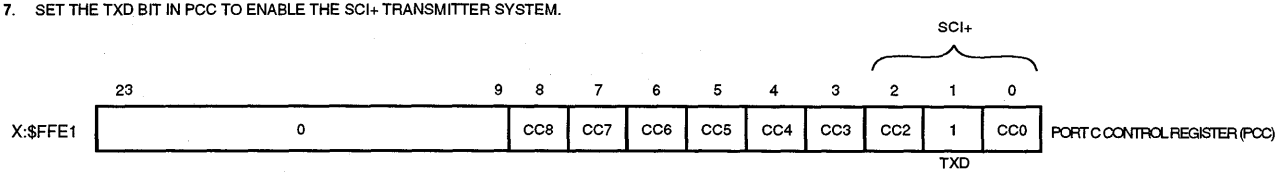


Figure 12-23. SCI+ Character Reception with Exception

1. SET UP THE IMP DISC REGISTER FOR DESIRED CONNECTION OPTION.
2. HARDWARE OR SOFTWARE RESET
3. PROGRAM SCR WITH DESIRED MODE AND FEATURES.
4. TURN ON RECEIVER (TE = 1).
5. OPTIONALLY ENABLE RECEIVER INTERRUPTS (TIE = 1).



6. SET THE SCI+ CLOCK PRESCALER BIT AND THE CLOCK DIVIDER BITS IN THE SCCR.
7. SET THE TXD BIT IN PCC TO ENABLE THE SCI+ TRANSMITTER SYSTEM.



CCx	Function
0	GPIO
1	Serial Interface

8. THE TRANSMITTER WILL FIRST BROADCAST A PREAMBLE OF ONES BEFORE BEGINNING DATA TRANSMISSION:
   10 ONES WILL BE TRANSMITTED FOR THE 10-BIT ASYNCHRONOUS MODE.
   11 ONES WILL BE TRANSMITTED FOR THE 11-BIT ASYNCHRONOUS MODE.

NOTE: If TE is cleared while transmitting a character, the transmission of the character will be completed before the transmitter is disabled.

Figure 12-24. Asynchronous SCI+ Transmitter Initialization

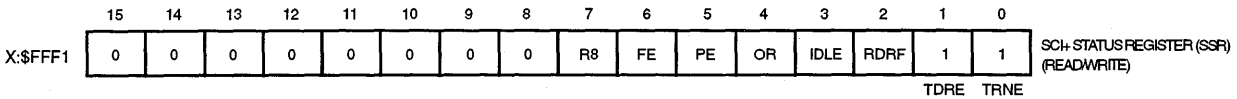
eration by programming the SCR. As a minimum, the word format (WDS2, WDS1, and WDS0) must be selected, and (4) the transmitter must be enabled (TE=1). If (5) interrupts are to be used, set TIE equals one. Use Table 12-3 through Table 12-6 to set (6) the baud rate (SCP and CD0–CD11 in the SCCR). Once the SCI+ is completely configured, it can be enabled by (7) setting the TXD bit in the PCC. Transmission begins with (8) a preamble of ones.

If polling is used to transmit data (see Figure 12-25), the polling routine can look at either TDRE or TRNE to determine when to load another byte into STX. If TDRE is used (1), one byte may be loaded into STX. If TRNE is used (2), two bytes may be loaded into STX if enough time is allowed for the first byte to begin transmission (see 12.3.4.4.2SCI+ Transmit Registers). If interrupts are used (3), then an interrupt is generated when STX is empty. The interrupt routine, which can be a fast interrupt or a long interrupt, writes (4) one byte into STX. If multidrop mode is being used and this byte is an address, STXA should be used instead of STX. Writing STX or STXA (5) clears TDRE in the SSR. When the transmit data shift register is empty (6), the byte in STX (or STXA) is latched into the transmit data shift register, TRNE is cleared, and TDRE is set.

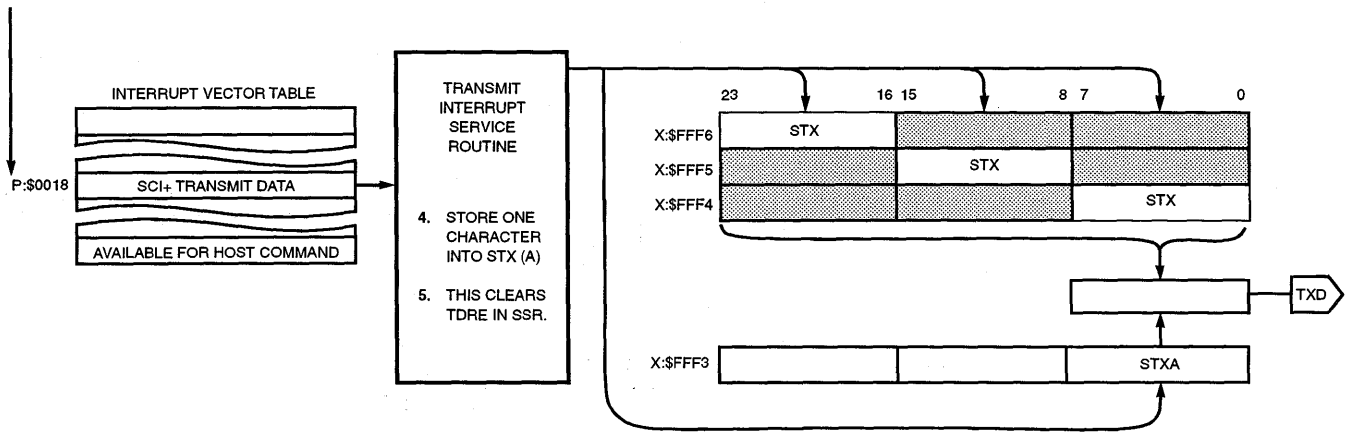
There is a provision to send a break or preamble. A break (space) consists of a period of zeros with no start or stop bits that is as long or longer than a character frame. A preamble (mark) is an inverted break. A preamble of 10 or 11 ones (depending on the word length selected by WDS2, WDS1, and WDS0) can be sent with the following procedure (see Figure 12-26). (1) Write the last byte to STX and (2) wait for TDRE equals one. This is the byte that will be transmitted immediately before the preamble. (3) Clear TE and then again set it to one. Momentarily clearing TE causes the output to go high for one character frame. If TE remains cleared for a longer period, the output will remain high for an even number of character frames until TE is set. (4) Write the first byte to follow the preamble into SRX before the preamble is complete and resume normal transmission. Sending a break follows the same procedure except that instead of clearing TE, SBK is set in the SCR to send breaks and then reset to resume normal data transmission.

The example presented in Figure 12-29 uses the SCI+ in the asynchronous mode to transfer data into buffers. Interrupts are used, allowing the DSP to perform other tasks while the data transfer is occurring. This program can be tested by connecting the SCI+ transmit and receive pins. Equates are used for convenience and readability.

The program sets the reset vector to run the program after reset, puts a MOVEP instruction at the SCI+ receive interrupt vector location, and puts a MOVEP and BCLR at the SCI+ transmit interrupt vector location so that, after transmitting a byte, the transmitter is disabled until another byte is ready for transmission. The SCI+ is initialized by setting the interrupt level, which configures the SCR and SCCR, and then is enabled by writing the PCC. The main program begins by enabling interrupts, which allows data to be received. Data is transmitted by moving a byte of data to the transmit register and by enabling interrupts. The jump-to-self instruction (SEND JMP SEND) is used to wait while interrupts transfer the data.

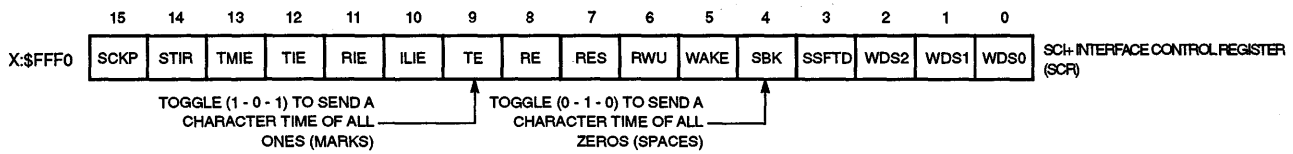


1. WHEN STX IS EMPTY, THEN TDRE = 1.
2. WHEN STX IS EMPTY AND THE TRANSMIT DATA SHIFT REGISTER IS EMPTY THEN TRNE = 1.
3. IF TIE = 1 IN SCR AND TDRE = 1 IN SSR, THEN AN INTERRUPT IS GENERATED.



6. THE CHARACTER IN STX IS COPIED INTO TRANSMIT DATA SHIFT REGISTER.
   TRNE IS CLEARED.
   TDRE IS SET.
   GO TO STEP 2.

Figure 12-25. Asynchronous SCI+ Character Transmission



- 10 OR 11 ONES/ZEROS WILL BE SENT DEPENDING ON THE WORD LENGTH SPECIFIED BY WDS2, WDS1, WDS0.

MARKS (ONES)

1. WRITE THE LAST BYTE TO STX.
2. WAIT FOR TRDE = 1. THE LAST BYTE IS NOW IN THE TRANSMIT SHIFT REGISTER.
3. CLEAR TE AND SET BACK TO ONE. THIS QUEUES THE PREAMBLE TO FOLLOW THE LAST BYTE.
4. WRITE THE FIRST BYTE TO FOLLOW THE PREAMBLE INTO SRX.

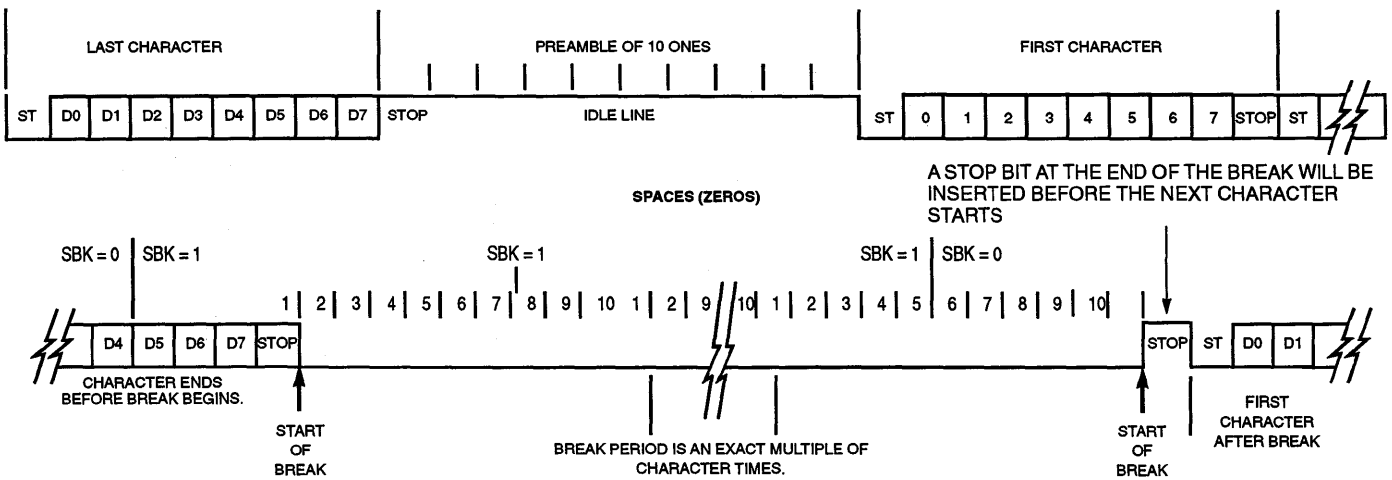


Figure 12-26. Transmitting Marks and Spaces

```
;*****************************************************************************
        *****************
;       SCI+ ASYNC WITH INTERRUPTS AND SINGLE BYTE BUFFERS*
;*****************************************************************************
        *****************
;***********************************************
;       SCI+ and other EQUATES*
;***********************************************
START   EQU$0040;Start of program
PCC     EQU$FFE1;Port C control register
SCR     EQU$FFF0;SCI+ interface control register
SCCR    EQU$FFF2;SCI+ clock control register
SRX     EQU$FFF4;SCI+ receive register
STX     EQU$FFF4;SCI+ transmit register
BCR     EQU$FFFE;Bus control register
IPR     EQU$FFFF;Interrupt priority register
RXBUF   EQU$100;Receive buffer
TXBUF   EQU$200;Transmit buffer
```

Figure 12-27. SCI+ Asynchronous Transmit/Receive Example (Sheet 1 of 3)

```
;***********************************************
;       RESET VECTOR*
;***********************************************
        ORGP:$0000
        JMPSTART
;***********************************************
;       SCI+ RECEIVE INTERRUPT VECTOR*
;***********************************************
        ORGP:$0014;Load the SCI+ RX interrupt vectors
        MOVEPX:SRX,Y:(R0)+;Put the received byte in the receive
                ;buffer. This receive routine is
                ;implemented as a fast interrupt.
;***********************************************
;       SCI+ TRANSMIT INTERRUPT VECTOR*
;***********************************************
        ORGP:$0018;Load the SCI+ TX interrupt vectors
        MOVEPX:(R3)+,X:STX;Transmit a byte and
                ;increment the pointer in the
                ;transmit buffer.
        BCLR#12,X:SCR;Disable transmit interrupts
;*************************************************************************
        ****************
;       INITIALIZE THE SCI+ PORT AND RX, TX BUFFER POINTERS*
;*************************************************************************
        ****************
        ORGP:START;Start the program at location $40
        ORI#$03,MR;Mask interrupts temporarily
        MOVEP#$C000,X:IPR;Set interrupt priority to 2
        MOVEP#$0B02,X:SCR;Disable TX, enable RX interrupts
                ;Enable transmitter, receiver
                ;Point to point
                ;10-bit asynchronous
                ;(1 start, 8 data, 1 stop)
        MOVEP#$0022,X:SCCR;Use internal TX, RX clocks
                ;9600 BPS
        MOVEP#>$03,X:PCC;Select pins TXD and RXD for SCI+
        MOVERXBUF,R0;Initialize the receive buffer
        MOVETXBUF,R3;Initialize the transmit buffer
```

Figure 12-28. SCI+ Asynchronous Transmit/Receive Example (Sheet 2 of 3)

```
;***********************************************
;       MAIN PROGRAM*
;***********************************************
        ANDI#$FC,MR;Re-enable interrupts
        MOVE#>$41,X:(R3);Move a byte to the transmit buffer
        MOVER0,X:(R3)
        BSET#12,X:SCR;and enable interrupts so it
                ;will be transmitted
SEND    JMPSEND;Normally something more useful
                ;would be put here.
        END     ;End of example.
```

Figure 12-29. SCI+ Asynchronous Transmit/Receive Example (Sheet 3 of 3)

12.3.10 SCI+ Timer

The SCI+ clock determines the data transmission rate and can also be used to establish a periodic interrupt that can act as an event timer or be used in any other timing function. Figure 12-30 illustrates how the SCI+ timer is programmed. Bits CD11–CD0, SCP, and STIR in the SCCR work together to determine the time base. The crystal oscillator f_{osc} is first divided by 2 and then divided by the number CD11–CD0 in the SCCR. The oscillator is then divided by 1 (if SCP=0) or eight (if SCP=1). This output is used as is if STIR = 1 or, if STIR = 0, it is divided by 2 and then by 16 before being used. If TMIE in the SCR = 1 when the periodic timeout occurs, the SCI+ timer interrupt is recognized and pending. The SCI+ timer interrupt is automatically cleared when the interrupt is serviced. This interrupt will occur every time the periodic timer times out. If only the timer function is being used (i.e., PC0, PC1, and PC2 pins have been programmed as GPIO pins), the transmit interrupts should be turned off (TIE=0). Under individual reset, TDRE will remain set and the timer will continuously generate interrupts.

Figure 12-30 shows that an external clock can be used for SCI+ receive and/or transmit, which frees the SCI+ timer to be programmed for a different interrupt rate. In addition, both the SCI+ timer interrupt and the SCI+ can use the internal time base if the SCI+ receiver and/or transmitter require the same clock period as the SCI+ timer.

The program in Figure 12-32 configures the SCI+ to interrupt the DSP at fixed intervals. The program starts by setting equates for convenience and clarity and then points the reset vector to the start of the program. The SCI+ timer interrupt vector location contains "move (R0)+", incrementing the contents of R0, which serves as an elapsed time counter.

The timer initialization consists of enabling the SCI+ timer interrupt, setting the SCI+ baud rate counters for the desired interrupt rate, setting the interrupt mask, enabling the interrupt, and then enabling the SCI+ state machine.

12.3.11 Bootstrap Loading Through the SCI+ (Operating Mode 6)

When the DSP comes out of reset, it looks at the MODC, MODB, and MODA pins or signals in the IMP DISC register (whichever is selected) and sets the corresponding mode bits in the OMR. Since the internal serial connection to the SCI+ signals must be initialized before bring the DSP out of reset, the following procedure should be followed:

1. The DSP should be held in reset by the IMP through the control bits provided in the DISC register (SELR=1, RST=0) while the IMP initializes itself and configures the desired serial connection by programming the IC0-IC3 bits in the DISC register.

NOTE

Although not required, the DRESET pin can be grounded to hold the DSP in reset until the IMP is ready to release it from reset. This eliminates the need for separate reset circuitry for the $\overline{\text{DRESET}}$ pin.

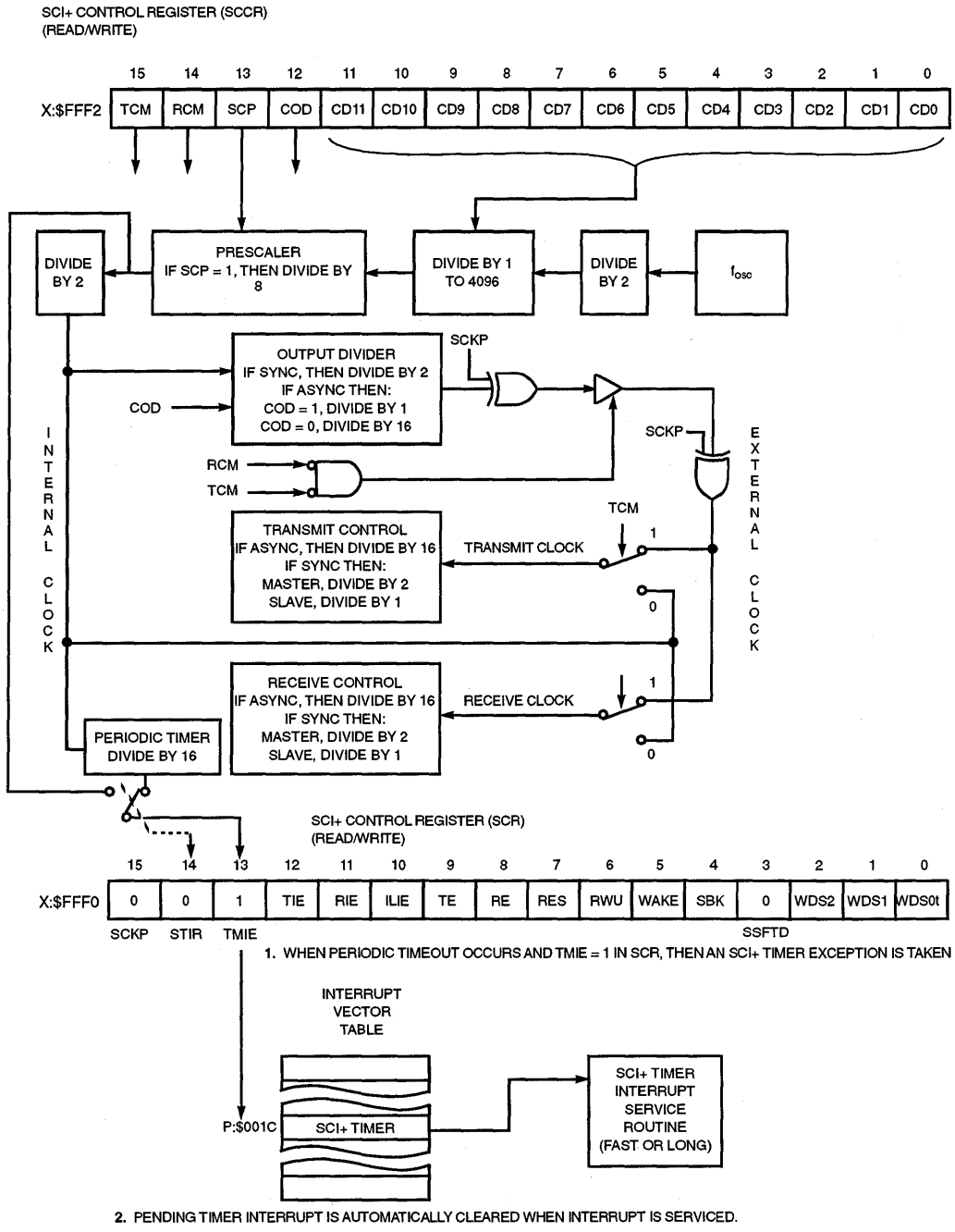


Figure 12-30. SCI+ Timer Operation

```
;*****************************************************************************
        *****************
;       TIMER USING SCI+ TIMER INTERRUPT*
;*****************************************************************************
        *****************
;***********************************************
;       SCI+ and other EQUATES*
;***********************************************
START   EQU$0040;Start of program
SCR     EQU$FFF0;SCI+ control register
SCCR    EQU$FFF2;SCI+ clock control register
IPR     EQU$FFFF;Interrupt priority register
;***********************************************
;       RESET VECTOR*
;***********************************************
        ORGP:$0000
        JMPSTART
;***********************************************
;       SCI+ TIMER INTERRUPT VECTOR*
;***********************************************
        ORGP:$001C;Load the SCI+ timer interrupt vectors
        MOVE(R0)+;Increment the timer interrupt counter
        NOP     ;This timer routine is implemented
                ;as a fast interrupt
;***********************************************
;       INITIALIZE THE SCI+ PORT*
;***********************************************
        ORGP:START;Start the program at location $40
        MOVE#0,R0;Initialize the timer interrupt counter
        MOVEP#$2000,X:SCR;Select the timer interrupt
        MOVEP#$013F,X:SCCR;Set the interrupt rate at 1 ms
                ;(arbitrarily chosen)
                ;Interrupts/second =
                ;fosc/(64X(7(SCP)-+1)X(CD+1))
                ;Note that this is the same equation
                ;as for SCI+ async baud rate
```

Figure 12-31. SCI+ Timer Example (Sheet 1 of 2)

```
                ;For 1 ms, SCP=0,
                ;CD=0001 0011 1111.
        MOVEP#$C000,X:IPR;Set the interrupt priority level-
                ;application specific.
        ANDI#$FC,MR;Enable interrupts, set MR bits I1 and
                ;I0=0
END     JMPEND ;Normally something more useful
                ;would be put here.
        END     ;End of example.
```

Figure 12-32. SCI+ Timer Example (Sheet 2 of 2)

2. After the IMP has completed its reset initialization routine it should, in a single access to the DISC register, 1)reset the RST bit to zero, 2)set the MODA,B,C bits to the desired DSP bootstrap mode (mode 6), and 3)set all the SEL bits to one. This access switches control of the DSP reset line to the IMP and holds DSP reset still asserted. The MOD signals are also applied to the DSP.

3. To release the DSP from reset, the IMP should, in a single access 1) set the RST bit to one, 2) program the MODA,B,C to the corresponding value desired for the IRQA,B and NMI signals (usually ones for the MOD bits to avoid an immediate DSP interrupt), and 3) set the SELA,B,C bits to the desired signal source (the corresponding SEL bits should be reset to zero if off chip interrupt sources are required).

4. The serial bootstrap load source, whether IMP SCC1 or external on the NMSI2 pins, should begin the loading process as describe below.

Figure 12-33 shows how the SCI+ is configured for receiving this code and Figure 12-33 shows the segment of bootstrap code that is used to load from the SCI+. The complete code used in the bootstrap program is given in Appendix C . This program (1) configures the SCI+, (2) loads the program size, (3) loads the location where the program will begin loading in program memory, and (4) loads the program.

First, the SCI+ control register is set to $0302 (see Figure 12-2) which enables the transmitter and receiver and configures the SCI+ for 10 bits **asynchronous** with **one start bit, 8 data bits, one stop bit, and no parity.** Next, the SCI+ clock control register is set to $C000 which configures the SCI+ to use external receive and transmit clocks on the TXCLK and RXCLK signals. This means that both the TCLK2 and RCLK2 IMP pins must be driven with the same clock for external serial loaders, or SCC1 must provide the same clock to both SCI+ TCLK and RCLK signals if the SCC1 is the serial loader. This **clock** must be **16 times the serial data rate.**

The next step is to receive the program size and then the starting address to load the program. These two numbers are three bytes each loaded least significant byte first. Each byte will be echoed back as it is received. After both numbers are loaded, the program size is in A0 and the starting address is in A1.

The program is then loaded one byte at a time, least significant byte first. After loading the program, the operating mode is set to zero, the CCR is cleared, and the DSP begins execution with the first instruction that was loaded.

12.4 SYNCHRONOUS SERIAL INTERFACE (SSI)

The synchronous serial interface (SSI) provides a full-duplex serial port for serial communication with a variety of serial devices including one or more industry-standard codecs, other DSPs, microprocessors, and peripherals which implement the Motorola SPI.

The user can independently define the following characteristics of the SSI: the number of bits per word, the protocol, the clock, and the transmit/receive synchronization.

The user can select among three modes: normal, on-demand, and network. The normal mode is typically used to interface with devices on a regular or periodic basis. The data-driven on-demand mode is intended to be used to communicate with devices on a nonperiodic basis. The network mode provides time slots in addition to a bit clock and frame synchronization pulse.

```
; This routine loads from the SCI+.
; MC:MB:MA=110 - external SCI+ clock
; MC:MB:MA=111 - reserved
SCILD   MOVEP   #$0302,X:SCR      ; Configure SCI+ Control Reg
        JMP     <EXTC             ; go to next bootrom segment
        NOP                       ; just to fill the last space

        ORG     PL:$100,PL:$100   ; starting address of 2nd ROM

EXTC    MOVEP   #$C000,X:SCCR     ; Configure SCI+ Clock Control Reg
        MOVEP   #7,X:PCC          ; Configure SCLK, TXD and RXD

_SCI1   DO      #6,_LOOP6         ; get 3 bytes for number of
                                  ; program words and 3 bytes
                                  ; for the starting address
        JCLR    #2,X:SSR,*        ; Wait for RDRF to go high
        MOVEP   X:SRXL,A2         ; Put 8 bits in A2
        JCLR    #1,X:SSR,*        ; Wait for TDRE to go high
        MOVEP   A2,X:STXL         ; echo the received byte
        REP     #8
        ASR     A
_LOOP6
        MOVE    A1,R0             ; starting address for load
        MOVE    A1,R1             ; save starting address
        DO      A0,_LOOP4         ; Receive program words

        DO      #3,_LOOP5
        JCLR    #2,X:SSR,*        ; Wait for RDRF to go high
        MOVEP   X:SRXL,A2         ; Put 8 bits in A2
        JCLR    #1,X:SSR,*        ; Wait for TDRE to go high
        MOVEP   A2,X:STXL         ; echo the received byte
        REP     #8
        ASR     A
_LOOP5
        MOVEM   A1,P:(R0)+        ; Store 24-bit result in P memory
_LOOP4
```

Figure 12-33. Bootstrap Code Fragment

The SSI functions with a range of 2 to 32 words of I/O per frame in the network mode. This mode is typically used in star or ring time division multiplex networks with other DSP56K processors and/or codecs. The clock can be programmed to be continuous or gated. Since the transmitter and receiver sections of the SSI are independent, they can be programmed to be synchronous (using a common clock) or asynchronous with respect to each other.

The SSI requires up to six pins, depending on its operating mode. The most common minimum configuration is three pins: transmit data (STD), receive data (SRD) and clock (SCK).

The SSI consists of independent transmitter and receiver sections and a common SSI clock generator. Three to six pins are required for operation, depending on the operating mode selected.

The following is a short list of SSI features:

• Three-Pin Interface:

TXD – Transmit Data
RXD – Receive Data
SCK – Serial Clock

- A 10 Mbps at 40 MHz ($f_{osc}/4$) serial interface
- Double Buffered
- User Programmable
- Separate Transmit and Receive Sections
- Control and Status Bits
- Interface to a Variety of Serial Devices, Including:
  Codecs (usually without additional logic)
    MC145502
    MC145503
    MC145505
    MC145402 (13-bit linear codec)
    MC145554 Family of Codecs
    MC145532

  Serial Peripherals (A/D, D/A)
    Most Industry-Standard A/D, D/A
    DSP56ADC16 (16-bit linear A/D)

  DSP56K to DSP56K Networks
  Motorola SPI Peripherals and Processors
  Shift Registers

- Interface to Time Division Multiplexed Networks without Additional Logic
- Six Pins:
  STD SSI Transmit Data
  SRD SSI Receive Data
  SCK SSI Serial Clock
  SC0 Serial Control 0 (defined by SSI mode)
  SC1 Serial Control 1 (defined by SSI mode)
  SC2 Serial Control 2 (defined by SSI mode)

- On-chip Programmable Functions Include:
  Clock – Continuous, Gated, Internal, External
  Synchronization Signals – Bit Length and Word Length
  TX/RX Timing – Synchronous, Asynchronous
  Operating Modes – Normal, Network, On-Demand
  Word Length – 8, 12, 16, 24 Bits
  Serial Clock and Frame Sync Generator

- Four Interrupt Vectors:
  Receive
  Receive with Exception
  Transmit
  Transmit with Exception

This interface is descriptively named "synchronous" because all serial transfers are synchronized to a clock. Additional synchronization signals are used to delineate the word frames. The normal mode of operation is used to transfer data at a periodic rate, but only one word per period. The network mode is similar in that it is also intended for periodic transfers; however, it will support up to 32 words (time slots) per period. This mode can be used to build time division multiplexed (TDM) networks. In contrast, the on-demand mode is intended for nonperiodic transfers of data. This mode can be used to transfer data serially at high speed when the data becomes available. This mode offers a subset of the SPI protocol.

12.4.1 SSI Data and Control Pins

The SSI has three dedicated I/O pins which are used for transmit data (STD), receive data (SRD), and serial clock (SCK), where SCK may be used by both the transmitter and the receiver for synchronous data transfers or by the transmitter only for asynchronous data transfers. Three other pins may also be used, depending on the mode selected; they are serial control pins SC0, SC1, and SC2. They may be programmed as SSI control pins in the port C control register. Table 12-7 shows the definition of SC0, SC1, SC2, and SCK in the various configurations. The following paragraphs describe the uses of these pins for each of the SSI operating modes. Figure 12-34 and Figure 12-35 show the internal clock path connections in block diagram form. The receiver and transmitter clocks can be internal or external depending on the SYN, SCD0, and SCKD bits in CRB.

Table 12-7. Definition of SC0, SC1, SC2, and SCK

SSI Pin Name (Control Bit Name)	Asynchronous (SYN=0)		Synchronous (SYN=1)	
	Continuous Clock (GCK=0)	Gated Clock (GCK=1)	Continuous Clock (GCK=0)	Gated Clock (GCK=1)
SC0=0 (in)	RXC External	RXC External	Input F0	Input F0
SC0=1 (out) (SCD0)	RXC Internal	RXC Internal	Output F0	Output F0
SC1=0 (in)	FSR External	Not Used	Input F1	Input F1
SC1=1 (out) (SCD1)	FSR Internal	FSR Internal	Output F1	Output F1
SC2=0 (in)	FST External	Not Used	FS* External	Not Used
SC2=1 (out) (SCD2)	FST Internal	FST Internal	FS* Internal	FS* Internal
SCK=0 (in)	TXC External	TXC External	*XC External	*XC External
SCK=1 (out (SCKD)	TXC Internal	TXC Internal)	*XC Internal	*XC Internal

TXC – Transmitter Clock
RXC – Receiver Clock
*XC – Transmitter/Receiver Clock (synchronous operation)
FST – Transmitter Frame Sync
FSR – Receiver Frame Sync
FS* – Transmitter/Receiver Frame Sync (synchronous operation)
F0 – Flag 0
F1 – Flag 1

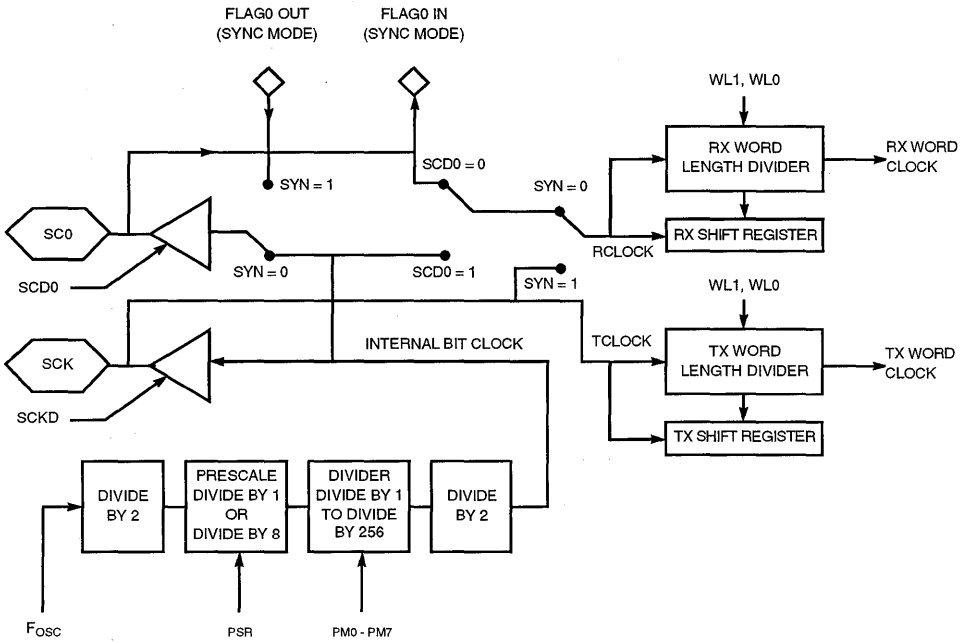


Figure 12-34. SSI Clock Generator Functional Block Diagram

Table 12-8. SSI Clock Sources, Inputs, and Outputs

SYN	SCKD	SCD0	R Clock Source	RX Clock Out	T Clock Source	TX Clock Out
Asynchronous						
0	0	0	EXT, SC0	–	EXT, SCK	–
0	0	1	INT	SC0	EXT, SCK	–
0	1	0	EXT, SC0	–	INT	SCK
0	1	1	INT	SC0	INT	SCK
Synchronous						
1	0	0	EXT, SCK	–	EXT, SCK	–
1	0	1	EXT, SCK	–	EXT, SCK	–
1	1	0	INT	SCK	INT	SCK
1	1	1	INT	SCK	INT	SCK

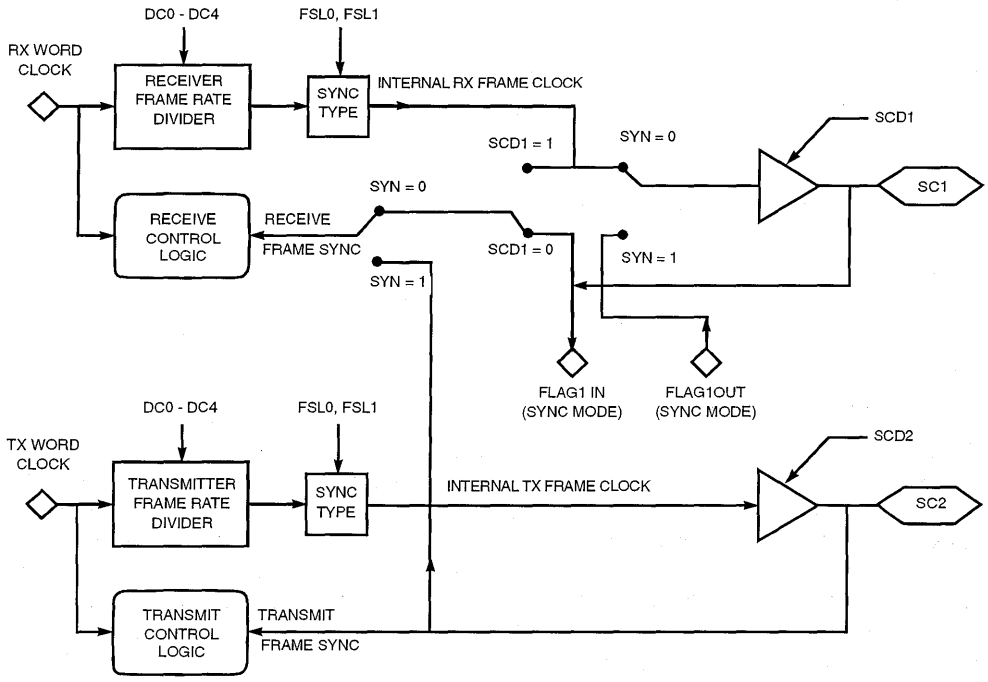


Figure 12-35. SSI Frame Sync Generator Functional Block Diagram

12.4.1.1 SERIAL TRANSMIT DATA PIN (STD)

STD is used for transmitting data from the serial transmit shift register. STD is an output when data is being transmitted. Data changes on the positive edge of the bit clock. STD goes to high impedance on the negative edge of the bit clock of the last data bit of the word (i.e., during the second half of the last data bit period) with external gated clock, regardless of the mode. With an internally generated bit clock, the STD pin becomes high impedance after the last data bit has been transmitted for a full clock period, assuming another data word does not follow immediately. If a data word follows immediately, there will not be a high-impedance interval.

Codecs label the MSB as bit 0; whereas, the DSP labels the LSB as bit 0. Therefore, when using a standard codec, the DSP MSB (or codec bit 0) is shifted out first when SHFD=0, and the DSP LSB (or codec bit 7) is shifted out first when SHFD=1. STD may be programmed as a general-purpose pin called PC8 when the SSI STD function is not being used.

12.4.1.2 SERIAL RECEIVE DATA PIN (SRD)

SRD receives serial data and transfers the data to the SSI receive shift register. SRD may be programmed as a general-purpose I/O pin called PC7 when the SSI SRD function is not being used. Data is sampled on the negative edge of the bit clock.

12.4.1.3 SERIAL CLOCK (SCK)

SCK is a bidirectional pin providing the serial bit rate clock for the SSI interface. The SCK is a clock input or output used by both the transmitter and receiver in synchronous modes or by the transmitter in asynchronous modes (see Table 12-3).

NOTE

Although an external serial clock can be independent of and asynchronous to the DSP system clock, it must exceed the minimum clock cycle time of 8T (i.e., the system clock frequency must be at least four times the external SSI clock frequency). The SSI needs at least four DSP phases (DSP phase=T) inside each half of the serial clock.

12.4.1.4 SERIAL CONTROL PIN (SC0)

The function of this pin is determined solely on the selection of either synchronous or asynchronous mode (see Table 12-7 and Table 12-8). For asynchronous mode, this pin will be used for the receive clock I/O. For synchronous mode, this pin is used for serial flag I/O. A typical application of flag I/O would be multiple device selection for addressing in codec systems. The direction of this pin is determined by the SCD0 bit in the CRB as described in Table 12-9. When configured as an output, this pin will be either serial output flag 0, based on control bit OF0 in CRB, or a receive shift register clock output. When configured as an input, this pin may be used either as serial input flag 0, which will control status bit IF0 in the SSISR, or as a receive shift register clock input.

Table 12-9. SSI Operation: Flag 0 and Rx Clock

SYN	GCK	SCD0	Operation
Synchronous	Continuous	Input	Flag 0 Input
Synchronous	Continuous	Output	Flag 0 Output
Synchronous	Gated	Input	Flag 0 Input
Synchronous	Gated	Output	Flag 0 Output
Asynchronous	Continuous	Input	Rx Clock – External
Asynchronous	Continuous	Output	Rx Clock – Internal
Asynchronous	Gated	Input	Rx Clock – External
Asynchronous	Gated	Output	Rx Clock – Internal

12.4.1.5 SERIAL CONTROL PIN (SC1)

The function of this pin is determined solely on the selection of either synchronous or asynchronous mode (see Table 12-7). In asynchronous mode (such as a single codec with asynchronous transmit and receive), this pin is the receiver frame sync I/O. For synchronous mode with continuous clock, this pin is serial flag SC1 and operates like the previously described SC0. SC0 and SC1 are independent serial I/O flags but may be used together for

multiple serial device selection. SC0 and SC1 can be used unencoded to select up to two codecs or may be decoded externally to select up to four codecs. The direction of this pin is determined by the SCD1 bit in the CRB. When configured as an output, this pin will be either a serial output flag, based on control bit OF1, or it will make the receive frame sync signal available. When configured as an input, this pin may be used as a serial input flag, which will control status bit IF1 in the SSI status register, or as a receive frame sync from an external source for continuous clock mode. In the gated clock mode, external frame sync signals are not used. This pin is used for frame sync I/O (see Table 12-7). SC2 is the frame

Table 12-10. Serial Control Pin (SC2)

SYN	GCK	SCD1	Operation
Synchronous	Continuous	Input	Flag 1 Input
Synchronous	Continuous	Output	Flag 1 Output
Synchronous	Gated	Input	Flag 1 Input
Synchronous	Gated	Output	Flag 1 Output
Asynchronous	Continuous	Input	RX Frame Sync – External
Asynchronous	Continuous	Output	RX Frame Sync – Internal
Asynchronous	Gated	Input	–
Asynchronous	Gated	Output	RX Frame Sync – Internal

sync for both the transmitter and receiver in synchronous mode and for the transmitter only in asynchronous mode. The direction of this pin is determined by the SCD2 bit in CRB. When configured as an output, this pin is the internally generated frame sync signal. When configured as an input, this pin receives an external frame sync signal for the transmitter (and the receiver in synchronous operation). In the gated clock mode, external frame sync signals are not used.

Table 12-11. SSI Operation: Tx and Rx Frame Sync

SYN	GCK	SCD2	Operation
Synchronous	Continuous	Input	TX and RX Frame Sync
Synchronous	Continuous	Output	TX and RX Frame Sync
Synchronous	Gated	Input	–
Synchronous	Gated	Output	TX and RX Frame Sync
Asynchronous	Continuous	Input	TX Frame Sync – External
Asynchronous	Continuous	Output	TX Frame Sync – Internal
Asynchronous	Gated	Input	–
Asynchronous	Gated	Output	TX Frame Sync – Internal

12.4.2 SSI Programming Model

The SSI can be viewed as two control registers, one status register, a transmit register, a receive register, and special-purpose time slot register. These registers are illustrated in Figure 12-36 and Figure 12-37. The following paragraphs give detailed descriptions and operations of each of the bits in the SSI registers. The SSI registers are not prefaced with an "S" (for serial) as are the SCI+ registers.

12.4.2.1 SSI CONTROL REGISTER A (CRA)

CRA is one of two 16-bit read/write control registers used to direct the operation of the SSI. The CRA controls the SSI clock generator bit and frame sync rates, word length, and number of words per frame for the serial data. The high-order bits of CRA are read as zeros by the DSP CPU. The CRA control bits are described in the following paragraphs.

12.4.2.1.1 CRA Prescale Modulus Select (PM7–PM0) Bits 0–7

The PM0–PM7 bits specify the divide ratio of the prescale divider in the SSI clock generator. A divide ratio from 1 to 256 (PM=0 to $FF) may be selected. The bit clock output is available at the transmit clock (SCK) and/or the receive clock (SC0) pins of the DSP. The bit clock output is also available internally for use as the bit clock to shift the transmit and receive shift registers. Careful choice of the crystal oscillator frequency and the prescaler modulus will allow the industry-standard codec master clock frequencies of 2.048 MHz, 1.544 MHz, and 1.536 MHz to be generated. Hardware and software reset clear PM0–PM7.

12.4.2.1.2 CRA Frame Rate Divider Control (DC4–DC0) Bits 8–12

The DC4–DC0 bits control the divide ratio for the programmable frame rate dividers used to generate the frame clocks (see Figure 12-35). In network mode, this ratio may be interpreted as the number of words per frame minus one. In normal mode, this ratio determines the word transfer rate. The divide ratio may range from 1 to 32 (DC=00000 to 11111) for normal mode and 2 to 32 (DC=00001 to 11111) for network mode.

A divide ratio of one (DC=00000) in network mode is a special case (see 12.4.7.4 On-Demand Mode Examples). In normal mode, a divide ratio of one (DC=00000) provides continuous periodic data word transfers. A bit-length sync (FSL1=1, FSL0=0) must be used in this case. Hardware and software reset clear DC4–DC0.

12.4.2.1.3 CRA Word Length Control (WL0, WL1) Bits 13 and 14

The WL1 and WL0 bits are used to select the length of the data words being transferred via the SSI. Word lengths of 8, 12, 16, or 24 bits may be selected according to the assignments

Table 12-12. CRA Word Length Control

WL1	WL0	Number of Bits/Word
0	0	8
0	1	12
1	0	16
1	1	24

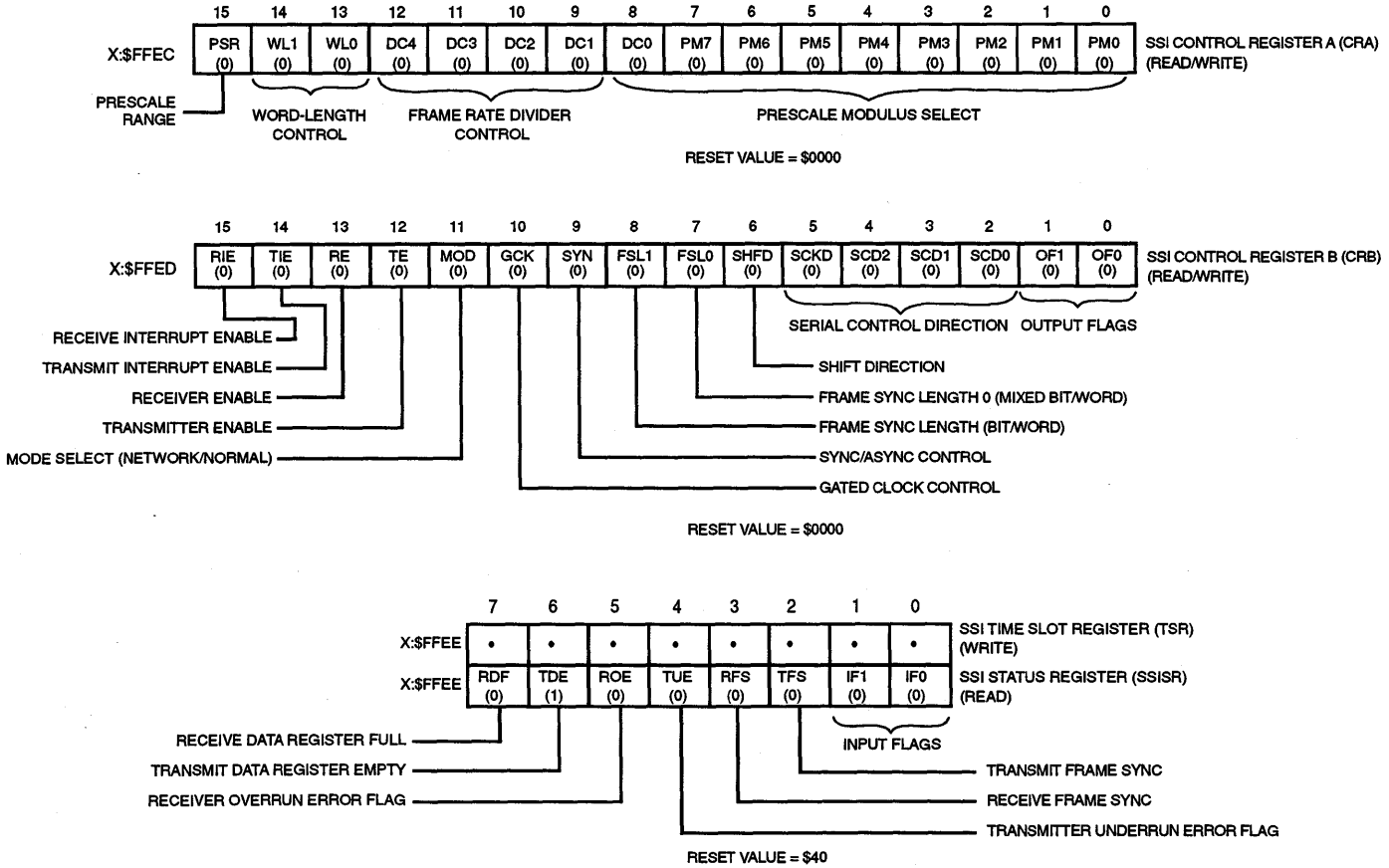
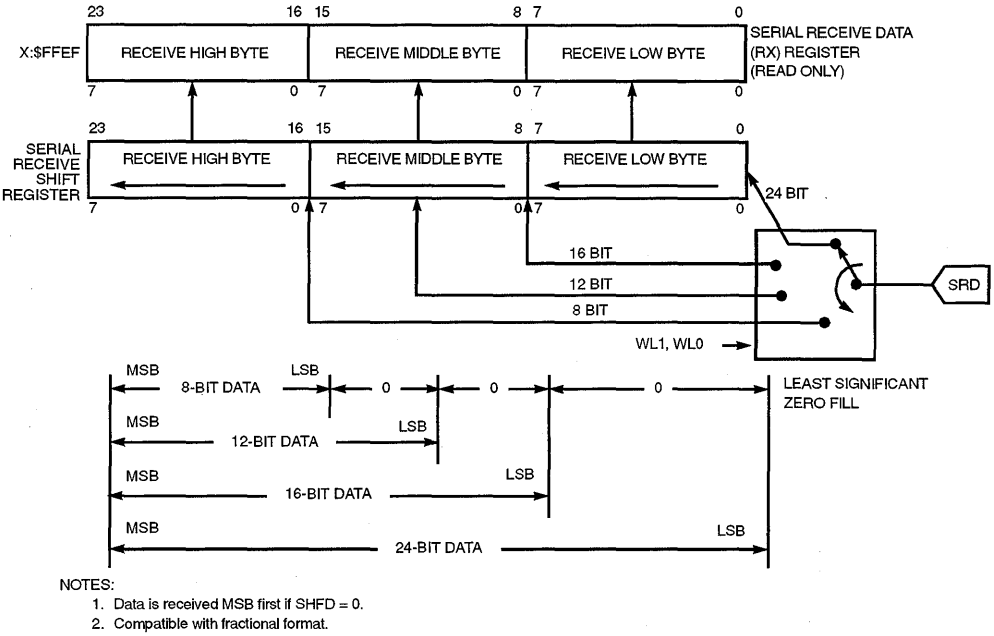
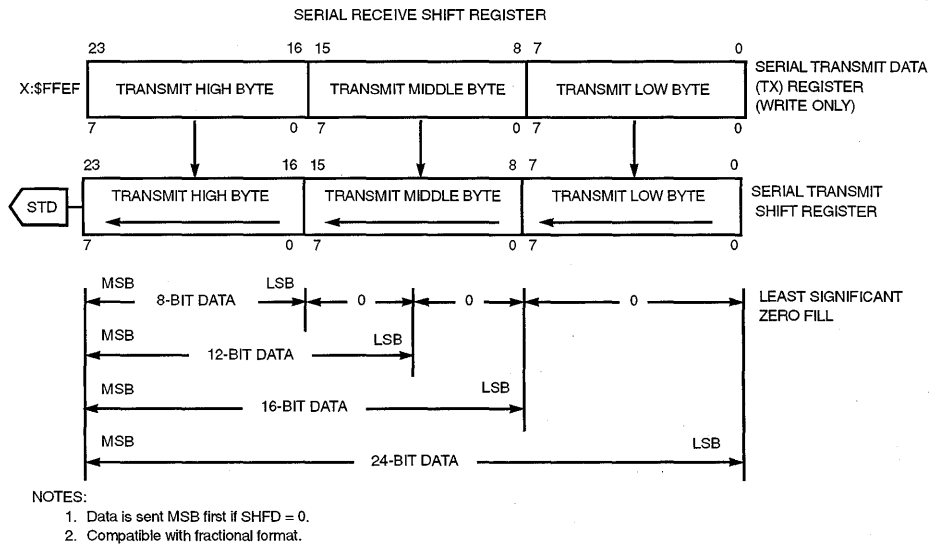


Figure 12-36. SSI Programming Model — Control and Status Registers



NOTES:
1. Data is received MSB first if SHFD = 0.
2. Compatible with fractional format.

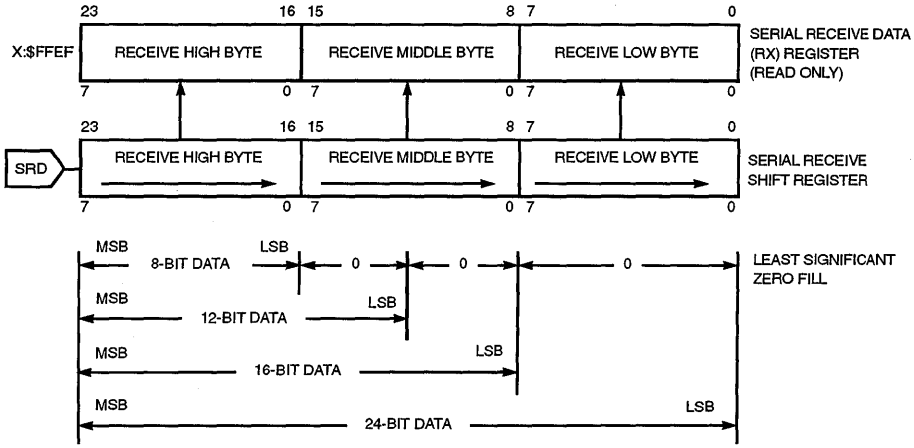
(a) Receive Registers for SHFD = 0



NOTES:
1. Data is sent MSB first if SHFD = 0.
2. Compatible with fractional format.

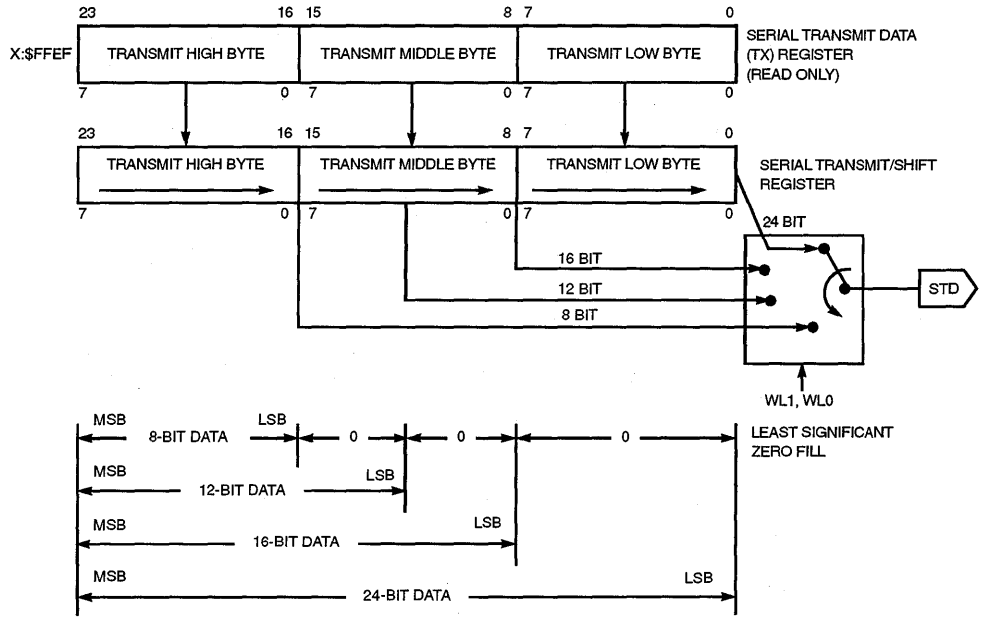
(b) Transmit Registers for SHFD = 0

Figure 12-37. SSI Programming Model (SHFD = 0)



NOTES:
1. Data is received LSB first if SHFD = 1.
2. Compatible with fractional format.

(c) Receive Registers for SHFD = 1



NOTES:
1. Data is received LSB first if SHFD = 1.
2. Compatible with fractional format.

(d) Transmit Registers for SHFD = 1

Figure 12-38. SSI Programming Model (SHFD = 1)

12

shown in Table 12-12. These bits control the number of active clock transitions in the gated clock modes and control the word length divider (see Figure 12-34 and Figure 12-35), which is part of the frame rate signal generator for continuous clock modes. The WL control bits also control the frame sync pulse length when FSL0 and FSL1 select a WL bit clock (see Figure 12-34). Hardware and software reset clear WL0 and WL1.

12.4.2.1.4 CRA Prescaler Range (PSR) Bit 15

The PSR controls a fixed divide-by-eight prescaler in series with the variable prescaler. This bit is used to extend the range of the prescaler for those cases where a slower bit clock is desired (see Figure 12-34). When PSR is cleared, the fixed prescaler is bypassed. When PSR is set, the fixed divide-by-eight prescaler is operational. This allows a 128-kHz master clock to be generated for MC14550x series codecs.

The maximum internally generated bit clock frequency is fosc/4, the minimum internally generated bit clock frequency is fosc/4/8/256=fosc/8192. Hardware and software reset clear PSR.

12.4.2.2 SSI CONTROL REGISTER B (CRB)

The CRB is one of two 16-bit read/write control registers used to direct the operation of the SSI. CRB controls the SSI multifunction pins, SC2, SC1, and SC0, which can be used as clock inputs or outputs, frame synchronization pins, or serial I/O flag pins. The serial output flag control bits and the direction control bits for the serial control pins are in the SSI CRB. Interrupt enable bits for each data register interrupt are provided in this control register. When read by the DSP, CRB appears on the two low-order bytes of the 24-bit word, and the high-order byte reads as zeros. Operating modes are also selected in this register. Hardware and software reset clear all the bits in the CRB. The relationships between the SSI pins (SC0, SC1, SC2, and SCK) and some of the CRB bits are summarized in Tables 12-7, 12-14, and 12-15. The SSI CRB bits are described in the following paragraphs.

12.4.2.2.1 CRB Serial Output Flag 0 (OF0) Bit 0

When the SSI is in the synchronous clock mode and the serial control direction zero bit (SCD0) is set, indicating that the SC0 pin is an output, then data present in OF0 will be written to SC0 at the beginning of the frame in normal mode or at the beginning of the next time slot in network mode. Hardware and software reset clear OF0.

12.4.2.2.2 CRB Serial Output Flag 1 (OF1) Bit 1

When the SSI is in the synchronous clock mode and the serial control direction one (SCD1) bit is set, indicating that the SC1 pin is an output, then data present in OF1 will be written to the SC1 pin at the beginning of the frame in normal mode or at the beginning of the next time slot in network mode (see 12.4.7 Operating Modes – Normal, Network, and On-Demand).

The normal sequence for setting output flags when transmitting data is to poll TDE (TX empty), to first write the flags, and then write the transmit data to the TX register. OF0 and OF1 are double buffered so that the flag states appear on the pins when the TX data is transferred to the transmit shift register (i.e., the flags are synchronous with the data). Hardware and software reset clear OF1.

NOTE

The optional serial output pins (SC0, SC1, and SC2) are controlled by the frame timing and are not affected by TE or RE.

12.4.2.2.3 CRB Serial Control 0 Direction (SCD0) Bit 2

SCD0 controls the direction of the SC0 I/O line. When SCD0 is cleared, SC0 is an input; when SCD0 is set, SC0 is an output (see Tables 12-7 and 12-8, and Figure 12-39). Hardware and software reset clear SCD0.

12.4.2.2.4 CRB Serial Control 1 Direction (SCD1) Bit 3

SCD1 controls the direction of the SC1 I/O line. When SCD1 is cleared, SC1 is an input; when SCD1 is set, SC1 is an output (see Tables 12-7 and 12-8 and Figure 12-39). Hardware and software reset clear SCD1.

12.4.2.2.5 CRB Serial Control 2 Direction (SCD2) Bit 4

SCD2 controls the direction of the SC2 I/O line. When SCD2 is cleared, SC2 is an input; when SCD2 is set, SC2 is an output (see Tables 12-7 and 12-8, and Figure 12-39). Hardware and software reset clear SCD2.

12.4.2.2.6 CRB Clock Source Direction (SCKD) Bit 5

SCKD selects the source of the clock signal used to clock the transmit shift register in the asynchronous mode and both the transmit shift register and the receive shift register in the synchronous mode. When SCKD is set, the internal clock source becomes the bit clock for the transmit shift register and word length divider and is the output on the SCK pin. When SCKD is cleared, the clock source is external; the internal clock generator is disconnected from the SCK pin, and an external clock source may drive this pin. Hardware and software reset clear SCKD.

12.4.2.2.7 CRB Shift Direction (SHFD) Bit 6

This bit causes the transmit shift register to shift data out MSB first when SHFD equals zero or LSB first when SHFD equals one. Receive data is shifted in MSB first when SHFD equals zero or LSB first when SHFD equals one. Hardware reset and software reset clear SHFD.

12.4.2.2.8 CRB Frame Sync Length (FSL0 and FSL1) Bits 7 and 8

These bits select the type of frame sync to be generated or recognized. If FSL1 equals zero and FSL0 equals zero, a word-length frame sync is selected for both TX and RX that is the length of the data word defined by bits WL1 and WL0. If FSL1 equals one and FSL0 equals zero, a 1-bit clock period frame sync is selected for both TX and RX. When FSL0 equals one, the TX and RX frame syncs are different lengths. Hardware reset and software reset clear FSL0 and FSL1

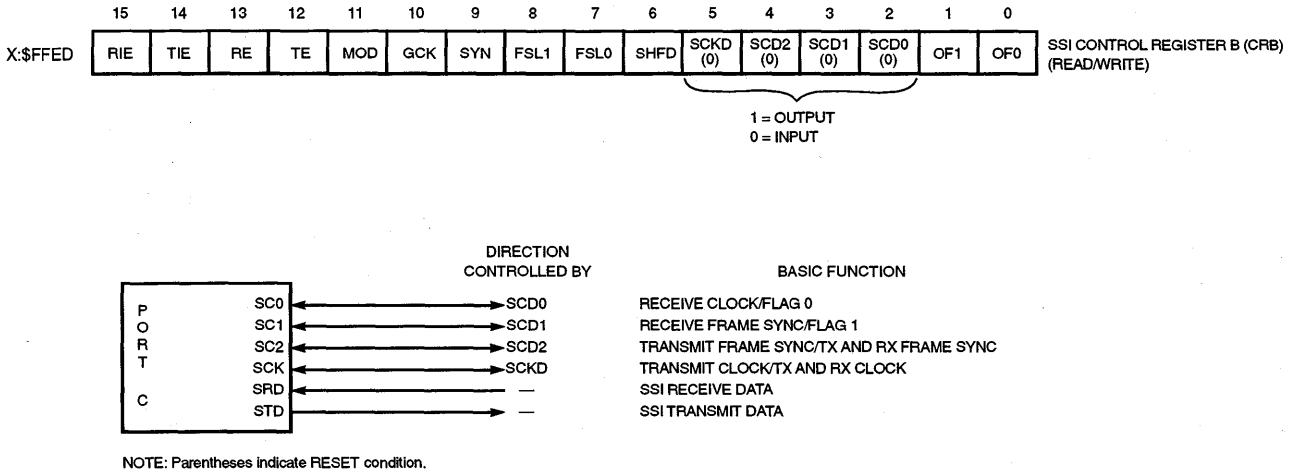


Figure 12-39. Serial Control, Direction Bits

Table 12-13. CRB Frame Sync Length

FSL1	FSL0	Frame Sync Length
0	0	WL bit clock for both TX/RX
0	1	One-bit clock for TX and WL bit clock for RX
1	0	One-bit clock for both TX/RX
1	1	One-bit clock for RX and WL bit clock for TX

12.4.2.2.9 CRB Sync/Async (SYN) Bit 9

SYN controls whether the receive and transmit functions of the SSI occur synchronously or asynchronously with respect to each other. When SYN is cleared, asynchronous mode is chosen and separate clock and frame sync signals are used for the transmit and receive sections. When SYN is set, synchronous mode is chosen and the transmit and receive sections use common clock and frame sync signals. Hardware reset and software reset clear SYN.

