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	Active
Core Processor	M68000
Number of Cores/Bus Width	1 Core, 8/16-Bit
Speed	25MHz
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	144-LQFP
Supplier Device Package	144-LQFP (20x20)
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=mc68302ag25c

UNITED STATES

ALABAMA, Huntsville	(205) 464-6800
ARIZONA, Tempe	(602) 897-5056
CALIFORNIA, Agoura Hills	(818) 706-1929
CALIFORNIA, Los Angeles	(310) 417-8848
CALIFORNIA, Irvine	(714) 753-7360
CALIFORNIA, Rosevllle	(916) 922-7152
CALIFORNIA, San Diego	(619) 541-2163
CALIFORNIA, Sunnyvale	(408) 749-0510
COLORADO, Colorado Springs	(719) 599-7497
COLORADO, Denver	(303) 337-3434
CONNECTICUT, Wallingford	(203) 949-4100
FLORIDA, Maitland	(407) 628-2636
FLORIDA, Pompano Beach/ Fort Lauderdal	(305) 486-9776
FLORIDA, Clearwater	(813) 538-7750
GEORGIA, Atlanta	(404) 729-7100
IDAHO, Boise	(208) 323-9413
ILLINOIS, Chicago/Hoffman Estates	(708) 490-9500
INDIANA, Fort Wayne	(219) 436-5818
INDIANA, Indianapolis	(317) 571-0400
INDIANA, Kokomo	(317) 457-6634
IOWA, Cedar Rapids	(319) 373-1328
KANSAS, Kansas City/Mission	(913) 451-8555
MARYLAND, Columbia	(410) 381-1570
MASSACHUSETTS, Marborough	(508) 481-8100
MASSACHUSETTS, Woburn	(617) 932-9700
MICHIGAN, Detroit	(313) 347-6800
MINNESOTA, Minnetonka	(612) 932-1500
MISSOURI, St. Louis	(314) 275-7380
NEW JERSEY, Fairfield	(201) 808-2400
NEW YORK, Fairport	(716) 425-4000
NEW YORK, Hauppauge	(516) 361-7000
NEW YORK, Poughkeepsie/Fishkill	(914) 473-8102
NORTH CAROLINA, Raleigh	(919) 870-4355
OHIO, Cleveland	(216) 349-3100
OHIO, Columbus Worthington	(614) 431-8492
OHIO, Dayton	(513) 495-6800
OKLAHOMA, Tulsa	(800) 544-9496
OREGON, Portland	(503) 641-3681
PENNSYLVANIA, Colmar	(215) 997-1020
Philadelphia/Horsham	(215) 957-4100
TENNESSEE, Knoxville	(615) 690-5593
TEXAS, Austin	(512) 873-2000
TEXAS, Houston	(800) 343-2692
TEXAS, Plano	(214) 516-5100
VIRGINIA, Richmond	(804) 285-2100
WASHINGTON, Bellevue	(206) 454-4160
Seattle Access	(206) 622-9960
WISCONSIN, Milwaukee/Brookfield	(414) 792-0122

CANADA

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

INTERNATIONAL

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

FULL LINE REPRESENTATIVES

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

HYBRID COMPONENTS RESELLERS

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

DNS—Done Not Synchronized

This bit is set if operand packing is performed between 16-bit memory and an 8-bit peripheral and the 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 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 DONE with no errors occurring during any IDMA transfer bus cycle. DONE will not be set if the channel terminates due to an error. DONE is cleared by writing a one or by a software RST in the CMR. Writing a zero has no effect on this bit.

3.1.3 Interface Signals

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

3.1.3.1 DREQ and DACK

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

3.1.3.2 DONE

This bidirectional signal is used to indicate the last IDMA transfer. With internal request modes, the IDMA activates 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, 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. DONE is an output if the transfer count is exhausted.

