Welcome to E-XFL.COM

Understanding Embedded - Microprocessors

Embedded microprocessors are specialized computing chips designed to perform specific tasks within an embedded system. Unlike general-purpose microprocessors found in personal computers, embedded microprocessors are tailored for dedicated functions within larger systems, offering optimized performance, efficiency, and reliability. These microprocessors are integral to the operation of countless electronic devices, providing the computational power necessary for controlling processes, handling data, and managing communications.

Applications of Embedded - Microprocessors

Embedded microprocessors are utilized across a broad spectrum of applications, making them indispensable in

Details	
Product Status	Obsolete
Core Processor	M68000
Number of Cores/Bus Width	1 Core, 8/16-Bit
Speed	20MHz
Co-Processors/DSP	Communications; RISC CPM
RAM Controllers	DRAM
Graphics Acceleration	No
Display & Interface Controllers	-
Ethernet	-
SATA	-
USB	-
Voltage - I/O	5.0V
Operating Temperature	0°C ~ 70°C (TA)
Security Features	-
Package / Case	132-BQFP Bumpered
Supplier Device Package	132-PQFP (46x46)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc68302fc20c

LIST OF FIGURES

Section 6
Electrical Characteristics

NOTE

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

Table 2-9. Internal Registers

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

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

NOTE:

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

External Cycle Steal

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

3.1.4.5 Block Transfer Termination

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

Transfer Count Exhausted

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

NOTE

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

bus cycle after the completion of the current instruction.

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

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

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

The interrupt acknowledge cycle carries out a standard M68000 bus read cycle, except that FC2–FC0 are encoded as 111, A3–A1 are encoded with the interrupt priority level (1–7, with 7 (i.e., 111) being the highest), and A19–A16 are driven high. $\overline{UDS}$ 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 2-5), and then jumps to that 4-byte address. That 4-byte address is the location of the first instruction in the interrupt handler.

