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	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	132-BQFP Bumpered
Supplier Device Package	132-PQFP (46x46)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc68302eh25cr2

Appendix F
Design Checklist

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

SECTION 3
SYSTEM INTEGRATION BLOCK (SIB)

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

The SIB includes the following functions:

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

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

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

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

This bit is valid only in the dedicated mode.

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

NOTE

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

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

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

ET6—IRQ6 Edge-/Level-Triggered

This bit is valid only in the dedicated mode.

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

NOTE

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

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

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

This bit is valid only in the dedicated mode.

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

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

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

3.5.3 Timer 3 - Software Watchdog Timer

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

3.5.3.1 Software Watchdog Timer Operation

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

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

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

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

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

3.5.3.2 Software Watchdog Reference Register (WRR)

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

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

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

HWDCN–HWDCN0—Hardware Watchdog Count 2–0

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

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

NOTE

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

3.8.7 Reducing Power Consumption

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

3.8.7.1 Power-Saving Tips

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

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

3.8.7.2 Low-Power (Standby) Modes

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

4.5.11.6 UART Address Recognition

In multidrop systems, more than two stations may be present on a network, with each having a specific address. Figure 4-18 shows two examples of such a configuration. Frames comprised of many characters may be broadcast, with the first character acting as a destination address. To achieve this, the UART frame is extended by one bit, called the address bit, to distinguish between an address character and the normal data characters. The UART can be configured to operate in a multidrop environment in which two modes are supported:

Automatic Multidrop Mode—The IMP automatically checks the incoming address character and accepts the data following it only if the address matches one of two 8-bit preset values. In this mode, UM1–UM0 = 11 in the UART mode register.

Nonautomatic Multidrop Mode—The IMP receives all characters. An address character is always written to a new buffer (it may be followed by data characters in the same buffer). In this mode, UM1–UM0 = 01 in the UART mode register.

Each UART controller has two 8-bit address registers (UADDR1 and UADDR2) for address recognition. In the automatic mode, the incoming address is checked against the lower order byte of the UART address registers. Upon an address match, the address match (M) bit in the BD is set/cleared to indicate which address character was matched. The data following it is written to the same data buffer.

NOTE

For 7-bit characters, the eighth bit (bit 7) in UADDR1 and UADDR2 should be zero.

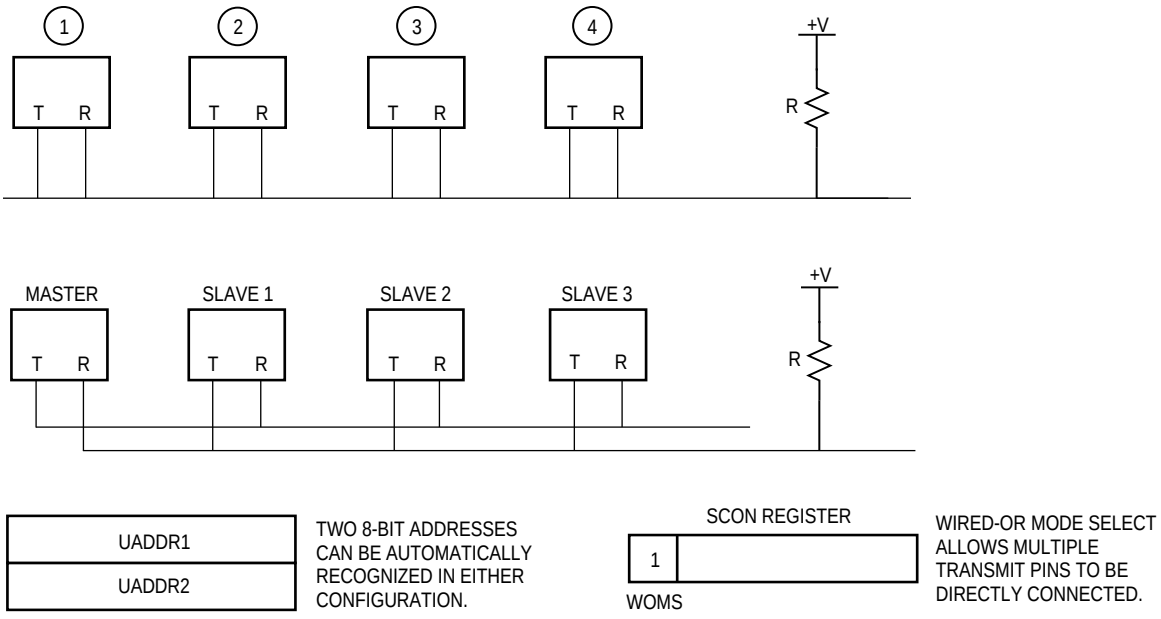


Figure 4-18. Two Configurations of UART Multidrop Operation

ceives the address character and writes it to a new buffer. No address recognition is performed.