12.4.2.2.10 CRB Gated Clock Control (GCK) Bit 10

GCK is used to select between a continuously running data clock or a clock that runs only when there is data to be sent in the transmit shift register. When GCK is cleared, a continuous clock is selected; when GCK is set, the clock will be gated. Hardware reset and software reset clear GCK.

NOTE

For gated clock mode with externally generated bit clock, internally generated frame sync is not defined.

12.4.2.2.11 CRB SSI Mode Select (MOD) Bit 11

MOD selects the operational mode of the SSI. When MOD is cleared, the normal mode is selected; when MOD is set, the network mode is selected. In the normal mode, the frame rate divider determines the word transfer rate – one word is transferred per frame sync during the frame sync time slot. In network mode, a word is (possibly) transferred every time slot. For more details, see 12.4.3 Operational Modes and Pin Definitions. Hardware and software reset clear MOD.

12.4.2.2.12 CRB SSI Transmit Enable (TE) Bit 12

TE enables the transfer of data from TX to the transmit shift register. When TE is set and a frame sync is detected, the transmit portion of the SSI is enabled for that frame. When TE is cleared, the transmitter will be disabled after completing transmission of data currently in the SSI transmit shift register. The serial output is three-stated, and any data present in TX will not be transmitted (i.e., data can be written to TX with TE cleared; TDE will be cleared, but data will not be transferred to the transmit shift register).

The normal mode transmit enable sequence is to write data to TX or TSR before setting TE. The normal transmit disable sequence is to clear TE and TIE after TDE equals one.

In the network mode, the operation of clearing TE and setting it again will disable the transmitter after completing transmission of the current data word until the beginning of the next frame. During that time period, the STD pin will remain in the high-impedance state. Hardware reset and software reset clear TE.

The on-demand mode transmit enable sequence can be the same as the normal mode, or TE can be left enabled.

NOTE

TE does not inhibit TDE or transmitter interrupts. TE does not affect the generation of frame sync or output flags.

12.4.2.2.13 CRB SSI Receive Enable (RE) Bit 13

When RE is set, the receive portion of the SSI is enabled. When this bit is cleared, the receiver will be disabled by inhibiting data transfer into RX. If data is being received while this bit is cleared, the remainder of the word will be shifted in and transferred to the SSI receive data register.

RE must be set in the normal mode and on-demand mode to receive data. In network mode, the operation of clearing RE and setting it again will disable the receiver after reception of the current data word until the beginning of the next data frame. Hardware and software reset clear RE.

NOTE

RE does not inhibit RDF or receiver interrupts. RE does not affect the generation of a frame sync.

12.4.2.2.14 CRB SSI Transmit Interrupt Enable (TIE) Bit 14

The DSP will be interrupted when TIE and the TDE flag in the SSI status register is set. (In network mode, the interrupt takes effect in the next frame synch, not in the next time slot.) When TIE is cleared, this interrupt is disabled. However, the TDE bit will always indicate the transmit data register empty condition even when the transmitter is disabled with the TE bit. Writing data to TX or TSR will clear TDE, thus clearing the interrupt. Hardware and software reset clear RE.

There are two transmit data interrupts that have separate interrupt vectors:

1. Transmit data with exceptions – This interrupt is generated on the following condition:
   TIE=1, TDE=1, and TUE=1
2. Transmit data without exceptions – This interrupt is generated on the following condition:
   TIE=1, TDE=1, and TUE=0

See **Section 7 Processing States** in the *DSP56000 Digital Signal Processor Family Manual* for more information on exceptions.

12.4.2.2.15 CRB SSI Receive Interrupt Enable (RIE) Bit 15

When RIE is set, the DSP will be interrupted when RDF in the SSI status register is set. (In network mode, the interrupt takes effect in the next frame synch, not in the next time slot.) When RIE is cleared, this interrupt is disabled. However, the RDF bit still indicates the receive data register full condition. Reading the receive data register will clear RDF, thus clearing the pending interrupt. Hardware and software reset clear RIE.

There are two receive data interrupts that have separate interrupt vectors:

1. Receive data with exceptions – This interrupt is generated on the following condition:
   RIE=1, RDF=1, and ROE=1
2. Receive data without exceptions – This interrupt is generated on the following condition:
   RIE=1, RDF=1, and ROE=0

See **Section 7 Processing States** in the *DSP56000 Digital Signal Processor Family Manual* for more information on exceptions.

12.4.2.3 SSI STATUS REGISTER (SSISR)

The SSISR is an 8-bit read-only status register used by the DSP to interrogate the status and serial input flags of the SSI. When the SSISR is read to the internal data bus, the register contents occupy the low-order byte of the data bus, and the high-order portion is zero filled. The status bits are described in the following paragraphs.

12.4.2.3.1 SSISR Serial Input Flag 0 (IF0) Bit 0

The SSI latches data present on the SC0 pin during reception of the first received bit after frame sync is detected. IF0 is updated with this data when the receive shift register is transferred into the receive data register. The IF0 bit is enabled only when SCD0 is cleared and SYN is set, indicating that SC0 is an input and the synchronous mode is selected (see Table 12-7); otherwise, IF0 reads as a zero when it is not enabled. Hardware, software, SSI individual, and STOP reset clear IF0.

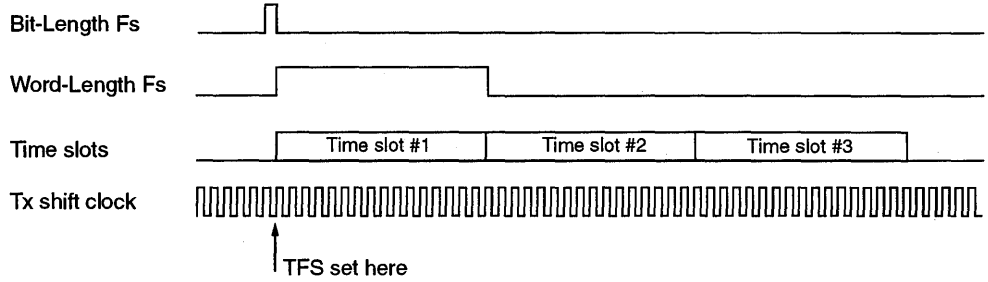
12.4.2.3.2 SSISR Serial Input Flag 1 (IF1) Bit 1

The SSI latches data present on the SC1 pin during reception of the first received bit after frame sync is detected. The IF1 flag is updated with the data when the receiver shift register is transferred into the receive data register. The IF1 bit is enabled only when SCD1 is cleared and SYN is set, indicating that SC1 is an input and the synchronous mode is selected (see Table 12-7); otherwise, IF1 reads as a zero when it is not enabled. Hardware, software, SSI individual, and STOP reset clear IF1.

12.4.2.3.3 SSISR Transmit Frame Sync Flag (TFS) Bit 2

When set, TFS indicates that a transmit frame sync occurred in the current time slot. TFS is set at the start of the first time slot in the frame and cleared during all other time slots. If word-wide transmit frame sync is selected (FSL0=FSL1), this indicates that the frame sync was high at least at the beginning of the time slot if external frame sync is selected, or high throughout the time slot if internal frame sync was selected. If bit-wide transmit frame sync is selected (FSL0≠FSL1), this indicates that the frame sync (either internal or external) was

high during the last Tx clock bit period prior to the current time slot, and that the frame sync falling edge corresponds to the assertion of the first output data bit, as shown below.



Data written to the transmit data register during the time slot when TFS is set will be transmitted (in network mode) during the second time slot in the frame. TFS is useful in network mode to identify the start of the frame. This is illustrated in a typical transmit interrupt handler:

```
                MOVEP     X:(R4)+,X:SSITx
                JCLR      #2,X:SSISR,_NoTFS;1 = FIRST TIME SLOT
                          ;Do something
                JMP       _DONE
_NoTFS
                          ;Do something else
_DONE
```

NOTE

In normal mode, TFS will always read as a one when transmitting data because there is only one time slot per frame – the "frame sync" time slot.

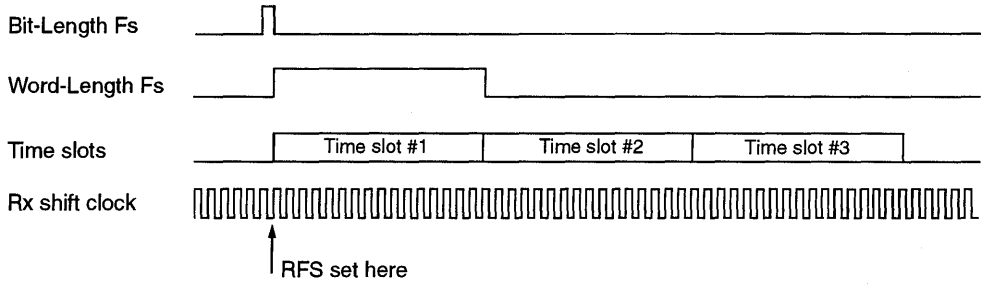
12

TFS, which is cleared by hardware, software, SSI individual, or STOP reset, is not affected by TE.

12.4.2.3.4 SSISR Receive Frame Sync Flag (RFS) Bit 3

When set, RFS indicates that a receive frame sync occurred during reception of the word in the serial receive data register. This indicates that the data word is from the first time slot in the frame. If word-wide receive frame sync is selected (FSL1=0), this indicates that the frame sync was high at least at the beginning of the time slot. If bit-wide receive frame sync is selected (FSL1=1), this indicates that the frame sync (either internal or external) was high

during the last bit period prior to the current time slot, and that the frame sync falling edge corresponds to the assertion of the first output data bit, as shown below.



When RFS is clear and a word is received, it indicates (only in network mode) that the frame sync did not occur during reception of that word. RFS is useful in network mode to identify the start of the frame. This is illustrated in a typical receive interrupt handler:

```
        MOVEPX:SSIRx,X:(R4)+
        JCLR#3,X:SSISR,_NoRFS;1 = FIRST TIME SLOT
         ;Do something
        JMP_DONE
_NoRFS
         ;Do something else
_DONE
```

NOTE

In normal mode, RFS will always read as a one when reading data because there is only one time slot per frame – the "frame sync" time slot.

RFS, which is cleared by hardware, software, SSI individual, or STOP reset, is not affected by RE.

12

12.4.2.3.5 SSISR Transmitter Underrun Error Flag (TUE) Bit 4

TUE is set when the serial transmit shift register is empty (no new data to be transmitted) and a transmit time slot occurs. When a transmit underrun error occurs, the previous data (which is still present in the TX) will be retransmitted.

In the normal mode, there is only one transmit time slot per frame. In the network mode, there can be up to 32 transmit time slots per frame.

TUE does not cause any interrupts; however, TUE does cause a change in the interrupt vector used for transmit interrupts so that a different interrupt handler may be used for a transmit underrun condition. If a transmit interrupt occurs with TUE set, the transmit data with exception status interrupt will be generated; if a transmit interrupt occurs with TUE clear, the transmit data without errors interrupt will be generated.

Hardware, software, SSI individual, and STOP reset clear TUE. TUE is also cleared by reading the SSISR with TUE set, followed by writing TX or TSR.

12.4.2.3.6 SSISR Receiver Overrun Error Flag (ROE) Bit 5

This flag is set when the serial receive shift register is filled and ready to transfer to the receiver data register (RX) and RX is already full (i.e., RDF=1). The receiver shift register is not transferred to RX. ROE does not cause any interrupts; however, ROE does cause a change in the interrupt vector used for receive interrupts so that a different interrupt handler may be used for a receive error condition. If a receive interrupt occurs with ROE set, the receive data with exception status interrupt will be generated; if a receive interrupt occurs with ROE clear, the receive data without errors interrupt will be generated.

Hardware, software, SSI individual, and STOP reset clear ROE. ROE is also cleared by reading the SSISR with ROE set, followed by reading the RX. Clearing RE does not affect ROE.

12.4.2.3.7 SSISR SSI Transmit Data Register Empty (TDE) Bit 6

This flag is set when the contents of the transmit data register are transferred to the transmit shift register; it is also set for a disabled time slot period in network mode (as if data were being transmitted after the TSR was written). Thirdly, it can be set by the hardware, software, SSI individual, or STOP reset. When set, TDE indicates that data should be written to the TX or to the time slot register (TSR). TDE is cleared when the DSP writes to the transmit data register or when the DSP writes to the TSR to disable transmission of the next time slot. If TIE is set, a DSP transmit data interrupt request will be issued when TDE is set. The vector of the interrupt will depend on the state of the transmitter underrun bit.

12.4.2.3.8 SSISR SSI Receive Data Register Full (RDF) Bit 7

RDF is set when the contents of the receive shift register are transferred to the receive data register. RDF is cleared when the DSP reads the receive data register or cleared by hardware, software, SSI individual, or STOP reset. If RIE is set, a DSP receive data interrupt request will be issued when RDF is set. The vector of the interrupt request will depend on the state of the receiver overrun bit.

12.4.2.3.9 SSI Receive Shift Register

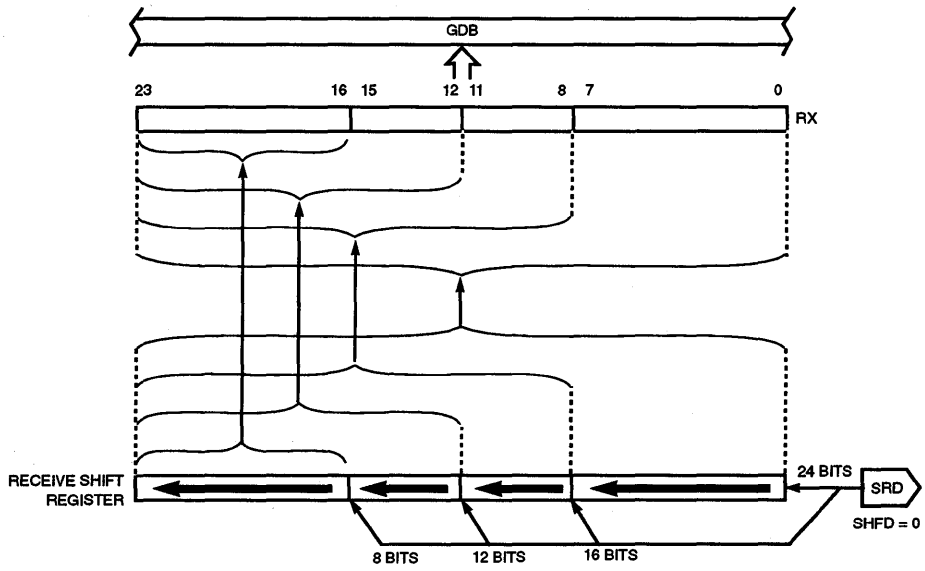
This 24-bit shift register receives the incoming data from the serial receive data pin. Data is shifted in by the selected (internal/external) bit clock when the associated frame sync I/O (or gated clock) is asserted. Data is assumed to be received MSB first if SHFD equals zero and LSB first if SHFD equals one. Data is transferred to the SSI receive data register after 8, 12, 16, or 24 bits have been shifted in, depending on the word-length control bits in the CRA (see Figure 12-40).

12.4.2.3.10 SSI Receive Data Register (RX)

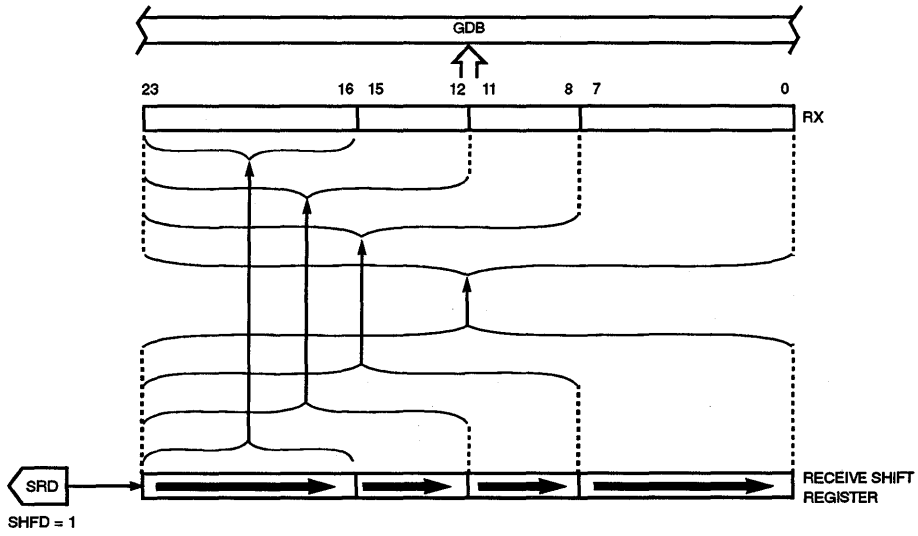
RX is a 24-bit read-only register that accepts data from the receive shift register as it becomes full. The data read will occupy the most significant portion of the receive data register (see Figure 12-40). The unused bits (least significant portion) will read as zeros. The DSP is interrupted whenever RX becomes full if the associated interrupt is enabled.

12.4.2.3.11 SSI Transmit Shift Register

This 24-bit shift register contains the data being transmitted. Data is shifted out to the serial transmit data pin by the selected (internal/external) bit clock when the associated frame sync



(a) SHFD = 0



(b) SHFD = 1

Figure 12-40. Receive Data Path

I/O (or gated clock) is asserted. The number of bits shifted out before the shift register is considered empty and may be written to again can be 8, 12, 16, or 24 bits (determined by the word-length control bits in CRA). The data to be transmitted occupies the most significant portion of the shift register. The unused portion of the register is ignored. Data is shifted out of this register MSB first if SHFD equals zero and LSB first if SHFD equals one (see Figure 12-41).

12.4.2.3.12 SSI Transmit Data Register (TX)

TX is a 24-bit write-only register. Data to be transmitted is written into this register and is automatically transferred to the transmit shift register. The data written (8, 12, 16, or 24 bits) should occupy the most significant portion of TX (see Figure 12-41). The unused bits (least significant portion) of TX are don't care bits. The DSP is interrupted whenever TX becomes empty if the transmit data register empty interrupt has been enabled.

12.4.2.3.13 Time Slot Register (TSR)

TSR is effectively a null data register that is used when the data is not to be transmitted in the available transmit time slot. For the purposes of timing, TSR is a write-only register that behaves like an alternative transmit data register, except that, rather than transmitting data, the transmit data pin is in the high-impedance state for that time slot.

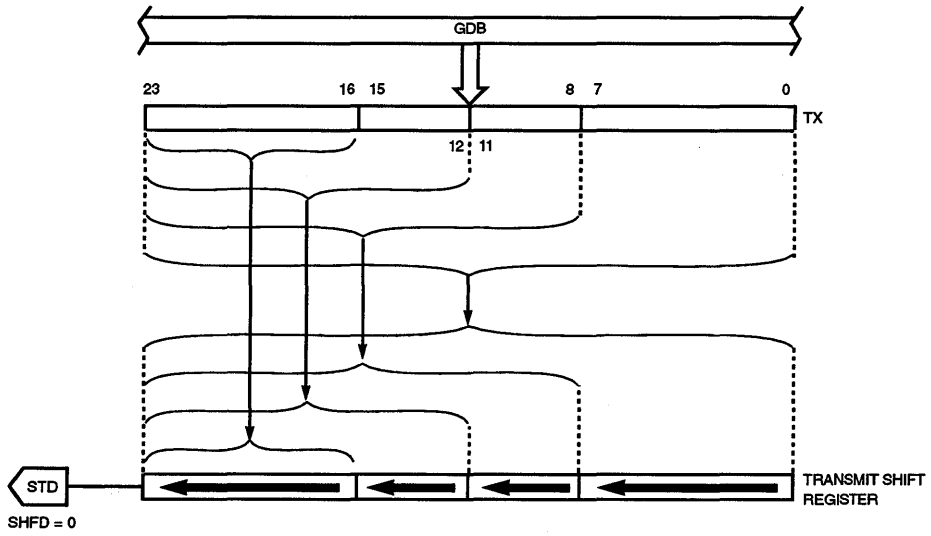
12.4.3 Operational Modes and Pin Definitions

Table 12-14 and Table 12-15 completely describe the SSI operational modes and pin definitions (Table 12-7 is a simplified version of these tables). The operational modes are as follows:

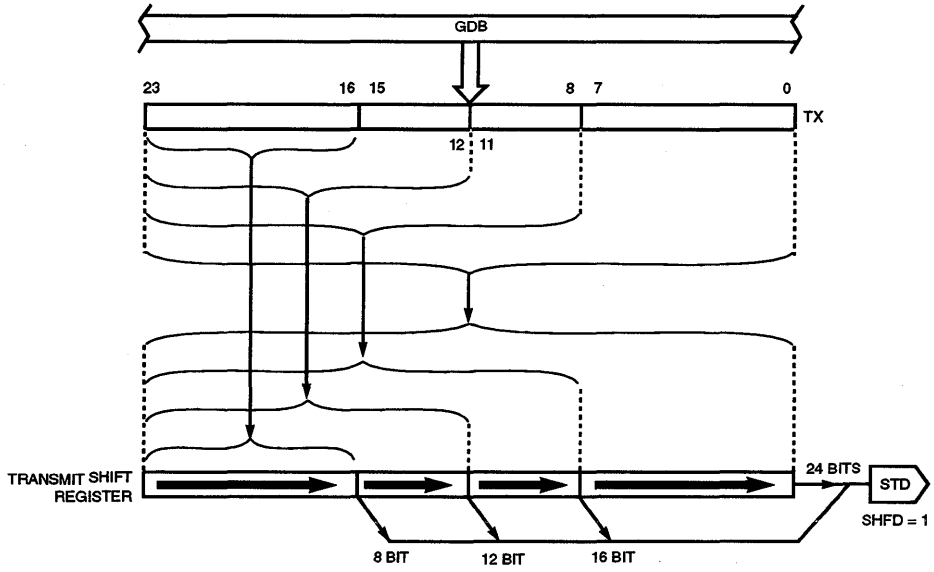
1. Continuous Clock
   - Mode 1 – Normal with Internal Frame Sync
   - Mode 2 – Network with Internal Frame Sync
   - Mode 3 – Normal with External Frame Sync
   - Mode 4 – Network with External Frame Sync
2. Gated Clock
   - Mode 5 – External Gated Clock
   - Mode 6 – Normal with Internal Gated Clock
   - Mode 7 – Network with Internal Gated Clock
3. Special Case (Both Gated and Continuous Clock)
   - Mode 8 – On-Demand Mode (Transmitter Only)
   - Mode 9 – Receiver Follows Transmitter Clocking

12.4.4 Registers After Reset

Hardware or software reset clears the port control register bits, which configure all I/O as general-purpose input. The SSI will remain in reset while all SSI pins are programmed as general-purpose I/O (CC8–CC3=0) and will become active only when at least one of the SSI I/O pins is programmed as not general-purpose I/O. Table 12-16 shows how each type of reset affects each SSI register bit.



(a) SHFD = 0



(b) SHFD = 1

Figure 12-41. Transmit Data Path

Table 12-14. Mode and Pin Definition Table – Continuous Clock

Control Bits								Mode		SC0		SC1		SC2		SCK	
MOD	GCLK	SYN	SCD2	SCD1	SCD0	SCKD	DC4-DC0	TX	RX	In	Out	In	Out	In	Out	In	Out
0	0	0	1	1	X	X	X	1	1	RXC	RXC	—	FSR	—	FST	TXC	TXC
0	0	1	1	X	X	X	X	1	1	F0	F0	F1	F1	—	FS*	*XC	*XC
1	0	0	1	1	X	X	1	2	2	RXC	RXC	—	FSR	—	FST	TXC	TXC
1	0	1	1	X	X	X	1	2	2	F0	F0	F1	F1	—	FS*	*XC	*XC
0	0	0	0	1	X	X	X	3	1	RXC	RXC	—	FSR	FST	—	TXC	TXC
0	0	0	1	0	X	X	X	1	3	RXC	RXC	FSR	—	—	FST	TXC	TXC
0	0	0	0	0	X	X	X	3	3	RXC	RXC	FSR	—	FST	—	TXC	TXC
0	0	1	0	X	X	X	X	3	3	F0	F0	F1	F1	FS*	—	*XC	*XC
1	0	0	0	1	X	X	X	4	2	RXC	RXC	—	FSR	FST	—	TXC	TXC
1	0	0	1	0	X	X	1	2	4	RXC	RXC	FSR	—	—	FST	TXC	TXC
1	0	0	0	0	X	X	X	4	4	RXC	RXC	FSR	—	FST	—	TXC	TXC
1	0	1	0	X	X	X	X	4	4	F0	F0	F1	F1	FS*	—	*XC	*XC
1	0	0	1	1	X	X	0	8	2	RXC	RXC	—	FSR	—	FST	TXC	TXC
1	0	1	1	X	X	X	0	8	9	F0	F0	F1	F1	—	FS*	*XC	*XC
1	0	0	1	0	X	X	0	8	4	RXC	RXC	FSR	—	—	FST	TXC	TXC

DC4-DC0 = 0 means that bits DC4 = 0, DC3 = 0, DC2 = 0, DC1 = 0, and DC0 = 0
DC4-DC0 = 1 means that bits DC4-DC0≠0
TXC — Transmitter Clock
RXC — Receiver Clock
*XC — Transmitter/Receiver Clock (Synchronous Operation)
FST — Transmitter Frame Sync
FSR — Receiver Frame Sync
FS* — Transmitter/Receiver Frame Sync (Synchronous Operation)
F0 — Flag 0
F1 — Flag 1

12.4.5 SSI Initialization

The correct way to initialize the SSI is as follows:

1. Hardware, software, SSI individual, or STOP reset
2. Program SSI control registers
3. Configure SSI pins (at least one) as not general-purpose I/O

During program execution, CC8–CC3 may be cleared, causing the SSI to stop serial activity and enter the individual reset state. All status bits of the interface will be set to their reset

Table 12-15. Mode and Pin Definition Table – Gated Clock

Control Bits								Mode		SC0		SC1		SC2		SCK	
MOD	GCLK	SYN	SCD2	SCD1	SCD0	SCKD	DC4-DC0	TX	RX	In	Out	In	Out	In	Out	In	Out
0	1	0	X	X	1	1	X	6	6	—	RXC	?	FSR	?	FST	—	TXC
0	1	1	X	X	X	1	X	6	6	F0	F0	F0	F1	?	FS*	—	*XC
0	1	0	X	X	1	0	X	5	6	—	RXC	?	FSR	?	?	TXC	—
0	1	0	X	X	0	0	X	5	5	RXC	—	?	?	?	?	TXC	—
0	1	1	X	X	X	0	X	5	5	F0	F0	F1	F1	?	?	*XC	—
1	1	0	X	X	1	1	0	8	7	—	RXC	?	FSR	?	FST	—	TXC
1	1	0	X	X	0	1	0	8	5	RXC	—	?	?	?	FST	—	TXC
1	1	1	X	X	X	1	0	8	9	F0	F0	F1	F1	?	FS*	—	*XC
0	1	0	X	X	0	1	X	6	5	RXC	—	?	?	?	FST	—	TXC

DC4–DC0=0 means that bits DC4=0, DC3=0, DC2=0, DC1=0, and DC0=0.
TXC – Transmitter Clock
RXC – Receiver Clock
*XC – Transmitter/Receiver Clock (Synchronous Operation)
FST – Transmitter Frame Sync
FSR – Receiver Frame Sync
FS* – Transmitter/Receiver Frame Sync (Synchronous Operation)
F0 – Flag 0
F1 – Flag 1
? – Undefined

state; however, the contents of CRA and CRB are not affected. This procedure allows the DSP program to reset each interface separately from the other internal peripherals.

The DSP program must use an SSI reset when changing the MOD, GCK, SYN, SCKD, SCD2, SCD1, or SCD0 bits to ensure proper operation of the interface. Figure 12-42 is a

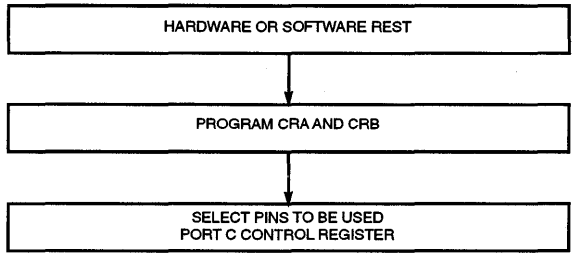


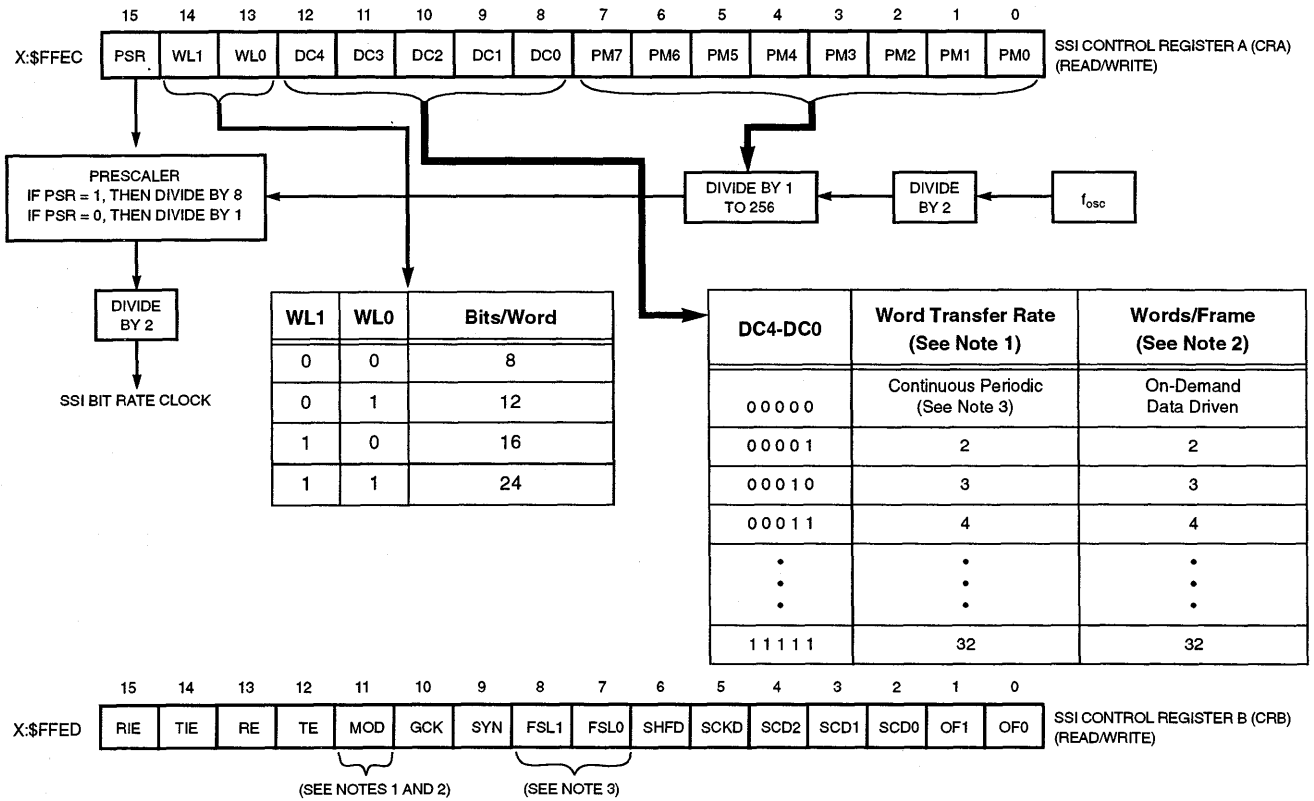
Figure 12-42. SSI Initialization Block Diagram

flowchart illustrating the three initialization steps previously listed. Figure 12-43, Figure 12-44, and Figure 12-45 provide additional detail to the flowchart.

Table 12-16. SSI Registers After Reset

Register Name	Register Data	Bit Number	Reset			
			HW Reset	SW Reset	Individual Reset	ST Reset
CRA	PSR	15	0	0	–	–
	WL(2–0)	13,14	0	0	–	–
	DC(4–0)	8–12	0	0	–	–
	PM(7–0)	0–7	0	0	–	–
CRB	RIE	15	0	0	–	–
	TIE	14	0	0	–	–
	RE	13	0	0	–	–
	TE	12	0	0	–	–
	MOD	11	0	0	–	–
	GCK	10	0	0	–	–
	SYN	9	0	0	–	–
	FSL1	8	0	0	–	–
	FSL0	7	0	0	–	–
	SHFD	6	0	0	–	–
	SCKD	5	0	0	–	–
	SCD(2–0)	2–4	0	0	–	–
	OF(1–0)	0,1	0	0	–	–
SSISR	RDF	7	0	0	0	0
	TDE	6	1	1	1	1
	ROE	5	0	0	0	0
	TUE	4	0	0	0	0
	RFS	3	0	0	0	0
	TFS	2	0	0	0	0
	IF(1–0)	0,1	0	0	0	0
RDR	RDR (23–0)	23–0	–	–	–	–
TDR	TDR (23–0)	23–0	–	–	–	–
RSR	RDR (23–0)	23–0	–	–	–	–
TSR	RDR (23–0)	23–0	–	–	–	–

NOTES:
1. RSR – SSI receive shift register
2. TSR – SSI transmit shift register
3. HW – Hardware reset is caused by asserting the external pin $\overline{\text{RESET}}$.
4. SW – Software reset is caused by executing the RESET instruction.
5. IR – Individual reset is caused by SSI peripheral pins (i.e., PCC(3–8)) being configured as general-purpose I/O.
6. ST – Stop reset is caused by executing the STOP instruction.



WL1	WL0	Bits/Word
0	0	8
0	1	12
1	0	16
1	1	24

DC4-DC0	Word Transfer Rate (See Note 1)	Words/Frame (See Note 2)
00000	Continuous Periodic (See Note 3)	On-Demand Data Driven
00001	2	2
00010	3	3
00011	4	4
• • •	• • •	• • •
11111	32	32

NOTES:
1. NORMAL — MOD = 0
2. NETWORK — MOD = 1
3. FSL1 = 1, FSL0 = 0

Figure 12-43. SSI CRA Initialization Procedure

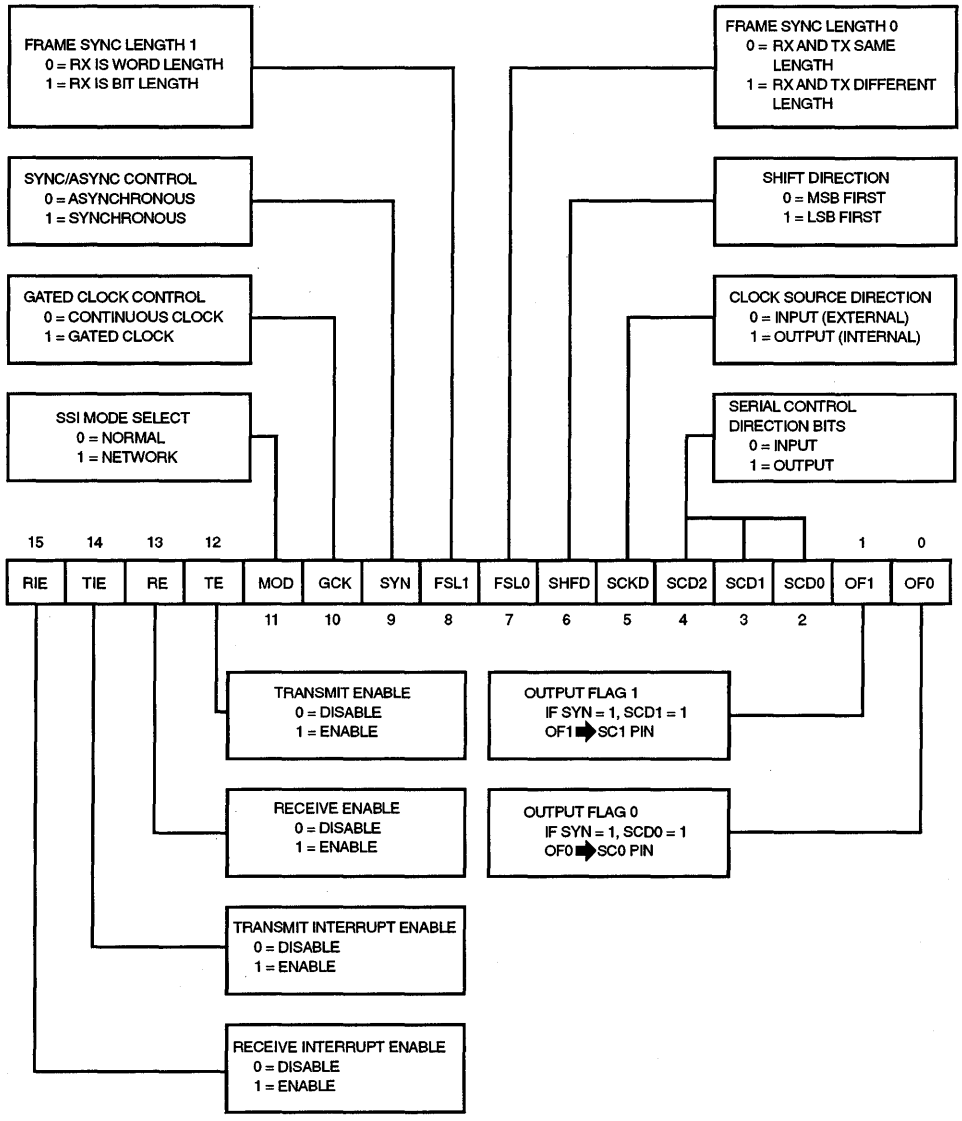


Figure 12-44. SSI CRB Initialization Procedure

Figure 12-45 shows the six control bits in the PCC, which select the six SSI pins as either general-purpose I/O or as SSI pins. The STD pin can only transmit data; the SRD pin can only receive data. The other four pins can be inputs or outputs, depending on how they are programmed. This programming is accomplished by setting bits in CRA and CRB as shown in Figure 12-39. The CRA (see Figure 12-43) sets the SSI bit rate clock with PSR and PM0–PM7, sets the word length with WL1 and WL0, and sets the number of words in a frame with DC0–DC4. There is a special case where DC4–DC0 equals zero (one word per frame). De-

pending on whether the normal or network mode is selected (MOD=0 or MOD=1, respectively), either the continuous periodic data mode is selected, or the on-demand data driven mode is selected. The continuous periodic mode requires that FSL1 equals one and FSL0 equals zero. Figure 12-44 shows the meaning of each individual bit in the CRB. These bits should be set according to the application requirements.

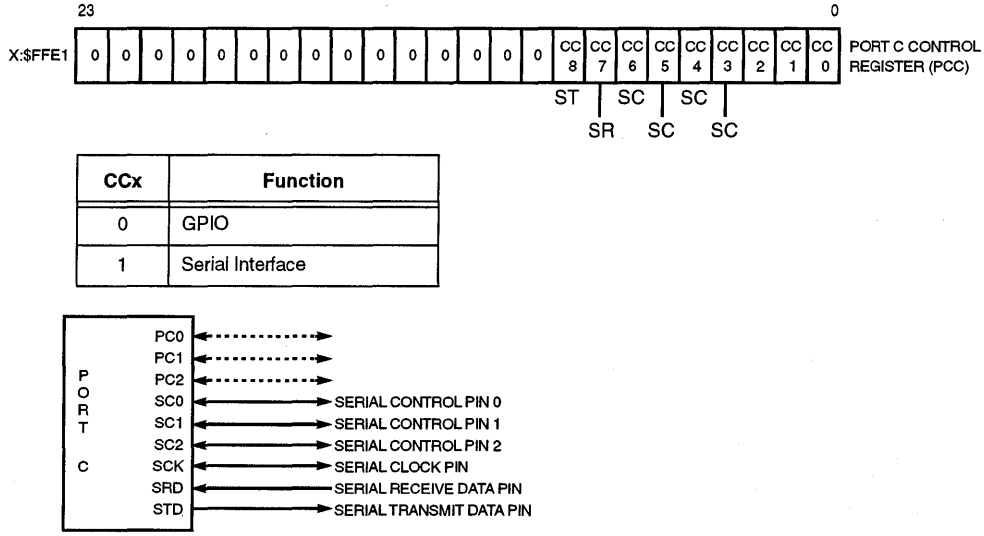


Figure 12-45. SSI Initialization Procedure

Table 12-17(a) and Table 12-17(b) provide a convenient listing of PSR and PM0–PM7 settings for the common data communication rates and the highest rate possible for the SSI for the chosen crystal frequencies. The crystal frequency selected for Table 12-17(a) is the one used by the DSP56002ADS board; the one selected for Table 12-17(b) is the closest one to 40 MHz that divides down to exactly 128 kHz. If an exact baud rate is required, the crystal frequency may have to be selected. Table 12-17 gives the PSR and PM0–PM7 settings in addition to the required crystal frequency for three common telecommunication frequencies.

12.4.6 SSI Exceptions

The SSI can generate four different exceptions (see Figure 12-46 and Figure 12-47):

1. SSI Receive Data – occurs when the receive interrupt is enabled, the receive data register is full, and no receive error conditions exist. Reading RX clears the pending interrupt. This error-free interrupt can use a fast interrupt service routine for minimum overhead.
2. SSI Receive Data with Exception Status – occurs when the receive interrupt is enabled, the receive data register is full, and a receiver overrun error has occurred. ROE is cleared by first reading the SSISR and then reading RX.

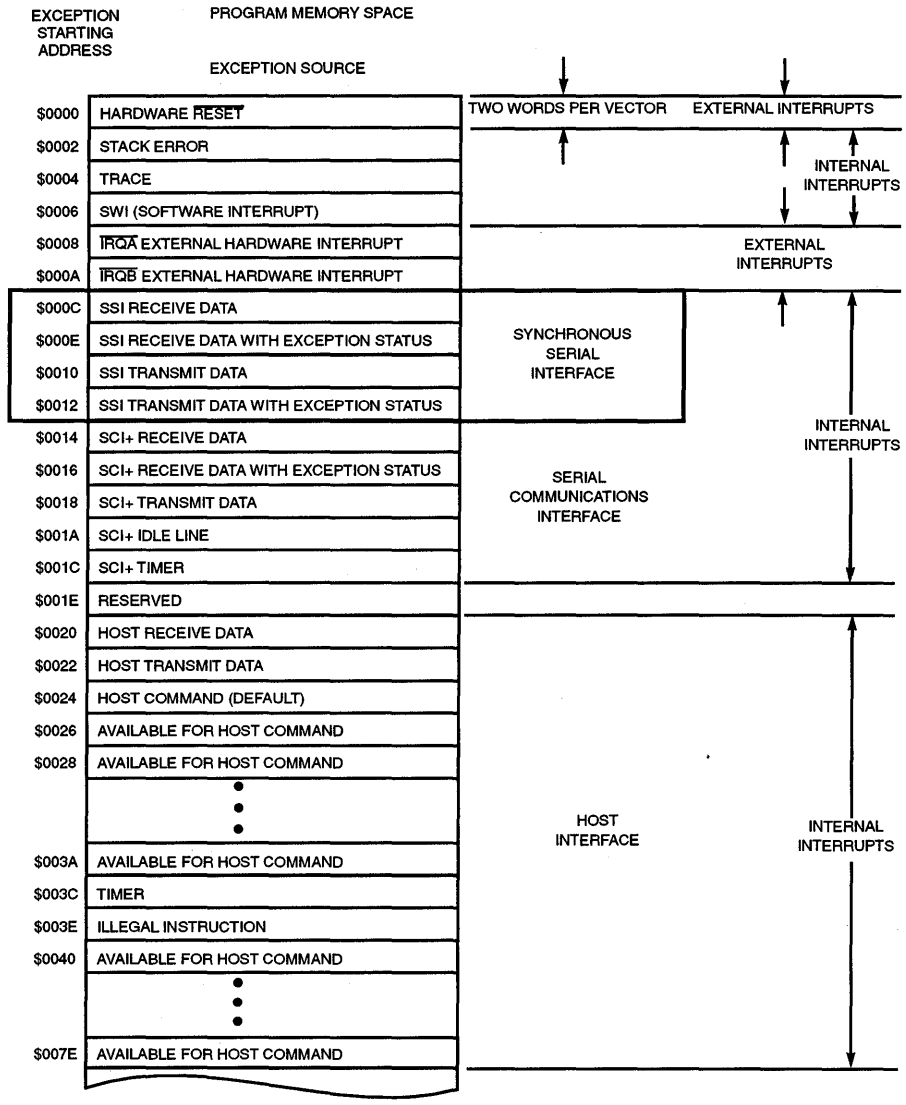


Figure 12-46. SSI Exception Vector Locations

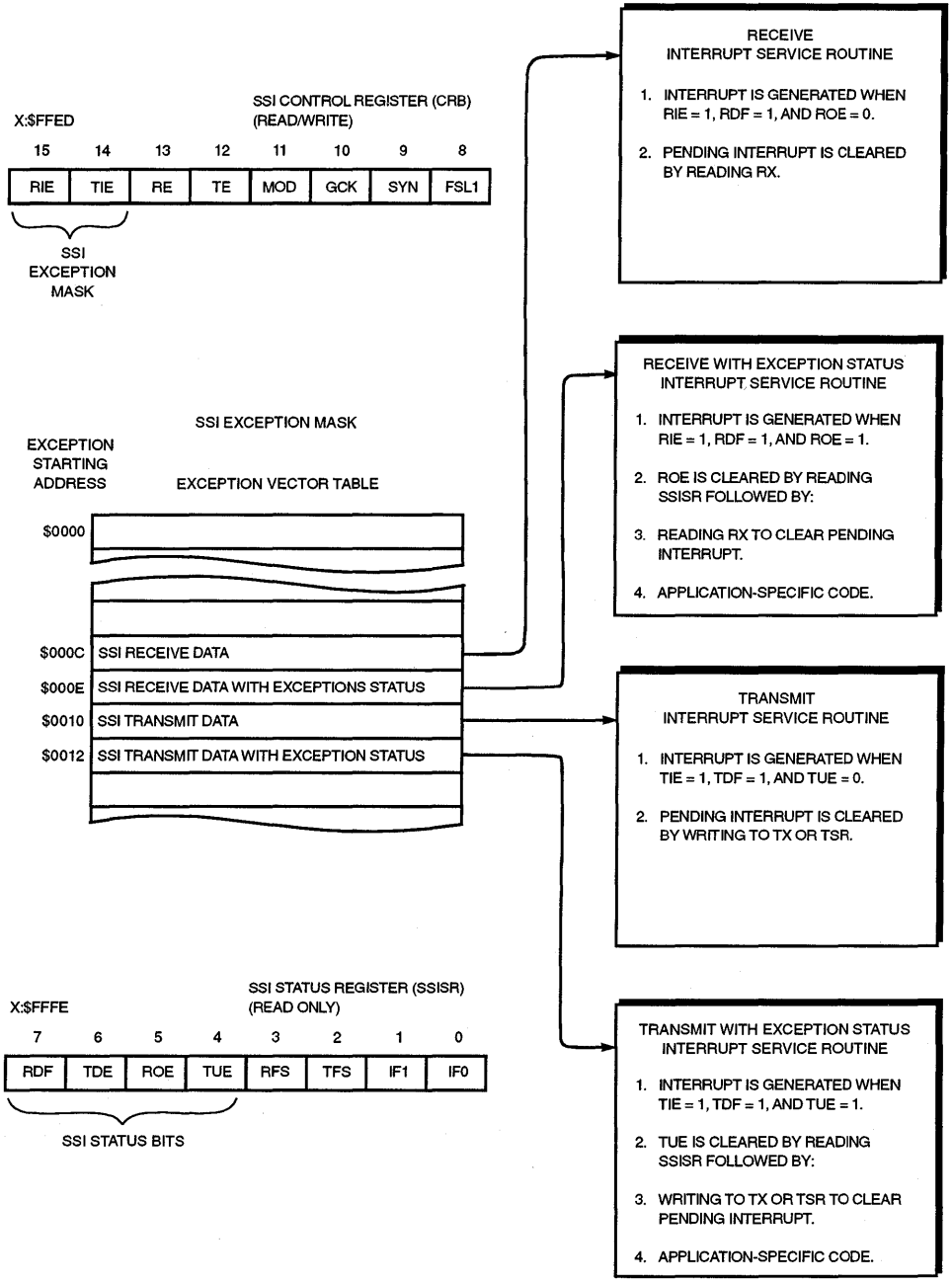


Figure 12-47. SSI Exceptions

Table 12-17. (a) SSI Bit Rates for a 40-MHz Crystal

Bit Rate (BPS)	PSR	PM
1000	1	$4E1
2000	1	$270
4000	1	$138
8000	1	$9B
16K	1	$4D
32K	1	$26
64K	0	$9B
128K	0	$4D
10M	0	$00

BPS = $f_{osc} \div (4 \times (7(PSR) + 1) \times (PM + 1))$ where
f_{osc}=40 MHz
PSR = 0 or 1
PM = 0 to $FFF

Table 12-17. (b) SSI Bit Rates for a 39.936-MHz Crystal

Bit Rate (BPS)	PSR	PM
1000	1	$4DF
2000	1	$26F
4000	1	$137
8000	1	$9B
16K	1	$4D
32K	1	$26
64K	0	$9B
128K	0	$4D
9.984M	0	$00

BPS = $f_{osc} \div (4 \times (7(PSR) + 1) \times (PM + 1))$ where
f_{osc}=39.936 MHz
PSR = 0 or 1
PM = 0 to $FFF

Table 12-18. Crystal Frequencies Required for Codecs

Bit Rate (BPS)	PSR	PM	Crystal Frequency
1.536M	0	$05	36.864 MHz
1.544M	0	$05	37.056 MHz
2.048M	0	$03	28.672 MHz

BPS = $f_{osc} \div (4 \times (7(PSR) + 1) \times (PM + 1))$
PSR = 0 or 1
PM = 0 to $FFF

3. SSI Transmit Data – occurs when the transmit interrupt is enabled, the transmit data register is empty, and no transmitter error conditions exist. Writing to TX or the TSR will clear this interrupt. This error-free interrupt may use a fast interrupt service routine for minimum overhead.
4. SSI Transmit Data with Exception Status – occurs when the transmit interrupt is enabled, the transmit data register is empty, and a transmitter underrun error has occurred. TUE is cleared by first reading the SSISR and then writing to TX or the TSR to clear the pending interrupt.

Table 12-19. SSI Operating Modes

Operating Format	Serial Clock	TX, RX Sections	Typical Applications
Normal	Continuous	Asynchronous	Single Asynchronous Codec; Stream-Mode Channel Interface
Normal	Continuous	Synchronous	Multiple Synchronous Codecs
Normal	Gated	Asynchronous	DSP-to-DSP; Serial Peripherals (A/D,D/A)
Normal	Gated	Synchronous	SPI-Type Devices; DSP to MCU
Network	Continuous	Asynchronous	TDM Networks
Network	Continuous	Synchronous	TDM Codec Networks, TDM DSP Networks
On Demand	Gated	Asynchronous	Parallel-to-Serial and Serial-to-Parallel Conversion
On Demand	Gated	Synchronous	DSP to SPI Peripherals

12.4.7 Operating Modes – Normal, Network, and On-Demand

The SSI has three basic operating modes and many data/operation formats. These modes can be programmed by several bits in the SSI control registers. Table 12-19 lists the SSI operating modes and some of the typical applications in which they may be used.

The data/operation formats are selected by choosing between gated and continuous clocks, synchronization of transmitter and receiver, selection of word or bit frame sync, and whether the LSB is transferred first or last. The following paragraphs describe how to select a particular data/operation format and describe examples of normal-mode and network-mode applications. The on-demand mode is selected as a special case of the network mode.

The SSI can function as an SPI master or SPI slave, using additional logic for arbitration, which is required because the SSI interface does not perform SPI master/slave arbitration. An SPI master device always uses an internally generated clock; whereas, an SPI slave device always uses an external clock.

12.4.7.1 DATA/OPERATION FORMATS

The data/operation formats available to the SSI are selected by setting or clearing control bits in the CRB. These control bits are MOD, GCK, SYN, FSL1, FSL0, and SHFD.

12.4.7.1.1 Normal/Network Mode Selection

Selecting between the normal mode and network mode is accomplished by clearing or setting the MOD bit in the CRB (see Figure 12-50). For normal mode, the SSI functions with one data word of I/O per frame (see Figure 12-48). For the network mode, 2 to 32 data words of I/O may be used per frame. In either case, the transfers are periodic. The normal mode is typically used to transfer data to/from a single device. Network mode is typically used in time division multiplexed (TDM) networks of codecs or DSPs with multiple words per frame (see Figure 12-49, which shows two words in a frame with either word-length or bit-length frame sync). The frame sync shown in Figure 12-50 is the word-length frame sync. A bit-length frame sync can be chosen by setting FSL1 and FSL0 for the configuration desired.

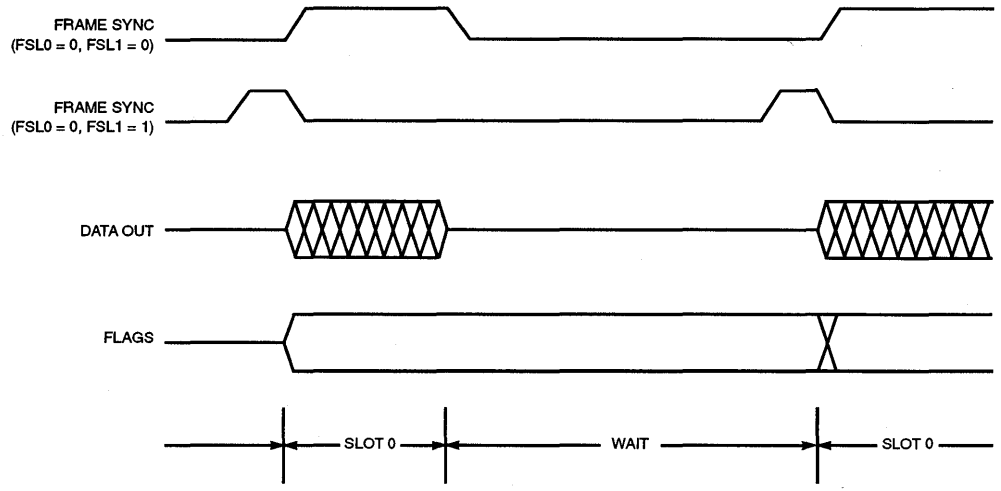


Figure 12-48. Normal Mode, External Frame Sync (8 Bit, 1 Word in Frame)

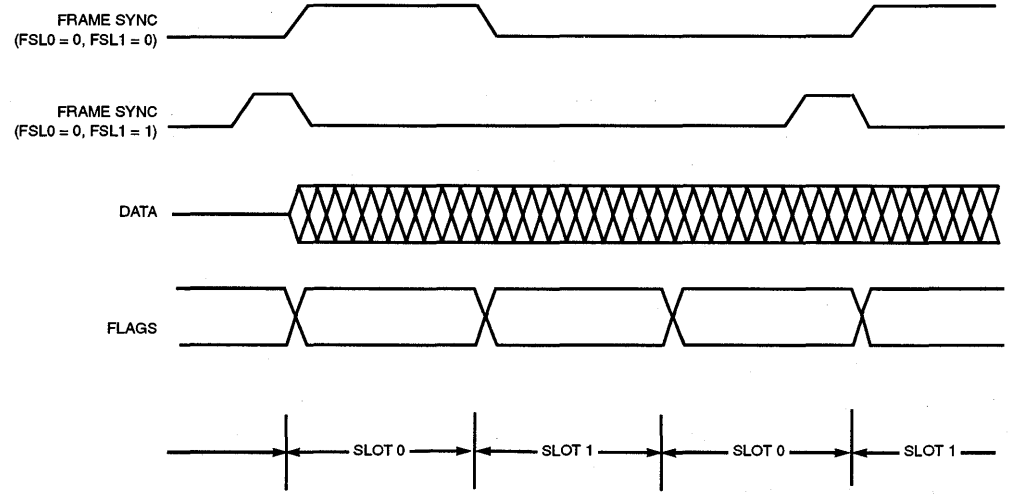


Figure 12-49. Network Mode, External Frame Sync (8 Bit, 2 Words in Frame)

12.4.7.1.2 Continuous/Gated Clock Selection

The TX and RX clocks may be programmed as either continuous or gated clock signals by the GCK bit in the CRB. A continuous TX and RX clock is required in applications such as communicating with some codecs where the clock is used for more than just data transfer. A gated clock, in which the clock only toggles while data is being transferred, is useful for

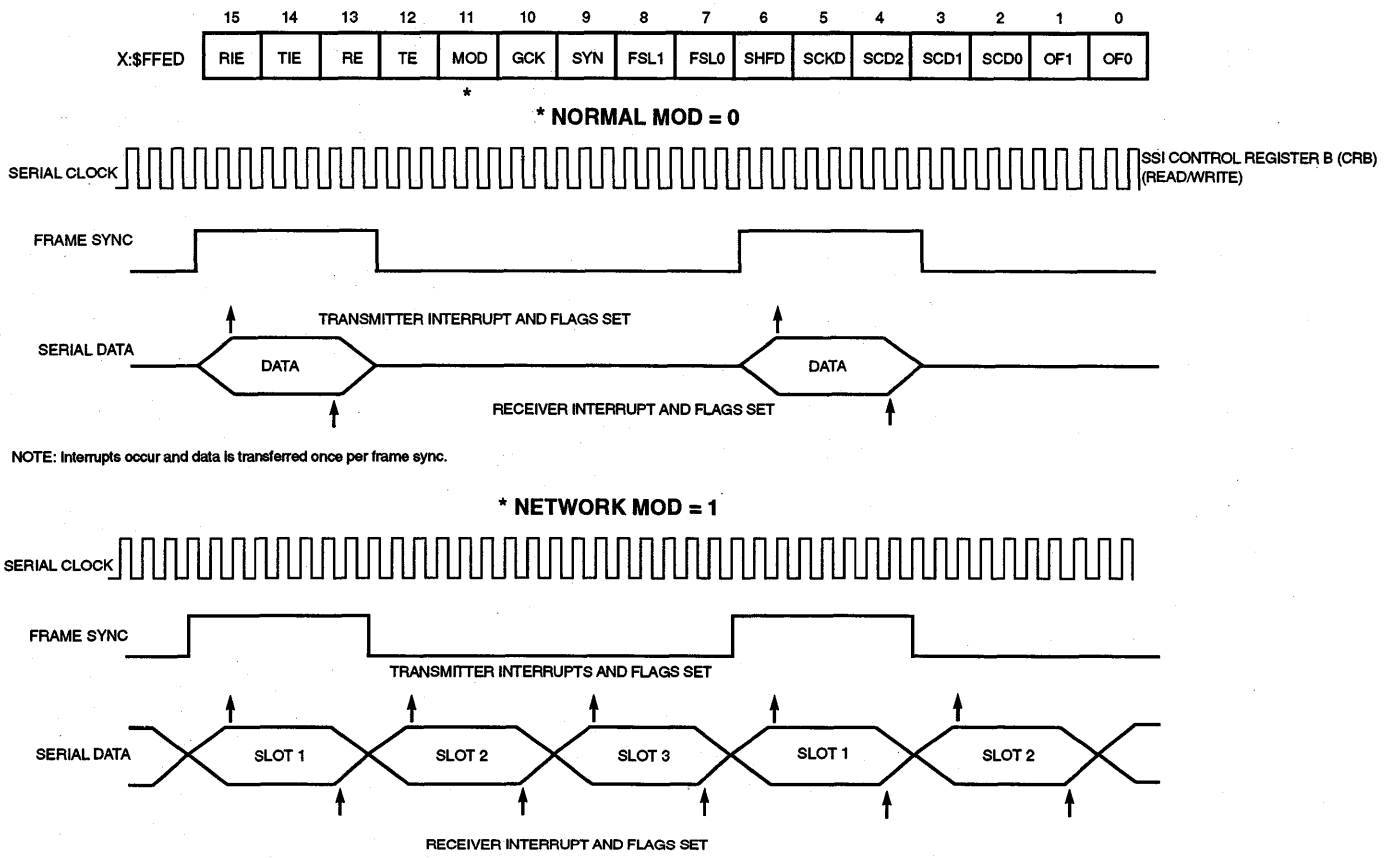


Figure 12-50. CRB MOD Bit Operation

many applications and is required for SPI compatibility. The frame sync outputs may be used as a start conversion signal by some A/D and D/A devices.

Figure 12-51 illustrates the difference between continuous clock and gated clock systems. A separate frame-sync signal is required in continuous clock systems to delimit the active clock transitions. Although the word-length frame sync is shown in Figure 12-51, a bit-length frame sync can be used (see Figure 12-52). In gated clock systems, frame synchronization is inherent in the clock signal; thus a separate sync signal is not required (see Figure 12-53 and Figure 12-54). The SSI can be programmed to generate frame sync outputs in gated clock mode but does not use frame sync inputs.

Input flags (see Figure 12-53 and Figure 12-54) are latched on the negative edge of the first data bit of a frame. Output flags are valid during the entire frame.

12.4.7.1.3 Synchronous/Asynchronous Operating Modes

The transmit and receive sections of this interface may be synchronous or asynchronous – i.e., the transmitter and receiver may use common clock and synchronization signals (synchronous operating mode, see Figure 12-58) or they may have their own separate clock and sync signals (asynchronous operating mode). The SYN bit in CRB selects synchronous or asynchronous operation. Since the SSI is designed to operate either synchronously or asynchronously, separate receive and transmit interrupts are provided.

Figure 12-55 illustrates the operation of the SYN bit in the CRB. When SYN equals zero, the SSI TX and RX clocks and frame sync sources are independent. If SYN equals one, the SSI TX and RX clocks and frame sync come from the same source (either external or internal).

Data clock and frame sync signals can be generated internally by the DSP or may be obtained from external sources. If internally generated, the SSI clock generator is used to derive bit clock and frame sync signals from the DSP internal system clock. The SSI clock generator consists of a selectable fixed prescaler and a programmable prescaler for bit rate clock generation and also a programmable frame-rate divider and a word-length divider for frame-rate sync-signal generation.

Figures 12-56 (a), 12-56 (b), 12-56 (c), and 12-56 (d) show the definitions of the SSI pins during each of the four main operating modes of the SSI I/O interface. Figure 12-56 (a) uses a gated clock (from either an external source or the internal clock), which means that frame sync is inherent in the clock. Since both the transmitter and receiver use the same clock (synchronous configuration), both use the SCK pin. SC0 and SC1 are designated as flags or can be used as general purpose-parallel I/O. SC2 is not defined if it is an input; SC2 is the transmit and receive frame sync if it is an output.

Figure 12-56 (b) shows a gated clock (from either an external source or the internal clock), which means that frame sync is inherent in the clock. Since this configuration is asynchronous, SCK is the transmitter clock pin (input or output) and SC0 is the receiver clock pin (input or output). SC1 and SC2 are designated as receive or transmit frame sync, respectively, if they are selected to be outputs; these bits are undefined if they are selected to be inputs. SC1 and SC2 can also be used as general-purpose parallel I/O.