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

3.2.1.2 Interrupt Controller Overview

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

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

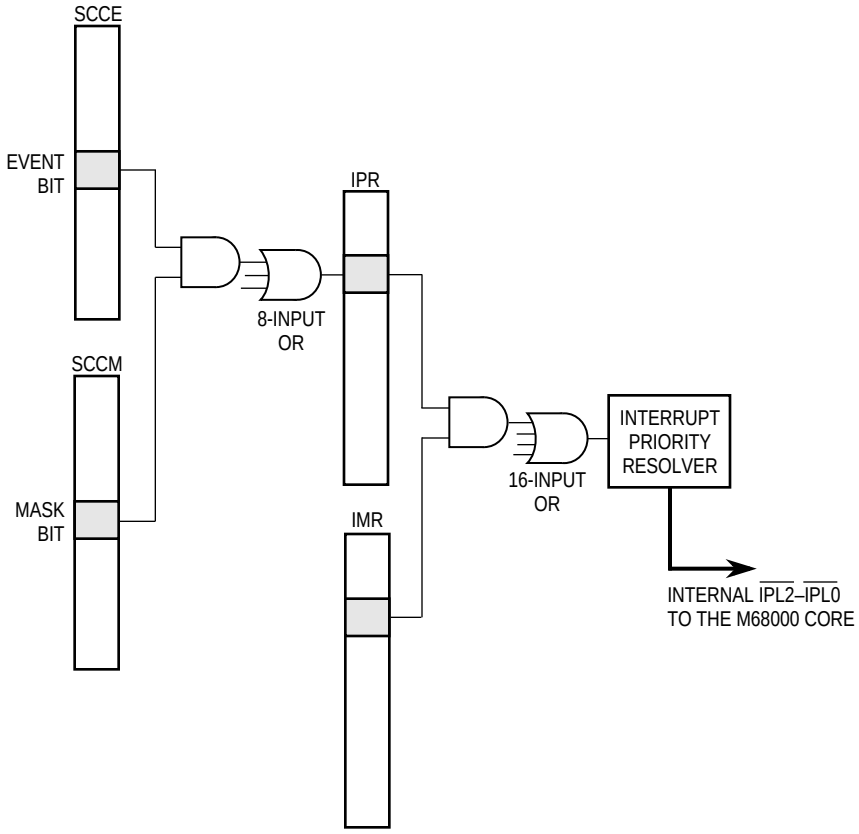


Figure 3-3. Interrupt Request Logic Diagram for SCCs

3.2.4 Interrupt Vector

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

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

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

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

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

3.8 SYSTEM CONTROL

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

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

3.8.1 System Control Register (SCR)

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

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

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

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

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

Figure 3-11. System Control Register

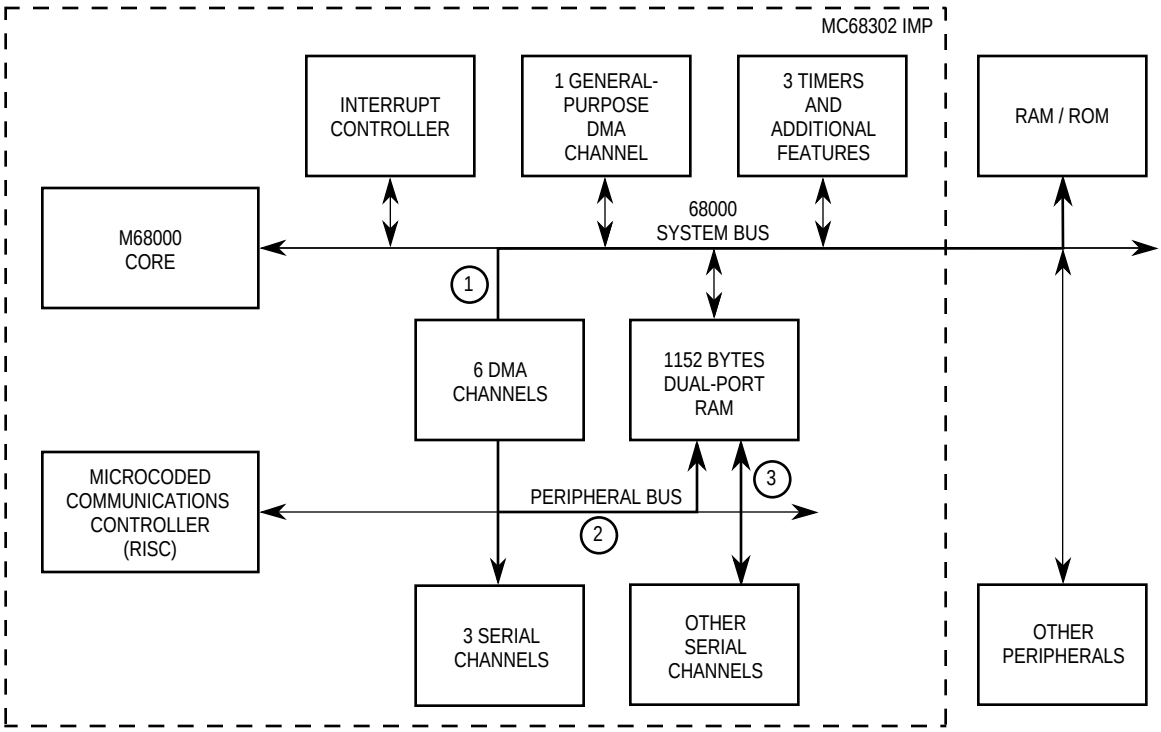


Figure 4-2. Three Serial Data Flow Paths

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

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

NOTE

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

for an internal clock, TCS and RCS may both be zero, or, for an external clock, they may both be one. The other two combinations are not allowed in this mode.

NOTE

If external loopback is desired (i.e., external to the MC68302), then the DIAG1–DIAG0 bits should be set for either normal or software operation, and an external connection should be made between the TXD and RXD pins. Clocks may be generated internally, externally, or an internally generated TCLK may be externally connected to RCLK. If software operation is used, the $\overline{\text{RTS}}$, $\overline{\text{CD}}$, and $\overline{\text{CTS}}$ pins need not be externally connected. If normal operation is used, the $\overline{\text{RTS}}$ pin may be externally connected to the $\overline{\text{CD}}$ pin, and the $\overline{\text{CTS}}$ pin may be grounded.

NOTE

Do not use this mode for loopback operation of IDL in the Serial Interface. Instead program the diag bits to Normal Operation, and (1) assert the L1GR pin externally from the S/T chip, or (2) configure the SDIAG1-0 bits in the SIMODE to Internal Loopback or Loopback Control.

10 = Automatic echo

In this mode, the channel automatically retransmits the received data on a bit-by-bit basis. The receiver operates normally, but the transmitter simply retransmits the received data. The $\overline{\text{CD}}$ pin must be asserted for the receiver to receive data, and the $\overline{\text{CTS}}$ line is ignored. The data is echoed out the TXD pin with a few nanosecond delay from RXD. No transmit clock is required, and the ENT bit in the SCC mode register does not have to be set.

NOTE

The echo function may also be accomplished in software by receiving buffers from an SCC, linking them to transmit buffer descriptors, and then transmitting them back out of that SCC.

11 = Software operation (CTS, CD lines under software control)

In this mode, the CTS and CD lines are just inputs to the SCC event (SCCE)and status (SCCS) registers. The SCC controller does not use these lines to enable/disable reception and transmission, but leaves low (i.e., active) in this mode. Transmission delays from RTS low are zero TCLKs (asynchronous protocols) or one TCLK (synchronous protocols).

NOTE

The MC68302 provides several tools for enabling and disabling transmission and/or reception. Choosing the right tool is application and situation dependent. For the receiver, the tools are 1) the empty bit in the receive buffer descriptor, 2) the ENR bit, and 3) the ENTER HUNT MODE command. For the transmitter, the

RCCR, CHARACTER

The UART controller can automatically recognize special characters and generate interrupts. It also allows a convenient method for inserting flow control characters into the transmit stream. See 4.5.11.7 UART Control Characters and Flow Control for more details.

