μPD7201

Multiprotocol Serial Communications Controller

Technical Manual

CHAPTER 1 INTRODUCTION 1
1.1 Features 1

CHAPTER 2 PIN DESCRIPTION 3

CHAPTER 3 PROTOCOLS 9
3.1 Asynchronous Protocol 9
3.2 Synchronous Character-Oriented Protocols 9
3.3 Synchronous Bit-Oriented Protocols 10

CHAPTER 4 FUNCTIONAL DESCRIPTION 13
4.1 Transmitter 14
4.1.1 Asynchronous Mode 17
4.1.2 COP Synchronous Modes 18
4.1.3 SDLC (HDLC BOP Synchronous) Mode 19
4.2 Receiver 20
4.2.1 Asynchronous Mode 22
4.2.2 COP Synchronous Modes 24
4.2.3 SDLC (HDLC BOP Synchronous) Mode 25
4.3 Bus Interface Controller 26
4.3.1 Bus Control Logic 27
4.3.2 Interrupt Control Logic 27
4.3.3 DMA Control Logic 33
4.3.4 Clock and Reset Control Logic 35

CHAPTER 5 PROGRAMMING THE MPSC^2 37
5.1 MPSC^2 Registers 37
5.1.1 Control Register 0 38
5.1.2 Control Register 1 40
5.1.3 Control Register 2 (Channel A) 43
5.1.4 Control Register 2 (Channel B) 45
5.1.5 Control Register 3 45
5.1.6 Control Register 4 47
5.1.7 Control Register 5 49
5.1.8 Control Register 6 51
5.1.9 Control Register 7 52
5.1.10 Status Register 0 53
5.1.11 Status Register 1 56
5.1.12 Status Register 2 59
5.2 MPSC^2 Programming Examples 60

CHAPTER 6 APPLICATION HINTS 83
6.1 Designing with the MPSC^2 83
6.1.1 8080/86-type Processors 83
6.1.2 Other Processor Types 83
6.2 Using the MPSC^2 with DMA Controllers 85
6.3 Vectored Interrupts without using $\overline{PRI}$ 87
6.4 To DMA or Not to DMA 88
6.5 Handling an SDLC Underrun Fault 88
6.6 Sending Synchronous Pad Characters 89
6.7 Transmitting Bisync Transparent Mode 89
6.8 Vectoring the MPSC^2 in Non-Vectored Mode 89

APPENDIX A COMMANDS AND ELECTRICAL SPECS 91

APPENDIX B REGISTER SUMMARY 97

CHAPTER 1 INTRODUCTION

The NEC uPD7201 Multiprotocol Serial Communications Controller ($MPSC^2$) is a versatile device designed to give you high-level control of your data communication protocols with maximum flexibility and minimum processor overhead. The $MPSC^2$ contains two complete full duplex channels in a 40-pin package and incorporates a variety of sophisticated features to simplify your protocol management.

1.1 Features

Implements the three basic data/communications protocols

- Asynchronous
- Character-oriented synchronous (monosync, bisync, external sync)
- Bit-oriented synchronous (SDLC/HDLC)

Provides extensive error checking

- Parity
- CRC-16
- CRC-CCITT
- Break/Abort detection
- Framing Error detection

Enhanced data reliability

- Double-buffered transmitters
- Quadruply-buffered receivers
- Programmable transmitter underrun handling

Simplified system design

- Simple interface to most microprocessors
- Automatic Interrupt vectoring for most microprocessors
- Four DMA channels for maximum throughput with standard 8237/8257-type DMA controllers
- Single-phase TTL clock
- Single +5 volt supply

Applications

- Computers
- Terminals
- Data communications equipment
- Instrumentation
- Control systems
- Mass storage devices as a serial data controller

This section describes the various pin functions available on the MPSC^2. Some pin numbers are used twice because of their programmability and dual functionality. Those pins that have more than one function are marked with an * in the following descriptions. Refer to Section 5 for detailed information on selecting pin functions.

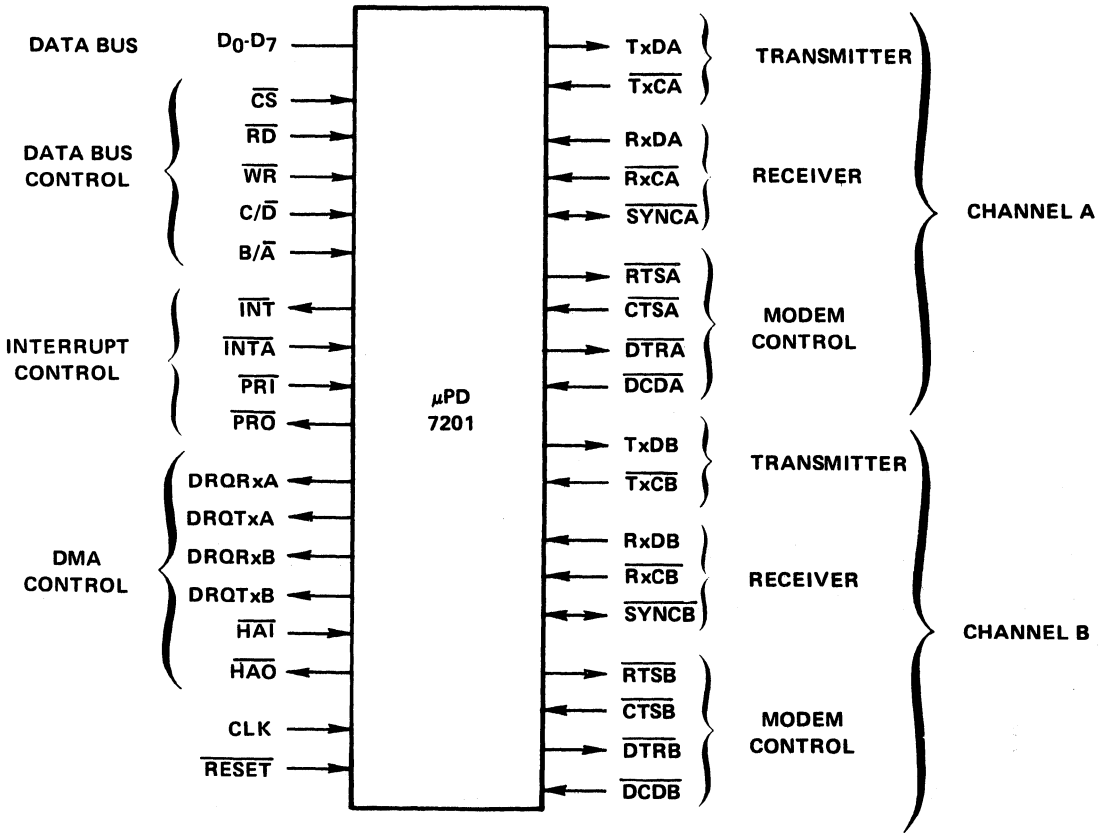


Figure 2.1 Functional Pinout

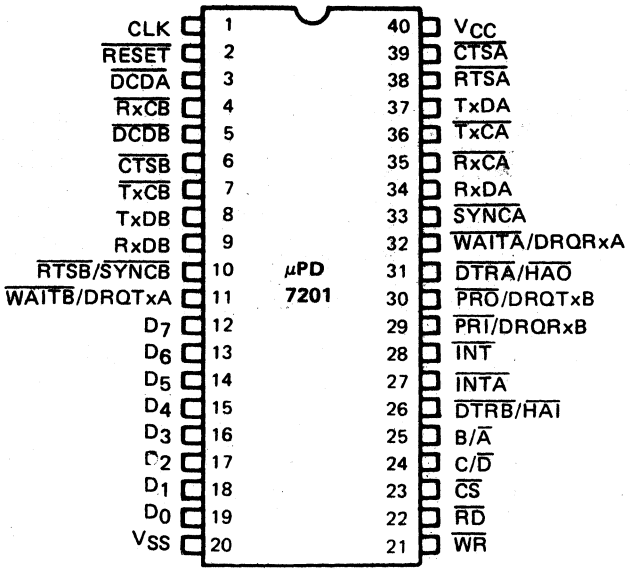


Figure 2.2 Pin Configuration

Pin Descriptions

12-19 D_0-D_7 Data Bus (bidirectional three-state)

The data bus lines are connected to the system data bus. Data or status from the MPSC^2 is output on these lines when $\overline{CS}$ and $\overline{RD}$ are active and data or commands are latched into the MPSC^2 on the rising edge of $\overline{WR}$ when $\overline{CS}$ is active.

23 $\overline{CS}$ Chip Select (input, active low)

Chip select allows the MPSC^2 to transfer data or commands during a read or write cycle.

25 B/$\overline{A}$ Channel Select (input)

A low selects channel A and a high selects channel B for access during a read or write cycle.

24 C/$\overline{D}$ Control/Data Select (input)

This input, with $\overline{RD}$, $\overline{WR}$ and B/$\overline{A}$, selects the data registers (C/$\overline{D}$ = 0) or the control and status registers (C/$\overline{D}$ = 1) for access over the data bus.

22 $\overline{RD}$ Read (input, active low)

This input (with either $\overline{CS}$ during a read cycle or $\overline{HAI}$ during a DMA cycle)

notifies the MPSC^2 to read data or status from the device.

21 $\overline{WR}$ Write (input, active low)
This input (with either $\overline{CS}$ during a read cycle or $\overline{HAI}$ during a DMA cycle) notifies the MPSC^2 to write data or control information to the device.

2 $\overline{RESET}$ Reset (input, active low)
A low on this input (one complete CLK cycle minimum) initializes the MPSC^2 to the following conditions: receivers and transmitters disabled, TxDA and TxDB set to marking (high), and Modem Control Outputs $\overline{DTRA}$, $\overline{DTRB}$, $\overline{RTSA}$, $\overline{RTSB}$ set high. Additionally, all interrupts are disabled, and all interrupt and DMA requests are cleared. After a reset, you must rewrite all control registers before restarting operation.

1 CLK System Clock (input)
A TTL-level system clock signal is applied to this input. The system clock frequency must be at least 4.5 times the data clock frequency applied to any of the data clock inputs $\overline{TxCA}$, $\overline{TxCB}$, $\overline{RxCA}$ or $\overline{RxCB}$.

28 $\overline{INT}$ Interrupt Request (output, open drain, active low)
$\overline{INT}$ is pulled low when an internal interrupt request is accepted.

27 $\overline{INTA}$ Interrupt Acknowledge (input, active low)
The processor generates two or three $\overline{INTA}$ pulses (depending on the processor type) to signal all peripheral devices that an interrupt acknowledge sequence is taking place. During the interrupt acknowledge sequence, the MPSC^2, if so programmed, places information on the data bus to vector the processor to the appropriate interrupt service location.

29* $\overline{PRI}$ Interrupt Priority In (input, active low)
This input informs the MPSC^2 whether the highest priority device is requesting interrupt and is used with $\overline{PRO}$ to implement a priority resolution "daisy chain" when there is more than one interrupting device. The state of $\overline{PRI}$ and the programmed interrupt mode determine the MPSC^2's response to an interrupt acknowledge sequence.

30* $\overline{\text{PRO}}$ Interrupt Priority Out (output, active low)
This output is active when $\overline{\text{HAI}}$ is active and the MPSC^2 is not requesting interrupt ($\overline{\text{INT}}$ is inactive). The active state informs the next lower priority device that there are no higher priority interrupt requests pending during an interrupt acknowledge sequence.

11*, 32* $\overline{\text{WAITA}}$ $\overline{\text{WAITB}}$ Wait (outputs, open drain)
These outputs synchronize the processor with the MPSC^2 when block transfer mode is used. You may program it to operate with either the receiver or transmitter, but not both simultaneously. $\overline{\text{WAIT}}$ is normally inactive. For example, if the processor tries to perform an inappropriate data transfer such as a write to the transmitter when the transmitter buffer is full, the $\overline{\text{WAIT}}$ output for that channel is active until the MPSC^2 is ready to accept the data. The $\overline{\text{CS}}$, C/$\overline{\text{D}}$, B/$\overline{\text{A}}$, $\overline{\text{RD}}$, and $\overline{\text{WR}}$ inputs must remain stable while $\overline{\text{WAIT}}$ is active.

11*, 29*, 30*, 32* DRQTxA, DRQTxB, DRQRxA, DRQRxB
DMA Request (outputs, active high)
When these lines are active, they indicate to a DMA controller that a transmitter or receiver is requesting a DMA data transfer.

26* $\overline{\text{HAI}}$ Hold Acknowledge In (input, active low)
This input notifies the MPSC^2 that the host processor has acknowledged the DMA request and has placed itself in the hold state. The MPSC^2 then performs a DMA cycle for the highest priority outstanding DMA request, if any.

31* $\overline{\text{HAO}}$ Hold Acknowledge Out (output, active low)
This output, with $\overline{\text{HAI}}$, implements a priority daisy chain for multiple DMA devices. $\overline{\text{HAO}}$ is active when $\overline{\text{HAI}}$ is active and there are no DMA requests pending in the MPSC^2.

8, 37 TxDA, TxDB Transmit Data (outputs, marking high)
Serial data from the MPSC^2 is output on these pins.

7, 36 $\overline{\text{TxCA}}$, $\overline{\text{TxCB}}$ Transmitter Clocks (inputs, active low)
The transmit clock controls the rate at which data is shifted out at TxD. You may program the MPSC^2 so that the clock rate is 1x, 16x, 32x, or 64x the data rate. Data changes on the falling edge of $\overline{\text{TxC}}$. $\overline{\text{TxC}}$ features a Schmitt-trigger input for relaxed rise and fall time requirements.

9, 34 RxDA, RxDB Receiver Data (inputs, marking high)
Serial data to the MPSC^2 is input on these pins.

4, 35 $\overline{\text{RxCA}}$, $\overline{\text{RxCB}}$ Receiver Clocks (inputs, active low)
The receiver clock controls the sampling and shifting of serial data at RxD. You may program the MPSC^2 so that the clock rate is 1x, 16x, 32x, or 64x the data rate. RxD is sampled on the rising edge of $\overline{\text{RxC}}$. $\overline{\text{RxC}}$ features a Schmitt-trigger input for relaxed rise and fall time requirements.

26*, 31* $\overline{\text{DTRA}}$, $\overline{\text{DTRB}}$ Data Terminal Ready (outputs, active low)
The $\overline{\text{DTR}}$ pins are general-purpose outputs which may be set or reset with commands to the MPSC^2.

10, 38* $\overline{\text{RTSA}}$, $\overline{\text{RTSB}}$ Request to Send (outputs, active low)
When you operate the MPSC^2 in one of the synchronous modes, $\overline{\text{RTSA}}$ and $\overline{\text{RTSB}}$ are general-purpose outputs that you may set or reset with commands to the MPSC^2. In asynchronous mode, $\overline{\text{RTS}}$ is active immediately as soon as it is programmed on. However, when programmed off, $\overline{\text{RTS}}$ remains active until the transmitter is completely empty. This feature simplifies the programming required to perform modem control.

3, 5 $\overline{\text{DCDA}}$, $\overline{\text{DCDB}}$ Data Carrier Detect (inputs, active low)
Data carrier detect generally indicates the presence of valid serial data at RxD. You may program the MPSC^2 so that the receiver is enabled only when $\overline{\text{DCD}}$ is low. You may also program the MPSC^2 so that any change in state that lasts longer than the minimum specified pulse width causes an interrupt and latches the $\overline{\text{DCD}}$ status bit to the new state.

6, 39 $\overline{\text{CTSA}}$, $\overline{\text{CTSB}}$ Clear to Send (inputs, active low)
Clear to send generally indicates that the receiving modem or peripheral is ready to receive data from the MPSC^2. You may program the MPSC^2 so that the transmitter is enabled only when $\overline{\text{CTS}}$ is low. As with $\overline{\text{DCD}}$, you may program the MPSC^2 to cause an interrupt and latch the new state when $\overline{\text{CTS}}$ changes state for longer than the minimum specified pulse width.

10, 33* $\overline{\text{SYNCA}}$, $\overline{\text{SYNCB}}$ Synchronization (inputs/outputs, active low)
The function of the $\overline{\text{SYNC}}$ pin depends upon the MPSC^2 operating mode. In asynchronous mode, $\overline{\text{SYNC}}$ is an input that the processor can read. It can be programmed to generate an interrupt in the same manner as $\overline{\text{DCD}}$ and $\overline{\text{CTS}}$.

In external sync mode, $\overline{SYNC}$ is an input which notifies the MPSC[2] that synchronization has been achieved (see Figure 2.3 for detailed timing). Once synchronization is achieved, hold $\overline{SYNC}$ low until synchronization is lost or a new message is about to start.

In internal synchronization modes (monosync, bisync, SDLC), $\overline{SYNC}$ is an output which is active wherever a SYNC character match is made (see Figure 2.4 for detailed timing). There is no qualifying logic associated with this function. Regardless of character boundaries, $\overline{SYNC}$ is active on any match.

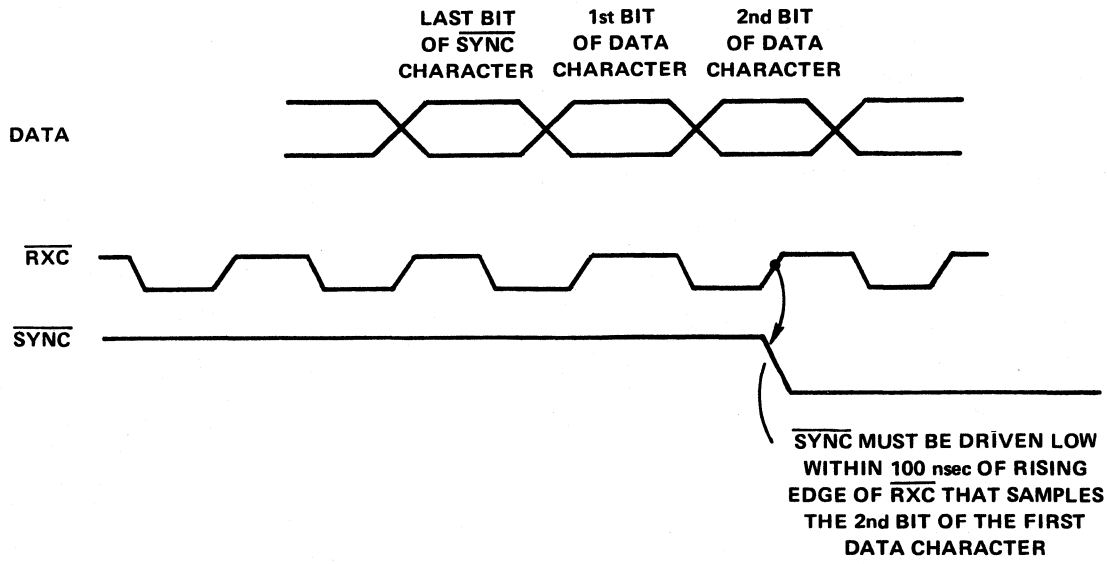


Figure 2.3 $\overline{SYNC}$ Input, External Synchronization

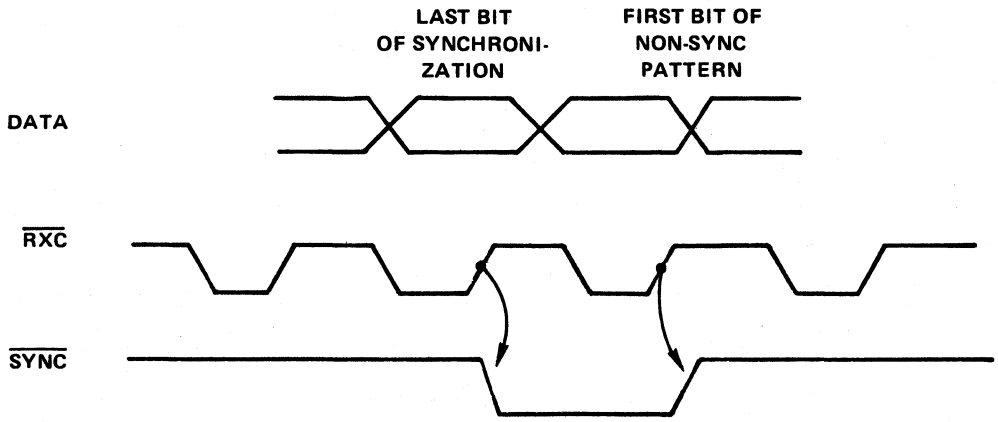


Figure 2.4 $\overline{SYNC}$ Output, Internal Synchronization

CHAPTER 3 PROTOCOLS

A protocol defines a set of rules for transmitting information and control from one place to another. In parallel protocols as you might find on a microprocessor bus, dedicated "control" lines handle functions such as timing, type of information, and error checking. Since the object of serial data communications is to minimize the number of wires, the protocol used must place all of this control information in the serial data stream.

The basic protocol unit or frame can be built into increasingly complex protocols by defining special control characters and fields, and by grouping frames together into larger units. Virtually all communications protocols currently in use are based on one of three basic protocols: Asynchronous, Synchronous Character- or Count-Oriented Protocols (COPs), and Bit-Oriented Protocols (BOPs).

3.1 Asynchronous Protocol

In asynchronous protocol, each character transmitted has its own framing information in the form of a start and stop bit(s). Each character is a "message" in itself and may be asynchronous with respect to any others. You can implement error detection by adding a parity bit to each character. The transmitter makes the parity bit 1 or 0 so that the character plus parity contains an even or odd number of ones for even parity or odd parity, respectively. Figure 3.1 illustrates the asynchronous data format.

3.2 Synchronous Character Oriented Protocols

In synchronous character-oriented protocols (COPs), the start and stop bits associated with each character are eliminated. A synchronization (sync) character that is not part of the data is transmitted before the data to establish proper framing. The synchronization character is usually 8 or 16 bits long. Monosync and IBM Bisync are typical examples of COPs (Figure 3.2). Since the framing information is presented only at the beginning, the transmitter must insert fill characters to maintain synchronization. Sync characters are commonly used for this purpose.

As with the asynchronous protocol, a parity bit may be used with each character to provide error checking. A more reliable check is performed by calculating a special 16-bit block check character called a Cyclic Redundancy Check (CRC) for the entire data block and transmitting this character at the end of the data. The most commonly used CRC polynomial for COPs is called CRC-16.

A disadvantage of the character-oriented protocol is having to use special characters such as $\overline{\text{SYNC}}$ to define various portions of a message when you send non-character binary data ("transparent data" in bisync terminology). To do this, you must transmit special DLE sequences and selectively exclude certain characters from the CRC calculation for both the transmitter and receiver. The MPSC^2 features special circuitry to simplify this operation.

3.3 Synchronous Bit-Oriented Protocols

Synchronous Bit-Oriented Protocols (BOPs) use a special set of rules to distinguish between data and framing characters. This eliminates some of the problems associated with COPs. The most common BOPs in use are the almost-identical HDLC and SDLC protocols shown in Figure 3.3.

The rules for SDLC (henceforth we will refer only to SDLC although the same information applies to HDLC as well) are quite simple. The basic transmission unit is called a frame and is delineated by a special flag character 01111110 (flags cannot be used as filler like the COP sync character). The data or information field may consist of any number of bits; not necessarily an integral number of n-bit characters. Since data could contain the 01111110 pattern, the transmitter performs the following operation: if five consecutive ones are transmitted, the transmitter inserts a zero bit before the next data bit. Likewise, the receiver must delete any zero that follows five consecutive ones. Six consecutive ones indicate a flag character and eight or more ones indicate a special abort condition.

Error checking is done with a 16-bit CRC character inserted between the end of the information field and the End Of Frame flag. The CRC-CCITT polynomial is generally used. The end of a frame is determined by counting 16 bits (CRC) back from the End Of Frame flag. Special circuitry in the receiver must inform the processor of the boundary between the end of the information field and the beginning of the CRC when the information field is not an integral number of n-bit characters. The MPSC[2] performs all of the above functions necessary to implement Bit-Oriented Protocols.

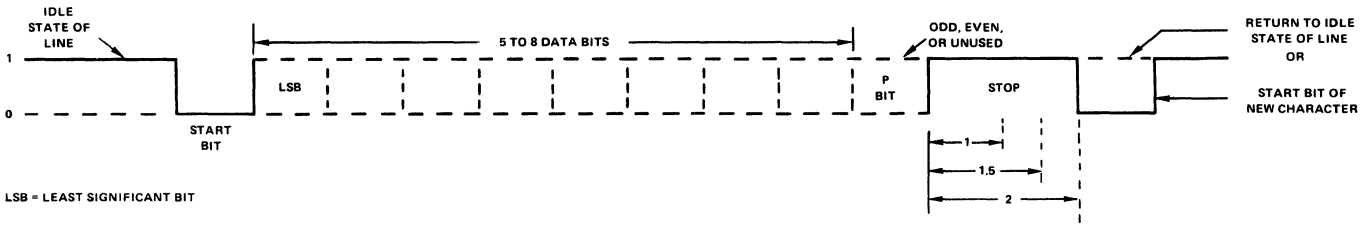


Figure 3.1 - Asynchronous Data Character Format

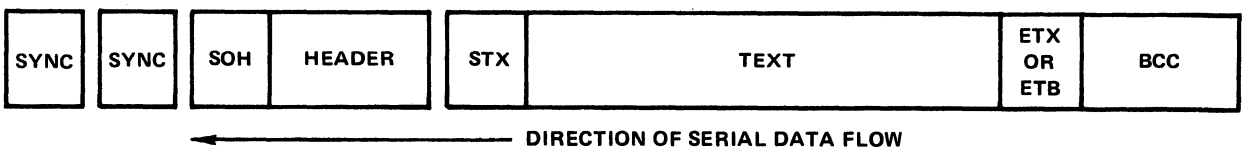


Figure 3.2 - BISYNC Message Format

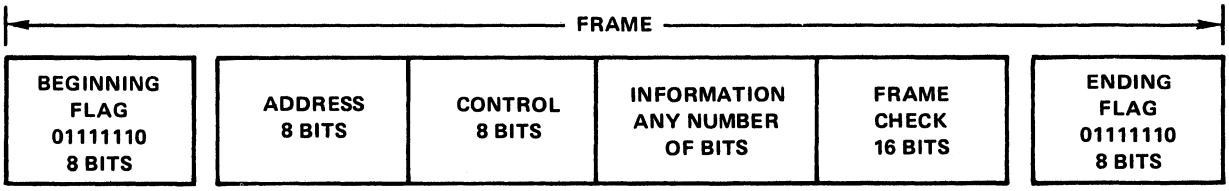


Figure 3.4 - Basic SDLC Frame

CHAPTER 4 FUNCTIONAL DESCRIPTION

The MPSC^2 provides two complete serial communications controllers in a single package implementing the following functions:

Parallel-to-Serial and Serial-to-Parallel data conversion.
Buffering of outgoing and incoming data, allowing the processor time to respond.
Insertion and deletion of framing bits and characters.
Calculation and checking of Parity and CRC error checking.
Informing the processor when and what action needs to be taken.
Interfacing with the outside world over discrete modem control lines.

The MPSC^2 can be logically divided into the following functional groups (Figure 4.1):

Two identical serial I/O controller channels, each consisting of a Transmitter section and a Receiver section,

and a common Processor Interface that connects the MPSC^2 with the host processor and provides overall device control.