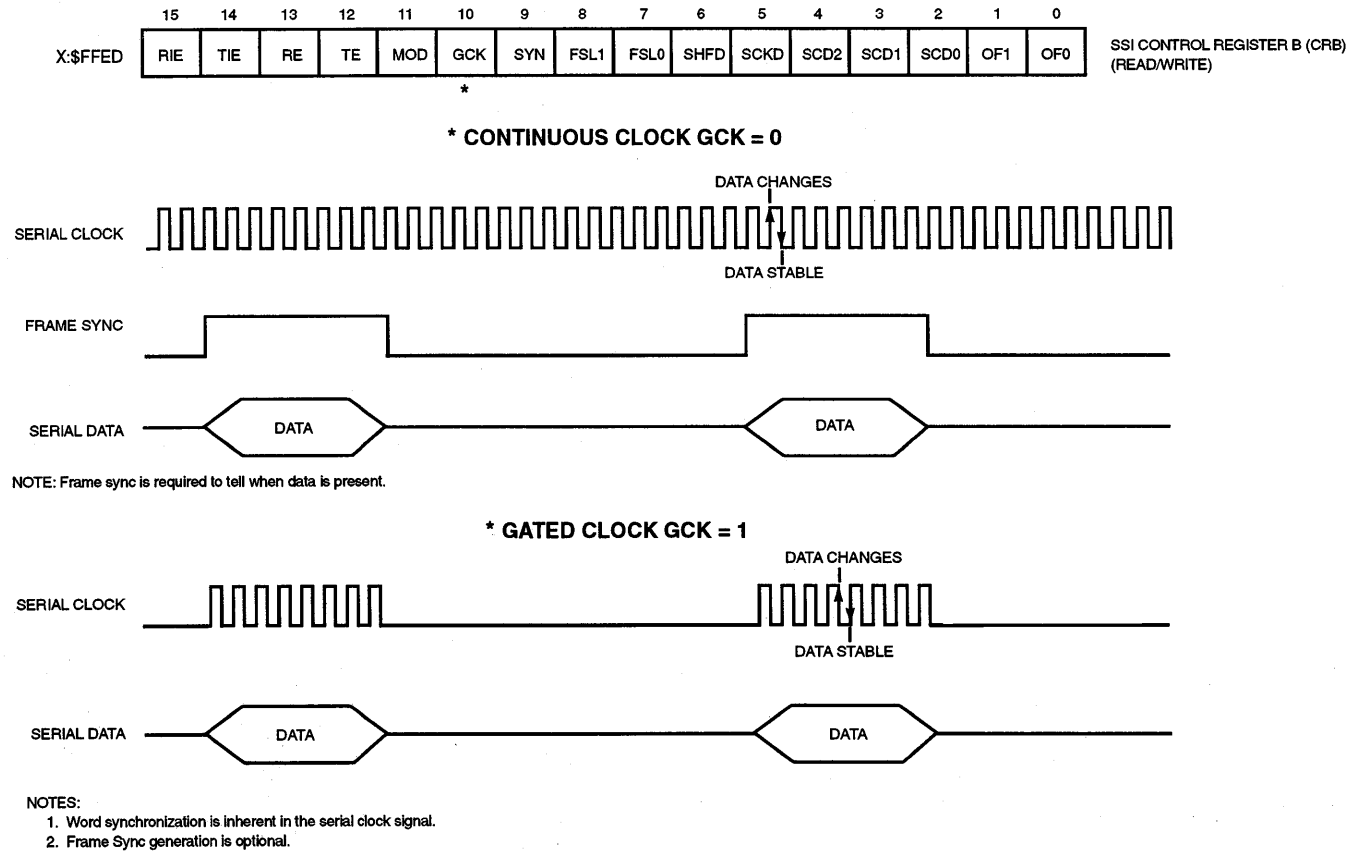


Figure 12-51. CRB GCK Bit Operation

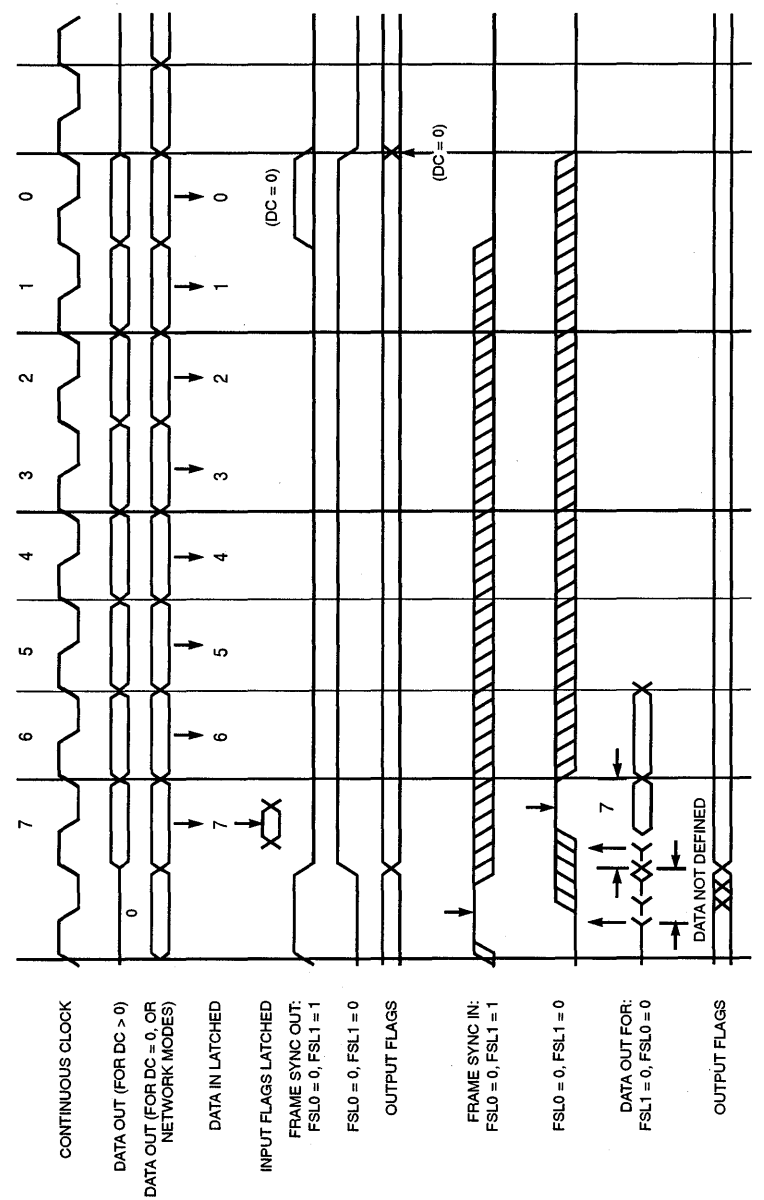


Figure 12-52. Continuous Clock Timing Diagram (8-Bit Example)

12

Figure 12-56 (c) shows a continuous clock (from either an external source or the internal clock), which means that frame sync must be a separate signal. SC2 is used for frame sync, which can come from an internal or external source. Since both the transmitter and receiver use the same clock (synchronous configuration), both use the SCK pin. SC0 and SC1 are designated as flags or can be used as general-purpose parallel I/O.

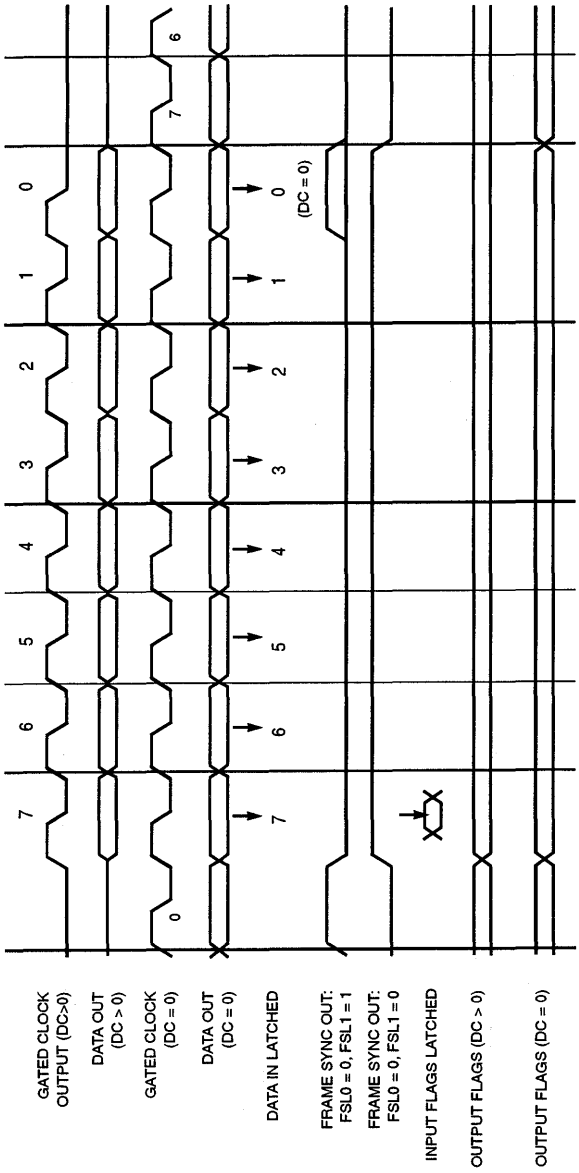
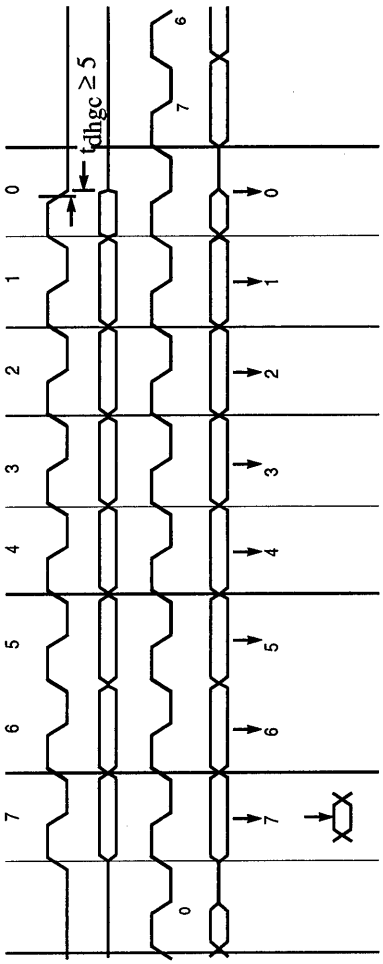


Figure 12-53. Internally Generated Clock Timing (8-Bit Example)

12

Figure 12-56 (d) shows a continuous clock (from either an external source or the internal clock), which means that frame sync must be a separate signal. SC1 is used for the receive frame sync, and SC2 is used for the transmit frame sync. Either frame sync can come from an internal or external source. Since the transmitter and receiver use different clocks (asynchronous configuration), SCK is used for the transmit clock, and SC0 is used for the receive clock.



NOTES:
1. Output enabled on rising edge of first clock input.
2. Output disabled on falling edge of last clock pulse.
3. t_{dhgc} is guaranteed by circuit design.
4. Frame syncs (in or out) are not defined for external gated clock mode.

Figure 12-54. Externally Generated Gated Clock Timing (8-Bit)

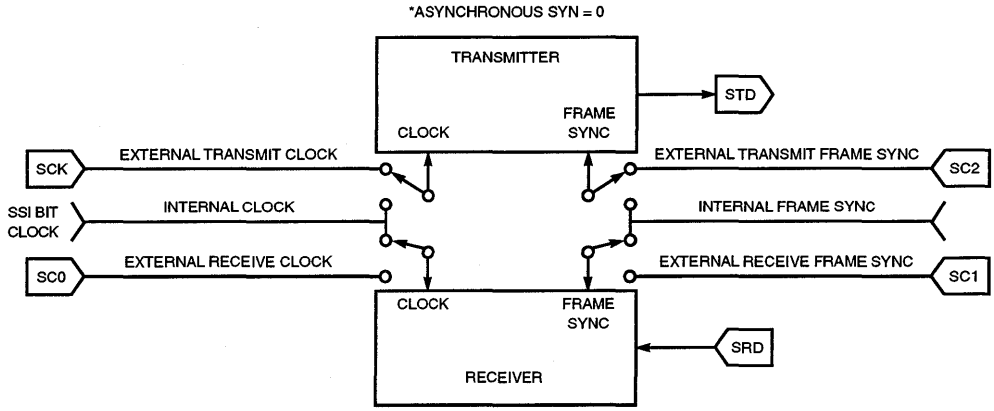
12

12.4.7.1.4 Frame Sync Selection

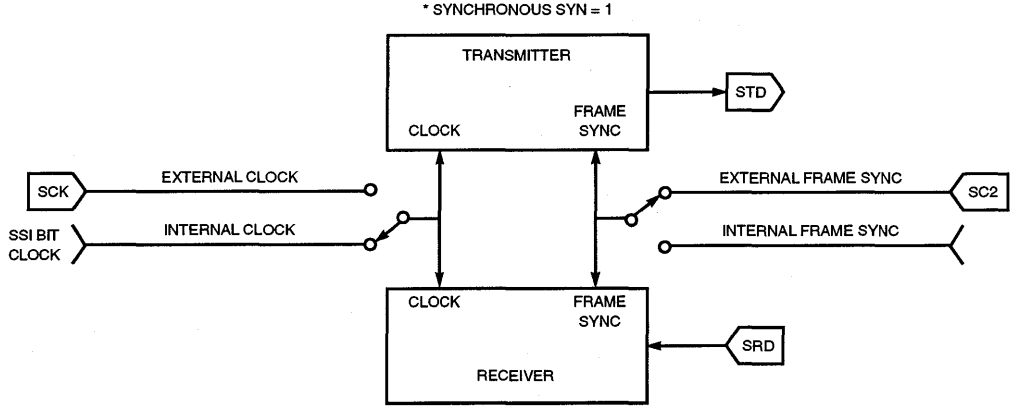
The transmitter and receiver can operate totally independent of each other. The transmitter can have either a bit-long or word-long frame-sync signal format, and the receiver can have the same or opposite format. The selection is made by programming FSL0 and FSL1 in the CRB as shown in Figure 12-57

SSI CONTROL REGISTER B (CRB)
(READ/WRITE)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X:$F	RIE	TIE	RE	TE	MOD	GCK	SYN*	FSL1	FSL0	SHFD	SCKD	SCD2	SCD1	SCD0	OF1	OF0



NOTE: Transmitter and receiver may have different clocks and frame syncs.



NOTE: Transmitter and receiver may have the same clock frame syncs.

Figure 12-55. CRB SYN Bit Operation

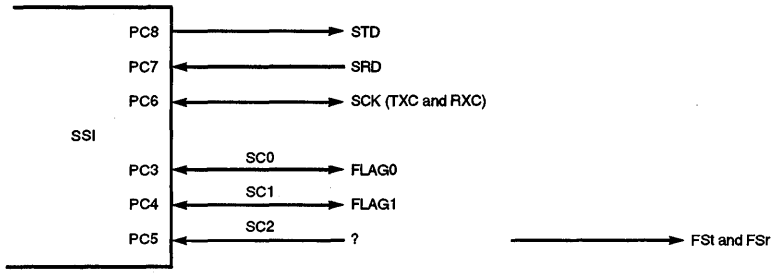


Figure 12-56 (a). Gated Clock — Synchronous Operation

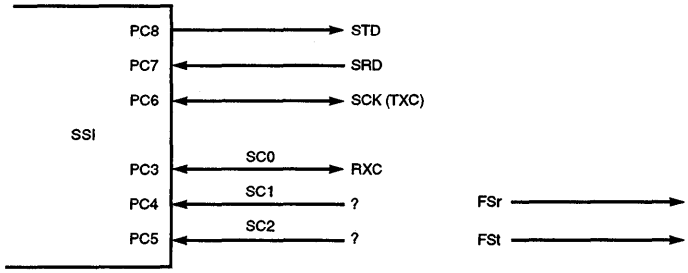


Figure 12-56 (b). Gated Clock — Asynchronous Operation

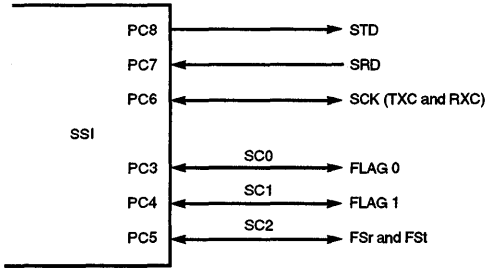


Figure 12-56 (c). Continuous Clock — Synchronous Operation

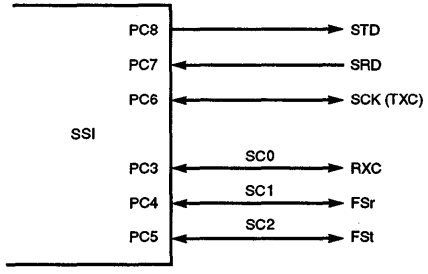
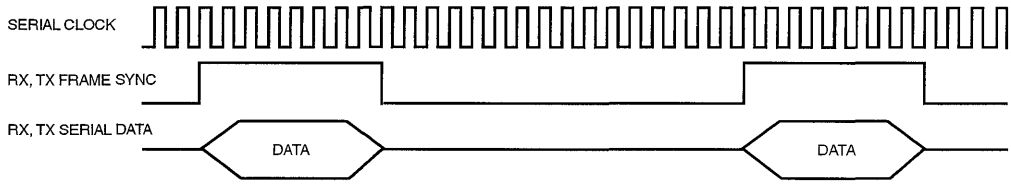


Figure 12-56 (d). Continuous Clock — Asynchronous Operation

SSI CONTROL REGISTER B (CRB)
(READ/WRITE)

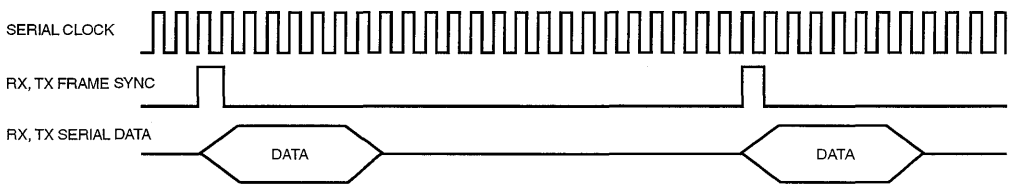
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X:$FFED	RIE	TIE	RE	TE	MOD	GCK	SYN	FSL1	FSL0	SHFD	SCKD	SCD2	SCD1	SCD0	OF1	OF0
								*	*							

*** WORD LENGTH: FSL1 = 0, FSL0 = 0**



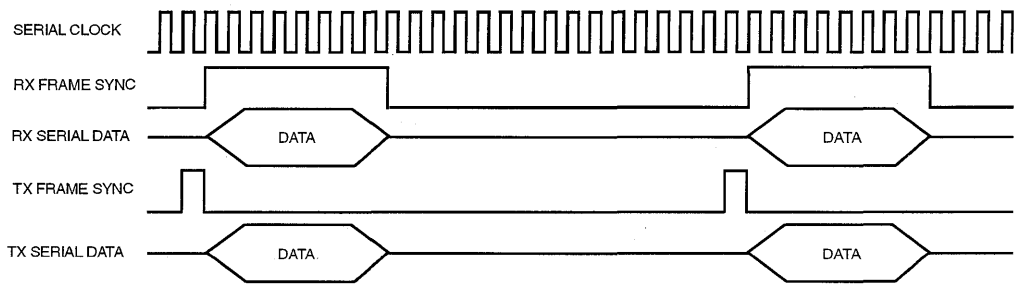
NOTE: Frame sync occurs while data is valid.

*** ONE BIT: FSL1 = 1, FSL0 = 0**



NOTE: Frame sync occurs for one bit time preceding the data.

*** MIXED FRAME LENGTH: FSL1 = 0, FSL0 = 1**



*** MIXED FRAME LENGTH: FSL1 = 1, FSL0 = 1**

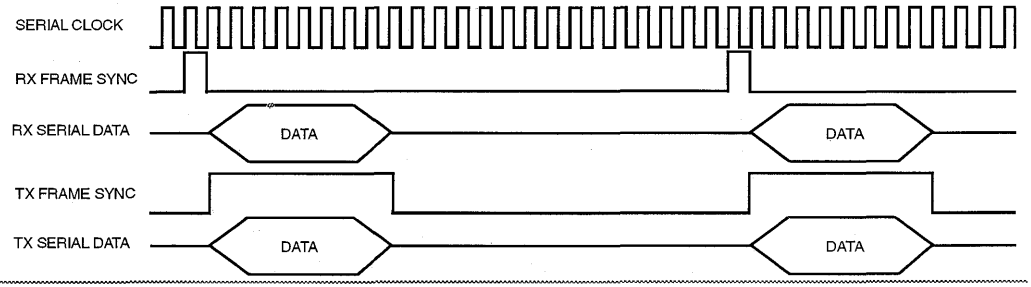


Figure 12-57. CRB FSL0 and FSL1 Bit Operation

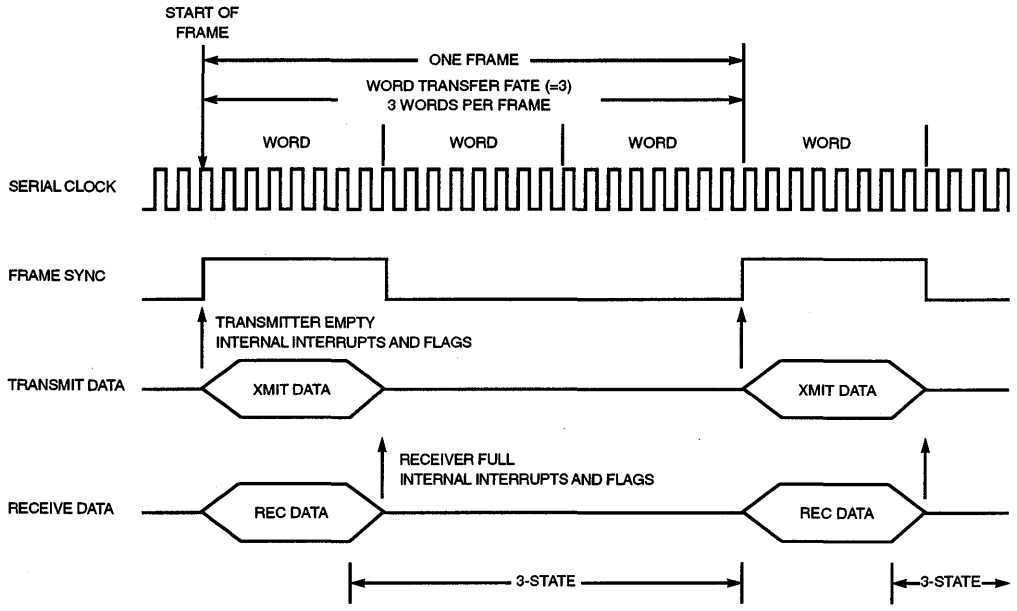


Figure 12-58. Synchronous Communication

1. If FSL1 equals zero (see Figure 12-59), the RX frame sync is asserted during the entire data transfer period. This frame sync length is compatible with Motorola codecs, SPI serial peripherals, serial A/D and D/A converters, shift registers, and telecommunication PCM serial I/O.
2. If FSL1 equals one (see Figure 12-60), the RX frame sync pulses active for one bit clock immediately before the data transfer period. This frame sync length is compatible with Intel and National components, codecs, and telecommunication PCM serial I/O.

The ability to mix frame sync lengths is useful in configuring systems in which data is received from one type device (e.g., codec) and transmitted to a different type device.

FSL0 controls whether RX and TX have the same frame sync length (see Section Figure 12-57. CRB FSL0 and FSL1 Bit Operation). If FSL0 equals zero, RX and TX have the same frame sync length, which is selected by FSL1. If FSL0 equals one, RX and TX have different frame sync lengths, which are selected by FSL1.

The SSI receiver looks for a receive frame sync leading edge only when the previous frame is completed. If the frame sync goes high before the frame is completed (or before the last bit of the frame is received in the case of a bit frame sync), the current frame sync will not be recognized, and the receiver will be internally disabled until the next frame sync. Frames do not have to be adjacent – i.e., a new frame sync does not have to immediately follow the previous frame. Gaps of arbitrary periods can occur between frames. The transmitter will be three-stated during these gaps.

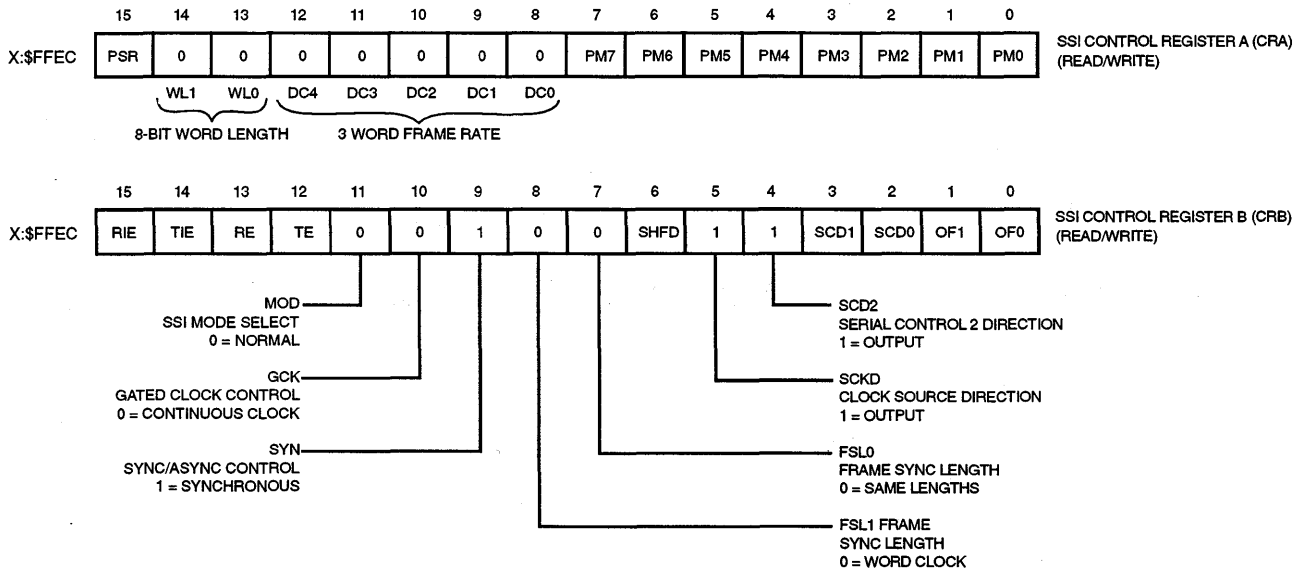


Figure 12-59. Normal Mode Initialization for FLS1=0 and FSL0=0

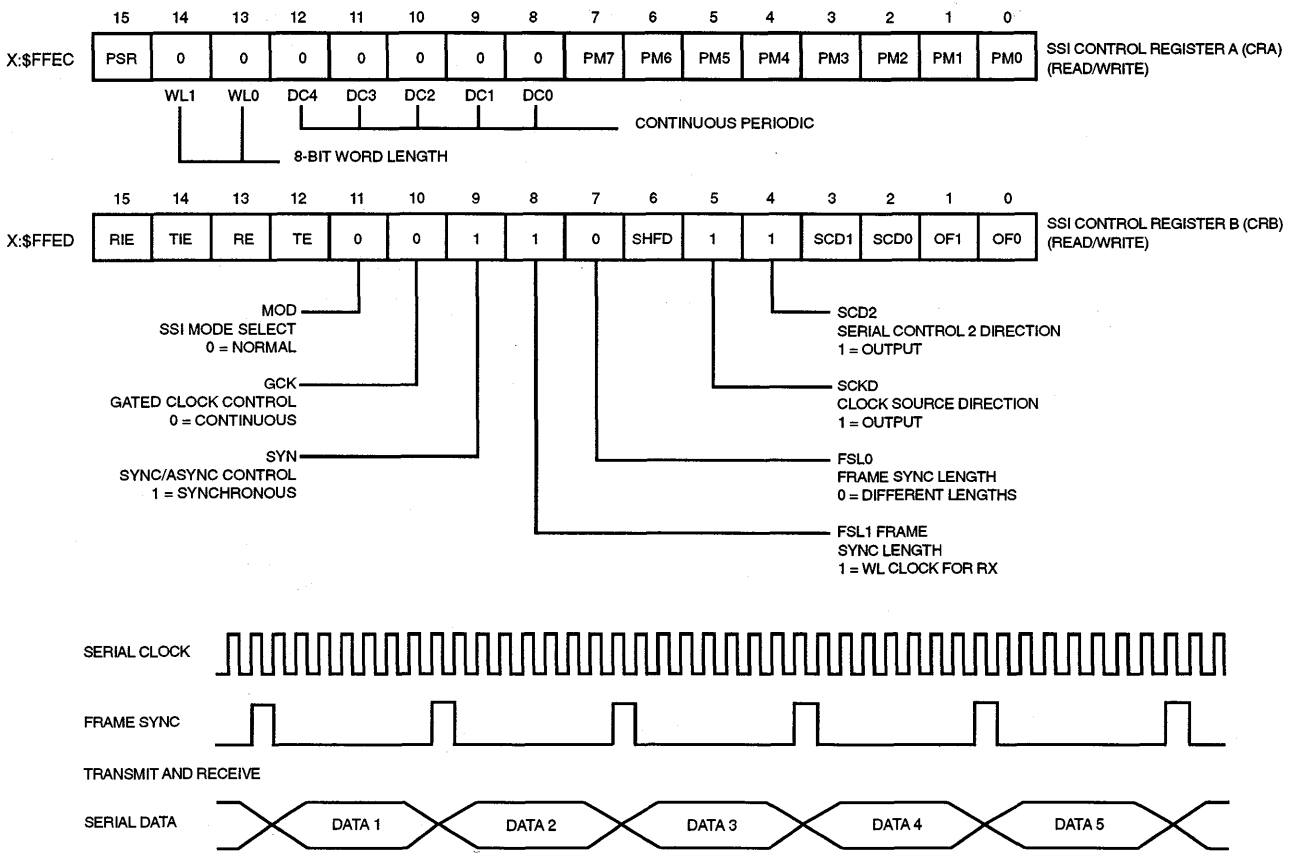


Figure 12-60. Normal Mode Initialization for FSL1=1 and FSL0=0

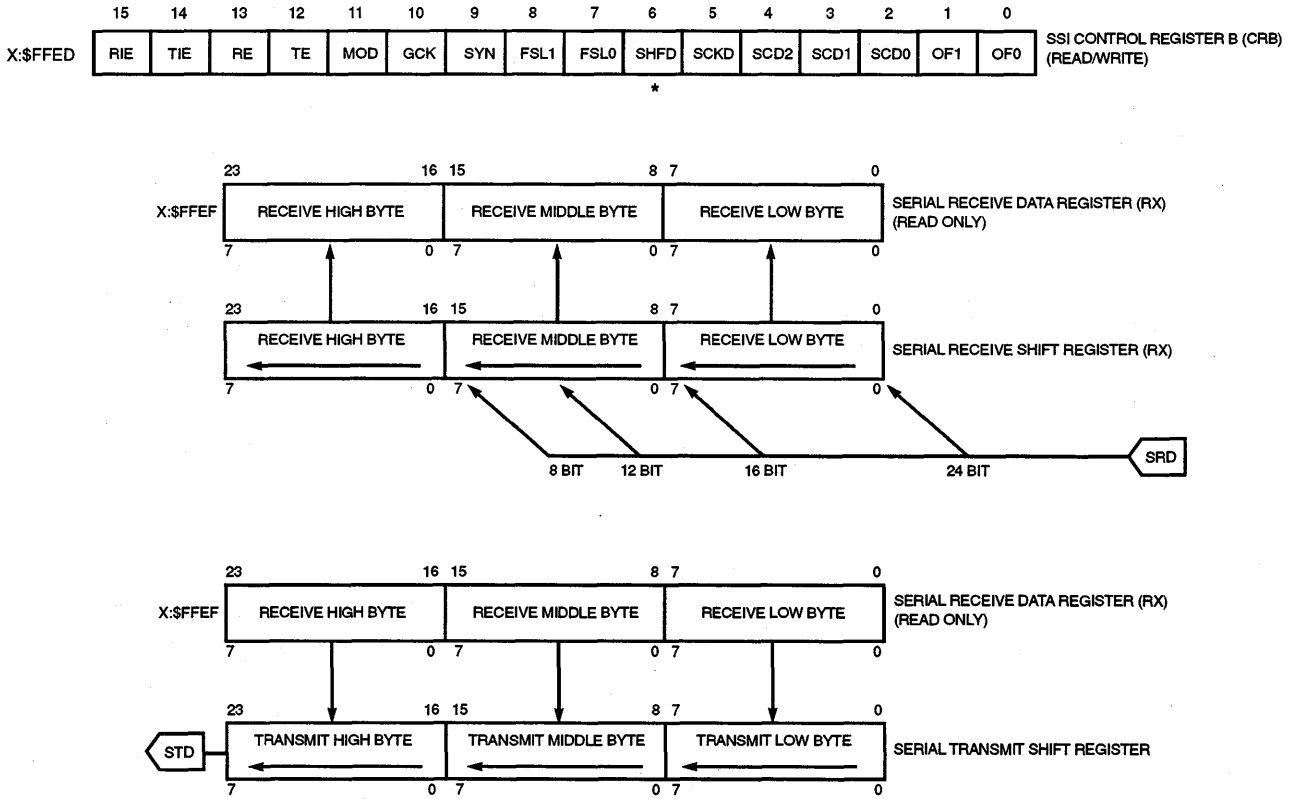


Figure 12-61. CRB SHFD Bit Operation (SHFD = 0)

12.4.7.1.5 Shift Direction Selection

Some data formats, such as those used by codecs, specify MSB first. Other data formats, such as the AES-EBU digital audio, specify LSB first. To interface with devices from both systems, the shift registers in the SSI are bidirectional. The MSB/LSB selection is made by programming SHFD in the CRB.

Figure 12-61 illustrates the operation of the SHFD bit in the CRB. If SHFD equals zero (see Figure 12-61) data is shifted into the receive shift register MSB first and shifted out of the transmit shift register MSB first. If SHFD equals one (see Figure 12-62) data is shifted into the receive shift register LSB first and shifted out of the transmit shift register LSB first.

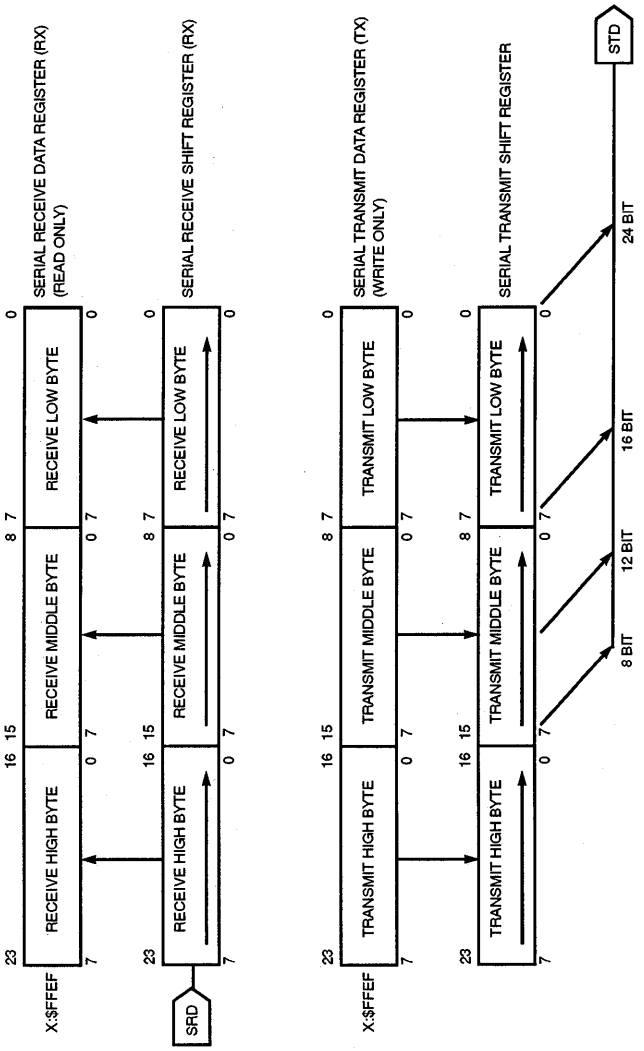


Figure 12-62. CRB SHFD Bit Operation (SHFD=1)

12.4.7.2 NORMAL MODE EXAMPLES

The normal SSI operating mode characteristically has one time slot per serial frame, and data is transferred every frame sync. When the SSI is not in the normal mode, it is in the network mode. The MSB is transmitted first (SHFD=0), with overrun and underrun errors detected by the SSI hardware. Transmit flags are set when data is transferred from the transmit data register to the transmit shift register. The receive flags are set when data is transferred from the receive shift register to the receive data register.

Figure 12-63 shows an example of using the SSI to connect an MC15500 codec with a DSP56002. No glue logic is needed. The serial clock, which is generated internally by the DSP, provides the transmit and receive clocks (synchronous operation) for the codec. SC2 provides all the necessary handshaking. Data transfer begins when the frame sync is asserted. Transmit data is clocked out and receive data is clocked in with the serial clock while the frame sync is asserted (word-length frame sync). At the end of the data transfer, DSP internal interrupts programmed to transfer data to/from will occur, and the SSISR will be updated.

12.4.7.2.1 Normal Mode Transmit

The conditions for data transmission from the SSI are as follows:

1. Transmitter is Enabled (TE=1)
2. Frame sync (or clock in gated clock mode) is active

When these conditions occur in normal mode, the next data word will be transferred from TX to the transmit shift register, the TDE flag will be set (transmitter empty), and the transmit interrupt will occur if TIE equals one (transmit interrupt enabled). The new data word will be transmitted immediately.

The transmit data output (STD) is three-stated, except during the data transmission period. The optional frame sync output, flag outputs, and clock outputs are not three-stated even if both receiver and transmitter are disabled.

The optional output flags are always updated at the beginning of the frame, regardless of TE. The state of the flag does not change for the entire frame.

Figure 12-65 is an example of transmitting data using the SSI in the normal mode with a continuous clock, a bit-length frame sync, and 16-bit data words. The purpose of the program is to interleave and transmit right and left channels in a compact disk player. Four SSI pins are used:

1. SC0 is used as an output flag to indicate right-channel data (OF0=1) or left-channel data (OF0=0)
2. SC2 is TX and RX frame sync out
3. STD is transmit data out
4. SCK clocks the transmit data out

Equates are set for convenience and readability. Test data is then put in the low X: memory locations. The transmit interrupt vector contains a JSR instruction (which forms a long inter-

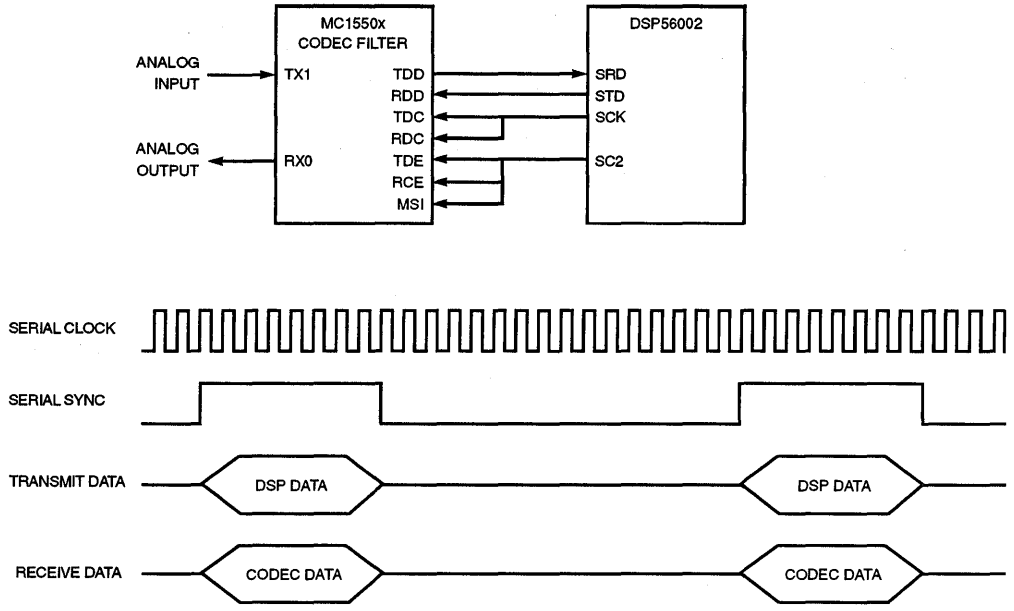


Figure 12-63. Normal Mode Example

rupt). The data pointer and channel flag are initialized before initializing CRA and CRB. It is assumed that the DSP CPU and SSI have been previously reset.

At this point, the SSI is ready to transmit except that the interrupt is masked because the MR was cleared on reset and port C is still configured as general-purpose I/O. Unmasking the interrupt and enabling the SSI pins allows transmission to begin. A "jump to self" instruction causes the DSP to hang and wait for interrupts to transmit the data. When an interrupt occurs, a JSR instruction at the interrupt vector location causes the XMT routine to be executed. Data is then moved to the TX register, and the data pointer is incremented. The flag is tested by the JSET instruction and, if it is set, a jump to left occurs, and the code for the left channel is executed. If the flag is not set, the code for the right channel is executed. In either case, the channel flag in X0 and then the output flag are set to reflect the channel being transmitted. Control is then returned to the main program, which will wait for the next interrupt.

```
;************************************************
;       SSI and other I/O EQUATES*
;************************************************
IPR     EQU$FFFF
CRA     EQU$FFEC
CRB     EQU$FFED
PCC     EQU$FFE1
TX      EQU$FFEF
FLG     EQU$0010
        ORGX:0
        DC      $AAAA00          ;Data to transmit.
        DC      $333300
        DC      $CCCC00
        DC      $F0F000
;************************************************
;       INTERRUPT VECTOR*
;************************************************
        ORG     P:$0010
        JSR     XMT
;************************************************
;       MAIN PROGRAM*
;************************************************
        ORG     P:$40
        MOVE    #0,R0            ;Pointer to data buffer.
        MOVE    #3,M0            ;Set modulus to 4.
        MOVE    #0,X0            ;Initialize channel flag for SSI flag.
        MOVE    X0,X:FLG         ;Start with right channel first.
;************************************************
;       Initialize SSI Port*
;************************************************
        MOVEP   #$3000,X:IPR     ;Set interrupt priority register for SSI.
        MOVEP   #$401F,X:CRA     ;Set continuous clock=5.12/32 MHz
                                 ;word length=16.
        MOVEP   #$5334,X:CRB     ;Enable TIE and TE; make clock and
                                 ;frame sync outputs; frame
                                 ;sync=bit mode; synchronous mode;
                                 ;make SC0 an output.
```

Figure 12-64. Normal Mode Transmit Example (Sheet 1 of 2)

```
;*************************************************
;       Init SSI Interrupt.
;*************************************************
        ANDI    #$FC,MR          ;Unmask interrupts.
        MOVEP   #$01F8,X:PCC     ;Turn on SSI port.
        JMP     *                ;Wait for interrupt.
;*************************************************
;       MAIN INTERRUPT ROUTINE.
;*************************************************
XMT     MOVEP   X:(R0);p1,X:TX   ;Move data to TX register.
        JSET    #0,X:FLG,LEFT    ;Check channel flag.
RIGHT   BCLR    #0,X:CRB         ;Clear SC0 indicating right channel data
        MOVE    #>$01,X0         ;Set channel flag to 1 for next data.
        MOVE    X0,X:FLG
        RTI
LEFT    BSET    #0,X:CRB         ;Set SC0 indicating left channel data.
        MOVE    #>$00,X0         ;Clear channel flag for next data.
        MOVE    X0,X:FLG
        RTI
        END
```

Figure 12-65. Normal Mode Transmit Example (Sheet 2 of 2)

12.4.7.2.2 Normal Mode Receive

If the receiver is enabled, a data word will be clocked in each time the frame sync signal is generated (internal) or detected (external). After receiving the data word, it will be transferred from the SSI receive shift register to the receive data register (RX), RDF will be set (receiver full), and the receive interrupt will occur if it is enabled (RIE=1).

The DSP program has to read the data from RX before a new data word is transferred from the receive shift register; otherwise, the receiver overrun error will be set (ROE=1).

12

Figure 12-67 illustrates the program that receives the data transmitted by the program shown in Figure 12-65. Using the flag to identify the channel, the receive program receives the right- and left-channel data and separates the data into a right data buffer and a left data buffer. The program shown in Figure 12-67 begins by setting equates and then using a JSR instruction at the receive interrupt vector location to form a long interrupt. The main program starts by initializing pointers to the right and left data buffers. The IPR, CRA, and CRB are then initialized. The clock divider bits in the CRA do not have to be set since an external receive clock is specified (SCKD=0). Pin SC0 is specified as an input flag (SYN=1, SCD0=0); pin SC2 is specified as TX and RX frame sync (SYN=1, SCD2=0). The SSI port is then enabled and interrupts are unmasked, which allows the SSI port to begin data reception. A jump-to-self instruction is then used to hang the processor and allow interrupts to receive the data. Normally, the processor would execute useful instructions while waiting for the receive interrupts. When an interrupt occurs, the JSR instruction at the interrupt vector location transfers control to the RCV subroutine. The input flag is tested, and data is put in the left or right data buffer depending on the results of the test. The RTI instruction then returns control to the main program, which will wait for the next interrupt.

```
;************************************************
;       SSI and other I/O EQUATES*
;************************************************
IPR     EQU$FFFF
SSISR   EQU$FFEE
CRA     EQU$FFEC
CRB     EQU$FFED
PCC     EQU$FFE1
RX      EQU$FFEF
FLG     EQU$0010
;************************************************
;       INTERRUPT VECTOR*
;************************************************
        ORGP:$000C
        JSRRCV
;************************************************
;       MAIN PROGRAM*
;************************************************
        ORGP:$40
        MOVE#0,R0;Pointer to memory buffer for
        MOVE#$08,R1;received data. Note data will be
        MOVE#1,M0;split between two buffers which are
        MOVE#1,M1;modulus 2.
```

Figure 12-66. Normal Mode Receive Example (Sheet 1 of 2)

```
;************************************************
;       Initialize SSI Port*
;************************************************
        MOVEP#$3000,X:IPR;Set interrupt priority register for SSI.
        MOVEP#$4000,X:CRA;Set word length = 16 bits.
        MOVEP#$A300,X:CRB;Enable RIE and RE; synchronous
                ;mode with bit frame sync;
                ;clock and frame sync are
                ;external; SC0 is an output.
;************************************************
;       Init SSI Interrupt*
;************************************************
        ANDI#$FC,MR;Unmask interrupts.
        MOVEP#$01F8,X:PCC;Turn on SSI port.
        JMP*    ;Wait for interrupt.
;************************************************
;       MAIN INTERRUPT ROUTINE*
;************************************************
RCV     JSET#0,X:SSISR, RIGHT;Test SCO flag.
LEFT    MOVEPX:RX,X:(R0)+;If SCO clear, receive data
        RTI     ;into left buffer (R0).
RIGHT   MOVEPX:RX,X:(R1)+;If SCO set, receive data
        RTI     ;into right buffer (R1).
        END
```

Figure 12-67. Normal Mode Receive Example (Sheet 2 of 2)

12.4.7.3 NETWORK MODE EXAMPLES

The network mode, the typical mode in which the DSP would interface to a TDM codec network or a network of DSPs, is compatible with Bell and CCITT PCM data/operation formats.

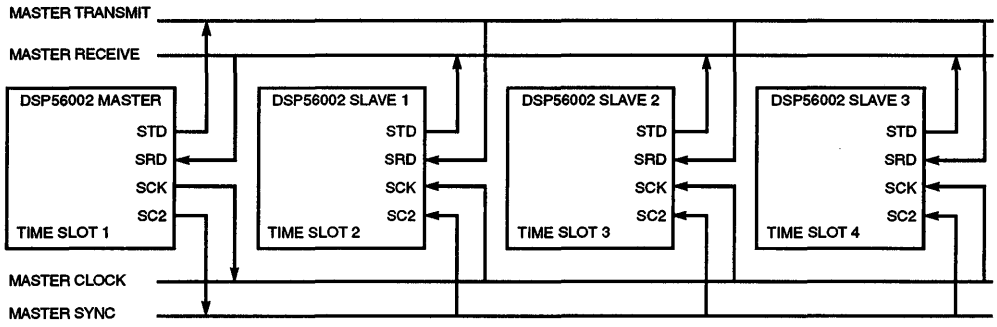


Figure 12-68. Network Mode Example

The DSP may be a master device (see Figure 12-68) that controls its own private network or a slave device that is connected to an existing TDM network, occupying one or more time slots. The key characteristic of the network mode is that each time slot (data word time) is identified by an interrupt or by polling status bits, which allows the option of ignoring the time slot or transmitting data during the time slot. The receiver operates in the same manner except that data is always being shifted into the receive shift register and transferred to the RX. The DSP reads the receive data register and uses or discards the contents. Overrun and underrun errors are detected.

The frame sync signal indicates the beginning of a new data frame. Each data frame is divided into time slots; transmission or reception can occur in each time slot (rather than in just the frame sync time slot as in normal mode). The frame rate dividers (controlled by DC4, DC3, DC2, DC1, and DC0) control the number of time slots per frame from 2 to 32. Time-slot assignment is totally under software control. Devices can transmit on multiple time slots, receive multiple time slots, and the time-slot assignment can be changed dynamically.

12

A simplified flowchart showing operation of the network mode is shown in Figure 12-69. Two counters are used to track the current transmit and receive time slots. Slot counter one (SLOTCT1) is used to track the transmit time slot; slot counter two (SLOTCT2) is used for receive. When the transmitter is empty, it generates an interrupt; a test is then made to see if it is the beginning of a frame. If it is the beginning of a frame, SLOTCT1 is cleared to start counting the time slots. If it is not the beginning of a frame, SLOTCT1 is incremented. The next test checks to see if the SSI should transmit during this time slot. If it is time to transmit, data is written to the TX; otherwise, dummy data is written to the TSR, which prevents a transmit underrun error from occurring and three-states the STD pin. The DSP can then return to what it was doing before the interrupt and wait for the next interrupt to occur. SLOTCT1 should reflect the data in the shift registers to coincide with TFS. Software must recognize that the data being written to TX will be transmitted in time slot SLOTCT1 plus one.

The receiver operates in a similar manner. When the receiver is full, an interrupt is generated, and a test is made to see if this is the beginning of a frame. If it is the beginning of a frame, SLOTCT2 is cleared to start counting the time slots. If it is not the beginning of a frame, SLOTCT2 is incremented. The next test checks to see if the data received is intended for this DSP. If the current time slot is the one assigned to the DSP receiver, the data is kept;

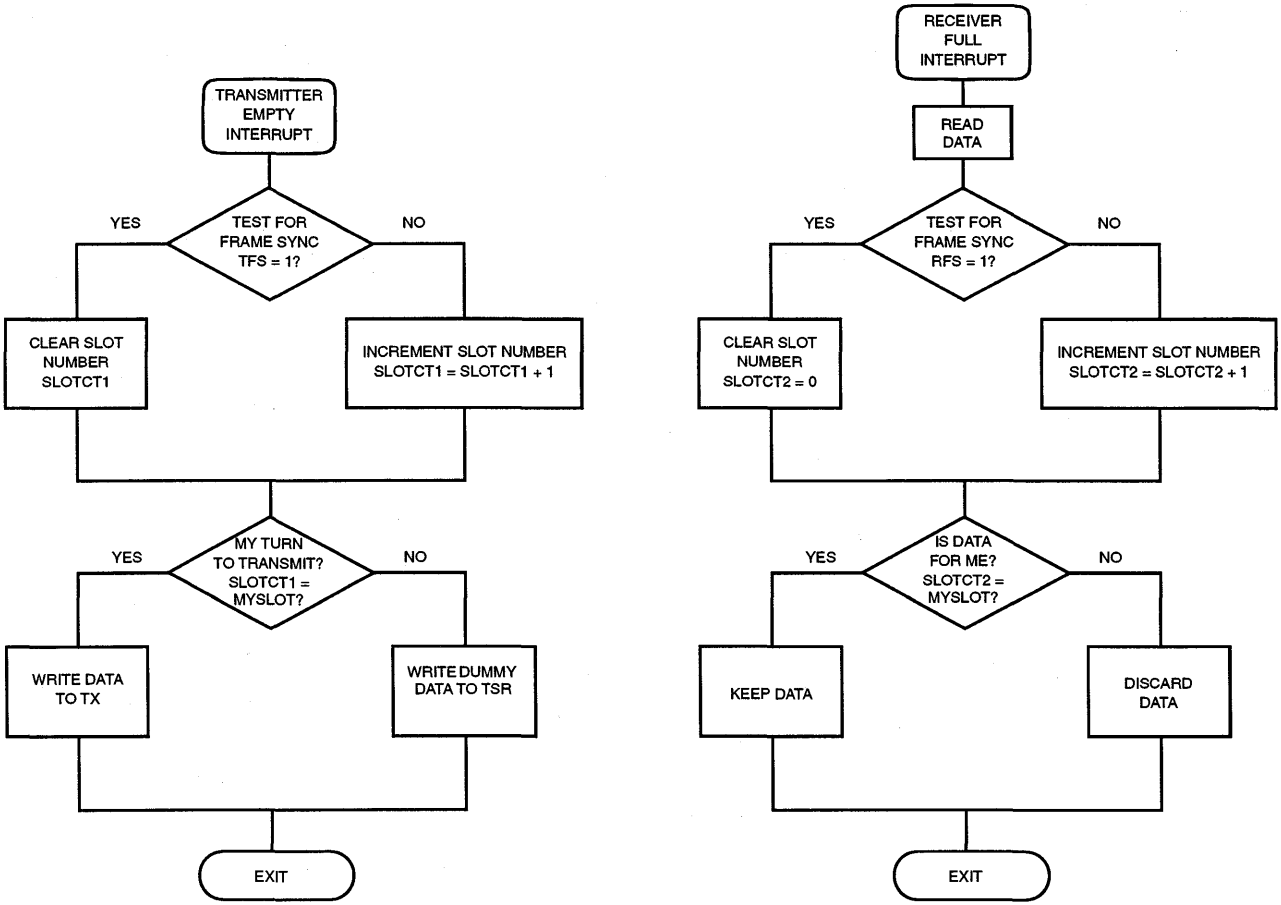


Figure 12-69. TDM Network Software Flowchart

otherwise, the data is discarded, and the DSP can then return to what it was doing before the interrupt. SLOTCT2 should reflect the data in the receive shift register to coincide with the RFS flag. Software must recognize that the data being read from RX is for time slot SLOTCT2 minus two.

Initializing the network mode is accomplished by setting the bits in CRA and CRB as follows (see Figure 12-70):

1. The word length must be selected by setting WL1 and WL0. In this example, an 8-bit word length was chosen (WL1=0 and WL0=0).
2. The number of time slots is selected by setting DC4–DC0. Four time slots were chosen for this example (DC4–DC0=$03).
3. The serial clock rate must be selected by setting PSR and PM7–PM0 (see Table 12-17 and Table 12-18).
4. RE and TE must be set to activate the transmitter and receiver. If interrupts are to be used, RIE and TIE should be set. RIE and TIE are usually set after everything else is configured and the DSP is ready to receive interrupts.
5. The network mode must be selected (MOD=1).
6. A continuous clock is selected in this example by setting GCK=0.
7. Although it is not required for the network mode, synchronous clock control was selected (SYN=1).
8. The frame sync length was chosen in this example as word length (FSL1=0) for both transmit and receive frame sync (FSL0=0). Any other combinations could have been selected, depending on the application.
9. Control bits SHFD, SCKD, SCD2, SCD1, SCD0, and the flag bits (OF1 and OF0) should be set as needed for the application.

12

12.4.7.3.1 Network Mode Transmit

When TE is set, the transmitter will be enabled only after detection of a new data frame sync. This procedure allows the SSI to synchronize to the network timing.

Normal startup sequence for transmission in the first time slot is to write the data to be transmitted to TX, which clears the TDE flag. Then set TE and TIE to enable the transmitter on the next frame sync and to enable transmit interrupts.

Alternatively, the DSP programmer may decide not to transmit in the first time slot by writing any data to the time slot register (TSR). This will clear the TDE flag just as if data were going to be transmitted, but the STD pin will remain in the high-impedance state for the first time slot. The programmer then sets TE and TIE.

When the frame sync is detected (or generated), the first data word will be transferred from TX to the transmit shift register and will be shifted out (transmitted). TX being empty will cause TDE to be set, which will cause a transmitter interrupt. Software can poll TDE or use interrupts to reload the TX register with new data for the next time slot. Software can also write to TSR to prevent transmitting in the next time slot. Failing to reload TX (or writing to

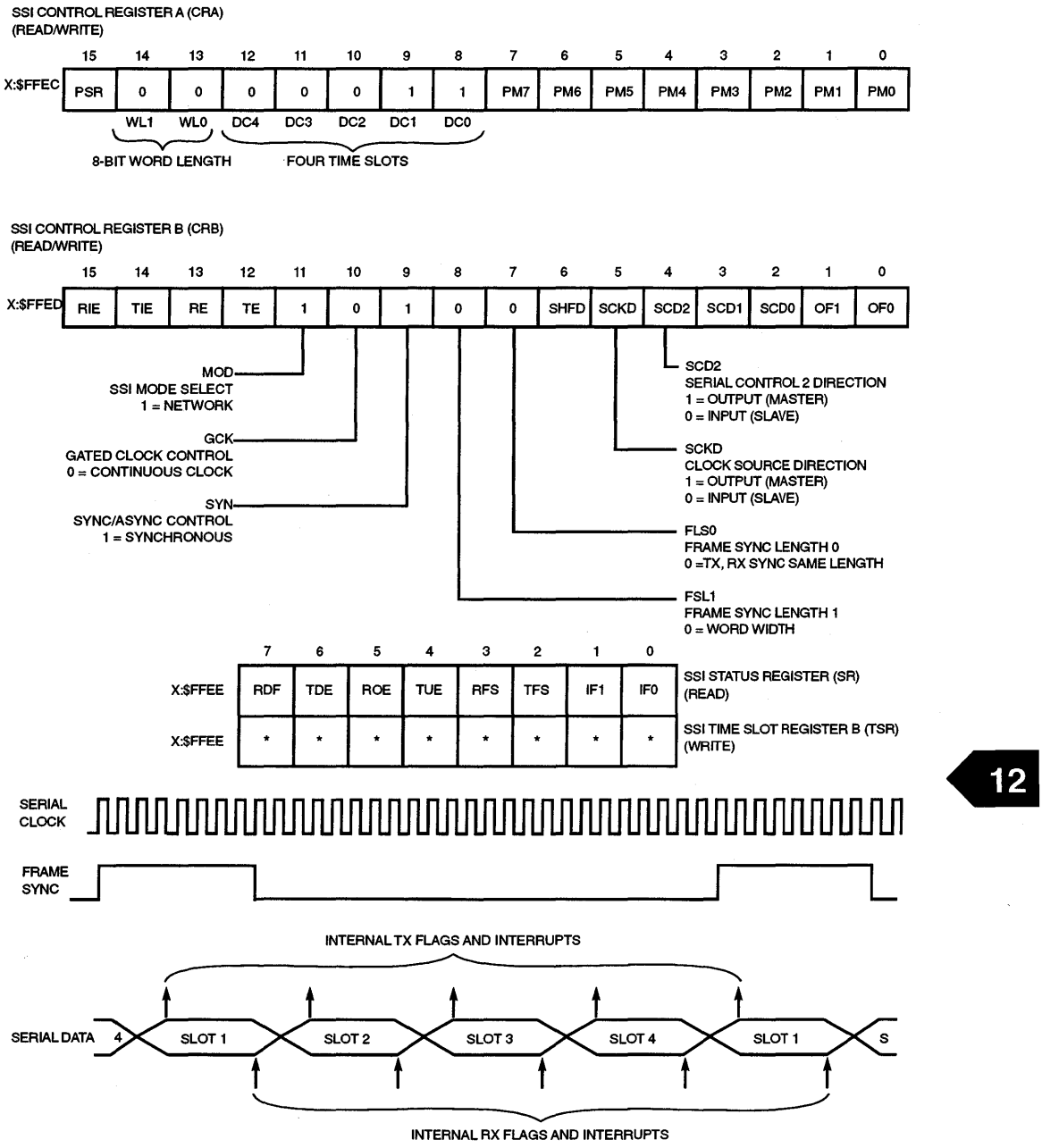


Figure 12-70. Network Mode Initialization

the TSR) before the transmit shift register is finished shifting (empty) will cause a transmitter underrun. The TUE error bit will be set, causing the previous data to be retransmitted.

The operation of clearing TE and setting it again will disable the transmitter after completion of transmission of the current data word until the beginning of the next frame sync period. During that time, the STD pin will be three-stated. When it is time to disable the transmitter, TE should be cleared after TDE is set to ensure that all pending data is transmitted.

The optional output flags are updated every time slot regardless of TE.