10 = The DDCMP protocol is implemented over the asynchronous channel.

11 = Multidrop mode is enabled as in the 01 case, and the IMP automatically checks the address of the incoming address character and either accepts or discards the data following the address.

FRZ—Freeze Transmission

This bit allows the user to halt the UART transmitter and to continue transmission from the next character in the buffer at a later time.

0 = Normal operation (or resume transmission after FRZ is set).

1 = The UART completes transmission of any data already transferred to the UART FIFO (up to three characters) and then stops transmitting data. The UART continues to receive normally.

CL—Character Length

0 = 7-bit character length. On receive, bit 7 in memory is written as zero. On transmit, bit 7 in memory is a don't care.

1 = 8-bit character length

RTSM—RTS Mode

0 = $\overline{\text{RTS}}$ is asserted whenever the transmitter is enabled and there are characters to transmit. $\overline{\text{RTS}}$ is negated after the last stop bit of a transmitted character when both the shift register and the transmit FIFO are empty. RTS is also negated at the end of a buffer to guarantee accurate reporting of the CTS bit in the BD.

1 = $\overline{\text{RTS}}$ is asserted whenever the transmitter is enabled (the ENT bit is set).

SL—Stop Length

This bit selects the number of the stop bits transmitted by the UART. The receiver is always enabled for one stop bit. Fractional stop bits are configured in the DSR (see 4.5.11.12 Fractional Stop Bits).