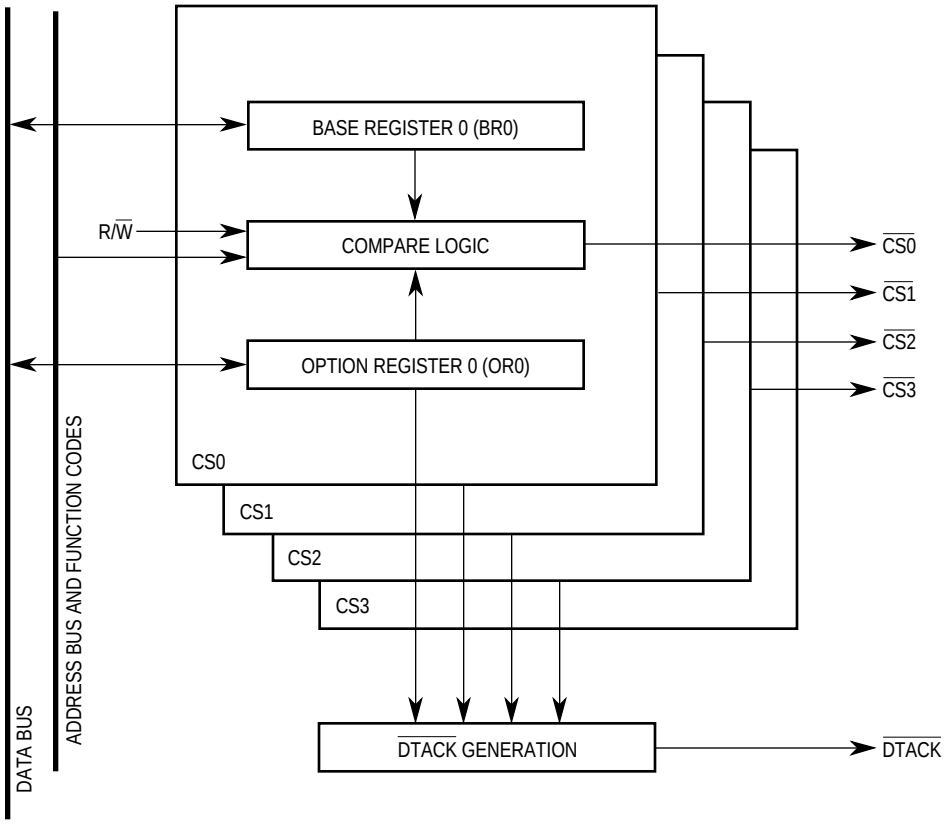


Figure 3-9. Chip-Select Block Diagram

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

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

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

NOTE

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

LPP16—Low-power Clock Prescale Divide by 16

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

After a system reset, this bit defaults to zero.

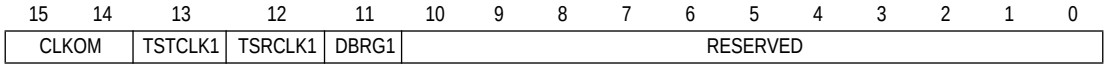
LPREC—Low-Power Recovery

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

After a system reset, the bit defaults to zero.

3.9 CLOCK CONTROL REGISTER

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



CLKOM—CLKO Mode

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

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

TSTCLK1—Three-state TCLK1

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

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

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

4.2 SDMA CHANNELS

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

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

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

4.5.6.2 Maximum Receive Buffer Length Register (MRBLR)

Each SCC has one MRBLR that is used to define the receive buffer length for that SCC. The MRBLR defines the maximum number of bytes that the IMP will write to a receive buffer on that SCC before moving to the next buffer. The IMP may write fewer bytes to the buffer than MRBLR if a condition such as an error or end of frame occurs, but it will never write more bytes than the MRBLR value. Thus, buffers supplied by the user for use by the IMP should always be of size MRBLR (or greater) in length.

The transmit buffers for an SCC are not affected in any way by the value programmed into MRBLR. Transmit buffers may be individually chosen to have varying lengths, as needed. The number of bytes to be transmitted is chosen by programming the data length field in the Tx BD.

NOTE

MRBLR was not intended to be changed dynamically while an SCC is operating. However, if it is modified in a single bus cycle with one 16-bit move (NOT two 8-bit back-to-back bus cycles), then a dynamic change in receive buffer length can be successfully achieved, which occurs when the CP moves control to the next Rx BD in the table. Thus, a change to MRBLR will not have an immediate effect. To guarantee the exact Rx BD on which the change will occur, the user should change MRBLR only while the SCC receiver is disabled (see 4.5.6 SCC Parameter RAM Memory Map).

NOTE

The MRBLR value should be greater than zero in all modes. In the HDLC and transparent modes, the MRBLR should have an even value.