To summarize, the network mode transmitter generates interrupts every time slot and requires the DSP program to respond to each time slot. These responses can be:

1. Write data register with data to enable transmission in the next time slot
2. Write the time slot register to disable transmission in the next time slot
3. Do nothing – transmit underrun will occur the at beginning of the next time slot, and the previous data will be transmitted

Figure 12-71 differs from the program shown in Figure 12-65 only in that it uses the network mode to transmit only right-channel data. A time slot is assigned for the left-channel data, which could be inserted by another DSP using the network mode. In the "Initialize SSI Port" section of the program, two words per frame are selected using CRA, and the network mode is selected by setting MOD to one in the CRB. The main interrupt routine, which waits to move the data to TX, only transmits data if the current time slot is for the right channel. If the current time slot is for the left channel, the TSR is written, which three-states the output to allow another DSP to transmit the left channel during the time slot.

```
;************************************************
;       SSI and other I/O EQUATES*
;************************************************
IPR     EQU     $FFFF
CRA     EQU     $FFEC
CRB     EQU     $FFED
PCC     EQU     $FFE1
TX      EQU     $FFEF
TSR     EQU     $FFEE
FLG     EQU     $0010
        ORG     X:0
        DC      $AAAA00          ;Data to transmit
        DC      $333300
        DC      $CCCC00
        DC      $F0F000
;************************************************
;       INTERRUPT VECTOR*
;************************************************
        ORG     P:$0010
        JSR     XMT
;************************************************
;       MAIN PROGRAM*
;************************************************
        ORG     P:$40
        MOVE    #0,R0            ;Pointer to data buffer
        MOVE    #3,M0            ;Set modulus to 4
        MOVE    #0,X0            ;Initialize user flag for SSI flag.
        MOVE    X0,X:FLG         ;Start with the right channel
```

Figure 12-71. Network Mode Transmit Example Program (Sheet 1 of 2)

```
;************************************************
;       Initialize SSI Port*
;************************************************
        MOVEP   #$3000,X:IPR      ;Set interrupt priority register for SSI
        MOVEP   #$411F,X:CRA      ;Set continuous clock=5.12/32 MHz
                                  ;word length=16
        MOVEP   #$5B34,X:CRB      ;Enable TIE and TE; make clock and
                                  ;frame sync outputs; frame
                                  ;sync=bit mode; synchronous mode;
                                  ;make SC0 an output
;************************************************
;       Init SSI Interrupt*
;************************************************
        ANDI    #$FC,MR           ;Unmask interrupts
        MOVEP   #$01F8,X:PCC      ;Turn on SSI port
        JMP     *                 ;Wait for interrupt
;************************************************
;       MAIN INTERRUPT ROUTINE*
;************************************************
XMT
        JSET    #0,X:FLG,LEFT     ;Check user flag
RIGHT   BCLR    #0,X:CRB          ;Clear SC0 indicating right channel data
        MOVEP   X:(R0)+,X:TX      ;Move data to TX register
        MOVE    #>$01,X0          ;Set user flag to 1
        MOVE    X0,X:FLG          ;for next data
        RTI
LEFT    BSET    #0,X:CRB          ;Set SC0 indicating left channel data
        MOVEP   X0,X:TSR          ;Write to TSR register
        MOVE    #>$00,X0          ;Clear user flag
        MOVE    X0,X:FLG          ;for next data
        RTI
        END
```

Figure 12-72. Network Mode Transmit Example Program (Sheet 2 of 2)

```
;************************************************
;       SSI and other I/O EQUATES.
;************************************************
IPR     EQU     $FFFF
SSISR   EQU     $FFEE
CRA     EQU     $FFEC
CRB     EQU     $FFED
PCC     EQU     $FFE1
RX      EQU     $FFEF
;************************************************
;       INTERRUPT VECTOR.
;************************************************
        ORG     P:$000C
        JSR     RCV
;************************************************
;       MAIN PROGRAM.
;************************************************
        ORG     P:$40
        MOVE    #0,R0           ;Pointer to memory buffer for
        MOVE    #$08,R1         ;received data. Note data will be
        MOVE    #3,M0           ;split between two buffers which are
        MOVE    #3,M1           ;modulus 4.
;************************************************
;       Initialize SSI Port.
;************************************************
        MOVEP   #$3000,X:IPR    ;Set interrupt priority register for SSI.
        MOVEP   #$4100,X:CRA    ;Set word length = 16 bits.
        MOVEP   #$AB00,X:CRB    ;Enable RIE and RE; synchronous
                                ;mode with bit frame sync;
                                ;clock and frame sync are
                                ;external; SC0 is an input.
```

Figure 12-73. Network Mode Receive Example Program (Sheet 1 of 2)

```
;************************************************
;       Init SSI Interrupt.
;************************************************
        ANDI    #$FC,MR         ;Unmask interrupts.
        MOVEP   #$01F8,X:PCC    ;Turn on SSI port.
        JMP     *               ;Wait for interrupt.
;************************************************
;       MAIN INTERRUPT ROUTINE.
;************************************************
RCV     JSET    #0,X:SSISR, RIGHT;Test SC0 flag.
LEFT    MOVEP   X:RX,X:(R0)+    ;If SC0 clear, receive data
        RTI                     ;into left buffer (R0).
RIGHT   MOVEP   X:RX,X:(R1)+    ;If SC0 set, receive data
        RTI                     ;into right buffer (R1).
        END
```

Figure 12-74. Network Mode Receive Example Program (Sheet 2 of 2)

12.4.7.3.2 Network Mode Receive

The receive enable will occur only after detection of a new data frame with RE set. The first data word is shifted into the receive shift register and is transferred to the RX, which sets

RDF if a frame sync was received (i.e., this is the start of a new frame). Setting RDF will cause a receive interrupt to occur if the receiver interrupt is enabled (RIE=1).

The second data word (second time slot in the frame) begins shifting in immediately after the transfer of the first data word to the RX. The DSP program has to read the data from RX (which clears RDF) before the second data word is completely received (ready to transfer to RX), or a receive overrun error will occur (ROE=1), and the data in the receiver shift register will not be transferred and will be lost.

If RE is cleared and set again by the DSP program, the receiver will be disabled after receiving the current time slot in progress until the next frame sync (first time slot). This mechanism allows the DSP programmer to ignore data in the last portion of a data frame.

NOTE

The optional frame sync output and clock output signals are not affected, even if the transmitter and/or receiver are disabled. TE and RE do not disable bit clock and frame sync generation.

To summarize, the network mode receiver receives every time slot data word unless the receiver is disabled. An interrupt can occur after the reception of each data word, or the programmer can poll RDF. The DSP program response can be

1. Read RX and use the data
2. Read RX and ignore the data
3. Do nothing – the receiver overrun exception will occur at the end of the current time slot
4. Toggle RE to disable the receiver until the next frame, and read RX to clear RDF

Figure 12-73 is essentially the same program shown in Figure 12-66 except that this program uses the network mode to receive only right-channel data. In the "Initialize SSI Port" section of the program, two words per frame are selected using the DC bits in the CRA, and the network mode is selected by setting MOD to one in the CRB. If the program in Figure 12-71 is used to transmit to the program in Figure 12-73, the correct data will appear in the data buffer for the right channel, but the buffer for the left channel will probably contain $000000 or $FFFFFF, depending on whether the transmitter output was high or low when TSR was written and whether the output was three-stated.

12.4.7.4 ON-DEMAND MODE EXAMPLES

A divide ratio of one (DC=00000) in the network mode is defined as the on-demand mode of the SSI because it is the only data-driven mode of the SSI – i.e., data is transferred whenever data is present (see Figure 12-75 and Figure 12-76). STD and SCK from DSP1 are connected to DSP2 – SRD and SC0, respectively. SC0 is used as an input clock pin in this application. Receive data and receive data clock are separate from the transmit signals. On-demand data transfers are nonperiodic, and no time slots are defined. When there is a clock in the gated clock mode, data is transferred. Although they are not necessarily needed, frame sync and flags are generated when data is transferred. Transmitter underruns (TUE) are impossible in this mode and are therefore disabled. In the on-demand transmit mode,

two additional SSI clock cycles are automatically inserted between each data word transmitted. This procedure guarantees that frame sync will be low between every transmitted data word or that the clock will not be continuous between two consecutive words in the gated clock mode. The on-demand mode is similar to the SCI+ shift register mode with SSFTD equals one and SCKP equals one. The receiver should be configured to receive the bit clock and, if continuous clock is used, to receive an external frame sync. Therefore, for all full-duplex communication in on-demand mode, the asynchronous mode should be used. The on-demand mode is SPI compatible.

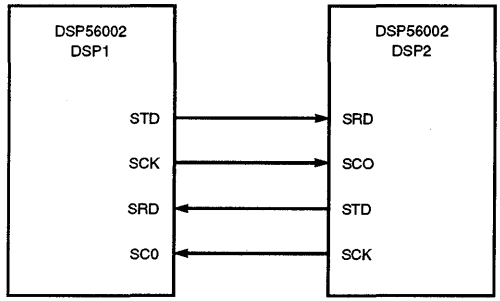
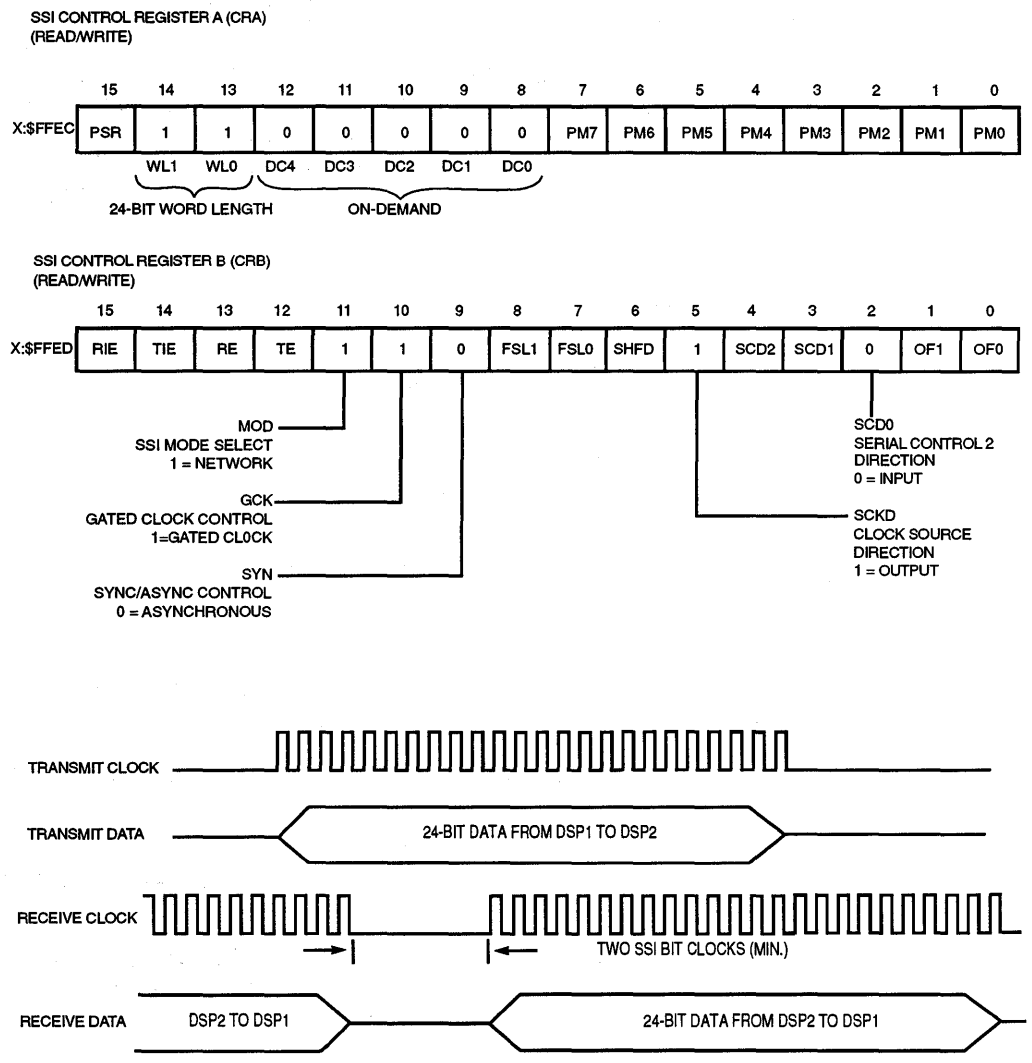


Figure 12-75. On Demand Example

Initializing the on-demand mode for the example illustrated in Figure 12-76 is accomplished by setting the bits in CRA and CRB as follows:

1. The word length must be selected by setting WL1 and WL0. In this example, a 24-bit word length was chosen (WL1=1 and WL0=1).
2. The on-demand mode is selected by clearing DC4–DC0.
3. The serial clock rate must be selected by setting PSR and PM7–PM0 (see Table 12-17 and Table 12-18).
4. RE and TE must be set to activate the transmitter and receiver. If interrupts are to be used, RIE and TIE should be set. RIE and TIE are usually set after everything else is configured and the DSP is ready to receive interrupts.
5. The network mode must be selected (MOD=1).
6. A gated clock (GCK=1) is selected in this example. A continuous clock example is shown in Figure 12-71.
7. Asynchronous clock control was selected (SYN=0) in this example.
8. Since gated clock is used, the frame sync is not necessary. FSL1 and FSL0 can be ignored.
9. SCKD must be an output (SCKD=1).
10. SCD0 must be an input (SCD0=0).
11. Control bit SHFD should be set as needed for the application. Pins SC1 and SC2 are undefined in this mode (see Table 12-15) and should be programmed as general-purpose I/O pins.



NOTE: Two SSI bit clock times are automatically inserted between each data word. This guarantees frame sync will be low between every data word transmitted and the clock will not be continuous for two consecutive data words.

Figure 12-76. On-Demand Data-Driven Network Mode

12.4.7.4.1 On-Demand Mode – Continuous Clock

This special case will not generate a periodic frame sync. A frame sync pulse will be generated only when data is available to transmit (see Figure 12-77(a)). The frame sync signal indicates the first time slot in the frame. The on-demand mode requires that the transmit frame sync be internal (output) and the receive frame sync be external (input). Therefore,

for simplex operation, the synchronous mode could be used; however, for full-duplex operation, the asynchronous mode must be used. Data transmission that is data driven is enabled by writing data into TX. Although the SSI is double buffered, only one word can be written to TX, even if the transmit shift register is empty. The receive and transmit interrupts function as usual using TDE and RDF; however, transmit and receive underruns are impossible for on-demand transmission and are disabled. This mode is useful for interfacing to codecs requiring a continuous clock.

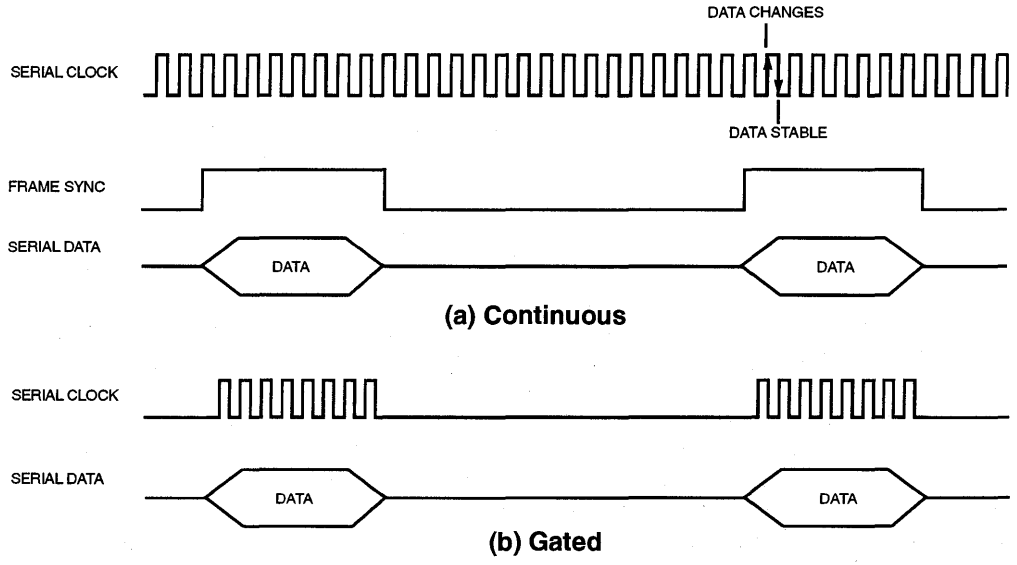


Figure 12-77. Clock Modes

12.4.7.4.2 On-Demand Mode – Gated Clock

Gated clock mode (see Figure 12-77(b)) is defined for on-demand mode, but the gated clock mode is considered a frame sync source; therefore, in gated clock mode, the transmit clock must be internal (output) and the receive clock must be external (input). For on-demand mode, with internal (output) synchronous gated clock, output clock is enabled for the transmitter and receiver when TX data is transferred to the transmit data shift register. This SPI master operating mode is shown in Figure 12-78. Word sync is inherent in the clock signal, and the operation format must provide frame synchronization.

Figure 12-79 is the block diagram for the program presented in Figure 12-81. This program contains a transmit test program that was written as a scoping loop (providing a repetitive sync) using the on-demand, gated, synchronous mode with no interrupts (polling) to transmit data to the program shown in Figure 12-82. The program also demonstrates using GPIO pins as general-purpose control lines. PC3 is used as an external strobe or enable for hardware such as an A/D converter.

The transmit program sets equates for convenience and readability. Test data is then written to X: memory, and the data pointer is initialized. Setting M0 to two makes the buffer circular (modulo 3), which saves the step of resetting the pointer each loop. PC3 is configured as a

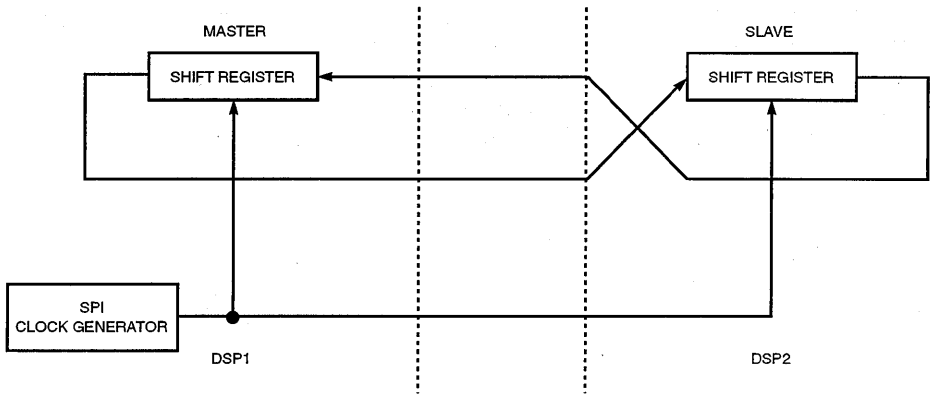


Figure 12-78. SPI Configuration

general-purpose output for use as a scope sync, and CRA and CRB are then initialized. Setting the PCC bits begins SSI operation; however, no data will be transmitted until data is written to TX. PC3 is set high at the beginning of data transmission; data is then moved to TX to begin transmission. A JCLR instruction is then used to form a wait loop until TDE equals one and the SSI is ready for another data word to be transmitted. Two more data words are transmitted in this fashion (this is an arbitrary number chosen for this test loop). An additional wait is included to make sure that the frame sync has gone low before PC3 is cleared, indicating on the scope that transmission is complete. A wait of 100 NOPs is implemented by using the REP instruction before starting the loop again.

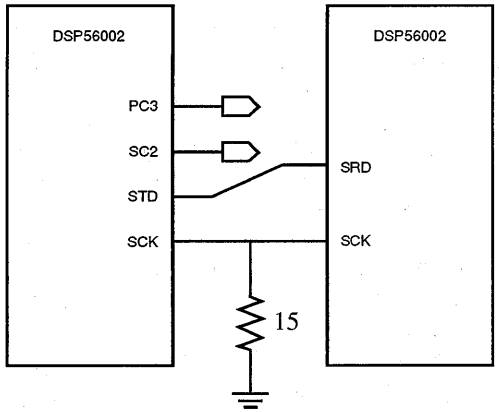


Figure 12-79. On-Demand Mode Example — Hardware Configuration

```
;************************************************
;       SSI and other I/O EQUATES.
;************************************************
CRA     EQU     $FFEC
CRB     EQU     $FFED
PCC     EQU     $FFE1
PCD     EQU     $FFE5
SSISR   EQU     $FFEE
TX      EQU     $FFEF
PCDDR   EQU     $FFE3
        ORG     X:0
        DC      $AA0000          ;Data to transmit.
        DC      $330000
        DC      $F00000
;************************************************
;       MAIN PROGRAM.
;************************************************
        ORG     P:$40
        MOVE    #0,R0            ;Pointer to data buffer
        MOVE    #2,M0            ;Length of buffer is 3
```

Figure 12-80. On-Demand Mode Transmit Example Program (Sheet 1 of 2)

```
        MOVEP   #$08,X:PCDDR     ;SC0 (PC3) as general purpose output.
        MOVEP   #$001F,X:CRA     ;Set Word Length=8, CLK=5.12/32 MHz.
        MOVEP   #$1E30,X:CRB     ;Enable transmitter, Mode=On- Demand,
                                 ;Gated clock on, synchronous mode,
                                 ;Word frame sync selected, frame
                                 ;sync and clock are internal and
                                 ;output to port pins.
        MOVEP   #$1F0,X:PCC      ;Set PCC for SSI and
LOOP0   BSET    #3,X:PCD         ;Set PC3 high (this is example enable
                                 ;or strobe for an external device
                                 :such as an ADC).
        MOVEP   X:(R0);p1,X:TX   ;Move data to TX register
TDE1    JCLR    #6,X:SSISR,TDE1  ;Wait for TDE (transmit data register
                                 ;empty) to go high.
        MOVEP   X:(R0);p1,X:TX   ;Move next data to TX.
TDE2    JCLR    #6,X:SSISR,TDE2  ;Wait for TDE to go high.
        MOVEP   X:(R0);p1,X:TX   ;Move data to TX.
TDE3    JCLR    #6,X:SSISR,TDE3  ;Wait for TDE=1.
FSC     JSET    #5,X:PCD,FSC     ;Wait for frame sync to go low. NOTE:
                                 ;State of frame sync is directly
                                 ;determined by reading PC5.
        BCLR    #3,X:PCD         ;Set PC3 lo (example external enable).
;anything goes here (i.e., any processing)
        REP     #100
        NOP
        JMP     LOOP0            ;Continue sequence forever.
        END
```

Figure 12-81. On-Demand Mode Transmit Example Program (Sheet 2 of 2)

Figure 12-82 is the receive program for the scoping loop program presented in Figure 12-81. The receive program also uses the on-demand, gated, synchronous mode with no interrupts (polling). Initialization for the receiver is slightly different than for the transmitter. In

CRB, RE is set rather than TE, and SCKD and SCD2 are inputs rather than outputs. After initialization, a JCLR instruction is used to wait for a data word to be received (RDF=1). When a word is received, it is put into the circular buffer and loops to wait for another data word. The data in the circular buffer will be overwritten after three words are received (does not matter in this application).

```
;***********************************************
;       SSI and other I/O EQUATES*
;***********************************************
CRA     EQU     $FFEC
CRB     EQU     $FFED
PCC     EQU     $FFE1
PCD     EQU     $FFE5
SSISR   EQU     $FFEE
RX      EQU     $FFEF
PCDDR   EQU     $FFE3
;***********************************************
;       MAIN PROGRAM*
;***********************************************
        ORG     P:$40
        MOVE    #0,R0           ;Pointer to data buffer
        MOVE    #2,M0           ;Length of buffer is 3
        MOVEP   #$001F,X:CRA    ;Set Word Length=8, CLK=5.12/32 MHz.
        MOVEP   #$1E30,X:CRB    ;Enable receiver, Mode=On-Demand,
                                ;gated clock on, synchronous mode,
                                ;Word frame sync selected, frame
                                ;sync and clock are external.
        MOVEP   #$1F0,X:PCC     ;Set PCC for SSI
LOOP
RDF1    JCLR    #7,X:SSISR,RDF1 ;Wait for RDF (receive data register
                                ;Full) go to high.
        MOVEP   X:RX,X:(R0)+    ;Read data from RX into memory.
        JMP     LOOP            ;Continue sequence forever.
        END
```

Figure 12-82. On-Demand Mode Receive Example Program

12.4.8 Flags

Two SSI pins (SC1 and SC0) are available in the synchronous mode for use as serial I/O flags. The control bits (OF1 and OF0) and status bits (IF1 and IF0) are double buffered to/from SC1 and SC0. Double buffering the flags keeps them in sync with TX and RX. The direction of SC1 and SC0 is controlled by SCD1 and SCD0 in CRB.

Figure 12-83 shows the flag timing for a network mode example. Initially, neither TIE nor TE is set, and the flag outputs are the last flag output value. When TIE is set, a TDE interrupt occurs (the transmitter does not have to be enabled for this interrupt to occur). Data (D1) is written to TX, which clears TDE, and the transmitter is enabled by software. When the frame sync occurs, data (D1) is transferred to the transmit shift register, setting TDE. Data (D1) is shifted out during the first word time, and the output flags are updated. These flags will remain stable until the next frame sync. The TDE interrupt is then serviced by writing data (D2) to TX, clearing TDE. After the TSR completes transmission, the transmit pin is three-stated until the next frame sync.

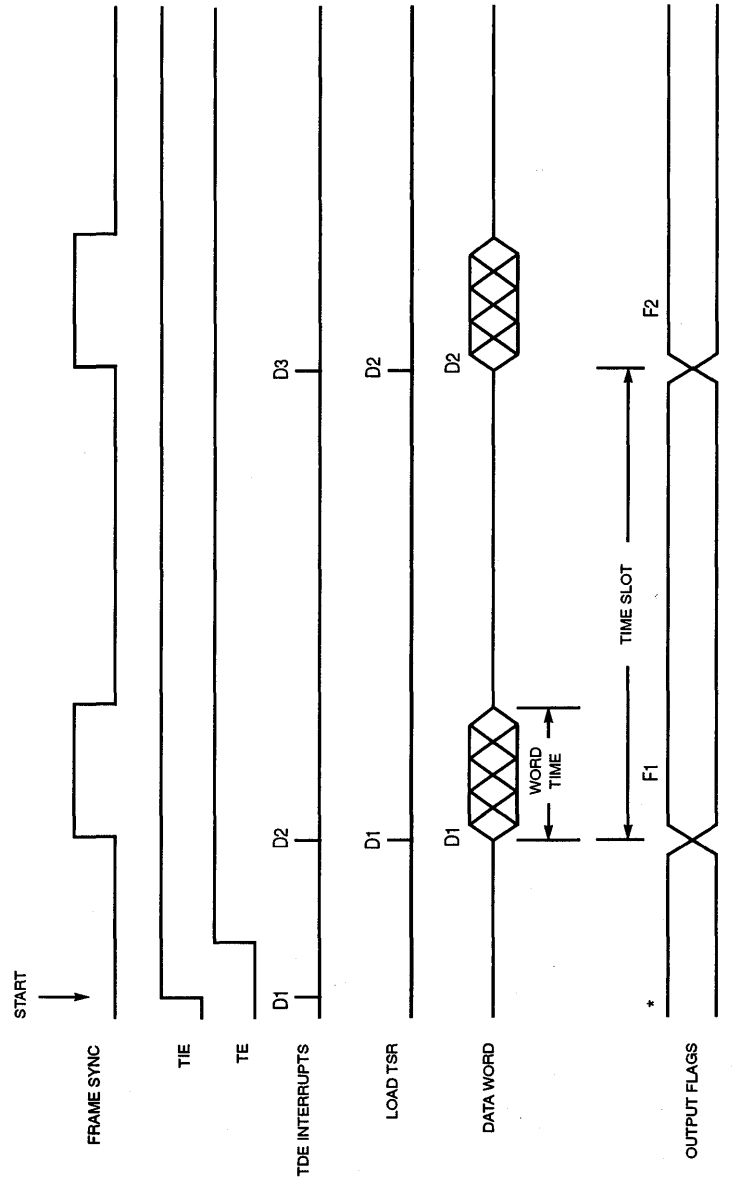
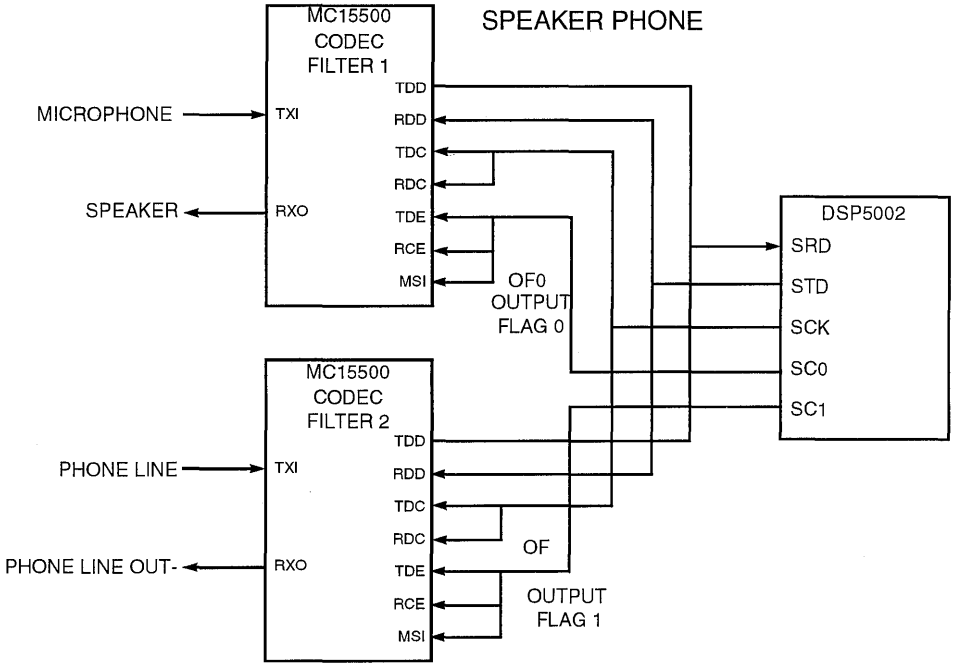


Figure 12-83. Output Flag Timing

12

Figure 12-84 shows a speaker phone example that uses a DSP56002 and two codecs. No additional logic is required to connect the codecs to the DSP. The two serial output flags in this example (OF1 and OF0) are used as chip selects to enable the appropriate codec for I/O. This procedure allows the transmit lines to be ORed together. The appropriate output flag pin changes at the same time as the first bit of the transmit word and remains stable until the next transmit word (see Figure 12-85). Applications include serial-device chip selects, implementing multidrop protocols, generating Bell PCM signaling frame syncs, and outputting status information.



NOTE: SC0 and SC1 are output flag 0 and 1 used to software select either filter 1 or 2.

Figure 12-84. Output Flag Example

12

Initializing the flags (see Figure 12-85) is accomplished by setting SYN, SCD1, and SCD0. No other control bits affect the flags. The synchronous control bit must be set (SYN=1) to select the SC1 and SC0 pins as flags. SCD1 and SCD0 select whether SC1 and SC0 are inputs or outputs (input=0, output=1).The other bits selected in Figure 12-85 are chosen for the speaker phone example in Figure 12-84. In this example, the codecs require that the SSI be set for normal mode (MOD=0) with a gated clock (GCK=1) out (SCKD=1).

Serial input flags, IF1 and IF0, are latched at the same time as the first bit is sampled in the receive data word (see Figure 12-86). Since the input was latched, the signal on the input flag pin can change without affecting the input flag until the first bit of the next receive data word. To initialize SC1 or SC0 as input flags, the synchronous control bit in CRB must be set to one (SYN=1) and SCD1 set to zero for pin SC1, and SCD0 must be set to zero for pin SC0. The input flags are bits 1 and 0 in the SSISR (at X:$FFEE).

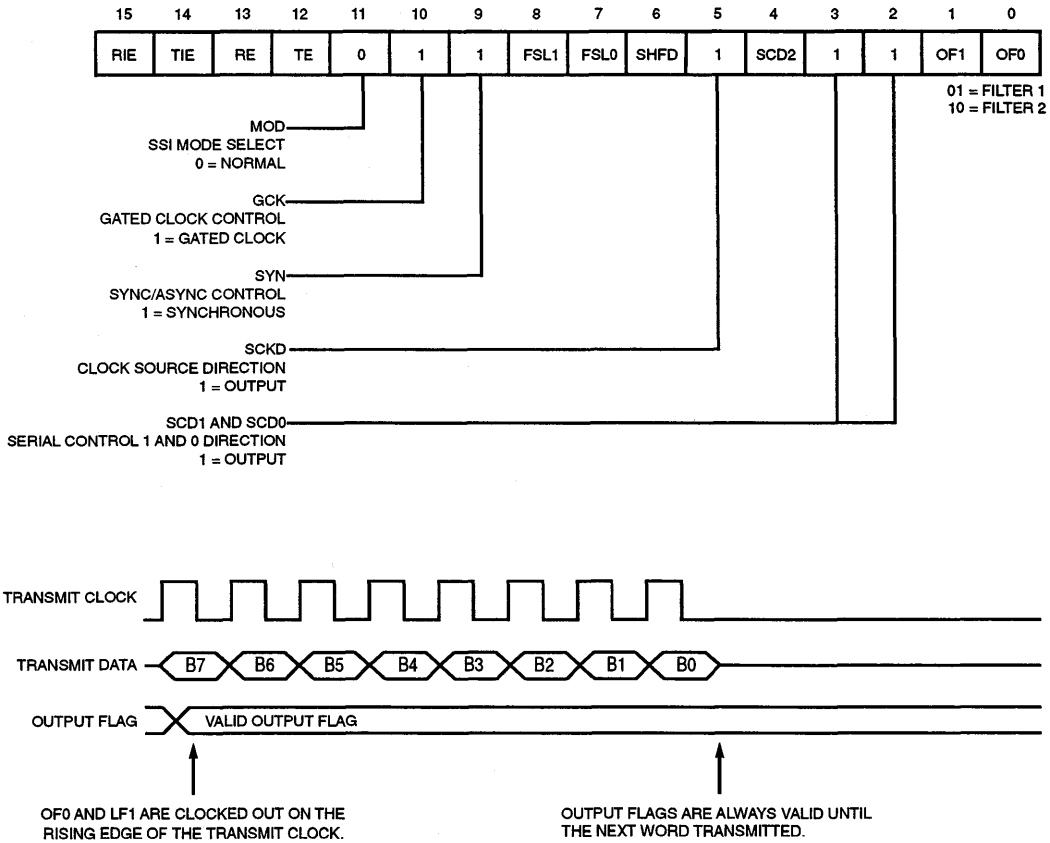


Figure 12-85. Output Flag Initialization

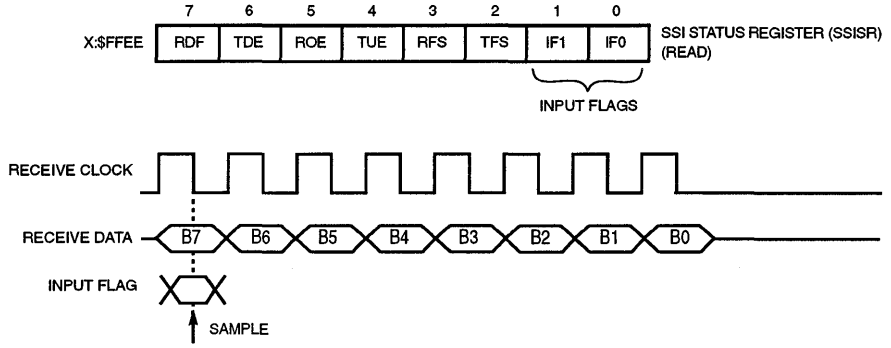


Figure 12-86. Input Flags

12.4.9 Example Circuits

The DSP-to-DSP serial network shown in Figure 12-87 uses no additional logic chips for the network connection. All serial data is synchronized to the data source (all serial clocks and serial syncs are common). This basic configuration is useful for decimation and data reduction when more processing power is needed than one DSP can provide. Cascading DSPs in this manner is useful in several network topologies including star and ring networks.

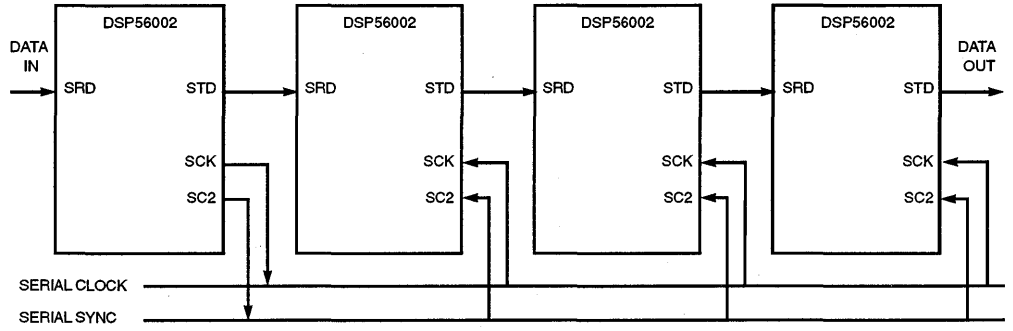


Figure 12-87. SSI Cascaded Multi-DSP System

TDM networks are useful to reduce the wiring needed for connecting multiple processors. A TDM parallel topology, such as the one shown in Figure 12-88, is useful for interpolating filters. Serial data can be received simultaneously by all DSPs, processing can occur in parallel, and the results are then multiplexed to a single serial data out line. This configuration can be cascaded and/or looped back on itself as needed to fit a particular application (see Figure 12-89). The serial and parallel configurations can be combined to form the array processor shown in Figure 12-90. A nearest neighbor array, which is applicable to matrix relaxation processing, is shown in Figure 12-91. To simplify the drawing, only the center DSP is connected in this illustration. In use, all DSPs would have four three-state buffers connected to their STD pin. The flags (SC0 and SC1) on the control master operate the three-state buff-

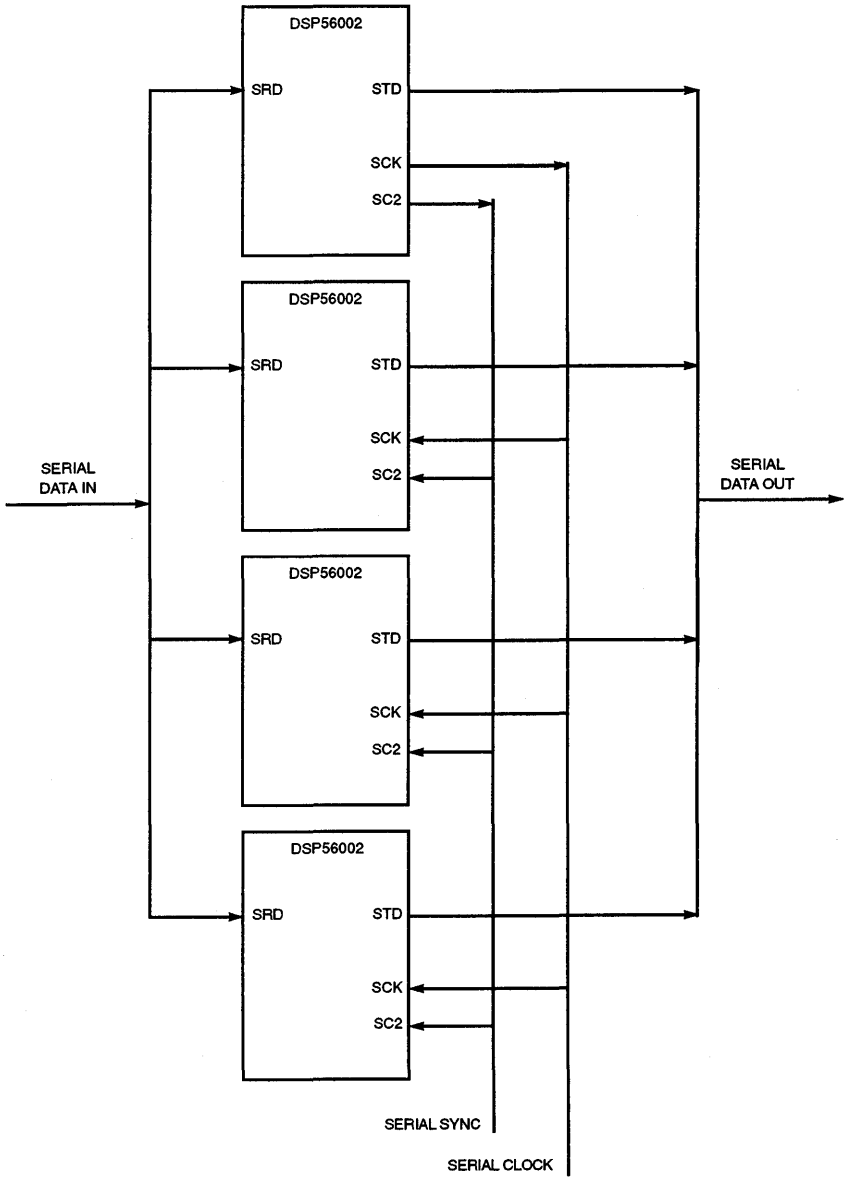


Figure 12-88. SSI TDM Parallel DSP Network

ers, which control the direction that data is transferred in the matrix (north, south, east, or west).

The bus architecture shown in Figure 12-92 allows data to be transferred between any two DSPs. However, the bus must be arbitrated by hardware or a software protocol to prevent

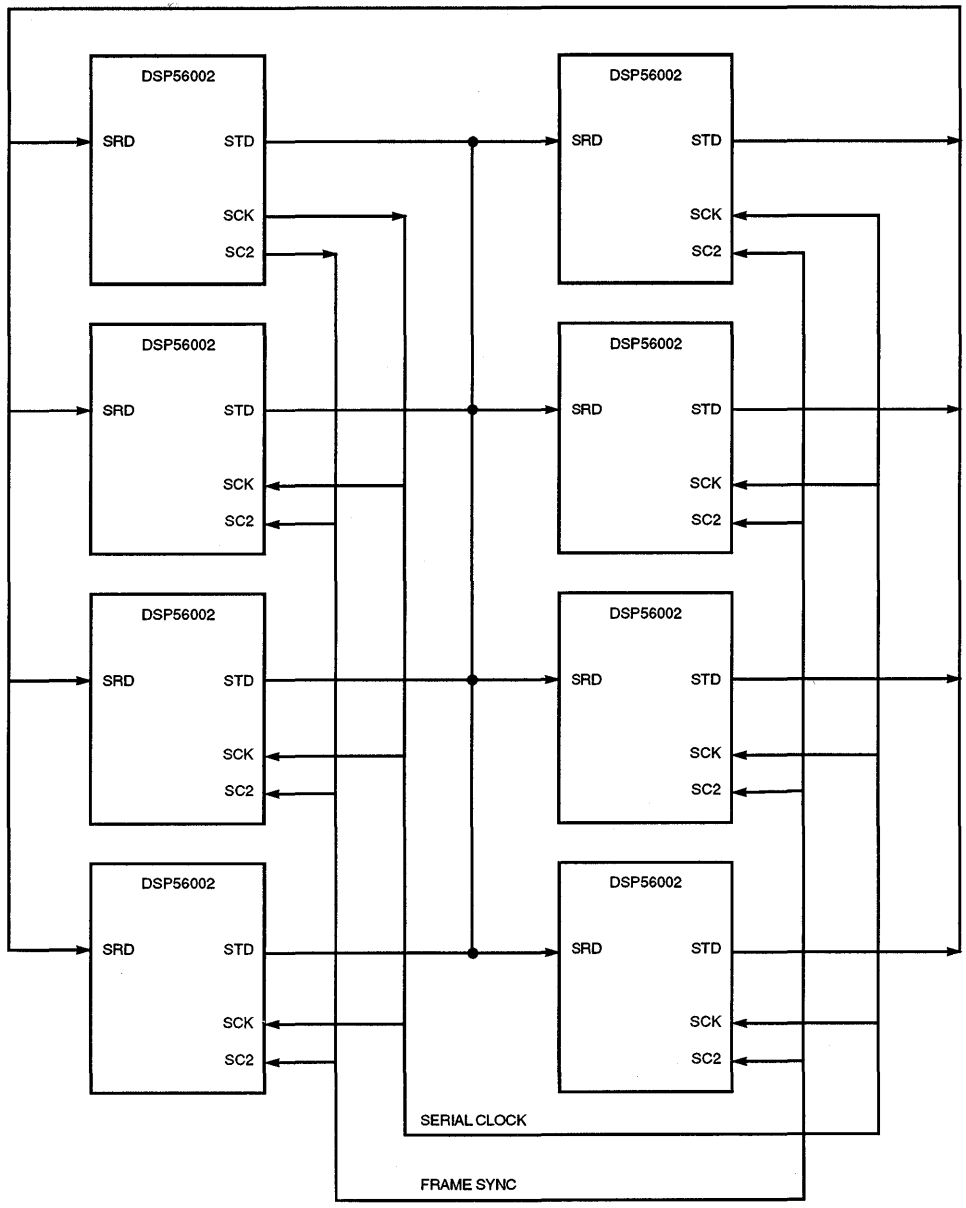


Figure 12-89. SSI TDM Connected Parallel Processing Array

collisions. The master/slave configuration shown in Figure 12-93 also allows data to be transferred between any two DSPs but simplifies network control.

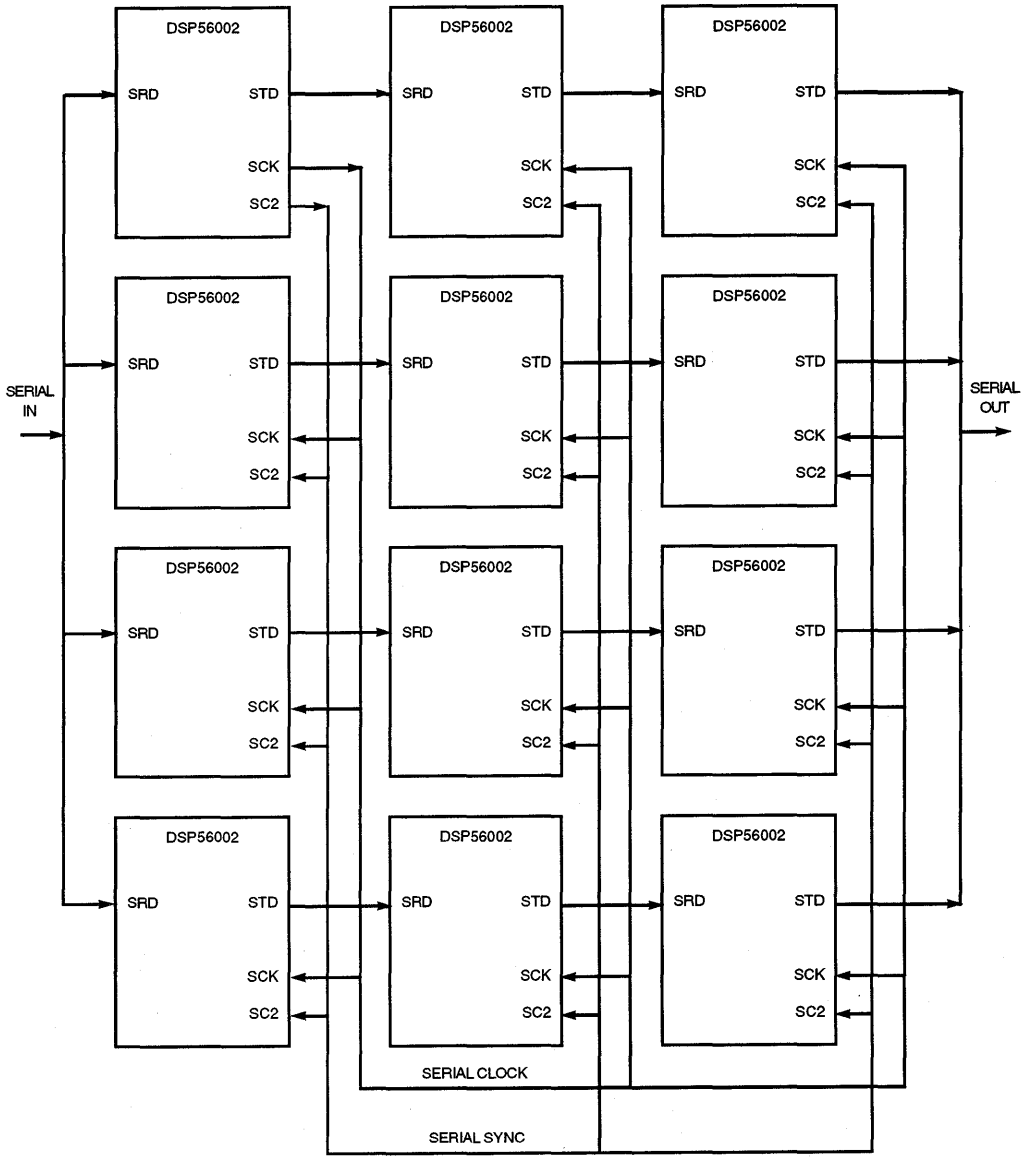


Figure 12-90. SSI TDM Serial/Parallel Processing Array

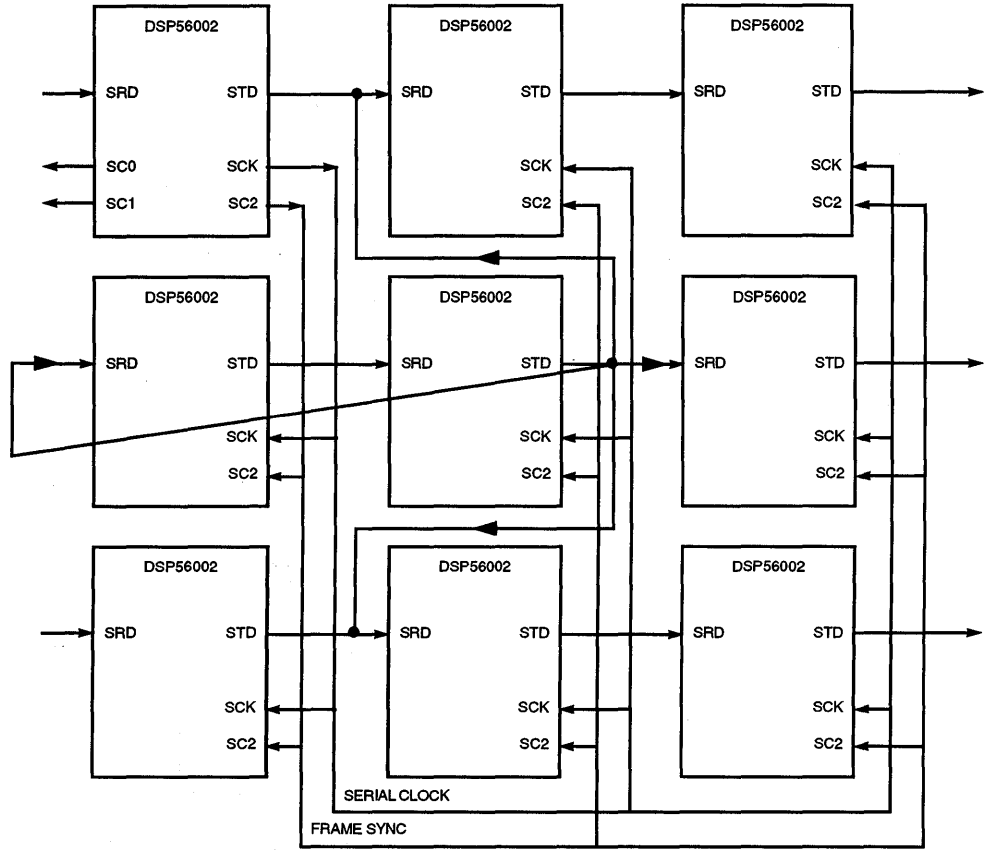


Figure 12-91. SSI Parallel Processing — Nearest Neighbor Array

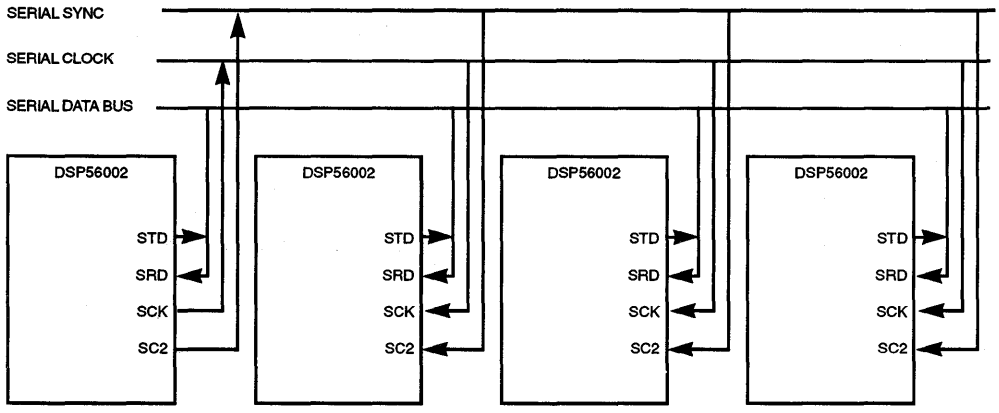
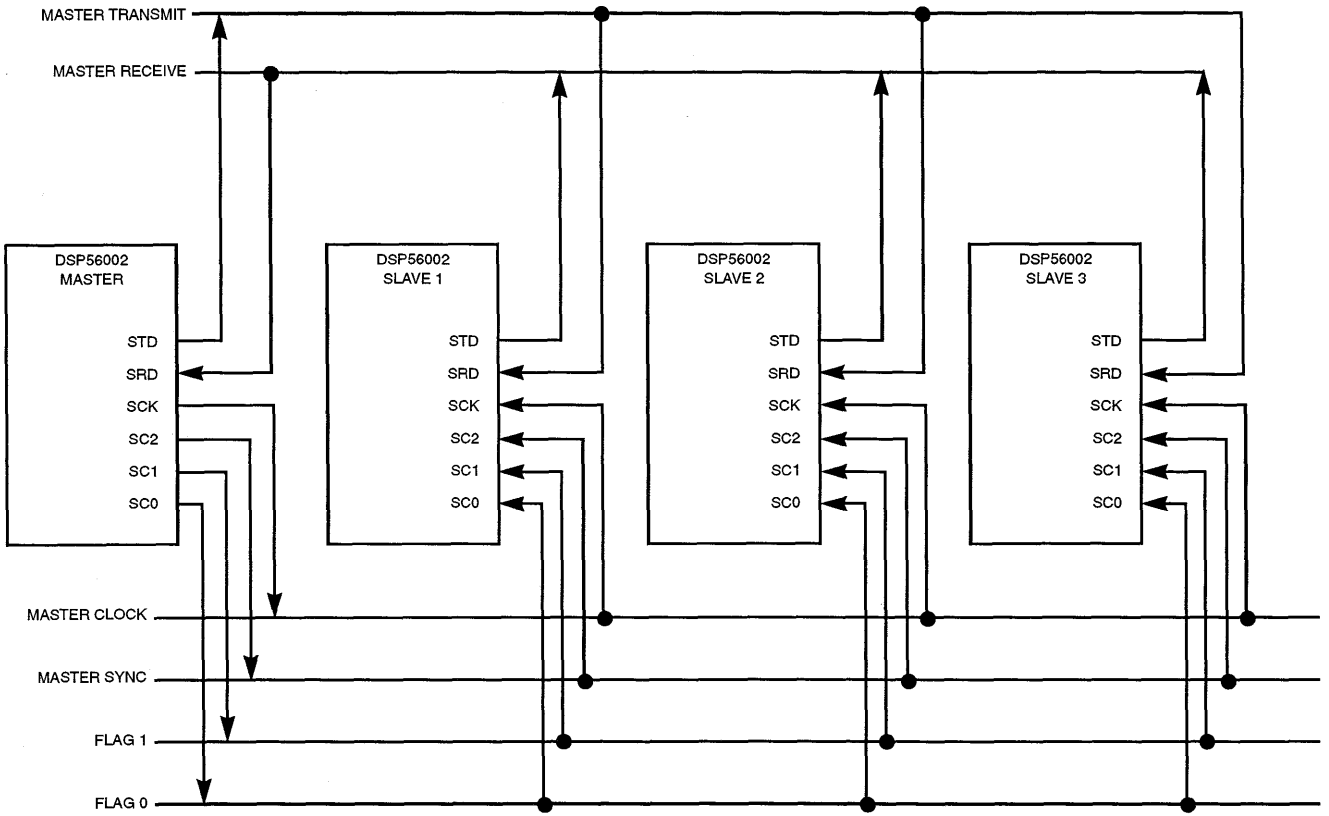


Figure 12-92. SSI TDM Bus DSP Network



NOTE: Flags can specify data types: control, address, and data.

Figure 12-93. SSI TDM Master-Slave DSP Network

SECTION 13
IEEE 1149.1 TEST ACCESS PORT

13.1 INTRODUCTION

The MC68356 provides a dedicated user-accessible test access port (TAP) that is fully compatible with the *IEEE 1149.1 Standard Test Access Port and Boundary Scan Architecture.* Problems associated with testing high density circuit boards have led to development of this proposed standard under the sponsorship of the Test Technology Committee of IEEE and the Joint Test Action Group (JTAG). The MC68356 implementation supports circuit-board test strategies based on this standard.

The MC68356 TAP contains addition signal, $\overline{\text{TRST}}$, which provides an asynchronous reset to the TAP.

Since the TAP pins are multiplexed with the DSP ONCE port pins, an additional pin, JTAG_$\overline{\text{ONCE}}$, controls whether the TAP is active (JTAG_$\overline{\text{ONCE}}$ = 1) or the DSP ONCE port is active (JTAG_$\overline{\text{ONCE}}$ = 0).

The test logic includes a test access port (TAP) consisting of six signal pins, a 16-state controller, and two test data registers. A boundary scan register links all device signal pins into a single shift register. The test logic, implemented utilizing static logic design, is independent of the device system logic. The MC68356 implementation provides the following capabilities:

1. Perform boundary scan operations to test circuit-board electrical continuity
2. Bypass the MC68356 for a given circuit-board test by effectively reducing the boundary scan register to a single cell
3. Sample the MC68356 system pins during operation and transparently shift out the result in the boundary scan register
4. Disable the output drive to pins during circuit-board testing

NOTE

Certain precautions must be observed to ensure that the IEEE 1149.1 test logic does not interfere with nontest operation. See 13.11 NON-IEEE 1149.1 Operation for details.

Additionally, although not specifically part of IEEE 1149.1, the MC68356 contains a signal which can be used to three-state all MC68356 output signals. This signal is called three-state, $\overline{\text{TRIS}}$.

13.2 OVERVIEW

This section, which includes aspects of the IEEE 1149.1 implementation that are specific to the MC68356, is intended to be used with the supporting IEEE 1149.1 document. The discussion includes those items required by the standard to be defined and, in certain cases, provides additional information specific to the MC68356 implementation. For internal details and applications of the standard, refer to the IEEE 1149.1 document.

An overview of the MC68356 implementation of IEEE 1149.1 is shown in Figure 13-2. The MC68356 implementation includes a 3-bit instruction register and two test registers, a 1-bit bypass register and a D262-bit boundary scan register.

In the MC68356, TAP pins are multiplexed with the DSP ONCE port pins. The JTAG_$\overline{\text{ONCE}}$ pin controls whether the TAP is active or the DSP ONCE port is active:

(JTAG_$\overline{\text{ONCE}}$ = 1) - TAP is active. The TAP interface includes the following signals:

- TCK = a test clock input to synchronize the test logic. This pin has an internal pull-up resistor (which is enabled when JTAG_$\overline{\text{ONCE}}$ = 1).
- $\overline{\text{TMS}}$= a test mode select input that is sampled on the rising edge of TCK to sequence the test controller's state machine. This pin has an internal pull-up resistor.
- TDI = a test data input that is sampled on the rising edge of TCK. This pin has an internal pull-up resistor (which is enabled when JTAG_$\overline{\text{ONCE}}$ = 1).
- TDO = a three-state test data output that is actively driven in the shift-IR and shift-TMS controller states. TDO changes on the falling edge of TCK.
- $\overline{\text{TRST}}$= an asynchronous reset which provides initialization of the TAP controller and other logic as required by the standard.This pin has an internal pull-up resistor (which is enabled when JTAG_$\overline{\text{ONCE}}$ = 1).

(JTAG_$\overline{\text{ONCE}}$ = 0) - DSP ONCE port is active. The TAP controller receives default values that cause it to be inactive and internal pull-ups to the DSCK/OS1/TCK, DSI/OS0/TDI, $\overline{\text{TRST}}$ pins are disabled. The $\overline{\text{DR}}$/$\overline{\text{TMS}}$ pin retains its internal an pull-up.

NOTE

JTAG_$\overline{\text{ONCE}}$ pin has an internal pull-down resistor of 50 kohm. To force JTAG_$\overline{\text{ONCE}}$ =1 the user must drive this pin with pull-up of 10 kohm or less.

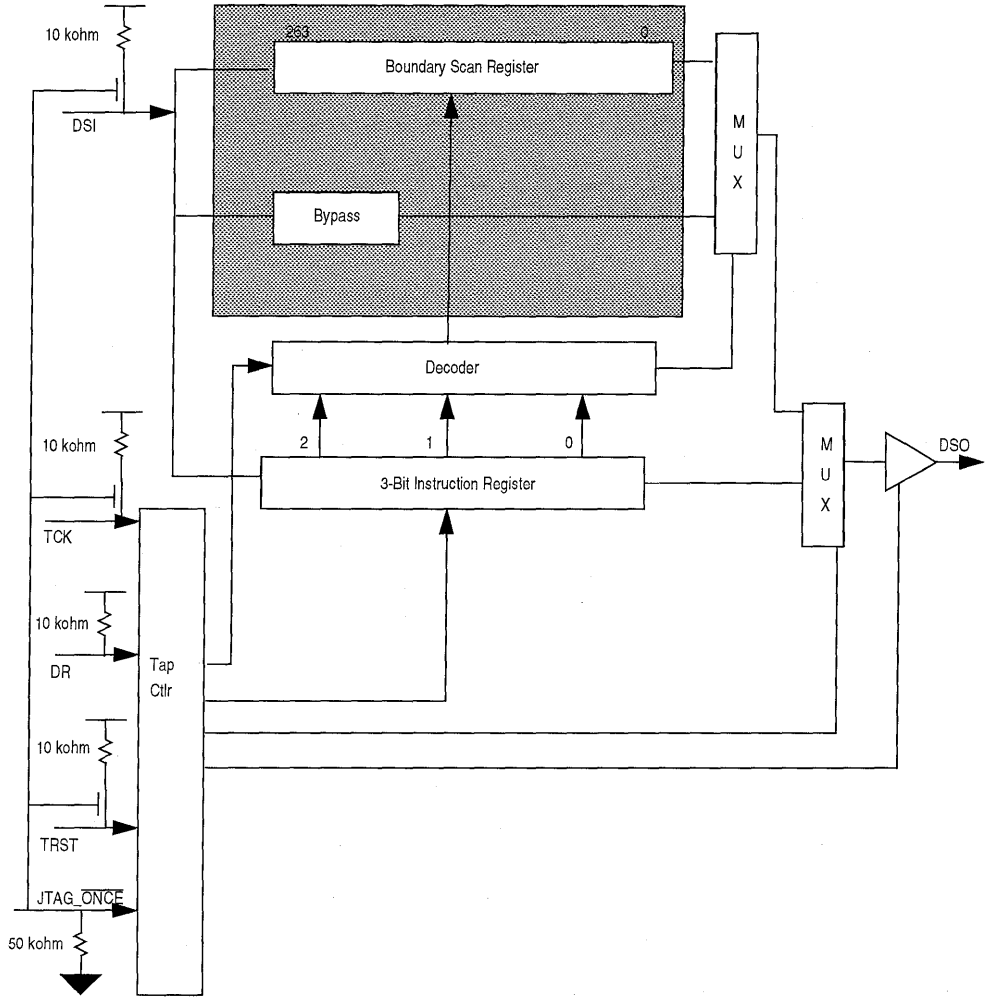


Figure 13-1. Test Logic Block Diagram

13.3 TAP CONTROLLER

The TAP controller is responsible for interpreting the sequence of logical values on the $\overline{\text{TMS}}$ signal. It is a synchronous state machine that controls the operation of the JTAG logic. The state machine is shown in Figure 13-2. The value shown adjacent to each arc represents the value of the $\overline{\text{TMS}}$ signal sampled on the rising edge of TCK signal. For a description of the TAP controller states, please refer to the IEEE 1149.1 document.