0 = One stop bit

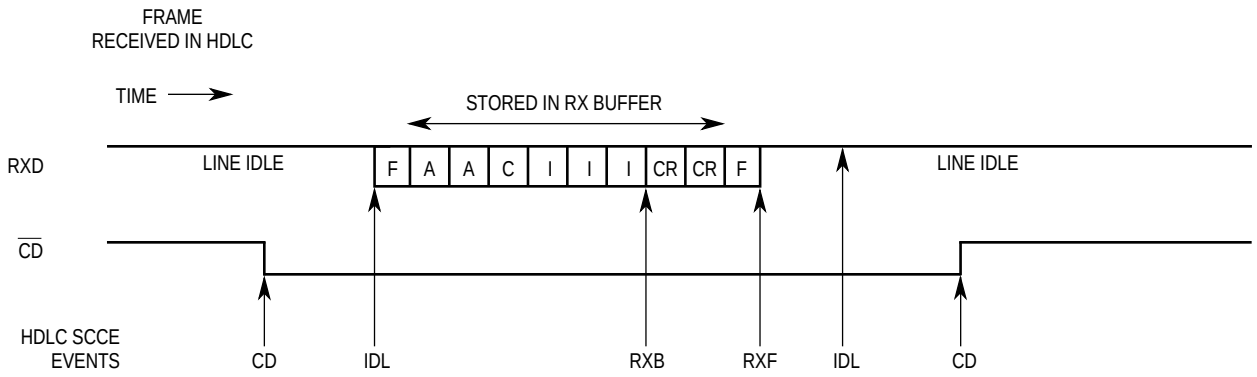
1 = Two stop bits

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

4.5.11.14 UART Receive Buffer Descriptor (Rx BD)

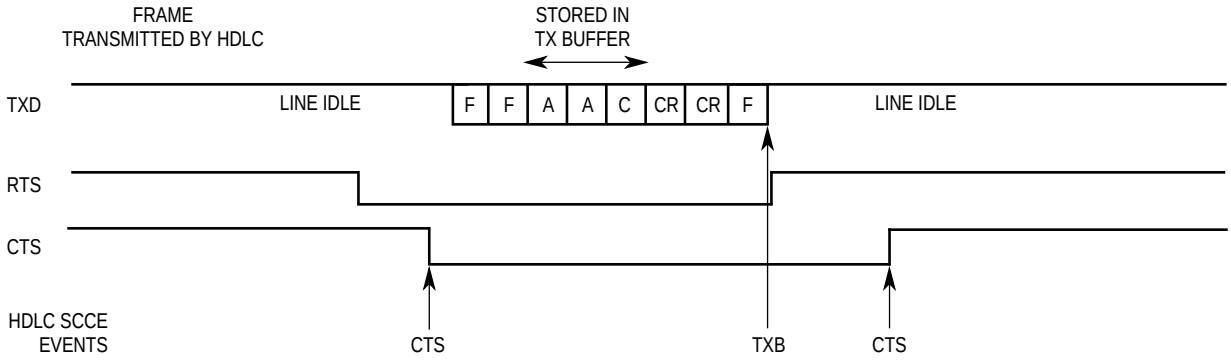
The CP reports information about each buffer of received data by its BDs. The Rx BD is shown in Figure 4-20. The CP closes the current buffer, generates a maskable interrupt, and starts to receive data in the next buffer due to any of the following events:

1. Reception of a user-defined control character (when reject (R) bit = 0)
2. Detection of an error during message processing
3. Detection of a full receive buffer
4. Reception of a programmable number of consecutive IDLE characters



NOTES:
1. RXB event assumes receive buffers are 6 bytes each.
2. The second IDL event occurs after 15 ones are received in a row.

LEGEND:
F = Flag A = Address byte C = Control byte I = Information byte CR = CRC byte



NOTE: TXB event shown assumes all three bytes were put into a single buffer.
Example shows one additional opening flag. This is programmable.

Figure 4-29. HDLC Interrupt Events Example

7	6	5	4	3	2	1	0
CTS	CD	IDL	TXE	RXF	BSY	TXB	RXB

CTS—Clear-To-Send Status Changed

A change in the status of the $\overline{\text{CTS}}$ 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 $\overline{\text{CD}}$ line was detected on the HDLC channel. The SCC status register may be read to determine the current status.

4.5.13.14 Programming the BISYNC Controllers

There are two general techniques that the software may employ to handle data received by the BISYNC controllers. The simplest way is to allocate single-byte receive buffers, request (in the status word in each BD) an interrupt on reception of each buffer (i.e., byte), and implement the BISYNC protocol entirely in software on a byte-by-byte basis. This simple approach is flexible and may be adapted to any BISYNC implementation. The obvious penalty is the overhead caused by interrupts on each received character.

A more efficient method is as follows. Multibyte buffers are prepared and linked to the receive buffer table. Software is used to analyze the first (two to three) bytes of the buffer to determine what type of block is being received. When this has been determined, reception can continue without further intervention to the user's software until a control character is encountered. The control character signifies the end of the block, causing the software to revert back to a byte-by-byte reception mode.

To accomplish this, the RCH bit in the BISYNC mask register should initially be set, enabling an interrupt on every byte of data received. This allows the software to analyze the type of block being received on a byte-by-byte basis. After analyzing the initial characters of a block, the user should either set the receiver transparent mode (RTR) bit in the BISYNC mode register or issue the RESET BCS CALCULATION command. For example, if DLE-STX is received, transparent mode should be entered. By setting the appropriate bit in the BISYNC mode register, the BISYNC controller automatically strips the leading DLE from <DLE-character> sequences. Thus, control characters are only recognized when they follow a DLE character. The RTR bit should be cleared after a DLE-ETX is received.

Alternatively, after receiving an SOH, the RESET BCS CALCULATION command should be issued. This command causes the SOH to be excluded from BCS accumulation and the BCS to be reset. Note that the RBCS bit in the BISYNC mode register (used to exclude a character from the BCS calculation) is not needed here since SYNCs and leading DLEs (in transparent mode) are automatically excluded by the BISYNC controller.

After recognizing the type of block above, the RCH interrupt should be masked. Data reception then continues without further interruption of the M68000 core until the end of the current block is reached. This is defined by the reception of a control character matching that programmed in the receive control characters table.

The control characters table should be set to recognize the end of the block as follows:

Control Characters	E	B	H
ETX	0	1	1
ITB	0	1	0
ETB	0	1	1
ENQ	0	0	0
Next Entry	1	X	X

After the end of text (ETX), a BCS is expected; then the buffer should be closed. Hunt mode should be entered when line turnaround occurs. ENQ characters are used to abort transmis-

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.

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

$\overline{\text{DREQ}}$/PA13—DMA Request

This input is asserted by a peripheral device to request an operand transfer between that peripheral device and memory. In the cycle steal request generation mode, this input is edge-sensitive. In burst mode, it is level-sensitive.