4.5.6.3 Receiver Buffer Descriptor Number (RBD#)

The RBD# for each SCC channel defines the next BD to which the receiver will move data when it is in the IDLE state or defines the current BD during frame processing. The RBD# is the BD offset from the SCC base in the Rx BD table. For Rx BD 0, RBD# = $00; for Rx BD 1, RBD# = $08, etc. Upon reset, the CP main controller sets this register to zero. The user can change this register only after the ENR bit is clear and after the ENTER HUNT MODE command has been issued. In most applications, this parameter will never need to be modified by the user.

4.5.6.4 Transmit Buffer Descriptor Number (TBD#)

The TBD# for each SCC channel defines the next BD from which the transmitter will move data when it is in the IDLE state or defines the current BD during frame transmission. The TBD# is the BD offset from the SCC base in the Tx BD table. For Tx BD 0, TBD# = $40; for Tx BD 1, TBD# = $48, etc. Upon reset, the CP main controller sets this register to $40. The user can change this register only after the STOP TRANSMIT command has been issued. In most applications, this parameter will never need to be modified by the user.

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.

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.

Bits 11, 9–8—Reserved for future use.

V.110—V.110 Mode

0 = DDCMP mode; synchronous DDCMP is chosen.
1 = V.110 mode; the V.110 protocol description is in 4.5.15 V.110 Controller.

SYNF—Transmit SYN1–SYN2 or IDLE between Messages and Control the RTS Pin

0 = Send ones between messages. $\overline{\text{RTS}}$ is negated between messages.

NOTE

The DDCMP controller can transmit ones in both NRZ and NRZI data encoded formats. The minimum number of ones transmitted is 17.

1 = Send SYN1–SYN2 pairs between messages. $\overline{\text{RTS}}$ is always asserted. Note that SYN1 and SYN2 may be the same character.

ENC—Data Encoding Format

0 = Nonreturn to Zero (NRZ). A one is a high level; a zero is a low level.
1 = Nonreturn to Zero Inverted (NRZI). A one is represented by no change in the level; a zero is represented by a change in the level. The receiver decodes NRZI, but a clock must be supplied. The transmitter encodes NRZI.

COMMON SCC MODE BITS—See 4.5.3 SCC Mode Register (SCM) for a description of the DIAG1, DIAG0, ENR, ENT, MODE1, and MODE0 bits.

4.5.14.10 DDCMP Receive Buffer Descriptor (Rx BD)

The CP reports information about the received data for each buffer using the BDs. The Rx BD is shown in Figure 4-36. The CP closes the current buffer, generates a maskable interrupt, and starts to receive data in the next buffer after any of the following events:

- Receiving the received message length number of bytes (RMLG)
- Detecting an error
- Detecting a full receive buffer
- Issuing the ENTER HUNT MODE command

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET + 0	E	X	W	I	L	H	T2	T1	—	—	CF	FR	PR	CR	OV	CD
OFFSET + 2	DATA LENGTH															
OFFSET + 4	RX BUFFER POINTER (24-bits used, upper 8 bits must be 0)															
OFFSET +6																

Figure 4-36. DDCMP Receive Buffer Descriptor

The first word of the Rx BD contains control and status bits. Bits 15–12 are written by the user before the buffer is linked to the Rx BD table, and bits 5–0 and 11–8 are set by the IMP

I—Interrupt

0 = No interrupt is generated after this buffer has been serviced.
1 = Either TX or TXE in the DDCMP event register will be set when this buffer has been serviced by the DDCMP controller, which can cause interrupts.

L—Last

0 = This buffer is not the last in the message.
1 = The last bit is set by the processor to indicate that this buffer is the last buffer in the current message.

NOTE

The DDCMP controller checks the TC bit, not the last bit, to determine whether to append the CRC sequence. The DDCMP controller will transmit the programmable number of SYN1–SYN2 pairs before transmitting the next buffer (message) when the last bit is set.

TC—Tx CRC

0 = Do not transmit a CRC sequence after the buffer's last data byte.
1 = Transmit a CRC16 sequence after the buffer's last data byte.

When the last bit is not set but TC is set (e.g., in a header buffer), the DDCMP controller will append the next buffer immediately following the CRC sequence. The preset value for the CRC16 calculation is located in the PCRC register and should be initialized to all zeros or all ones.

OL—Optional Last

This bit allows the user to transmit abutted messages in DDCMP.