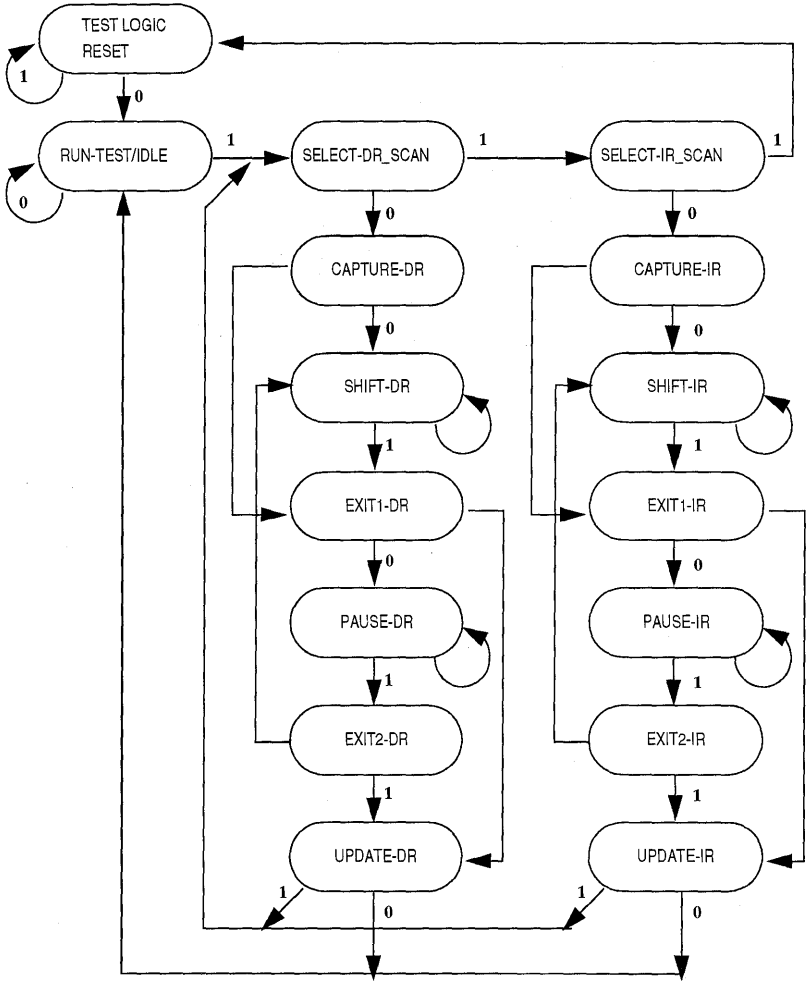


Figure 13-2. TAP Controller State Machine

13.4 BOUNDARY SCAN REGISTER

The MC68356 IEEE 1149.1 implementation has a 262-bit boundary scan register. This register contains bits for all device signal and clock pins and associated control signals. The XTAL, XFC, DXTAL, and PCAP pins are associated with analog signals and are not included in the boundary scan register.

All MC68356 bidirectional pins have a single register bit in the boundary scan register for pin data, and are controlled by an associated control bit in the boundary scan register. 63 bits in the boundary scan register define the output enable signal for associated groups of bidi-

rectional and three-stateable pins. The control bits and their bit positions are listed in Table 13-1

Table 13-1. Boundary Scan Control Bits

Name	Bit Number	Name	Bit Number	Name	Bit Number
kio_ha_jdir	12	kio_done_jdir	94	kio_txd3_jdir	151
kio_a_jdir	13	kio_rts2_jdir	96	kio_tin1_jdir	152
kio_pcd6_jdir	30	kio_dreq_jdir	97	kio_tout2_jdir	161
kio_pcd1_jdir	31	kio_dack_jdir	100	kio_tout1_jdir	162
kio_pcd7_jdir	32	kio_txd1_jdir	101	kio_wdog_jdir	165
kio_pcd5_jdir	36	kio_cd2_jdir	108	kio_fc_jdir	167
kio_pcd4_jdir	37	kio_rclk1_jdir	109	kio_pb10_jdir	169
kio_pcd3_jdir	40	kio_sds2_jdir	112	kio_iack7_jdir	170
kio_pcd2_jdir	41	kio_rxd2_jdir	113	kio_iack6_jdir	175
kio_pcd0_jdir	53	kio_tclk2_jdir	118	kio_avec_jdir	176
kio_rmc_jdir	62	kio_cts2_jdir	119	kio_d_jdir	183
kio_pc_jdir	63	kio_rxd3_jdir	126	TS0	202
kio_bg_jdir	76	kio_cd3_jdir	127	Ctrl7/sc0	225
kio_br_jdir	77	kio_rclk3_jdir	129	Ctrl6/sc1	226
kio_bgack_jdir	79	kio_tclk3_jdir	130	Ctrl5/sck	228
kio_dtack_jdir	80	kio_tin2_jdir	139	Ctrl4/sc2	229
kio_tclk1_jdir	84	kio_iack1_jdir	140	Ctrl3/srd	232
kio_ects_jdir	85	kio_pb8_jdir	144	Ctrl2/std	233
kio_txd2_jdir	89	kio_pb9_jdir	145	EPCPDE/std	237
kio_rclk2_jdir	90	kio_pb11_jdir	148	BRT0/DBTS	250
kio_uds_jdir	93	kio_pa12_jdir	149	Ctrl1	251

The boundary scan bit definitions are shown in Table 13-2.

The first column in the table defines the bit's ordinal position in the boundary scan register. The shift register cell nearest DSO (i.e., first to be shifted out) is defined as bit zero; the last bit to be shifted out is 261.

The second column references one of the four MC68356 cell types depicted in Figure 13-3 through Figure 13-6, which describe the cell structure for each type.

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

The fourth column lists the pin type for convenience where TS-Output indicates a three-stateable output pin, I/O indicates a bidirectional pin, and OD-I/O denotes an open-drain bidirectional pin.

The last column indicates the associated boundary scan register control bit for bidirectional, three-state, and open-drain output pins.

Bidirectional pins include a single scan cell for data (IO.Cell) as depicted in Figure 13-4.

These bits are controlled by the cell shown in Figure 13-5. The value of the control bit determines whether the bidirectional pin is an input or an output.

Table 13-2. Boundary Scan Bit Definition.

Bit Number	Cell/Signal Name	Pin Name	Pin Type	Output Control Cell
0	I/O	tris_	input	(-)
1	I/O	a23	I/O	kio_ha_jdir
2	I/O	a22	I/O	kio_ha_jdir
3	I/O	a21	I/O	kio_ha_jdir
4	I/O	a20	I/O	kio_ha_jdir
5	I/O	a19	I/O	kio_ha_jdir
6	I/O	a18	I/O	kio_ha_jdir
7	I/O	a17	I/O	kio_ha_jdir
8	I/O	a16	I/O	kio_ha_jdir
9	I/O	a15	I/O	kio_ha_jdir
10	I/O	a14	I/O	kio_ha_jdir
11	I/O	a13	I/O	kio_ha_jdir
12	ctrl	kio_ha_jdir	(-)	(-)
13	ctrl	kio_a_jdir	(-)	(-)
14	I/O	a12	I/O	kio_ha_jdir
15	I/O	a11	I/O	kio_a_jdir
16	I/O	a10	I/O	kio_a_jdir
17	I/O	a9	I/O	kio_a_jdir
18	I/O	a8	I/O	kio_a_jdir

Table 13-2. Boundary Scan Bit Definition.

Bit Number	Cell/Signal Name	Pin Name	Pin Type	Output Control Cell
19	I/O	a7	I/O	kio_a_jdir
20	I/O	a6	I/O	kio_a_jdir
21	I/O	a5	I/O	kio_a_jdir
22	I/O	a4	I/O	kio_a_jdir
23	I/O	a3	I/O	kio_a_jdir
24	I/O	a2	I/O	kio_a_jdir
25	I/O	a1	I/O	kio_a_jdir
26	I/O	pc_a2	input	(-)
27	I/O	pc_a3	input	(-)
28	I/O	pc_a4	input	(-)
29	I/O	pc_a5	input	(-)
30	ctrl	kio_pcd6_jdir	(-)	(-)
31	ctrl	kio_pcd1_jdir	(-)	(-)
32	ctrl	kio_pcd7_jdir	(-)	(-)
33	I/O	pc_d7	I/O	kio_pcd7_jdir
34	I/O	pc_d6	I/O	kio_pcd6_jdir
35	I/O	pc_d5	I/O	kio_pcd5_jdir
36	ctrl	kio_pcd5_jdir	(-)	(-)
37	ctrl	kio_pcd4_jdir	(-)	(-)
38	I/O	pc_d4	I/O	kio_pcd4_jdir
39	I/O	pc_d3	I/O	kio_pcd3_jdir
40	ctrl	kio_pcd3_jdir	(-)	(-)
41	ctrl	kio_pcd2_jdir	(-)	(-)
42	I/O	pc_d2	I/O	kio_pcd2_jdir
43	I/O	pc_d1	I/O	kio_pcd1_jdir
44	I/O	pc_d0	I/O	kio_pcd0_jdir
45	I/O	pc_a6	input	(-)
46	I/O	pc_a7	input	(-)
47	I/O	pc_a8	input	(-)

Table 13-2. Boundary Scan Bit Definition.

Bit Number	Cell/Signal Name	Pin Name	Pin Type	Output Control Cell
48	I/O	pc_a9	input	(-)
49	I/O	discpu	input	(-)
50	I/O	pc_iord_	input	(-)
51	I/O	pc_iowr_	input	(-)
52	I/O	pc_reg_	input	(-)
53	ctrl	kio_pcd0_jdir	(-)	(-)
54	I/O	pc_oe_	input	(-)
55	I/O	pc_ce1_	input	(-)
56	I/O	pc_we_	input	(-)
57	I/O	berr_	output	(-)
58	I/O	halt_	output	(-)
59	I/O	imp_reset_	output	(-)
60	I/O	rmc_	I/O	kio_rmc_jdir
61	I/O	iac	output	(-)
62	ctrl	kio_rmc_jdir	(-)	(-)
63	ctrl	kio_pc_jdir	(-)	(-)
64	I/O	bclr_	I/O	kio_pc_jdir
65	I/O	pc_rdy	output	(-)
66	I/O	pc_en	input	(-)
67	I/O	busw	input	(-)
68	I/O	pc_ce2_	input	(-)
69	input	imp_extal	input	(-)
70	output	clko	output	(-)
71	I/O	modclk1	input	(-)
72	I/O	modclk0	input	(-)
73	I/O	pc_a25	input	(-)
74	I/O	imp_br_	I/O	kio_br_jdir
75	I/O	imp_bg_	I/O	kio_bg_jdir
76	ctrl	kio_bg_jdir	(-)	(-)

Table 13-2. Boundary Scan Bit Definition.

Bit Number	Cell/Signal Name	Pin Name	Pin Type	Output Control Cell
77	ctrl	kio_br_jdir	(-)	(-)
78	I/O	bgack_	I/O	kio_bgack_jdir
79	ctrl	kio_bgack_jdir	(-)	(-)
80	ctrl	kio_dtack_jdir	(-)	(-)
81	I/O	dtack_	I/O	kio_dtack_jdir
82	I/O	as_	I/O	kio_ects_jdir
83	I/O	oe_	output	(-)
84	ctrl	kio_tclk1_jdir	(-)	(-)
85	ctrl	kio_ects_jdir	(-)	(-)
86	I/O	lds_	I/O	kio_ects_jdir
87	I/O	wel_	output	(-)
88	I/O	rw_	I/O	kio_ects_jdir
89	ctrl	kio_txd2_jdir	(-)	(-)
90	ctrl	kio_rclk2_jdir	(-)	(-)
91	I/O	weh_	output	(-)
92	I/O	uds_	I/O	kio_uds_jdir
93	ctrl	kio_uds_jdir	(-)	(-)
94	ctrl	kio_done_jdir	(-)	(-)
95	I/O	done_	I/O	kio_done_jdir
96	ctrl	kio_rts2_jdir	(-)	(-)
97	ctrl	kio_dreq_jdir	(-)	(-)
98	I/O	dreq_	I/O	kio_dreq_jdir
99	I/O	dack_	I/O	kio_dack_jdir
100	ctrl	kio_dack_jdir	(-)	(-)
101	ctrl	kio_txd1_jdir	(-)	(-)
102	I/O	cts1_	input	(-)
103	I/O	cd1_	input	(-)
104	I/O	rxd1	input	(-)
105	I/O	txd1	I/O	kio_txd1_jdir

Table 13-2. Boundary Scan Bit Definition.

Bit Number	Cell/Signal Name	Pin Name	Pin Type	Output Control Cell
106	I/O	tclk1	I/O	kio_tclk1_jdir
107	I/O	brg1	output	(-)
108	ctrl	kio_cd2_jdir	(-)	(-)
109	ctrl	kio_rclk1_jdir	(-)	(-)
110	I/O	rclk1	I/O	kio_rclk1_jdir
111	I/O	rts1_	output	(-)
112	ctrl	kio_sds2_jdir	(-)	(-)
113	ctrl	kio_rxd2_jdir	(-)	(-)
114	I/O	rxd2	I/O	kio_rxd2_jdir
115	I/O	txd2	I/O	kio_txd2_jdir
116	I/O	rclk2	I/O	kio_rclk2_jdir
117	I/O	tclk2	I/O	kio_tclk2_jdir
118	ctrl	kio_tclk2_jdir	(-)	(-)
119	ctrl	kio_cts2_jdir	(-)	(-)
120	I/O	cts2_	I/O	kio_cts2_jdir
121	I/O	rts2_	I/O	kio_rts2_jdir
122	I/O	cd2_	I/O	kio_cd2_jdir
123	I/O	sds2	I/O	kio_sds2_jdir
124	I/O	rxd3	I/O	kio_rxd3_jdir
125	I/O	cd3_	I/O	kio_cd3_jdir
126	ctrl	kio_rxd3_jdir	(-)	(-)
127	ctrl	kio_cd3_jdir	(-)	(-)
128	I/O	rclk3	I/O	kio_rclk3_jdir
129	ctrl	kio_rclk3_jdir	(-)	(-)
130	ctrl	kio_tclk3_jdir	(-)	(-)
131	I/O	tclk3	I/O	kio_tclk3_jdir
132	I/O	pa12	I/O	kio_pa12_jdir
133	I/O	rts3_	output	(-)
134	I/O	txd3	I/O	kio_txd3_jdir

Table 13-2. Boundary Scan Bit Definition.

Bit Number	Cell/Signal Name	Pin Name	Pin Type	Output Control Cell
135	I/O	cts3_	input	(-)
136	I/O	tout2_	I/O	kio_tout2_jdir
137	I/O	tout1_	I/O	kio_tout1_jdir
138	I/O	tin2	I/O	kio_tin2_jdir
139	ctrl	kio_tin2_jdir	(-)	(-)
140	ctrl	kio_iack1_jdir	(-)	(-)
141	I/O	iack1_	I/O	kio_iack1_jdir
142	I/O	wdog_	I/O	kio_wdog_jdir
143	I/O	pb8	I/O	kio_pb8_jdir
144	ctrl	kio_pb8_jdir	(-)	(-)
145	ctrl	kio_pb9_jdir	(-)	(-)
146	I/O	pb9	I/O	kio_pb9_jdir
147	I/O	pb10	I/O	kio_pb10_jdir
148	ctrl	kio_pb11_jdir	(-)	(-)
149	ctrl	kio_pa12_jdir	(-)	(-)
150	I/O	pb11	I/O	kio_pb11_jdir
151	ctrl	kio_txd3_jdir	(-)	(-)
152	ctrl	kio_tin1_jdir	(-)	(-)
153	I/O	tin1	I/O	kio_tin1_jdir
154	I/O	iack7_	I/O	kio_iack7_jdir
155	I/O	iack6_	I/O	kio_iack6_jdir
156	I/O	ipl0_	input	(-)
157	I/O	ipl1_	input	(-)
158	I/O	ipl2_	input	(-)
159	I/O	avec_	I/O	kio_avec_jdir
160	I/O	fc2	I/O	kio_fc_jdir
161	ctrl	kio_tout2_jdir	(-)	(-)
162	ctrl	kio_tout1_jdir	(-)	(-)
163	I/O	fc1	I/O	kio_fc_jdir

Table 13-2. Boundary Scan Bit Definition.

Bit Number	Cell/Signal Name	Pin Name	Pin Type	Output Control Cell
164	I/O	fc0	I/O	kio_fc_jdir
165	ctrl	kio_wdog_jdir	(-)	(-)
166	ctrl	kio_fc_jdir	(-)	(-)
167	I/O	cs0_	output	(-)
168	I/O	cs1_	output	(-)
169	ctrl	kio_pb10_jdir	(-)	(-)
170	ctrl	kio_iack7_jdir	(-)	(-)
171	I/O	cs2_	output	(-)
172	I/O	cs3_	output	(-)
173	I/O	d0	I/O	kio_d_jdir
174	I/O	d1	I/O	kio_d_jdir
175	ctrl	kio_iack6_jdir	(-)	(-)
176	ctrl	kio_avec_jdir	(-)	(-)
177	I/O	d2	I/O	kio_d_jdir
178	I/O	d3	I/O	kio_d_jdir
179	I/O	d4	I/O	kio_d_jdir
180	I/O	d5	I/O	kio_d_jdir
181	I/O	d6	I/O	kio_d_jdir
182	I/O	d7	I/O	kio_d_jdir
183	ctrl	kio_d_jdir	(-)	(-)
184	I/O	d8	I/O	kio_d_jdir
185	I/O	d9	I/O	kio_d_jdir
186	I/O	d10	I/O	kio_d_jdir
187	I/O	d11	I/O	kio_d_jdir
188	I/O	d12	I/O	kio_d_jdir
189	I/O	d13	I/O	kio_d_jdir
190	I/O	d14	I/O	kio_d_jdir
191	I/O	d15	I/O	kio_d_jdir
192	ITAOB	IAB0	Output	(-)

Table 13-2. Boundary Scan Bit Definition.

Bit Number	Cell/Signal Name	Pin Name	Pin Type	Output Control Cell
193	ITAOB	IAB1	Output	(-)
194	ITAOB	IAB2	Output	(-)
195	ITAOB	IAB3	Output	(-)
196	ITAOB	IAB4	Output	(-)
197	ITAOB	IAB5	Output	(-)
198	ITAOB	IAB6	Output	(-)
199	ITAOB	IAB7	Output	(-)
200	ITAOB	IAB8	Output	(-)
201	ITAOB	IAB9	Output	(-)
202	TS0/abTs0	(-)	(-)	(-)
203	ITAOB	IAB10	Output	(-)
204	ITAOB	IAB11	Output	(-)
205	ITAOB	IAB12	Output	(-)
206	ITAOB	IAB13	Output	(-)
207	ITAOB	IAB14	Output	(-)
208	ITAOB	IAB15	Output	(-)
209	ITABM	PS_	Output	(-)
210	ITABM	DS_	Output	(-)
211	ITABM	XY_	Output	(-)
212	ITBSG	BSG_	Output	(-)
213	ITGWT	BRWT_	Output	(-)
214	ITCRD	RD_	Output	(-)
215	ITCWR	WR_	Output	(-)
216	ITCOC	CKOUT	Output	(-)
217	ITGIL	SELECT	Input	(-)
218	ITPRT	EXTAL2	Input	(-)
219	ITRIN	RESET_	Input	(-)
220	ITMIT	MODA	Input	(-)
221	ITMIT	MODB	Input	(-)

Table 13-2. Boundary Scan Bit Definition.

Bit Number	Cell/Signal Name	Pin Name	Pin Type	Output Control Cell
222	ITMIT	MODC	Input	(-)
223	ITPBD	SC0	I/O	Ctrl7
224	ITPBD	SC1	I/O	Ctrl6
225	Ctrl7/sc0	(-)	(-)	(-)
226	Ctrl6/sc1	(-)	(-)	(-)
227	ITPBD	SCK	I/O	Ctrl5
228	Ctrl5/sck	(-)	(-)	(-)
229	Ctrl4/sc2	(-)	(-)	(-)
230	ITPBD	SC2	I/O	Ctrl4
231	ITPBD	SRD	I/O	Ctrl3
232	Ctrl3/srd	(-)	(-)	(-)
233	Ctrl2/std	dir	(-)	(-)
234	ITPBD	STD	I/O	Ctrl2
235	ITDBD	IDB23	I/O	Ctrl1
236	ITDBD	IDB22	I/O	Ctrl1
237	EPCPDE/std	dir	(-)	(-)
238	ITDBD	IDB21	I/O	Ctrl1
239	ITDBD	IDB20	I/O	Ctrl1
240	ITDBD	IDB19	I/O	Ctrl1
241	ITDBD	IDB18	I/O	Ctrl1
242	ITDBD	IDB17	I/O	Ctrl1
243	ITDBD	IDB16	I/O	Ctrl1
244	ITDBD	IDB15	I/O	Ctrl1
245	ITDBD	IDB14	I/O	Ctrl1
246	ITDBD	IDB13	I/O	Ctrl1
247	ITDBD	IDB12	I/O	Ctrl1
248	ITDBD	IDB11	I/O	Ctrl1
249	ITDBD	IDB10	I/O	Ctrl1
250	BRT0/DBTS	(-)	(-)	(-)

Table 13-2. Boundary Scan Bit Definition.

Bit Number	Cell/Signal Name	Pin Name	Pin Type	Output Control Cell
251	Ctrl1	(-)	(-)	(-)
252	ITDBD	IDB9	I/O	Ctrl1
253	ITDBD	IDB8	I/O	Ctrl1
254	ITDBD	IDB7	I/O	Ctrl1
255	ITDBD	IDB6	I/O	Ctrl1
256	ITDBD	IDB5	I/O	Ctrl1
257	ITDBD	IDB4	I/O	Ctrl1
258	ITDxBD	IDB3	I/O	Ctrl1
259	ITDBD	IDB2	I/O	Ctrl1
260	ITDBD	IDB1	I/O	Ctrl1
261	ITDBD	IDB0	I/O	Ctrl1

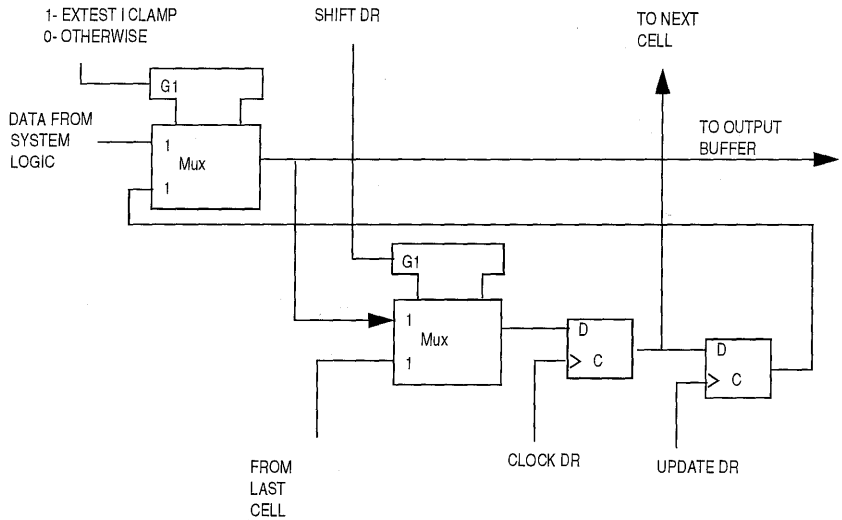


Figure 13-3. Output Latch Cell (O.latch)

13

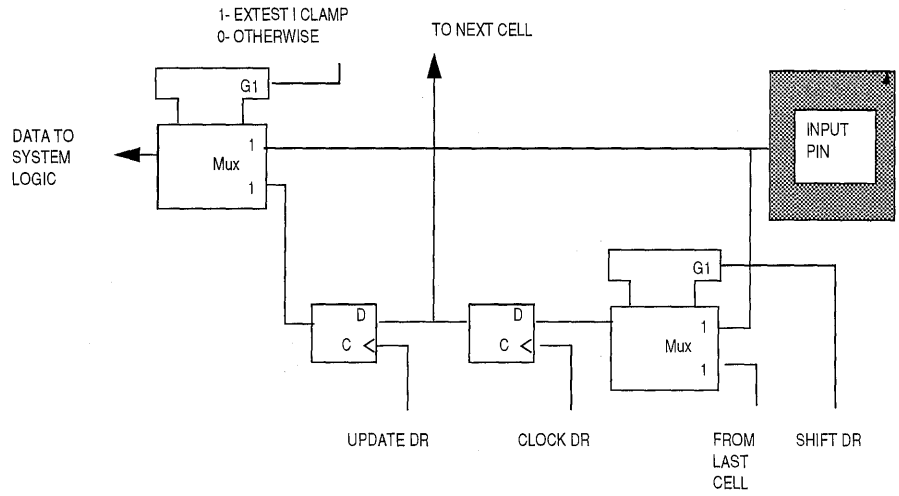


Figure 13-4. Input Pin Cell (I.Pin)

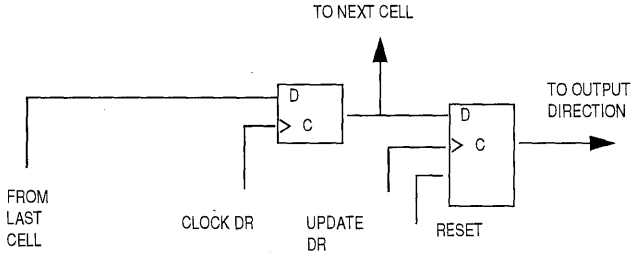


Figure 13-5. Control Cell (IO.Ctl)

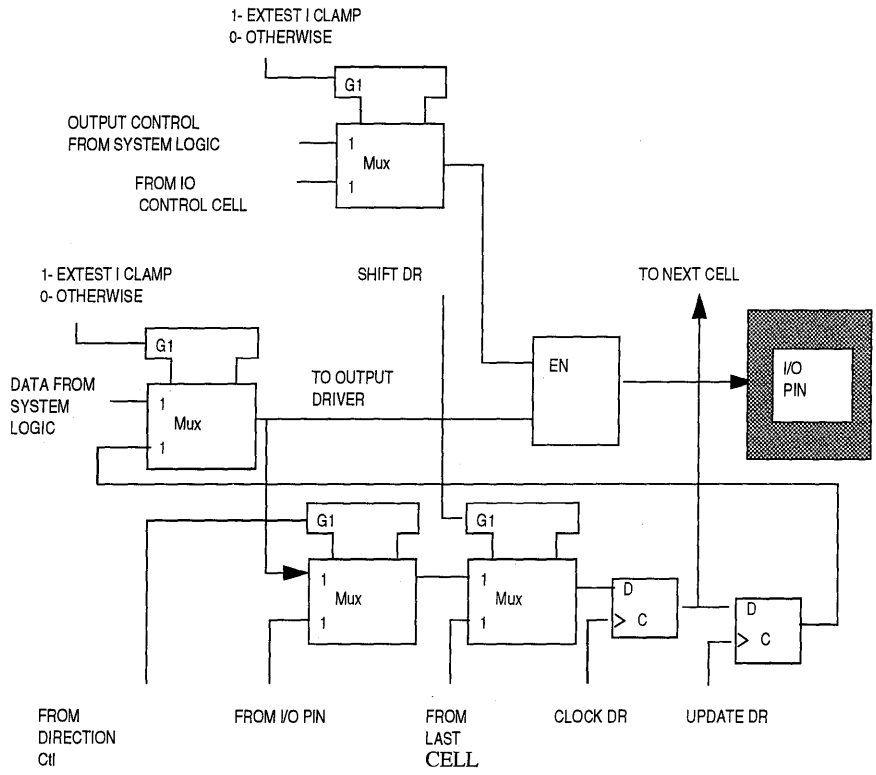
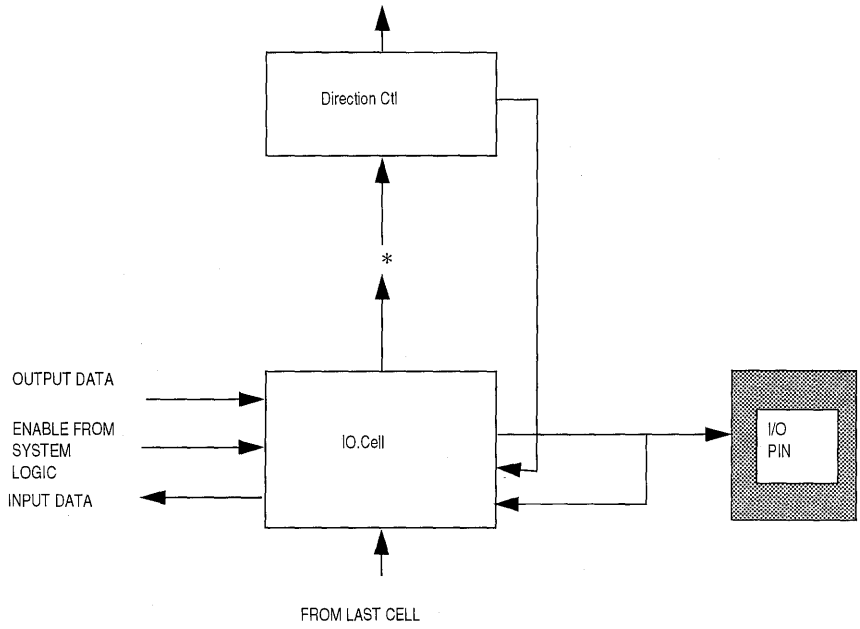


Figure 13-6. Bidirectional Data Cell (IO.Cell)

13



NOTE

More than one IO.Cell could be serially connected and controlled by a single IO.Ctl cell.

Figure 13-7. General Arrangement for Bidirectional Pins

13.5 INSTRUCTION REGISTER

The MC68356 IEEE 1149.1 implementation includes the three mandatory public instructions (EXTEST, SAMPLE/PRELOAD, and BYPASS), and also supports the optional CLAMP instruction defined by IEEE 1149.1. One additional public instruction (HI-Z) provides the capability for disabling all device output drivers. The MC68356 includes a 3-bit instruction register without parity consisting of a shift register with three parallel outputs. Data is transferred from the shift register to the parallel outputs during the update-IR controller state. The three bits are used to decode the five unique instructions shown in Table 13-3.

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

Table 13-3. Instructions

Code			Instruction
B2	B1	B0	
0	0	0	EXTEST
0	0	1	SAMPLE/PRELOAD
X	1	X	BYPASS
1	0	0	HI-Z
1	0	1	CLAMP and BYPASS

During the capture-IR controller state, the parallel inputs to the instruction shift register are loaded with the CLAMP command code.

13.6 EXTEST

The external test (EXTEST) instruction selects the 262-bit boundary scan register.

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

13.7 SAMPLE/PRELOAD

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

NOTE

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

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

13.8 BYPASS

The BYPASS instruction selects the single-bit bypass register as shown in Figure 13-8. This creates a shift-register path from TDI to the bypass register and, finally, to DSO, circum-

venting the 262-bit boundary scan register. This instruction is used to enhance test efficiency when a component other than the MC68356 becomes the device under test.

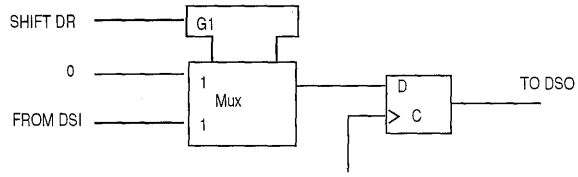


Figure 13-8. Bypass Register

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

13.9 CLAMP

The CLAMP instruction selects the single-bit bypass register as shown in Figure 13-8 and the state of all the signals driven from system output pins is completely defined by the data previously shifted into the boundary scan register (for example, using the SAMPLE/PRELOAD instruction).

13.10 HI-Z

The HI-Z instruction is not included in the IEEE 1149.1 standard. It is provided as a manufacturer's optional public instruction to prevent having to backdrive the output pins during circuit-board testing. When HI-Z is invoked, all output drivers, including the two-state drivers, are turned off (i.e., high impedance). The instruction selects the bypass register.

NOTE

On the MC68356, the TRIS_ pin may also be used during system reset to perform the same function.

13.11 NON-IEEE 1149.1 OPERATION

In non-IEEE 1149.1 operation the TAP must be in test-logic-reset state.

TAP is forced into test-logic-reset state by:
- driving the $\overline{\text{TRST}}$ signal low, or
- when JTAG_$\overline{\text{ONCE}}$ is set, driving the TDI input high for at least 5 rising edges of TCK.

When the TAP is in the test-logic-reset state, the state can be maintained by:
- driving the $\overline{\text{TRST}}$ input low, or

- driving JTAG_$\overline{\text{ONCE}}$ input low, or
- driving JTAG_$\overline{\text{ONCE}}$ and $\overline{\text{TMS}}$ inputs high.

NOTE

In default, JTAG_$\overline{\text{ONCE}}$ pin is driven low by an internal pull-down, DR_ is driven high by internal pull-up, and the conditional pull-ups of the $\overline{\text{TRST}}$, TDI and TCK pins are disconnected. Therefore, by connecting GND to the $\overline{\text{TRST}}$ pin, the user ensures that the TAP is always in the test-logic-reset state.

13.12 MC68356 RESTRICTIONS

The control afforded by the output enable signals using the boundary scan register and the EXTEST instruction requires a compatible circuit-board test environment to avoid device-destructive configurations. The user must avoid situations in which the MC68356 output drivers are enabled into actively driven networks.

The MC68356 features low-power modes. In order to reduce the power consumption during low power modes, it is recommended to keep the TAP in test-logic-reset state. Leaving the TAP controller test-logic-reset state negates the ability to achieve low-power, but does not otherwise affect device functionality.

The methods to keep the TAP in test-logic-reset state are described in 13.11 NON-IEEE 1149.1 Operation.

SECTION 14
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.

14.1 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
Current Drain per Pin		excluding V_{CC} and V_{SS}	mA
Operating Temperature Range MC68356ZP MC68356CZP	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})

14.2 THERMAL CHARACTERISTICS

Characteristic	Symbol	Value	Unit
Thermal Resistance for BGA	θ_{JA}	47[1]	°C/W
	θ_{JA}	30[2]	°C/W
	θ_{JA}	15[3]	°C/W

Notes:
1. Assumes natural convection and a single layer board (no thermal vias).
2. Assumes natural convection, a multi layer board with thermal vias[4], 1 watt 68356 dissipation, and a board temperature rise of 20°C above ambient.
3. Assumes natural convection, a multi layer board with thermal vias[4],1 watt 68356 dissipation, and a board temperature rise of 10°C above ambient.
4. For more information on the design of thermal vias on multilayer boards and BGA layout considerations in general, refer to AN-1231/D, "Plastic Ball Grid Array Application Note" available from your local Motorola sales office.

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

$P_D = (V_{DD} \bullet I_{DD}) + P_{I/O}$

where:

$P_{I/O}$ is the power dissipation on pins.

14.3 POWER CONSIDERATIONS

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

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

where:

T_A=Ambient Temperature, ∞C

q_{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

$$P_D = K \Pi (T_J + 273 \infty C) (2)$$

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

$$K = P_D \bullet (T_A + 273 \infty C) + q_{JA} \bullet P_D^2 (3)$$

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.

14.4 POWER DISSIPATION (SEE NOTE)

Characteristic Power Dissipation	Symbol	5 Volt		Unit
		Typ	Max	
IMP Normal Mode at 25 MHz	$P_{D(I)}$	105	TBD	mA
IMP Normal Mode at 20 MHz	$P_{D(I)}$	84	TBD	mA
DSP Normal Mode at 60 MHz	$P_{D(D)}$	120	TBD	mA
DSP Normal Mode at 45 MHz	$P_{D(D)}$	90	TBD	mA
IMP Standby Mode	$P_{DSB(I)}$	7	TBD	mA
IMP Doze Mode	$P_{DDZ(I)}$	500	TBD	µA
IMP Stop Mode	$P_{DSP(I)}$	100	TBD	µA
DSP Wait Mode	$P_{DSP(D)}$	10	TBD	mA
DSP Stop Mode	$P_{DSP(D)}$	2	TBD	µA

Note: These values are preliminary estimates. Test values are TBD.

14.4.1 Layout Practices

Each V_{CC} pin on the MC68356 should be provided with a low-impedance path to the board's supply. Each GND pin should likewise be provided with a low-impedance path to ground. The power supply pins drive distinct groups of logic on chip. The V_{CC} power supply should be bypassed to ground using at least four 0.1 µF by-pass capacitors located as close as possible to the four sides of the package. The capacitor leads and associated printed circuit traces connecting to chip V_{CC} and GND should be kept to less than 1/2" per capacitor lead. A four-layer board is recommended, employing two inner layers as V_{CC} and GND planes. All output pins on the MC68356 have fast rise and fall times. Printed circuit (PC) trace interconnection length should be minimized in order to minimize undershoot and reflections caused by these fast output switching times. This recommendation particularly applies to the address and data buses as well as the $\overline{RD}$, $\overline{WR}$, $\overline{IRQA}$, $\overline{IRQB}$, and $\overline{NMI}$ pins. Maximum PC trace lengths on the order of 6" are recommended. Capacitance calculations should consider all device loads as well as parasitic capacitances due to the PC traces. Attention to proper PCB layout and bypassing becomes especially critical in systems with higher capacitive loads because these loads create higher transient currents in the V_{CC} and GND circuits. Pull up all unused inputs or signals that will be inputs during reset. Special care should be taken to minimize the noise levels on the PLL supply pins.

14.4.2 Power Dissipation Considerations.

Power dissipation is a key issue in portable applications. Some of the factors which affect current consumption are described in this section. Most of the current consumed by CMOS devices is alternating current (AC) which is charging and discharging the capacitances of the pins and internal nodes. Therefore, the total current consumption is the sum of these internal and external currents.

This current consumption is described by the formula:

$I = CVf$
where C=node/pin capacitance
V=voltage swing and
f=frequency of node/pin toggle.

Example: for a port A address pin loaded with a 50pF capacitance, operating at 5.5V and with a 40MHz clock, toggling at its maximum possible rate (which is 10MHz), the current consumption is:

$$I = 50 \cdot 10^{-12} \cdot 5.5 \cdot 10 \cdot 10^{6} = 2.75\text{mA}$$

The maximum internal current value (I_{CCI}-max), reflects the maximum possible switching of the internal buses, which is not necessarily a real application case. The typical internal current value ($I_{CCI\text{-}typ}$) reflects the average switching of the internal buses.

For applications which require very low current consumption it is recommended to:

a) minimize external memory accesses, and use internal memory accesses instead;
b) minimize the number of pins which are switching;
c) minimize the capacitive load on the pins;
d) connect the unused inputs to pull-up or pull-down resistors.

14.4.3 DC Electrical Characteristics

Characteristic	Symbol	Min	Max	Unit
Input High Voltage (Except pins noted below)	V_{IH}	2.0	V_{DD}	V
Input High Voltage(TRIS, TRST, JTAG_ONCE, DR, DSO, DSCK, DSI, PC_A2-PC_A9, PC_A25, PC_D7-PC_D0, PC_IORD, PC_IOWR, PC_REG, PC_OE, PC_WE, PC_CE1, PC_CE2, PC_RDY, PC_EN, BUSW, MODCLK1, MODCLK0, DACK, CTS1, CD1, RXD1, TXD1, TCLK1, BRG1, RCLK1, RTS1, RXD2, TXD2, RCLK2, TCLK2, CTS2, RTS2, CD2, SDS2, RXD3, CD3, RCLK3, TCLK3, PA12, RTS3, TXD3, CTS3, TOUT2, TOUT1, TIN2, TIN1, IACK1, IACK6, IACK7, WDOG, PB8-11) (These pins have schmitt trigger inputs)	V_{IH}	2.5	V_{DD}	V
Input Low Voltage (Except DEXTAL,EXTAL, MODA/IRQA, MODB/IRQB, MODC/IRQC)	V_{IL}	V_{SS} – 0.3	0.8	V
Input High Voltage (DEXTAL,EXTAL)	V_{CIH}	0.7 (V_{CC})	V_{CC} + 0.3	V
Input Undershoot Voltage (DEXTAL,EXTAL)	V_{CIL}	V_{SS} – 0.3	0.6	V
Input High Voltage (DRESET,RESET)	V_{IHR}	2.5	V_{DD}	V
Input High Voltage (MODA/IRQA, MODB/IRQB, MODC/IRQC)	V_{IHM}	3.5	V_{DD}	V
Input Low Voltage (MODA/IRQA, MODB/IRQB, MODC/IRQC)	V_{ILM}	– 0.5	2.0	V
Input Leakage Current: Input Pins I/O Pins	I_{IN}	— —	20 TBD	µA
Input Capacitance All Pins	C_{IN}	—	15	pF
Three-State Leakage Current (2.4/0.5 V)	I_{TSI}	– 20	20	µA
Open Drain Leakage Current (2.4 V)	I_{OD}	—	20	µA
Output High Voltage (I_{OH} = 400 µA) (see Note)	V_{OH}	2.4	—	V
Output Low Voltage (I_{OL} = 3.2 mA) A1–A23, PB0–PB11, FC0–FC2, CS0–CS3, IACK, AVEC, BG, RCLK1, RCLK2, RCLK3, TCLK1, TCLK2, TCLK3, RTS1, RTS2, RTS3, SDS2, PA12, RXD2, RXD3, CTS2, CD2, CD3, DREQ, BRG1 (I_{OL} = 5.3 mA) AS, UDS, LDS, R/W, BERR, BGACK, BCLR, DTACK, DACK, D0–D15, RESET (I_{OL} = 7.0 mA) TXD1, TXD2, TXD3 (I_{OL} = 8.9 mA) DONE, HALT, BR (as output) (I_{OL} = 3.2 mA) CLKO (I_{OL} = 3.2 mA) All other pins	V_{OL}	— — — — — —	0.5 0.5 0.5 0.5 0.4 0.5	V
Output Drive CLKO, CKOUT Output Drive Except CLKO, CKOUT)	O_{CLK} O_{ALL}	— —	50 100	pF pF
Output Drive Derating Factor for CLKO of 0.030 ns/pF Output Drive Derating Factor for CLKO of 0.025 ns/pF Output Drive Derating Factor for All Other Pins 0.025 ns/pF Output Drive Derating Factor for All Other Pins 0.05 ns/pF	O_{KF} O_{KF} O_{KF} O_{KF}	20 50 20 100	50 130 100 200	pF pF pF pF
Power	V_{DD}	4.75	5.25	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)).

14.5 IMP CHARACTERISTICS

The IMP and DSP characteristics are divided into two groups because of their different maximum clock speeds and different methods of characterization.

14.5.1 IMP AC ELECTRICAL SPECIFICATIONS CONTROL TIMING

(GND = 0 Vdc, TA = 0 to 70°C; The electrical specifications in this document are preliminary; See Figure 14-1)

			25 MHz		
Num.	**Characteristic**	**Symbol**	**Min**	**Max**	**Unit**
	System Frequency	fsys	dc1	25.00	MHz
	Crystal Frequency	f_{XTAL}	25	6000	kHz
	On-Chip VCO System Frequency	fsys	10	25	MHz
	Start-up Time With external clock (oscillator disabled) or after changing the multiplication factor MF. With external crystal oscillator enabled.	t_{pll} t_{osc}		2500 75,000	clks
	CLKO stability	ΔCLK	TBD	TBD	%
1	CLKO Period	t_{cyc}	40	—	ns
1A	EXTAL Duty Cycle, MF ≥ 5	t_{dcyc}	40	60	%
	EXTAL Duty Cycle, MF < 5, and user requires specification 5B.	t_{dcyc}	49	51	%
1C	External Clock Input Period	t_{EXTcyc}	40	—	ns
2, 3	CLKO1 Pulse Width (measured at 1.5v)	t_{CW1}	19	—	ns
4, 5	CLKO1 Rise and Fall Times (full drive)	t_{Crf1}	—	2	ns
5B	EXTAL to CLKO1 skew—PLL enabled (MF<5)	t_{EXTP1}		+/–4	ns

Note: The minimum VCO frequency and the PLL default values put some restrictions on the minimum system frequency.

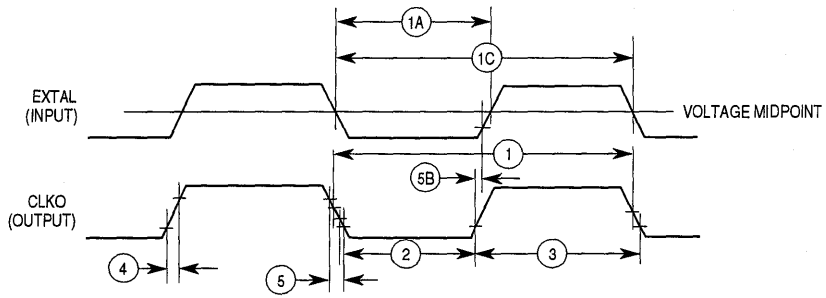


Figure 14-1. Clock Timing

14.5.2 AC Electrical Characteristics - IMP Phased Lock Loop (PLL) Characteristics

Characteristics	Expression	Min	Max	Unit
VCO frequency when PLL enabled	MF * Ef	10	f (Note 1.)	MHz
PLL external capacitor (XFC pin to VCCSYN)	MF * C_{XFC} (Note 1.) @ MF $\leq$ 5 @ MF > 5	 MF * 340 MF * 380	 MF * 480 MF * 970	pF

1. CXFC is the value of the PLL capacitor (connected between XFC pin and VCCSYN) for MF=1. The recommended value for C_{XFC} is 400pF for MF $\leq$ 5 and 540pF for MF> 5. The maximum VCO frequency is limited to the internal operation frequency, as defined above.

Examples:

1. MODCK1,0 = 01; MF = 1 $\Rightarrow$ 340 $\leq c_{XFC} \leq$ 480 pF

2. MODCK1,0 = 01; crystal is 32.768 KHz (or 4.192 MHz), initial MF = 401, initial frequency = 13.14 MHz; later, MF is changed to 762 to support a frequency of 25 MHz.

Minimum c_{XFC} is: 762 x 380 = 289 nF, maximum c_{XFC} is: 401 x 970 = 390 nF. The recommended c_{XFC} for 25 MHz is: 762 x 540 = 414 nF.

289 nF < c_{XFC} < 390 nF and closer to 414 nF. The proper available value for c_{XFC} is 390 nF.

3. MODCK1 pin = 1, crystal is 32.768 KHz (or 4.192 MHz), initial MF = 401, initial frequency = 13.14 MHz; later, MF is changed to 1017 to support a frequency of 33.34 MHz.

Minimum c_{XFC} is: 1017 x 380 = 386 nF. Maximum c_{XFC} is: 401 x 970 = 390 nF $\Rightarrow$ 386 nF < c_{XFC} < 390 nF.

The proper available value for c_{XFC} is 390 nF.

3A. In order to get higher range, higher crystal frequency can be used (i.e. 50 KHz), in this case:

Minimum c_{XFC} is: 667 x 380 = 253 nF. Maximum c_{XFC} is: 401 x 970 = 390 nF $\Rightarrow$ 253 nF < c_{XFC} < 390 nF.

14.5.3 IMP 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	I_{IL}	—	± 10	µA	$V_{in} = V_{ss}$
Input High Level Current	I_{IH}	—	± 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

14.5.4 AC Electrical Specifications—IMP Bus Master Cycles

(see Figure 14-2, Figure 14-3, Figure 14-4, and Figure 14-5))

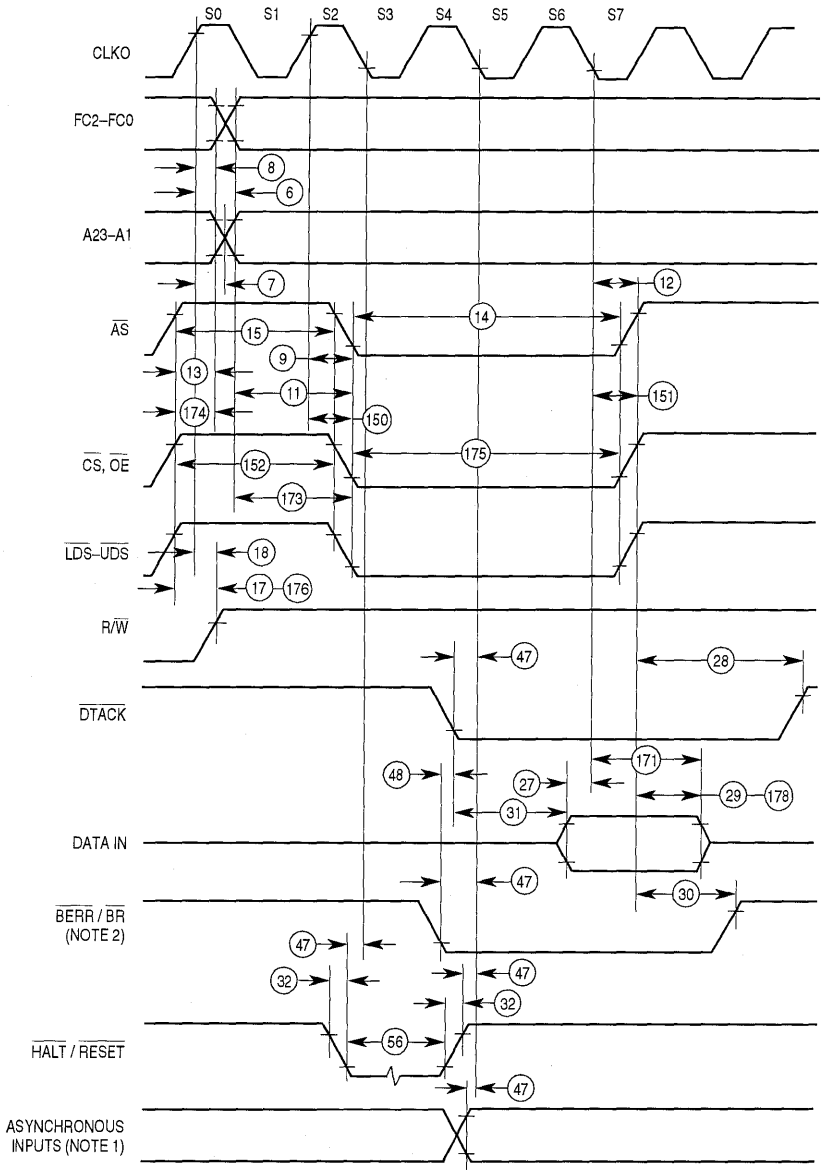
Num.	Characteristic	Symbol	25 MHz at 5.0 V.		Unit
			Min	Max	
6	Clock High to FC, Address Valid	$t_{CHFCADV}$	0	30	ns
7	Clock High to Address, Data Bus High Impedance (Maximum)	t_{CHADZ}	—	33	ns
8	Clock High to Address, FC Invalid (Minimum)	t_{CHAFI}	0	—	ns
9	Clock High to $\overline{AS}$, $\overline{DS}$ Asserted (see Note 1)	t_{CHSL}	3	20	ns
11	Address, FC Valid to $\overline{AS}$, $\overline{DS}$ Asserted (Read) $\overline{AS}$ Asserted Write (see Note 2)	t_{AFCVSL}	10	—	ns
12	Clock Low to $\overline{AS}$, $\overline{DS}$ Negated (see Note 1)	t_{CLSH}	—	20	ns
13	$\overline{AS}$, $\overline{DS}$ Negated to Address, FC Invalid (see Note 2)	t_{SHAFI}	10	—	ns
14	$\overline{AS}$ (and $\overline{DS}$ Read) Width Asserted (see Note 2)	t_{SL}	80	—	ns
14A	$\overline{DS}$ Width Asserted, Write (see Note 2)	t_{DSL}	40	—	ns
15	$\overline{AS}$, $\overline{DS}$ Width Negated (see Note 2)	t_{SH}	40	—	ns
16	Clock High to Control Bus High Impedance	t_{CHCZ}	—	33	ns
17	$\overline{AS}$, $\overline{DS}$ Negated to R/$\overline{W}$ Invalid (see Note 2)	t_{SHRH}	10	—	ns
18	Clock High to R/$\overline{W}$ High (see Note 1)	t_{CHRH}	—	20	ns
20	Clock High to R/$\overline{W}$ Low (see Note 1)	t_{CHRL}	—	20	ns
20A	$\overline{AS}$ Asserted to R/$\overline{W}$ Low (Write) (see Notes 2 and 6)	t_{ASRV}	—	7	ns
21	Address FC Valid to R/$\overline{W}$ Low (Write) (see Note 2)	t_{AFCVRL}	10	—	ns
22	R/$\overline{W}$ Low to $\overline{DS}$ Asserted (Write) (see Note 2)	t_{RLSL}	20	—	ns
23	Clock Low to Data-Out Valid	t_{CLDO}	—	20	ns
25	$\overline{AS}$, $\overline{DS}$, Negated to Data-Out Invalid (Write) (see Note 2)	t_{SHDOI}	10	—	ns
26	Data-Out Valid to $\overline{DS}$ Asserted (Write) (see Note 2)	t_{DOSL}	10	—	ns
27	Data-In Valid to Clock Low (Setup Time on Read) (see Note 5)	t_{DICL}	5	—	ns
28	$\overline{AS}$, $\overline{DS}$ Negated to $\overline{DTACK}$ Negated (Asynchronous Hold) (see Note 2)	t_{SHDAH}	0	75	ns
29	$\overline{AS}$, $\overline{DS}$ Negated to Data-In Invalid (Hold Time on Read)	t_{SHDII}	—	—	ns
30	$\overline{AS}$, $\overline{DS}$ Negated to $\overline{BERR}$ Negated	t_{SHBEH}	0	—	ns
31	$\overline{DTACK}$ Asserted to Data-In Valid (Setup Time) (see Notes 2 and 5)	t_{DALDI}	—	33	ns
32	$\overline{HALT}$ and $\overline{RESET}$ Input Transition Time	t_{RHr}, t_{RHf}	—	150	ns
33	Clock High to $\overline{BG}$ Asserted	t_{CHGL}	—	20	ns
34	Clock High to $\overline{BG}$ Negated	t_{CHGH}	—	20	ns
35	$\overline{BR}$ Asserted to $\overline{BG}$ Asserted (see Note 11)	t_{BRLGL}	2.5	4.5	clks
36	$\overline{BR}$ Negated to $\overline{BG}$ Negated (see Note 7)	t_{BRHGH}	1.5	2.5	clks

Num.	Characteristic	Symbol	25 MHz		Unit
			Min	Max	
37	$\overline{BGACK}$ Asserted to $\overline{BG}$ Negated	t_{GALGH}	2.5	4.5	clks
37A	$\overline{BGACK}$ Asserted to $\overline{BR}$ Negated (see Note 8)	t_{GALBRH}	10	1.5	ns/ clks
38	$\overline{BG}$ Asserted to Control, Address, Data Bus High Impedance ($\overline{AS}$ Negated)	t_{GLZ}	—	33	ns
39	$\overline{BG}$ Width Negated	t_{GH}	1.5	—	clks
40	$\overline{BGACK}$ Asserted to Address Valid	t_{GALAV}	15	—	ns
41	$\overline{BGACK}$ Asserted to $\overline{AS}$ Assserted	t_{GALASA}	20	—	ns
44	$\overline{AS}$, $\overline{DS}$ Negated to $\overline{AVEC}$ Negated	t_{SHVPH}	0	33	ns
46	$\overline{BGACK}$ Width Low	t_{GAL}	1.5	—	clks
47	Asynchronous Input Setup Time (see Note 5)	t_{ASI}	7	—	ns
48	$\overline{BERR}$ Asserted to $\overline{DTACK}$ Asserted (see Notes 2 and 3)	t_{BELDAL}	7	—	ns
53	Data-Out Hold from Clock High	t_{CHDOI}	0	—	ns
55	R/$\overline{W}$ Asserted to Data Bus Impedance Change	t_{RLDBD}	0	—	ns
56	$\overline{HALT}$/$\overline{RESET}$ Pulse Width (see Note 4)	t_{HRPW}	10	—	clks
57	$\overline{BGACK}$ Negated to $\overline{AS}$, $\overline{DS}$, R/$\overline{W}$ Driven	t_{GASD}	1.5	—	clks
57A	$\overline{BGACK}$ Negated to FC	t_{GAFD}	1	—	clks
58	$\overline{BR}$ Negated to $\overline{AS}$, $\overline{DS}$, R/$\overline{W}$ Driven (see Note 7)	t_{RHSD}	1.5	—	clks
58A	$\overline{BR}$ Negated to FC (see Note 7)	t_{RHFD}	1	—	clks
60	Clock High to $\overline{BCLR}$ Asserted	t_{CHBCL}	—	20	ns
61	Clock High to $\overline{BCLR}$ High Impedance (See Note 10)	t_{CHBCH}	—	20	ns
62	Clock Low (S0 Falling Edge during read) to $\overline{RMC}$ Asserted	t_{CLRML}	—	20	ns
63	Clock High (during write) to $\overline{RMC}$ Negated	t_{CHRMH}	—	20	ns
64	$\overline{RMC}$ Negated to $\overline{BG}$ Asserted (see Note 9)	t_{RMHGL}	—	20	ns

NOTES:

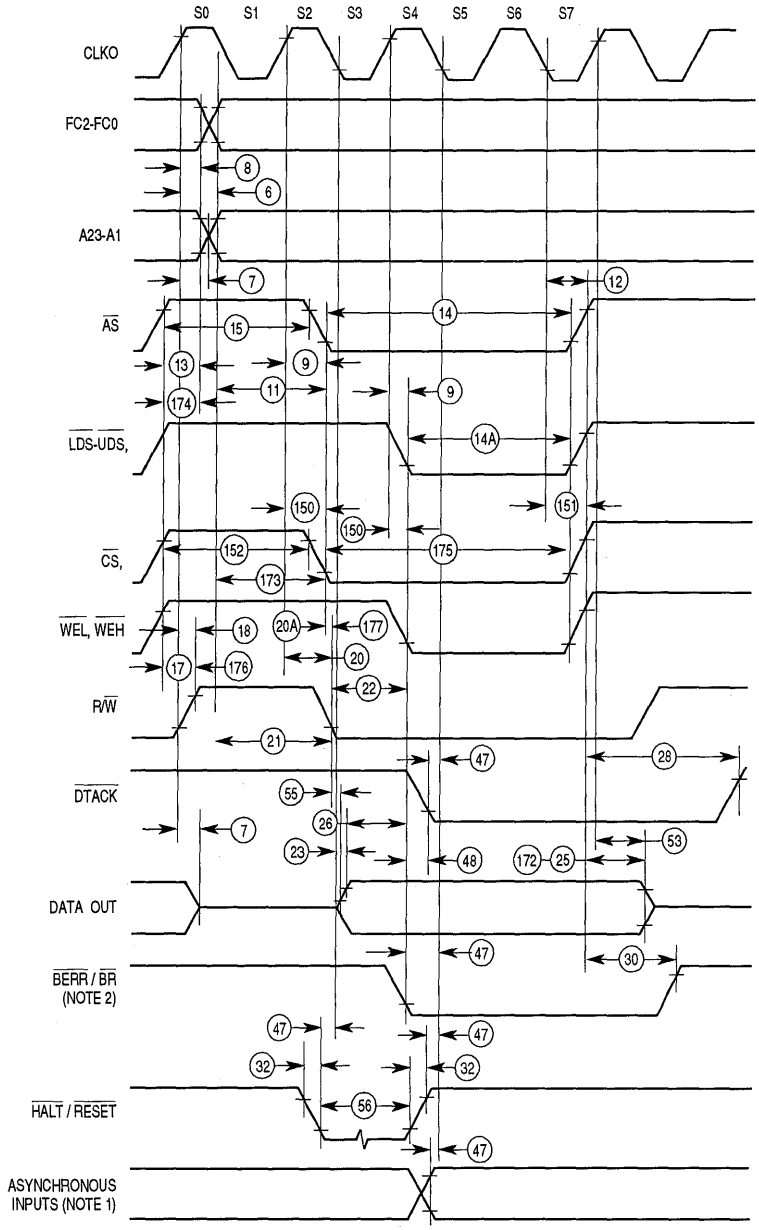
1. For loading capacitance of less than or equal to 50 pF, subtract 4 ns from the value given in the maximum columns.
2. Actual value depends on clock period since signals are driven/latched on different CLKO edges. To calculate the actual spec for other clock frequencies, the user may derive the formula 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., 40 ns with a 25-MHz device), and Sa is the actual spec in the data sheet. Thus, for spec 14 at 25 MHz:
   M = 5(40 ns/2) – 80 ns = 20 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) – 20 ns = 180 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 $\overline{\text{DTACK}}$ and $\overline{\text{BERR}}$, #48 may be ignored. In the absence of $\overline{\text{DTACK}}$, $\overline{\text{BERR}}$ is a synchronous input using the asynchronous input setup time (#47).
4. For power-up, the IMP must be held in the reset state for a minimum 100 ms to allow stabilization of on-chip circuit. This time could be longer to allow the PLL to lock (see 14.5.1 IMP AC Electrical Specifications Control Timing. 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 MC68356 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.