If neither of these capabilities are desired, initialize CHARACTER1 to $8000 and CHARACTER8 to $0000 to disable both functions.

4.5.11.4 UART Programming Model

An SCC configured as a UART uses the same data structure as the other protocols. The UART data structure supports multibuffer operation. The UART may also be programmed to perform address comparison whereby messages not destined for a given programmable address are discarded. Also, the user can program the UART to accept or reject control characters. If a control character is rejected, an interrupt may be generated. The UART enables the user to transmit break and preamble sequences. Overrun, parity, noise, and framing errors are reported using the buffer descriptor (BD) table and/or error counters. An indication of the status of the line (idle) is reported through the status register, and a maskable interrupt is generated upon a status change.

In its simplest form, the UART can function in a character-oriented environment. Each character is transmitted with accompanying stop bits and parity (as configured by the user) and is received into separate one-byte buffers. Reception of each buffer may generate a maskable interrupt.

Many applications may want to take advantage of the message-oriented capabilities supported by the UART using linked buffers to receive or transmit data. In this case, data is handled in a message-oriented environment; users can work on entire messages rather than operating on a character-by-character basis. A message may span several linked buffers. For example, rather than being interrupted after the reception of each character, a terminal driver may want to wait until an end-of-line character has been typed by a user before handling the input data.

As another example, when transmitting ASCII files, the data may be transferred as messages ending on the end-of-line character. Each message could be both transmitted and received as a circular list of buffers without any intervention from the M68000 core. This technique achieves both ease in programming and significant savings in processor overhead.

On the receive side, the user may define up to eight control characters. Each control character may be configured to designate the end of a message (such as end of line) or to generate a maskable interrupt without being stored in the data buffer. This latter option is useful when flow-control characters such as XON or XOFF need to alert the M68000 core, yet do not belong to the message being received. Flow-control characters may also be transmitted at any time.

In the message-oriented environment, the data stream is divided into buffers. However, the physical format of each character (stop bits, parity, etc.) is not altered.

CD—Carrier Detect Lost

The carrier detect signal was negated during message reception.

Data Length

Data length contains the number of octets written by the CP into this BD's data buffer. It is written by the CP once as the BD is closed.

NOTE

The actual amount of memory allocated for this buffer should be greater than or equal to the contents of maximum receive buffer length register (MRBLR).

Rx Buffer Pointer

The receive buffer pointer, which always points to the first location of the associated data buffer, may be even or odd. The buffer may reside in either internal or external memory.

NOTE

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

4.5.11.15 UART Transmit Buffer Descriptor (Tx BD)

Data is presented to the CP for transmission on an SCC channel by arranging it in buffers referenced by the channel's Tx BD table. The CP confirms transmission (or indicates error conditions) through the BDs to inform the M68000 core that the buffers have been serviced. The Tx BD shown in Figure 4-22.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET + 0	R	X	W	I	CR	A	P	—	—	—	—	—	—	—	—	CT
OFFSET + 2	DATA LENGTH															
OFFSET + 4 OFFSET +6	TX BUFFER POINTER (24-bits used, upper 8 bits must be 0)															

Figure 4-22. UART Transmit Buffer Descriptor

The first word of the Tx BD contains status and control bits. The following bits are prepared by the user before transmission and set by the CP after the buffer has been transmitted.

R—Ready

0 = This buffer is not currently ready for transmission. The user is free to manipulate the BD (or its associated buffer). The CP clears this bit after the buffer has been transmitted or after an error condition has been encountered.

1 = The data buffer, which has been prepared for transmission by the user, has not been transmitted or is currently transmitting. No fields of this BD may be written by the user once this bit is set.

NOTE

This error can occur only on synchronous links.

2. Clear-To-Send Lost (Collision) During Message Transmission. When this error occurs and the channel is not programmed to control this line with software, the channel terminates buffer transmission, closes the buffer, sets the CTS lost (CT) bit in the BD, and generates the transmit error (TXE) interrupt (if enabled). The channel resumes transmission after the reception of the RESTART TRANSMIT command.

Reception Errors:

1. Carrier Detect Lost During Message Reception. When this error occurs and the channel is not programmed to control this line with software, the channel terminates message reception, closes the buffer, sets the carrier detect lost (CD) bit in the BD, and generates the receive block (RBK) interrupt (if enabled). This error has the highest priority. The rest of the message is lost, and other errors in that message are not checked.
   The channel will enter hunt mode immediately. It is possible that a SYN1–SYN2-(SOH,DLE,ENQ) sequence in data will be incorrectly interpreted as the start of the next header, but this "header" will have a CRC error.
2. Overrun Error. The DDCMP controller maintains an internal three-byte FIFO for receiving data. The CP begins programming the SDMA channel (if the data buffer is in external memory) and updating the CRC when the first word is received into the FIFO. If the receive FIFO overrun error occurs, the channel writes the received data byte to the internal FIFO on top of the previously received byte. The previous data byte is lost. Then the channel closes the buffer, sets the overrun (OV) bit in the BD, and generates the receive block (RBK) interrupt (if enabled).
   The channel will enter hunt mode immediately. It is possible that a SYN1–SYN2-(SOH,DLE,ENQ) sequence in data will be incorrectly interpreted as the start of the next header, but this “header” will have a CRC error.
3. CRC1 (Header CRC) Error. When this error occurs, the channel writes the received CRC to the data buffer, closes the buffer, sets the CRC error (CR) bit in the BD, generates the RBK interrupt (if enabled), increments the error counter (CRC1EC), and enters hunt mode.
   When this error occurs on data-and maintenance-message header fields, the channel will enter hunt mode immediately. It is possible that a SYN1–SYN2-(SOH,DLE,ENQ) sequence in data will be incorrectly interpreted as the start of the next header, but this “header” will have a CRC error.
4. CRC2 (Data or Maintenance CRC) or CRC3 (Control Message) Error. When this error occurs, the channel writes the received CRC to the data buffer, closes the buffer, sets the CRC error (CR) bit in the BD, and generates the RBK interrupt (if enabled). The channel also increments the CRC2EC counter and enters hunt mode.
5. Framing Error. A framing error is detected by the DDCMP controller when no stop bit is detected in a received data string. When this error occurs, the channel writes the received character to the buffer, closes the buffer, sets the framing error (FR) bit in the BD, and generates the RBK interrupt (if enabled). When this error occurs, parity is not checked for this character.

ler sets its corresponding bit in this register. Interrupts generated by this register may be masked in the V.110 mask register.

The V.110 event register is a memory-mapped register that may be read at any time. A bit is cleared by writing a one (writing a zero does not affect a bit's value). More than one bit may be cleared at a time. All unmasked bits must be cleared before the CP will clear the internal interrupt request signal. This register is cleared at reset.

7	6	5	4	3	2	1	0
—	—	—	TXE	RXF	BSY	TX	—

Bits 7–5, 0—Reserved for future use.

TXE—Tx Error

An error (underrun) occurred on the transmitter channel.

RXF—Receive Frame

A complete frame has been received on the V.110 channel.

BSY—Busy Condition

A data byte was received and discarded due to lack of buffers. The receiver will automatically enter hunt mode.

TX —Tx Buffer

A buffer has been transmitted.This bit is set on the second to last bit of an 80-bit frame.

4.5.15.10 V.110 Mask Register

The SCC mask register (SCCM) is referred to as the V.110 mask register when the SCC is operating as a V.110 controller. It is an 8-bit read-write register that has the same bit format as the V.110 event register. If a bit in the V.110 mask register is a one, the corresponding interrupt in the event register will be enabled. If the bit is zero, the corresponding interrupt in the event register will be masked. This register is cleared upon reset.

4.5.16 Transparent Controller

The transparent controller allows transmission and reception of serial data over an SCC without any modification to that data stream. Transparent mode provides a clear channel on which no bit-level manipulation is performed by the SCC. Any protocol run over transparent mode is performed in software. The job of an SCC in transparent mode is to function simply as a high-speed serial-to-parallel and parallel-to-serial converter. This mode is also referred to as totally transparent or promiscuous operation.

There are several basic applications for transparent mode. First, some data may need to be moved serially but requires no protocol superimposed. An example of this is voice data. Second, some board-level applications require a serial-to-parallel and parallel-to-serial conversion. Often this conversion is performed to allow communication between chips on the same board. The SCCs on the MC68302 can do this very efficiently with very little M68000 core intervention. Third, some applications require the switching of data without interfering with

4.7.4.2 SMC1 Transmit Buffer Descriptor

The CP reports information about this transmit byte through the BD.

15	14	13	12 — 10	9	8	7 — 0
R	L	AR	—	AB	EB	DATA

R—Ready

0 = This bit is cleared by the CP after transmission. The Tx BD is now available to the M68000 core.

1 = This bit is set by the M68000 core to indicate that the data byte associated with this BD is ready for transmission.

In GCI mode, when the IMP implements the monitor channel protocol, it will clear this bit after receiving an acknowledgment on the A bit. When the SMC1 data should be transmitted and this bit is cleared, the channel will retransmit the previous data until new data is provided by the M68000 core.

L—Last (EOM)

This bit is valid only in GCI mode when the IMP implements the monitor channel protocol. When this bit is set, the SMC1 channel will transmit the buffer's data and then the end of message (EOM) indication on the E bit.

AR—Abort Request

This bit is valid only in GCI mode when the IMP implements the monitor channel protocol. This bit is set by the IMP when an abort request was received on the A bit. The SMC1 transmitter will transmit EOM on the E bit.

Bits 12–10—Reserved for future use.

AB—Transmit A Bit Value

This bit is valid only in GCI mode when the monitor channel is in transparent mode.

EB—Transmit E Bit Value

This bit is valid only in GCI mode when the monitor channel is in transparent mode.