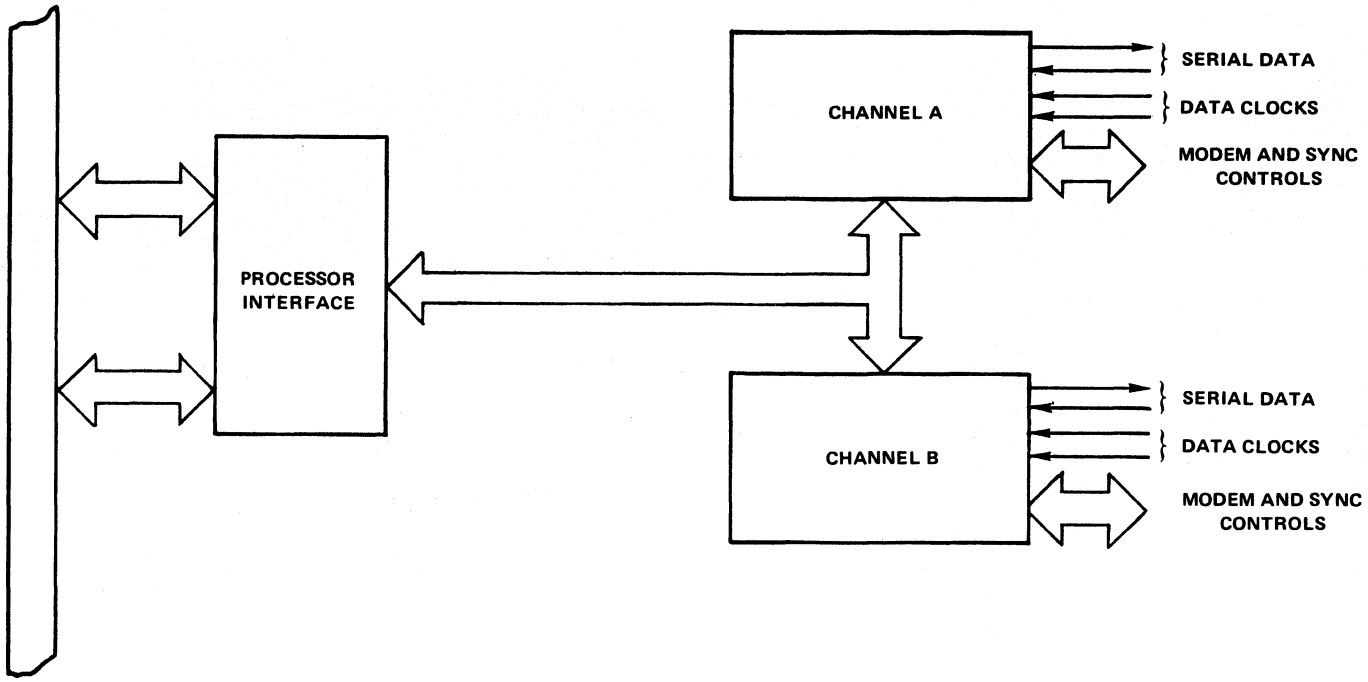


Figure 4.1 - Block Diagram

4.1 Transmitter

The MPSC^2 Transmitter performs all the functions necessary to convert parallel data to the appropriate serial bit streams required by various protocols. The major components of the transmitter are shown in Figure 4.2. Control and status register fields pertinent to the operation of the transmitter are summarized in Table 4.1.

The primary data flow through the transmitter begins at the internal data bus. There, characters written to the MPSC^2 are placed in the buffer register. When any character present in the shift register has been transferred out, or if the shift register is empty, the contents of the buffer register are transferred to the shift register and output with the least significant bit first. Then, a Transmitter Buffer Becoming Empty indication (flag) is given. This double buffering allows the processor one full character time from this flag to respond with the next character without interrupting data transmission. You should note that it is the transfer of a character from the data buffer to the shift register rather than the empty condition itself that causes the Transmitter Buffer Becoming Empty indication. At initialization or after a Reset Transmitter Interrupt/DMA Pending Command is issued to control register 0 (CR0) you must write one character to the buffer to reset this flag. The Transmitter Buffer Empty bit in status register 0 (SR0), always reflects the presence or absence of a character in the buffer.

After a hardware or software reset, the transmitter data output (TxD) is in high (marking) state. You can pull TxD low (spacing) any time by setting the Send Break bit (CR5 bit 4). TxD remains low until the Send Break bit is reset and any data currently being transmitted is destroyed.

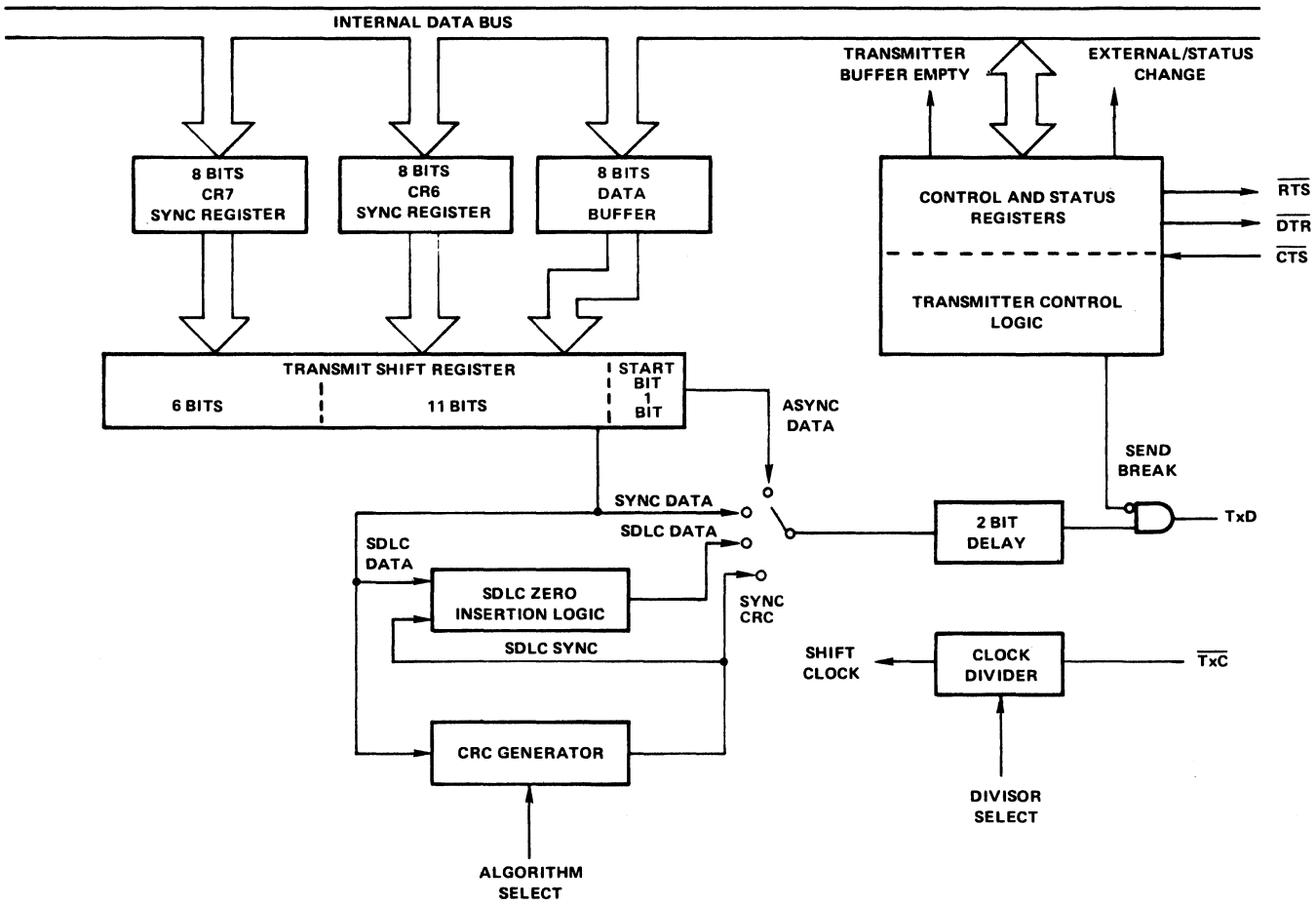


Figure 4.2 - Block Diagram MPSC^2 Transmitter

Table 4.1 - Transmitter Control and Status Registers

CONTROL REGISTER	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	CRC CONTROL		COMMAND			REGISTER POINTER		
1							Trans. Int Enable	Ext/Status Int Enable
4	Clock Mode		Sync Format Select		Sync/Async Mode Select		Parity Control	
5	DTR	Bits/Char		Send Break	Transmitter Enable	CRC Type	RTS	CRC Enable
6	SYNC 1							
7	SYNC 2							
3			Auto Enables					

STATUS REGISTER	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0		Trans Underrun/ $\overline{EOM}$	CTS			Trans Buffer Empty		
1								All Async Characters Sent

You can change the number of bits transmitted for each character at any time by modifying the bits/char field (CR5, D_5-D_6) before you load the character into the buffer.

The rate at which data is shifted out is determined by the transmitter clock input ($\overline{TxC}$) and the clock mode field (CR4 Bits 6-7). You can select a clock divisor so that the data clock ($\overline{TxC}$) rate is equal to 1x, 16x, 32x, or 64x the actual data rate. This field also controls the receiver clock and must be set to 1x for synchronous modes (see Section 4.2.2 for use in asynchronous mode). Each new bit is shifted out on the falling edge of $\overline{TxC}$.

The following is a general discussion of the operation of the MPSC^2 in various protocol modes. For a detailed description of the registers and examples, see Chapter 5.

4.1.1 Asynchronous Mode

After you select asynchronous mode, initialize the various parameters (number of bits/character, number of stop bits, etc.) and enable the transmitter (CR5 bit 3 = 1). TxD remains in the high (marking) state. When the first character is written to the data buffer, it is transferred to the shift register and the Transmitter Buffer Becoming Empty flag is set. A parity bit, if enabled, and the specified number of stop bits (1, 1 1/2 or 2) are appended to the character. The character plus the start bit are shifted out serially through a one-bit delay. After the character has been completely sent, the next character is loaded into the shift register and the process continues. When no more characters are available, TxD remains high and the All Async Characters Sent flag (SR1 bit 0) is set until the next character is loaded. The transmitter may be disabled at any time (CR5 bit 3 = 0); however, transmission of the character currently being sent, if any, is completed. Disabling the transmitter does not reset the Transmitter Buffer Becoming Empty flag or any resultant interrupts or DMA requests. You can clear this flag either by writing a character to the data buffer for later transmission or by issuing a Reset Transmitter Interrupt/DMA Pending Command.

The modem control output $\overline{RTS}$ (Request To Send) may be set or reset at any time with CR5 bit 1. $\overline{RTS}$ immediately goes to the active state (low) when this bit is set. When reset, $\overline{RTS}$ does not go high until the shift register and the data buffer are empty.

The function of the modem control input, $\overline{CTS}$ (Clear To Send), depends upon the Auto Enables Control (CR3 bit 5). When Auto Enables is reset, any transition of $\overline{CTS}$ sets the External/Status Change flag but has no affect upon transmission. When Auto Enables is set, character transmission cannot begin until $\overline{CTS}$ goes low. If $\overline{CTS}$ goes high, any character currently being transmitted is completed and the transmitter is then disabled until $\overline{CTS}$ again goes low. The $\overline{CTS}$ flag, SR0 bit 5, reflects the inverted state of the external $\overline{CTS}$ pins, that is, $\overline{CTS}$ flag = 1 when $\overline{CTS}$ = low.

4.1.2 COP Synchronous Modes

The MPSC^2 gives you three distinct COP operating modes: monosync (8-bit sync character), bisync (16-bit sync character), and external sync (the transmitter operates in the same manner as Monosync). When bisync mode is selected, you should program the eight least significant bits (first byte) of the sync character into CR6 and the eight most significant bits (second byte) into CR7. For monosync and external sync modes you should program CR6 with the 8-bit sync character.

During operation in COP modes, the MPSC^2 transmitter may be in any one of the following phases:

Disabled Phase: Transmitter Enable is off (CR5, D3=0) or CTS is low when the auto enables function is used;

Idle Phase: Sync characters are being sent;

Data Phase: Data from the processor is being transmitted;

CRC Phase: (if CRC is used) when the CRC check characters are being transmitted.

After selecting the desired protocol and initializing parameters, the transmitter enters and remains in the Disabled Phase, with TxD high until the Transmitter Enable bit is set. Once this is done the transmitter enters the Idle Phase, transmits the first sync character and continues to send sync characters until a character is written into the transmit buffer. When the first data character is loaded into the data buffer and the current sync character has been sent, the transmitter enters Data Phase and sends data characters while setting the Transmitter Buffer Becoming Empty flag each time it is ready for the next character.

During the Data Phase, the transmitter may run out of data to send for one of two reasons: (1) The processor is busy and is not able to provide the next data characters within a message, or (2) the data portion of the message is complete and it is time to enter the CRC Phase (or the Idle Phase if CRC is not used). The MPSC^2 automatically handles both of these conditions through a mechanism called the Idle/CRC Latch, the state of which may be read from SR0 D_6.

When the transmitter is initialized the Idle/CRC Latch is set, indicating that the transmitter will enter the Idle Phase and begin sending sync characters when there is no data to send. Entering this phase also sets the Transmitter Buffer Becoming Empty flag (if not already set) to indicate with SR0 $D_6 = 1$, that the Idle Phase has been entered.

However, if you reset the Idle/CRC Latch with a Reset Idle/CRC Latch command to CR0, a lack of data causes the MPSC^2 to enter the CRC Phase and begin sending the 16-bit CRC character calculated up to that point. Entering the CRC Phase sets the Idle/CRC Latch which, in turn, sets the External/Status Change flag indicating that the MPSC^2 is sending CRC. After you reset the flag, you may send the next data character to the transmitter and it will be sent immediately following the CRC, or you may do nothing. In either case, the Idle/CRC Latch is now set again so the transmitter enters the Idle Phase when no further data is available.

You can disable the transmitter during any phase of operation. If the transmitter is disabled during the Idle or Data Phases the MPSC^2 finishes sending the current character and goes to the Disabled Phase (TxD high). If disabled during the CRC Phase, a 16-bit CRC is sent; however, the remainder of the CRC is supplanted by sync with bit positions matching.

The CRC Generator may be programmed to either of two polynomials, CRC-16 ($x^{16} + x^{15} + x^2 + 1$) or CRC-CCITT ($x^{16} + x^{12} + x^5 + 1$). The CRC Generator may be reset to 0 at any time by issuing a Reset CRC Generator Command to CR0. Since it is sometimes necessary to exclude certain characters from the CRC calculation, the MPSC^2 features a CRC enable/disable control (CR5 D_0) that may be changed just prior to loading a character into the transmitter buffer to include or exclude that and subsequent characters in the CRC calculation.

4.1.3 SDLC (/HDLC BOP Synchronous) Mode

In SDLC mode, the MPSC^2 transmitter operates similarly to monosync transmission with the following exceptions:

WR6 is not used for the transmitter sync character. SDLC flags (sync) are generated internally.

Data and CRC are passed through zero insertion logic before transmission. This logic inserts a 0 bit after transmitting five contiguous ones to distinguish information from framing flags.

A special Send SDLC Abort Command is available in CR0. Issuing this command causes at least 8 but less than 14 ones to be transmitted, destroying any data in the transmitter shift register and buffer. After sending the abort, the transmitter enters Idle Phase.

Resetting the CRC generator initializes it to all ones rather than zeroes and the result bits are inverted before transmission.

4.2 Receiver

The MPSC^2 receiver reverses the process performed by the transmitter. It converts the serial data stream of the various protocols back to parallel data for the processor. The major components of the receiver are shown in Figure 4.4. Control and status registers pertinent to the operation of the receiver are summarized in Table 4.2.

The primary data path through the receiver begins at the receiver data input RxD. Data passes through a two-bit time delay and into the receiver shift register (the sync data path is described later). The point of entry into the shift register and hence the number of bits per character is determined by the mode of operation and the Bits/Character field of CR3 (D_6-D_7). You can change this field at any time provided that the character that is currently being assembled has not yet reached the new number of bits/character. If the number of bits/character specified is less than eight, the character appears right-justified in the data buffer (with the parity bit, if parity is enabled) and the left side is filled with ones (see Figure 4.3).

1	1	P	D_4	D_3	D_2	D_1	D_0

5 BITS/CHARACTER; PARITY ENABLED

Figure 4.3 Data Format Example for Less Than 8 Bits/Character

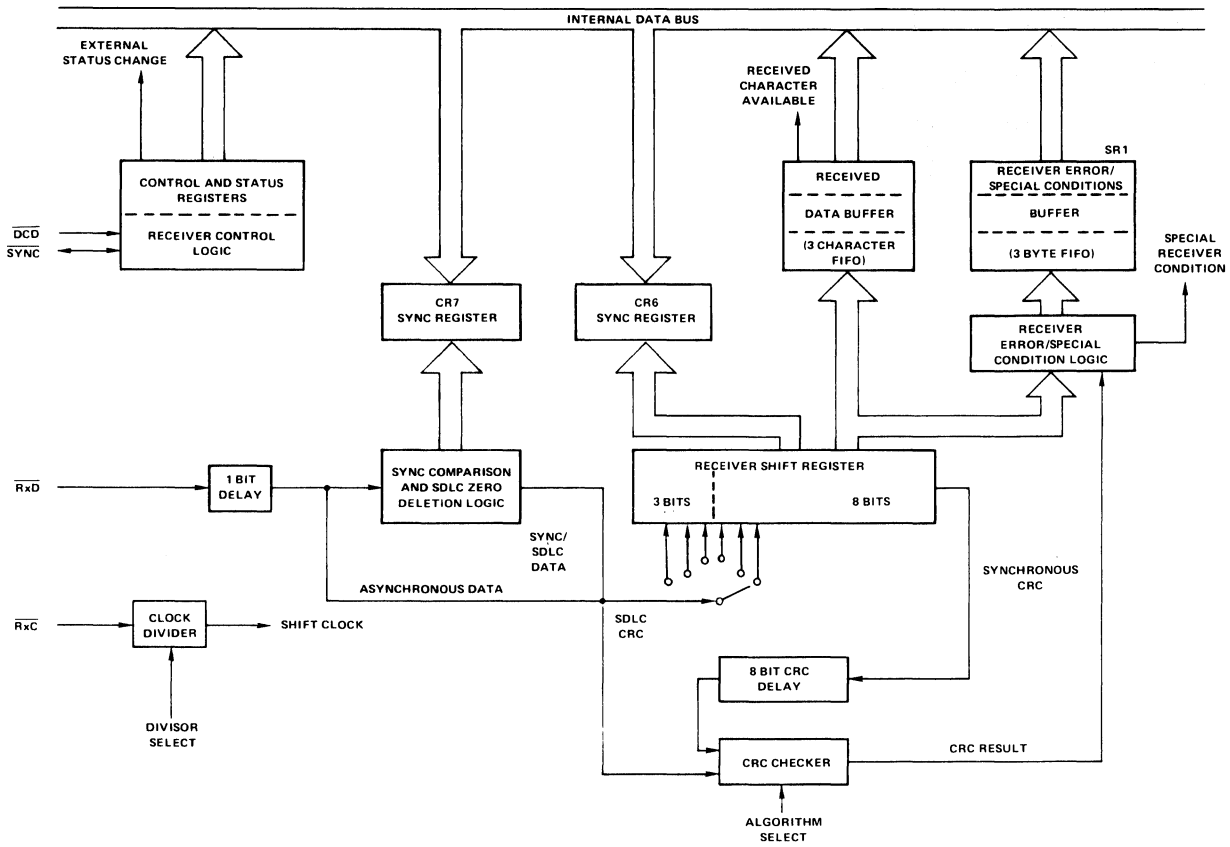


Figure 4.4 Block Diagram MPSC^2 Receiver

Table 4.2 Receiver Control and Status Registers

CONTROL REGISTER	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	CRC CONTROL		COMMAND			REGISTER POINTER		
1				Receiver Interrupt Control				Ext/Status Interrupt Enable
3	Bits/Char		Auto Enables	Enter Sync Hunt Phase	Receiver CRC Enable	SDLC Address Search mode	Sync Char Load Inhibit	Receiver Enables
4	Clock Mode		Sync Format Select		Sync/Async Mode Select		Parity Control	
5						CRC Type		
6	SYNC 1							
7	SYNC 2							

STATUS REGISTER	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	Break/ Abort			Sync/Hunt Mode	DCD			Received Character Available
1	SDLC End of Frame	CRC/ Framing Error	Receiver Overrun Error	Parity Error	SDLC I-Field Residue Code			

Once the character has been assembled in the shift register, it is passed to a three-character First In-First Out buffer (FIFO) and the Received Character Available flag (and SR0 D_0) is set to inform the processor that a character is available. The three-character buffer allows the processor up to four character times to service the receiver without losing data. This feature enhances data reliability at high speeds while relaxing software timing requirements. The Received Character Available flag is reset when all characters in the buffer have been read, i.e., the buffer is empty.

As each character is transferred to the buffer, it is checked for errors or special conditions and that information is placed in a parallel FIFO error buffer so that the status associated with each character can be read with that character through status register 1. Reading a character from the data buffer moves the next character and its status to the top of the FIFO. You should read the status first, if it is of interest, and then the data.

The rate at which data is shifted into the receiver is controlled by the receiver clock input ($\overline{RxC}$) and the clock mode field (CR4 D_6-D_7). This field also controls the transmitter clock mode. In any of the synchronous modes, you must select the 1x clock mode. In asynchronous mode you may select a divisor such that the clock rate ($\overline{RxC}$) equals 1x, 16x, 32x, or 64x the actual data rate. However, if you select the 1x mode, the clock must be externally synchronized with the data (see Section 4.1.3). RxD is always sampled on the rising edge of $\overline{RxC}$.

The data carrier detect ($\overline{DCD}$) input works the same way as $\overline{CTS}$ except that it enables the receiver when auto enables is set.

4.2.1 Asynchronous Mode

After initializing and enabling the MPSC^2 Receiver, the receiver logic begins sampling the RxD input for a high-to-low (marking-to-spacing) transition on each rising edge of $\overline{RxC}$. When the transition is found, the receiver waits 1/2 bit time, (for example, eight clock periods if the clock mode is 16x) and samples again to ensure that RxD is still low, improving the MPSC^2's noise immunity. If RxD is still low, the MPSC^2 assumes this is the middle of the start bit and one bit time later begins to sample RxD to assemble the required number of data and parity (if enabled) bits.

Once the character is assembled, the MPSC^2 waits one more bit time and again samples RxD. If RxD is not high, the stop bit is missing and a Framing Error is indicated when the character is passed to the data buffer. If a Framing Error has occurred, the MPSC^2 receiver waits 1/2 bit time before beginning to sample again to avoid interpreting the Framing Error as a new start bit.

Note that in the 1x Clock mode, the receiver simply waits one clock period after the first high-to-low transition is detected and then begins assembling the character. It is for this reason that data and clock must be synchronized in this mode.

The Break/Abort bit, D_7 of SR0 is set when a null character plus Framing Error is detected (i.e. RxD is low for more than one full character time). Break detection also sets the External/Status Change flag. When RxD returns high and the break has ended, D_7 is reset to 0 and the External Status Change flag is once again set. After the break, a single null character is present in the data buffer. It should be read and discarded.

The following errors may occur during operation and are flagged in status register 1.

Framing Error — See above discussion.

Parity Error — If parity is enabled and a parity error occurs, the Parity Error bit D_4 is set. Once a Parity Error has occurred, the Parity Error bit remains set for subsequent characters until reset by an Error Reset command to CR0. You need only check the end of a message or block to determine if a parity error occurred.

Overrun Error — If the data buffer is full with three characters and a fourth character is received, the last character in the buffer is overwritten and the Overrun Error bit D_5 is set. Like Parity Error, Overrun Error remains set until the Error Reset command is issued.

4.2.2 COP Synchronous Modes

The MPSC^2 gives you three distinct COP operating modes: (1) monosync (8-bit sync character), (2) bisync (16-bit character), and (3) external sync (the $\overline{\text{SYNC}}$ pin is used as an input to inform the MPSC^2 that synchronization has been achieved externally).

When monosync mode is selected, CR7 should be programmed with the 8-bit sync character to be matched by the receiver.

In bisync mode CR6 should contain the least significant bits (first byte) and CR7 should contain the most significant bits (second byte) of the 16-bit character to be matched.

In external sync mode, no sync character is required by the receiver. During operation in the COP modes, the MPSC^2 receiver is in one of two phases: (1) Sync Hunt Phase or (2) Data Phase. The receiver automatically enters Sync Hunt Phase when it is enabled (CR3, D_0).

In monosync mode, the incoming data stream passes through and is compared to the sync character in CR7. When a match is found, the receiver switches to Data Phase and begins to pass data to the shift register. If you determine at any time that synchronization has been lost, you may re-enter the Sync Hunt Phase by setting the Enter Hunt Phase bit (D_4) in CR3. When the Hunt Phase is entered or left, the External/Status Change flag is set. When SR0 D_4 (Sync/Hunt) = one, it indicates that the receiver is in Hunt Phase.

Operation is similar in bisync mode, however, when a match is found, CR6 is also checked against the shift register contents and the Hunt Phase is left only if the bytes match. In both monosync and bisync modes, the $\overline{\text{SYNC}}$ pin is used as an output which goes momentarily low any time a sync pattern is detected whether the receiver is in Hunt or Data Phase. See Figure 2.3 for a detailed timing diagram.

You can inhibit the transfer of sync characters to the data register by setting the Sync Char Load Inhibit bit (CR3, D_1). Since the CRC calculation on sync is not inhibited by this bit, you should use it only to strip leading sync characters from a message if you are using CRC Block Check.

Because of the 8-bit delay between the shift register and the CRC checker, CRC status (SR1, D_6) is not valid immediately after the CRC character is received. CRC status is valid 16 bit times after the last CRC character is transferred to the receive buffer, or 20 bit times after the last CRC bit is shifted in at RxD.

4.2.3 SDLC (/HDLC BOP Synchronous) Mode

The MPSC^2 provides you with high-level processing capability for handling bit-oriented protocols. When you select SDLC Mode, CR7 must be programmed with the SDLC Flag character 01111110.

When operating in SDLC mode, the receiver can be in one of three phases: Hunt Phase, Address Search Phase, or Data Phase.

The receiver automatically enters Hunt Phase when first enabled. The incoming data stream passes through the one-bit delay and enters the Sync Comparison/Zero Deletion logic where the following three operations are performed.

First, whenever a 0 bit follows five consecutive ones, that 0 is deleted from the data stream. Second, if six consecutive ones are received, a Flag Character Received indication is given internally. Third, if eight or more ones are received, an abort is indicated and the External/Status Change Flag is set. Flags and aborts are not transferred to the receiver shift register.

Once a flag is detected, the receiver leaves Hunt Phase (setting the External/Status Change Flag) and, if Address Search Mode (CR3-D_2) is enabled, it enters Address Search Phase. Once this phase is entered, the MPSC^2 receiver compares the first 8-bit non-flag character with the contents of control register 6. If the two values match, or the received character is the global address 11111111, the receiver immediately enters Data Phase and character assembly begins with this character. If no match is found and the value is not the global address, the receiver remains in Address Search Phase and no data characters are assembled until a flag followed by the correct address is encountered. If Address Search Mode is not enabled, Data Phase is entered immediately and character assembly begins with the first non-flag character. Since all messages are framed with flag characters, you can skip an incoming message at any time simply by setting the Enter Hunt Phase bit (D_4) in CR3.

Once in Data Phase, characters are assembled according to the number of bits or characters specified until the next End of Frame flag is encountered. The receiver then sets the Special Receive Condition flag and transfers the character currently being assembled to the receiver buffer regardless of the number of bits actually assembled. A special residue code placed in the status buffer (SR1) uses the number of bits assembled to indicate the boundary between the data and CRC characters (see Section 5.1 for a more detailed description of the residue code). If Address Search Mode is enabled, the receiver once again enters Address Search Phase.

Unlike the COP mode of operation, data from the Sync Comparison/Zero Deletion logic passes directly to the CRC checker. As a result, when the End of Frame Flag is detected, the CRC calculation is complete and the error status is passed to the status buffer along with the residue code. The CRC checker is automatically reset to all ones at this time.

4.3 Bus Interface Controller

The bus interface controller is the interface between the transmitter and receiver sections and the processor bus. The major components of this section are shown in Figure 4.5. The control and status registers pertinent to the operation of the control section are illustrated in Table 4.4.

The bus interface controller can be divided into four major components:

Bus Control Logic
Interrupt Control Logic
DMA Control Logic
Clock and Reset Control Logic

All of these components interact to provide a flexible high-performance interface between the bus architecture defined by your processor and application and the various internal elements that make up the MPSC^2.

4.3.1 Bus Control Logic

The bus control logic determines the direction and internal source or destination of data and control transfers between the MPSC^2 and the processor bus. During operation of the MPSC^2, the bus control logic may operate in any of three distinct modes: Processor Read/Write, Interrupt Acknowledge, and DMA Cycle. These last two modes are described in detail in Sections 4.3.2 and 4.3.3.

Processor Read/Write mode is the normal mode of operation. The processor transfers data or commands and status to or from the MPSC^2 with its instruction set. The MPSC^2 is enabled for Processor Read/Write mode when the chip select ($\overline{CS}$) input is made active (low). The direction of the transfer is controlled by enabling either the read ($\overline{RD}$) or write ($\overline{WR}$) inputs. The B/$\overline{A}$ input determines the source/destination channel for the transfer and the C/$\overline{D}$ input specifies whether the transfer is character data or control/status information. These inputs are generally connected to the two low-order address lines. Figure 6.1 illustrates a typical connection between a processor and the MPSC^2.