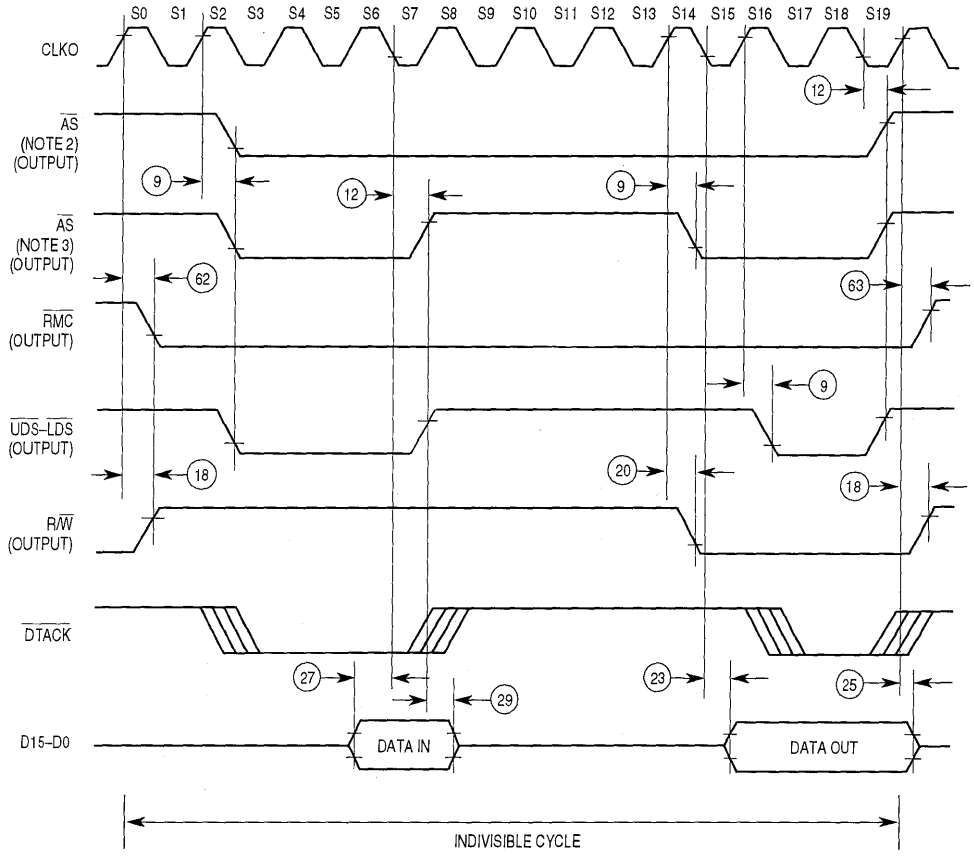
NOTES:
1. Setup time for the asynchronous inputs IPL2–IPL0 guarantees their recognition at the next falling edge of the clock.
2. BR need fall at this time only to insure being recognized at the end of the bus cycle.
3. Timing measurements are referenced to and from a low voltage of 0.8 volt and a high 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 14-2. 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 14-3. 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 14-4. Read-Modify-Write Cycle Timing Diagram

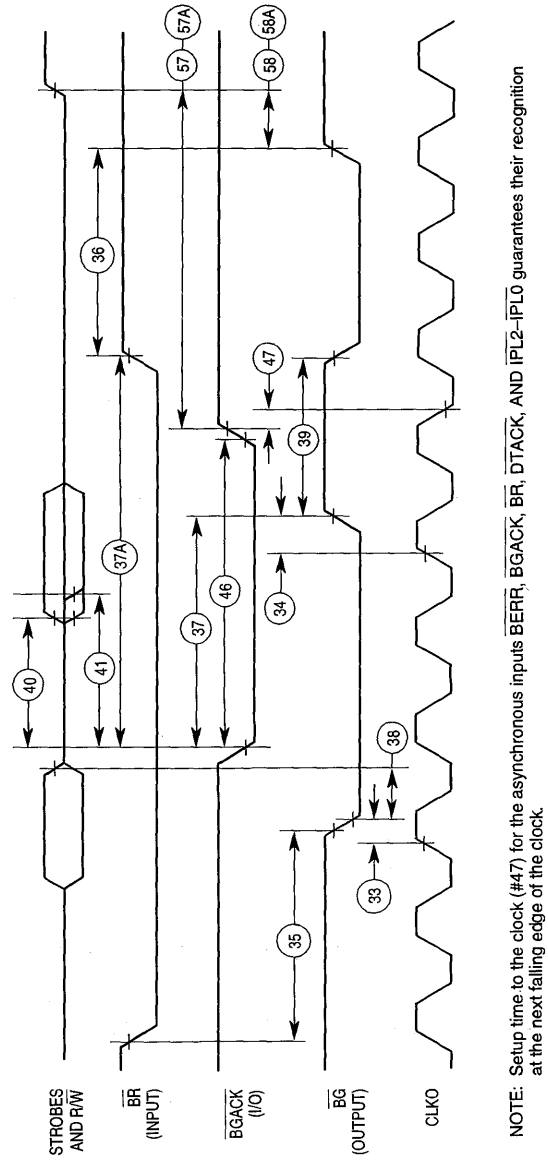
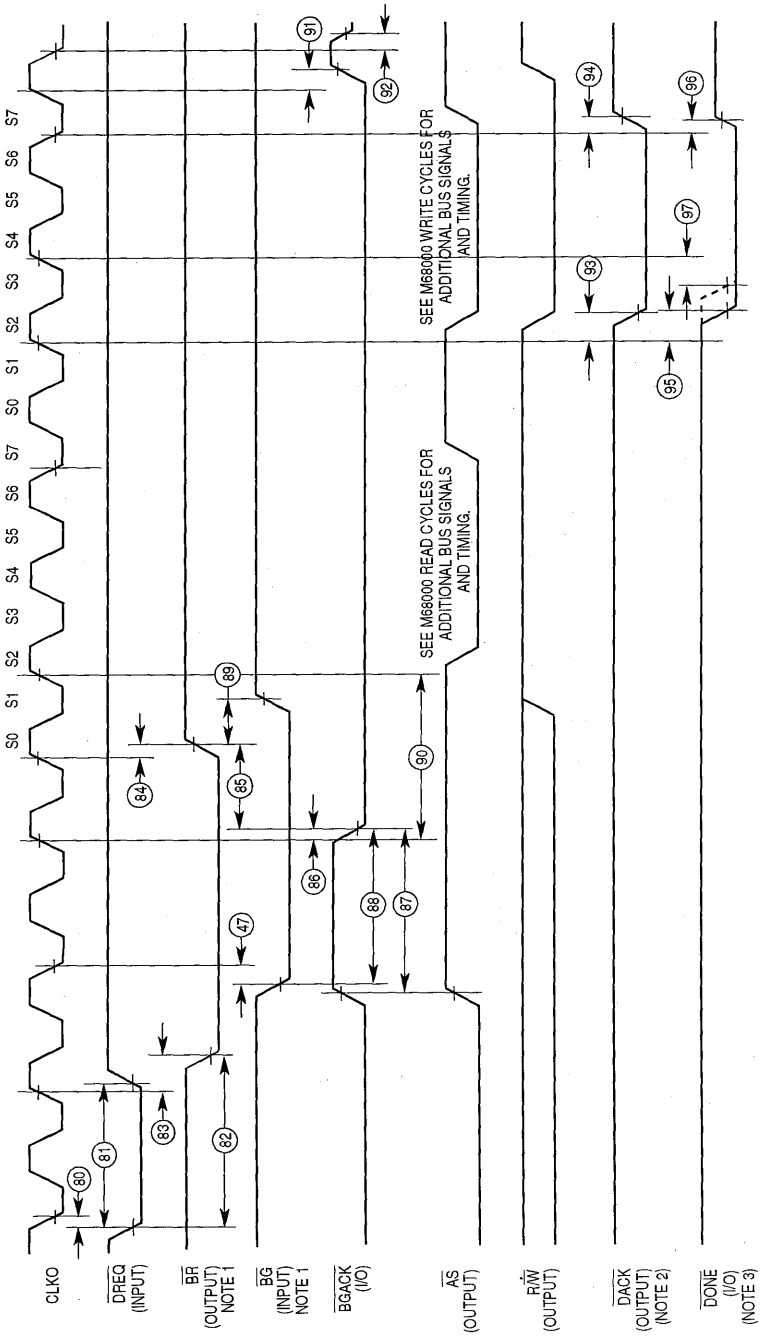


Figure 14-5. Bus Arbitration Timing Diagram

14.5.5 IMP AC Electrical Specifications—DMA (see Figure 14-6 and Figure 14-7))

Num.	Characteristic	Symbol	25 MHz at 5.0 V.		Unit
			Min	Max	
80	$\overline{REQ}$ Asynchronous Setup Time (see Note 1)	t_{REQASI}	10	—	ns
81	$\overline{REQ}$ Width Low (see Note 2)	t_{REQL}	2	—	clks
82	$\overline{REQ}$ Low to $\overline{BR}$ Low (see Notes 3 and 4)	$t_{REQLBRL}$	—	2	clks
83	Clock High to $\overline{BR}$ Low (see Notes 3 and 4)	t_{CHBRL}	—	20	ns
84	Clock High to $\overline{BR}$ High Impedance (see Notes 3 and 4)	t_{CHBRZ}	—	20	ns
85	$\overline{BGACK}$ Low to $\overline{BR}$ High Impedance (see Notes 3 and 4)	t_{BKLBRZ}	20	—	ns
86	Clock High to $\overline{BGACK}$ Low	t_{CHBKL}	—	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 +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 +20	clks ns
89	$\overline{BR}$ High Impedance to $\overline{BG}$ High (see Notes 3 and 4)	t_{BRHBGH}	0	—	ns
90	Clock on which $\overline{BGACK}$ Low to Clock on which $\overline{AS}$ Low	$t_{CLBKLAL}$	2	2	clks
91	Clock High to $\overline{BGACK}$ High	t_{CHBKH}	—	20	ns
92	Clock Low to $\overline{BGACK}$ High Impedance	t_{CLBKZ}	—	10	ns
93	Clock High to $\overline{DACK}$ Low	t_{CHACKL}	—	20	ns
94	Clock Low to $\overline{DACK}$ High	t_{CLACKH}	—	20	ns
95	Clock High to $\overline{DONE}$ Low (Output)	t_{CHDNL}	—	20	ns
96	Clock Low to $\overline{DONE}$ High Impedance	t_{CLDNZ}	—	20	ns
97	$\overline{DONE}$ Input Low to Clock High (Asynchronous Setup)	t_{DNLTCH}	10	—	ns

NOTES:
1. $\overline{DREQ}$ is sampled on the falling edge of $\overline{CLK}$ 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 $\overline{AS}$, $\overline{HALT}$, 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 14-6. DMA Timing Diagram (IDMA)

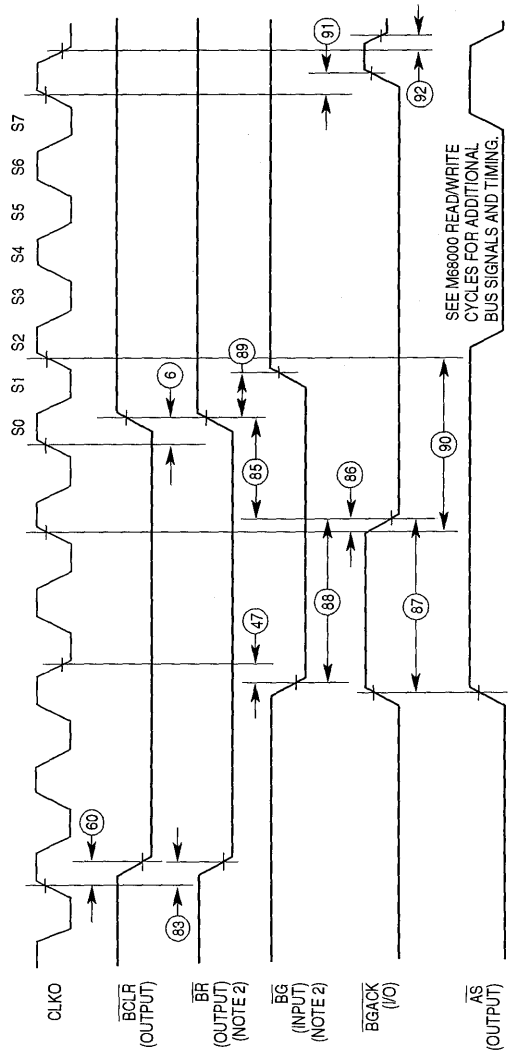


Figure 14-7. DMA Timing Diagram (SDMA)

14.5.6 IMP AC Electrical Specifications—External Master Internal Asynchronous Read/Write Cycles

(see Figure 14-8 and Figure 14-9)

Num.	Characteristic	Symbol	25 MHz at 5.0 V.		Unit
			Min	Max	
100	R/$\overline{W}$ Valid to $\overline{DS}$ Low	t_{RWVDSL}	0	—	ns
101	$\overline{DS}$ Low to Data-In Valid	t_{DSLDIV}	—	20	ns
102	$\overline{DTACK}$ Low to Data-In Hold Time	t_{DKLDH}	0	—	ns
103	$\overline{AS}$ Valid to $\overline{DS}$ Low	t_{ASVDSL}	0	—	ns
104	$\overline{DTACK}$ Low to $\overline{AS}$, $\overline{DS}$ High	t_{DKLDSH}	0	—	ns
105	$\overline{DS}$ High to $\overline{DTACK}$ High	t_{DSHDKH}	—	30	ns
106	$\overline{DS}$ Inactive to $\overline{AS}$ Inactive	t_{DSIASI}	0	—	ns
107	$\overline{DS}$ High to R/$\overline{W}$ High	t_{DSHRWH}	0	—	ns
108	$\overline{DS}$ High to Data High Impedance	t_{DSHDZ}	—	30	ns
108A	$\overline{DS}$ High to Data-Out Hold Time (see Note)	t_{DSHDH}	0	—	ns
109A	Data Out Valid to $\overline{DTACK}$ Low	t_{DOVDKL}	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.

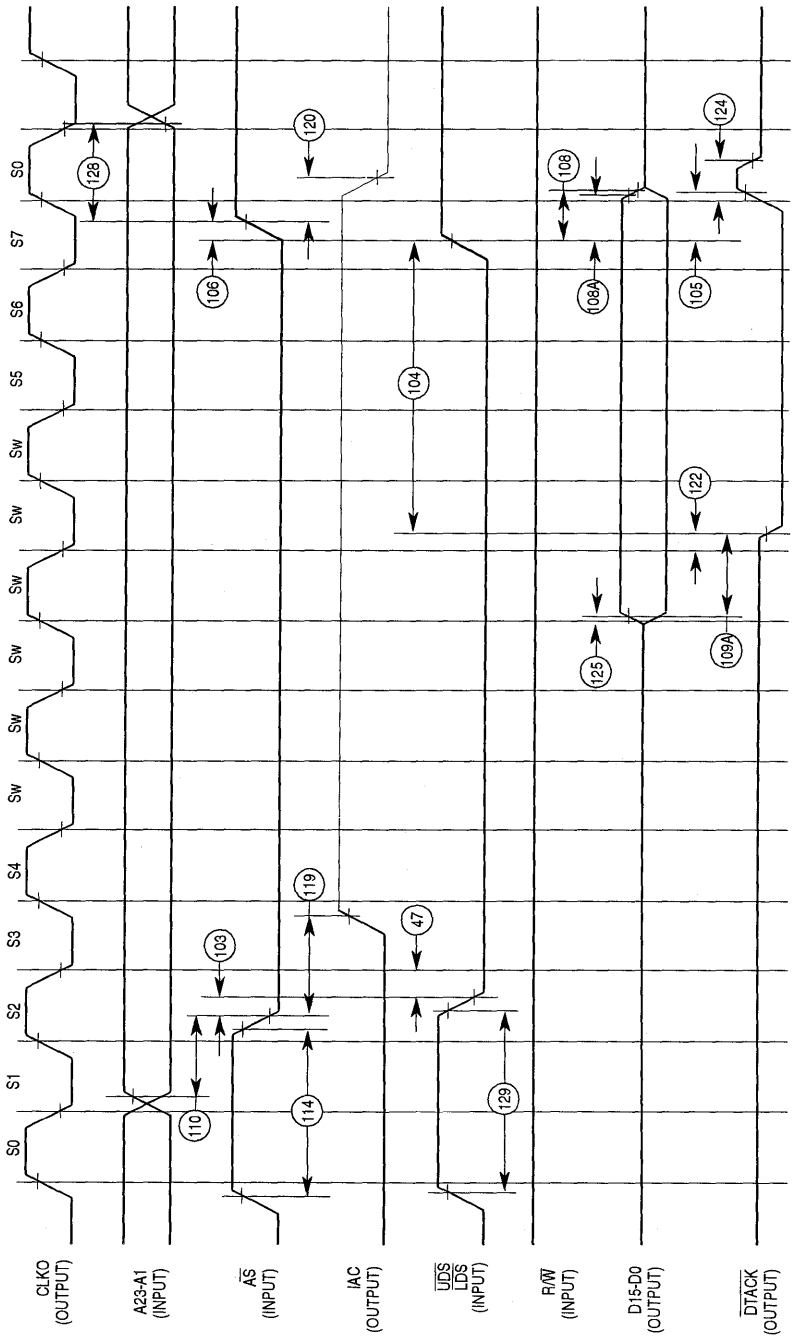


Figure 14-8. External Master Internal Asynchronous Read Cycle Timing Diagram

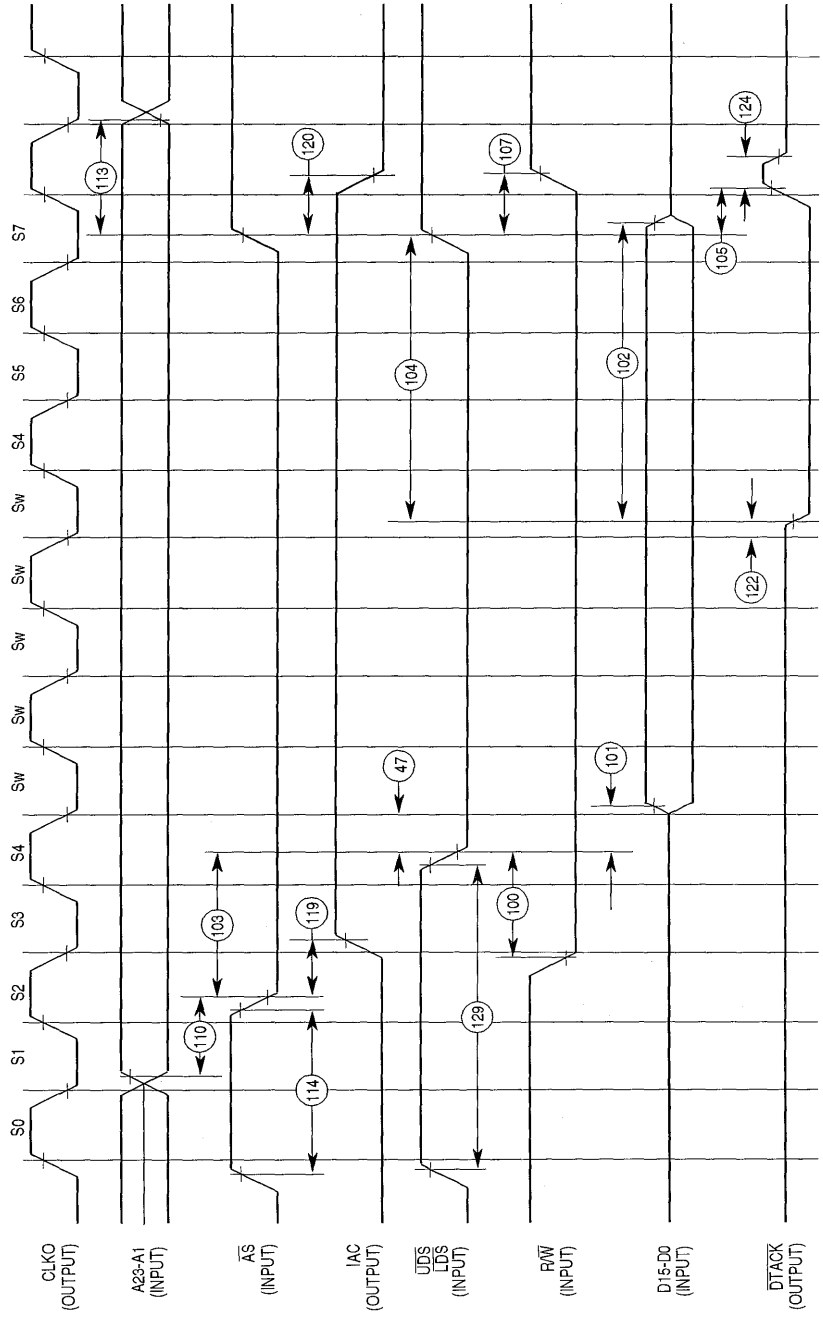


Figure 14-9. External Master Internal Asynchronous Write Cycle Timing Diagram

14.5.7 IMP AC Electrical Specifications—External Master Internal Synchronous Read/Write Cycles

(see Figure 14-10, Figure 14-11, and Figure 14-12)

Num.	Characteristic	Symbol	25 MHz at 5.0 V.		Unit
			Min	Max	
110	Address Valid to $\overline{AS}$ Low	t_{AVASL}	10	—	ns
111	$\overline{AS}$ Low to Clock High	t_{ASLCH}	20	—	ns
112	Clock Low to $\overline{AS}$ High	t_{CLASH}	—	30	ns
113	$\overline{AS}$ High to Address Hold Time on Write	t_{ASHAH}	0	—	ns
114	$\overline{AS}$ Inactive Time	t_{ASH}	1	—	clk
115	$\overline{UDS}/\overline{LDS}$ Low to Clock High (see Note 2)	t_{SLCH}	27	—	ns
116	Clock Low to $\overline{UDS}/\overline{LDS}$ High	t_{CLSH}	—	30	ns
117	R/$\overline{W}$ Valid to Clock High (see Note 2)	t_{RWVCH}	20	—	ns
118	Clock High to R/$\overline{W}$ High	t_{CHRWH}	—	30	ns
119	$\overline{AS}$ Low to IAC High	t_{ASLIAH}	—	27	ns
120	$\overline{AS}$ High to IAC Low	t_{ASHIAL}	—	27	ns
121	$\overline{AS}$ Low to $\overline{DTACK}$ Low (0 Wait State)	t_{ASLDTL}	—	30	ns
122	Clock Low to $\overline{DTACK}$ Low (1 Wait State)	t_{CLDTL}	—	20	ns
123	$\overline{AS}$ High to $\overline{DTACK}$ High	t_{ASHDTH}	—	30	ns
124	$\overline{DTACK}$ High to $\overline{DTACK}$ High Impedance	t_{DTHDTZ}	—	10	ns
125	Clock High to Data-Out Valid	t_{CHDOV}	—	20	ns
126	$\overline{AS}$ High to Data High Impedance	t_{ASHDZ}	—	30	ns
127	$\overline{AS}$ High to Data-Out Hold Time	t_{ASHDOI}	0	—	ns
128	$\overline{AS}$ High to Address Hold Time on Read	t_{ASHAI}	0	—	ns
129	$\overline{UDS}/\overline{LDS}$ Inactive Time	t_{SH}	1	—	clk
130	Data-In Valid to Clock Low	t_{CLDIV}	20	—	ns
131	Clock Low to Data-In Hold Time	t_{CLDIH}	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.

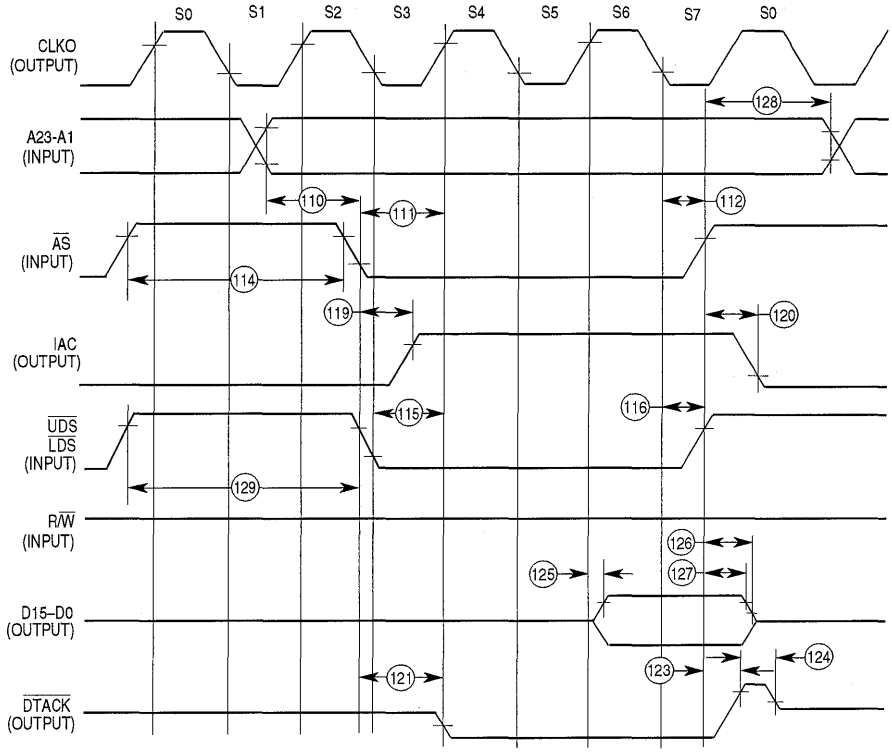


Figure 14-10. External Master Internal Synchronous Read Cycle Timing Diagram

14

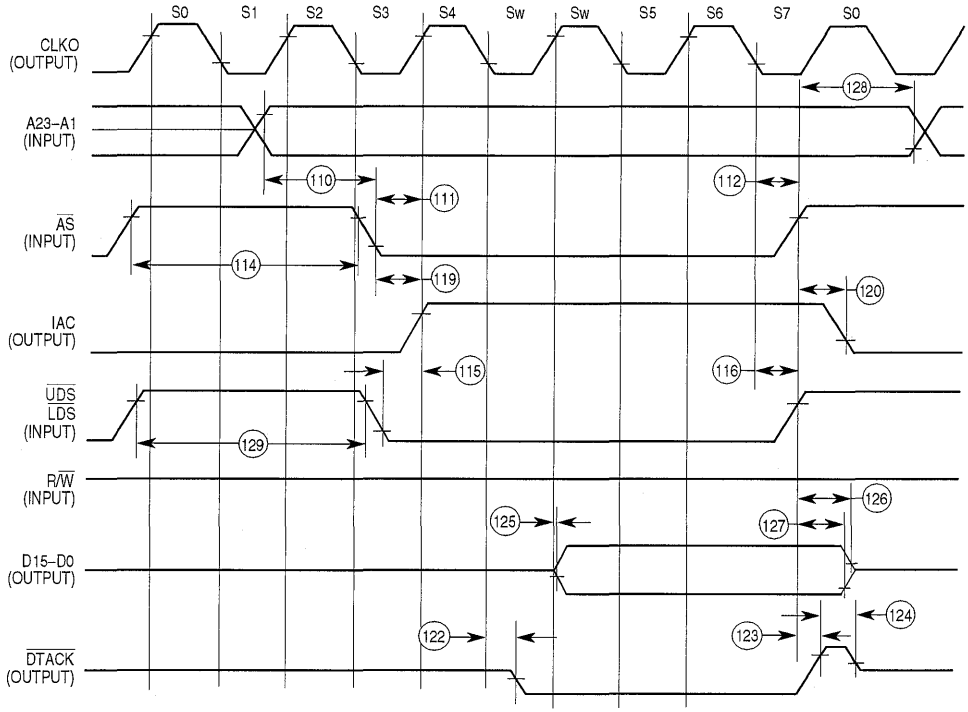


Figure 14-11. External Master Internal Synchronous Read Cycle Timing Diagram (One Wait State)

14

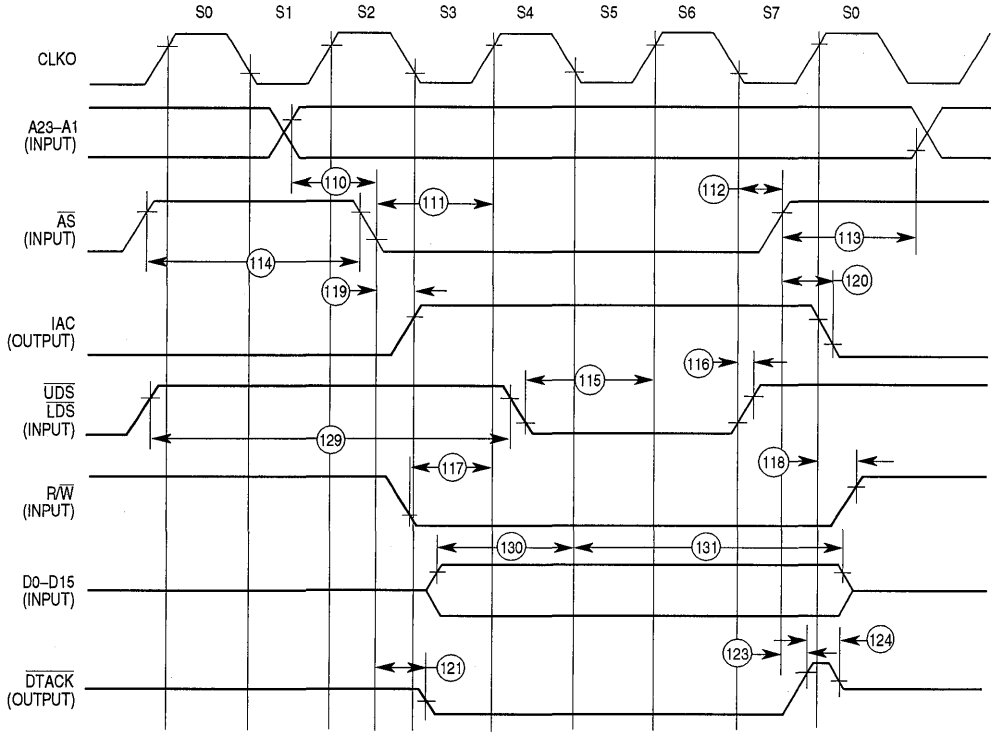


Figure 14-12. External Master Internal Synchronous Write Cycle Timing Diagram

14.5.8 IMP AC Electrical Specifications—Internal Master Internal Read/Write Cycles (see Figure 14-13)

Num.	Characteristic	Symbol	25 MHz at 5.0 V.		Unit
			Min	Max	
140	Clock High to IAC High	t_{CHIAH}	—	27	ns
141	Clock Low to IAC Low	t_{CLIAL}	—	27	ns
142	Clock High to $\overline{DTACK}$ Low	t_{CHDTL}	—	30	ns
143	Clock Low to $\overline{DTACK}$ High	t_{CLDTH}	—	27	ns
144	Clock High to Data-Out Valid	t_{CHDOV}	—	20	ns
145	$\overline{AS}$ High to Data-Out Hold Time	t_{ASHDOH}	0	—	ns

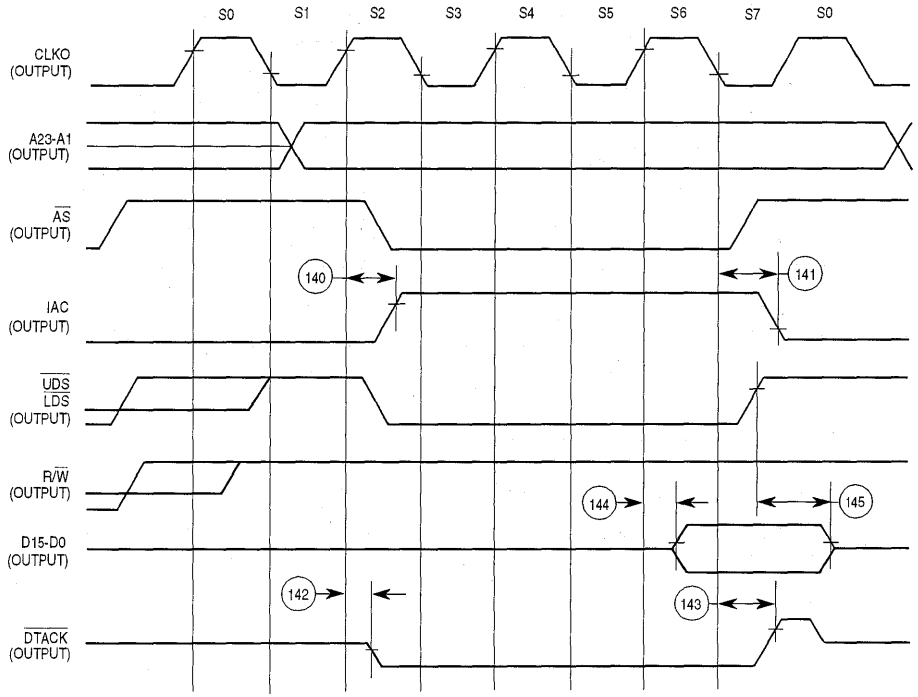


Figure 14-13. Internal Master Internal Read Cycle Timing Diagram

14.5.9 IMP AC Electrical Specifications—Chip-Select Timing Internal Master (see Figure 14-14)

Num.	Characteristic	Symbol	25 MHz at 5.0 V.		Unit
			Min	Max	
150	Clock High to $\overline{CS}$, $\overline{IACK}$, $\overline{OE}$, $\overline{WEL}$, $\overline{WEH}$ Low (see Note 2)	$t_{CHCSIAKL}$	0	27	ns
151	Clock Low to $\overline{CS}$, $\overline{IACK}$, $\overline{OE}$, $\overline{WEL}$, $\overline{WEH}$ High (see Note 2)	$t_{CLCSIAKH}$	0	27	ns
152	$\overline{CS}$ Width Negated	t_{CSH}	40	—	ns
153	Clock High to $\overline{DTACK}$ Low (0 Wait State)	t_{CHDTKL}	—	30	ns
154	Clock Low to $\overline{DTACK}$ Low (1–6 Wait States)	t_{CLDTKL}	—	20	ns
155	Clock Low to $\overline{DTACK}$ High	t_{CLDTKH}	—	27	ns
156	Clock High to $\overline{BERR}$ Low (see Note 1)	t_{CHBERL}	—	27	ns
157	Clock Low to $\overline{BERR}$ High Impedance (see Note 1)	t_{CLBERH}	—	27	ns
158	$\overline{DTACK}$ High to $\overline{DTACK}$ High Impedance	$t_{DTKHDTKZ}$	—	10	ns
171	Input Data Hold Time from S6 Low	t_{IDHCL}	5	—	ns
172	$\overline{CS}$ Negated to Data-Out Invalid (Write)	t_{CSNDOI}	7	—	ns
173	Address, FC Valid to $\overline{CS}$ Asserted	t_{AFVCSA}	15	—	ns
174	$\overline{CS}$ Negated to Address, FC Invalid	t_{CSNAFI}	12	—	ns
175	$\overline{CS}$ Low Time (0 Wait States)	t_{CSLT}	80	—	ns
176	$\overline{CS}$ Negated to R/$\overline{W}$ Invalid	t_{CSNRWI}	7	—	ns
177	$\overline{CS}$ Asserted to R/$\overline{W}$ Low (Write)	t_{CSARWL}	—	8	ns
178	$\overline{CS}$ Negated to Data-In Invalid (Hold Time on Read)	t_{CSNDII}	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 14-2 and Figure 14-3 for diagrams.

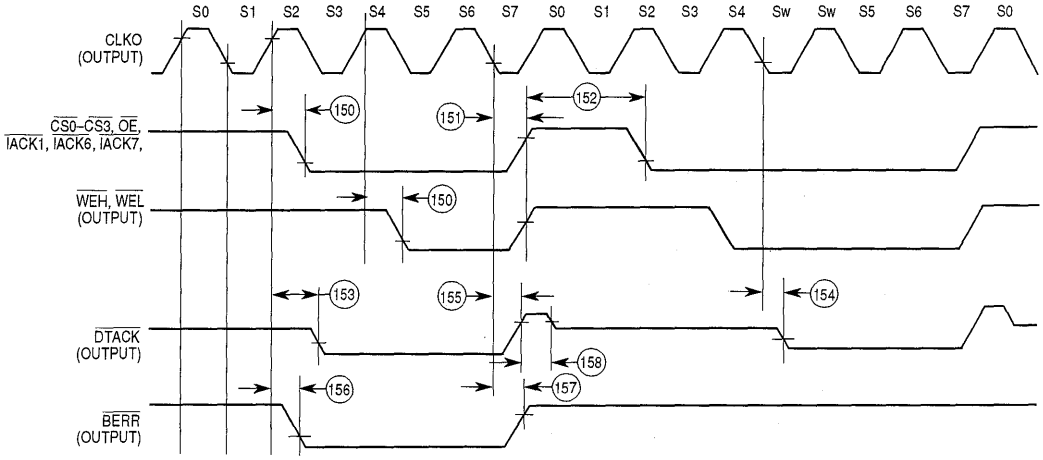


Figure 14-14. Internal Master Chip-Select Timing Diagram

14.5.10 IMP AC Electrical Specifications—Chip-Select Timing External Master (see Figure 14-15)

Num.	Characteristic	Symbol	25 MHz at 5.0 V.		Unit
			Min	Max	
154	Clock Low to $\overline{DTACK}$ Low (1–6 Wait States)	t_{CLDTKL}	—	20	ns
160	$\overline{AS}$ Low to $\overline{CS}$ Low	t_{ASLCSL}	—	20	ns
161	$\overline{AS}$ High to $\overline{CS}$ High	t_{ASHCSH}	—	20	ns
162	Address Valid to $\overline{AS}$ Low	t_{AVASL}	10	—	ns
163	R/$\overline{W}$ Valid to $\overline{AS}$ Low (see Note 1)	t_{RWVASL}	10	—	ns
164	$\overline{AS}$ Negated to Address Hold Time	t_{ASHAI}	0	—	ns
165	$\overline{AS}$ Low to $\overline{DTACK}$ Low (0 Wait State)	$t_{ASLDTKL}$	—	30	ns
167	$\overline{AS}$ High to $\overline{DTACK}$ High	$t_{ASHDTKH}$	—	20	ns
168	$\overline{AS}$ Low to $\overline{BERR}$ Low (see Note 2)	$t_{ASLBERL}$	—	20	ns
169	$\overline{AS}$ High to $\overline{BERR}$ High Impedance (see Notes 2 and 3)	$t_{ASHBERH}$	—	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.

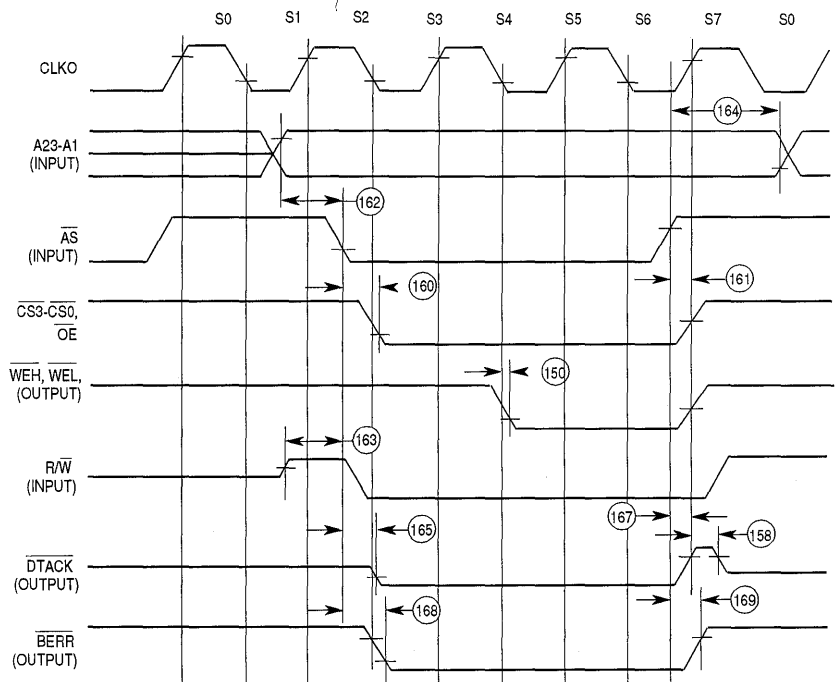


Figure 14-15. External Master Chip-Select Timing Diagram

14.5.11 IMP AC Electrical Specifications—Parallel I/O (see Figure 14-16)

Num.	Characteristic	Symbol	25 MHz at 5.0 V.		Unit
			Min	Max	
180	Input Data Setup Time (to Clock Low)	t_{DSU}	14	—	ns
181	Input Data Hold Time (from Clock Low)	t_{DH}	19	—	ns
182	Clock High to Data-Out Valid (CPU Writes Data, Control, or Direction)	t_{CHDOV}	—	24	ns

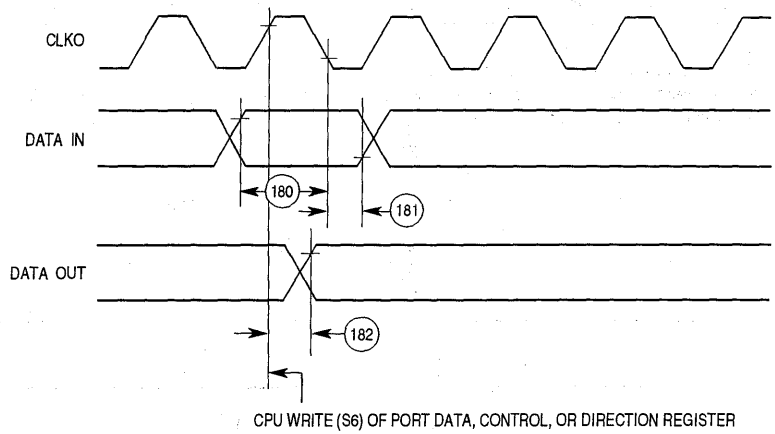


Figure 14-16. Parallel I/O Data-In/Data-Out Timing Diagram

14.5.12 IMP AC Electrical Specifications—Interrupts (see Figure 14-17)

Num.	Characteristic	Symbol	25 MHz at 5.0 V.		Unit
			Min	Max	
190	Interrupt Pulse Width Low $\overline{IRQ}$ (Edge Triggered Mode)	t_{IPW}	34	—	ns
191	Minimum Time Between Active Edges	t_{AEMT}	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.

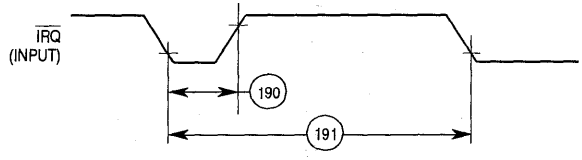


Figure 14-17. Interrupts Timing Diagram

14.5.13 IMP AC Electrical Specifications—Timers (see Figure 14-18)

Num.	Characteristic	Symbol	25 MHz at 5.0 V.		Unit
			Min	Max	
200	Timer Input Capture Pulse Width	t_{TPW}	34	—	ns
201	TIN Clock Low Pulse Width	t_{TICLT}	34	—	ns
202	TIN Clock High Pulse Width and Input Capture High Pulse Width	t_{TICHT}	2	—	clk
203	TIN Clock Cycle Time	t_{cyc}	3	—	clk
204	Clock High to TOUT Valid	t_{CHTOV}	—	24	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.

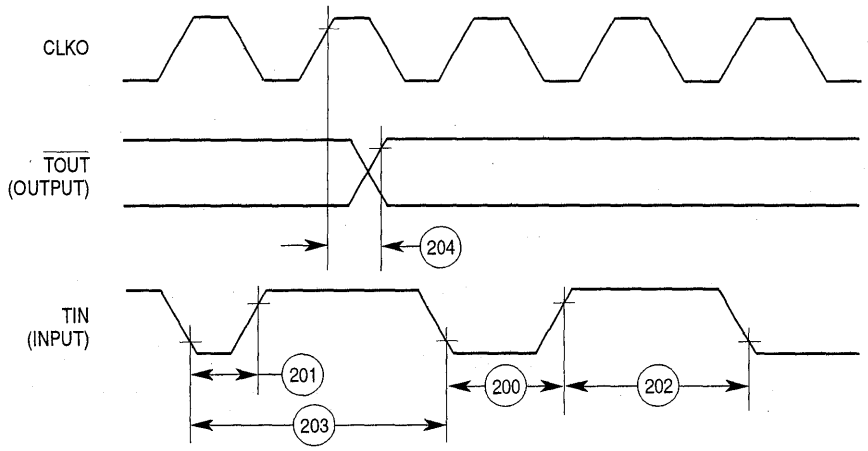


Figure 14-18. Timers Timing Diagram

14.5.14 IMP AC Electrical Specifications—Serial Communications Port

(see Figure 14-19).

Num.	Characteristic	25 MHz at 5.0 V		Unit
		Min	Max	
250	SPCLK Clock Output Period	4	64	clks
251	SPCLK Clock Output Rise/Fall Time	0	8	ns
252	Delay from SPCLK to Transmit (see Note 1)	0	24	ns
253	SCP Receive Setup Time (see Note 1)	24	—	ns
254	SCP Receive Hold Time (see Note 1)	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.

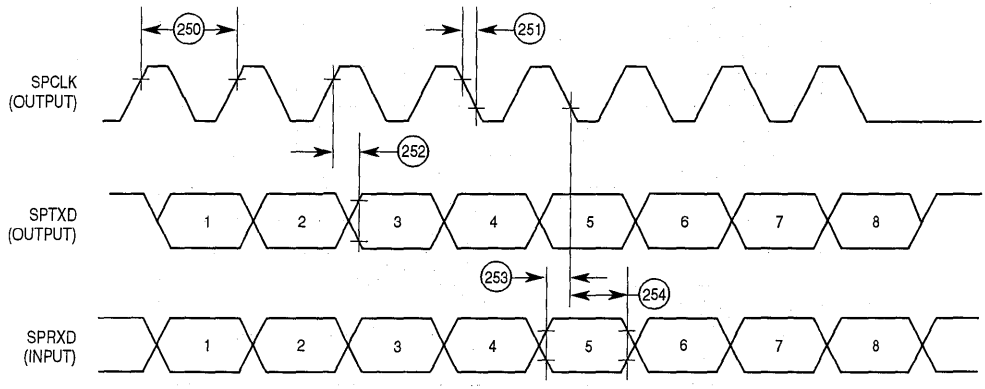


Figure 14-19. Serial Communication Port Timing Diagram

14.5.15 IMP AC Electrical Specifications—IDL Timing (All timing measurements, unless otherwise specified, are referenced to the L1CLK at 50% point of V_{DD}) (see Figure 14-20)

Num.	Characteristic	25 MHz at 5.0 V		Unit
		Min	Max	
260	L1CLK (IDL Clock) Frequency (see Note 1)	—	10	MHz
261	L1CLK Width Low	37	—	ns
262	L1CLK Width High (see Note 3)	P+10	—	ns
263	L1TXD, L1RQ, SDS1–SDS2 Rising/Falling Time	—	14	ns
264	L1SY1 (sync) Setup Time (to L1CLK Falling Edge)	20	—	ns
265	L1SY1 (sync) Hold Time (from L1CLK Falling Edge)	34	—	ns
266	L1SY1 (sync) Inactive Before 4th L1CLK	0	—	ns
267	L1TxD Active Delay (from L1CLK Rising Edge)	0	50	ns
268	L1TxD to High Impedance (from L1CLK Rising Edge) (see Note 2)	0	34	ns
269	L1RxD Setup Time (to L1CLK Falling Edge)	34	—	ns
270	L1RxD Hold Time (from L1CLK Falling Edge)	34	—	ns
271	Time Between Successive IDL syncs	20	—	L1CLK
272	L1RQ Valid before Falling Edge of L1SY1	1	—	L1CLK
273	L1GR Setup Time (to L1SY1 Falling Edge)	34	—	ns
274	L1GR Hold Time (from L1SY1 Falling Edge)	34	—	ns
275	SDS1–SDS2 Active Delay from L1CLK Rising Edge	7	50	ns
276	SDS1–SDS2 Inactive Delay from L1CLK Falling Edge	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 with 130 pF.
3. Where P = 1/CLKO. Thus, for a 25-MHz CLKO rate, P = 40 ns.

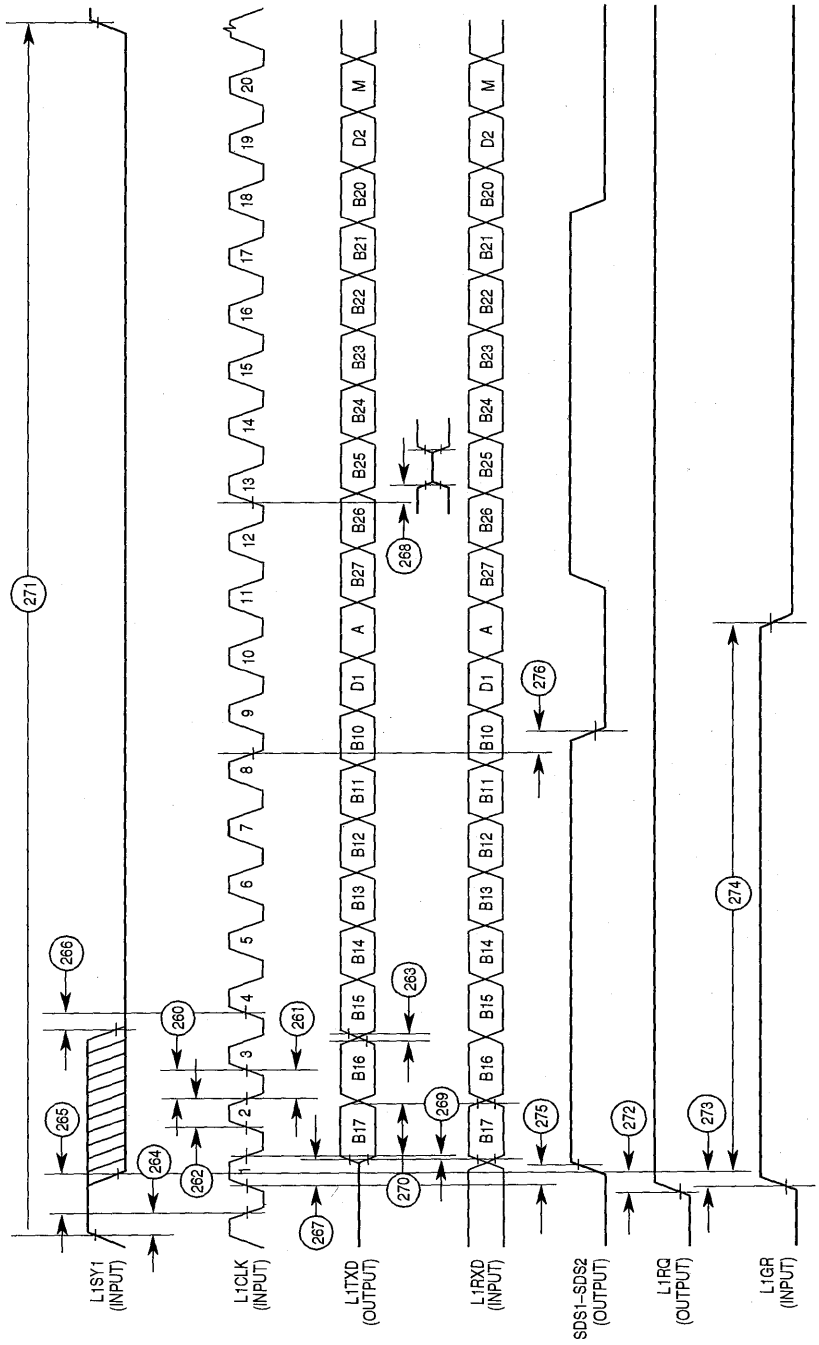


Figure 14-20. IDL Timing Diagram

14.5.16 IMP 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 14-21).

Num.	Characteristic	25 MHz at 5.0 V		Unit
		Min	Max	
	L1CLK GCI Clock Frequency (Normal Mode) (see Note 1)	—	512	kHz
280	L1CLK Clock Period Normal Mode (see Note 1)	1800	2100	ns
281	L1CLK Width Low/High Normal Mode	840	1450	ns
282	L1CLK Rise/Fall Time Normal Mode (see Note 4)	—	—	ns
	L1CLK (GCI Clock) Period (MUX Mode) (see Note 1)	—	6.668	MHz
280	L1CLK Clock Period MUX Mode (see Note 1)	150	—	ns
281	L1CLK Width Low MUX Mode	55	—	ns
281A	L1CLK Width High MUX Mode (see Note 5)	P+10	—	ns
282	L1CLK Rise/Fall Time MUX Mode (see Note 4)	—	—	ns
283	L1SY1 Sync Setup Time to L1CLK Falling Edge	20	—	ns
284	L1SY1 Sync Hold Time from L1CLK Falling Edge	34	—	ns
285	L1TxD Active Delay (from L1CLK Rising Edge) (see Note 2)	0	70	ns
286	L1TxD Active Delay (from L1SY1 Rising Edge) (see Note 2)	0	70	ns
287	L1RxD Setup Time to L1CLK Rising Edge	14	—	ns
288	L1RxD Hold Time from L1CLK Rising Edge	34	—	ns
289	Time Between Successive L1SY1in Normal SCIT Mode	64 192	— —	L1CLK L1CLK
290	SDS1–SDS2 Active Delay from L1CLK Rising Edge (see Note 3)	7	60	ns
291	SDS1–SDS2 Active Delay from L1SY1 Rising Edge (see Note 3)	7	60	ns
292	SDS1–SDS2 Inactive Delay from L1CLK Falling Edge	7	60	ns
293	GCIDCL (GCI Data Clock) Active Delay	0	34	ns

14

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 25-MHz CLKO rate, P = 40 ns.

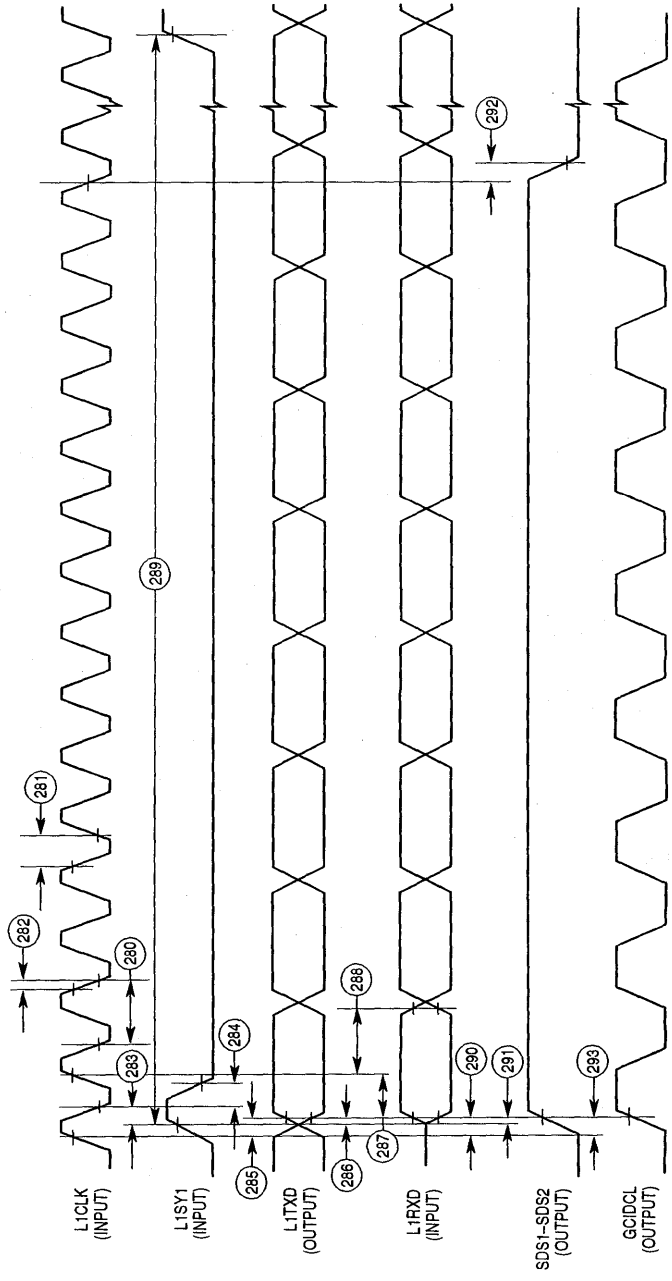


Figure 14-21. GCI Timing Diagram

14.5.17 IMP 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 14-22 and Figure 14-23).

Num.	Characteristic	25 MHz at 5.0 V		Unit
		Min	Max	
300	L1CLK (PCM Clock) Frequency (see Note 1)	—	10.0	MHz
301	L1CLK Width Low	37	—	ns
301A	L1CLK Width High (see Note 4)	P+10	—	ns
302	L1SY0–L1SY1 Setup Time to L1CLK Rising Edge	0	—	ns
303	L1SY0–L1SY1 Hold Time from L1CLK Falling Edge	27	—	ns
304	L1SY0–L1SY1 Width Low	1	—	L1CLK
305	Time Between Successive Sync Signals (Short Frame)	8	—	L1CLK
306	L1TxD Data Valid after L1CLK Rising Edge (see Note 2)	0	47	ns
307	L1TxD to High Impedance (from L1CLK Rising Edge)	0	34	ns
308	L1RxD Setup Time (to L1CLK Falling Edge) (see Note 3)	14	—	ns
309	L1RxD Hold Time (from L1CLK Falling Edge) (see Note 3)	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.
3. Specification valid for both sync methods.
4. Where P = 1/CLKO. Thus, for a 25-MHz CLKO rate, P =40 ns.

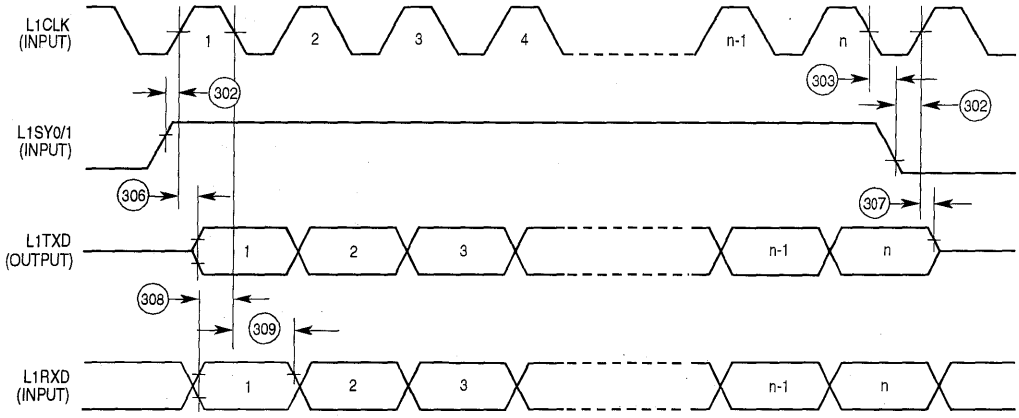
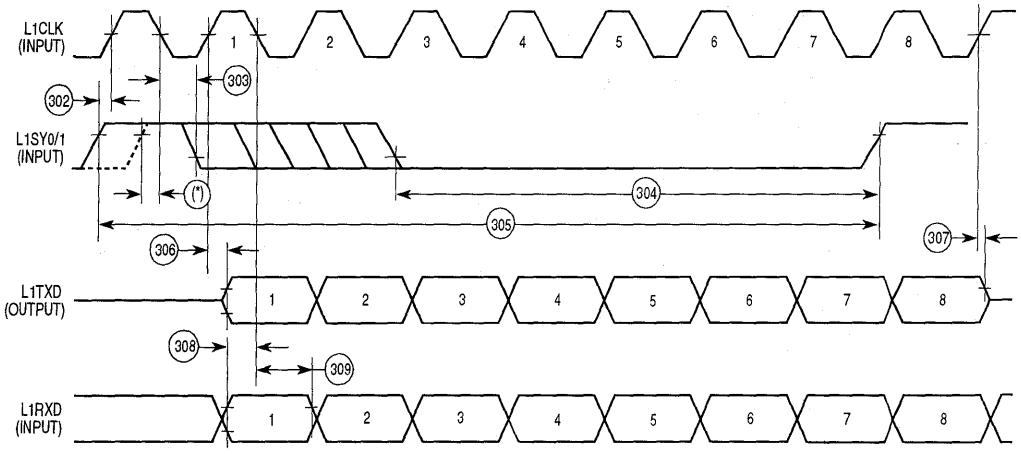


Figure 14-22. PCM Timing Diagram (SYNC Envelopes Data)



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 14-23. PCM Timing Diagram (SYNC Prior to 8-Bit Data)

14.5.18 IMP 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 the 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 14-24).

Num.	Characteristic	25 MHz at 5.0 V		25 MHz at 5.0 V		Unit
		Internal Clock		External Clock		
		Min	Max	Min	Max	
315	RCLK1 and TCLK1 Frequency (see Note 1)	—	8.33	—	10	MHz
316	RCLK1 and TCLK1 Low (see Note 4)	45	—	P+10	—	ns
316a	RCLK1 and TCLK1 High	45	—	35	—	ns
317	RCLK1 and TCLK1 Rise/Fall Time (see Note 3)	—	14	—	—	ns
318	TXD1 Active Delay from TCLK1 Falling Edge	0	25	0	40	ns
319	$\overline{RTS1}$ Active/Inactive Delay from TCLK1 Falling Edge	0	25	0	65	ns
320	$\overline{CTS1}$ Setup Time to TCLK1 Rising Edge	35	—	7	—	ns
321	RXD1 Setup Time to RCLK1 Rising Edge	35	—	7	—	ns
322	RXD1 Hold Time from RCLK1 Rising Edge (see Note 2)	7	—	35	—	ns
323	$\overline{CD1}$ Setup Time to RCLK1 Rising Edge	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 $\overline{CD}$ hold time when $\overline{CD}$ 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 25-MHz CLKO rate, P =40 ns.