Data—Data Field

The data field contains the data to be transmitted by SMC1.

4.7.4.3 SMC2 Receive Buffer Descriptor

In the IDL mode, this BD is identical to the SMC1 receive BD. In the GCI mode, SMC2 is used to control the C/I channel.

15	14 — 6	5 — 2	1	0
E	RESERVED	C/I	0	0

These eight pins can be used either as NMSI1 in nonmultiplexed serial interface (NMSI) mode or as an ISDN physical layer interface in IDL, GCI, and PCM highway modes. The input buffers have Schmitt triggers.

Table 5-7 shows the functionality of each pin in NMSI, GCI, IDL, and PCM highway modes.

Table 5-7. Mode Pin Functions

Signal Name	NMSI1		GCI		IDL		PCM	
RXD1/L1RXD	I	RXD1	I	L1RXD	I	L1RXD	I	L1RXD
TXD1/L1TXD	O	TXD1	O	L1TXD	O	L1TXD	O	L1TXD
RCLK1/L1CLK	I/O	RCLK1	I	L1CLK	I	L1CLK	I	L1CLK
TCLK1/L1SY0	I/O	TCLK1	O	SDS1	O	SDS1	I	L1SY0
$\overline{\text{CD1}}$/L1SY1	I	$\overline{\text{CD1}}$	I	L1SYNC	I	L1SYNC	I	L1SY1
$\overline{\text{CTS1}}$/L1GR	I	$\overline{\text{CTS1}}$	I	L1GR	I	L1GR		
$\overline{\text{RTS1}}$/L1RQ	O	$\overline{\text{RTS1}}$	O	GCIDCL	O	L1RQ	O	$\overline{\text{RTS}}$
BRG1	O	BRG1	O	BRG1	O	BRG1	O	BRG1

NOTES:
1. In IDL and GCI mode, SDS2 is output on the PA7 pin.
2. $\overline{\text{CD1}}$ may be used as an external sync in NMSI mode.
3. $\overline{\text{RTS}}$ is the $\overline{\text{RTS1}}$, $\overline{\text{RTS2}}$, or RTS3 pin according to which SCCs are connected to the PCM highway.

RXD1/L1RXD—Receive Data/Layer-1 Receive Data

This input is used as the NMSI1 receive data in NMSI mode and as the receive data input in IDL, GCI, and PCM modes.

TXD1/L1TXD—Transmit Data/Layer-1 Transmit Data

This output is used as NMSI1 transmit data in NMSI mode and as the transmit data output in IDL, GCI, and PCM modes. TXD1 may be configured as an open-drain output in NMSI mode. L1TXD in IDL and PCM mode is a three-state output. In GCI mode, it is an open-drain output.

RCLK1/L1CLK—Receive Clock/Layer-1 Clock

This pin is used as an NMSI1 bidirectional receive clock in NMSI mode or as an input clock in IDL, GCI, and PCM modes. In NMSI mode, this signal is an input when SCC1 is working with an external clock and is an output when SCC1 is working with its baud rate generator. The RCLK1 output can be three-stated by setting bit 12 in the CKCR register (see 3.9 Clock Control Register).

TCLK1/L1SY0/SDS1—Transmit Clock/PCM Sync/Serial Data Strobe 1

This pin is used as an NMSI1 bidirectional transmit clock in NMSI mode, as a sync signal in PCM mode, or as the SDS1 output in IDL/GCI modes. In NMSI mode, this signal is an input when SCC1 is working with an external clock and is an output when SCC1 is working with its baud rate generator. The TCLK1 output can be three-stated by setting bit 13 in the CKCR register (see 3.9 Clock Control Register).

6.4 POWER DISSIPATION

Characteristic	Symbol	Typ	Max	Unit
Power Dissipation at 25 MHz- Rev C.8µ (see Notes 1 & 2)	PD	85	130	mA
Power Dissipation at 25 MHz -Rev C.65µ (see Notes 1 & 2)	PD	65	90	mA
Power Dissipation at 20 MHz-Rev C.8µ (see Notes 1 & 2)	PD	65	100	mA
Power Dissipation at 20 MHz-Rev C.65µ (see Notes 1 & 2)	PD	50	80	mA
Power Dissipation at 16.67 MHz-Rev C.8µ (see Notes 1 & 2)	PD	54	85	mA
Power Dissipation at 16.67 MHz-Rev C.65µ (see Notes 1 & 2)	PD	44	70	mA
Power Dissipation at 4 MHz-Rev C.8µ (see Notes 1,2 & 3)	PD	40	-	mA
Power Dissipation at 4 MHz-Rev C.65µ (see Notes 1, 2 & 3)	PD	27	-	mA
Power Dissipation at 3.3V 20 MHz-Rev C.65µ (see Note 2)	PD	30	60	mA
Power Dissipation at 3.3V 16.67 MHz-Rev C.65µ (see Note 2)	PD	25	50	mA

NOTES:
1. Values measured with maximum loading of 130 pF on all output pins. Typical means 5.0 V at 25°C. Maximum means guaranteed maximum over maximum temperature (85°C) and voltage (5.5 V).
2. The IMP is tested with the M68000 core executing, all three baud rate generators enabled and clocking at a rate of 64 kHz, and the two general-purpose timers running with a prescaler of 256. Power measurements are not significantly impacted by baud rate generators or timers until their clocking frequency becomes a much more sizable fraction of the system frequency than in these test conditions.
3. The M68000 core will not operate at 4 MHz. This is only for low power mode.