0 = Normal operation. The SYNF bit in the DDCMP mode register determines the pattern transmitted between messages.
1 = Abutted messages. The CP checks the ready bit of the next Tx BD after processing the current BD, and, if set, abuts the next message to the current message. If the ready bit is not set, the SYNF bit determines the transmitted pattern.

Bits 8–2—Reserved for future use.

The following status bits are written by the DDCMP controller after it has finished transmitting the associated data buffer.

UN—Underrun

The DDCMP controller encountered a transmitter underrun condition while transmitting the associated data buffer.

NOTE

This error can occur only on synchronous links.

CT—CTS Lost

$\overline{\text{CTS}}$ in NMSI mode or grant in IDL/GCI mode was lost during message transmission.

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.

4.5.16.2 Transparent Channel Buffer Reception Processing

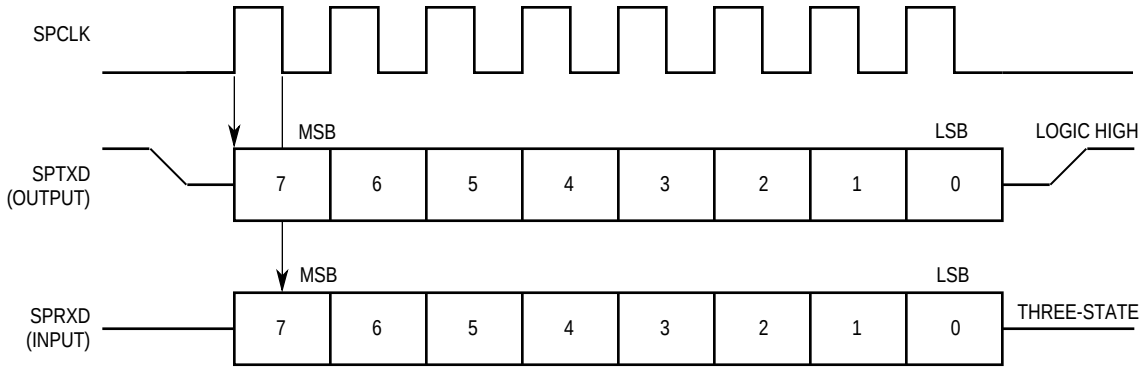
When the M68000 core enables the transparent receiver, it will enter hunt mode. In this mode, it waits to achieve synchronization before receiving data. See 4.5.16.5 Transparent Synchronization for details.

Once data reception begins, the transparent receiver begins moving data from the receive FIFO to the receive buffer, always moving a 16-bit word at a time. After each word is moved to memory, the RCH bit in the transparent event register is set, which can generate a maskable interrupt, if desired. The transparent receiver continues to move data to the receive buffer until the buffer is completely full, as defined by the byte count in MRBLR. The receive buffer length (stored in MRBLR) and starting address must always be even, so the minimum receive buffer length must be 2.

After a buffer is filled, the transparent receiver moves to the next Rx BD in the table and begins moving data to its associated buffer. If the next buffer is not available when needed, a busy condition is signified by the setting of the BSY bit in the transparent event register, which can generate a maskable interrupt.

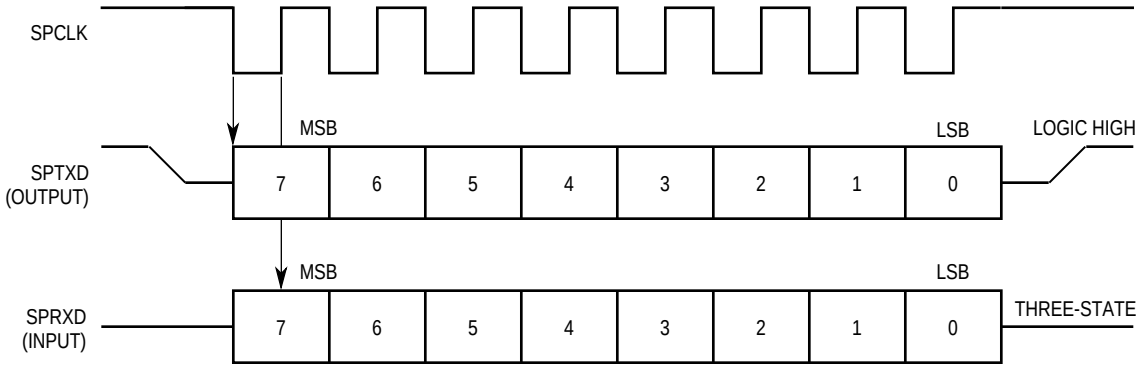
Received data is always packed into memory a word at a time, regardless of how it is received. For example, in NMSI mode, the first word of data will not be moved to the receive buffer until after the sixteenth receive clock occurs. In PCM highway mode, the same principle applies except that the clocks are only internally active during an SCC time slot. For example, if each SCC time slot is seven bits long, the first word of data will not be moved to the receive buffer until after the second bit of the third time slot, regardless of how much time exists between individual time slots.