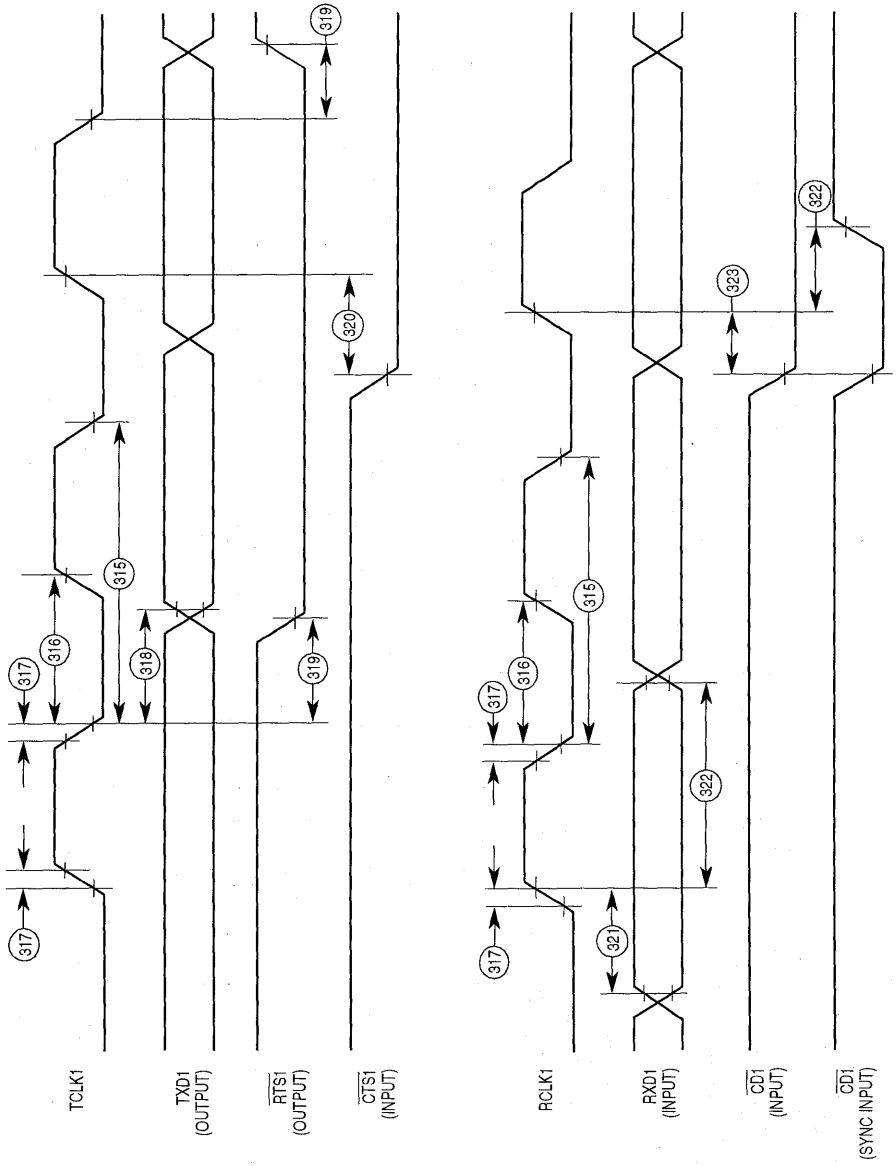


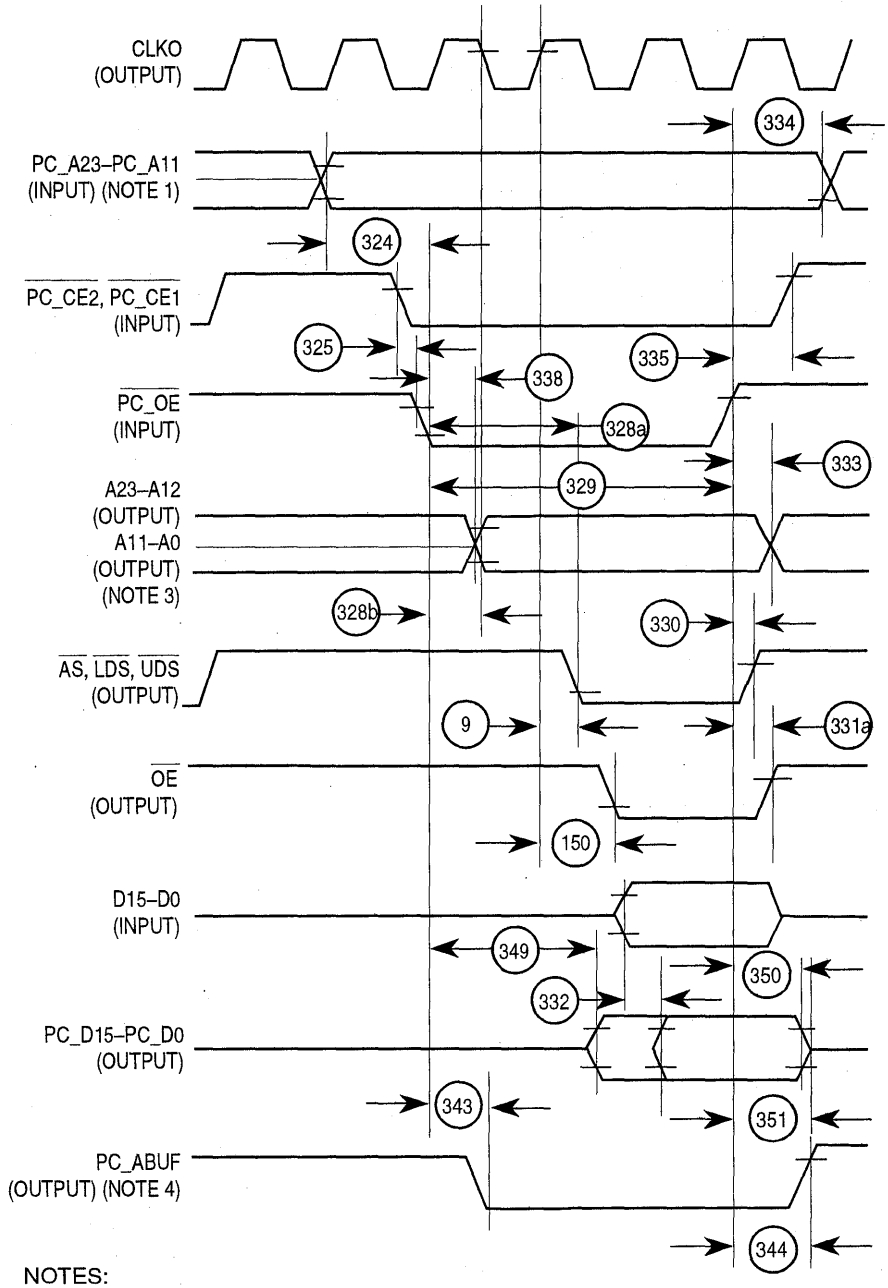
Figure 14-24. NMSI Timing Diagram

14.5.19 AC Electrical Specifications—PCMCIA Interface

Num.	Characteristic	25 MHz at 5.0 V		Unit
		Min	Max	
324	PC Address Setup before PC_OE or PC_WE Low	10		ns
325	PC_CE Setup before PC_OE or PC_WE Low	0		ns
326	PC_CE Hold from PC_WE or PC_OE High (See Note 1)	20		ns
327	PC_Address Hold from PC_OE or PC_WE High (See Note 1)	20		ns
328a	PC_WE or PC_OE Low to AS Low (only for FAST Burst Mode)	1	2.5+ 20	clk ns
328b	Asynchronous Input Setup	10		ns
329	Minimum PC_OE or PC_WE Width (only for FAST Burst Mode) (See Note 2)	3.5		clk
330	AS, LDS, UDS Hold from PC_OE or PC_WE High (only for FAST Burst Mode)	0	20	ns
331a	PC_OE High to OE High (only for FAST Burst Mode)		15	ns
331b	PC_WE High to WEH, WEL High (only for FAST Burst Mode)		15	ns
332	68k Data Bus Valid to PC_Data Bus Valid (See Note 3)		25	ns
333	PC_OE or PC_WE High to 68k Address Bus Invalid (only for FAST Burst Mode)		25	ns
334	PC_Address Hold from PC_OE or PC_WE High (See Note 4)	20		ns
335	PC_CE Hold from PC_OE or PC_WE High (See Note 4)	20		ns
336	Clock High to PC_ABUF Low		20	ns
337	Clock High to PC_ABUF High		20	ns
338	PC_OE or PC_WE Low to 68k Address Bus Valid (only for FAST Burst Mode)		30	ns
339	AS Low to 68k Data Out (only for FAST Burst Mode)	0	20	ns
340	PC_Data Valid to 68k Data Valid		25	ns
341	WEL,WEH High to 68k Data Invalid (only for FAST Burst Mode)	0		ns
342	PC_Data Setup before PC_WE High		30	ns
343	PC_OE or PC_WE Low to PC_ABUF Low (only for FAST Burst Mode)		20	ns
344	PC_OE or PC_WE High to PC_ABUF High (only for FAST Burst Mode)		20	ns
345	PC_Data Bus Hold Time from PC_OE High	20		ns
346	PC_OE or PC_WE or PC_IORD or PC_IOWR Low to PC_WAIT Low		30	ns
347	PC_OE or PC_WE or PC_IORD or PC_IOWR High from PC_WAIT High (See Note 5)	0		
348	Clock High to PC_WAIT High		20	ns
349	PC_OE Low to PC_Data Driven	5		ns
350	PC_OE High to PC_Data Invalid	0		ns
351	PC_Data High-Z from PC_OE High		30	ns
352	PC_Address Setup before IOWR or IORD	5		ns
353	PC_CE and PC_REG Setup before IOWR or IORD	5		ns
354	PC_OE Low to PC_Data Valid		25	ns
355	Clock Low to PC_Data Valid		25	ns

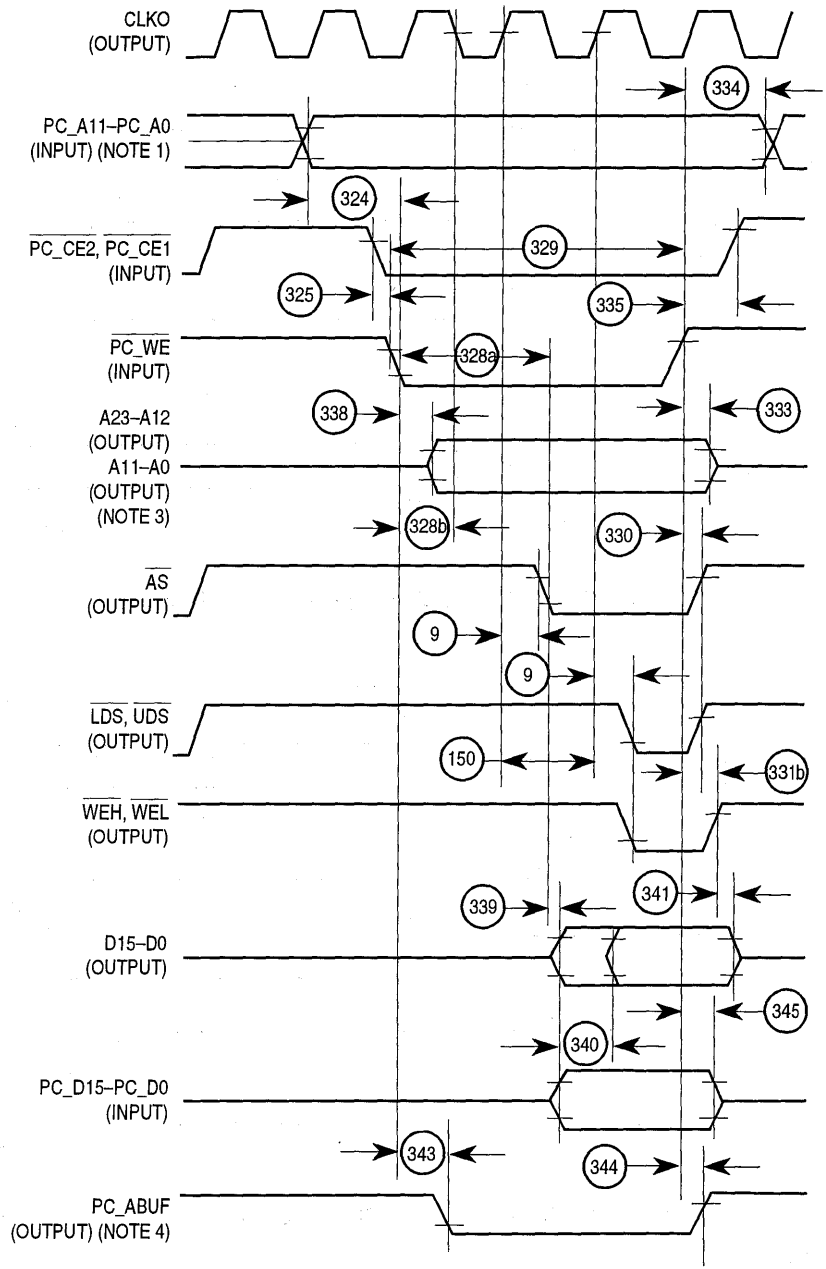
356	Minimum Pulse Width of $\overline{\text{PC\_IOWR}}$		2.5	clk
357	$\overline{\text{PC\_WAIT}}$ High to PC_Data Valid (See note 6)		0	ns
358	$\overline{\text{PC\_IORD}}$ High to PC_Data Invalid	0		ns
359	$\overline{\text{PC\_IORD}}$ Low to PC_Data Valid		1.5+ 30	clk ns
360	$\overline{\text{PC\_IORD}}$ High to PC_data High-Z		30	ns
361	PC_Data Setup before $\overline{\text{PC\_IOWR}}$ Low	0		ns
362	PC_Data Hold from $\overline{\text{PC\_IOWR}}$ High	0		ns
363	PC_Address, $\overline{\text{PC\_CE}}$ Hold from $\overline{\text{PC\_IORD}}$ or $\overline{\text{PC\_IOWR}}$ High	10		ns
364	$\overline{\text{PC\_REG}}$ Hold from $\overline{\text{PC\_IORD}}$ or $\overline{\text{PC\_IOWR}}$ High	0		ns

NOTES:
1. Hold Time Required for Register Accesses Only
2. If the Asynchronous Input Setup (#328) is met this value is 1.5 clks+10ns
3. Delay through chip assuming chip is driving PC_Data Bus
4. Hold Time Required for PCMCIA to 68k Bus Accesses
5. Max is Determined by PC Host Timing
6. PC_IO Cycles with $\overline{\text{PC\_WAIT}}$ Enabled



NOTES:
1. PC_A11 is input if ABUF/PC_A11 bit in PCMR = 0
2. Asynchronous input; if set-up is met $\overline{AS}$ is asserted as shown
3. If ABUF/A11 = 1 in PCMR A23–A12 are input to chip
4. Asserted if ABUF/A11 = 1 in PCMR

Figure 14-25. PCMCIA to 68K Fast Burst Read



NOTES:
1. PC_A11 is input if ABUF/PC_A11 bit in PCMR = 0
2. Asynchronous input; if set-up is met AS is asserted as shown
3. If ABUF/A11 = 1 in PCMR A23–A12 are input to chip
4. Asserted if ABUF/A11 = 1 in PCMR

Figure 14-26. PCMCIA to 68K Fast Burst Write

14

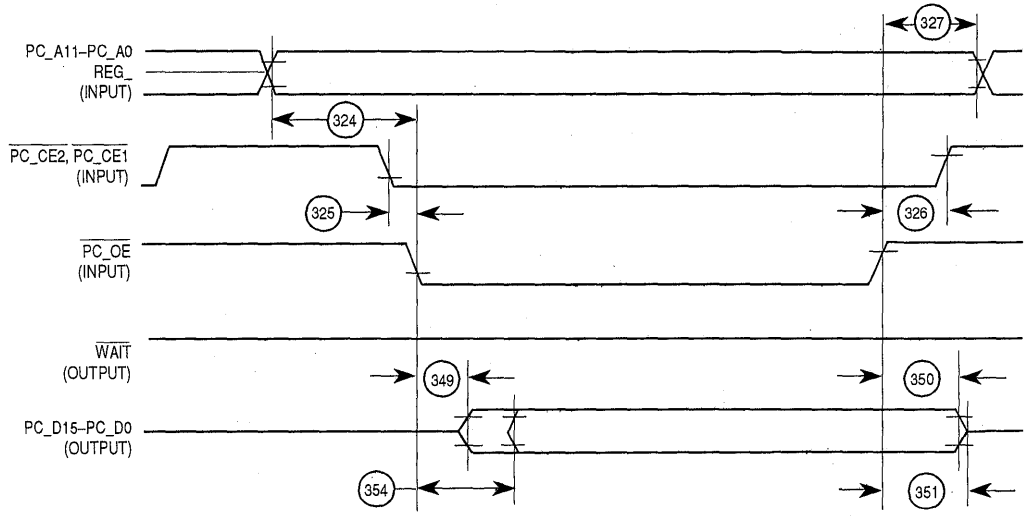


Figure 14-27. PCMCIA to 68K Register Read

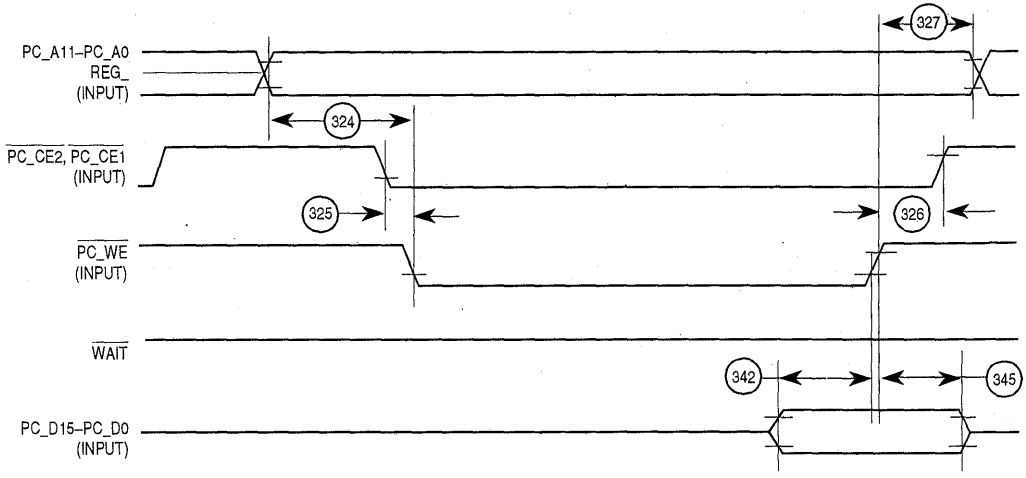
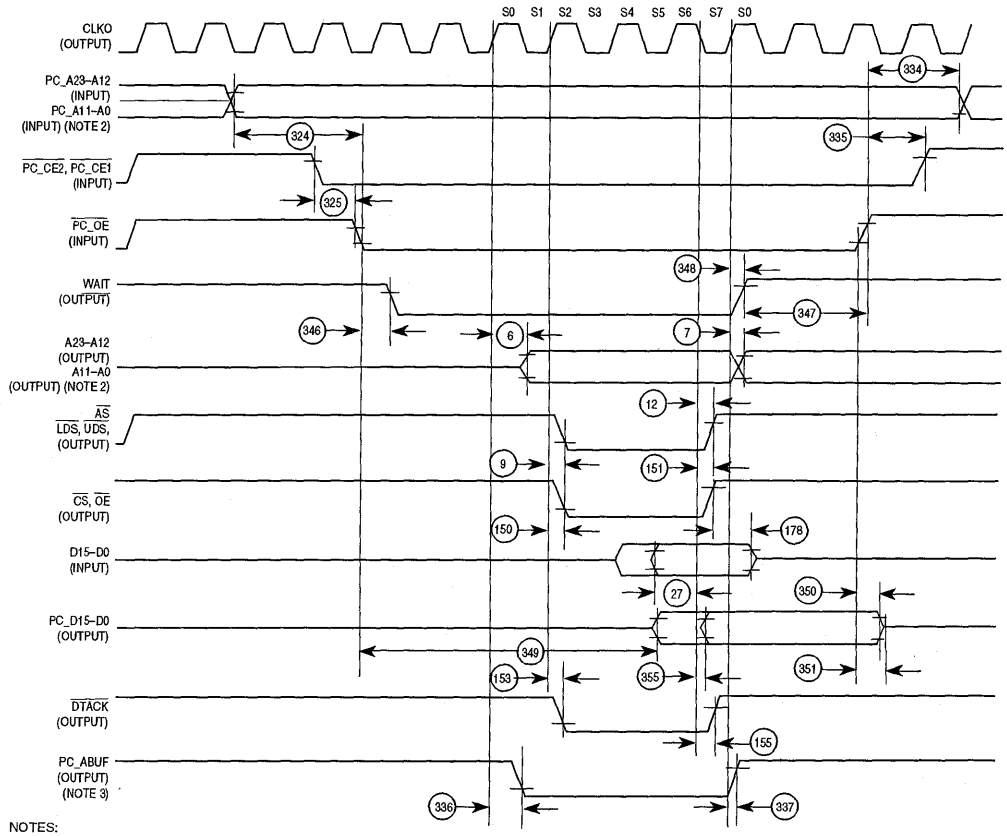


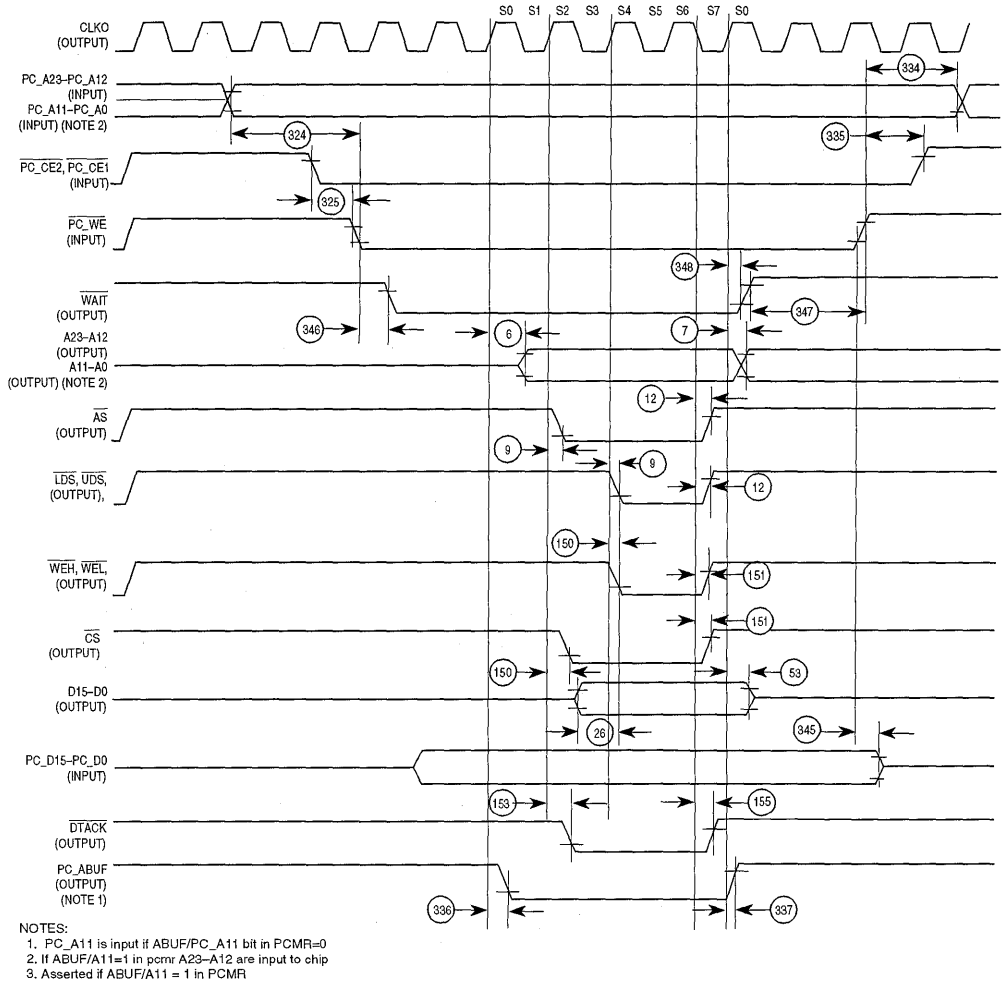
Figure 14-28. PCMCIA to 68K Register Write



NOTES:
1. PC_A11 is input if ABUF/PC_A11 bit in PCMR=0
2. If ABUF/A11=1 in PCMR A23–A12 are input to chip
3. Asserted if ABUF/A11 = 1 in PCMR

Figure 14-29. PCMCIA Normal Read

14



NOTES:
1. PC_A11 is input if ABUF/PC_A11 bit in PCMR=0
2. If ABUF/A11=1 in pcmr A23–A12 are input to chip
3. Asserted if ABUF/A11 = 1 in PCMR

14

Figure 14-30. PCMCIA Normal Write

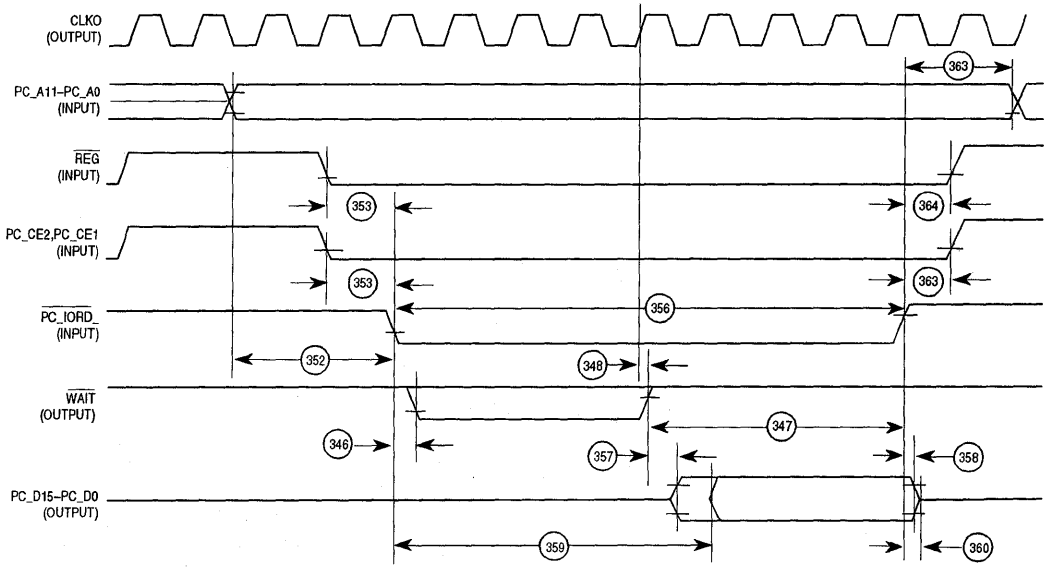


Figure 14-31. PCMCIA I/O (16550 Emulation) Read

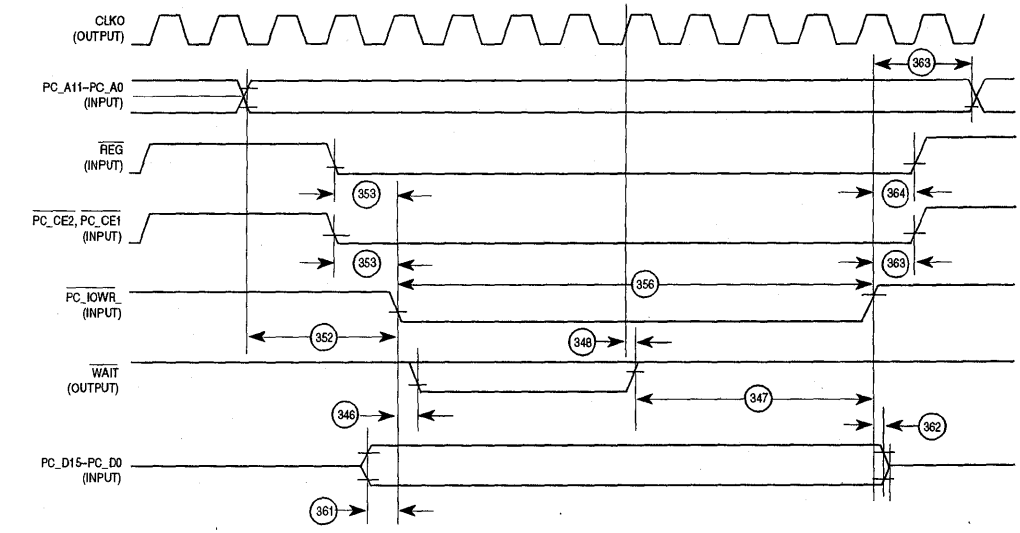


Figure 14-32. PCMCIA I/O (16550 Emulation) Write

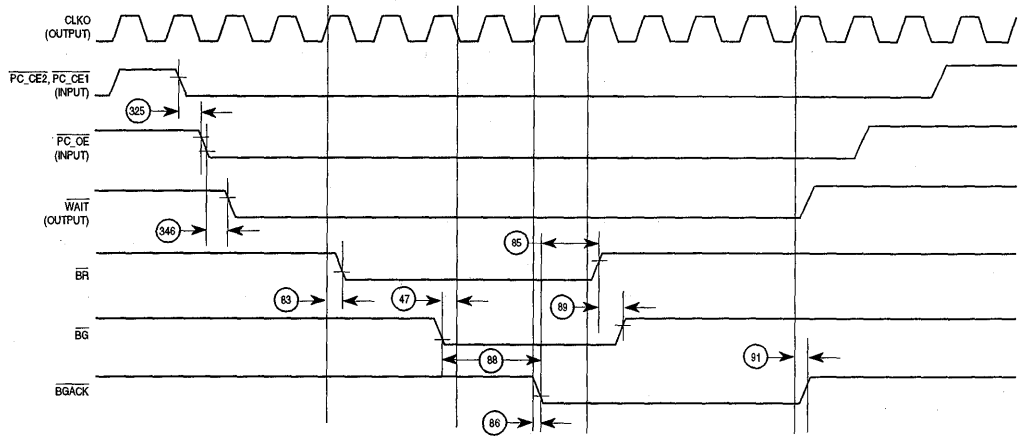


Figure 14-33. PCMCIA to 68K Arbitration

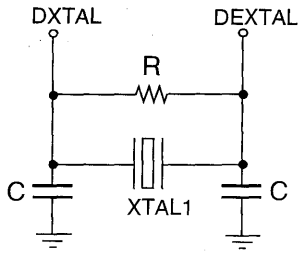
14.6 DSP AC ELECTRICAL CHARACTERISTICS

The timing waveforms in these tables are tested with a V_{IL} maximum of 0.5 V and a V_{IH} minimum of 2.4 V for all pins, except DEXTAL, $\overline{\text{DRESET}}$, MODA, MODB, and MODC. These four pins are tested using the input levels set forth in 14.4.3 DC Electrical Characteristics. AC timing specifications which are referenced to a device input signal are measured with respect to the 50% point of the respective input signal's transition. DSP output levels are measured with the production test machine V_{OL} and V_{OH} reference levels set at 0.8 V and 2.0 V respectively.

14.6.1 AC Electrical Characteristics - Internal Clocks

For each occurrence of T_H, T_L, T_C or I_{CYC} substitute with the numbers in the following table:

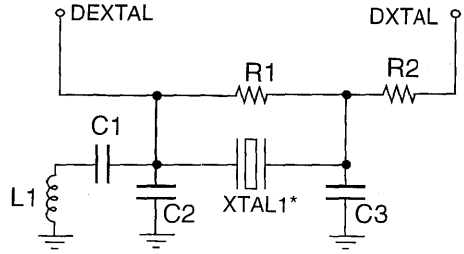
Characteristics	Symbol	Expression
Internal Operation Frequency	f	
Internal Clock High Period - with PLL disabled	T_H	ETH
- with PLL enabled and MF ≤ 4		(Min) 0.48*ETc*DF/MF (Max) 0.52*ETc*DF/MF
- with PLL enabled and MF>4		(Min) 0.467*ETc*DF/MF (Max) 0.533*ETc*DF/MF
Internal Clock Low Period - with PLL disabled	T_L	ETL
- with PLL enabled and MF ≤ 4		(Min) 0.48*ETc*DF/MF (Max) 0.52*ETc*DF/MF
- with PLL enabled and MF>4		(Min) 0.467*ETc*DF/MF (Max) 0.533*ETc*DF/MF
Internal Clock Cycle Time	Tc	ETc*DF/MF
Instruction Cycle Time	Icyc	2*Tc



Fundamental Frequency Crystal Oscillator

Suggested Component Values
For f_{osc} = 4 MHz:
R = 680 KΩ ± 10%
C = 20 pf ± 20%

For f_{osc} = 30 MHz:
R = 680 KΩ ± 10%
C = 20 pf ± 20%
Notes:
(1) The suggested crystal source is ICM, # 433163 - 4.00 (4MHz fundamental, 20 pf load) or # 436163 - 30.00 (30 MHz fundamental, 20 pf load).

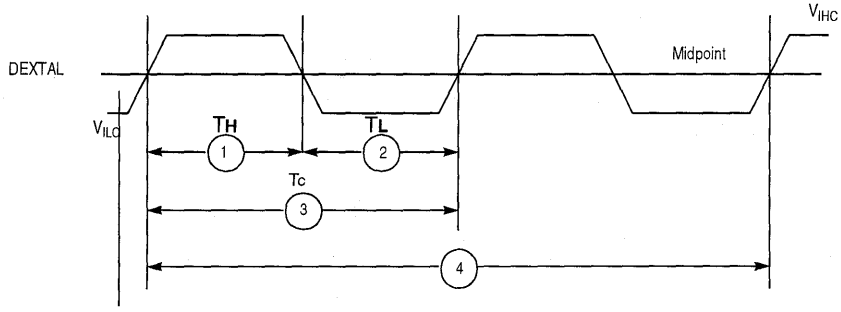


3rd Overtone Crystal Oscillator

Suggested Component Values
R1 = 470 K Ω ± 10%
R2 = 330 Ω ± 10%
C1 = 0.1 μf ± 20%
C2 = 26 pf ± 20%
C3 = 20 pf ± 10%
L1 = 2.37 μH ± 10%
XTAL = 40 MHz, AT cut, 20 pf load,
50Ω max series resistance

Notes:
(1) *3rd overtone crystal.
(2) The suggested crystal source is ICM, # 471163 - 40.00 (40 MHz 3rd overtone, 20 pf load).
(3) R2 limits crystal current
(4) Reference Benjamin Parzen, The Design of Crystal and Other Harmonic Oscillators, John Wiley& Sons, 1983

Figure 14-34. Crystal Oscillator Circuits



Note: The midpoint is $V_{ILC} + 0.5 (V_{IHC} - V_{ILC})$.

Figure 14-35. External Clock Timing

14.6.2 DSP AC Electrical Characteristics - External Clock Operation

The DSP system clock may be derived from the on-chip crystal oscillator as shown in Figure 14-33, or it may be externally supplied. An externally supplied square wave voltage source should be connected to DEXTAL, leaving DXTAL physically not connected (see Figure 14-35) to the board or socket. The rise and fall time of this external clock should be 4ns maximum.

Num	Characteristics	Symbol	60 MHz		Unit
			Min	Max	
	Frequency of Operation (DEXTAL Pin)	E_f	0	60	MHz
1	Clock Input High (see Note) -with PLL disabled (46.7% - 53.3% duty cycle) -with PLL enabled (42.5% - 57.5% duty cycle)	E_{TH}	7.8 7.1	infinity 235.5 µs	ns
2	Clock Input Low (see Note) - with PLL disabled (46.7% - 53.3% duty cycle) - with PLL enabled (42.5% - 57.5% duty cycle)	E_{TL}	7.8 7.1	infinity 235.5 µs	ns
3	Clock Cycle Time - with PLL disabled - with PLL enabled	E_{TC}	16.67 16.67	infinity 409.6 µs	ns
4	Instruction Cycle Time = $I_{cyc} = 2 \times T_C$ (see Note) - with PLL disabled - with PLL enabled	I_{CYC}	33.3 33.3	infinity 819.2 µs	ns

Note: External clock input high and external clock input low are measured at 50% of the input transition.

14.6.3 AC Electrical Characteristics - DSP Phased Lock Loop (DPLL) Characteristics

14

Characteristics	Expression	Min	Max	Unit
VCO frequency when PLL enabled	MF * Ef	10	f (Note 1.)	MHz
PLL external capacitor (PCAP pin to PVCC)	MF * Cpcap (Note 1.) @ MF ≤ 4 @ MF > 4	 MF * 340 MF * 380	 MF * 480 MF * 970	pF

1. Cpcap is the value of the PLL capacitor (connected between PCAP pin and PVCC) for MF=1. The recommended value for Cpcap is 400pF for MF ≤ 4 and 540pF for MF> 4. The maximum VCO frequency is limited to the internal operation frequency, as defined above.

14.6.4 DSP AC Electrical Characteristics - Reset, Stop, Mode Select and Interrupt Timing

(See Figure 14-36 through Figure 14-43)

WS = Number of wait states (1 WS = Tc) programmed into external bus access using BCR (WS = 0 – 15)

Num	Characteristics	60 MHz		Unit
		Min	Max	
9	Delay from $\overline{\text{DRESET}}$ Assertion to Address High Impedance (periodically sampled and not 100% tested).	—	26	ns
10	Minimum Stabilization Duration • Internal Osc. PLL Disabled (see Note 1) • External clock PLL Disabled (see Note 2.) • External clock PLL Enabled (see Note 2.)	75000*Tc 25*Tc 2500*Tc	— — —	ns
11	Delay from Asynchronous $\overline{\text{DRESET}}$ Deassertion to First External Address Output (Internal DRESET Deassertion)	8*Tc	9*Tc+20	ns
12	Synchronous Reset Setup Time from $\overline{\text{DRESET}}$ Deassertion to CKOUT transition #1	8.5	Tc	ns
13	Synchronous Reset Delay Time from the CKOUT transition #1 to the First External Address Output	8*Tc	8*Tc+6	ns
14	Mode Select Setup Time	21	—	ns
15	Mode Select Hold Time	0	—	ns
16	Minimum Edge-Triggered Interrupt Request Assertion Width	13	—	ns
16a	Minimum Edge-Triggered Interrupt Request Deassertion Width	13	—	ns
17	Delay from $\overline{\text{IRQA}}$, $\overline{\text{IRQB}}$, $\overline{\text{NMI}}$ Assertion to External Memory Access Address Out Valid • Caused by First Interrupt Instruction Fetch • Caused by First Interrupt Instruction Execution	 5*Tc+TH 9*Tc+TH	 — —	ns
18	Delay from $\overline{\text{IRQA}}$, $\overline{\text{IRQB}}$, $\overline{\text{NMI}}$ Assertion to General Purpose Transfer Output Valid Caused by First Interrupt Instruction Execution	11*Tc+TH	—	ns
19	Delay from Address Output Valid Caused by First Interrupt Instruction Execute to Interrupt Request Deassertion for Level Sensitive Fast Interrupts	—	2Tc+TL+(Tc*WS)–23	ns
20	Delay from $\overline{\text{RD}}$ Assertion to Interrupt Request Deassertion for Level Sensitive Fast Interrupts	—	2Tc+(Tc*WS)–21	ns
21	Delay from $\overline{\text{WR}}$ Assertion to Interrupt Request Deassertion for Level Sensitive Fast Interrupts • WS=0 • WS > 0	 — —	2*Tc–21 Tc+TL+(Tc*WS)–21	ns

14

Num	Characteristics	60 MHz Min	60 MHz Max	Unit
22	Delay from General-Purpose Output Valid to Interrupt Request Deassertion for Level Sensitive Fast Interrupts - If Second Interrupt Instruction is: • Single Cycle • Two Cycles	 — —	 T_L–31 (2*Tc)+T_L–31	ns
23	Synchronous Interrupt Setup Time from $\overline{IRQA}$, $\overline{IRQB}$, NMI Assertion to the CKOUT transition #2	10	Tc	ns
24	Synchronous Interrupt Delay Time from the CKOUT transition #2 to the First External Address Output Valid Caused by the First Instruction Fetch after Coming out of Wait State	13*Tc+TH	13*Tc+TH+6	ns
25	Duration for $\overline{IRQA}$ Assertion to Recover from Stop State	12	—	ns
26	Delay from $\overline{IRQA}$ Assertion to Fetch of First Interrupt Instruction (when exiting 'STOP') • Internal Crystal Oscillator Clock, OMR bit 6 = 0 • Stable External Clock,OMR bit 6 = 1 • Stable External Clock,PCTL bit 17= 1 (See Note 1)	 65548*Tc 20*Tc 13*Tc	 — — —	ns
27	Duration of Level Sensitive $\overline{IRQA}$ Assertion to ensure interrupt service (when exiting 'STOP') • Internal Crystal Osc. Clock, OMR bit 6 = 0 • Stable External Clock, OMR bit 6 = 1 • Stable External Clock, PCTLbit 17 = 1 (See Note 1)	 65534*Tc+TL 6*Tc+TL 12	 — — —	ns
28	Delay from Level Sensitive $\overline{IRQA}$ Assertion to Fetch of First Interrupt Instruction (when exiting 'STOP') • Internal Crystal Osc. ClockOMR bit 6 = 0 • Stable External Clock, OMR bit 6 = 1 • Stable External Clock, PCTL bit 17= 1 (See Note 1)	 65548*Tc 20*Tc 13*Tc	 — — —	ns

Notes:

1. A clock stabilization delay is required when using the on-chip crystal oscillator in two cases:
   1) after power-on reset, and
   2) when recovering from Stop mode.
   During this stabilization period, Tc, T_H and T_L will not be constant. Since this stabilization period varies, a delay of 75,000*Tc is typically allowed to assure that the oscillator is stable before executing programs. While it is possible to set OMR bit 6 = 1 when using the internal crystal oscillator, it is not recommended and these specifications do not guarantee timings for that case.
2. Circuit stabilization delay is required during reset when using an external clock in two cases:
   1) after power-on reset, and
   2) when recovering from Stop mode.

NOTE

When using fast interrupts and $\overline{IRQA}$ and $\overline{IRQB}$ are defined as level-sensitive, then timings 19 through 22 apply to prevent multiple interrupt service. To avoid these timing restrictions, the deassertive edge-triggered mode is recommended when using fast interrupt. Long interrupts are recommended when using level-sensitive mode.

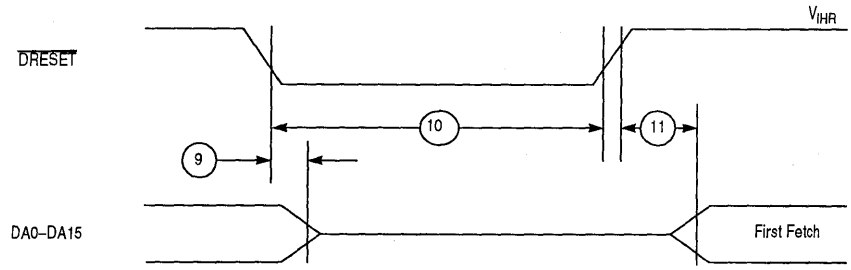


Figure 14-36. Reset Timing

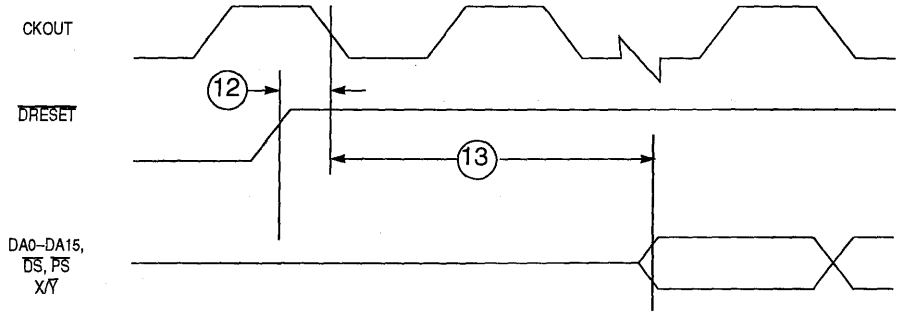


Figure 14-37. Synchronous Reset Timing

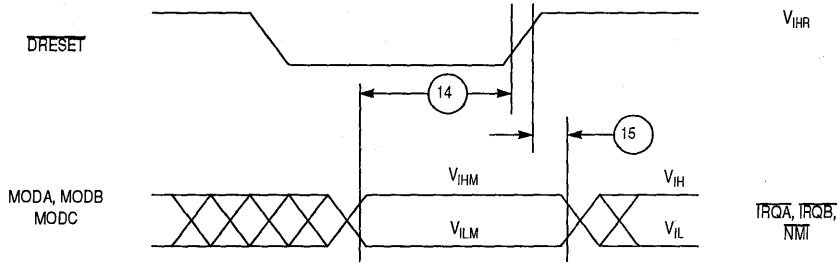


Figure 14-38. Operating Mode Select Timing

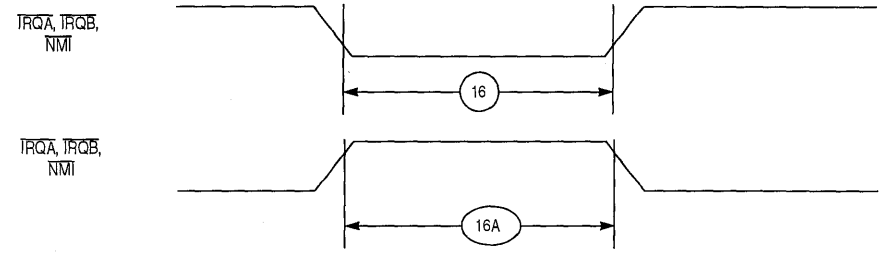
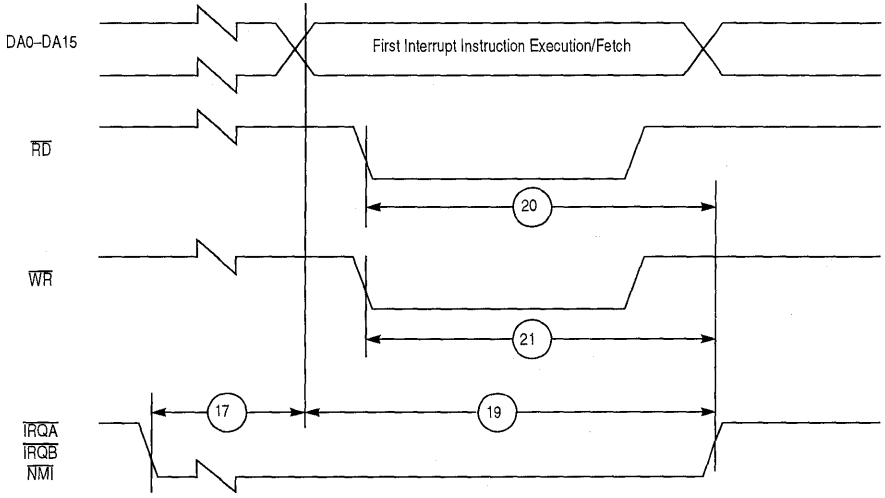


Figure 14-39. External Interrupt Timing (Negative Edge-Triggered)



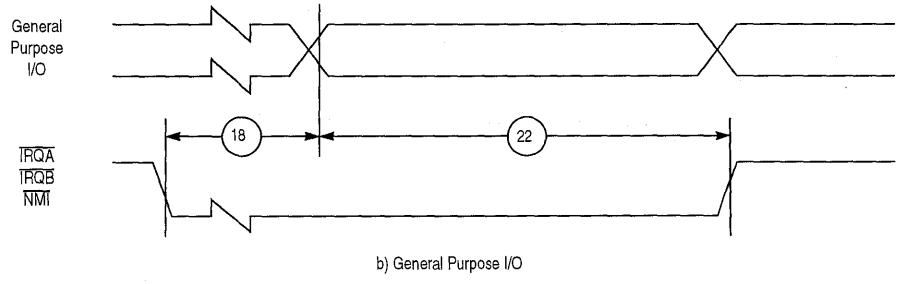


Figure 14-40. External Level-Sensitive Fast Interrupt Timing

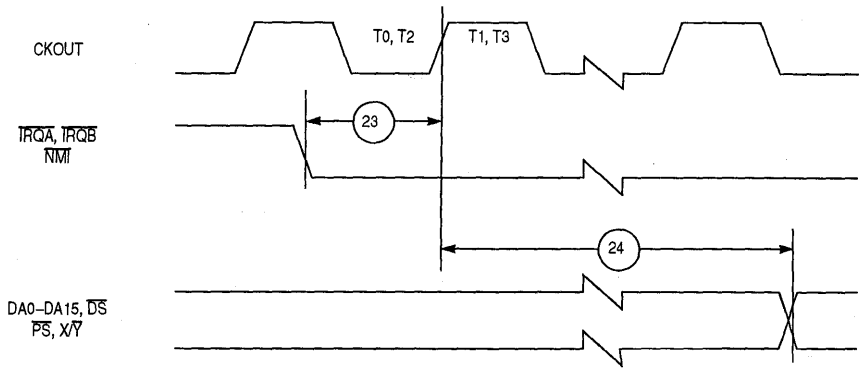


Figure 14-41. Synchronous Interrupt from Wait State Timing

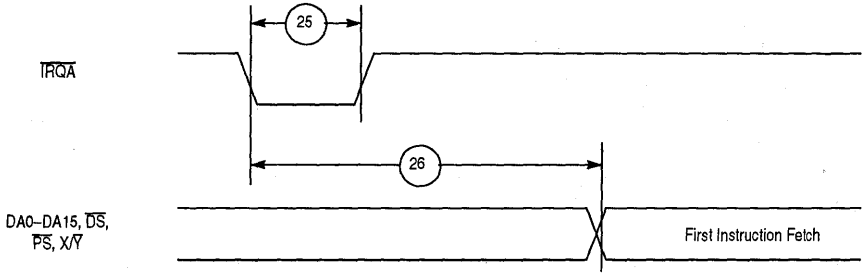


Figure 14-42. Recovery from Stop State Using IRQA

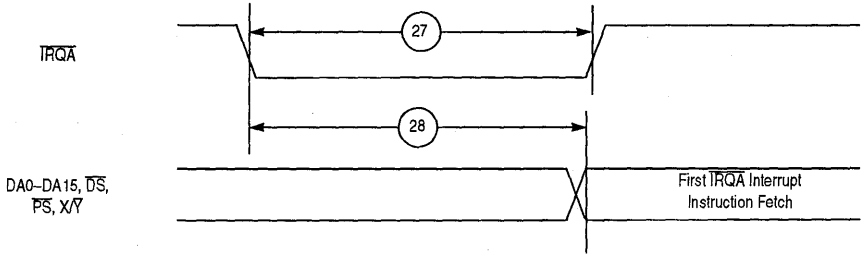


Figure 14-43. Recovery from Stop State Using IRQA Interrupt Service

14.6.5 Host Port Usage Considerations

Careful synchronization is required when reading multibit registers that are written by another asynchronous system. This is a common problem when two asynchronous systems are connected. The situation exists in the host port. The considerations for proper operation are discussed below.

14.6.5.1 Host Programmer Considerations

14.6.5.1.1 Unsynchronized Reading of Receive Byte Registers

When reading receive byte registers RXH, RXM, or RXL, the host programmer should use interrupts or poll the RXDF flag which indicates that data is available. This assures that the data in the receive byte registers will be stable.

14.6.5.1.2 Overwriting Transmit Byte Registers

The host programmer should not write to the transmit byte registers, TXH, TXM, or TXL, unless the TXDE bit is set indicating that the transmit byte registers are empty. This guarantees that the transmit byte registers will transfer valid data to the HRX register.

14.6.5.1.3 Synchronization of Status Bits from DSP to Host

HC, HREQ, DMA, HF3, HF2, TRDY, TXDE, and RXDF status bits are set or cleared from inside the DSP and read by the host processor. The host can read these status bits very quickly without regard to the clock rate used by the DSP, but the possibility exists that the state of the bit could be changing during the read operation. This is generally not a system problem, since the bit will be read correctly in the next pass of any host polling routine.

A potential problem exists when reading status bits HF3 and HF2 as an encoded pair. If the DSP changes HF3 and HF2 from 00 to 11, there is a small probability that the host could read the bits during the transition and receive 01 or 10 instead of 11. If the combination of HF3 and HF2 has significance, the host could read the wrong combination.

Solution: Read the bits twice and check for consensus.

14.6.5.1.4 Overwriting the Host Vector

The host programmer should change the host vector register only when the host command bit (HC) is clear. This change will guarantee that the DSP interrupt control logic will receive a stable vector.

14.6.5.1.5 Cancelling a Pending Host Command Exception

The host processor may elect to clear the HC bit to cancel the host command exception request at any time before it is recognized by the DSP. Because the host does not know exactly when the exception will be recognized (due to exception processing synchronization and pipeline delays), the DSP may execute the host exception after the HC bit is cleared. For these reasons, the HV bits must not be changed at the same time the HC bit is cleared.

14.6.5.2 DSP Programer Considerations

14.6.5.2.1 Reading HF0 and HF1 as an Encoded Pair

DMA, HF1, HF0, and HCP, HTDE, and HRDF status bits are set or cleared by the host processor side of the interface. These bits are individually synchronized to the DSP clock.

A potential problem exists when reading status bits HF1 and HF2 as an encoded pair, i.e., the four combinations 00, 01, 10, and 11 each have significance. A very small probability exists that the DSP will read the status bits synchronized during transition. The solution to this potential problem is to read the bits twice for consensus.

14.6.6 AC Electrical Characteristics - SCI Timing

t_{SCC} = Synchronous Clock Cycle Time (for internal clock, t_{SCC} is determined by the SCI clock control register and T_C).

Table 14-1. SCI Synchronous Mode Timing

Num	Characteristics	60 MHz		Unit
		Min	Max	
55	Synchronous Clock Cycle — t_{SCC}	8*Tc	—	ns
56	Clock Low Period	4*Tc–10.5	—	ns
57	Clock High Period	4*Tc –10.5	—	ns
63	Clock Falling Edge to Output Data Valid (External Clock)	—	32.5	ns
64	Output Data Hold After Clock Rising Edge (External Clock)	Tc+3	—	ns
65	Input Data Setup Time Before Clock Rising Edge (External Clock)	16	—	ns
66	Input Data Hold Time After Clock Rising Edge (External Clock)	21	—	ns

14

Table 14-2. SCI Asynchronous Minimum Clock Cycle Time

Characteristics	60 MHz		Unit
	Min	Max	
Asynchronous Clock Cycle-t_{ACC}	64*Tc	—	ns

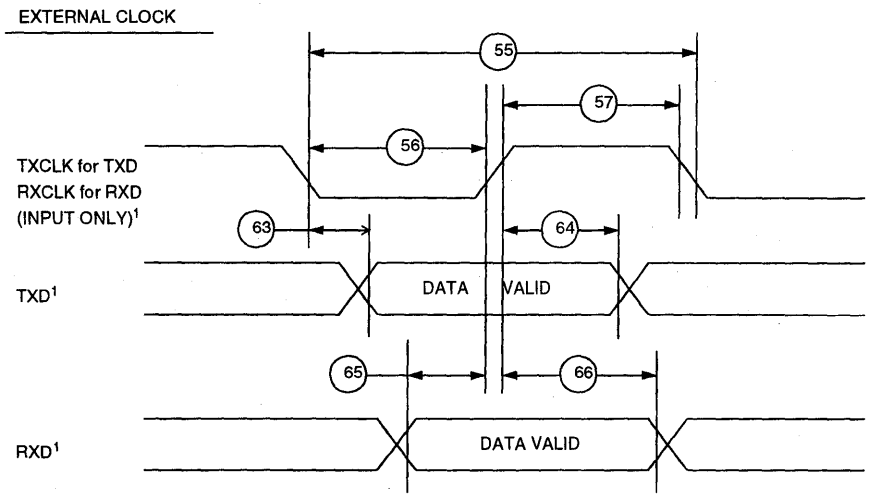


Figure 14-44. SCI+ Synchronous Mode Timing

Note 1 - The SCI+ signals do not have dedicated pins and the actual pins used for these signals depend on the SCI+ connection mode set up in the DISC register.

14.6.7 AC Electrical Characteristics - SSI Timing

See Figure 14-45 and Figure 14-46
t_{SSICC} = SSI clock cycle time
TXC (SCK Pin) = Transmit Clock
RXC (SC0 or SCK Pin) = Receive Clock
FST (SC2 Pin) = Transmit Frame Sync
FSR (SC1 or SC2 Pin) = Receive Frame Sync
i ck = Internal Clock
x ck = External Clock
g ck = Gated Clock
i ck a = Internal Clock, Asynchronous Mode (Asynchronous implies that TXC and RXC are two different clocks)
i ck s = Internal Clock, Synchronous Mode (Synchronous implies that TXC and RXC are the same clock)
bl = bit length
wl = word length

Num	Characteristics	60 MHz		Case	Unit
		Min	Max		
80	Clock Cycle-t_{SSICC} (see Note 1)	4*Tc 3*Tc	— —	i ck x ck	ns
81	Clock High Period for internal clock for external clock	2*Tc–10.8 Tc+Tl	— —		ns
82	Clock Low Period for internal clock for external clock	2*Tc–10.8 Tc+Tl	— —		ns
84	RXC Rising Edge to FSR Out (bl) High	— —	40.8 25.8	x ck i ck a	ns
85	RXC Rising Edge to FSR Out (bl) Low	— —	35.8 25.8	x ck i ck a	ns
86	RXC Rising Edge to FSR Out (wl) High	— —	35.8 20.8	x ck i ck a	ns
87	RXC Rising Edge to FSR Out (wl) Low	— —	35.8 20.8	x ck i ck a	ns
88	Data In Setup Time Before RXC (SCK in Synchronous Mode) Falling Edge	3.3 15.8 13	— — —	x ck i ck a i ck s	ns
89	Data In Hold Time After RXC Falling Edge	18 3.3	— —	x ck i ck	ns
90	FSR Input (bl) High Before RXC Falling Edge	0.8 17.4	— —	x ck i ck a	ns
91	FSR Input (wl) High Before RXC Falling Edge	3.3 18.3	— —	x ck i ck a	ns
92	FSR Input Hold Time After RXC Falling Edge	18.3 3.3	— —	x ck i ck	ns

14

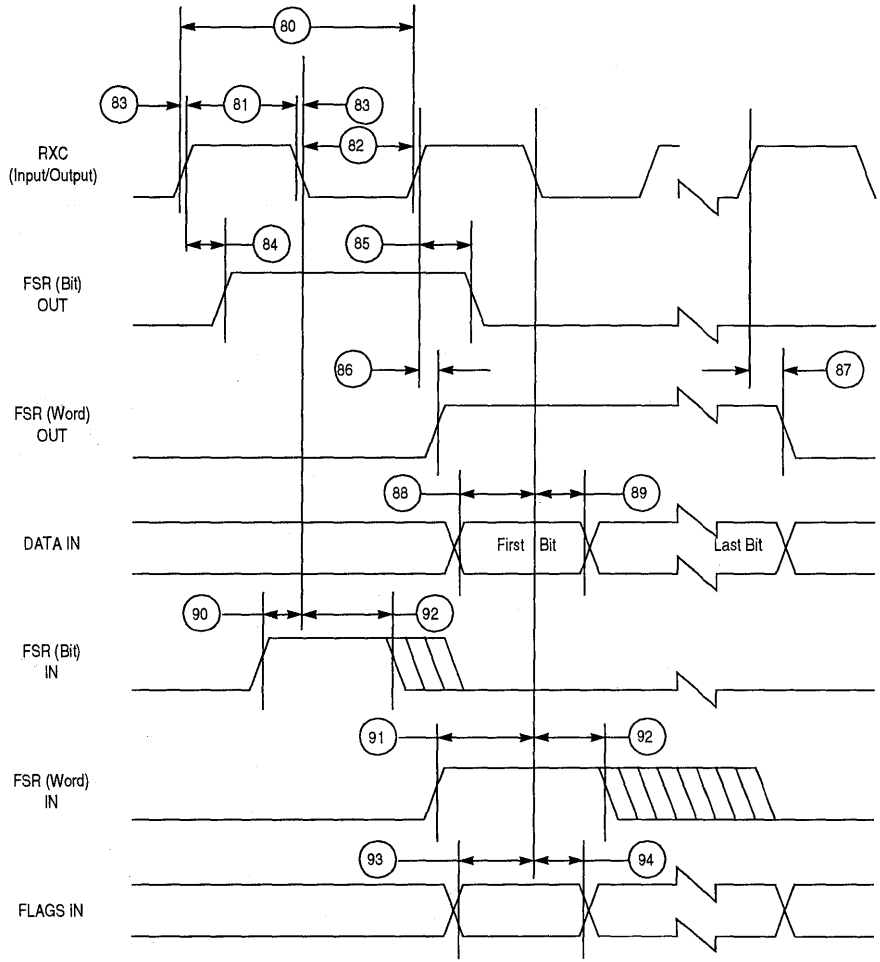
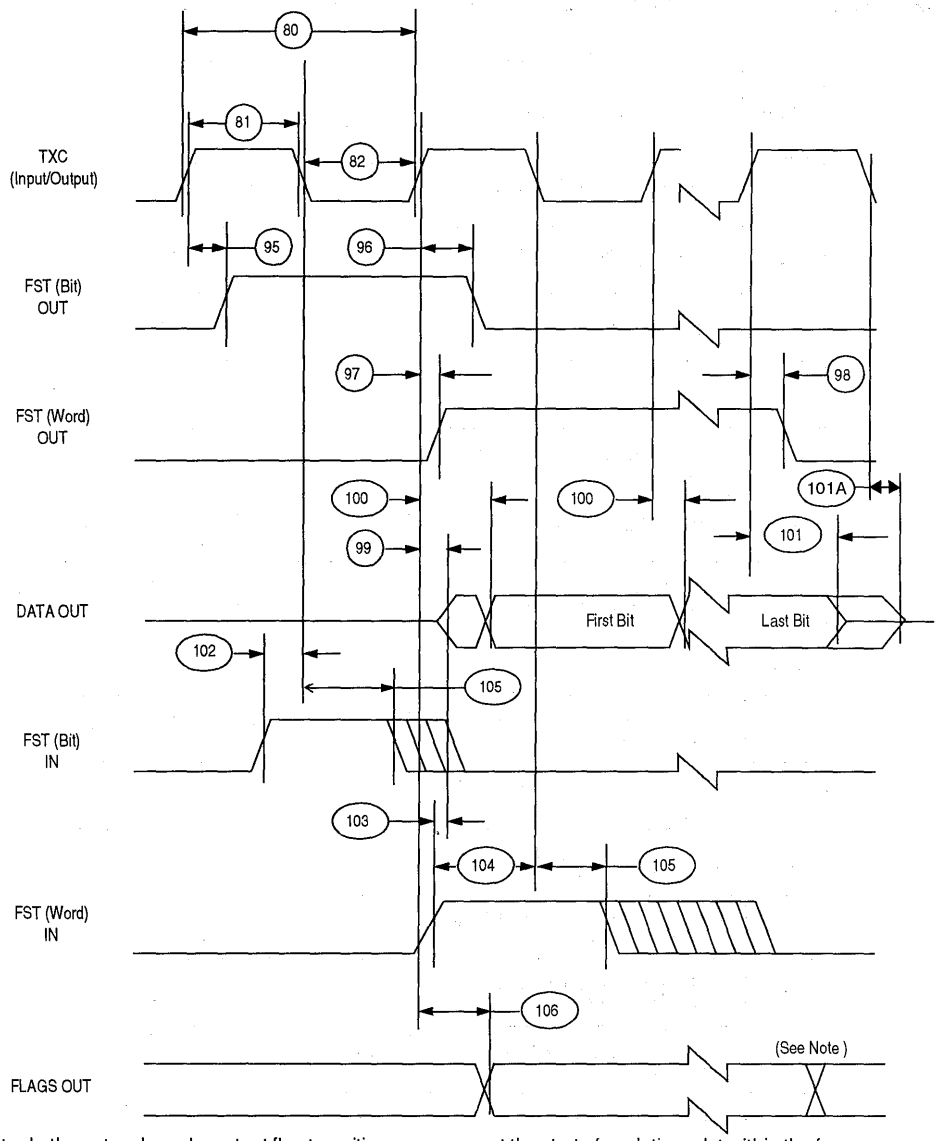


Figure 14-45. SSI Receiver Timing

AC Electrical Characteristics - SSI Timing (Continued)

Num	Characteristics	60 MHz		Case	Unit
		Min	Max		
93	Flags Input Setup Before RXC Falling Edge	0.8 16.7	— —	x ck i ck s	ns
94	Flags Input Hold Time After RXC Falling Edge	18.3 3.3	— —	x ck i ck s	ns
95	TXC Rising Edge to FST Out (bl) High	— —	31.6 15.8	x ck i ck	ns
96	TXC Rising Edge to FST Out (bl) Low	— —	33.3 18.3	x ck i ck	ns
97	TXC Rising Edge to FST Out (wl) High	— —	30.8 18.3	x ck i ck	ns
98	TXC Rising Edge to FST Out (wl) Low	— —	33.3 18.3	x ck i ck	ns
99	TXC Rising Edge to Data Out Enable from High Impedance	—	33.3+TH 20.8	x ck i ck	ns
100	TXC Rising Edge to Data Out Valid	—	33.3+TH 22.4	x ck i ck	ns
101	TXC Rising Edge to Data Out High Impedance	— —	35.8 20.8	x ck i ck	ns
101A	TXC Falling Edge to Data Out High Impedance	—	TC+TH	g ck	ns
102	FST Input (bl) Setup Time Before TXC Falling Edge	0.8 18.3	—	x ck i ck	ns
103	FST Input (wl) to Data Out Enable from High Impedance	—	30.8		ns
104	FST Input (wl) Setup Time Before TXC Falling Edge	0.8 20.0	— —	x ck i ck	ns
105	FST Input Hold Time After TX Falling Edge	18.3 3.3	— —	x ck i ck	ns
106	Flag Output Valid After TXC Rising Edge	— —	32.5 20.8	x ck i ck	ns

Notes:
1. For internal clock, external clock cycle is defined by Icyc and SSI control register.



Note: In the network mode, output flag transitions can occur at the start of each time slot within the frame. In the normal mode, the output flag state is asserted for the entire frame period.

Figure 14-46. SSI Transmitter Timing

14.6.8 AC Electrical Characteristics — Capacitance Derating — External Bus Asynchronous Timing

See Figure 14-47
WS = Number of Wait States, Determined by BCR Register (WS = 0 to 15)

The DSP external bus timing specifications are designed and tested at the maximum capacitive load of 50 pF, including stray capacitance. Typically, the drive capability of the external bus pins (DA0–DA15, DD0–DD23, $\overline{PS}$, $\overline{DS}$, $\overline{RD}$, $\overline{WR}$, X/$\overline{Y}$) derates linearly at 1 ns per 12 pF of additional capacitance from 50 pF to 250 pF of loading. Port B and C pins derate linearly at 1 ns per 5 pF of additional capacitance from 50 pF to 250 pF of loading.

Active low lines should be "pulled up" in a manner consistent with the AC and DC specifications.