$\overline{\text{DACK}}$/PA14—DMA Acknowledge

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

$\overline{\text{DONE}}$/PA15—DONE

This bidirectional, open-drain signal is asserted by the IDMA or by a peripheral device during any IDMA bus cycle to indicate that the data being transferred is the last item in a block. The IDMA asserts this signal as an output during a bus cycle when the byte count register is decremented to zero. Otherwise, this pin is an input to the IDMA to terminate IDMA operation.

5.17 IACK OR PIO PORT B PINS

The IACK or PIO port B pins are shown in Figure 5-14.

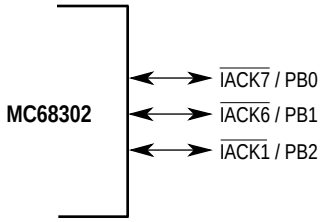


Figure 5-14. IACK or PIO Port B Pins

Each one of these three pins can be used either as an interrupt acknowledge signal or as a general-purpose parallel I/O port. Note that the IMP interrupt controller does not require the use of the IACK pins when it supplies the interrupt vector for the external source. The input buffers have Schmitt triggers.

$\overline{\text{IACK7}}$/PB0

$\overline{\text{IACK6}}$/PB1

$\overline{\text{IACK1}}$/PB2—Interrupt Acknowledge/Port B I/O

As $\overline{\text{IACK1}}$, $\overline{\text{IACK6}}$, and $\overline{\text{IACK7}}$, these active low output signals indicate to the external device that the MC68302 is executing an interrupt acknowledge cycle. The external device must then place its vector number on the lower byte of the data bus or use AVEC for autovectoring (unless internal vector generation is used).

5.18 TIMER PINS

The timer pins are shown in Figure 5-15.

6.3 POWER CONSIDERATIONS

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

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

where:

T_A = Ambient Temperature, °C

θ_{JA} = Package Thermal Resistance, Junction to Ambient, °C/W

P_D = $P_{INT} + P_{I/O}$

P_{INT} = I_{DD} x V_{DD}, Watts—Chip Internal Power

$P_{I/O}$ = Power Dissipation on Input and Output Pins—User Determined

For most applications $P_{I/O} < 0.3 \bullet P_{INT}$ and can be neglected.

If $P_{I/O}$ is neglected, an approximate relationship between P_D and T_J is

$$P_D = K \div (T_J + 273°C) \quad (2)$$

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

$$K = P_D \bullet (T_A + 273°C) + \theta_{JA} \bullet P_D^2 \quad (3)$$

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

APPENDIX B
DEVELOPMENT TOOLS AND SUPPORT

B.1 MOTOROLA SOFTWARE OVERVIEW

A software development package is offered as a set of independent modules which provides the following features:

- IMP Chip Evaluation
  A development board, described in B.5 302 Family ADS System, can be used as a prototyping vehicle for user code.
- IMP Chip Drivers
  Written in C, the source code of the IMP drivers is available on electronic media. These drivers were developed to support the needs of the protocol implementations described below.
- Protocol Implementations
  Modules implementing common ISO/OSI layer 2 and 3 protocols are available under license. These are in the form of binary relocatable object code as well as source code written in C.
- Portability
  All of the modules may be ported to different MC68302 implementations and combined with user-developed code. Well-defined software interface documentation is available for all provided modules.

B.2 MOTOROLA SOFTWARE MODULES

Chip driver routines written in C illustrate initialization of the IMP, interrupt handling, and the management of data transmission and reception on all channels.