6.5 DC ELECTRICAL CHARACTERISTICS

Characteristic	Symbol	Min	Max	Unit
Input High Voltage (Except EXTAL)	V_{IH}	2.0	V_{DD}	V
Input Low Voltage (Except EXTAL)	V_{IL}	V_{SS} - 0.3	0.8	V
Input High Voltage (EXTAL)	V_{CIH}	4.0	VDD	V
Input Low Voltage (EXTAL)	V_{CIL}	V_{SS} - 0.3	0.6	V
Input Leakage Current	I_{IN}	—	20	µA
Input Capacitance All Pins	C_{IN}	—	15	pF
Three-State Leakage Current (2.4/0.5 V)	I_{TSI}	—	20	µA
Open Drain Leakage Current (2.4 V)	I_{OD}	—	20	µA
Output High Voltage (I_{OH} = 400 µA) (see Note)	V_{OH}	V_{DD}–1.0	—	V
Output Low Voltage	V_{OL}			V
(I_{OL} = 3.2 mA) A1–A23, PB0–PB11, FC0–FC2, $\overline{CS0}$–$\overline{CS3}$ $\overline{IAC}$, $\overline{AVEC}$, $\overline{BG}$, RCLK1, RCLK2, RCLK3, TCLK1, TCLK2, TCLK3, $\overline{RTS1}$, $\overline{RTS2}$, $\overline{RTS3}$, SDS2, PA12, RXD2, RXD3, CTS2, CD2, CD3 $\overline{DREQ}$, BRG1		—	0.5	
(I_{OL} = 5.3 mA) $\overline{AS}$, $\overline{UDS}$, $\overline{LDS}$, R/$\overline{W}$, $\overline{BERR}$ $\overline{BGACK}$, $\overline{BCLR}$, $\overline{DTACK}$, $\overline{DACK}$, $\overline{RMC}$, D0–D15, $\overline{RESET}$		—	0.5	
(I_{OL} = 7.0 mA) TXD1, TXD2, TXD3		—	0.5	
(I_{OL} = 8.9 mA) $\overline{DONE}$, $\overline{HALT}$, $\overline{BR}$ (as output)		—	0.5	
(I_{OL} = 3.2 mA) CLKO		—	0.4	
Output Drive CLKO	O_{CLK}	—	50	pF
Output Drive ISDN I/F (GCI Mode)	O_{GCI}	—	150	pF
Output Drive All Other Pins	O_{ALL}	—	130	pF
Output Drive Derating Factor for CLKO of 0.030 ns/pF	O_{KF}	20	50	pF
Output Drive Derating Factor for CLKO of 0.035 ns/pF	O_{KF}	50	130	pF
Output Drive Derating Factor for All Other Pins 0.035 ns/pF	O_{KF}	20	130	pF
Output Drive Derating Factor for All Other Pins 0.055 ns/pF	O_{KF}	130	220	pF
Power	V_{DD}	4.5	5.5	V
Common	V_{SS}	0	0	V

NOTE: The maximum I_{OH} for a given pin is one-half the I_{OL} rating for that pin. For an I_{OH} between 400 µA and I_{OL}/2 mA, the minimum V_{OH} is calculated as: V_{DD} - (1 +.05 V/mA(I_{OH} -.400 mA)).

NOTE: All AC specs are assume an output load of 130pf (except for CLKO).