Table 4.3 Read/Write Selection

$\overline{CS}$	B/$\overline{A}$	C/$\overline{D}$	$\overline{RD}$	$\overline{WR}$	OPERATION
1	X	X	X	X	NO OPERATION. THE MPSC^2 IS DESELECTED.
0	X	X	1	1	NO OPERATION. THE MPSC^2 IS DESELECTED.
0	0	0	1	0	WRITE A CHAR TO CHANNEL A TRANSMITTER.
0	0	0	0	1	READ A CHAR FROM CHANNEL A RECIVER.
0	0	1	1	0	WRITE A CONTROL BYTE TO CHANNEL A.
0	0	1	0	1	READ A STATUS BYTE FROM CHANNEL A.
0	1	0	1	0	WRITE A CHAR TO CHANNEL B TRANSMITTER.
0	1	0	0	1	READ A CHAR FROM CHANNEL B RECEIVER.
0	1	1	1	0	WRITE A CONTROL BYTE TO CHANNEL B.
0	1	1	0	1	READ A STATUS BYTE FROM CHANNEL B.
0	X	X	0	0	ILLEGAL.

4.3.2 Interrupt Control Logic

The interrupt control logic performs two functions: it prioritizes various internal input requests, and places the appropriate information on the data bus during an Interrupt Acknowledge cycle (if you enabled the MPSC^2's vectored interrupt feature).

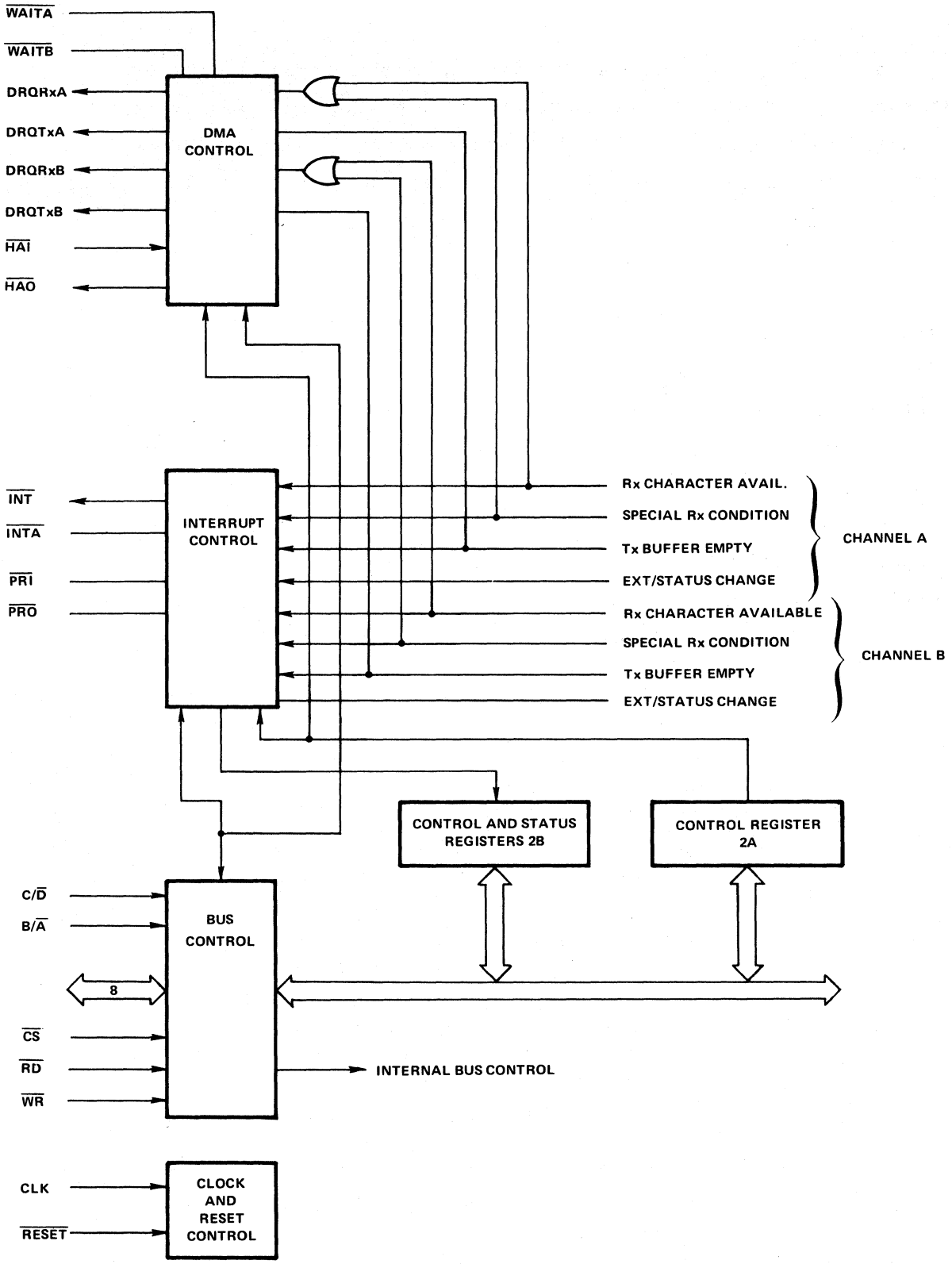


Figure 4.5 - Bus Interface Controller

Table 4.4 - Bus Interface Controller Control and Status Registers

CONTROL REGISTER	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	*Relevant commands
CR0			COMMAND*			REGISTER POINTER			Channel Reset
CR2A		0	Vector Mode Select			Priority	DMA Mode Select		End of Interrupt
CR2B	INTERRUPT VECTOR								

STATUS REGISTER	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CR0							Interrupt Pending	
CR2B	INTERRUPT VECTOR							

Each MPSC[2] channel can generate four different types of interrupt requests:

Received Character Available
Special Received Condition (character received but with an error or SDLC End of Frame flag received)
Transmitter Buffer Empty
External input ($\overline{CTS}$, $\overline{DCD}$, $\overline{SYNC}$, Internal Status (Sync,Idle/CRC Latch) Change)

When any of these requests occurs, the interrupt control logic determines whether to accept the request at that time, issue an interrupt request by setting the $\overline{INT}$ output low when the request is accepted, and, if Vectored Interrupt mode is enabled, place the interrupt information on the data bus during the times that the interrupt acknowledge input ($\overline{INTA}$) is activated by the processor.

As an example, assume that the channel A $\overline{DCD}$ input has just changed state causing an External/Status Change interrupt request. The following sequence occurs:

If all the following conditions are true:

External/Status Change interrupts are enabled

No higher priority interrupt requests are pending

$\overline{PRI}$ is active

The MPSC^2 is not acknowledging a pending lower priority interrupt request

Then the interrupt control logic accepts the interrupt request and sets $\overline{INT}$ active and $\overline{PRO}$ inactive.

If Vectored Interrupt mode is enabled, the MPSC^2 may place information on the data bus in response to a series of $\overline{INTA}$ pulses as shown in the following chart.

Table 4.5 Vectored Interrupt Mode

Interrupt Mode Select	PRI	INTA Cycle		
		1	2	3
8080/5 Master	0	CD HEX (CALL OP)	VECTOR	0
	1	CD HEX (CALL OP)	HI-Z	HI-Z
8080/5 Slave	0	HI-Z	VECTOR	0
	1	HI-Z	HI-Z	HI-Z
8086	0	HI-Z	VECTOR	*
	1	HI-Z	HI-Z	*

*The 8086 issues 2 Interrupt Acknowledge pulses rather than 3.

When operating in the 8080/5 modes, the MPSC^2 issues an 8080-type CALL CD vv Hex instruction where vv is the contents of control register 2B (modified by the cause of the interrupt if the Status Affects Vector feature is enabled). In particular, an MPSC^2 programmed for 8085 Master mode always places the CALL opcode on the data bus regardless of whether that MPSC^2 has a pending interrupt request. To avoid problems caused by momentary bus contention, you should never program more than one device to operate in this mode.

In 8086 mode, the MPSC^2 places the vector on the data bus during the second interrupt acknowledge to vector the processor to the approximate location in low memory.

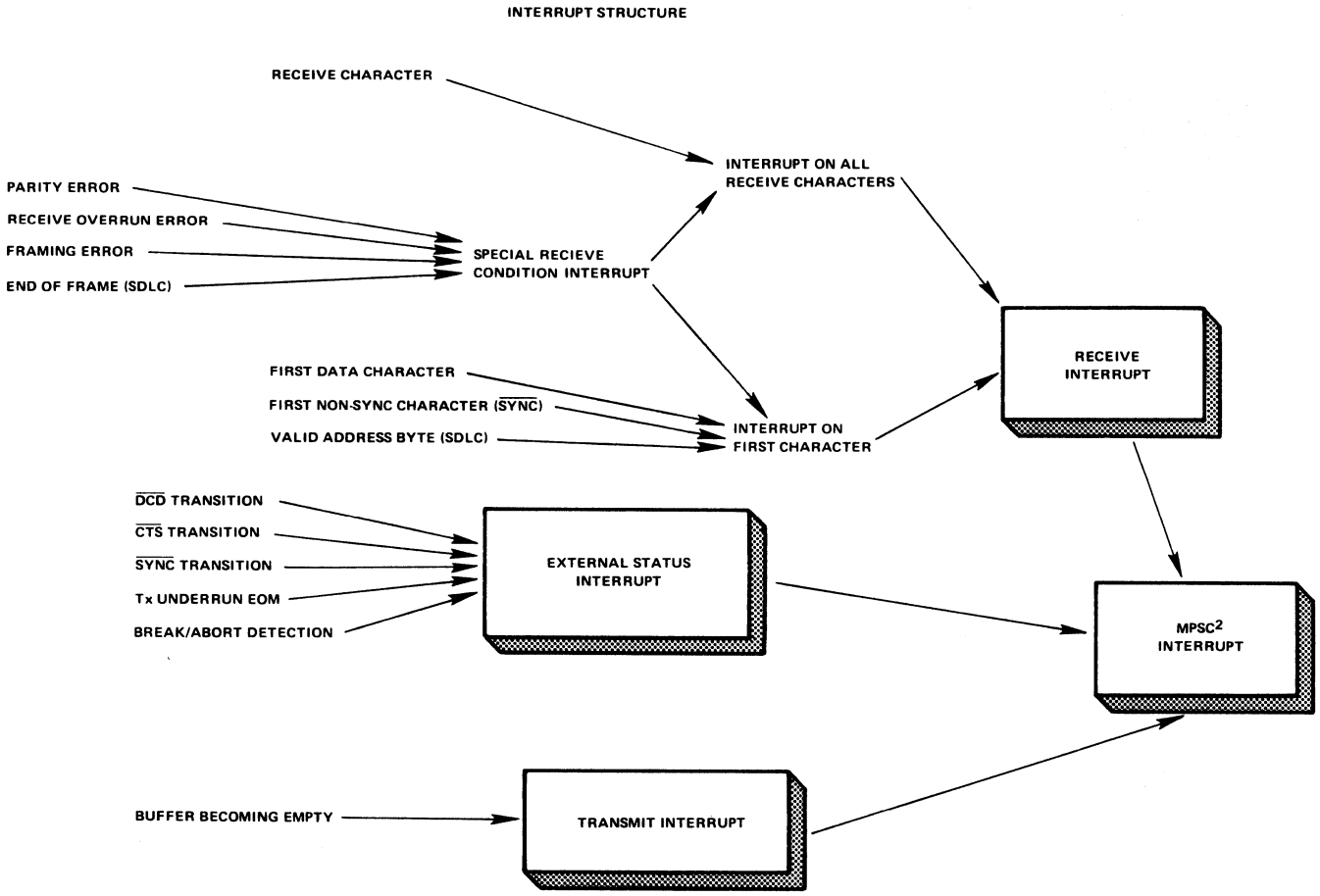


Figure 4.6 MPSC^2 Interrupt Conditions

Figure 4.7 illustrates the action of the interrupt control logic during an interrupt acknowledge sequence.

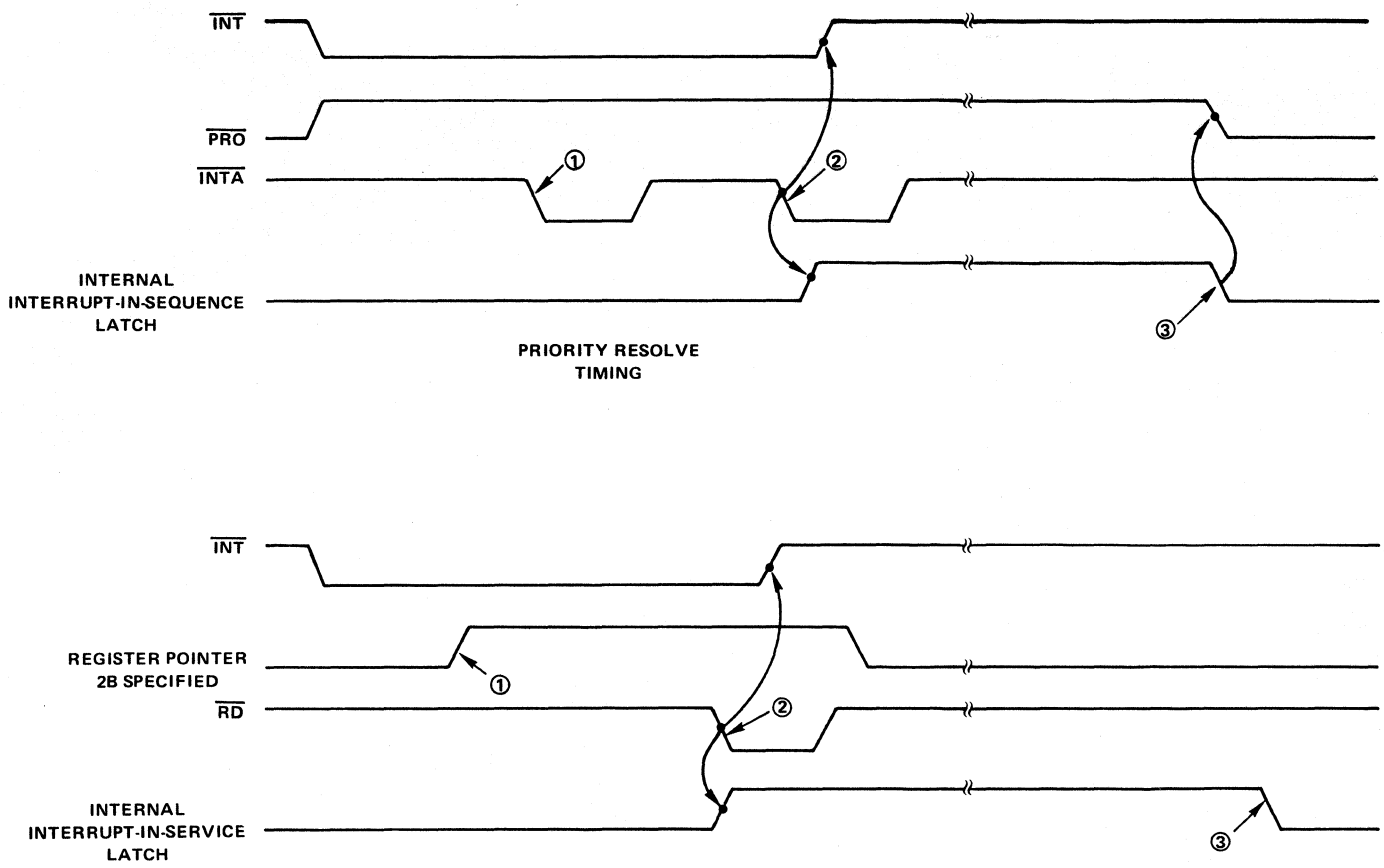


Figure 4.7 Interrupt Timing

At the beginning of the first Interrupt Acknowlege cycle, the interrupt prioritization logic is frozen to permit any late interrupt requests by higher priority devices to ripple through and resolve internal priorities before the second interrupt pulse.

At the end of the second $\overline{\text{INTA}}$ pulse, the $\overline{\text{INT}}$ output is released by the acknowledging device and the interrupt prioritization logic is re-enabled with an Interrupt In Service flag set. As long as this flag is set, $\overline{\text{PRO}}$ is held high and only internal interrupt requests with a priority higher than the one currently being serviced are accepted.

While the interrupt is being serviced, the processor issues an End of Interrupt (EOI) command to the MPSC^2 to reset the interrupt control logic to its previous state. This scheme permits nested interrupts to be serviced and the priority daisy chain to be properly maintained.
When the MPSC^2 is operated in Non-vectored Interrupt mode, the interrupt control logic operates in a similar manner except that $\overline{\text{INTA}}$ is not used and no vector information is placed on the data bus. Rather, the interrupt acknowledge sequence is simulated by reading the vector (modified if Status Affects Vector is enabled) in status register 2B.

4.3.3 DMA Control Logic

The function of the DMA logic is somewhat similar to that of the interrupt control logic in that service requests must be accepted, prioritized, and information placed on (or, in this case, accepted from as well) the data bus at the appropriate times. However, the purpose of the DMA control logic is to enable the MPSC^2 to avoid interrupting the processor to make a data transfer. This is accomplished by activating an external controller to move the data directly from the MPSC^2 to memory, or vice versa.

The DMA control logic accepts requests from four sources: (1) Received Data Available in channel A, (2) Transmitter Buffer Becoming Empty in channel A., (3) Data Available in channel B, and (4) Transmitter Buffer Becoming Empty in channel B. When an internal DMA request is made by one of the above sources and DMA mode is enabled for that channel, the appropriate DMA request output (e.g. DRQRxA when received data is available in channel A) is made active. This causes the external DMA controller to request control of the processor bus with a hold request. The MPSC^2's daisy chain output, $\overline{HAO}$, is at this point locked in the inactive (high) state.

Some time later, the external DMA controller gains control of the processor bus as the processor asserts its hold acknowledge output.

The DMA Controller now places the source or destination address on the address bus and asserts the I/O read or write control line for a data transfer from or to the MPSC^2, respectively. The MPSC^2 also receives the processor hold acknowledge signal possibly through higher priority MPSC^{2}s not requesting DMA, at its $\overline{HAI}$ input. When $\overline{HAI}$ is asserted, the DMA control logic freezes all internal requests, determines which one has the highest priority, and performs the transfer when I/O read or write is received from the DMA controller at $\overline{RD}$ or $\overline{WR}$. Once the transfer is complete, the prioritization logic is re-enabled and new or pending requests can be serviced. Figure 4.8 illustrates some of the timing details of a DMA transfer.

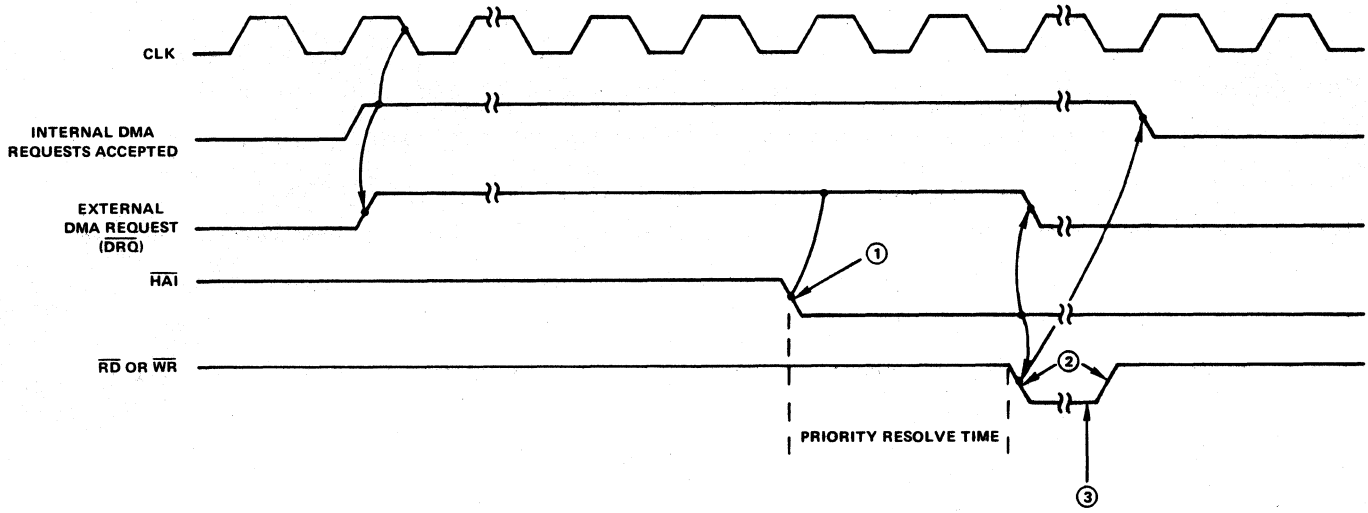


Figure 4.8 DMA Data Transfer Timing

From the above explanation you should note two points. First, in the case of multiple DMA requests from one MPSC^2, both the MPSC^2 and the external DMA controller establish priorities independently to determine which request to service first. As a result, you MUST connect the MPSC^2's DMA request outputs to the DMA controller so that both make the same priority decisions. For example, when using the MPSC^2 with an 8257-type DMA controller and the priority bit (CR2A-D_2) = 0, you must set the controller to the fixed priority mode (as opposed to rotating priority), and connect the MPSC^2's DRQRxA output to the 8257's DRQ 0 input, DRQTxA to DRQ 1, and so on.

The second point is that many DMA controllers, such as the 8257, may begin the transfer by asserting $\overline{RD}$ or $\overline{WR}$ before the MPSC^2 can receive $\overline{HAI}$ through the daisy chain and resolve request proirities. Because of this, you should always derive HLDA to the DMA Controller from $\overline{HAI}$ of the MPSC^2(s) to which it is connected. Additionally, a delay circuit from $\overline{HAI}$ to HLDA is recommended. Figure 6.5 shows a typical MPSC^2/DMA interface which conforms to these points.

The mechanism that controls the $\overline{WAIT}$ outputs of the MPSC^2 is related to the DMA logic. When enabled, the wait logic pulls the $\overline{WAIT}$ line active when the processor attempts to perform a data transfer operation at an inappropriate time. If $\overline{WAIT}$ is connected to the processor's $\overline{WAIT}$ (or READY) input, it waits until the line is released by the MPSC^2 before completing the data transfer. Since the processor is dedicated to either a read or write operation at any one time, only one $\overline{WAIT}$ output is required for each channel. You may assign it to operate with either the transmitter or the receiver. Figure 4.9 illustrates the basic wait feature timing.

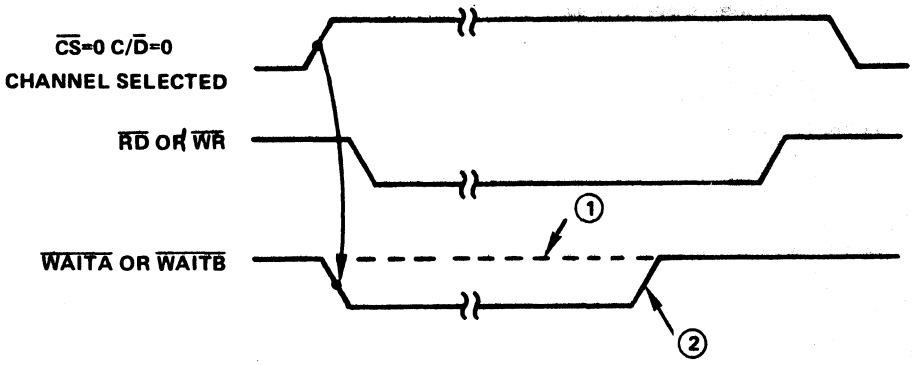


Figure 4.9 Wait Mode Timing

4.3.4 Clock and Reset Control Logic

The clock input of the MPSC^2 controls the various timing states of the MPSC^2 and is usually connected to the processor clock. The clock is not used by the bus control logic and data transfers need not be synchronized to it in any way. The receiver and transmitter sections use the clock, and it must be at least 4.5x the highest data clock frequency you plan to use. The DMA control logic also uses the clock, and it should be the same clock seen by the external DMA Controller.

The $\overline{\text{RESET}}$ input is used at power-up and at any other time that you wish to reset the MPSC^2 to its initial state. After a reset, all transmitters and receivers are disabled, any pending interrupt and DMA requests are cleared, and the modem control outputs $\overline{\text{DTR}}$ and $\overline{\text{RTS}}$ are reset (high). When you reset the MPSC^2, you must hold the $\overline{\text{RESET}}$ input low for at least one complete clock cycle.

CHAPTER 5 PROGRAMMING THE MPSC^2

The software operation of the MPSC^2 is very straightforward. Its consistent register organization and high-level command structure help to minimize the number of operations required to implement complex protocol designs. Programming is further simplified by the MPSC^2's extensive interrupt and status reporting capabilities.

This section is divided into two parts. The first is a detailed description of the commands, bits, and fields in the various MPSC^2 control and status registers. The second part provides programming examples and flowcharts for the MPSC^2's various operating modes to assist you in developing software for your specific application.

5.1 The MPSC^2 Registers

The MPSC^2 interfaces to the system software with a number of control and status registers associated with each channel. Commonly used commands and status bits are accessed directly through control and status registers 0. Other functions are accessed indirectly with a register pointer to minimize the address space that must be dedicated to the MPSC^2.

Table 5.1 Control Registers

CONTROL REGISTER	FUNCTION
0	FREQUENTLY USED COMMANDS AND REGISTER POINTER CONTROL
1	INTERRUPT CONTROL
2	PROCESSOR/BUS INTERFACE CONTROL
3	RECEIVER CONTROL
4	MODE CONTROL
5	TRANSMITTER CONTROL
6	SYNC/ADDRESS CHARACTER
7	SYNC CHARACTER

Table 5.2 Status Registers

STATUS REGISTER	FUNCTION
0	BUFFER AND "EXTERNAL/STATUS" STATUS
1	RECEIVED CHARACTER ERROR AND SPECIAL CONDITION STATUS
2 (CHANNEL B ONLY)	INTERRUPT VECTOR

All control and status registers except CR2 are separately maintained for each channel. Control and status registers 2 are linked with the overall operation of the MPSC^2 and have different meanings when addressed through different channels.

When initializing the MPSC^2, control register 2A (and 2B if desired) should be programmed first to establish the MPSC^2 processor/bus interface mode. You may then program each channel to be used separately, beginning with control register 4 to set the protocol mode for that channel. The remaining registers may then be programmed in any order.

5.1.1 Control Register 0

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CRC CONTROL COMMAND		COMMAND			REGISTER POINTER		

Figure 5.1 Control Register 0

Register Pointer (D_0-D_2)

The register pointer specifies which register number is accessed at the next Control Register Write or Status Register Read. After a hardware or software reset, the register pointer is set to 0. Therefore, the first control byte goes to control register 0. When the register pointer is set to a value other than 0, the next control or status ($C/\overline{D}$ = 1) access is to the specified register, after which the pointer is reset to 0. You can freely combine other commands in control register 0 with setting the register pointer.

Command (D_3-D_5)

Commands commonly used during the operation of the MPSC^2 are grouped in control register 0. They are:

Null (000)

This command has no effect and is used when you wish to set only the register pointer or issue a CRC command.

Send Abort (001)

When operating in SDLC mode, this command causes the MPSC^2 to transmit the SDLC abort code, issuing 8 to 13 consecutive ones. Any data currently in the transmitter or the transmitter buffer is destroyed. After sending the abort, the transmitter reverts to the Idle Phase (flags).

Reset External/Status Interrupts (010)

When the External/Status Change flag is set, the condition bits D_0-D_2 of status register 0 are latched to allow you to capture short pulses that may occur. The Reset External/Status Interrupts Command clears a pending interrupt and re-enables the latches so that new interrupts may be sensed.

Channel Reset (011)

This command has the same effect on a single channel as an external reset at pin 2. A channel reset command to channel A resets the internal interrupt prioritization logic. This does not occur when you issue a Channel Reset command to channel B. You must reinitialize all control registers associated with the channel that you reset. After a channel reset, you must wait at least four system clock cycles before writing new commands or controls to that channel.

Enable Interrupt on Next Character (100)

When operating the MPSC^2 in Interrupt on First Received Character mode, you may issue this command at any time (generally at the end of a message), to re-enable the interrupt logic for the next received character.

Reset Pending Transmitter Interrupt/DMA Request (101)

You can reset a pending Transmitter Buffer Becoming Empty interrupt or DMA request without sending another character by issuing this command (typically at the end of a message). A new Transmitter Buffer Becoming Empty interrupt or DMA request is not made until another character has been loaded and transferred to the transmitter shift register or when, if operating in synchronous or SDLC mode, the CRC character has been completely sent and the first sync or flag character loaded into the transmitter shift register.

Error Reset (110)

This command resets a Special Receive Condition interrupt. It also re-enables the Parity and Overrun Error latches that allow you to check for these errors at the end of a message.