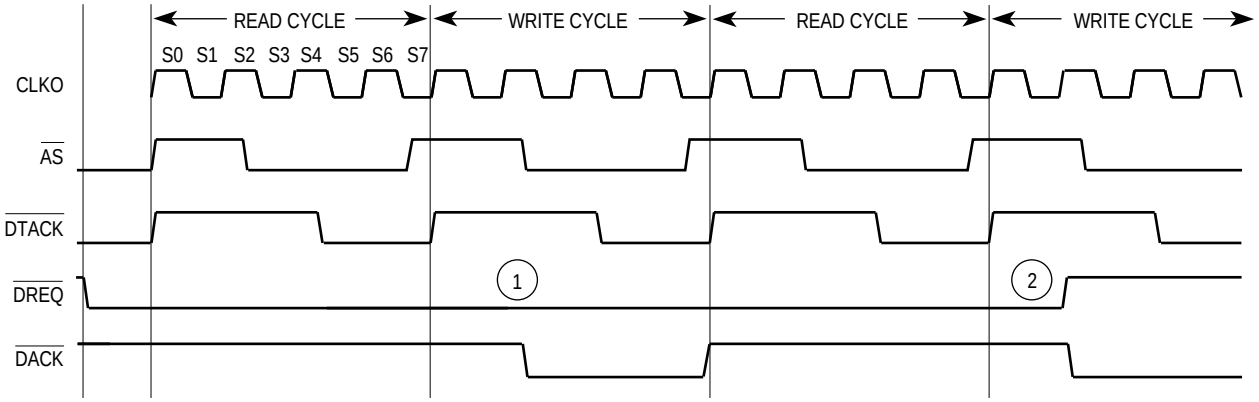
In addition to the chip drivers, protocol modules are provided. Layer 2 modules include LAPB and LAPD. (Support for layer 2 BISYNC and DDCMP functions are also provided with the chip driver package.) The layer 3 module supported is the X.25 packet layer protocol.

Since the modules do require some minimal operating system services, they are written to use our EDX operating system. Use of EDX with the protocol modules is not required as long as some other operating system support is provided by the user.

All modules communicate with each other using message passing. This technique simplifies the interfaces between the different modules and enables them to interface with other user-added modules with relative ease. Detailed descriptions of the software interfaces are available.

```
          BSET.B    #$7,(A2)          ;Set Ready bit of Tx BD
* The Tx BD sand data status is not checked since the only one is CTS lost,
* which is not applicable, since CTS is ignored in this application.
* The following updates A2 to point to the next Tx BD
          BTST.B    #$05,(A2)         ;test Wrap bit
          BNE.B     REINIT2           ;If set, reinit A2 to 700640
          ADDA.W    #$08,A2           ;else inc A2 by 8 to next Tx BD
          BRA.B     CONT              ;Jump to Continue on
REINIT2 MOVEA.L     #$700640,A2       ;Reinitialize A2
* Determine what the next byte to "echo" will be and then go to OUTERLOOP
 CONT     ADDO.W    #$1,A1            ;Increment A1 to next byte to send
          CMPA.L    #$30004,A1        ;Is A1 = 30004? ***
          BEQ.B     NEWA1             ;If so, go to NEWA1
          BRA.B     OUTLOOP           ;Jump back to outerloop and wait
NEWA1     SUBO.W    #$02.A1           ;Set A1 back to 30002 ?***
          BRA.B     OUTLOOP           ;Jump back to outerloop and wait
* The two lines with ***  above are dependent on the number of Rx BDs used.
* If the number is increased, these values should be increased by the same
* amount. These are the only lines dependent on the Rx BD or Tx BD setup.
*************************************************************************
*SCC3 Interrupt Routine
          ORG       $30500
          CLR.L     D1                ;clear D1
          MOVE.B    SCCE3,D1          ;Move SCCE3 status to D1
          MOVE.B    #$15,SCCE3        ;Clear only BRK. BSY and RX in SCCE3.

          BTST.B    #$2.D1            ;Is BSY set?
          BNE.B     BUSY              ;Jump to BUSY handler if set



* Test Break:
BRKTEST   BTST.B    #$4,D1            ;Is BRK set?
          BNE.B     BREAK             ;Jump to BREAK handler if set
*Test Receive:
RECTEST   BTST.B    #$0,D1            ;Is RX set?
          BNE.B     RECEIVE           ;Jump to RECEIVE handler if set
          JMP       ALMDONE           ;Jump to About Done (impossible)
* Busy handler:
BUSY      ADDQ.B    #1,D5             ;Inc Busy counter (no receive buffers)
          BSET.B    #$F,(A0)          ;set Empty bit of current Rx BD
          JMP       BRKTEST           ;Jump to test for BREAK

*Break handler:
BREAK     NOP                         ;This code ignores received breaks
* The UART BRKEC will record the number of breaks received
          JMP       RECTEST           ;Jump to test for RECEIVE
*Receive handler:
RECEIVE   ADDQ.W    #1,D3             ;Increment number of chars received
          ADDQ.B    #1,D6             ;D6 inc by 1 (character ready to send)

          ADDQ.W    #1,A0             ;Inc A0 to point to Rx BD byte status
          CMPI.B    #$0,(A0)          ;Does status = 00?
          BNE.B     BSTAT             ;Jump to Bad Status it not 00
INCPTR    SUBQ.W    #1,A0             ;Dec A0 to point to beginning of Rx BD
          ANDI.W    #$FF00,(A0)       ;Clear out Rx BD status
```