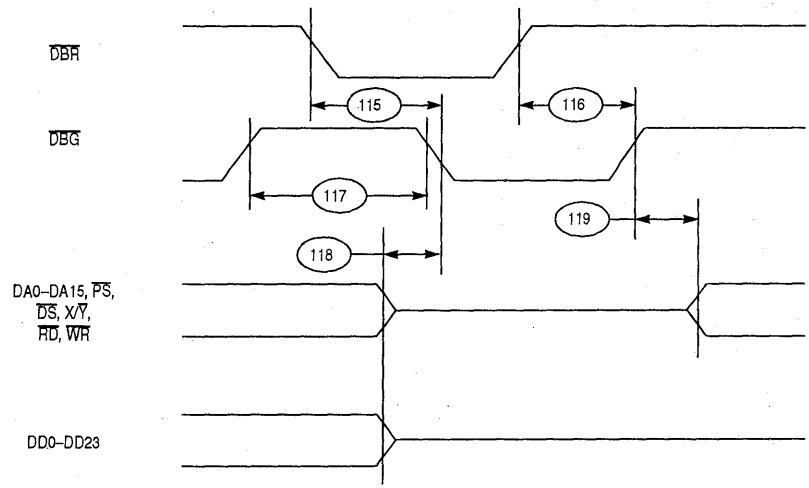


Figure 14-47. Bus Request / Bus Grant Timing

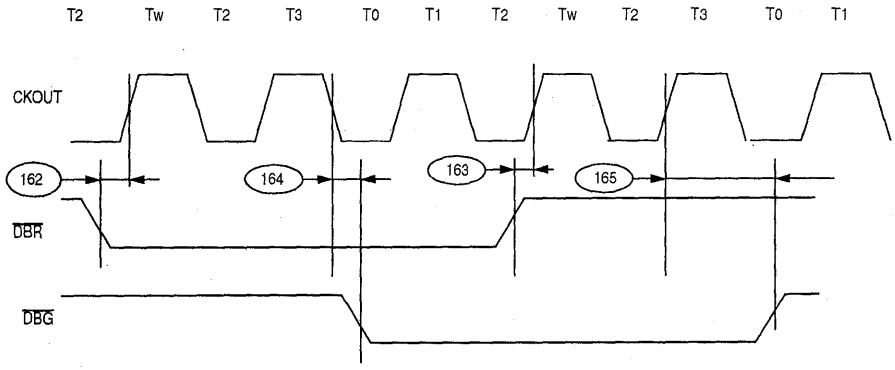


Figure 14-48. Synchronous Bus Request/Bus Grant Timing

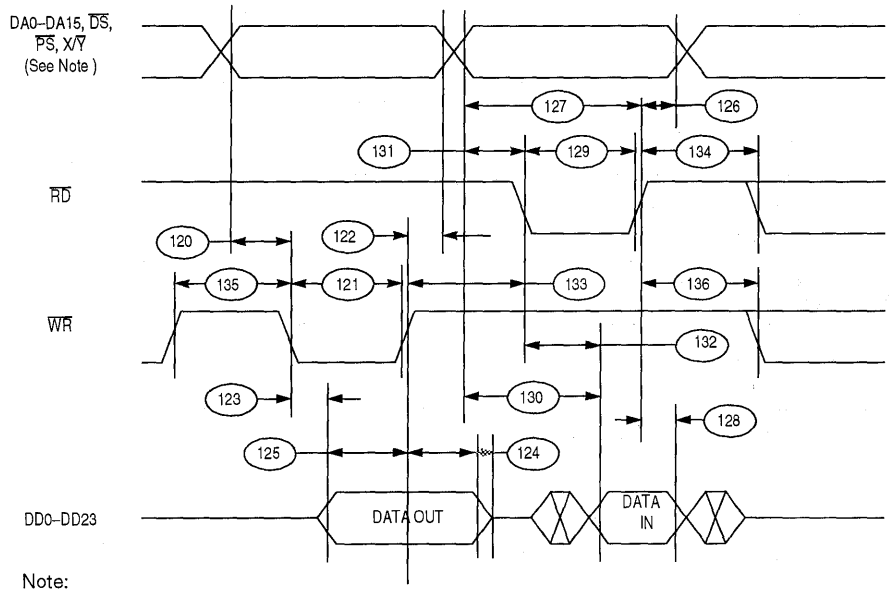
14.6.9 AC Electrical Characteristics - External Bus Asynchronous Timing

Num	Characteristics	60 MHz		Unit
		Min	Max	
115	Delay from $\overline{DBR}$ Assertion to $\overline{DBG}$ Assertion • (see Note 1) • (see Note 2) • (see Note 3) • (see Note 4) • (see Note 5)	 2Tc+Th Tc+Th Tc+Th Infinity Th	 4Tc+Th+14 4Tc+Th+(Tc*Ws)+14 6Tc+Th+(2*Tc*WS)+14 — Tc+Th+15	ns
116	Delay from $\overline{DBR}$ Deassertion to $\overline{DBG}$ Deassertion	2*Tc	4*Tc+12.5	ns
117	$\overline{DBG}$ Deassertion Duration • (see Note 5) • all other cases	 2*Tc	 — —	ns
118	Delay from Address, Data, and Control Bus High Impedance to $\overline{DBG}$ Assertion	Tc-5.5 2*Tc+Th-5.5	—	ns
119	Delay from $\overline{DBG}$ Deassertion to Address and Control Bus Enabled	0	—	ns
120	Address Valid to $\overline{WR}$ Assertion • WS = 0 • WS > 0	 TH– 6 Tc-6	 — —	ns
121	$\overline{WR}$ Assertion Width • WS = 0 • WS > 0	Tc–3	—	ns
122	$\overline{WR}$ Deassertion to Address Not Valid	TH – 6	—	ns
123	$\overline{WR}$ Assertion to Data Out Active • WS = 0 From High Impedance • WS > 0	TH-4 0	—	ns
124	Data Out Hold Time from $\overline{WR}$ Deassertion (the maximum specification is periodically sampled, and not 100% tested)	TH–5	TH–1.5	ns
125	Data Out Setup Time to $\overline{WR}$ Deassertion • WS = 0 • WS > 0	TL–0.7 WS*Tc+TI-0.8		ns
126	$\overline{RD}$ Deassertion to Address Not Valid	TH–1		ns

AC Electrical Characteristics - External Bus Asynchronous Timing (Continued)

Num	Characteristics	60 MHz		Unit
		Min	Max	
127	Address Valid to $\overline{RD}$ Deassertion • WS = 0 • WS > 0	Tc+TI-6 ((WS+1)*Tc+TI-6		ns
128	Input Data Hold Time to $\overline{RD}$ Deassertion	0		ns
129	$\overline{RD}$ Assertion Width • WS = 0 • WS > 0	Tc-4 ((WS+1)*Tc)-4		ns
130	Address Valid to Input Data Valid • WS = 0 • WS > 0		Tc+TL–11	ns
131	Address Valid to $\overline{RD}$ Assertion	TL–6		ns
132	$\overline{RD}$ Assertion to Input Data Valid • WS = 0 • WS > 0		Tc+TL–11	ns
133	$\overline{WR}$ Deassertion to $\overline{RD}$ Assertion	Tc–7		ns
134	$\overline{RD}$ Deassertion to $\overline{RD}$ Assertion	Tc–2.5		ns
135	$\overline{WR}$ Deassertion to $\overline{WR}$ Assertion • WS = 0 • WS > 0	Tc–3		ns ns
136	$\overline{RD}$ Deassertion to $\overline{WR}$ Assertion • WS = 0 • WS > 0	Tc–2.5		ns ns

Notes:
1. With no external access from the DSP.
2. During external read or write access.
3. During external read-modify-write access.
4. During STOP mode - external bus will not be released and $\overline{DBG}$ will not go low.
5. During WAIT mode.



Note:
During Read-Modify-Write instructions, the address lines do not change state.

Figure 14-49. External Bus Asynchronous Timing

AC Electrical Characteristics - External Bus Synchronous Timing

Num	Characteristics	60 MHz		Unit
		Min	Max	
140	CKOUT transition #1 To Address Valid		7	ns
141	CKOUT transition #2 To $\overline{WR}$ Assertion • WS = 0 (see Note) • WS > 0		4	ns ns
142	CKOUT transition #2 To $\overline{WR}$ Deassertion	1	5	ns
143	CKOUT transition #2 To $\overline{RD}$ Assertion		3.9	ns
144	CKOUT transition #2 To $\overline{RD}$ Deassertion	0	3	ns
145	CKOUT transition #1 To Data-Out Valid		5	ns
146	CKOUT transition #1 To Data-Out Invalid(See Note)	0		ns
147	Data-In Valid To CKOUT transition #2 (Setup)		5	ns
148	CKOUT transition #2 To Data-In Invalid (Hold)	0		ns
149	CKOUT transition #1 To Address Invalid See Note)	0		ns

Notes:

1. AC timing specifications which are referenced to a device input signal are measured in production with respect to the 50% point of the respective input signal's transition.
2. WS are wait state values specified in the BCR.
3. CKOUT transition #1 to data-out invalid (spec. T146) and CKOUT transition #1 to address invalid (spec. T149) indicate the time after which data/address are no longer guaranteed to be valid.
4. Timings are given from CKOUT midpoint to VOL or VOH of the corresponding pin(s).
5. CKOUT transition #1 is a falling edge of CKOUT for CKP = 0.
6. During read-modify-write instructions, the address lines do not change states.

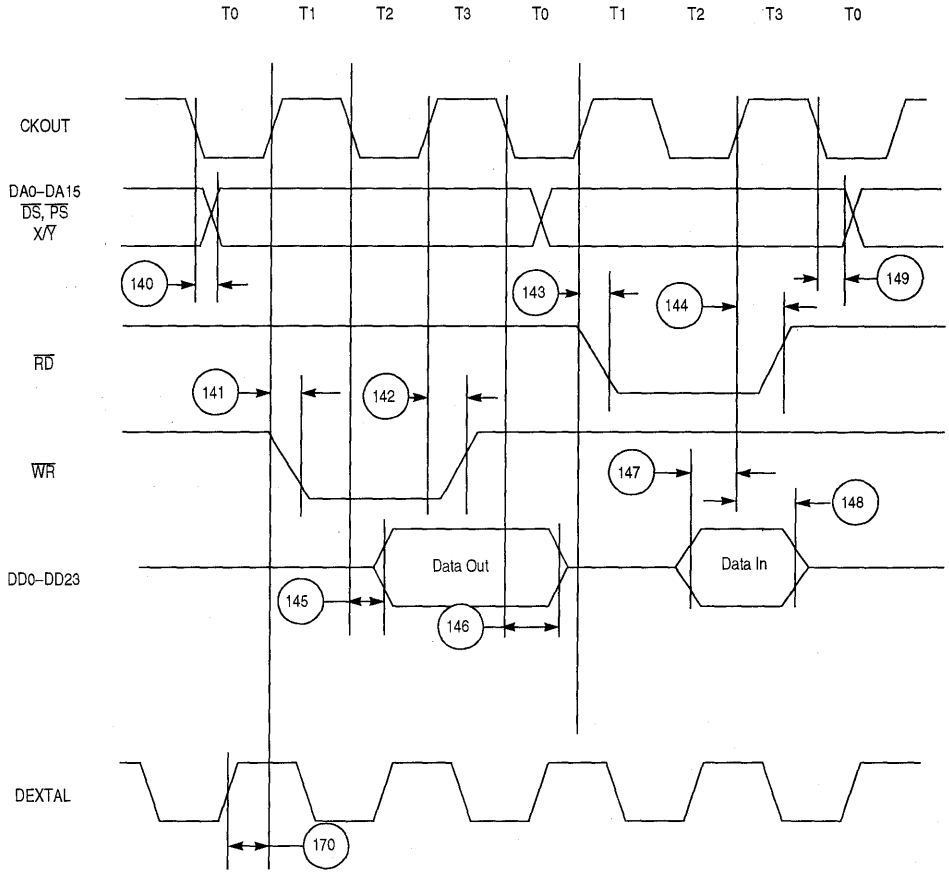
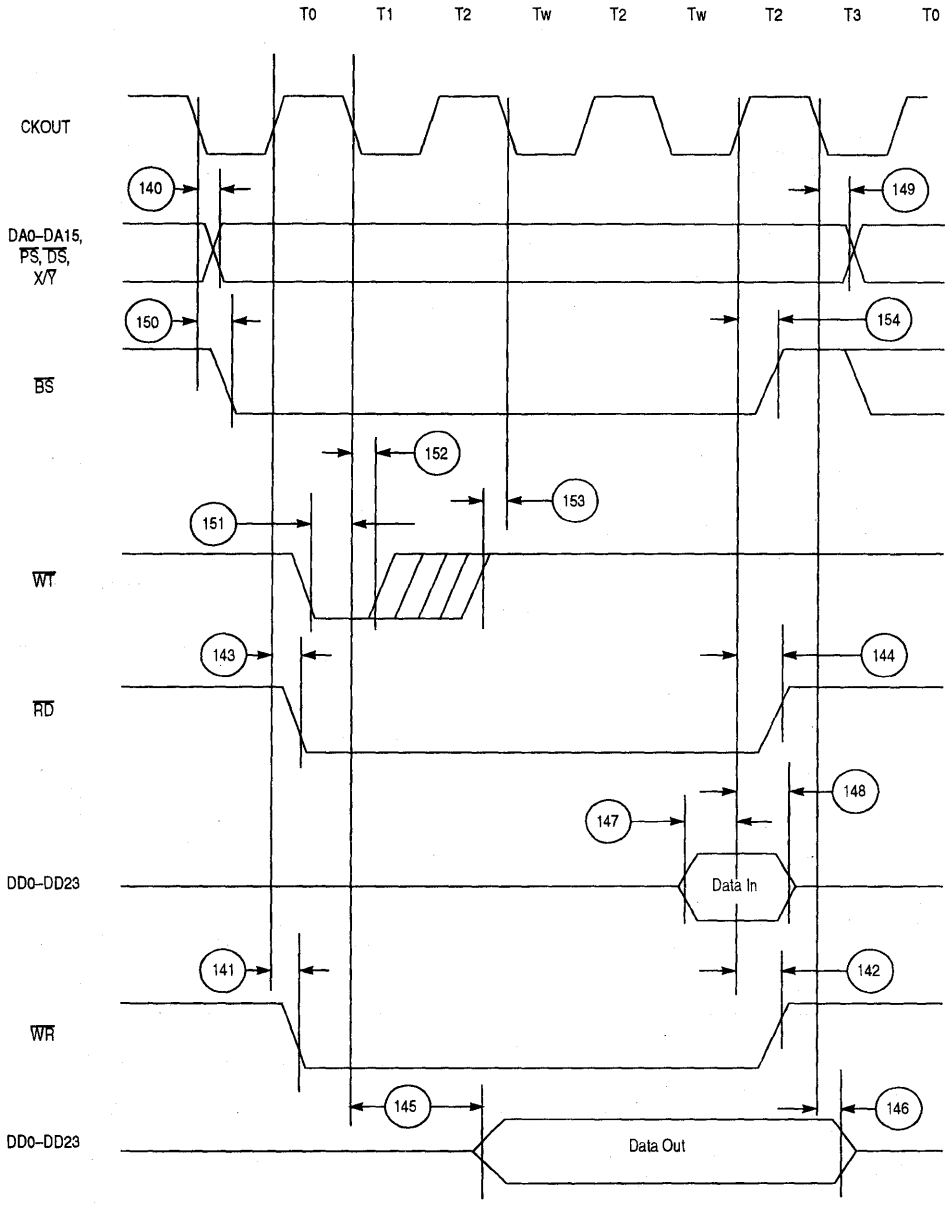


Figure 14-50. DSP56002 Synchronous Bus Timing

14.6.10 AC Electrical Characteristics - Bus Strobe / Wait Timing

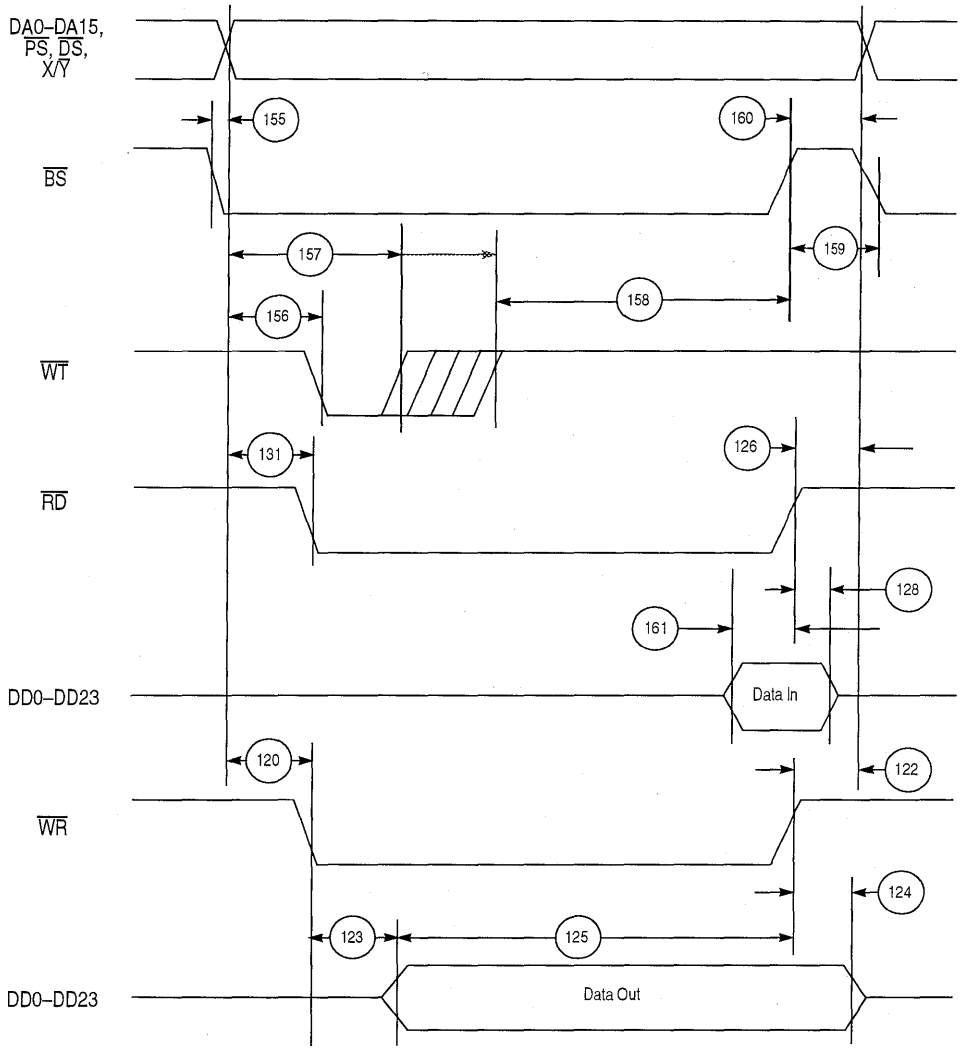
Num	Characteristics	60 MHz		Unit
		Min	Max	
150	CKOUT transition #1 To $\overline{BS}$ Assertion	—	5.6	ns
151	$\overline{WT}$ Assertion To CKOUT transition #1 (setup time)	5.3	—	ns
152	CKOUT transition #1 To $\overline{WT}$ Deassertion For Minimum Timing	0	Tc–7.9	ns
153	$\overline{WT}$ Deassertion To CKOUT transition #1 For Maximum Timing (2 wait states)	7.9	—	ns
154	CKOUT transition #2 To $\overline{BS}$ Deassertion	—	5.2	ns
155	$\overline{BS}$ Assertion To Address Valid	0	2.4	ns
156	$\overline{BS}$ Assertion To $\overline{WT}$ Assertion (See Note 2)	0	Tc–10.9	ns
157	$\overline{BS}$ Assertion To $\overline{WT}$ Deassertion (See Note 2 and Note 4)	(WS–1)*Tc	WS*Tc–13.5	ns
158	$\overline{WT}$ Deassertion To $\overline{BS}$ Deassertion	Tc+TL+3.3	2*Tc+TL+7.8	ns
159	Minimum $\overline{BS}$ Deassertion Width For Consecutive External Accesses	TH–1	—	ns
160	$\overline{BS}$ Deassertion To Address Invalid (see Note 3)	TH–4.6	—	ns
161	Data-In Valid To $\overline{RD}$ Deassertion (Set Up)	3.4	—	ns
162	$\overline{DBR}$ Assertion To CKOUT transition #2 For Min. Timing	9.5	Tc	ns
163	$\overline{DBR}$ Deassertion to CKOUT Transition #2 For Min. Timing	8	Tc	ns
164	CKOUT transition #1 to DBG Assertion	—	8.8	ns
165	CKOUT transition #1 to DBG Deassertion	—	5.3	ns
170	DEXTAL to CKOUT - PLL Disabled DEXTAL to CKOUT - PLL Enabled and MF < 5 (see note 6)	3 0.3	9.7 3.7	ns ns

Notes:
1. AC timing specifications which are referenced to a device input signal are measured in production with respect to the 50% point of the respective input signal's transition.
2. If wait states are also inserted using the BCR and if the number of wait states is greater than 2, then specification numbers 156 and 157 can be increased accordingly.
3. $\overline{BS}$ deassertion to address invalid indicates the time after which the address are no longer guaranteed to be valid.
4. The minimum number of wait states when using $\overline{BS}/\overline{WT}$ is two (2).
5. For read-modify-write instructions, the address lines will not change states between the read and the write cycle. However, $\overline{BS}$ will deassert before asserting again for the write cycle. If wait states are desired for each of the read and write cycle, the $\overline{WT}$ pin must be asserted once for each cycle.
6. When DEXTAL frequency is less than 33 MHz, then timing 170 is not guaranteed for a period of 1000 Tc after PLOCK assertion following the events below:
   – when enabling the PLL operation by software.
   – when changing the multiplication factor.
   – when recovering from the stop state if the PLL was turned off and it is supposed to turn on when exiting the stop state.



Note: During read-modify-write instructions, the address lines do not change state. However, BS will deassert before asserting again for the write cycle.

Figure 14-51. DSP56002 Synchronous BS / WT Timings



Note: During read-modify-write instructions, the address lines do not change state. However, BS will deassert before asserting again for the write cycle.

Figure 14-52. DSP56002 Asynchronous BS / WT Timings

14.6.11 AC Electrical Characteristics - OnCE Timing

Num	Characteristics	60 MHz		Unit
		Min	Max	
230	DSCK Low	40	—	ns
231	DSCK High	40	—	ns
232	DSCK Cycle Time	200	—	ns
233	$\overline{DR}$ Asserted to DSO ($\overline{ACK}$) Asserted	5Tc	—	ns
234	DSCK High to DSO Valid	—	42	ns
235	DSCK High to DSO Invalid	3	—	ns
236	DSI Valid to DSCK Low (Setup)	15	—	ns
237	DSCK Low to DSI Invalid (Hold)	3	—	ns
238	Last DSCK Low to OS0–OS1, $\overline{ACK}$ Active	3Tc+T_L	—	ns
239	DSO ($\overline{ACK}$) Asserted to First DSCK High	2Tc	—	ns
240	DSO ($\overline{ACK}$) Assertion Width	4Tc+T_H–3	5Tc+7	ns
241	DSO ($\overline{ACK}$) Asserted to OS0–OS1 High Impedance (see Note 2)	—	0	ns
242	OS0–OS1 Valid to CKOUT transition #2	Tc–21	—	ns
243	CKOUT transition #2 to OS0–OS1 Invalid	0	—	ns
244	Last DSCK Low of Read Register to First DSCK High of Next Command	7Tc+10	—	ns
245	Last DSCK Low to DSO Invalid (Hold)	3	—	ns
246	$\overline{DR}$ Assertion to CKOUT transition #2 for Wake Up from WAIT State	12	Tc	ns
247	CKOUT transition #2 to DSO After Wake Up from WAIT State	17Tc		ns
248	$\overline{DR}$ Assertion Width •a to recover from WAIT •b to recover from WAIT and enter DEBUG mode	 15 13Tc+15	 12Tc–15 —	ns
249	$\overline{DR}$ Assertion to DSO ($\overline{ACK}$) Valid (Enter Debug Mode) After Asynchronous Recovery from WAIT State	17Tc	—	ns
250A	$\overline{DR}$ Assertion Width to Recover from STOP • Stable External Clock, OMR bit 6 = 0 • Stable External Clock, OMR bit 6 = 1 • Stable External Clock, PCTL bit 17 = 1 (See Note 1)	 15 15 15	 65548Tc+T_L 20Tc+T_L 13Tc+T_L	ns

Num	Characteristics	60 MHz		Unit
		Min	Max	
250B	$\overline{DR}$ Assertion Width to Recover from STOP and enter DEBUG mode • Stable External Clock,OMR bit 6 = 0 • Stable External Clock,OMR bit 6 = 1 • Stable External Clock,PCTL bit 17 = 1 (See Note 1)	65549Tc+T_L 21Tc+T_L 14Tc+T_L	— — —	ns
251	$\overline{DR}$ Assertion to DSO ($\overline{ACK}$) Valid (Enter Debug Mode) After Recovery from STOP State • Stable External Clock, OMR bit 6 = 0 • Stable External Clock, OMR bit 6 = 1 • Stable External Clock, PCTL bit 17 = 1 (See Note 1)	65553Tc+T_L 25Tc+T_L 18Tc+T_L	— — —	

Notes:
1. A clock stabilization delay is required when using the on-chip crystal oscillator in two cases:
   1) after power-on reset, and
   2) when recovering from Stop mode.
   During this stabilization period, Tc, T_H and T_L will not be constant. Since this stabilization period varies, a delay of 75,000*Tc is typically allowed to assure that the oscillator is stable before executing programs. While it is possible to set OMR bit 6 = 1 when using the internal crystal oscillator, it is not recommended and these specifications do not guarantee timings for that case.
2. The maximum specified is periodically sampled and not 100% tested.

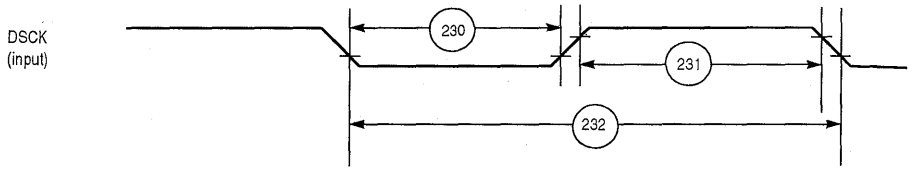


Figure 14-53. DSP56002 OnCE Serial Clock Timing

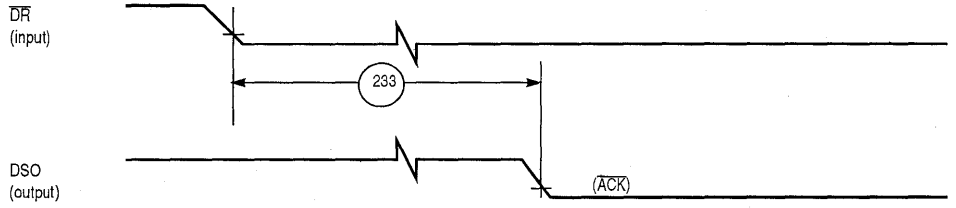
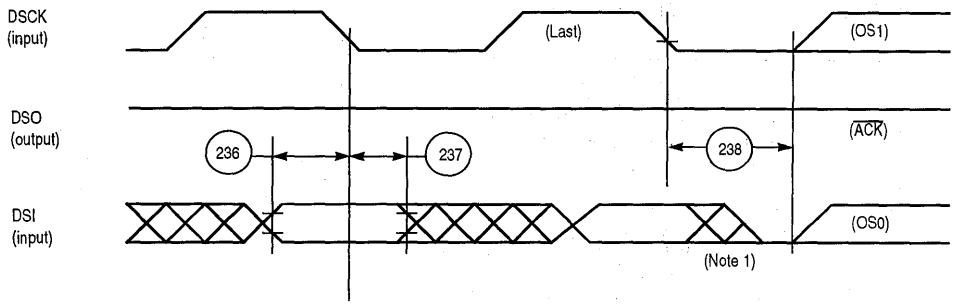
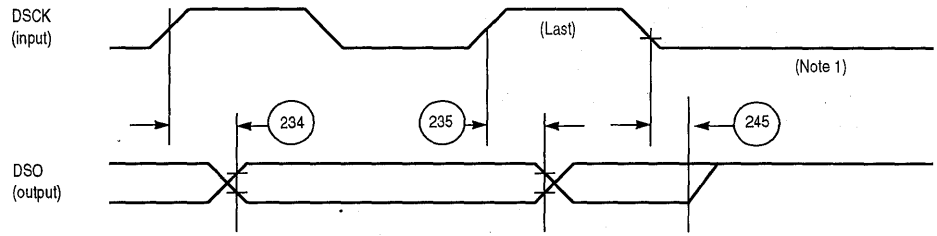


Figure 14-54. DSP56002 OnCE Acknowledge Timing



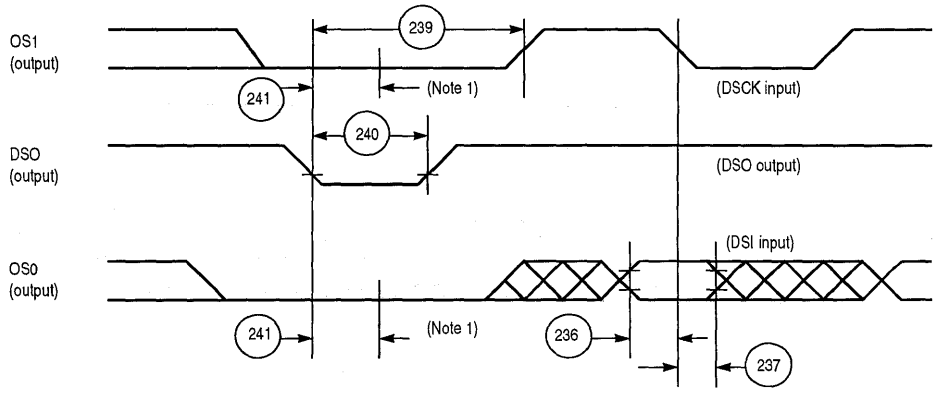
Note:
1. High Impedance, external pull-down resistor

Figure 14-55. DSP56002 OnCE Data I/O To Status Timing



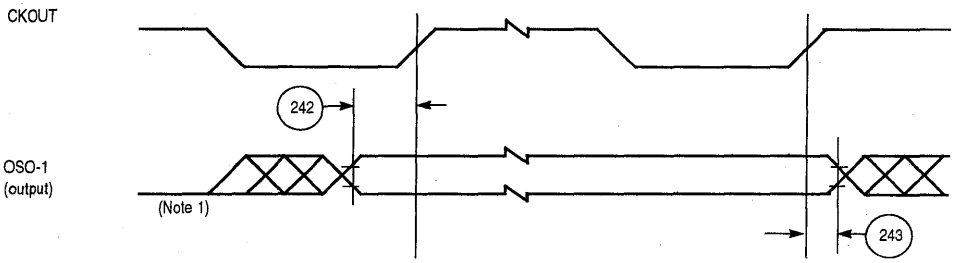
Note:
1. High Impedance, external pull-down resistor

Figure 14-56. DSP56002 OnCE Read Timing



Note:
1. High Impedance, external pull-down resistor

Figure 14-57. DSP56002 OnCE Data I/O To Status Timing



Notes:
High Impedance, external pull-down resistor

Figure 14-58. DSP56002 OnCE CKOUT To Status Timing

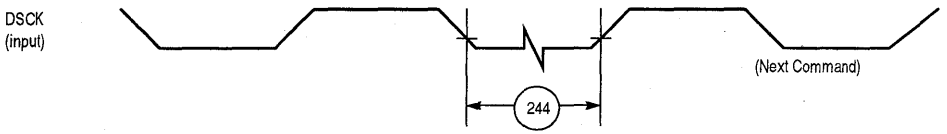


Figure 14-59. After Read Register Timing DSP56002 OnCE DSCK Next Command

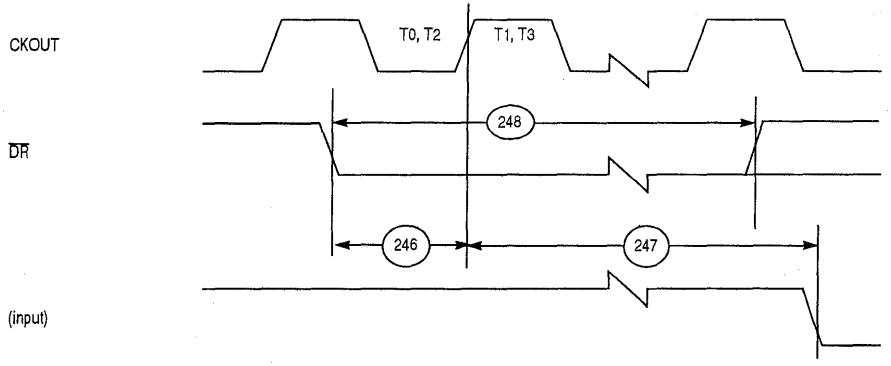


Figure 14-60. Synchronous Recovery from WAIT State

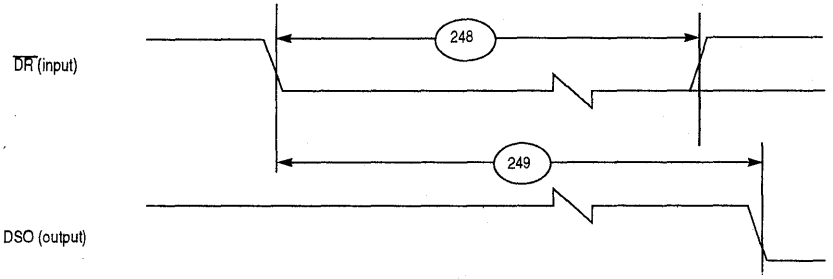


Figure 14-61. Asynchronous Recovery from WAIT State

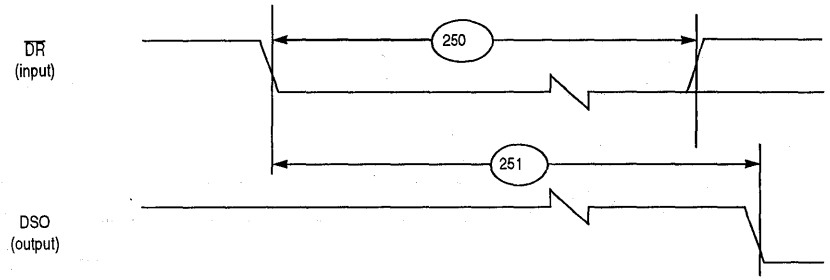


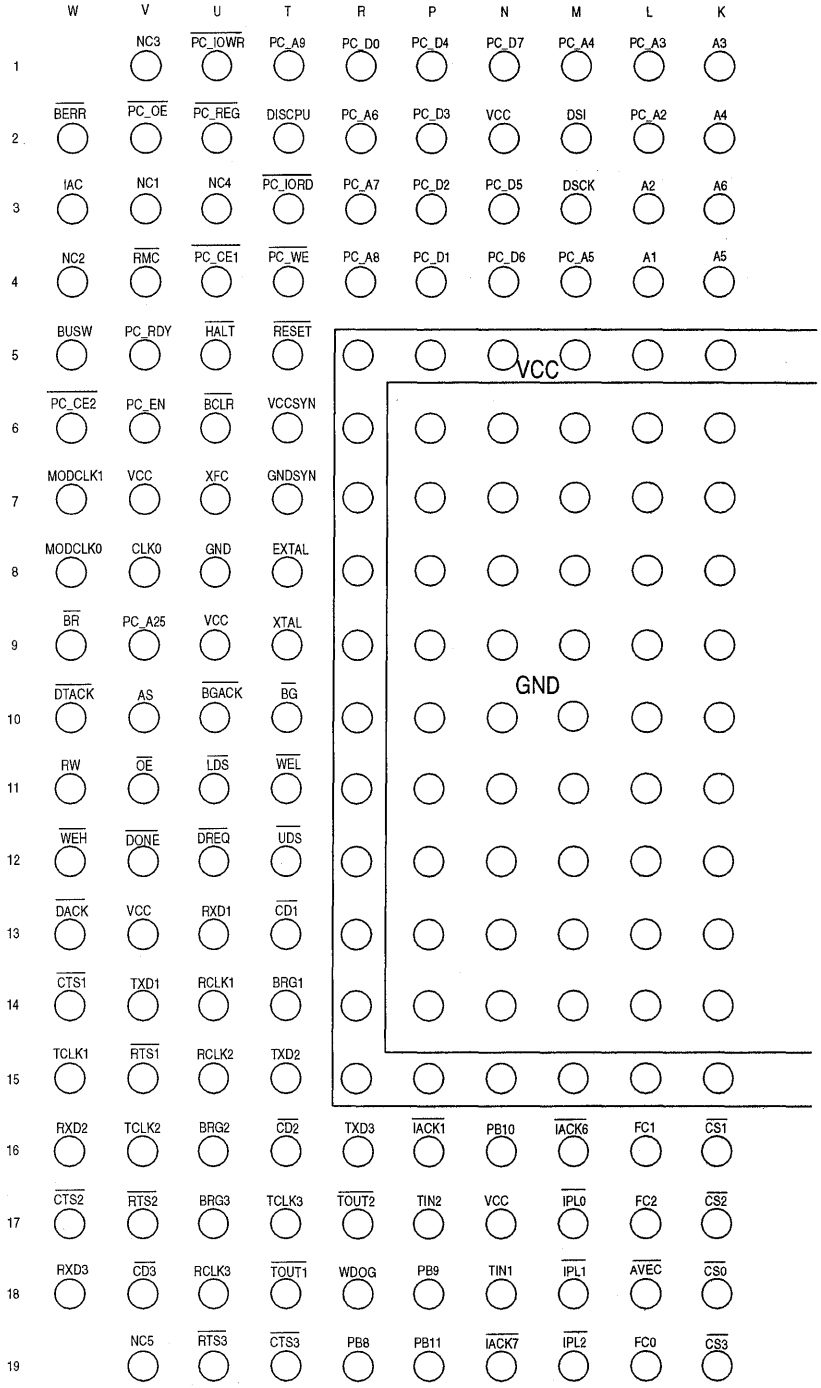
Figure 14-62. Asynchronous Recovery from STOP State

SECTION 15
MECHANICAL DATA AND ORDERING INFORMATION

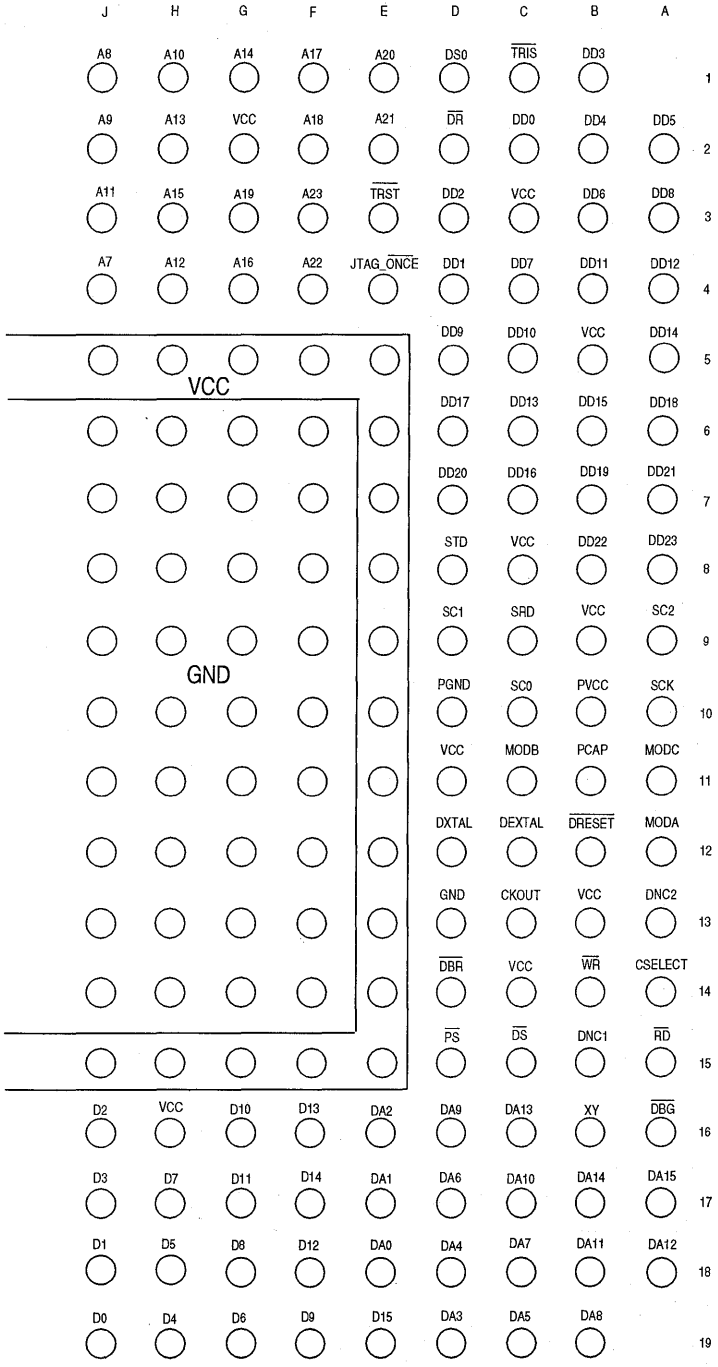
15.1 ORDERING INFORMATION

Package Type	Frequency (MHz)	Temperature	Order Number
Plastic Ball Grid Array (ZP Suffix) Plastic Ball Grid Array (CZP Suffix)	25(IMP), 60(DSP)	0°C to 70°C –40°C to + 85°C	MC68356ZP25 MC68356CZP25

15.2 PIN ASSIGNMENTS – PBGA-TOP VIEW



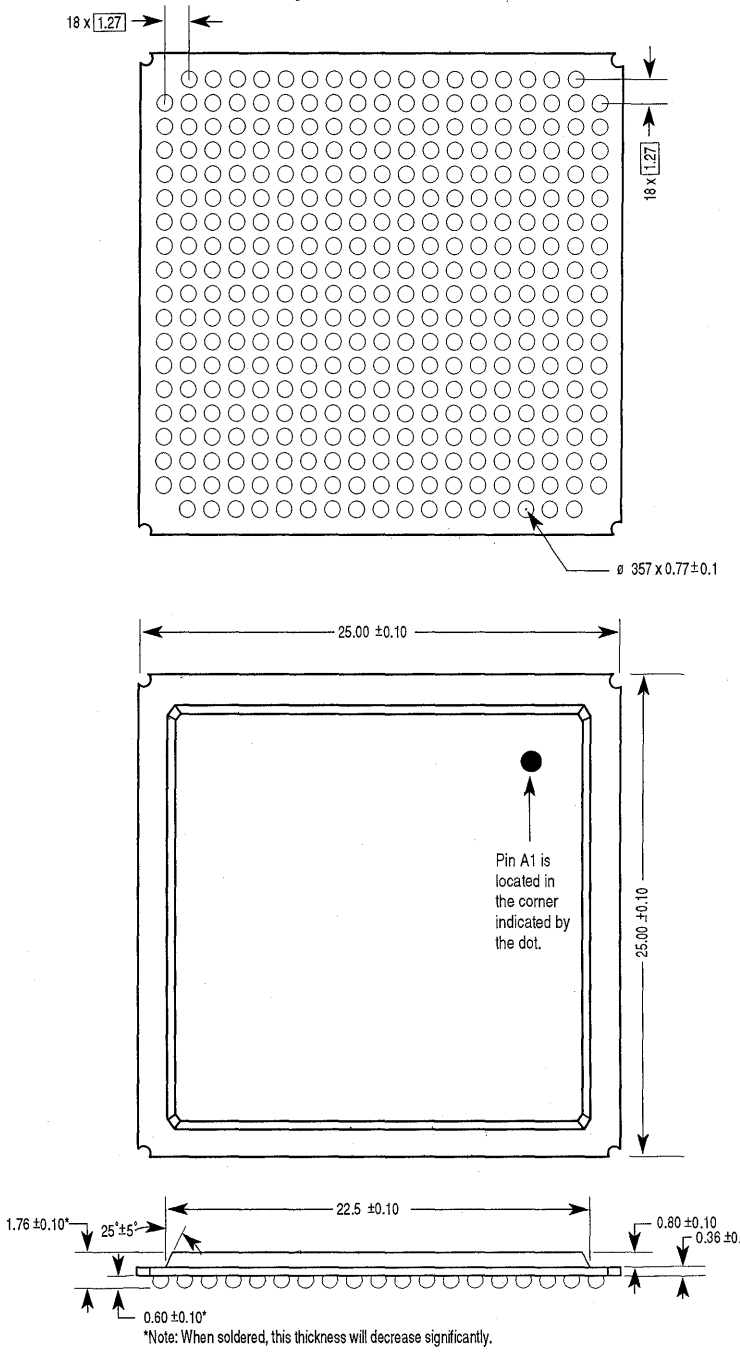
Top View


J H G F E D C B A
A8 A10 A14 A17 A20 DS0 TRIS DD3 1
A9 A13 VCC A18 A21 DR DD0 DD4 DD5 2
A11 A15 A19 A23 TRST DD2 VCC DD6 DD8 3
A7 A12 A16 A22 JTAG_ONCE DD1 DD7 DD11 DD12 4
DD9 DD10 VCC DD14 5
VCC
DD17 DD13 DD15 DD18 6
DD20 DD16 DD19 DD21 7
STD VCC DD22 DD23 8
SC1 SRD VCC SC2 9
GND
PGND SC0 PVCC SCK 10
VCC MODB PCAP MODC 11
DXTAL DEXTAL DRESET MODA 12
GND CKOUT VCC DNC2 13
DBR VCC WR CSELECT 14
PS DS DNC1 RD 15
D2 VCC D10 D13 DA2 DA9 DA13 XY DBG 16
D3 D7 D11 D14 DA1 DA6 DA10 DA14 DA15 17
D1 D5 D8 D12 DA0 DA4 DA7 DA11 DA12 18
D0 D4 D6 D9 D15 DA3 DA5 DA8 19

15

15.3 PACKAGE DIMENSIONS – PLASTIC BALL GRID ARRAY (PBGA)

For more information on the printed circuit board layout of the PBGA package, including thermal via design and suggested pad layout, please refer to AN-1231/D, "Plastic Ball Grid Array Application Note" available from your local Motorola sales office.



APPENDIX A
SCC PERFORMANCE

Since the MC68356 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 10-MHz serial clock for a 25-MHz IMP system clock. At this same 25-MHz system clock speed, the MC68356 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 MC68356 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.)

If the above restriction is adhered to, Table A-1 shows experimental performance data obtained with the device.

The frequency ratio stated represents the system (CLKO) 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 IMP system clock of 25 MHz can support an MC68356 serial rate of 2.5 Mbps. Typically, this 2.5-Mbps rate would be achieved with 1 bit every 2.5-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 2.5 Mbps.

Table A-1. Experimental SCC Data

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 1:9 1:10	UART 2 UART 2 UART BISYNC HDLC HDLC 2 HDLC 2 HDLC 2 HDLC	1:20 (*16) 1:396 (*16) 1:10 (*16) 1:98 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	UART UART UART BISYNC BISYNC HDLC HDLC	1:329 (*16) 1:305 (*16) 1:24 (*16) 1:496 1:177 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
1 Transp	1:10	—	—	
2 Transp	1:23	—	—	
3 Transp	1:35	—	—	

NOTES:

1. SCC performance calculation example with a 25-MHz IMP system clock:
   One HDLC channel can operate with a ratio of 1:9. Thus 25-MHz/9 gives 2.8 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. When the performance of a high-speed channel together with a low-speed channel was measured, the high-speed channel was always SCC1.
6. 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 25-MHz dock, SCC1 can run at 2.8 Mbps; SCC2 and SCC3 can run at 112 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).
7. 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.
8. All results assume the DRAM refresh controller is not operating; otherwise, performance is slightly reduced.

9. 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-, or async-framed channels, the user can also use the MC68356 IMP 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, SUPPORT AND RAM MICROCODE

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

By running the modules on the development board described in B.5 ADS302 Development System, it is possible to examine and evaluate the MC68302. Symbolic, user-friendly menus allow control of all parts of the IMP.

- 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.

B.2 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, as well as software tools specifically designed to work with the Motorola DADS board.

Similarly, the DSP is based on a full DSP56002 core, existing compilers, source-level debuggers, assemblers, and linkers designed for the DSP56000 Family may be successfully used. Many third parties supply such tools.

B.3 THIRD PARTY EMULATOR SUPPORT

Full in-circuit emulation support is available from multiple third parties, but is not discussed in this manual.

B.4 DADS DEVELOPMENT SYSTEM

The MC68356ADS is an integrated Applications Development System (ADS) designed to aid hardware and software developers of the MC68356 in quickly evaluating and developing applications for the MC68356. All of the hardware resources needed to download and debug application software are provided, such as large blocks of flash and static RAM for both processors, serial ports, clock generation circuitry, logic analyzer connectors, as well as a 56K OnCE port command converter and IMP(68302) monitor/debugger hardware and software.

With its on-board modem interface circuitry, the MC68356ADS provides an excellent platform for modem application development.

To serve as a convenient platform for software development, the MC68356ADS is provided with two separate monitor/debuggers: one for the integrated multiprotocol processor (IMP) section and one for the DSP56002 OnCE port interface. Both monitor/debuggers provide the operations of memory dump and set (with optional disassembly of 68K or 56K instructions), single instruction execution, breakpoints, and downloads. These two debugger interfaces can work together in separate Windows DOS shells on the same x86 based PC to communicate with the on-board IMP and DSP hardware. Future support for SUN platforms is planned.

The MC68356ADS also includes a OnCE port command converter. In addition to being able to access the on-board MC68356 DSP, the five-pin-interface from the command converter can also be connected to an off-board DSP56K family DSP through a special connector. This feature enables the user to use the command converter to interface to their target applications, as well as allowing the option of connecting the on-board MC68356 to external OnCE port command converters supplied by third-party vendors.

All the pins of the MC68356 are available unbuffered to the user through the expansion connectors and the logic analyzer connectors.

B.4.1 MC68356 APPLICATION DEVELOPMENT SYSTEM FEATURES

- General Features
  - —Onboard IMP(68302) Debugger Software with Host Debugger Interface.
  - —Command Converter Logic On Board Allowing DSP Emulation Functions through OnCE Port with Host Debugger Interface.
  - —Separate External Clock Generators for the IMP(68302) and the DSP(56002)
  - —PCMCIA Port Connector with Extender Card to Plug Directly into PCMCIA Sockets.
  - —Expansion Connectors Providing all the MC68356 Signals.
  - —DSP and 68000 Bus Signals Brought Out to Separate Logic Analyzer Connectors.
  - —Single +5Vdc Power Supply with Onboard 5V to +/-12Volt Converter.
- IMP(68302) Support Features
  - —CPM68000 Core Running at 25 MHz.
  - —512 Kbyte SRAM, 16-Bits Wide, 0 Wait-States.
  - —Option for Additional 512 Kbyte SRAM.
  - —1 Mbyte Flash Memory, 16-B its Wide.
  - —2 Kbyte EEPROM.
  - —MC68681 DUART.
  - —Serial RS-232 Terminal Connect.
  - —ADI Port Connector.

- DSP56002 Support Features
  - —DSP56000 CPU Running at 60 MHz.
  - —64 Kword Fast Static RAM.
  - —Sockets for 32 Kword Flash EPROM.
  - —OnCE Port Connector for Easy Hookup to the Command Converter on the Board or to Other Vendor's OnCE Controller.
  - —AES/EBU Connector.
- V.34 Compatible Modem Interface
  - —SGS-7544 Codec.
  - —Modem Data Access Arrangement (DAA) and PSTN Interface.
  - —On-Board Modem Speaker.
  - —Supports Ring Detect, Pulse Dial, Line Current Detect, Analog Loopback Paths.
  - —Low noise V.34 Line Interface Transformer.
  - —RJ-11 Connector

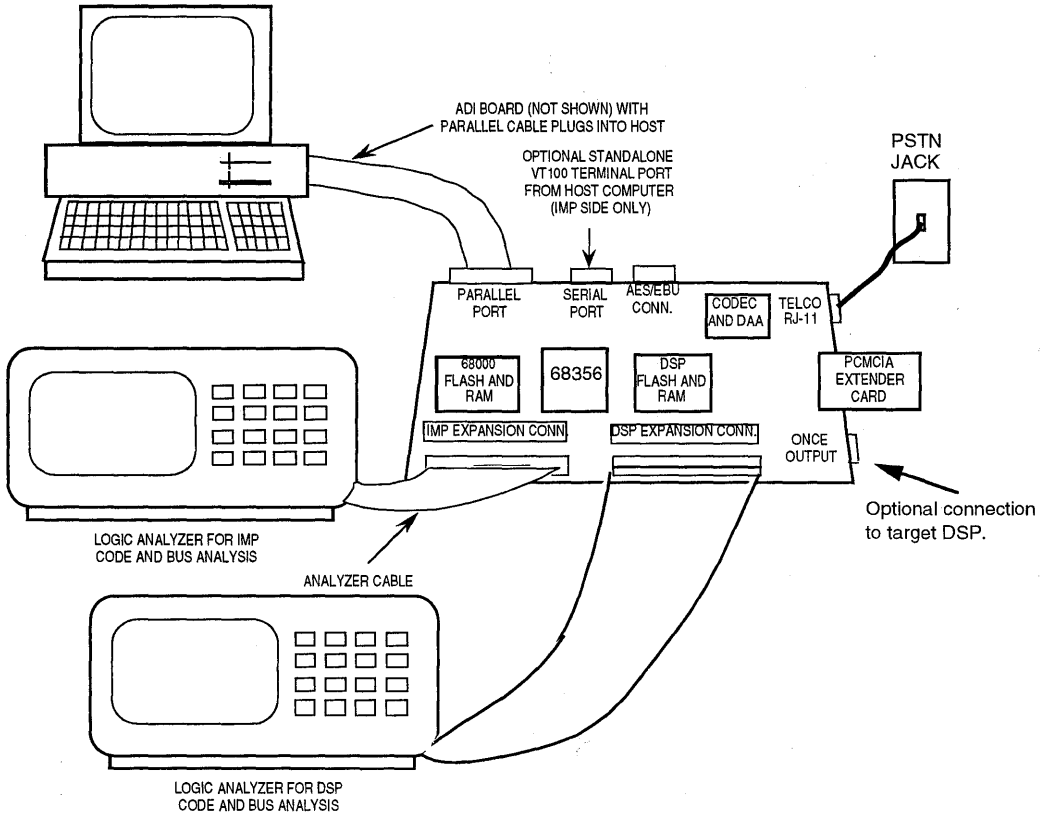


Figure B-1. MC68356ADS

MC68356 Application Development System Part Number List

Part Number	Description
M68356ADS	M68356 Development System
M68356ADI - PC	M68356 I/F Board for IBM PC
M68356ADI - SUN4	M68356 I/F Board for SUN 4

B.5 RISC MICROCODE FROM RAM

The MC68356 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.

B

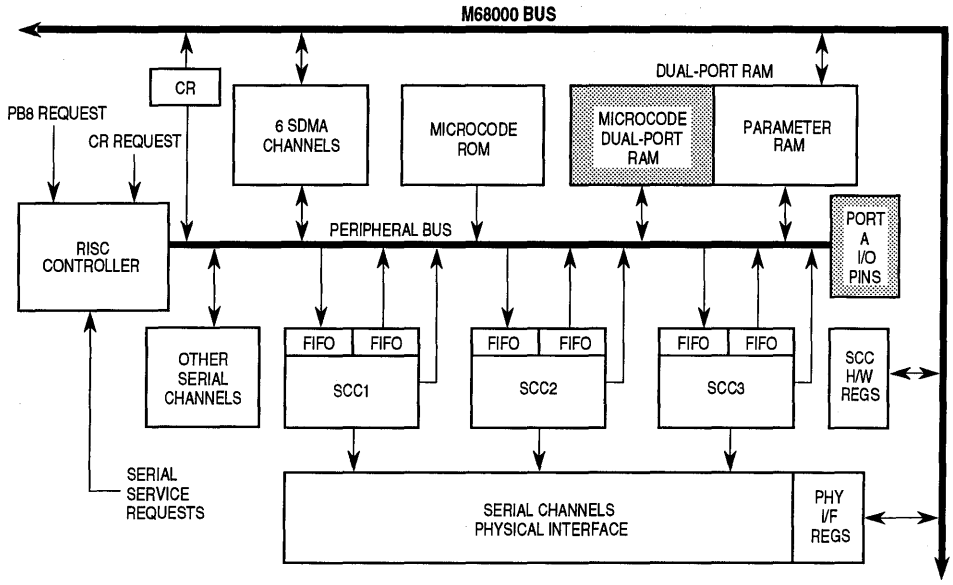


Figure B-2. CP Architecture Running RAM Microcode

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 $0F5. When the user writes a one to bit #7 of location $0F5 (the RME bit in the SCR), 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.

B.5.1 Microcode from RAM Initialization Sequence

1. Perform a total system reset of the IMP.
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. Set bit #7 at address $0F5 in supervisory space (the RME bit in the SCR register).
5. Write a software reset command to the CR.
6. Continue with the normal initialization sequence.

APPENDIX C
DSP BOOTSTRAP PROGRAM

C

Figure C-1. DSP Bootstrap Program (Part 1 of 4)

```
BOOTSTRAP CODE FOR 68356M - (C) Copyright 1994 Motorola Inc.
; Revised June, 28 1994.
;
; Bootstrap through the Host Interface, External EPROM or SCI.
;
; This is the Bootstrap program contained in the 68356M 64-word Boot
; ROM. This program can load the internal program RAM from an external
; EPROM or the Host Interface, and may load any program RAM segment
; from the SCI serial interface.
;
; If MC:MB:MA=001, then it loads the internal PRAM from 3*5,376 consecutive
; byte-wide P memory locations, starting at P:$C000 (bits 7-0). These
; will be condensed into 5,376 24-bit words and stored in contiguous PRAM
; memory locations starting at P:$0. After assembling one 24-bit word, it
; stores the result in internal PRAM memory. Note that the routine loads
; data starting with the least significant byte of P:$0.
;
; If MC:MB:MA=10x, then it loads the internal PRAM from the Host
; Interface, starting at P:$0. If only a portion of the P memory is to
; be loaded, the Host Interface bootstrap load program may be stopped by
; setting the Host Flag 0 (HF0). This will start execution of the loaded
; program at location P:$0 of the internal PRAM.
;
; If MC:MB:MA=11x, then it loads the program RAM from the SCI interface.
; The number of program words to be loaded and the starting address must
; be specified. The SCI bootstrap code expects to receive 3 bytes
; specifying the number of program words, 3 bytes specifying the address
; to start loading the program words and then 3 bytes for each program
; word to be loaded. The number of words, the starting address and the
; program words are received least significant byte first followed by the
; mid and then by the most significant byte. After receiving the
; program words, program execution starts in the same address where
; loading started. The SCI is programmed to work in asynchronous mode
; with 8 data bits, 1 stop bit and no parity. The clock source is
; external and the clock frequency must be 16x the baud rate.
; After each byte is received, it is echoed back through the SCI
; transmitter.
```

Figure C-2. DSP Bootstrap Program (Part 2 of 4)

```
BOOT            EQU     $C000           ; this is the location in P memory
                                        ; on the external memory bus
                                        ; where the external byte-wide
                                        ; EPROM would be located

PBC             EQU     $FFE0           ; Port B Control Register
HSR             EQU     $FFE9           ; Host Status Register
HRX             EQU     $FFEB           ; Host Receive Register

PCC             EQU     $FFE1           ; Port C Control Register
SCR             EQU     $FFF0           ; SCI Control Register
SSR             EQU     $FFF1           ; SCI Status Register
SCCR            EQU     $FFF2           ; SCI Clock Control Register
SRXL            EQU     $FFF4           ; SCI Receive Register Low
STXL            EQU     $FFF4           ; SCI Transmit Register Low
P_SIZE          EQU     $1500           ; Internal P_RAM size
                ORG     PL:$0,PL:$0     ; bootstrap code starts at $0

START           MOVE    #<0,R0          ; default P address where prog
                                        ; will begin loading
                MOVE    #P_SIZE,B1      ; B1 will keep the number of
                                        ; words to be loaded through Host
                JCLR    #4,OMR,EPROMLD  ; If MC:MB:MA=0xx, go load from
                                        ; EPROM
                JSET    #1,OMR,SCILD    ; If MC:MB:MA=11x, go load from
                                        ; SCI

; This is the routine that loads from the Host Interface.
; MC:MB:MA=100 - reserved
; MC:MB:MA=101 - Host

HOSTLD          BSET    #0,X:PBC        ; Configure Port B as Host
                DO      B1,_LOOP3       ; Load P_SIZE instruction words
_LBLA           JCLR    #3,X:HSR,_LBLB  ; if HF0=1, stop loading data.
                ENDDO                   ; Must terminate the do loop
                JMP     <_LOOP3

_LBLB           JCLR    #0,X:HSR,_LBLA  ; Wait for HRDF to go high
                                        ; (meaning data is present).
                MOVEP   X:HRX,P:(R0)+   ; Store 24-bit data in P mem.
_LOOP3                                  ; and go get another 24-bit word.
                JMP     <FINISH         ; finish bootstrap
```

Figure C-3. DSP Bootstrap Program (Part 3 of 4)

```
; This is the routine that loads from external EPROM.
; MC:MB:MA=001

EPROMLD         MOVE    #BOOT,R1        ; R1 = Ext address of EPROM
                DO      B1,_LOOP1       ; Load P_SIZE instruction words
                DO      #3,_LOOP2       ; Each instruction has 3 bytes
                MOVEM   P:(R1)+,A2      ; Get the 8 LSB from ext. P mem.
                REP     #8              ; Shift 8 bit data into A1
                ASR     A
_LOOP2                                  ; Get another byte.
                MOVEM   A1,P:(R0)+      ; Store 24-bit result in P mem.
_LOOP1                                  ; and go get another 24-bit word.

FINISH          MOVE    #<0,R1

; This is the exit handler that returns execution to normal
; expanded mode and jumps to the RESET vector.
BOOTEND         ANDI    #$EC,OMR        ; Set operating mode to 0
                                        ; (and trigger an exit from
                                        ; bootstrap mode).
                ANDI    #$0,CCR         ; Clear CCR as if RESET to 0.
                                        ; Delay needed for Op. Mode change
                JMP     (R1)            ; Then go to starting Prog addr.

; This is the routine that loads from the SCI.
; MC:MB:MA=110 - external SCI clock
; MC:MB:MA=111 - reserved
                ORG     PL:$0100,PL:$0100 ; starting address of 2nd ROM
SCILD           MOVEP   #$0302,X:SCR    ; Configure SCI Control Reg
                MOVEP   #$C000,X:SCCR   ; Configure SCI Clock Control Reg
                MOVEP   #7,X:PCC        ; Configure SCLK, TXD and RXD

_SCI1           DO      #6,_LOOP6       ; get 3 bytes for number of
                                        ; program words and 3 bytes
                                        ; for the starting address
                JCLR    #2,X:SSR,*      ; Wait for RDRF to go high
                MOVEP   X:SRXL,A2       ; Put 8 bits in A2
                JCLR    #1,X:SSR,*      ; Wait for TDRE to go high
                MOVEP   A2,X:STXL       ; echo the received byte
                REP     #8
                ASR     A
                MOVE    A1,R0           ; starting address for load
                MOVE    A1,R1           ; save starting address
                DO      A0,_LOOP4       ; Receive program words
```

C

Figure C-4. DSP Bootstrap Program (Part 4 of 4)

```
_LOOP6
                MOVE    A1,R0           ; starting address for load
                MOVE    A1,R1           ; save starting address
                DO      A0,_LOOP4       ; Receive program words

                DO      #3,_LOOP5
                JCLR    #2,X:SSR,*      ; Wait for RDRF to go high
                MOVEP   X:SRXL,A2       ; Put 8 bits in A2
                JCLR    #1,X:SSR,*      ; Wait for TDRE to go high
                MOVEP   A2,X:STXL       ; echo the received byte
                REP     #8
                ASR     A
_LOOP5
                MOVEM   A1,P:(R0)+      ; Store 24-bit result in P mem.
_LOOP4
                JMP     FINISH+1        ; Boot from SCI done
; End of bootstrap code. Number of program words: 64
```

INDEX

A

C

R

S

T

V

W

X

Y

1ATX31733-0 Printed in USA 11/94 BANTA CO. MOTO #61 10,000 MPU YGABAA