End of Interrupt (111) (Channel A only)

Once an interrupt request has been issued by the MPSC[2], all lower priority internal and external interrupts in the daisy chain are held off to permit the current interrupt to be serviced while allowing higher priority interrupts to occur. At some point in your interrupt service routine (generally at the end), you must issue the End of Interrupt command to channel A to re-enable the daisy chain and allow any pending lower priority internal interrupt requests to occur.

CRC Control Commands (D_6-D_7)

These commands control the operation of the CRC generator/checker logic.

Null (00)

This command has no effect and is used when issuing other commands or setting the register pointer.

Reset Receiver CRC Checker (01)

This command resets the CRC checker to 0 when the channel is in a synchronous mode and resets to all ones when in SDLC mode.

Reset Transmitter CRC Generator (10)

This command resets the CRC generator to 0 when the channel is in a synchronous mode and resets to all ones when in SDLC mode.

Reset Idle/CRC Latch (11)

This command resets the Idle/CRC latch so that when a transmitter underrun condition occurs (that is, the transmitter has no more characters to send), the transmitter enters the CRC Phase of operation and begins to send the 16-bit CRC character calculated up to that point. The latch is then set so that if the underrun condition persists, idle characters are sent following the CRC. After a hardware or software reset, the latch is in the set state.

5.1.2 Control Register 1

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
WAIT FUNCTION ENABLE	ø	WAIT ON RECEIVER TRANSMITTER	RECEIVER INTERRUPT MODE		CONDITION AFFECTS VECTOR	TRANSMITTER INTERRUPT ENABLE	EXT/STATUS INT ENABLE

Figure 5.2 Control Register 1

External/Status Interrupt Enable (D_0)

When this bit is set to one, the MPSC^2 issues an interrupt whenever any of the following occur:

transition of $\overline{DCD}$ input
transition of $\overline{CTS}$ input
transition of $\overline{SYNC}$ input
entering or leaving synchronous Hunt Phase break detection or termination
SDLC abort detection or termination
Idle/CRC latch becoming set (CRC being sent)

Transmitter Interrupt Enable (D_1)

When this bit is set to one, the MPSC^2 issues an interrupt when:

the character currently in the transmitter buffer is transferred to the shift register (Transmitter Buffer Becoming Empty) or,

the transmitter enters Idle Phase and begins transmitting sync or flag characters.

Status Affects Vector (D_2) (Programmed in Channel B for both channels)

When this bit is set to 0, the fixed vector programmed in CR2B during MPSC^2 initialization is returned in an interrupt acknowledge sequence. When this bit is set to 1, the vector is modified to reflect the condition that caused the interrupt. See Section 5.1.12 for a detailed explanation of the MPSC^2's vectored interrupt feature.

Receiver Interrupt Mode (D_3-D_4)

This field controls how the MPSC^2's interrupt/DMA logic handles the character received condition.

Receiver Interrupts/DMA Request Disabled (00)

The MPSC^2 does not issue an interrupt or a DMA request when a character has been received.

Interrupt on First Received Character Only (01)

(and issue a DMA Request)

In this mode, the MPSC^2 issues an interrupt only for the first character received after an Enable Interrupt on First Character Command (CR0) has been given. If the channel is in DMA mode, a DMA request is issued for each character received including the first. This mode is generally used when using the MPSC^2 in DMA or Block Transfer mode to signal the processor that the beginning of an incoming message has been received.

Interrupt (and issue a DMA Request) (10)

On All Received Characters

Parity Error is a Special Receive Condition

In this mode, an interrupt (and DMA request if DMA mode is selected) is issued whenever there is a character present in the receiver buffer. A parity error is considered a special receive condition.

Interrupt (and issue a DMA request) (11)

On All Received Characters

Parity Error is not a Special Receive Condition

This mode is the same as above except that a parity error is not considered a special receive condition. The following are considered special receive conditions and, when status affects vector is enabled, cause an interrupt vector different from that caused by a received character available condition:

Receiver Overrun Error
Asynchronous Framing Error
Parity Error (if specified)
SDLC End of Message (final flag received)

Wait on Receiver/Transmitter (D_5)

If the Wait function is enabled for Block mode transfers, setting this bit to 0 causes the MPSC^2 to issue a wait ($\overline{WAIT}$ output goes low) when the processor attempts to write a character to the transmitter while the transmitter buffer is full. Setting this bit to 1 causes the MPSC^2 to issue a wait when the processor attempts to read a character from the receiver while the receiver buffer is empty.

Wait Function Enable (D_7)

Setting this bit to 1 enables the wait function as described above and in Section 4.3.3.

5.1.3 Control Register 2 (Channel A)

D7	D6	D5	D4	D3	D2	D1	D0
PIN 10 $\overline{SYNCB}$/RTSB	0	INTERRUPT VECTOR MODE			PRIORITY	DMA MODE SELECT	

Figure 5.3 Control Register 2 (Channel A)

DMA Mode Select (D_0-D_1)

Setting this field establishes whether channels A and B are used in DMA mode (i.e. data transfers are performed by a DMA controller) or in non-DMA mode where transfers are performed by the processor in either Polled, Interrupt, or Block Transfer modes. The functions of some MPSC^2 pins are also controlled by this field.

Table 5.3 DMA Mode Selection

		Channel		Pin Function					
D_1	D_0	A	B	11	26	29	30	31	32
0	0	Non-DMA	Non-DMA	$\overline{WAITB}$	$\overline{DTRB}$	$\overline{PRI}$	$\overline{PRO}$	$\overline{DTRA}$	$\overline{WAITA}$
0	1	DMA	Non-DMA	DRQTxA	$\overline{HAI}$	$\overline{PRI}$	$\overline{PRO}$	$\overline{HAO}$	DRQRxA
1	0	DMA	DMA	DRQTxA	$\overline{HAI}$	DRQRxB	DRQTxB	$\overline{HAO}$	DRQRxA
1	1	Illegal	–	–	–	–	–	–	–

Priority (D_2)

This bit allows you to select the relative priorities of the various interrupt and DMA conditions according to your application.

Table 5.4 DMA/Interrupt Priorities

	Mode		DMA Priority Relation	Interrupt Priority Relation
D_2	CHA	CHB		
0	INT	INT	———————	RxA > TxA > RxB > TxB > ExTA > ExTB
1			———————	RxA > RxB > TxA > TxB > ExTA > ExTB
0	DMA	INT	RxA TxA	RxA > RxB > TxB > ExTA > ExTB
1			RxA TxA	RxA > RxB > TxB > ExTA > ExTB
0	DMA	DMA	RxA TxA RxB TxB	RxA > RxB > ExTA > ExTB
1			RxA RxB TxA TxB	RxA > RxB > ExTA > ExTB

Interrupt Vector Mode (D_3-D_5)

This field determines how the MPSC^2 responds to an interrupt acknowledge sequence from the processor. See Section 4.3.2 for a detailed description of the MPSC^2 response in these modes.

Table 5.5 Interrupt Acknowledge Sequence Response

D_5	D_4	D_3	Mode	Status Register 2B and Interrupt Vector bits affected when Condition Affects Vector is enabled
0	0	0	Non-Vectored	D_4 D_3 D_2
0	0	1	Non-Vectored	D_4 D_3 D_2
0	1	0	Non-Vectored	D_2 D_1 D_0
0	1	1	Illegal	–
1	0	0	8085 Master	D_4 D_3 D_2
1	0	1	8085 Slave	D_4 D_3 D_2
1	1	0	8086	D_2 D_1 D_0
1	1	1	Illegal	–

Pin 10 $\overline{\text{SYNCB}}$/$\overline{\text{RTSB}}$ Select (D_7)

Programming a 0 into this bit selects $\overline{\text{RTSB}}$ as the function of pin 10. A one selects $\overline{\text{SYNCB}}$ as the function.

5.1.4 Control Register 2 (Channel B)

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
INTERRUPT VECTOR							

Figure 5.4 Control Register 2 (Channel B)

Interrupt Vector (D_0-D_7)

When the MPSC^2 is used in Vectored Interruput mode, the contents of this register are placed on the bus during the appropriate portion of the interrupt acknowledge sequence. Its value is modified if status affects vector is enabled. You can read the value of CR2B at any time. This feature is particularly useful in determining the cause of an interrupt when using the MPSC^2 in Non-vectored Interrupt mode.

5.1.5 Control Register 3

D_7	D6	D5	D4	D3	D2	D1	D0
NUMBER OF RECEIVED BITS/CHARACTER		AUTO ENABLES	ENTER HUNT PHASE	RECEIVER CRC ENABLE	ADDRESS SEARCH MODE	SYNC CHARACTER LOAD INHIBIT	RECEIVER ENABLE

Figure 5.5 Control Register 3

Receiver Enable (D_0)

After the channel has been completely initialized, setting this bit to 1 allows the receiver to begin operation. You may set this bit to 0 at any time to disable the receiver.

Sync Character Load Inhibit (D_1)

In a synchronous mode, this bit inhibits the transfer of sync characters to the receiver buffer, thus performing a "sync stripping" operation. When using the MPSC^2's CRC checking ability, you should use this feature only to strip leading sync characters preceding a message since the load inhibit does not exclude sync characters embedded in the message from the CRC calculation. Synchronous protocols using other types of block checking such as checksum or LRC are free to strip embedded sync characters with this bit.

Address Search Mode (D_2)

In SDLC Mode, setting this bit places the MPSC^2 in Address Search mode where character assembly does not begin until the 8-bit character (secondary address field) following the starting flag of a message matches either the address programmed into CR6 or the global address 11111111.

Receiver CRC Enable (D_3)

This bit enables and disables (1 = enable) the CRC checker in COP mode to allow you to selectively include or exclude characters from the CRC calculation. The MPSC^2 features a one-character delay between the receiver shift register and the CRC checker so that the enabling or disabling takes effect with the last character transferred from the shift register to the receiver buffer. Therefore, you have one full character time in which to read the character and decide whether it should be included in the CRC calculation.

Enter Hunt Phase (D_4)

Although the MPSC^2 receiver automatically enters Sync Hunt Phase after a reset, there are times when you may wish to reenter it, such as when you have determined that synchronization has been lost or, in SDLC mode, to ignore the current incoming message. Writing a 1 into this bit at any time after initialization causes the MPSC^2 to reenter Hunt Phase.

Auto Enables (D_5)

Setting this bit to 1 causes the $\overline{\text{DCD}}$ and $\overline{\text{CTS}}$ inputs to act as enable inputs to the receiver and transmitter, respectively.

Number of Received Bits/Character (D_6-D_7)

This field specifies the number of data bits assembled to make each character.

You may change the value on the fly while a character is being assembled and if the change is made before the new number of bits has been reached, it affects that character. Otherwise the new specifications take effect on the next character received.

Table 5.6 Received Bits/Character

D_7	D_6	BITS/CHARACTER
0	0	5
0	1	7
1	0	6
1	1	8

5.1.6 Control Register 4

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CLOCK RATE		SYNC MODE		NUMBER OF STOP BITS SYNC MODE		PARITY EVEN/ODD	PARITY ENABLE

Figure 5.6 Control Register 4

Parity Enable (D_0)

Setting this bit to 1 adds an extra data bit containing parity information to each transmitted character. Each received character is expected to contain this extra bit and the receiver parity checker is enabled.

Parity Even/Odd (D_1)

Programming a 0 into this bit when parity is enabled causes the transmitted parity bit to take on the value required for odd parity. The received character is checked for odd parity. Conversely, a 1 in this bit signifies even parity generation and checking.

Number of Stop Bits/Sync Mode (D_2-D_3)

This field specifies whether the channel is used in synchronous (or SDLC) mode or in asynchronous mode. In asynchronous mode, this field also specifies the number of bit times used as the stop bit length by the transmitter. The receiver always checks for one stop bit.

Table 5.7 Stop Bits

D_3	D_2	MODE
0	0	SYNCHRONOUS MODES
0	1	ASYNCHRONOUS 1 BIT TIME (1 STOP BIT)
1	0	ASYNCHRONOUS 1½ BIT TIMES (1½ STOP BITS)
1	1	ASYNCHRONOUS 2 BIT TIMES (2 STOP BITS)

Sync Mode (D_4-D_5)

When the Stop Bits/Sync Mode field is programmed for synchronous modes (D_2 D_3 = 00), this field specifies the particular synchronous format to be used. This field is ignored in asynchronous mode.

Table 5.8 Synchronous Formats

SYNC MODE 1	SYNC MODE 2	
D_5	D_4	MODE
0	0	8-BIT INTERNAL SYNCHRONIZATION CHARACTER (MONOSYNC)
0	1	16-BIT INTERNAL SYNCHRONIZATION CHARACTER (BISYNC)
1	0	SDLC
1	1	EXTERNAL SYNCHRONIZATION ($\overline{\text{SYNC}}$ PIN BECOMES AN INPUT)

Clock Rate (D_6-D_7)

This field specifies the relationship between the transmitter and receiver clock inputs ($\overline{\text{TxC}}$, $\overline{\text{RxC}}$) and the actual data rate at TxD and RxD. When operating in a synchronous mode you must specify a 1x clock rate. In asynchronous modes, any of the rates may be specified, however, with a 1x clock rate the receiver cannot determine the center of the start bit. In this mode, you must externally synchronize the sampling (rising) edge of $\overline{\text{RxC}}$ with the data.

Table 5.9 Clock Rates

CLOCK RATE 1	CLOCK RATE 2	
D_7	D_6	CLOCK RATE
0	0	CLOCK RATE = 1x DATA RATE
0	1	CLOCK RATE = 16x DATA RATE
1	0	CLOCK RATE = 32x DATA RATE
1	1	CLOCK RATE = 64x DATA RATE

5.1.7 Control Register 5

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
DTR	NUMBER OF TRANSMITTED BITS/CHARACTER		SEND BREAK	TRANSMITTER ENABLE	CRC POLYNOMIAL SELECT	$\overline{RTS}$	TRANSMITTER CRC ENABLE

Figure 5.7 Control Register 5

Transmitter CRC Enable (D_0)

A 1 or a 0 enables or disables, respectively, CRC generator calculation. The enable or disable does not take effect until the next character is transferred from the transmitter buffer to the shift register, thus allowing you to include or exclude specific characters from the CRC calculation. By setting or resetting this bit just before loading the next character, it and subsequent characters are included or excluded from the calculation. If this bit is 0 when the transmitter becomes empty, the MPSC^2 goes to the Idle Phase, regardless of the state of the Idle/CRC latch.

$\overline{RTS}$ (D_1)

In synchronous and SDLC modes, setting this bit to 1 causes the RTS pin to go low while a 0 causes it to go high. In asynchronous mode, setting this bit to 0 does not cause $\overline{RTS}$ to go high until the transmitter is completely empty. This feature facilitates programming the MPSC^2 for use with asynchronous modems.

CRC Polynomial Select (D_2)

This bit selects the polynomial used by the transmitter and receiver for CRC generation and checking. A 1 selects the CRC-16 polynomial $(x^{16} + x^{15} + x^2 + 1)$. A 0 selects the CRC-CCITT Polynomial $(x^{16} + x^{12} + x^5 + 1)$. In SDLC mode, you must select CRC-CCITT. You may use either polynomial in other synchronous modes.

Transmitter Enable (D_3)

After a reset, the transmitted data output (TxD) is held high (marking) and the transmitter is disabled until this bit is set.

In asynchronous mode, TxD remains high until data is loaded for transmission.

In synchronous and SDLC modes, the MPSC^2 automatically enters Idle Phase and sends the programmed sync or flag characters.

When the transmitter is disabled in asynchronous mode, any character currently being sent is completed before TxD returns to the marking state.

If you disable the transmitter during the Data Phase in synchronous mode, the current character is sent, then TxD goes high (marking).

In SDLC mode, the current character is sent, but the marking line following is zero-inserted. That is, the line goes low for one bit time out of every five.

You should never disable the transmitter during the SDLC Data Phase unless a reset is to follow immediately. In either case, any character in the buffer register is held.

Disabling the transmitter during the CRC Phase causes the remainder of the CRC character to be bit-substituted with sync (or flag). The total number of bits transmitted is correct and TxD goes high after they are sent.

If you disable the transmitter during the Idle Phase, the remainder of the sync (flag) character is sent, then TxD goes high.

Send Break (D_4)

Setting this bit to 1 immediately forces the transmitter output (TxD) low (spacing). This function overrides the normal transmitter output and destroys any data being transmitted although the transmitter is still in operation. Resetting this bit releases the transmitter output.

Transmitted Bits/Character (D_5-D_6)

This field controls the number of data bits transmitted in each character. You may change the number of bits/character by rewriting this field just before you load the first character to use the new specification.

Table 5.10 Transmitted Bits/Character

TRANSMIT BITS PER CHARACTER 1	TRANSMIT BITS PER CHARACTER	
D_6	D_5	BITS/CHARACTER
0	0	5 OR LESS (SEE BELOW)
0	1	7
1	0	6
1	1	8

Normally each character is sent to the MPSC^2 right-justified and the unused bits are ignored. However, when sending five bits or less the data should be formatted as shown below to inform the MPSC^2 of the precise number of bits to be sent.

Table 5.11 Transmitted Bits/Character for 5 Characters and Less

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	NUMBER OF BITS/CHARACTER
1	1	1	1	0	0	0	D_0	1
1	1	1	0	0	0	D_1	D_0	2
1	1	0	0	0	D_2	D_1	D_0	3
1	0	0	0	D_3	D_2	D_1	D_0	4
0	0	0	D_4	D_3	D_2	D_1	D_0	5

$\overline{\text{DTR}}$ (Data Terminal Ready) (D_7)

When this bit is 1, the $\overline{\text{DTR}}$ output is low (active). Conversely, when this bit is 0, $\overline{\text{DTR}}$ is high.

5.1.8 Control Register 6

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SYNC BYTE 1							

Figure 5.8 Control Register 6

Sync Byte 1 (D_0-D_7)

Sync byte 1 is used in the following modes:

Monosync:	8-bit sync character transmitted during the Idle Phase
Bisync:	Least significant (first) 8 bits of the 16-bit transmit and receive sync character
External Sync:	Sync character transmitted during the Idle Phase
SDLC:	Secondary address value matched to Secondary Address field of the SDLC frame when the MPSC^2 is in Address Search Mode

5.1.9 Control Register 7

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SYNC BYTE 2							

Figure 5.9 Control Register 7

Sync Byte 2 (D_0-D_7)

Sync Byte 2 is used in the following modes:

Monosync:	8-bit sync character matched by the Receiver
Bisync:	Most significant (second) 8 bits of the 16-bit transmit and receive sync characters
SDLC:	You must program the flag character, 01111110, into control register 7 for flag matching by the MPSC^2 receiver

5.1.10 Status Register 0

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Break/ Abort	Idle/CRC	$\overline{CTS}$	Sync Status	$\overline{DCD}$	Transmitter Buffer Empty	Interrupt Pending	Received Character Available

Figure 5.10 Status Register 0

Received Character Available (D_0)

When this bit is set, it indicates that one or more characters are available in the receiver buffer for the processor to read. Once all of the available characters have been read, the MPSC^2 resets this bit until a new character is received.

Interrupt Pending (D_1 - Channel A Only)

The interrupt pending bit is used with the interrupt vector register (status register 2) to make it easier to determine the MPSC^2's interrupt status, particularly in Non-vectored Interrupt mode where the processor must poll each device to determine the interrupt source. In this mode, interrupt pending is set when you read status register 2B, the $\overline{PRI}$ input is active (low) and the MPSC^2 is requesting interrupt service.

You need not analyze the status registers of both channels to determine if an interrupt is pending. If status affects vector is enabled and interrupt pending is set, the vector you read from SR2 contains valid condition information.

In Vectored Interrupt mode, interrupt pending is set during the interrupt acknowledge cycle (on the leading edge of the 2nd $\overline{INTA}$ pulse) when the MPSC^2 is the highest priority device requesting interrupt service ($\overline{PRI}$ is active). In either mode, if there are no other pending interrupt requests, interrupt pending is reset when the End of Interrupt command is issued.

Transmitter Buffer Empty (D_2)

This bit is set whenever the transmitter buffer is empty, except during the transmission of CRC (the MPSC^2 uses the buffer to facilitate this function). After a reset, the buffer is considered empty and transmit buffer empty is set.

External/Status Flags

The following status bits reflect the state of the various conditions that cause an external/status interrupt. The MPSC^2 latches all external/status bits whenever a change occurs that would cause an external/status interrupt (regardless of whether this interrupt is enabled). This allows you to capture transient status changes on these lines with relaxed software timing requirements (see Appendix A for detailed timing specifications).

When you operate the MPSC^2 in interrupt-driven mode for external/status interrupts, you should read status register 0 when this interrupt occurs and issue a Reset External/Status Interrupt command to reenable the interrupt and the latches. To poll these bits without interrupts, you can issue the Reset External/Status Interrupt command to first update the status to reflect the current values.

$\overline{DCD}$ (D_3)

This bit reflects the inverted state of the $\overline{DCD}$ input. When $\overline{DCD}$ is low, the $\overline{DCD}$ status bit is high. Any transition on this bit causes an External/Status Interrupt request.

Sync Status (D_4)

The meaning of this bit depends on the operating mode of the MPSC^2.

Asynchronous mode: sync status reflects the inverted state of the $\overline{SYNC}$ input. When $\overline{SYNC}$ is low, sync status is high. Any transition on this bit causes an External/Status Interrupt request.

External Synchronization mode: sync status operates in the same manner as asynchronous mode. The MPSC^2's receiver synchronization logic is also tied to the sync status bit in external synchronization mode and a low-to-high transition ($\overline{SYNC}$ input going low) informs the receiver that synchronization has been achieved and character assembly begins (see Appendix A for detailed timing information).

A low-to-high transition on the $\overline{SYNC}$ input indicates that synchronization has been lost and is reflected both in sync status becoming zero and the generation of an External/Status interrupt. The receiver remains in Receive Data Phase until you set the Enter Hunt Phase bit in Control Register 3.

Monosync, Bisync, SDLC modes: In these modes, sync status indicates whether the MPSC^2 receiver is in the Sync Hunt or Receive Data Phase of operation. A 0 indicates that the MPSC^2 is in the Receive Data Phase and a one indicates that the MPSC^2 is in the Sync Hunt Phase, as after a reset or setting the Enter Sync Hunt Phase bit. As in the other modes, a transition on this bit causes an External/Status interrupt to be issued. You should note that entering Sync Hunt Phase after either a reset or when programmed causes an External/Status Interrupt request which you may clear immediately with a Reset External/Status Interrupt command.

$\overline{CTS}$ (D_5)

This bit reflects the inverted state of the $\overline{CTS}$ input. When $\overline{CTS}$ is low, the $\overline{CTS}$ status bit is high. Any transition on this bit causes an External/Status Interrupt request.

Idle/CRC (D_6)

This bit indicates the state of the Idle/CRC latch used in synchronous and SDLC modes. After reset this bit is 1, indicating that when the transmitter is completely empty, the MPSC^2 enters Idle Phase and automatically transmits sync or flag characters.

A zero indicates that the latch has been reset by the Reset Idle/CRC Latch command. When the transmitter is completely empty, the MPSC^2 sends the 16-bit CRC character and sets the latch again. An External/Status interrupt is issued when the latch is set, indicating that CRC is being sent. No interrupt is issued when the latch is reset.

Break/Abort (D_7)

In asynchronous mode, this bit indicates the detection of a break sequence (a null character plus framing error, that occurs when the RxD input is held low (spacing) for more than 1 character time). Break/Abort is reset when RxD returns high (marking).

In SDLC mode, Break/Abort indicates the detection of an abort sequence when 7 or more ones are received in sequence. It is reset when a zero is received.

Any transition of the Break/Abort bit causes an External/Status Interrupt.

5.1.11 Status Register 1

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
End of SDLC Frame	CRC Framing Error	Overrun Error	Parity Error	SDLC Residue Code			All Sent

Figure 5.11 Status Register 1

All Sent (D_0)

In asynchronous mode, this bit is set when the transmitter is empty and reset when a character is present in the transmitter buffer or shift register. This feature simplifies your modem control software routines. In synchronous and SDLC modes, this bit is always set to 1.

SDLC Residue Code (D_1-D_3)

Since the data portion of an SDLC message can consist of any number of bits and not necessarily an integral number of characters, the MPSC^2 features special logic to determine and report when the End of Frame flag has been received, the boundary between the data field, and the CRC character in the last few data characters that were just read.

When the end of frame condition is indicated, that is, status register 1 D_7 = 1 and Special Receive Condition interrupt (if enabled), the last bits of the CRC character are in the receiver buffer. The residue code for the frame is valid in the status register 1 byte associated with that data character (remember SR1 tracks the received data in its own buffer).

The meaning of the residue code depends upon the number of bits/characters specified for the receiver. The previous character refers to the last character read before the End of Frame, etc.

Table 5.12 Residue Codes

8 Bits/Character

D_3	D_2	D_1	Previous Character	2nd Previous Character
1	0	0	C C C C C C C C	C C C C C D D D
0	1	0	C C C C C C C C	C C C C D D D D
1	1	0	C C C C C C C C	C C C D D D D D
0	0	1	C C C C C C C C	C C D D D D D D
1	0	1	C C C C C C C C	C D D D D D D D
0	1	1	C C C C C C C C	D D D D D D D D (no residue)
1	1	1	C C C C C C C D	D D D D D D D D
0	0	0	C C C C C C D D	D D D D D D D D

7 Bits/Character

D_3	D_2	D_1	Previous Character	2nd Previous Character
1	0	0	C C C C C C C	C C C C C D D
0	1	0	C C C C C C C	C C C C D D D
1	1	0	C C C C C C C	C C C D D D D
0	0	1	C C C C C C C	C C D D D D D
1	0	1	C C C C C C C	C D D D D D D
0	1	1	C C C C C C C	D D D D D D D (no residue)
0	0	0	C C C C C C D	D D D D D D D

6 Bits/Character

D_3	D_2	D_1	Previous Character	2nd Previous Character
1	0	0	C C C C C C	C C C C C D
0	1	0	C C C C C C	C C C C D D
1	1	0	C C C C C C	C C C D D D
0	0	1	C C C C C C	C C D D D D
1	0	1	C C C C C C	C D D D D D
0	0	0	C C C C C C	D D D D D D (no residue)

5 Bits/Character

D_3	D_2	D_1	2nd Previous Character	3rd Previous Character
1	0	0	C C C C C	D D D D D (no residue)
0	1	0	C C C C D	D D D D D
1	1	0	C C C D D	D D D D D
0	0	1	C C D D D	D D D D D
0	0	0	C D D D D	D D D D D

Special Receive Condition Flags

The status bits described below (Parity error (if Parity is a Special Receive condition is enabled), Receiver Overrun Error, CRC/Framing Error, and End of SDLC Frame), all represent Special Receive conditions.

When any of these conditions occurs and interrupts are enabled, the MPSC[2] issues an interrupt request. In addition, if you enabled Condition Affects Vector mode, the vector generated (and the contents of SR2B for non-vectored interrupts) is different from that of a Received Character Available condition. Thus, you need not analyze SR1 with each character to determine that an error has occurred.

As a further convenience, the Parity Error and Receiver Overrun Error flags are latched, that is, once one of these errors occurs, the flag remains set for all subsequent characters until reset by the Error Reset command. With this facility, you need only read SR1 at the end of a message to determine if either of these errors occurred anywhere in the message. The other flags are not latched and follow each character available in the receiver buffer.

Parity Error (D_4)

This bit is set and latched when parity is enabled and the received parity bit does not match the sense (odd or even) calculated from the data bits.

Receiver Overrun Error (D_5)

This error occurs and is latched when the receiver buffer already contains three characters and a fourth character is completely received, overwriting the last character in the buffer.

CRC/Framing Error (D_6)

In asynchronous mode, a framing error is flagged (but not latched) when no stop bit is detected at the end of a character (i.e. RxD is low 1 bit time after the center of the last data or parity bit). When this condition occurs, the MPSC^2 waits an additional 1/2 bit time before sampling again so that the framing error is not interpreted as a new start bit.

In synchronous and SDLC modes, this bit indicates the result of the comparison between the current CRC result and the appropriate check value and is usually set to 1 since a message rarely indicates a correct CRC result until correctly completed with the CRC check character. Note that a CRC error does not result in a Special Receive Condition interrupt.

End of SDLC Frame (D_7)

This flag is used only in SDLC mode to indicate that the End of Frame flag has been received and that the CRC error flag and residue code is valid. You can reset this flag at any time by issuing an Error Reset command. The MPSC^2 also automatically resets this bit for you on the first character of the next message frame.