Figure D-10 illustrates the activation of external burst mode by using the DREQ signal as a level-sensitive input to the IDMA. If DREQ is asserted when the IDMA is accessing the peripheral (indicated by DACK being asserted) as shown in Figure D-10, then the IDMA will continue servicing the peripheral by performing another sequence of operand transfer cycles. The external burst mode stops when the DREQ signal is deasserted prior to the trailing edge of S3 in the cycle where DACK is asserted.



NOTE: DREQ is sampled on the falling edge of clock.

LEGEND:

(1) DREQ asserted prior to DTACK = continue burst mode transfer

(2) DREQ negated prior to DTACK = relinquish the bus

Figure D-10. Burst Mode Cycles

D.5.7 Internal Interrupt Sequence

An interrupt acknowledge cycle (IACK) occurs when an allowed internal or external interrupt request is pending and the priority of the interrupt is higher than the current microprocessor run level. The interrupt acknowledge cycle begins at the conclusion of instruction execution in state S0. All internal resources, including the IDMA, generate INRQ requests at level 4.

The four registers used in interrupt processing are as follows:

1. The interrupt mask register (IMR) contains the flags that, when set, allow the INRQ source to initiate service.
2. The interrupt pending register (IPR) contains bits that correspond to the INRQ source requesting service.
3. The interrupt in-service register (ISR) indicates which internal interrupts are currently being processed (usually only one at a time).
4. The global interrupt mode register (GIMR) has bits that specify interrupt modes such as the edge or level of an input that triggers an interrupt.

A level 4 interrupt may be generated by the IDMA upon completion of a data block transfer. Interrupt processing of IDMA transfers is possible by 1) setting the IDMA interrupt enable (bit 11) in the IMR and 2) setting one or both interrupt enable (INTN and INTE) bits in the CMR (see Table D-1). Once in the interrupt handler, four bits in the CSR indicate the reason for termination of an IDMA data block (see Table D-2).

D.6.8 SCC Configuration

SCC1 will be set according the rate adaptation protocol implemented. If V.120 is used, the protocol should be set to HDLC. If V.110 is used, program SCC1 to V.110 (configured under the DDCMP mode). The following values are suitable:

Register	Value	Comments
Configuration Register (SCONE)	$30FF	External receive and transmit clocks will be provided from the IDL through the serial Interface.
Mode Register (SCM1) for V.120	$000C	HDLC mode.
Mode Register (SCM1) for V. 110	$040E	V.110 mode.
Sync Register (DSR1)	$7E7E	HDLC flag.
Sync Register (DSR1)	$0100	V.110 sync pattern
Mask Register (SCCM1)		The mask should be set according to the interrupt events handled.

SCC2 will be set to handle a 9600 baud UART from a non-ISDN terminal. The following values are suitable:

Register	Value	Comments
Configuration Register (SCON2)	$30FF	Set internal receive and transmit clocks, 9600 baud if 16.6-MHz parallel clock used.
Mode Register (SCM2)	$010D	UART mode, 8 bits, no parity
Sync Register (DSR2)	$7E7E	Bits 14-12 control stop-bit shaving.
Mask Register (SCCM2)		The mask should be set in accordance with the interrupt events the software wants to handle.

SCC3 will be set to handle the HDLC frames of LAPD protocol over the D-channel. (Multiplexing SCC3 into IDL leaves the SCP pins available for the SCP port.) The following values are suitable:

Register	Value	Comments
Configuration Register (SCON3)	$3000	External receive and transmit clocks (will be provided from the IDL through the serial interface)
Mode Register (SCM3)	$010C	HDLC mode with retransmit option.
Sync Register (DSR3)	$7E7E	HDLC flag.
Mask Register (SCCM3)	$1B	The mask should be set in accordance with the interrupt events the software wants to handle.