Once synchronization is achieved for the receiver, the reception process continues unabated until a busy condition occurs, a $\overline{\text{CD}}$ lost condition occurs, or a receive overrun occurs. The busy condition error should be followed by an ENTER HUNT MODE command to the



NOTE: Transmitted data bits shift on rising edges; received bits are sampled on falling edges.

(a) CI=0



NOTE: Transmitted data bits shift on falling edges; received bits are sampled on rising edges.

(b) CI=1

Figure 4-44. SCP Timing

The SCP can be configured to operate in a local loopback mode, which is useful for local diagnostic functions.

Note that the least significant bit of the SCP is labeled as data bit 0 on the serial line; whereas, other devices, such as the MC145554 CODEC, may label the most significant bit as data bit 0. The MC68302 SCP bit 7 (most significant bit) is shifted out first.

The SCP key features are as follows:

- Three-Wire Interface (SPTXD, SPRXD, and SPCLK)
- Full-Duplex Operation
- Clock Rate up to 4.096 MHz
- Programmable Clock Generator
- Local Loopback Capability for Testing

tomatic vectoring only. AVEC instead of DTACK should be asserted during autovectoring and should be high otherwise.

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

5.10 MC68302 BUS INTERFACE SIGNAL SUMMARY

Table 5-2 and Table 5-3 summarize all bus signals discussed in the previous paragraphs. They show the direction of each pin for the following bus masters: M68000 core, IDMA, SDMA (includes DRAM refresh), and external. Each bus master can access either internal dual-port RAM and registers or an external device or memory. When an external bus master accesses the internal dual-port RAM or registers, the access may be synchronous or asynchronous.

When the M68000 core is disabled, BR and BG change their direction, and BCLR becomes bidirectional.

Table 5-2. Bus Signal Summary—Core and External Master

		M68000 Core Master Access To		External Master Access To	
Signal Name	**Pin Type**	**Internal Memory Space**	**External Memory Space**	**Internal Memory Space**	**External Memory Space**
A23–A1, FC2–FC0, AS, UDS, LDS, R/W, RMC	I/O	O	O	I	I
BCLR	I/O Open Drain	O	O	O	O
IAC	O	O	O	O	O
D15–D0 Read	I/O	O	I	O	I
D15–D0 Write	I/O	O	O	I	I
DTACK	I/O	O	**	O	**
BR	I/O Open Drain	I	I	I	I
BG	I/O	O	O	O	O
BGACK	I/O	I	I	I	I
HALT	I/O Open Drain	I/O	I/O	I	I
RESET	I/O Open Drain	I/O	I/O	I	I
BERR	I/O Open Drain	I/O***	I/O***	I/O***	I/O***
IPL2–IPL0	I	I	I	I	I
AVEC	I	I	I	I	I
IOUT2–IOUT0	O	O	O	O	O

**If DTACK is generated automatically (internally) by the chip-select logic, then it is an output. Otherwise, it is an input.
***BERR is an open-drain output, and may be asserted by the IMP when the hardware watchdog is used or when the chip-select logic detects address conflict or write protect violation. BERR may be asserted by external logic in all cases.

Table 5-3. Bus Signal Summary—IDMA and SDMA

		IDMA Master Access To		SDMA Master Access To	
Signal Name	**Pin Type**	**Internal Memory Space**	**External Memory Space**	**Internal Memory Space**	**External Memory Space**
A23–A1, FC–FC0, AS, UDS, LDS, R/W, RMC	I/O	O	O	N/A	O
BCLR	I/O Open Drain	I #	I #	N/A	O
IAC	O	O	O	N/A	O
D15—D0 Read	I/O	O	I	N/A	I
D15—D0 Write	I/O	O	O	N/A	O
DTACK	I/O	O	**	N/A	**
BR	I/O	O ##	O ##	N/A	O ##
BG	I/O	I ##	I ##	N/A	I ##
BGACK	I/O	O	O	N/A	O
HALT	I/O Open Drain	I	I	N/A	I
RESET	I/O Open Drain	I	I	N/A	I
BERR	I/O Open Drain	I/O***	I/O***	N/A	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.