5.1.12 Status Register 2

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Interrupt Vector							

Figure 5.12 Status Register 2

Interrupt Vector (D_0-D_7 - Channel B Only)

Reading status register 2B returns the interrupt vector that you programmed into control register 2B. If Condition Affects Vector mode is enabled, the value of the vector is modified as follows:

Table 5.13 Condition Affects Vector Modifications

8085 Modes 8086 Modes	D_4 D_2	D_3 D_1	D_2 D_0	CONDITION
	1	1	1	No Interrupt Pending
	0	0	0	Channel B Transmitter Buffer Empty
	0	0	1	Channel B External/Status Change
	0	1	0	Channel B Received Character Available
	0	1	1	Channel B Special Receive Condition
	1	0	0	Channel A Transmitter Buffer Empty
	1	0	1	Channel A External/Status Change
	1	1	0	Channel A Received Character Available
	1	1	1	Channel A Special Receive Condition

As you can see, code 111 can mean either channel A Special Receive condition or no interrupt pending. You can easily distinguish between the two by examining the Interrupt Pending bit (D_1) of status register 0, channel A. Remember, in Non-vectored Interrupt mode you must read the vector register first for Interrupt Pending to be valid.

5.2 MPSC^2 Programming Examples

ASYNC.01

```
********** Asynchronous Mode ***************

Init:
        ISSUE Channel Reset Command (CR0)
        SET Bus Interface Options (CR2A)
        SET Interrupt Vector (CR2B)- if used
        SET Operating Mode (CR4):
                Asynchronous Mode, Parity Select, # of Stop Bits,
                Clock Rate.
        SET Receive Enable, Auto Enables, Receive Character Length (CR2)
        SET Transmit Enable, Modem Controls, Transmit Char, Length (CR5)
        ISSUE Reset External/Status Interrupt Command
        SET Transmit Interrupt Enable, Receive Interrupt on Every
                Character, External Interrupt Enable, Wait Mode
                Disable.
**** End Of Initialization ******
```

```
Send:
        ISSUE First Byte To MPSC
        RETURN To Main Program OR Halt

Interrupt:
   CASE Interrupt Type DO:

Character_Received:
        READ Character from MPSC
        PROCESS Character
        ISSUE End Of Interrupt Command
        RETURN From Interrupt

Special Recieve Condition:
        READ SR1
        ISSUE Error Reset Command
        CALL Special Error Routine
        ISSUE End Of Interrupt Command
        RETURN From Interrupt

Transmitter Buffer Empty:
        IF Last Character Transferred was End of Message
           THEN ISSUE Reset Transmit Interrupt/DMA Pending Command
        ELSE
           Transfer Next Character to MPSC
           ISSUE End Of Interrupt Command
           RETURN From Interrupt

External/Status Change:
        READ SR1
        CALL Special Condition Routine
        ISSUE End Of Interrupt Command
        RETURN From Interrupt

**** END CASE ****

Terminate Transmit:
        RESET Transmit Enable, RTS (CR5)
        RETURN

Terminate Receive:
        RESET Receive Enable (CR1)
        RESET DTR (CR5)

ASYNC.01

        RETURN

END
```

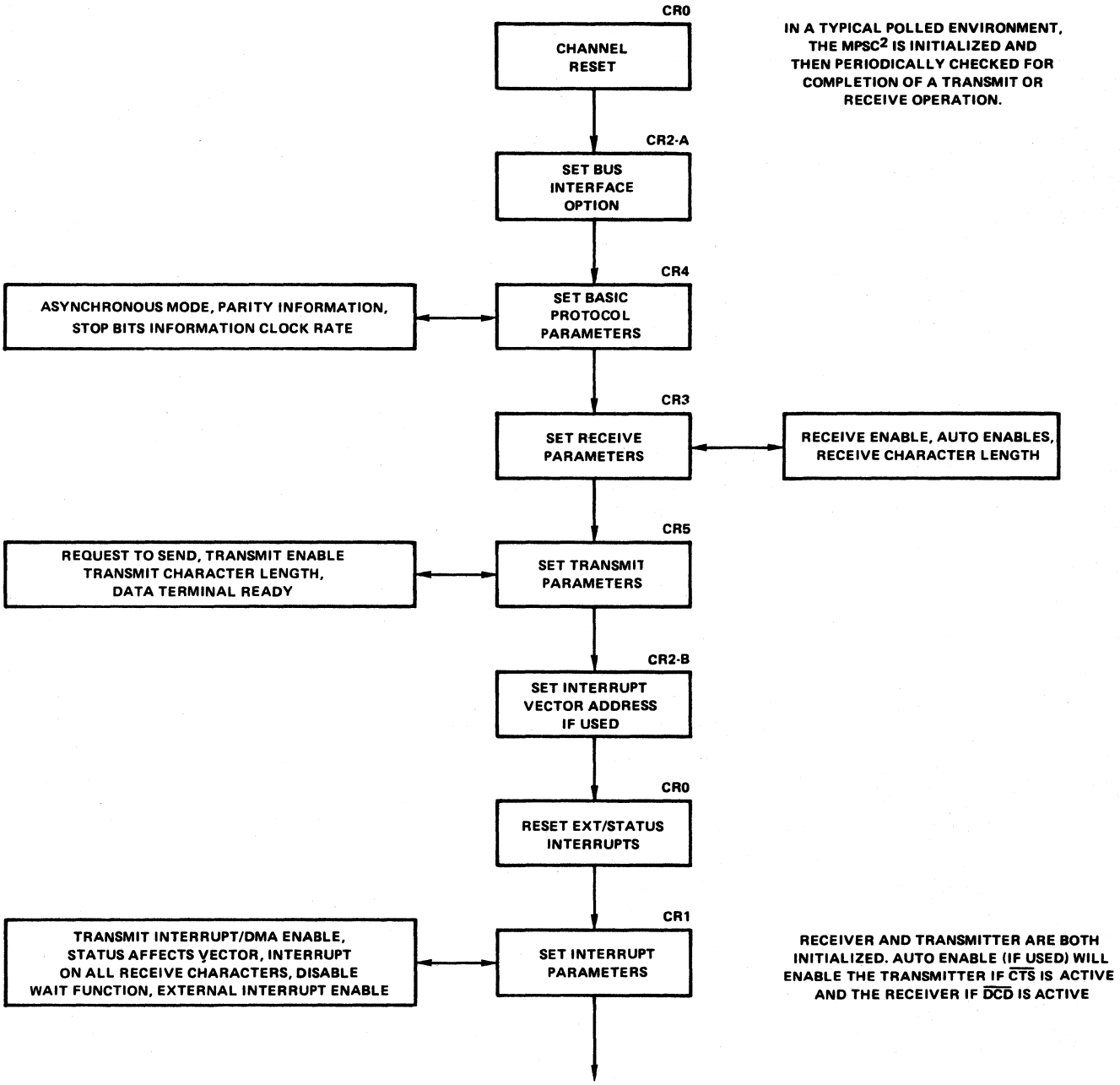
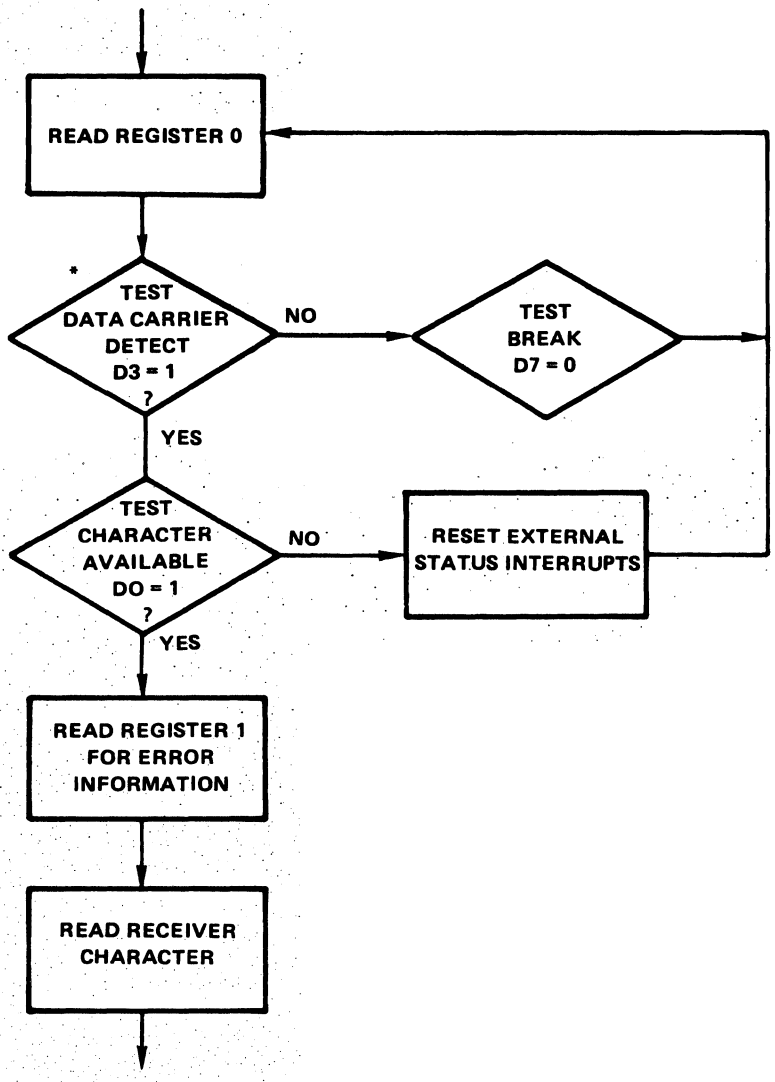
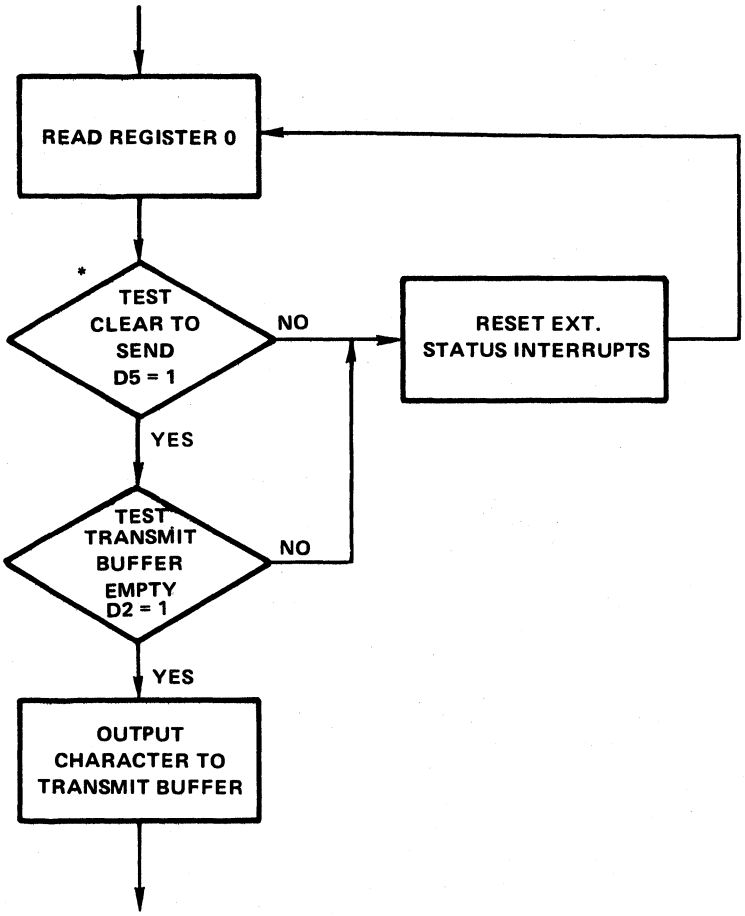


Figure 5.13 Asynchronous Initialization for Polled Transmit and Receive



*NOTE: IF AUTO ENABLE WAS SET (D5 = 1 IN CONTROL REGISTER 3) THIS STEP MAY BE OMITTED.

Figure 5.14 Asynchronous Receive



*IF AUTO ENABLE WAS SET (D5 = 1 IN CONTROL REGISTER 3), THIS STEP MAY BE OMITTED

Figure 5.15 Asynchronous Transmit

SYNC.PRG

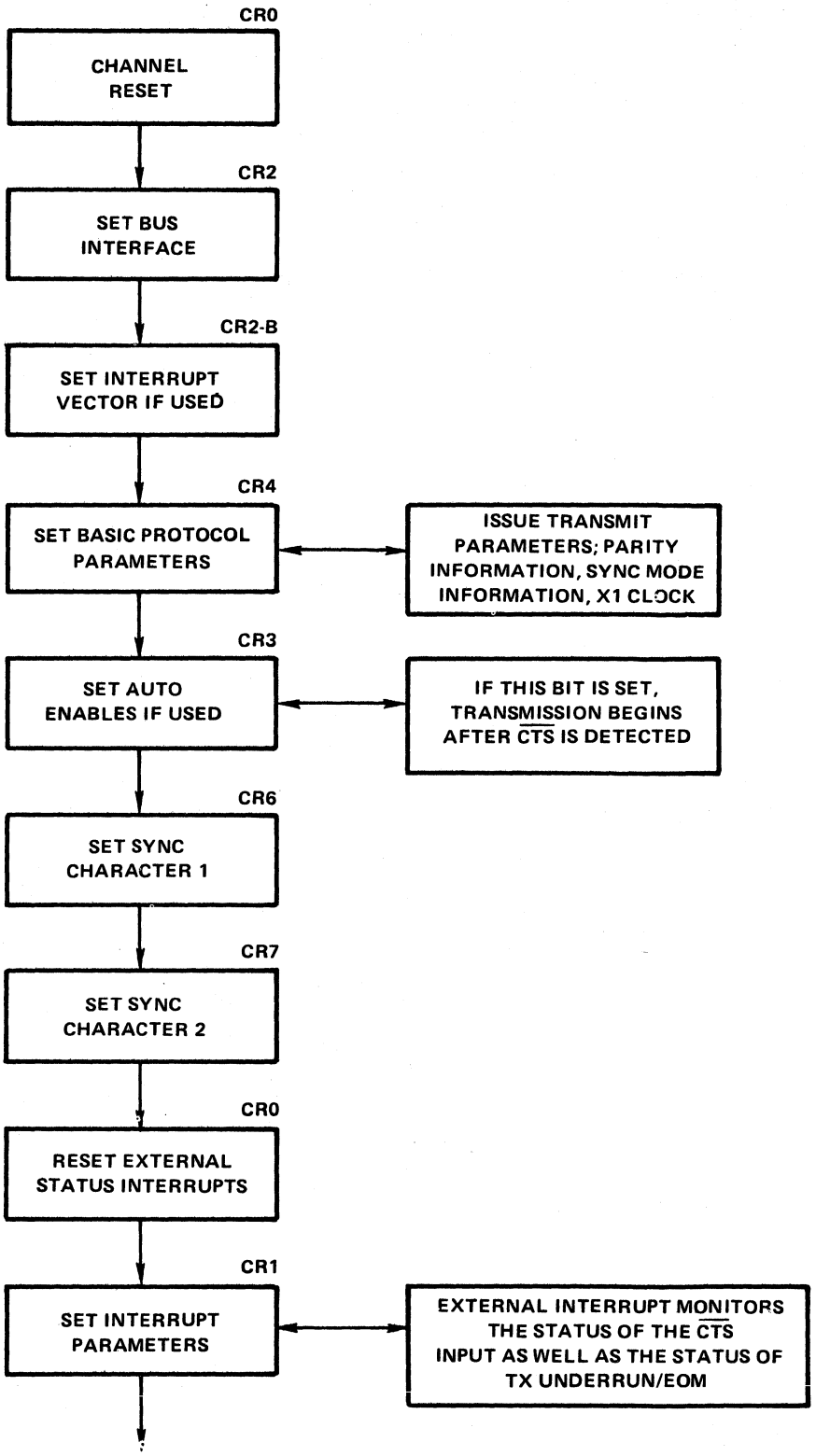
```
************ SYNCHRONOUS OPERATION EXAMPLE ****************

**** This example uses the Block Transfer Mode ****

Init:
        ISSUE Channel Reset Command
        SET Interface Option (CR2A)
        SET Interrupt Vector (CR2B)
        SET Parity Mode, Sync Mode, 1x Clock (CR4)
        SET Sync Character 1 (CR6)
        SET Sync Character 2 (CR7)
        RETURN

Initiate Transmit:
        ISSUE Reset External/Status Interrupt Command
        SET External Interrupt Enable, Transmit Interrupt Enable
                Wait Enable, Wait on Transmit (CR1)
        SET Transmit Enable, # of Bits/Character, RTS,
                CRC Polynomial Select.

**** Transmitter is now enabled and will automatically begin
        sending Sync characters ****

        WAIT Several Character Times (a good idea to help system
                gain synchronization)
Next Message:
        ISSUE Reset Transmit CRC Command

Send Character:
        GET Character
        IF Character Is To Be Included In CRC
        THEN
           SET CRC Generator On (CR5)
        ELSE
           SET CRC Generator Off (CR5)
        ENDIF

        WRITE Character To MPSC  (Processor will "Wait" until
                Transmitter buffer is empty)
        IF Character Was Not The Last
        THEN
           GOTO Send Character  (do next character)
        ELSE
           SET CRC Generator On (CR5)
           ISSUE Reset Idle/CRC Latch Command
           WAIT For External/Status Interrupt Indicating CRC
                Being Sent
           IF Next Message Is Ready To Be Transmitted
           THEN
              GOTO Next Message  (Next message will be sent immediately
                   following CRC)
```

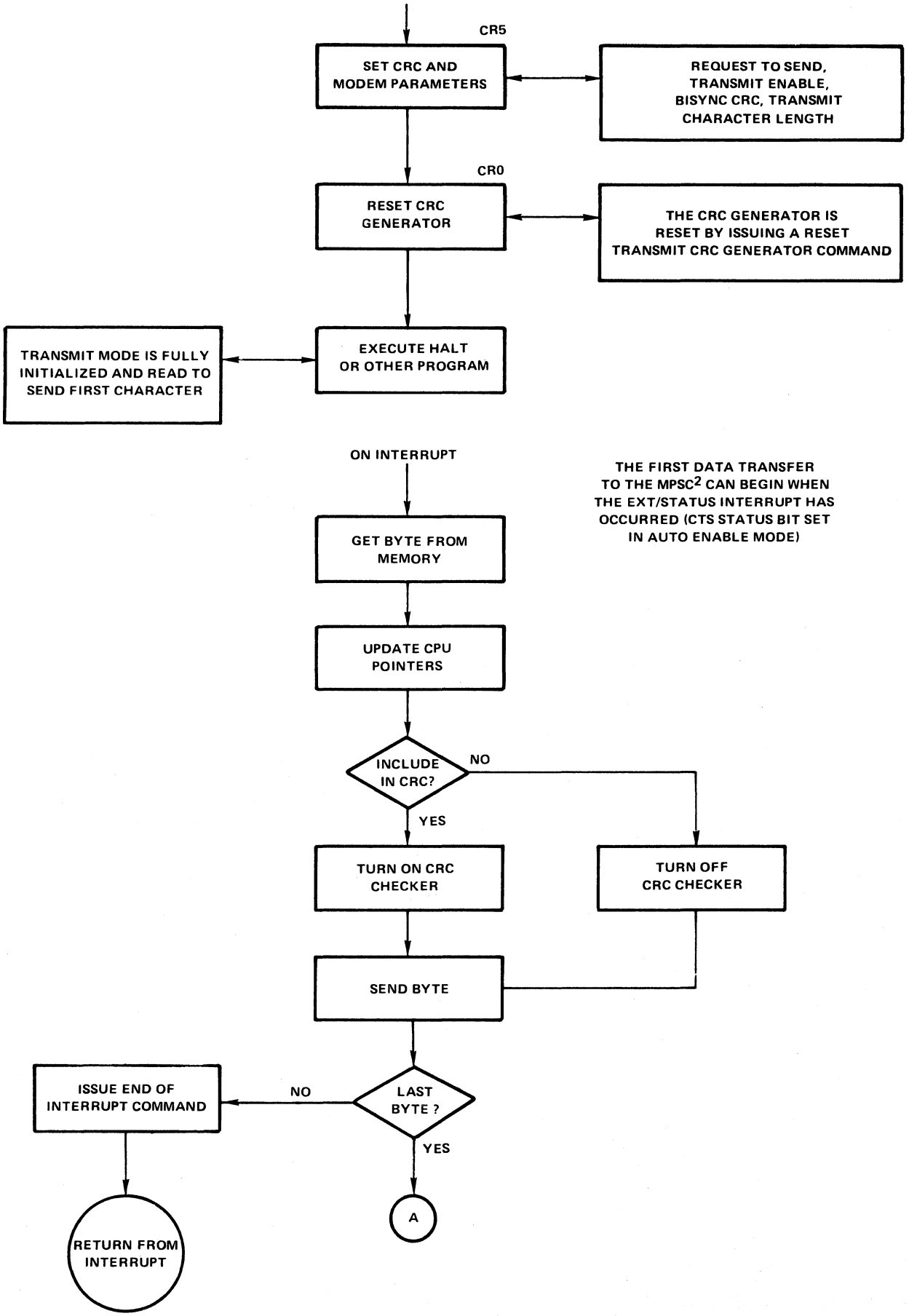
```
            ELSE
               WAIT For Transmit Buffer Empty Interrupt indicating
                 Trailing Sync Being Sent
               SET Transmitter Enable Off, RTS Off (CR5)
            ENDIF
         ENDIF

**** End of Transmit Routine ****

SYNC.PRG

**** Receive Routine ****

Receive Message:
         SET External/Status Interrupt Enable, Receive Interrupt
                 On First Character Mode, Wait Enabled, Wait on
                 Receive (CR1)
         SET Receiver Enable On, Sync Character Load Inhibit,
                 # of Bits/Character (CR1)
         SET DTR On (CR5)
         ISSUE Reset External Status Interrupt Command
         ISSUE Enable Interrupt On Next Received Character Command
         ISSUE Error Reset Command

**** Receiver is now enabled and in the Hunt Phase ****

         WAIT For External/Status Interrupt (indicating synchronization
                 has been acheived)
         Issue Error Reset Command
         WAIT For Received Character Available Interrupt (first non-sync
                 character is now available)
         ISSUE Reset CRC Checker Command
         SET Sync Character Load Inhibit Off

Get Character:
         GET Character from MPSC (processor will "Wait" until at least
                 1 character is available)

         IF Character Is To Be Included In CRC Calculation
         THEN
            Turn CRC Checker On (CR3)
         ELSE
            SET CRC Checker Off (CR3)
         ENDIF

         IF Character Is Part of Message Data
         THEN
            SAVE Character In Memory
         ENDIF

         IF Character Was NOT End Of Message
         THEN
            GOTO READ Character
         ENDIF

*** End Of Message ***
```

```
        SET CRC Checker On
        READ 2 CRC Characters
        READ 2 Character (these characters may be part of the next
               message but must be read before CRC will be valid)
        READ SR1 (this must be done immediately so that next
               character status will not overwrite)
        IF Parity OR Overrun OR CRC = Error
        THEN
           GOTO Error Processor
        ENDIF

        IF More Messages Are To Be Recieved
        THEN
           GOTO Get Next Message
```

SYNC.PRG

```
        ELSE
           SET DTR Off
           SET Receive Enable Off
        SET External/Status Interrupts Off, Reciever Interrupt
        Mode Disabled (CR1)
        RETURN

        END
           RETURN
```


CR0
CHANNEL RESET
CR2
SET BUS INTERFACE
CR2-B
SET INTERRUPT VECTOR IF USED
CR4
SET BASIC PROTOCOL PARAMETERS
ISSUE TRANSMIT PARAMETERS; PARITY INFORMATION, SYNC MODE INFORMATION, X1 CLOCK
CR3
SET AUTO ENABLES IF USED
IF THIS BIT IS SET, TRANSMISSION BEGINS AFTER CTS IS DETECTED
CR6
SET SYNC CHARACTER 1
CR7
SET SYNC CHARACTER 2
CR0
RESET EXTERNAL STATUS INTERRUPTS
CR1
SET INTERRUPT PARAMETERS
EXTERNAL INTERRUPT MONITORS THE STATUS OF THE CTS INPUT AS WELL AS THE STATUS OF TX UNDERRUN/EOM


CR5
SET CRC AND MODEM PARAMETERS
REQUEST TO SEND, TRANSMIT ENABLE, BISYNC CRC, TRANSMIT CHARACTER LENGTH
CR0
RESET CRC GENERATOR
THE CRC GENERATOR IS RESET BY ISSUING A RESET TRANSMIT CRC GENERATOR COMMAND
EXECUTE HALT OR OTHER PROGRAM
TRANSMIT MODE IS FULLY INITIALIZED AND READ TO SEND FIRST CHARACTER
ON INTERRUPT
THE FIRST DATA TRANSFER TO THE MPSC2 CAN BEGIN WHEN THE EXT/STATUS INTERRUPT HAS OCCURRED (CTS STATUS BIT SET IN AUTO ENABLE MODE)
GET BYTE FROM MEMORY
UPDATE CPU POINTERS
INCLUDE IN CRC?
NO
YES
TURN ON CRC CHECKER
TURN OFF CRC CHECKER
SEND BYTE
ISSUE END OF INTERRUPT COMMAND
NO
LAST BYTE ?
YES
A
RETURN FROM INTERRUPT