NOTE

In addition to programming the SCCs registers, the protocol-specific parameter RAM of each SCC should be initialized by the control software according to the protocol selected.

using transparent mode with the EXSYN bit set in SCM2.
DSR2 = $7E7E

6. Setting SCCE2 to $FF clears out any current status in the event register. SCCE2 = $FF
7. You must indicate in the SCCM2 register which (if any) transparent events you wish to cause interrupts. Bit 5 should be set to zero. Bit 3 is only valid in BISYNC mode and has no meaning in transparent mode. All other bits are valid. For this example, we will disable interrupts, but all events will still be set in SCCE2.
SCCM2 = $00
8. Setting IMR to $0400 allows interrupts from SCC2 to be enabled in the interrupt controller; however, since SCCM2 = $00, any SCC2 interrupt requests are prevented from reaching the interrupt controller, and this step has no effect.
IMR = $0400
9. Initialize the receive and transmit function codes to 000, and set the receive buffer size to 10 (hex) bytes. These are the general-purpose parameter RAM values for SCC2.
RFCR = $00
TFCR = $00
MRBLR = $0010
10. Initialize the BISYNC parameters. These parameters do not involve transparent mode; however, it is a good idea to initialize them in case BISYNC mode is ever accidentally entered by clearing the NTSYN bit in SCM2. The values for BSYNC and BDLE are arbitrary and were chosen so that the two registers have different values. The control characters table is disabled for good measure, although this too is not used in transparent mode.
PRCRC = $0000
PTCRC = $0000
PAREC = $0000
BSYNC = $0033
BDLE = $0044
CHARACTER1 = $8000
11. The Tx BD buffer starts at address $30000. It is 18 (hex) bytes long. (Notice that the MRBLR value equal to $0010 does not restrict the transmit buffer size.) The status $D800 says that the buffer is ready and in external RAM. Since the I bit is set, the TX bit in the SCCE2 register will be set upon completion, and this is the “last” buffer in this transmission. The next Tx BD is set up so that it is not ready; transmission will halt after one Tx BD.
Tx BD = $D800 $0018 $0003 $0000
Tx BD = $5800 $xxxx $xxxx $xxxx (This Tx BD is not yet ready.)
12. Two empty Rx BDs are needed to receive the transmit frame. Both are currently empty, and data is to be stored in external RAM buffers. Since the I bits are set, the RX bit in the SCCE2 register will be set when each buffer is filled with data. The third Rx BD is not ready yet. If it was ready, it would be filled with all $FFs (idles) after the first two buffers were filled.
Rx BD = $D000 $0000 $0004 $0000
Rx BD = $D000 $0000 $0004 $0010

Table E-1 (c). SCCx Register Set

Initialized by User	Offset Hex	Name	
	00	Reserved	
Yes	02	SCC Configuration Register (SCON)	
Yes	04	SCC Mode Register (SCM)	
Yes	06	SCC Data Synchronization Register (DSR)	
Yes	08	Event Register (SCCE)	Reserved
Yes	0A	Mask Register (SCCM)	Reserved
	0C	Status Register (SCCS)	Reserved
	0E	Reserved	

NOTE: The offset is from the MC68302 base address + ($880 for SCC1, $890 for SCC2, or $8A0 for SCC3).

Table E-1 (d). General Registers (Only One Set)

Initialized by User	Offset Hex	Name	
	860	Command Register (CR)	Reserved
Yes	8B2	Serial Interface Mask Register (SIMASK)	
Yes	8B4	Serial Interface Mask Register (SIMODE)	

NOTE: The offset is from the MC68302 base address.

E.3.1.1 COMMUNICATIONS PROCESSOR (CP) REGISTERS. The CP has one set of three registers that configure the operation of the serial interface for all three SCCs. These registers are discussed in the following paragraphs.

E.3.1.1.1 Command Register (CR). The command register is an 8-bit register located at offset $860 (on D15-D8 of a 16-bit data bus). This register is used to issue commands to the CP. The user should set the FLG bit when a command is written to the command register. The CP clears the FLG bit during command processing to indicate that it is ready for the next command. Reserved bits in registers should be written as zeros.

7	6	5	4	3	2	1	0
RST	GCI	OPCODE		—	CH. NUM.		FLG

RST—Software Reset Command (set by the user and cleared by the CP)

0 = No software reset command issued or cleared by CP during software reset sequence.
1 = Software reset command (FLG bit should also be set if it is not already set).

GCI—GCI Commands

0 = Normal operation.
1 = The OPCODE bits are used for GCI commands (user should set CH. NUM. to 10 and FLG to 1).