5.11 PHYSICAL LAYER SERIAL INTERFACE PINS

The physical layer serial interface has 24 pins, and all but one of them have multiple functionality. The pins can be used in a variety of configurations in ISDN or non-ISDN environments. Table 5-3 shows the functionality of each group of pins and their internal connection to the three SCC and one SCP controllers. The physical layer serial interface can be configured for non-multiplexed operation (NMSI) or multiplexed operation that includes IDL, GCI, and PCM highway modes. IDL and GCI are ISDN interfaces. When working in one of the multiplexed modes, the NMSI1/ISDN physical interface can be connected to all three SCC controllers.

Table 5-4. Serial Interface Pin Functions

First Function	Connected To	Second Function	Connected To
NMSI1 (8)	SCC1 Controller	ISDN Interface	SCC1/SCC2/SCC3
NMSI2 (8)	SCC2 Controller	PIO—Port A	Parallel I/O
NMSI3 (5)	SCC3 Controller	PIO—Port A	Parallel I/O
NMSI3 (3)	SCC3 Controller	SCP	SCP Controller

NOTE: Each one of the parallel I/O pins can be configured individually.

- Supports external loopback between two channels or on the same channel
- Trace option for reporting to system management each primitive issued by the LAPD module to the layer 3 module or to layer 2 management

The LAPB module features are as follows:

- Implements *1988 CCITT Recommendation X25,* chapters 2.1 through 2.4
- Supports up to twelve distinct physical channels with each operating as an independent station
- Modulo 8 or modulo 128 operation
- Applicable for DTE and DCE applications
- Uses dedicated transmit pool for fast control frame generation
- Dynamic modification of protocol parameters
- Independent of layer 1 and layer 3 implementation
- Message-oriented interface
- Independent configuration of upper and lower layer modules interfacing with each LAPB link
- Special mode for internal loopback (frames are not sent to the driver)
- Supports external loopback between two MC68302 serial channels or on the same MC68302 serial channel
- Trace option for reporting to system management each primitive issued by the LAPB module to layer 3 or to layer 2 management

The X.25 module features are as follows:

- Fully implements 1988 CCITT Recommendation X.25, chapters 3.1 through 7.3
- May be used with both layer 2 modules: LAPD or LAPB
- Unlimited number of layer 2 entities (interfaces)
- Supports up to 4095 logical channels for each interface
- Many DTE/DCE interface parameters (configurable for each interface):

  —- DTE/DCE
  —- Modulo 8/128
  —- Window size
  —- Maximum receive and transmit packet lengths

- Layer 4 message fragmentation/assembly using M-BIT
- Q-BIT support
- All standard CCITT X.25 facilities
- Compatible with X.213 interface primitives
- Link parameters (configurable separately for each interface):

  —- Interface ID

designed to comply with the requirements of the Hayes AT command set; however, it can be used with simpler character schemes as well (such as a carriage return).

The SCC receiver synchronizes on the falling edge of the START bit. Once a start bit is detected, each bit received is processed by the AutoBaud controller. The AutoBaud controller measures the length of the START bit to determine the receive baudrate and compares the length to values in a user supplied lookup table. After the baudrate is determined, the AutoBaud controller assembles the character and compares it against two user-defined characters. If a match is detected, the AutoBaud controller interrupts the host and returns the determined nominal start value from the lookup table. The AutoBaud controller continues to assemble the characters and interrupt the host until the host stops the reception process. The incoming message should contain a mixture of even and odd characters so that the user has enough information to decide on the proper character format (length and parity). The host then uses the returned nominal start value from the lookup table, modifies the SCC Configuration Register (SCON) to generate the correct baudrate, and reprograms the SCC to UART mode.

Many rates are supported including: 150, 300, 600, 1200, 2400, 4800, 9600, 14.4K, 19.2K, 38.4K, 57.6K, 64K, 96K, and 115.2K. To estimate the performance of the AutoBaud microcode package, the performance table in Appendix A of the MC68302 user's manual can be used. The maximum full-duplex rate for a BISYNC channel is one-tenth of the system clock rate. So a 16.67 MHz 68302 can support 115.2k autobaudrate with another low-speed channel (<50 kbps) and a 20 MHz MC68302 can support 115.2k AutoBaudrate with 2 low-speed channels. The performance can vary depending on system loading, configuration, and echoing mode.