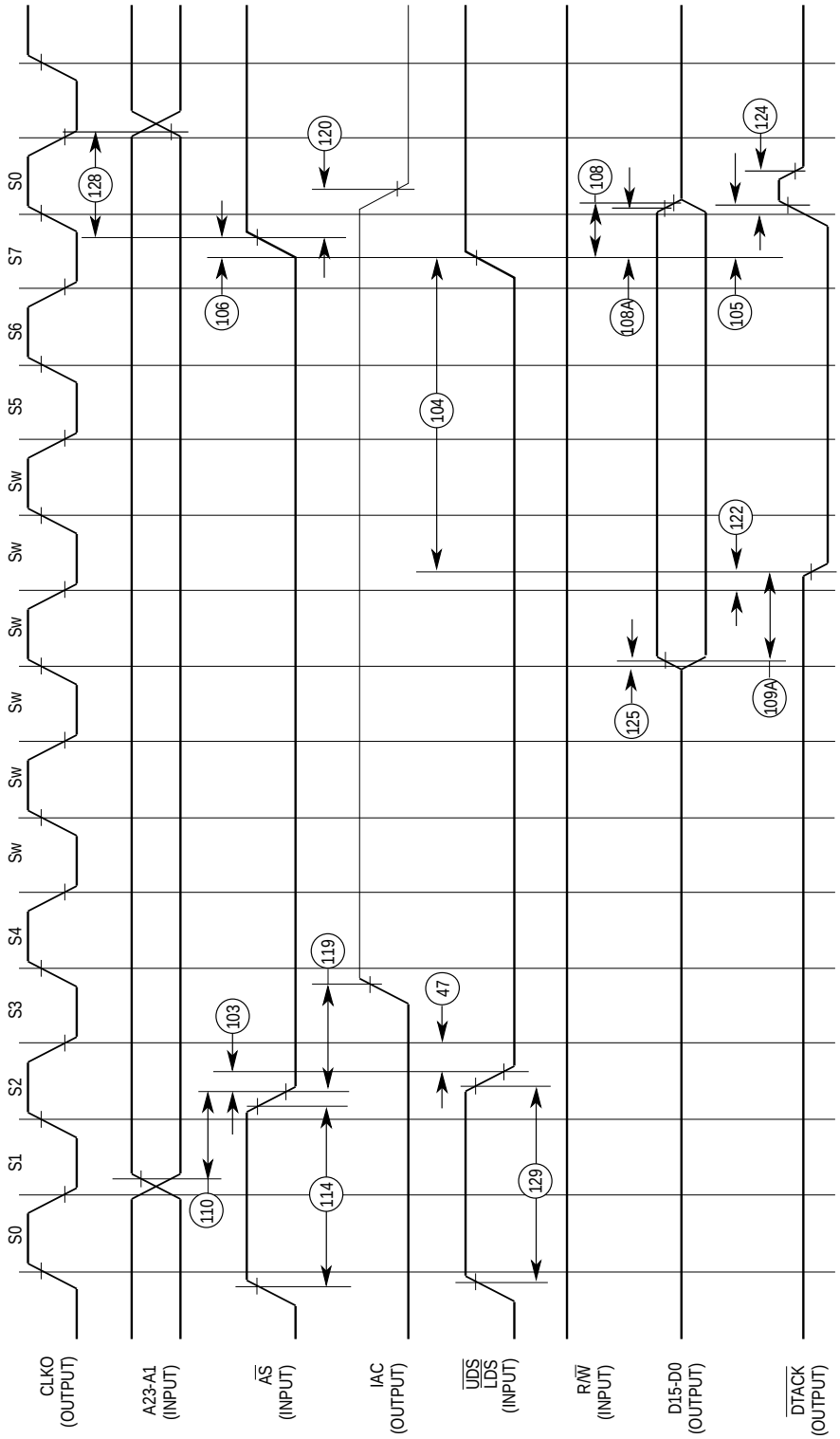
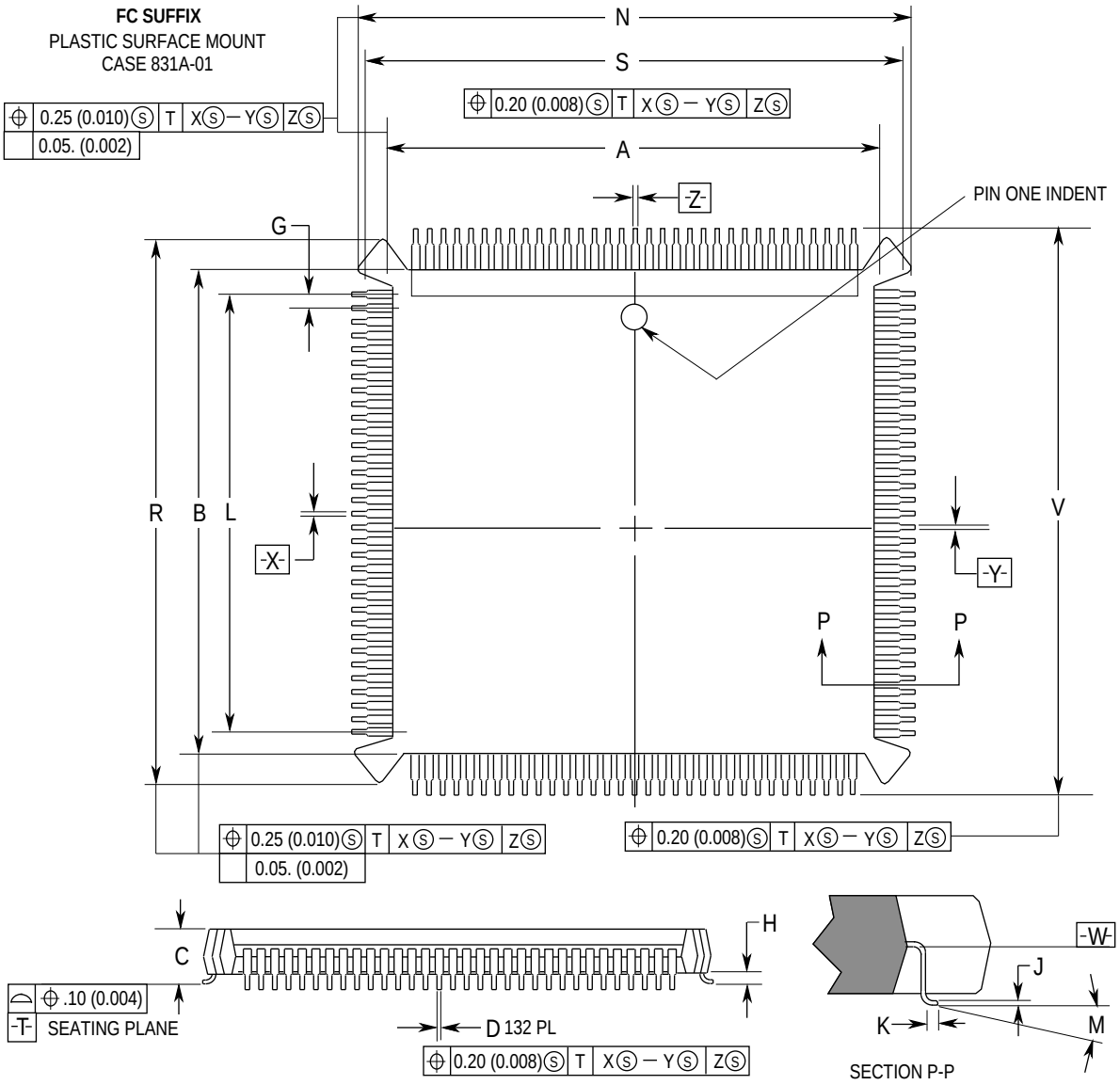