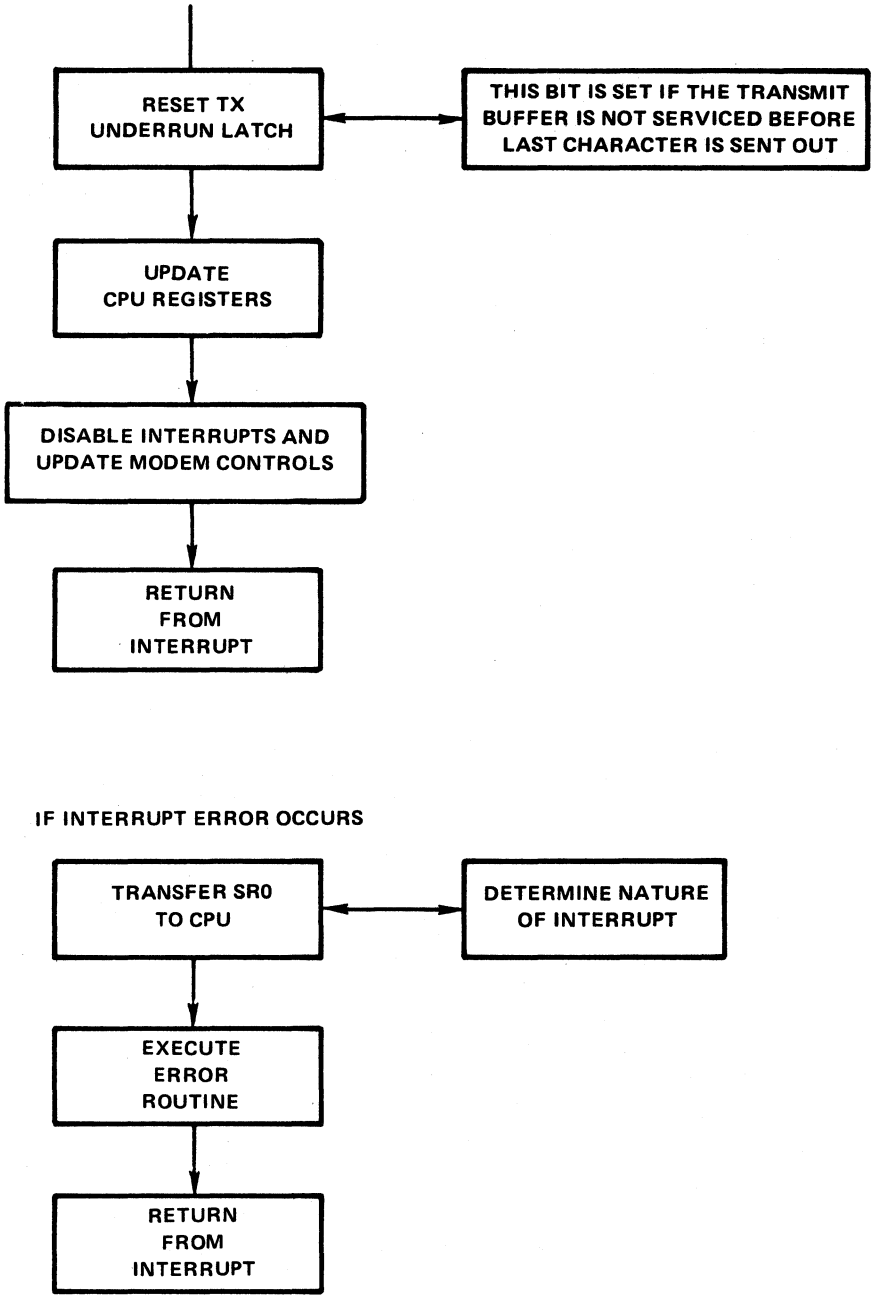
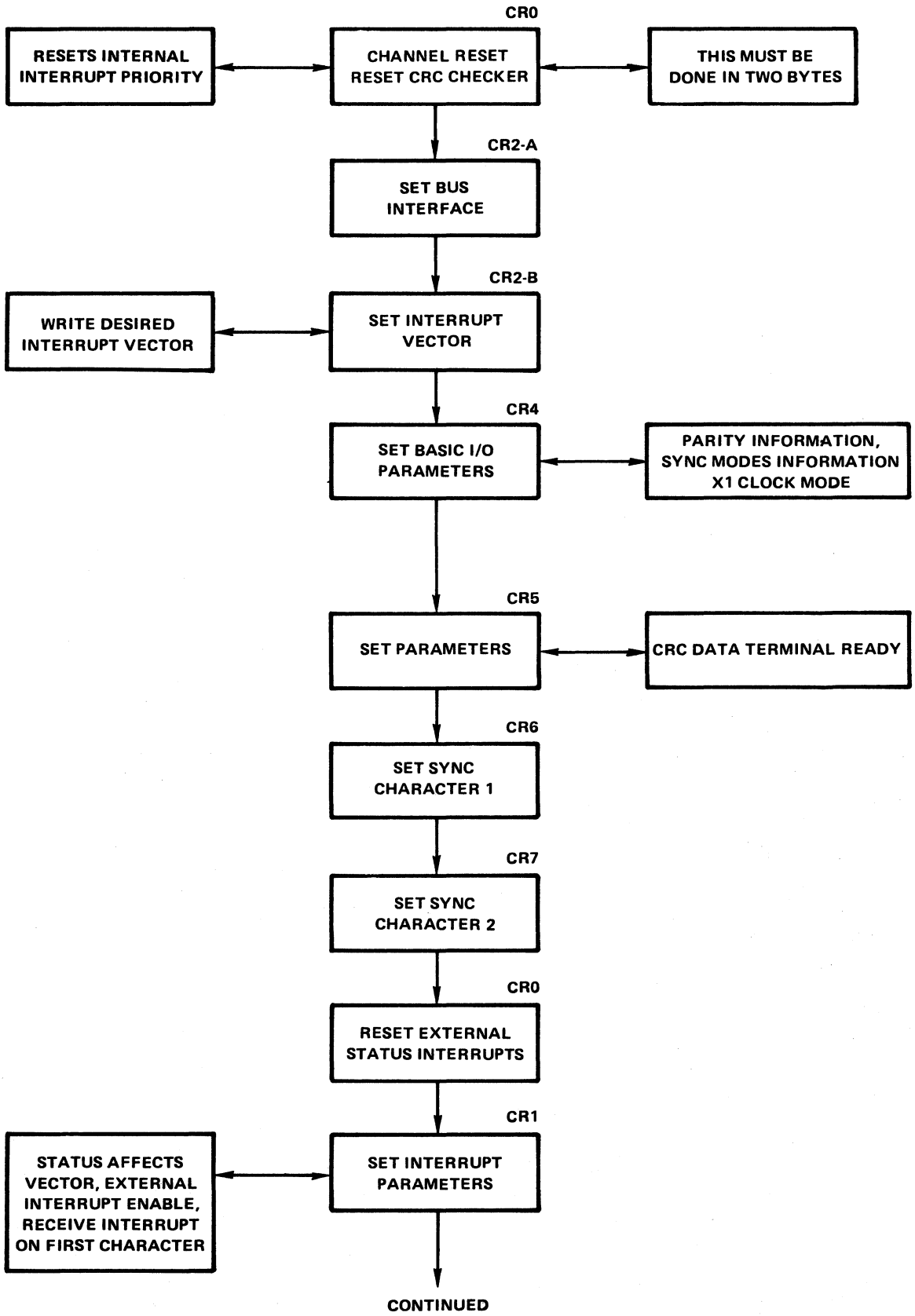
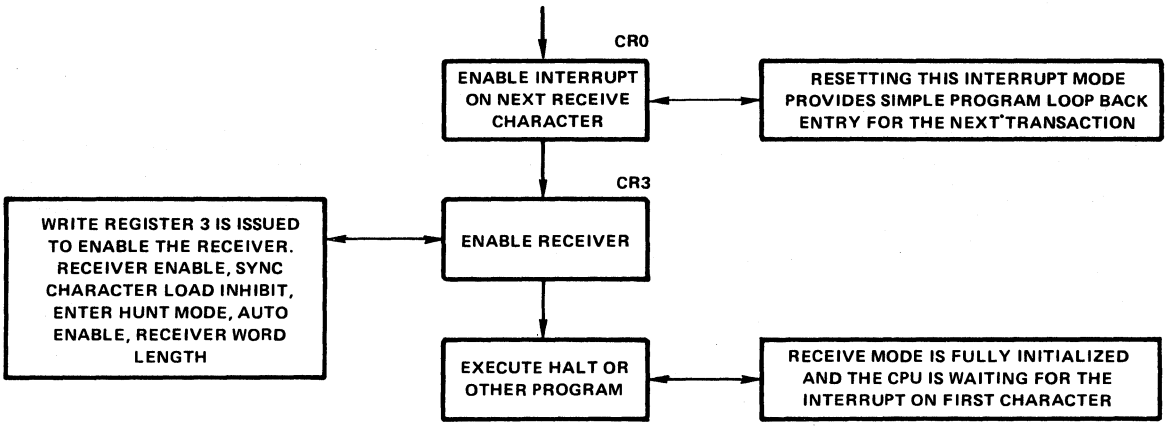


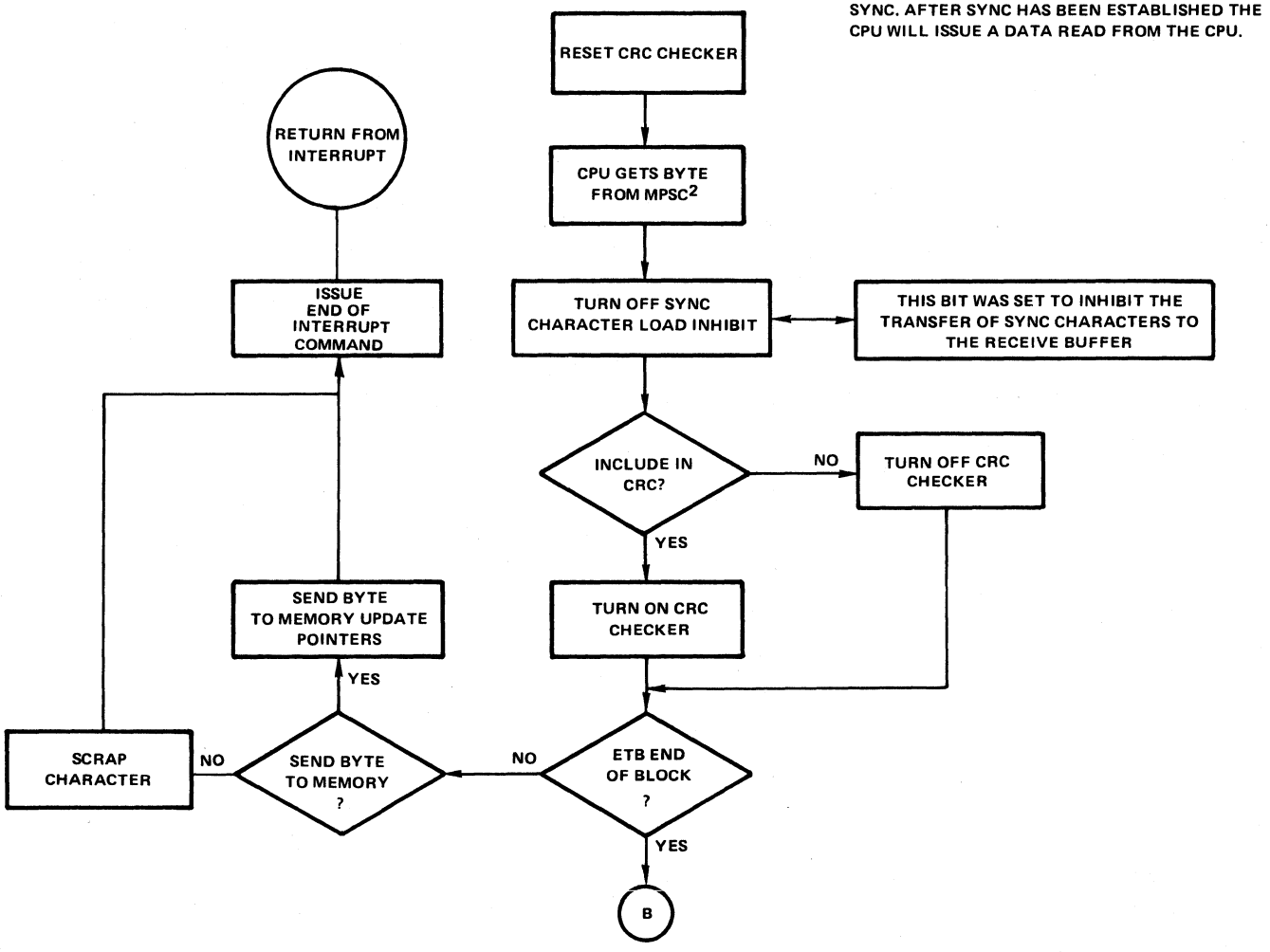
Figure 5.16 Bisync Initialization Transmit


CR0
RESETS INTERNAL INTERRUPT PRIORITY
CHANNEL RESET RESET CRC CHECKER
THIS MUST BE DONE IN TWO BYTES
CR2-A
SET BUS INTERFACE
CR2-B
WRITE DESIRED INTERRUPT VECTOR
SET INTERRUPT VECTOR
CR4
SET BASIC I/O PARAMETERS
PARITY INFORMATION, SYNC MODES INFORMATION X1 CLOCK MODE
CR5
SET PARAMETERS
CRC DATA TERMINAL READY
CR6
SET SYNC CHARACTER 1
CR7
SET SYNC CHARACTER 2
CR0
RESET EXTERNAL STATUS INTERRUPTS
CR1
STATUS AFFECTS VECTOR, EXTERNAL INTERRUPT ENABLE, RECEIVE INTERRUPT ON FIRST CHARACTER
SET INTERRUPT PARAMETERS
CONTINUED



BISYNC TRANSMIT
WHEN INTERRUPT ON FIRST
CHARACTER OCCURS.

DURING THE HUNT MODE, THE MPSC^2 DETECTS TWO CONTIGUOUS CHARACTERS TO ESTABLISH SYNC. AFTER SYNC HAS BEEN ESTABLISHED THE CPU WILL ISSUE A DATA READ FROM THE CPU.



RECEIVE CONTINUED

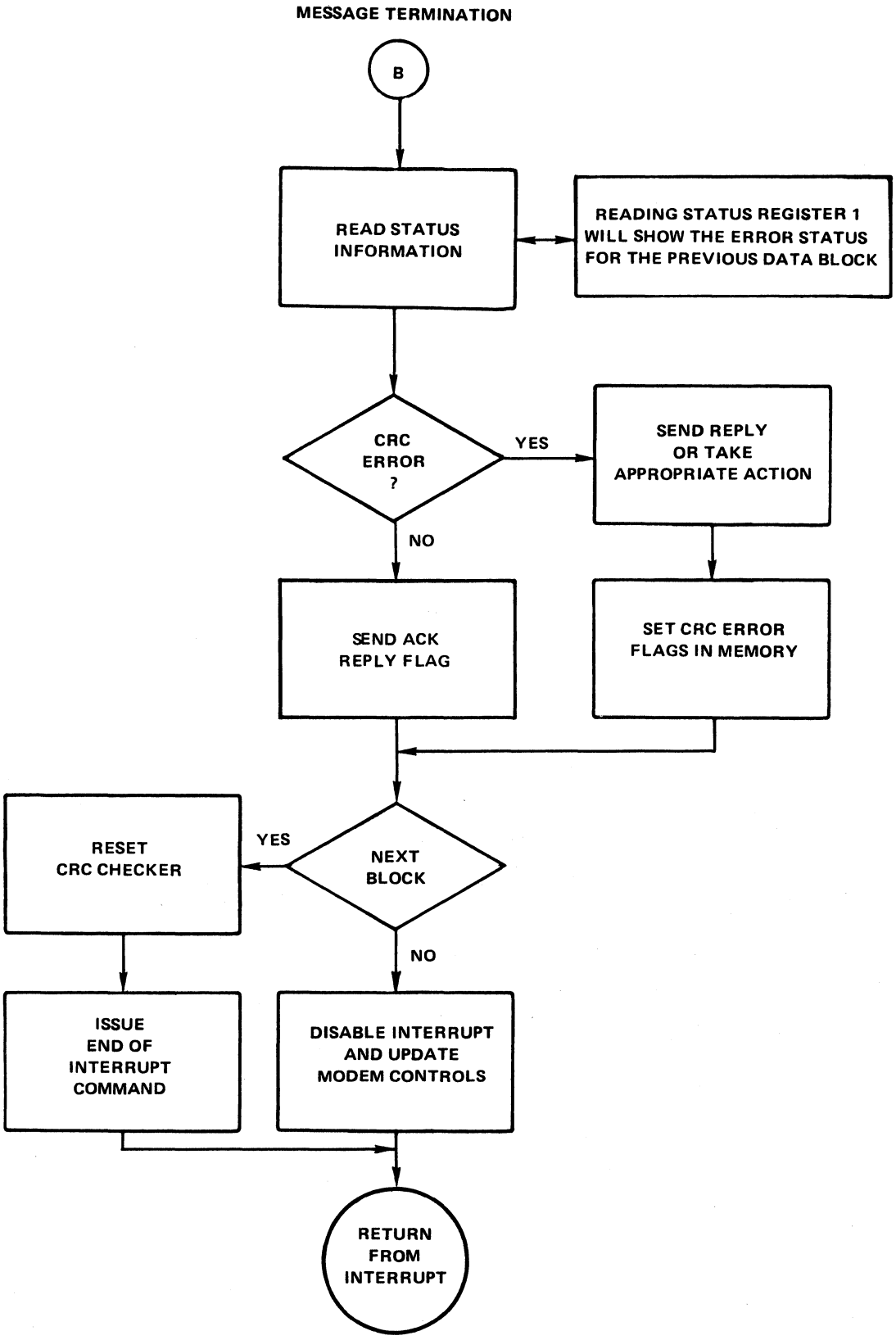
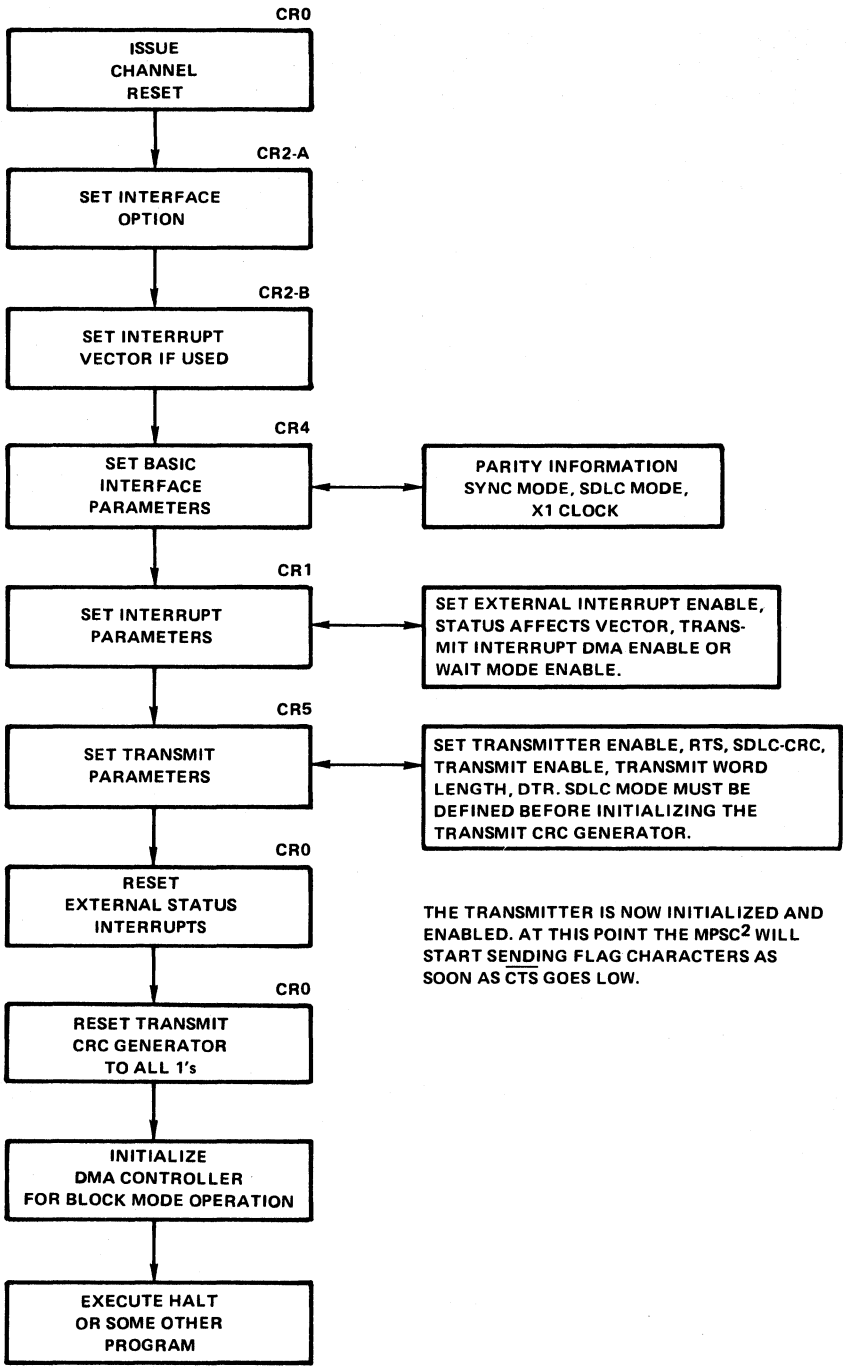


Figure 5.17 Bisync Initialization Receive

```
******* ***** SDLC OPERATION EXAMPLE **********************

**** This example uses DMA Transfer Mode ****

Initialize:
ISSUE Channel Reset Command
SET Interface Option (CR2A)
SET Interrupt Vector (CR2B)
SET SDLC Mode, 1x Clock (CR4)
SET SDLC Flag (CR7)= 01111110
SET SDLC Secondary Address (CR6)
RETURN

Initiate Transmit:
ISSUE Reset External Status Interrupt Command
SET External Interrupt Enable, Transmit Interrupt/DMA
Enable (CR1)
SET Transmit Enable, RTS, CRC-CCITT Polynomial (CR5)

**** The Transmitter is now enabled and will automatically begin
sending Flag characters ****

Send Message:
SET DMA Controller to Beggining Of Message, # of Characters
in Message.
ISSUE Reset Transmit CRC Generator Command
SET 8 Bits/Character (CR5)
WRITE Address byte to MPSC
SET # of Bits/Character (CR5)
ISSUE Reset EOM/CRC Latch Command

**** The MPSC will now transmit the message until the DMA Controller
completes the required number of transfers ****

WAIT for External/Status Change Interrupt (signifies CRC
being sent)
IF Next Message Ready to be Transmitted
THEN
   GOTO Send Message (since MPSC will automatically issue a
DMA request when ready, set DMA controller to address
byte preceeding message and skip the write)
ELSE
   ISSUE RESET External/Status Interrupt Command
   ISSUE RESET Transmit Interrupt/DMA Pending Command
   RETURN

**** End of Transmit Routine ****

Recieve Message:
SET External/Status Interrupt Enable, Receive Interrupt
on First Character (CR1)
SET Reciever Enable On, 8 Bits/Character, Receive CRC On,
Address Search Mode On (CR3)
SET DTR On, CRC-CCITT (CR5)
ISSUE Reset External/Status Interrupt Command
ISSUE Enable Interrupt On Next Character Command
```

```
**** Receiver is now enabled and in the Hunt Phase ****

WAIT for External/Status Interrupt (indicating that a Flag
character has been recieved)
ISSUE Reset External/Status Interrupt Command
RETURN From Interrupt


**** Reciever is now in the Address Search Phase ****

Next Message:
WAIT for Character Received Interrupt (indicating that an
address match or global address has occurred)
GET Address Character (for later processing)
SET DMA Controller
SET # of Bits/Character (CR3)

**** Receiver is now in the Data Phase and will transfer all
succeeding characters until the End of Frame Flag ****

WAIT for Special Receive Condition Interrupt (indicating
flag received)
READ SR1 to Obtain CRC Status and Residue Code
SET DMA Controller Off
IF More Messages Are To Be Received
THEN
   GOTO Next Message
ELSE
   SET DTR Off
   SET Receive Enable Off
   RETURN
ENDIF
```



THE EXTERNAL INTERRUPT MODE MONITORS THE STATUS OF $\overline{CTS}$ AND $\overline{DCD}$, AS WELL AS THE STATUS OF TX UNDERRUN/EOM LATCH. A TRANSMIT INTERRUPT OCCURS WHEN THE TRANSMIT BUFFER BECOMES EMPTY. THE EXTERNAL $\overline{WAIT}$ PIN CAN BE USED FOR BLOCK MODE TRANSFERS OR THE DRQ PINS (WHICH ARE EXTERNAL) CAN BE USED IN DMA OPERATION AS WELL.

THE TRANSMITTER IS NOW INITIALIZED AND ENABLED. AT THIS POINT THE MPSC^2 WILL START SENDING FLAG CHARACTERS AS SOON AS $\overline{CTS}$ GOES LOW.

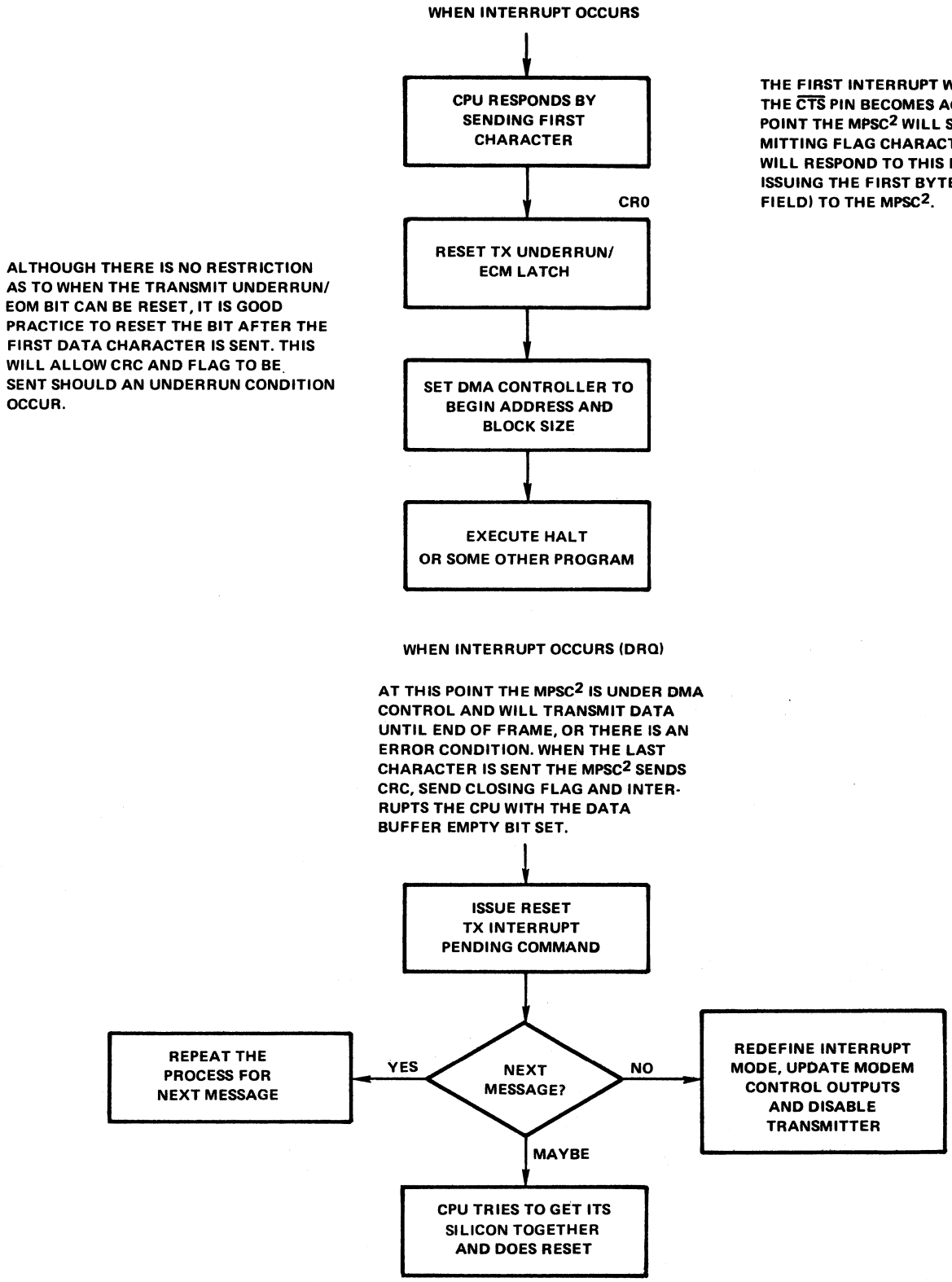
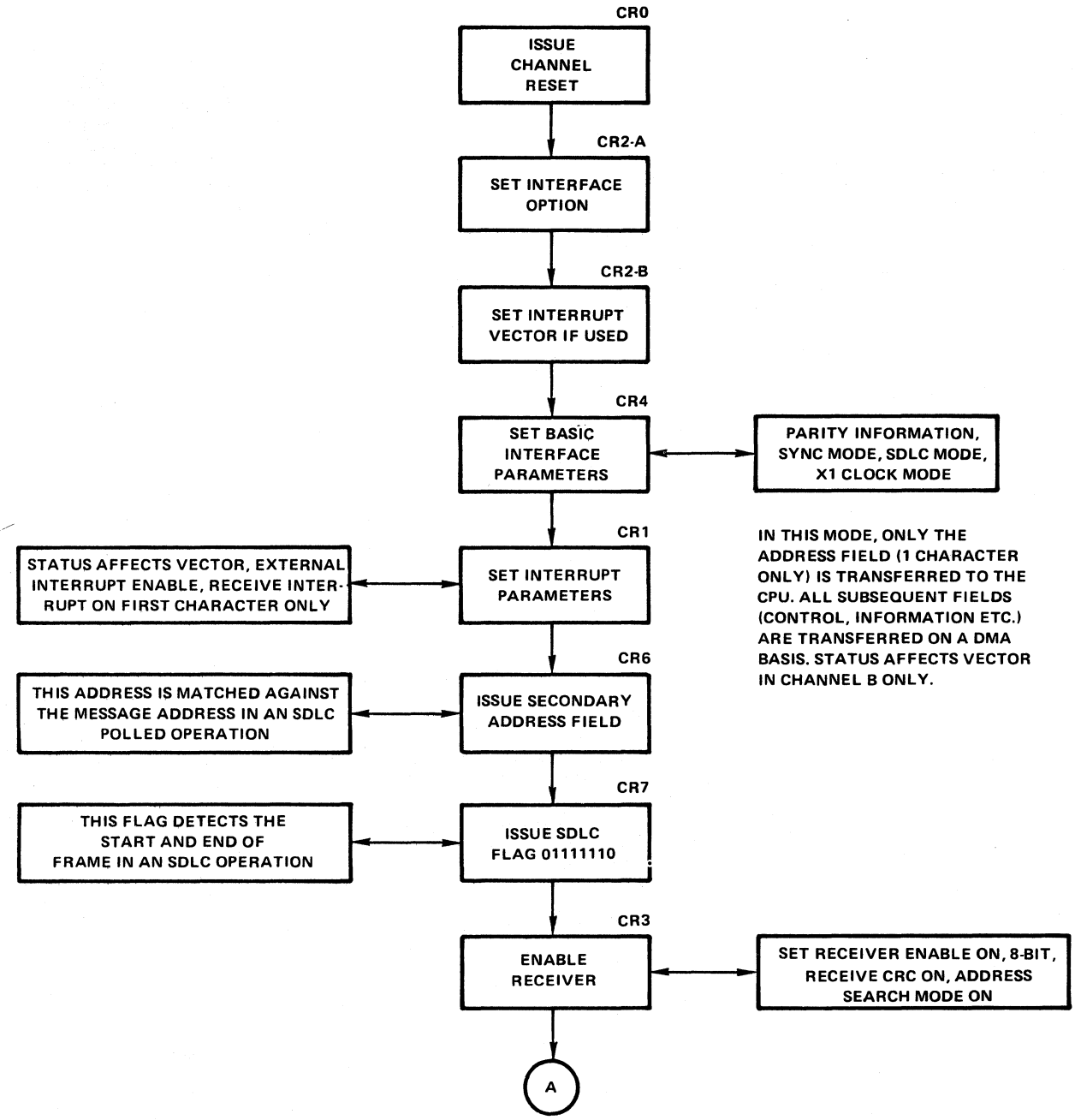
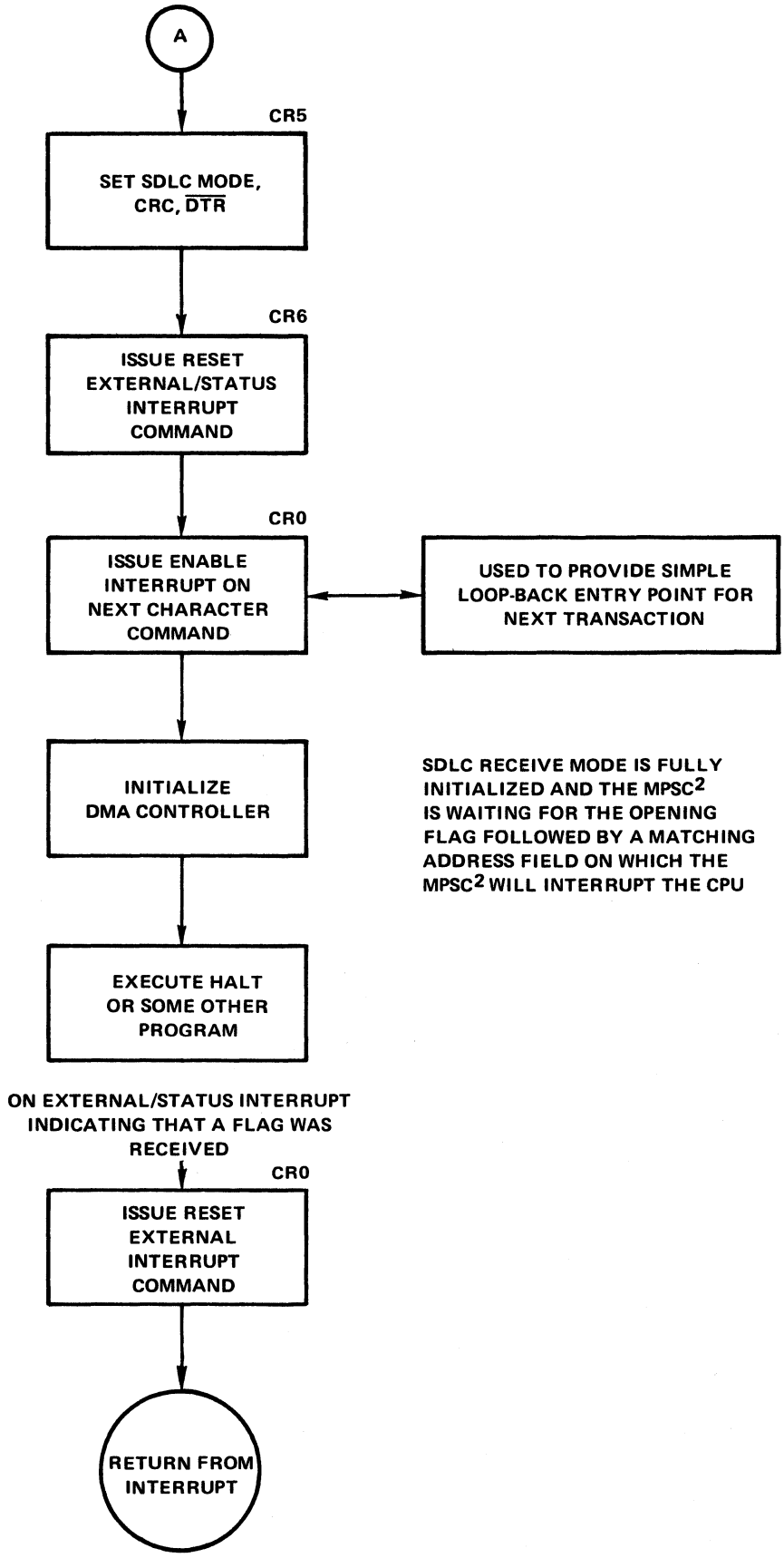
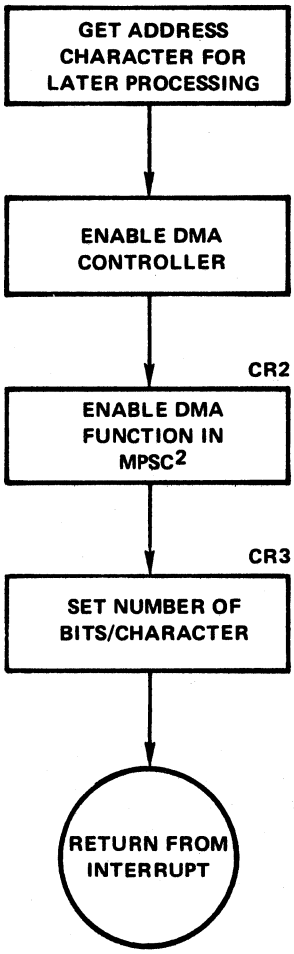