C.6 MICROCODE FROM RAM INITIALIZATION SEQUENCE

1. Perform a total system reset of the MC68302.
2. Write $0700 to the BAR. The base address of the internal dual-port RAM after this action is $700000 (hex). If a different base address is desired, the S-record file addresses should be modified to the desired address.
3. Load the S-record file data into the internal dual-port RAM. (In a production environment, the microcode may be copied from EPROM directly to the internal dual-port RAM.)
4. Write $0001 to address $0F8 in supervisory space.
5. Write a software reset command to the CR.
6. Continue with the normal initialization sequence.

Example 2. If you only need one transparent channel (and you had all three physical interface available), your six choices are as follows:

1. NMSI1 using SCC1
2. PCM (i.e., NMSI1 is converted into PCM pins) using SCC1
3. GCI (i.e., NMSI1 is converted into GCI pins) using SCC1
4. IDL (i.e., NMSI1 is converted into IDL pins) using SCC1
5. NMSI2 using SCC2
6. NMSI3 using SCC3

Example 3. If you need to interface one, two, or three transparent channels to a single time-multiplexed bus, then the choice is simply PCM highway using SCC1 and (either SCC2 or SCC3 or both).

Example 4. If you need to interface one, two, or three transparent channels to an ISDN basic rate bus, then the choices are as follows:

1. IDL using SCC1 and (either SCC2 or SCC3 or both)
2. GCI using SCC1 and (either SCC2 or SCC3 or both)

NOTE

The preceding four examples of physical interface combinations apply equally well to other MC68302-supported protocols such as HDLC.

Since the purposes of GCI and IDL are clear, the real challenge is choosing between NMSI and PCM. What are the advantages of PCM over NMSI? To really answer that, you will have to take a more detailed look at the timings discussed in the following paragraphs. However, one general statement can be made: PCM mode allows better control over how data is gated into and out of the SCC, but requires that data is transmitted and received simultaneously on the SCC. (There are no separate TCLK and RCLK pins in PCM mode. Instead, there is one clock pin (L1 CLK) that clocks transmit and receive data whenever the syncs are activated).

The choice of physical interface is made in the serial interface mode register (SIMODE); $0000 is the default value and sets all three SCCs to use the NMSI interface. As another example, the value $0009 chooses SCC1 and SCC2 to use the PCM mode and SCC3 to use the NMSI interface.

D.8.4 General Transparent Mode Behavior

Transparent mode is entered by selecting the BISYNC mode and setting the NTSYN bit in the SCC mode register (SCM). In most applications, it is also customary to set the EXSYN bit in the SCM as well. No other SCM bits are valid in transparent mode except REVD, which allows the bit ordering for each byte of the transmitted and received data to be reversed before sending it out or storing it in memory.

All transfers to and from memory in transparent mode are 16 bits to maximize performance. All bits in the transmit buffer are transmitted out of the SCC in transparent mode, regardless

stream with the L bit in all Tx BDs cleared, then the byte alignment timing will remain constant.

D.8.11 Initializing Transparent Mode

Full examples of the assembler code required to initialize the HDLC and UART protocols are given in D.3 MC68302 Buffer Processing and Interrupt Handling and D.4 Configuring A Uart on the MC68302. A transparent mode initialization follows the same flow as these subsections except that different values would be used. The HDLC and UART examples also show writing of the BAR and full configuration of the interrupt controller to allow SCC interrupts, etc., which are not duplicated here.

The following example shows a step-by-step list of the SCC-related registers as they would be initialized to create a transparent SCC2 channel in the NMSI mode. The registers in this example are configured for external loopback with TCLK2 externally connected to RCLK2, TXD2 externally connected to RXD2, $\overline{\text{CTS2}}$ a don't care, and $\overline{\text{RTS2}}$ externally connected to $\overline{\text{CD2}}$ (sync). The functionality of this configuration is the same as that shown in Figure D-28. This example may be easily checked on the ADS302 board, either with the menu interface software already on the ADS302 board or with user-written software downloaded to the ADS302 board. The external connections can be made by placing three jumper cables on row B of the serial bus connector P8: B5-to-B6, B7-to-B8, and B10-to-B11.