Figure 6-8. External Master Internal Asynchronous Read Cycle Timing Diagram

7.2.2 Plastic Surface Mount (PQFP)



DIM	MILLIMETERS		INCHES	
	MIN	MAX	MIN	MAX
A	24.06	24.20	0.947	0.953
B	24.06	24.20	0.947	0.953
C	4.07	4.57	0.160	0.180
D	0.21	0.30	0.008	0.012
G	0.64 BSC		0.025 BSC	
H	0.51	1.01	0.020	0.040
J	0.16	0.20	0.006	0.008
K	0.51	0.76	0.020	0.030
M	0°	8°	0°	8°
N	27.88	28.01	1.097	1.103
R	27.88	28.01	1.097	1.103
S	27.31	27.55	1.075	1.085
V	27.31	27.55	1.075	1.085

NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: INCHES
3. DIM A, B, N, AND R DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE MOLD PROTRUSION FOR DIMENSIONS A AND B IS 0.25 (0.010), FOR DIMENSIONS N AND R IS 0.18 (0.007).
4. DATUM PLANE -W- IS LOCATED AT THE UNDERSIDE OF LEADS WHERE LEADS EXIT PACKAGE BODY.
5. DATUMS X-Y AND Z TO BE DETERMINED WHERE CENTER LEADS EXIT PACKAGE BODY AT DATUM -W-.
6. DIM S AND V TO BE DETERMINED AT SEATING PLANE, DATUM -T-.
7. DIM A, B, N AND R TO BE DETERMINED AT DATUM PLANE -W-.

5. Write MRBLR.
6. Write CRC_Mask_L and CRC_Mask_H.
7. Write DISFC.
8. Write CRCEC.
9. Write ABTSC.
10. Write NMARC.
11. Write RETRC.
12. Write MFLR.
13. Write HMASK.
14. Write HADDR1, HADDR2, HADDR3, and HADDR4.

E.1.2.3 SCC INITIALIZATION.

15. Write SCON.
16. Write SCM without setting the ENR and ENT bits.
17. Write DSR.
18. Write SCCE with $FF to clear any previous events.
19. Write SCCM.
20. Write IMR.

E.1.2.4 SCC OPERATION.

21. Write the Rx buffer descriptor control/status, buffer pointer high, and buffer pointer low words for all of the buffer descriptors that are going to be used. Set the W bit in the last buffer descriptor to be used in the queue.
22. Prepare transmit buffers as required to transmit data on the SCC. Set the R bit in each Tx buffer descriptor's control/status word when the data buffer is ready for transmission. Set the W bit in the last Tx buffer descriptor in the table so that the IMP will use the first Tx buffer descriptor (after the user sets the R bit) for the next transmission.
23. Write SCM, setting the ENR and ENT bits to enable reception and transmission on the SCC.
24. Prepare more transmit buffers as required to transmit data on the SCC.

E.1.2.5 SCC INTERRUPT HANDLING.

1. Read the SCC event register.
2. Clear any unmasked bits that will be used in this interrupt routine.
3. Handle the interrupt events as required by the system.
4. Clear the appropriate SCC bit in the in-service register (ISR) of the interrupt controller.
5. Return from the interrupt.