Figure 5.18 SDLC Initialization Transmit


CR0
ISSUE CHANNEL RESET
CR2-A
SET INTERFACE OPTION
CR2-B
SET INTERRUPT VECTOR IF USED
CR4
SET BASIC INTERFACE PARAMETERS
PARITY INFORMATION, SYNC MODE, SDLC MODE, X1 CLOCK MODE
CR1
SET INTERRUPT PARAMETERS
STATUS AFFECTS VECTOR, EXTERNAL INTERRUPT ENABLE, RECEIVE INTERRUPT ON FIRST CHARACTER ONLY
IN THIS MODE, ONLY THE ADDRESS FIELD (1 CHARACTER ONLY) IS TRANSFERRED TO THE CPU. ALL SUBSEQUENT FIELDS (CONTROL, INFORMATION ETC.) ARE TRANSFERRED ON A DMA BASIS. STATUS AFFECTS VECTOR IN CHANNEL B ONLY.
CR6
ISSUE SECONDARY ADDRESS FIELD
THIS ADDRESS IS MATCHED AGAINST THE MESSAGE ADDRESS IN AN SDLC POLLED OPERATION
CR7
ISSUE SDLC FLAG 01111110
THIS FLAG DETECTS THE START AND END OF FRAME IN AN SDLC OPERATION
CR3
ENABLE RECEIVER
SET RECEIVER ENABLE ON, 8-BIT, RECEIVE CRC ON, ADDRESS SEARCH MODE ON
A


A
CR5
SET SDLC MODE, CRC, DTR
CR6
ISSUE RESET EXTERNAL/STATUS INTERRUPT COMMAND
CR0
ISSUE ENABLE INTERRUPT ON NEXT CHARACTER COMMAND
USED TO PROVIDE SIMPLE LOOP-BACK ENTRY POINT FOR NEXT TRANSACTION
INITIALIZE DMA CONTROLLER
SDLC RECEIVE MODE IS FULLY INITIALIZED AND THE MPSC2 IS WAITING FOR THE OPENING FLAG FOLLOWED BY A MATCHING ADDRESS FIELD ON WHICH THE MPSC2 WILL INTERRUPT THE CPU
EXECUTE HALT OR SOME OTHER PROGRAM
ON EXTERNAL/STATUS INTERRUPT INDICATING THAT A FLAG WAS RECEIVED
CR0
ISSUE RESET EXTERNAL INTERRUPT COMMAND
RETURN FROM INTERRUPT

WHEN INTERRUPT ON FIRST CHARACTER OCCURS



THE MPSC2 IS NOW IN THE ADDRESS SEARCH PHASE. DURING THIS PHASE THE MPSC2 INTERRUPTS WHEN THE PROGRAMMED ADDRESS MATCHES THE MESSAGE.

THE MPSC2 RECEIVER IS NOW IN THE DATA PHASE AND WILL TRANSFER ALL SUCCEEDING CHARACTERS BY THE DMA CONTROLLER UNTIL THE END OF FROM FLAG.

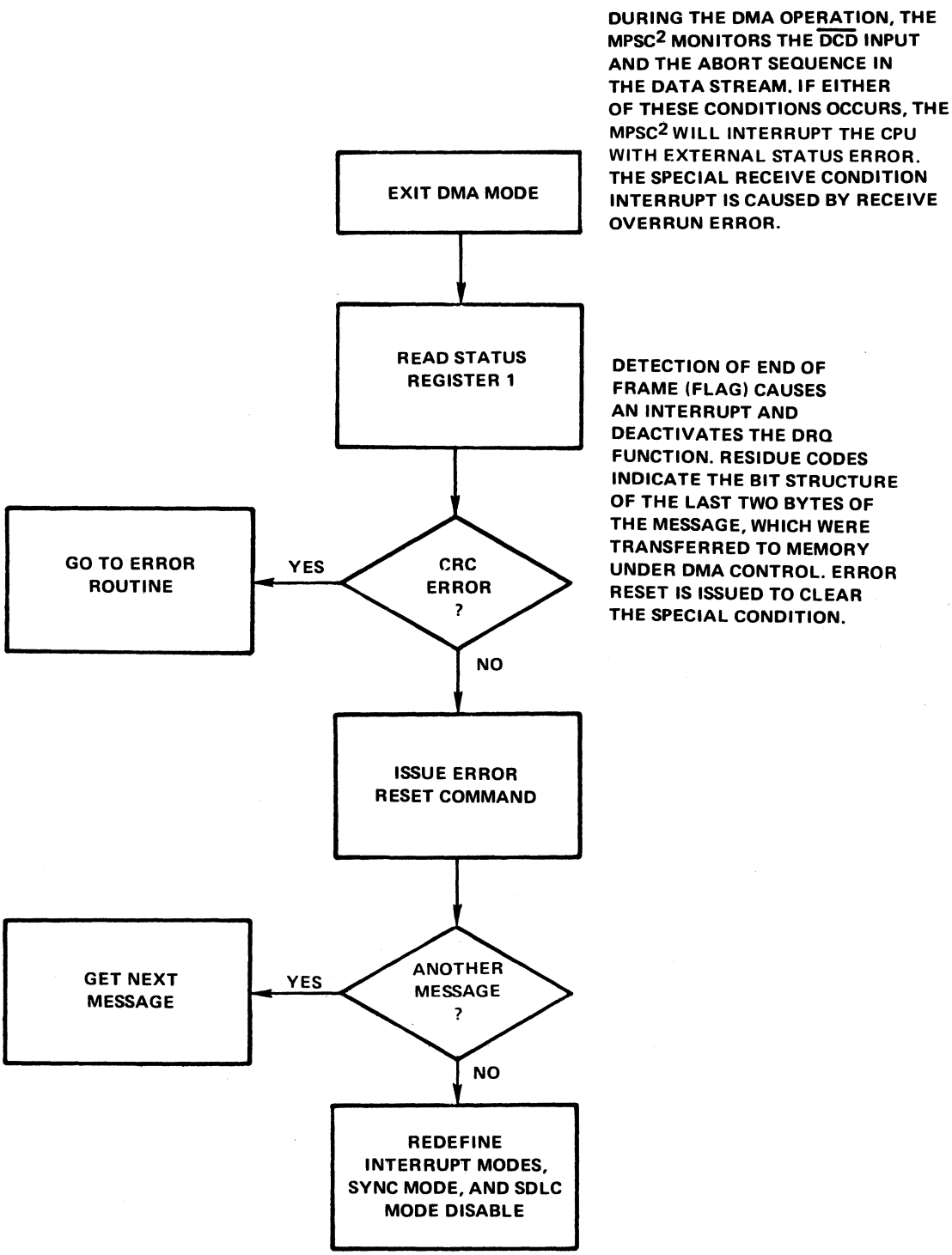


Figure 5.19 SDLC Initialization Receive

6.1 Designing with the MPSC^2

Designing the MPSC^2 into your system is generally straightforward and requires a minimal number of external devices.

6.1.1 8080/86-Type Processors

The bus interface used by the MPSC^2 is directly compatible with 8080/86-type buses. Figure 6.1 illustrates the basic interconnection scheme for these processors. This configuration supports polled, interrupt driven, and block mode operation.

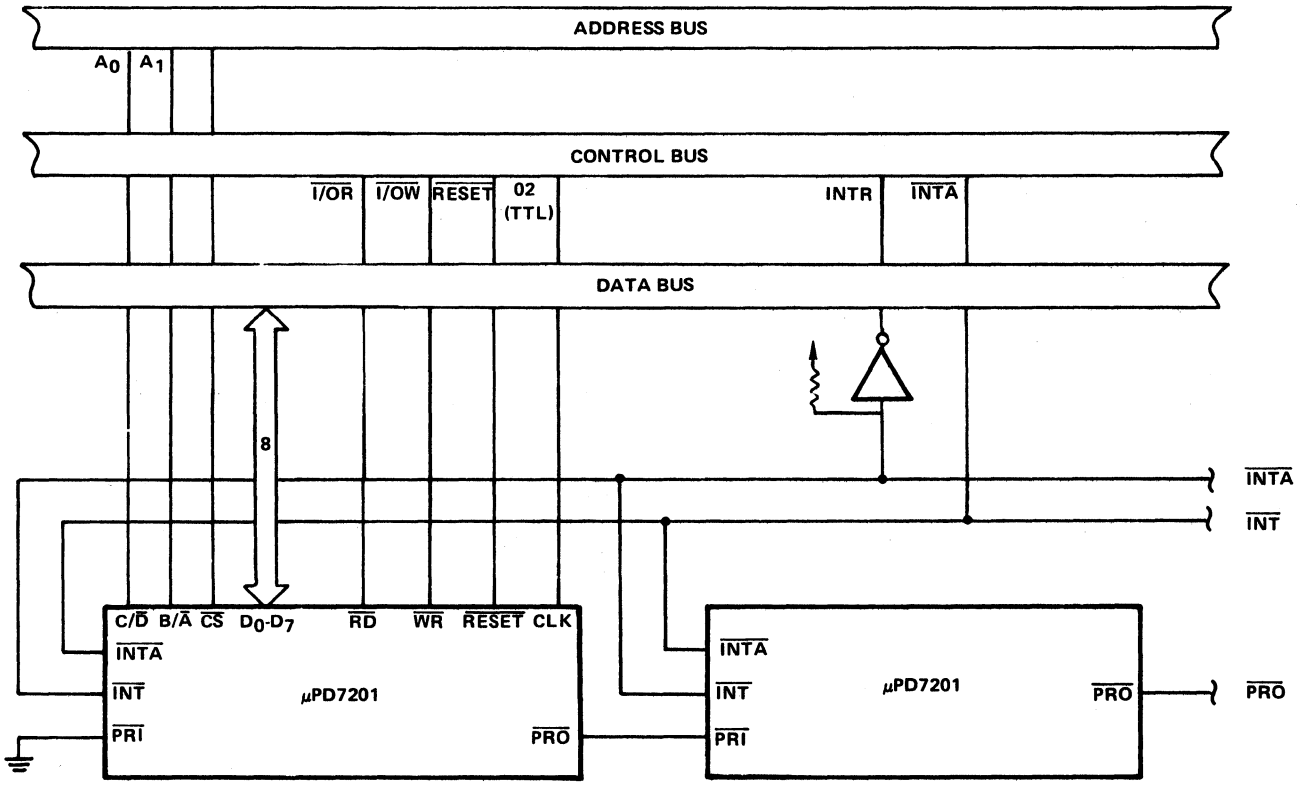


Figure 6.1 uPD7201 Interface to 8080 Standard System Bus (Non-DMA)

6.1.2 Other Processor Types

You may also connect the MPSC^2 to uPD780 (Z-80) and 6800/6502-type processors with a few additional gates. Figures 6.2 and 6.3, respectively, illustrate the circuits necessary to derive the correct signals. In both cases the MPSC^2 can be used in Non-vectored mode with minimal software overhead.

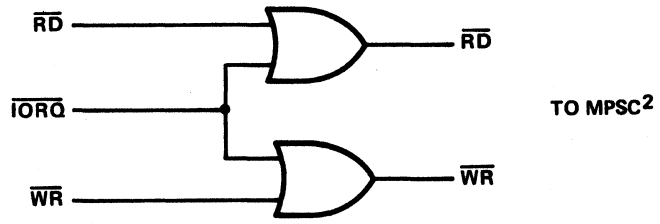


Figure 6.2 uPD780 (Z-80) to MPSC^2 Adapter

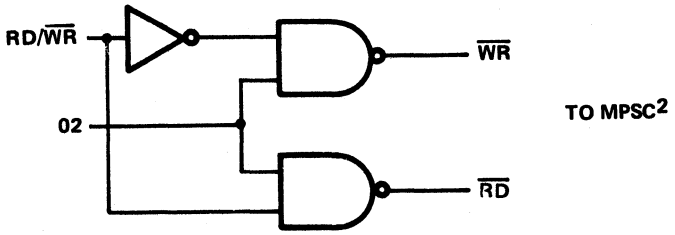


Figure 6.3 6800/6502 to MPSC^2 Adapter

The MPSC^2 can also be used in Vectored Interrupt mode with the uPD780 operated in Interrupt Mode 0. In this mode, the uPD780 handles interrupt requests in much the same manner as an 8080 processor, that is, an interrupt acknowledge sequence is executed during which the processor expects the next instruction to come from the interrupting device. The 8080 $\overline{\text{INTA}}$ signal is generated by combining M1 and IORQ from the uPD780. There is one key difference that must be noted. In accepting a multibyte instruction such as the CALL generated by the MPSC^2, the 8080 issues a separate $\overline{\text{INTA}}$ pulse for each byte. The uPD780, however, issues an $\overline{\text{INTA}}$ on the first byte only. Succeeding bytes are accessed with memory read cycles. In order for the MPSC^2 to operate properly, a circuit such as the one shown in Figure 6.4 should be used to derive the proper $\overline{\text{INTA}}$ sequence.

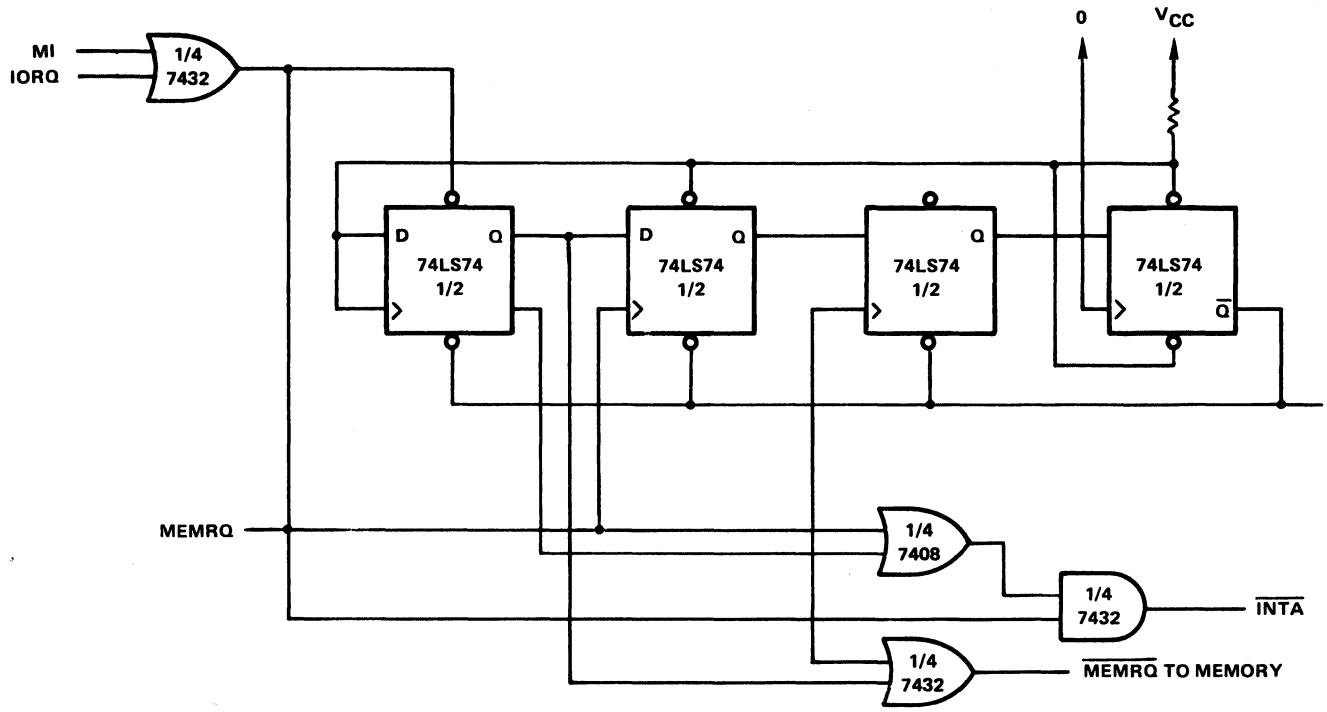


Figure 6.4 INTA Generator for Z-80

Most other types of processors may be readily accommodated. The bus control inputs $\overline{RD}$, $\overline{WR}$, $\overline{CS}$, C/$\overline{D}$, B/$\overline{A}$, and $\overline{INTA}$ have no timing requirements with respect to the system clock (CLK) and there is no hold time requirement for data after the trailing edge of $\overline{WR}$. The only timing constraint you must observe is that the address lines C/$\overline{D}$, B/$\overline{A}$, and $\overline{CS}$ must be stable by the leading edge of $\overline{RD}$ or $\overline{WR}$.

6.2 Using the MPSC^2 with DMA Controllers

You can greatly increase the data handling capacity of your serial I/O subsystem by using the MPSC^2 with a DMA controller such as the uPD8257 or uPD8237, to permit direct transfer of data between the MPSC^2 and memory. Figure 6.5 illustrates a typical MPSC^2/DMA configuration. In using the MPSC^2 in this manner, you should be aware of a few special considerations:

To minimize the number of pins required to implement four DMA channels, the MPSC^2 does not use the usual DRQ/DACK pins for each channel but rather only DRQ with a single Hold Acknowledge input, $\overline{HAI}$. This arrangement eliminates three pins and in addition permits daisychained MPSC^{2}s operating in DMA mode. However, it does require that the MPSC^2 and the DMA controller reach independent agreement on which DMA request is to be serviced in the case of multiple requests to the same controller.

To ensure that this agreement does occur, you should program the DMA controller for a fixed priority arrangement that agrees with the DMA priority you programmed into the MPSC^2 (see Section 5.1). You must also allow sufficient time for the MPSC^2 to determine its internal request priority before the DMA controller begins the data transfer. Activating the DMA controller's Hold Acknowledge input through the delay circuit shown in Figure 6.5 provides this time delay.

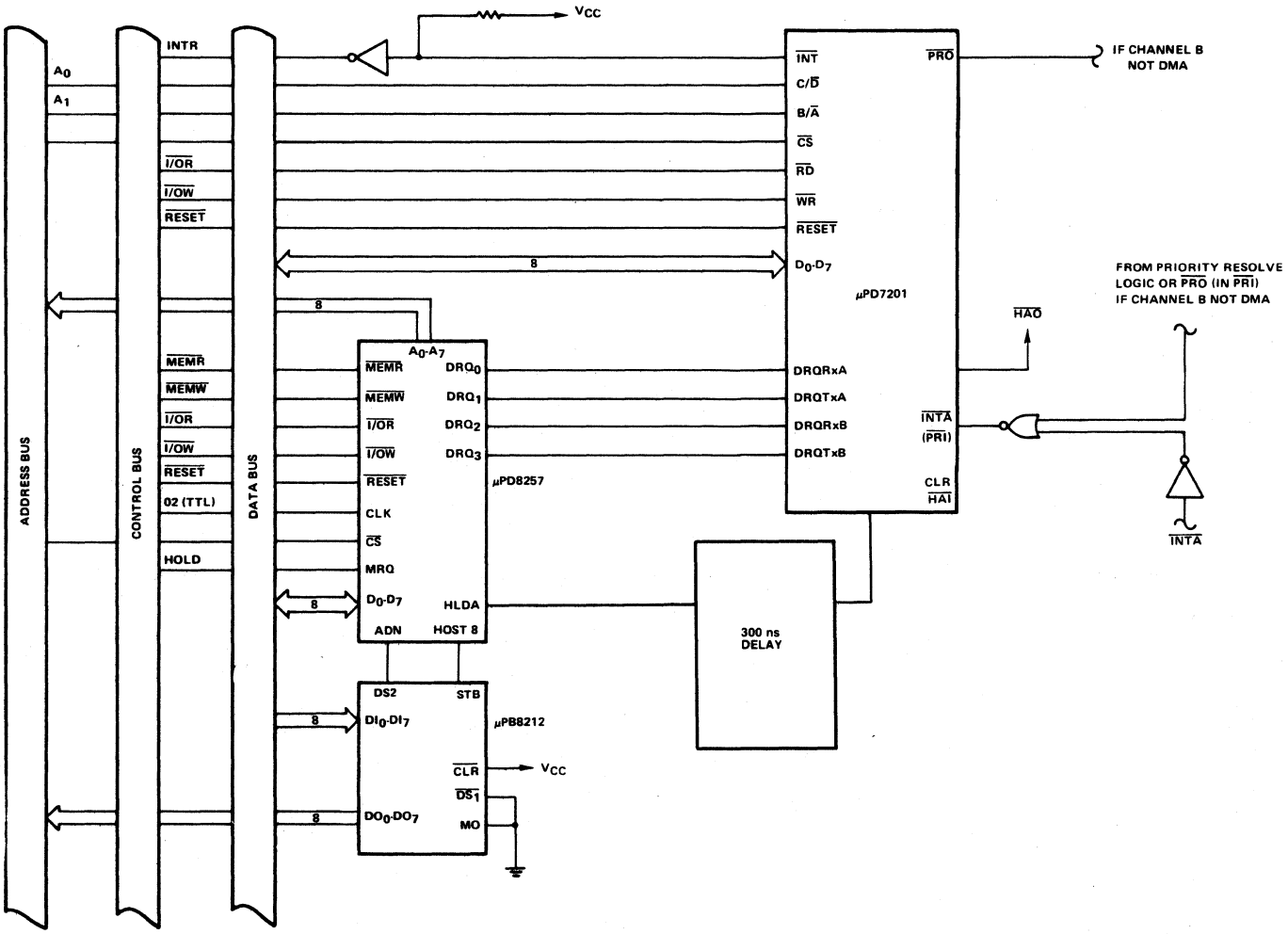


Figure 6.5 DMA Interface

6.3 Vectored Interrupts Without Using $\overline{PRI}$

There are circumstances when you may wish to use the MPSC^2's Vectored Interrupt feature and you cannot use $\overline{PRI}$ to inform the MPSC^2 whether it is the highest priority device requesting service. These situations can occur when both channels are being used in DMA mode (the $\overline{PRI}$ pin becomes DRQRxB) or when using other peripherals that are incompatible with daisychaining. To retain the Vectored Interrupt feature, you can pull $\overline{PRI}$ low if available (this is done automatically when both channels are DMA). Program the MPSC^2 for either 8080 Master or 8086 Vector mode, and gate $\overline{INTA}$ to the highest priority device with a circuit similar to Figure 6.6.