1. To use SCC2 in the NMSI mode, we need to chose NMSI2 pins instead of parallel I/O pins. To do this, we write a one to the PACNT register in every bit position that we want an SCC pin to be active. For this example, we will assume all seven NMSI2 pins are active.
   PACNT = $xx7F
2. The SIMODE register is set to its default setting. This configuration chooses NMSI mode on all three SCCs. Actually, all we need is that NMSI mode be selected for SCC2.
   SIMODE = $0000
3. The SCON register configures the clocking options. Here we chose to generate about a 65-kHz clock on the TCLK2 pin with the internal baud rate generator. RCLK2 will take its input externally; thus, we connect the TCLK2 pin externally to RCLK2. SCON2 = $1200
4. The setting shown for SCM2 sets the EXSYN and NTSYN bits, sets the DIAG1-DIAG0 bits for software operation, and sets the protocol to BISYNC (which is actually transparent since the NTSYN bit is set).
   Since we are implementing an external loopback with the MC68302, the DIAG1-DIAG0 bits are *not set* for loopback mode. Setting the DIAG1-DIAG0 bits for loopback mode causes *internal* loopback. (To implement an internal loopback, externally connect only $\overline{\text{RTS2}}$ to $\overline{\text{CD2}}$ (sync), set SCON2 to $0200, and SCM2 to $6013. Later, the very last step is to set SCM2 to $601 F. With internal loopback, RCLK and TCLK should be directly supplied with the same clock source—either both from the internal baud rate generators or both from the same externally generated clock source.)
   SCM2 = $6033
5. The DSR2 does not need to be written and can be left at its default value since we are

E.1.1.2 PER SCC REGISTERS. Each of the three SCCs has a set of the following six registers. These registers configure the SCC and the protocol operation. Some parameters and register bits are protocol independent. The HDLC functions have been given for those parameters and bits that are protocol specific.

E.1.1.2.1 Serial Configuration Register (SCON). This 16-bit register is located at offset $882 (SCC1), $892 (SCC2), and $8A2 (SCC3). The SCON register is used to select the clock source and baud rate for the SCC.

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

WOMS—Wired-OR Mode Select

0 = TXD driver operates normally.
1 = TXD driver functions as an open-drain output and may be wired together with other TXD pins.

EXTC—External Clock Source

0 = The internal main clock is the source for the baud rate generator.
1 = The external clock on the TIN1 pin is the source for the baud rate generator.

TCS—Transmit Clock Source

0 = Transmit clock source is the baud rate generator output.
1 = Transmit clock source is the clock signal on TCLK pin.

RCS—Receive Clock Source

0 = Receive clock source is the baud rate generator output.
1 = Receive clock source is the clock signal on RCLK pin.

CD10—CD0—Clock Divider

Used to preset the 11-bit counter that is decremented at the prescaler output rate.

DIV4—SCC Clock Prescaler Divide by 4

0 = Divide-by-1 prescaler.
1 = Divide-by-4 prescaler.

E.1.1.2.2 SCC Mode Register (SCM). This 16-bit register is located at offset $884 (SCC1), $894 (SCC2), and $8A4 (SCC3). The SCM register configures the operation of the SCC and defines HDLC specific parameters. Note that reserved bits in registers should be written as zeros.

15	14	13	12	11	10	9	8
NOF3	NOF2	NOF1	NOF0	C32	FSE	—	RTE

7	6	5	4	3	2	1	0
FLG	ENC	DIAG1	DIAG0	ENR	ENT	MODE1	M0DE0

1. Write the port A and port B control registers (PACNT and PBCNT) to configure SCC2 or SCC3 serial interface pins as peripheral pins, if SCC2 or SCC3 is used.
2. Write SIMODE to configure the SCCs physical interface.
3. Write SIMASK if IDL or GCI multiplexed mode was selected in SIMODE.

E.3.2.2 GENERAL AND TRANSPARENT PROTOCOL-SPECIFIC RAM INITIALIZATION.

4. Write RFCR/TFCR.
5. Write MRBLR.

E.3.2.3 SCC INITIALIZATION.

6. Write SCON.
7. Write SCM without setting the ENR and ENT bits.
8. Write DSR.
9. Write SCCE with $FF to clear any previous events.
10. Write SCCM.
11. Write IMR.

E.3.2.4 SCC OPERATION.

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

E.3.2.5 SCC INTERRUPT HANDLING.

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