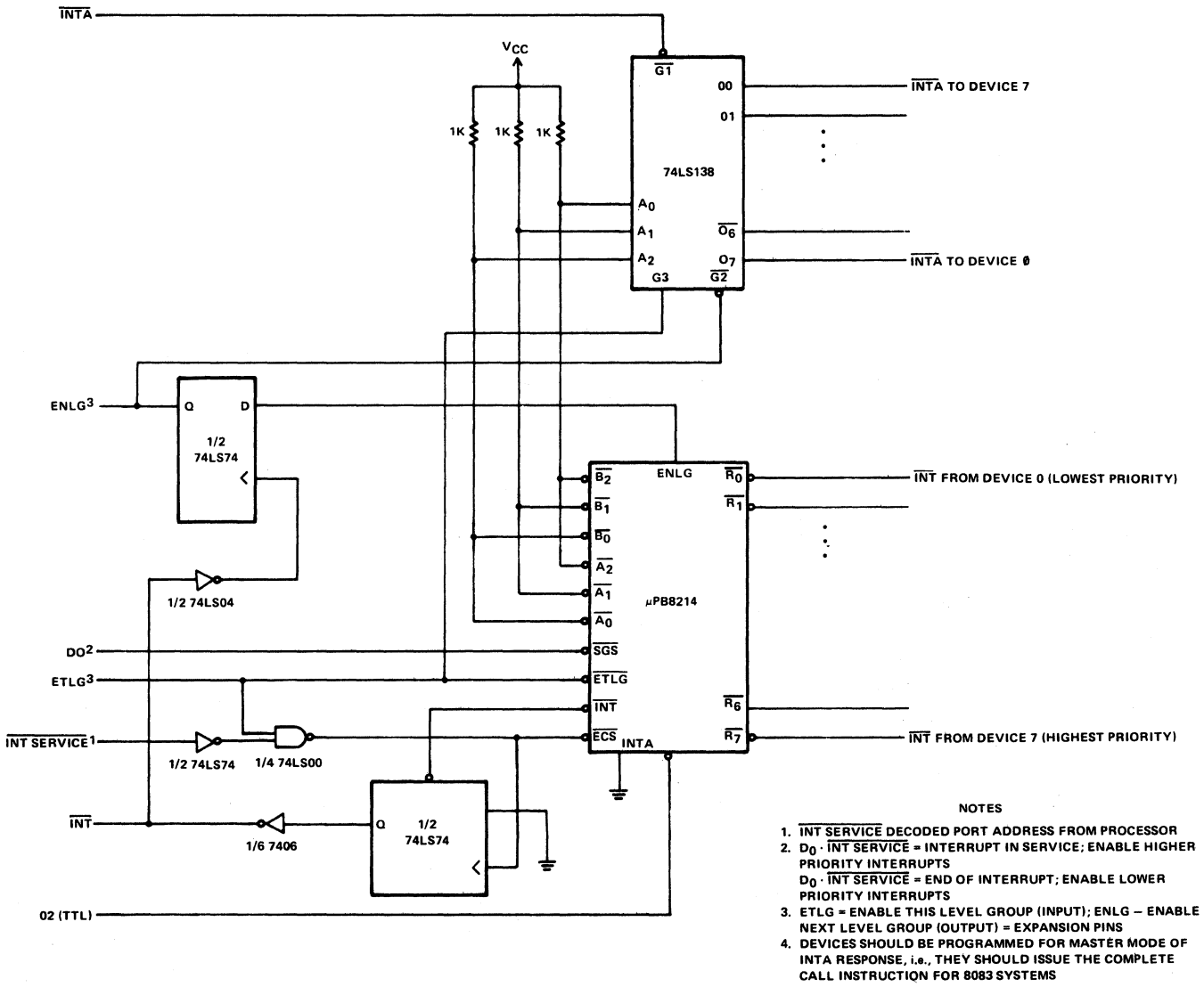


Figure 6.6 Priority Resolution Circuit for Non-daisychained devices

You should note that an 8259-type interrupt controller programmed for Master mode does not set its Slave Enable outputs until the second $\overline{INTA}$ pulse and so is incompatible with the MPSC^2's interrupt acknowledge timing.

6.4 To DMA Or Not To DMA...

When operating an MPSC^2 channel in DMA mode, there are normally some interrupts in parallel with DMA requests. Here are the rules:

Interrupt on Each Character Mode: Both an interrupt and DMA request are made when a character is received.

Interrupt on First Character: The first character received (after issuing an Enable Interrupt On Next Character) generates both an interrupt and a DMA request. Subsequent characters cause only a DMA request to be issued. As an exception, a Special Receive condition always causes both an interrupt and a DMA request.

Transmitter Buffer Becoming Empty: Only DMA requests are issued when the MPSC^2 is transmitting under DMA control.

6.5 Handling an SDLC Underrun Fault

Since SDLC-type protocols do not allow flags to be imbedded within a message as filler, a fault condition can sometimes occur where the transmitter runs out of data to send. This situation is particularly common in interrupt-driven systems that are heavily task-loaded. You can use the MPSC^2's Idle/CRC latch feature to detect these underrun faults and abort the message before an erroneous End of Frame flag is sent. This is accomplished by issuing a Reset Idle/CRC Latch command to the MPSC^2 immediately after loading it with the first character of the message. If an underrun condition occurs, the MPSC^2 automatically begins to send the CRC character calculated up to that point and issues an External/Status Change interrupt to indicate that the CRC is being sent. Since your software routine knows that the end of the message has not been reached, an underrun is indicated and your routine can immediately abort the message with a Send Abort command.

6.6 Sending Synchronous Pad Characters

If you want to send one or more pad characters between synchronous messages, you can do it two ways with the MPSC^2:

When the MPSC^2 issues an External/Status interrupt to indicate that CRC is being sent, you can begin loading your pad characters into the transmitter.

Instead of loading pad characters in response to the above interrupt, you can simply change the programmed sync character on the fly, and the MPSC^2 will transmit pads when it enters Idle Phase after sending CRC.

6.7 Transmitting Bisync Transparent Mode

Because of the ability to change the sync registers (CR6, CR7) on the fly, the MPSC^2 is truly compatible with bisync protocol's Transparent mode. On entering this mode, program CR6 with the DLE character and, if an underrun condition occurs, the correct DLE-SYN sequence is transmitted. On leaving

Transparent mode you should reset CR6 back to SYN.

6.8 Vectoring the MPSC^2 in Non-Vectored Mode

If you're using the MPSC^2 in Non-vectored Interrupt mode, you can still use the Condition Affects Vector feature to direct your software to the correct routine. The following example, written in 8080 assembler, assumes that the MPSC^2 has been programmed for either 8085 master or slave mode (D_3-D_5 modified) and that CR2B was programmed with a zero.

```
MPSCINT:
        PUSH B          ;Save state so registers are free for
        PUSH D          ;your service routine
        PUSH H
        PUSH PSW

        MVI  A,2        ;Set channel B register pointer to 2
        OUT  MPSCBC
        IN   MPSCBC     ;Register A = modified vector
```

```
        LXI   H,JMPTBL  ;HL--> vector jump table
        MVI   D,0       ;DE = offset into table
        MOV   E,A
        DAD   D         ;HL--> jump table + offset
        PCHL            ;Jump to jump table entry

JMPTBL  JMP   TBEB      ;Channel B transmitter buffer empty
        NOP
        JMP   EXTB      ;External/Status change
        NOP
        JMP   RCVB      ;Received character available

        NOP
        JMP   SPRB      ;Special receive condition
        NOP
        .
        .               ;Repeat for channel A interrupts
        .
        END
```

APPENDIX A

Commands:

000	Null
001	Send SDLC Abort
010	Reset External/Status Report
011	Channel Reset
100	Enable Interrupt on Next Character
101	Reset Reading Transmitter Interrupt/DMA Request
110	Error Reset
111	End of Interrupt (Channel A only)

CRC Control Commands:

00	Null
01	Reset Receiver CRC Checker
10	Reset Transmitter CRC Generator
11	Reset IDLE/CRC Latch

Absolute Maximum Ratings

Operating temperature. . . . . 0^{o} to $+70^{o}C$

Storage temperature

Ceramic package. . . . . -65^{o} to $+150^{o}C$

Plastic package. . . . . -60^{o} to $+125^{o}C$

Voltage on any pin. . . . . -0.5 to +7 volts *

* With respect to ground

Comment: Exposing the device to stresses above those listed in Absolute Maximum Ratings could cause permanent damage. The device is not meant to be operated under conditions outside the limits described in the operational sections of this specification. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

DC Characteristics

T_a = 0°C to +70°C; V_{CC} = +5V ±10%

PARAMETER	SYMBOL	LIMITS		UNIT	TEST CONDITIONS
		MIN	MAX		
Input Low Voltage	V_{IL}	-0.5	+0.8	V	
Input High Voltage	V_{IH}	+2.0	V_{CC}+0.5	V	
Output Low Voltage	V_{OL}		+0.45	V	I_{OL} = +2.0 mA
Output High Voltage	V_{OH}	+2.4		V	I_{OH} = -200 μA
Input Leakage Current	I_{IL}		±10	μA	V_{IN} = V_{CC} to 0V
Output Leakage Current	I_{OL}		±10	μA	V_{OUT} = V_{CC} to 0V
V_{CC} Supply Current	I_{CC}		180	mA	

Capacitance

T_a = 25°C; V_{CC} = GND = 0V

PARAMETER	SYMBOL	LIMITS		UNIT	TEST CONDITIONS
		MIN	MAX		
Input Capacitance	C_{IN}		10	pF	fc = 1 MHz
Output Capacitance	C_{OUT}		15	pF	Unmeasured pins Returned to GND
Input/Output Capacitance	$C_{I/O}$		20	pF	

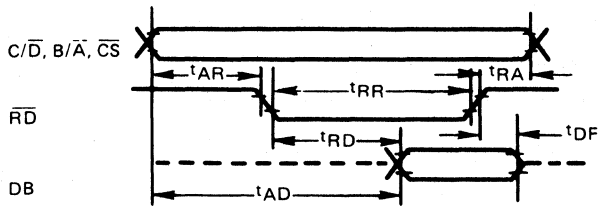
AC Characteristics

T_a = 0°C to +70°C; V_{CC} = +5V ±10%

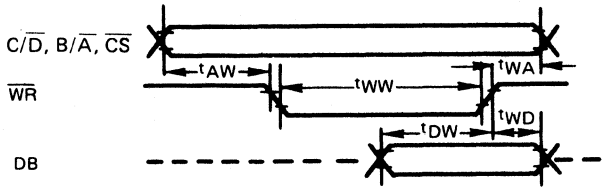
PARAMETER	SYMBOL	LIMITS		UNIT
		MIN	MAX	
Clock Cycle	t_{CY}	250	4000	ns
Clock High Width	t_{CH}	105	2000	ns
Clock Low Width	t_{CL}	105	2000	ns
Clock Rise and Fall Time	t_r, t_f	0	30	ns
Address Setup to $\overline{RD}$	t_{AR}	0		ns
Address Hold from $\overline{RD}$	t_{RA}	0		ns
$\overline{RD}$ Pulse Width	t_{RR}	250		ns
Data Delay from Address	t_{AD}		200	ns
Data Delay from $\overline{RD}$	t_{RD}		200	ns
Output Float Delay	t_{DF}	10	100	ns
Address Setup to $\overline{WR}$	t_{AW}	0		ns
Address Hold from $\overline{WR}$	t_{WA}	0		ns
$\overline{WR}$ Pulse Width	t_{WW}	250		ns
Data Setup to $\overline{WR}$	t_{DW}		150	ns
Data Hold from $\overline{WR}$	t_{WD}	0		ns
$\overline{PRO}$ Delay from $\overline{INTA}$	t_{IAPO}		200	ns
$\overline{PRI}$ Setup to $\overline{INTA}$	t_{PIN}	0		ns
$\overline{PRI}$ Hold from $\overline{INTA}$	t_{IP}	0		ns
$\overline{INTA}$ Pulse Width	t_{II}	250		ns
$\overline{PRO}$ Delay from $\overline{PRI}$	t_{PIPO}		100	ns
Data Delay from $\overline{INTA}$	t_{ID}		200	ns
Request Hold from $\overline{RD}/\overline{WR}$	t_{CO}		150	ns
$\overline{HAI}$ Setup to $\overline{RD}/\overline{WR}$	t_{LR}	300		ns
$\overline{HAI}$ Hold from $\overline{RD}/\overline{WR}$	t_{RL}	0		ns
$\overline{HAO}$ Delay from $\overline{HAI}$	t_{HIHO}		100	ns
Recovery Time Between Controls	t_{RV}	300		ns
$\overline{WAIT}$ Delay from Address	t_{CW}		120	ns
Data Clock Cycle	t_{DCY}	400		ns
Data Clock Low Width	t_{DCL}	180		ns
Data Clock High Width	t_{DCH}	180		ns
Tx Data Delay	t_{TD}		300	ns
Data Set up to $\overline{RxC}$	t_{DS}	0		ns
Data Hold from $\overline{RxC}$	t_{DH}	140		ns
$\overline{INT}$ Delay Time from $\overline{TxC}$	t_{ITD}		4 ~ 6	t_{CY}
$\overline{INT}$ Delay Time from $\overline{RxC}$	t_{IRD}		7 ~ 11	t_{CY}
Low Pulse Width	t_{PL}	200		ns
High Pulse Width	t_{PH}	200		ns
External $\overline{INT}$ from $\overline{CST}$, $\overline{DCD}$, $\overline{SYNC}$	t_{IPD}		500	ns
Delay from $\overline{RxC}$ to $\overline{SYNC}$	t_{DRxC}		100	ns

Timing Waveforms

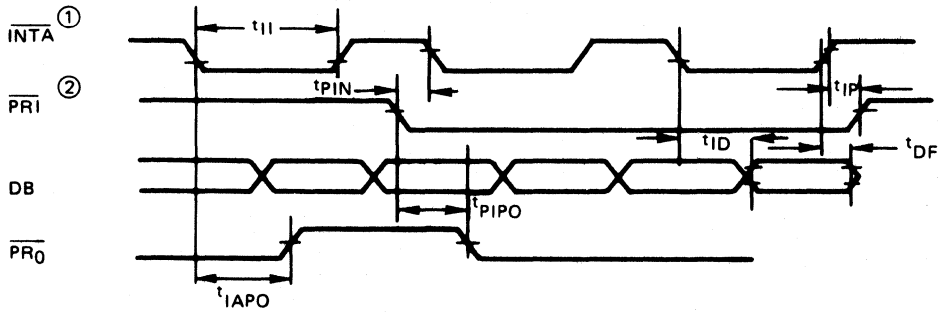
READ CYCLE



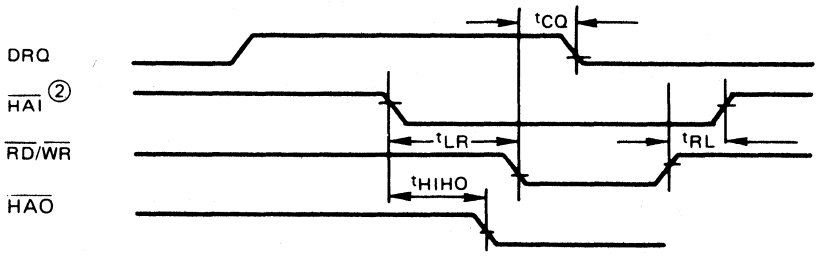
WRITE CYCLE



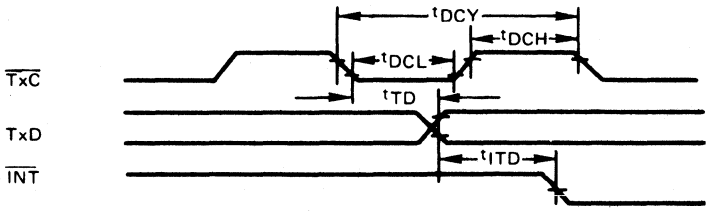
INTA CYCLE



DMA CYCLE

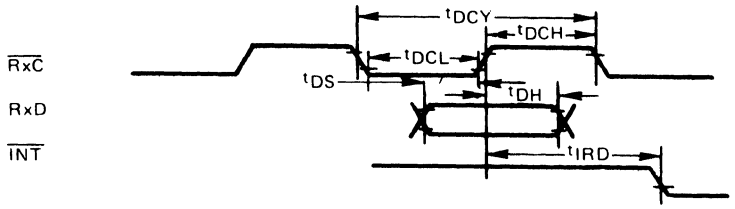


TRANSMIT DATA CYCLE

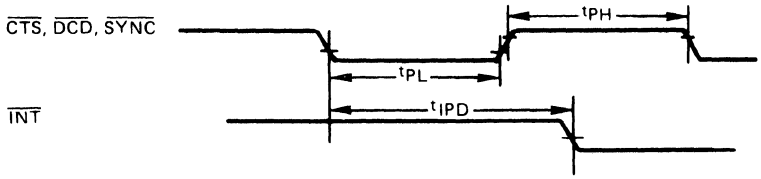


Notes: ① INTA signal acts as RD signal.
② PRI and HAI signals act as CS signal.

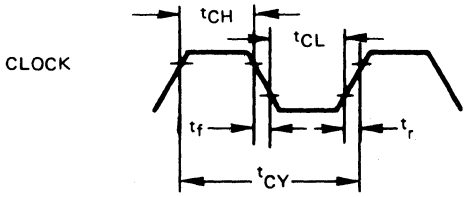
RECEIVE DATA CYCLE



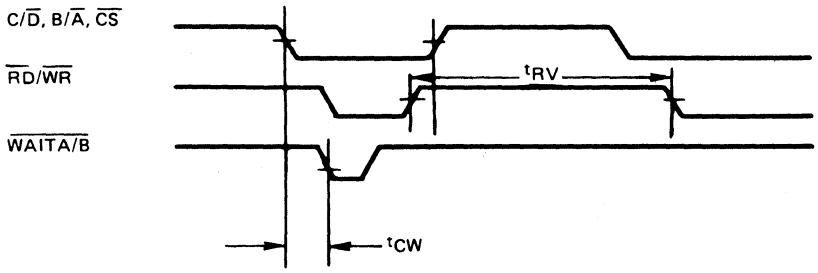
OTHER TIMING



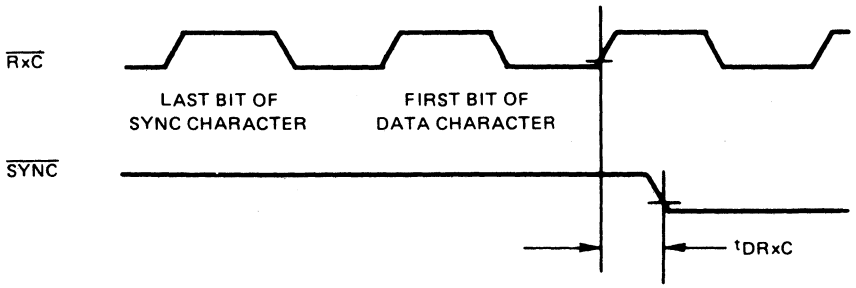
CLOCK



READ/WRITE CYCLE (SOFTWARE BLOCK TRANSFER MODE)



SYNC PULSE GENERATION (EXTERNAL SYNC MODE)



Control Register 0

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CRC CONTROL COMMAND		COMMAND			REGISTER POINTER		

Control Register 1

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
WAIT FUNCTION ENABLE	0	WAIT ON RECEIVER TRANSMITTER	RECEIVER INTERRUPT MODE		CONDITION AFFECTS VECTOR	TRANSMITTER INTERRUPT ENABLE	EXT/STATUS INT ENABLE

Control Register 2 (Channel A)

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
PIN 10 $\overline{\text{SYNCB}}$/RTSB	0	INTERRUPT VECTOR MODE			PRIORITY	DMA MODE SELECT	

Control Register 2 (Channel B)

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
INTERRUPT VECTOR							

Control Register 3

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
NUMBER OF RECEIVED BITS/CHARACTER		AUTO ENABLES	ENTER HUNT PHASE	RECEIVER CRC ENABLE	ADDRESS SEARCH MODE	SYNC CHARACTER LOAD INHIBIT	RECEIVER ENABLE

Control Register 4

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CLOCK RATE		SYNC MODE		NUMBER OF STOP BITS SYNC MODE		PARITY EVEN/ODD	PARITY ENABLE

Control Register 5

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
DTR	NUMBER OF TRANSMITTED BITS/CHARACTER		SEND BREAK	TRANSMITTER ENABLE	CRC POLYNOMIAL SELECT	$\overline{\text{RTS}}$	TRANSMITTER CRC ENABLE

Control Register 6

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SYNC BYTE 1							

Control Register 7

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SYNC BYTE 2							

NEC
NEC Electronics, Inc.

APPLICATION NOTE 28
MPSC² BIT-SYNCHRONOUS DATA COMMUNICATIONS

Introduction

The multiprotocol serial communication controllers designed today are a product of recent advances in technology and the continuing evolution of the computer systems. To keep up with this rapid growth NEC has designed a flexible VLSI multiprotocol serial communication controller called appropriately MPSC². Its innovative design satisfies a wide variety of serial data communication functions that will minimize both software and hardware requirements.

The MPSC² contains two completely independent fully-duplexed channels in a 40-pin package and incorporates a variety of sophisticated features that simplify protocol needs. Designed primarily as a parallel-to-serial and serial-to-parallel converter–controller it can be programmed to conform to virtually any protocol format.

Scope

The purpose of this application note is to provide the reader with the conceptual tools required to use the μPD7201/MPSC² in the SDLC mode. In order to utilize the serial attributes of the μPD7201 it is necessary to understand the SDLC protocol and how the 7201 operates.

General Description of the MPSC²

The MPSC² internal architecture includes an 8080/85, 8086/88 bus logic controller, a multilevel interrupt controller, a DMA logic controller and two fully-duplexed multiprotocol serial I/O channels. These control units are connected to a common internal data bus which can be accessed externally through the bus control logic.

Bus Control Logic

The bus control logic on the MPSC² determines the direction and the internal source and destination of data and control information being transferred between the MPSC² and the bidirectional data bus. The information transferred can be in the form of data, commands, or status. Table 1 illustrates the bus control logic selection.

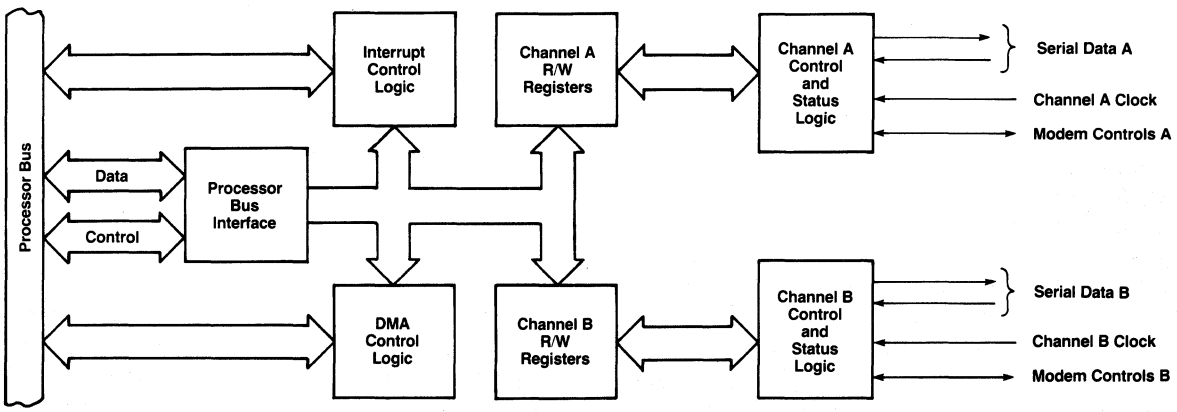
$\overline{CS}$	B/$\overline{A}$	C/$\overline{D}$	Read Operation	Write Operation
0	0	0	Read Channel A Data	Write Channel A Data
0	1	0	Read Channel B Data	Write Channel B Data
0	0	1	Read Channel A Control	Write Channel A Control
0	1	1	Read Channel B Control	Write Channel B Control
1	X	X	No Operation	No Operation

Table 1. Read/Write Selection

Features

- ☐ Implements the three basic data communications protocols
  - – Asynchronous
  - – Character-oriented synchronous (monosync, bisync, external sync, DDCMP
  - – Bit-oriented synchronous (SDLC, HDLC)
- ☐ Provides extensive error checking
  - – Parity
  - – CRC-16
  - – CRC-CCITT
  - – Break detection
  - – Abort detection
  - – Framing error detection
- ☐ Enhanced data reliability
  - – Double-buffered transmitters
  - – Quadruple-buffered receivers
  - – Programmable transmit underrun
  - – End of message handling
- ☐ Programmable interrupt logic
  - – 8080/85/86/88 compatible
  - – Eight-level priority controller
  - – Programmable priority modes
  - – Selectable vector or nonvector mode
  - – Status affects vector
  - – Priority input and output for expandability
- ☐ DMA priority and control logic
  - – Four independent DMA channels
  - – Priority input and output for expandability

Block Diagram of the MPSC² Internal Architecture



Internal Interrupt Controller

The multilevel interrupt control logic has been designed to minimize the software and real-time overhead associated with handling nested interrupts. (See Figure 1.) The logic manages eight levels of internal request and has built-in features for daisy-chaining to other MPSC^2 or slave controller devices. A selection of priority modes is available so that the manner in which the requests are processed by the MPSC^2 can be configured to match individual system requirements.

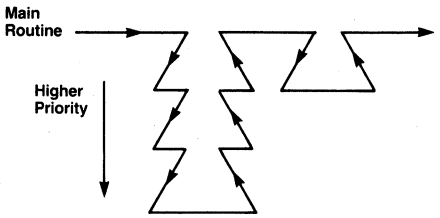


Figure 1. Nested Interrupt Representation

Internal DMA Control Logic

The DMA control logic is capable of interacting with a 4-channel DMA controller to increase throughput and simplify the transfer of data. This logic manages and prioritizes four levels of service requests. It also has built-in features for daisy-chaining to other MPSC^{2}s or slave controller devices when operating under DMA control.

Communication Channels

The MPSC^2 contains two completely independent fully-duplexed serial channels. Each incorporates a variety of sophisticated features to simplify personal communication requirements. Each channel can be configured independently and is capable of handling asynchronous, synchronous bit-oriented (monosync, bisync, DDCMP), and synchronous bit-oriented (SDLC, HDLC, LAP, and LAPB) protocols. (See Figure 2.)

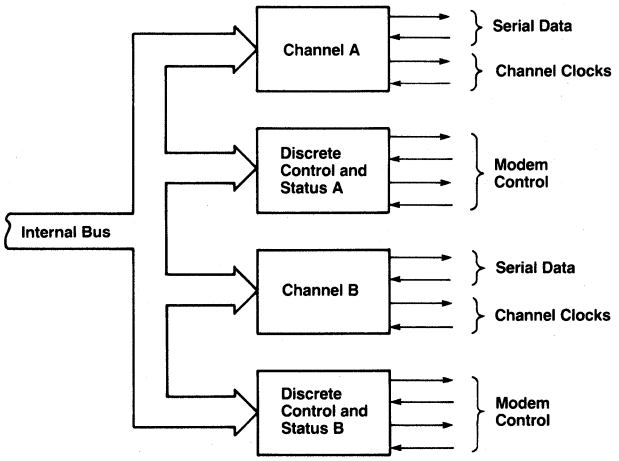


Figure 2. Block Diagram of MPSC^2's Independent Channels

Synchronous Data Link Control

The main task for the synchronous data link control (SDLC) is simply to establish and terminate an error-free logic connection between two stations. An example of this is the connection between a computer and a remote terminal that achieves high throughput and accurate transmission of serial bit-synchronous data.

SDLC operates on data transmission links that may be customer-owned, leased from a common carrier, or part of a public switched telephone network. It can operate in point-to-point, multipoint, and loop configurations.

SDLC and the MPSC^2

The trend toward distributed processing has been a major contributor to the efficiency and reliability of data communication, both physically and logically. Due to the complexity of the rules that govern the exchange in serial data communications these rules have been divided into subsets, each performing a very precise function. These subsets are known as layers. (See Figure 3.)

The layers are organized such that each layer offers services to the upper layers, and the upper layers use services provided by the lower layers. In this application note concentration will be only on the layers that deal more directly with the MPSC^2 itself. These layers are the data link layer and the physical layer.

Physical layer

The physical layer provides the mechanical and electrical functions and procedures that establish and maintain the physical connection. An example of some standards are X.21, V.24, V.35, and so forth.

Data link layer

The data link layer provides the functional and procedural means to establish and maintain the data link. This is done by utilizing data link control procedures or protocols such as SDLC in order to transmit data reliably.

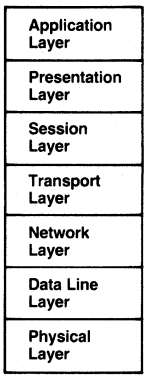


Figure 3. The Seven Layer ISO Architecture

SDLC Overview

The MPSC^2 is capable of handling both high-level synchronous data link control (HDLC) and synchronous data link control (SDLC). In the text that follows only SDLC will be referred to because of the similarity between the two protocols.

SDLC Protocol

In SDLC all transmissions occur in frames and conform to the one shown in Figure 4. As shown a frame starts with an 8-bit flag sequence, followed by a station address, a control sequence, an information sequence (but not necessarily), a frame check sequence, and ends with another flag sequence.

Flags

There are two flags in each SDLC frame, one at the beginning of the frame and one at the end. These flags are reference points for the positioning of the address field, the control field, the information field, and transmission error check information. Both beginning and ending flags have a binary configuration of 01111110.

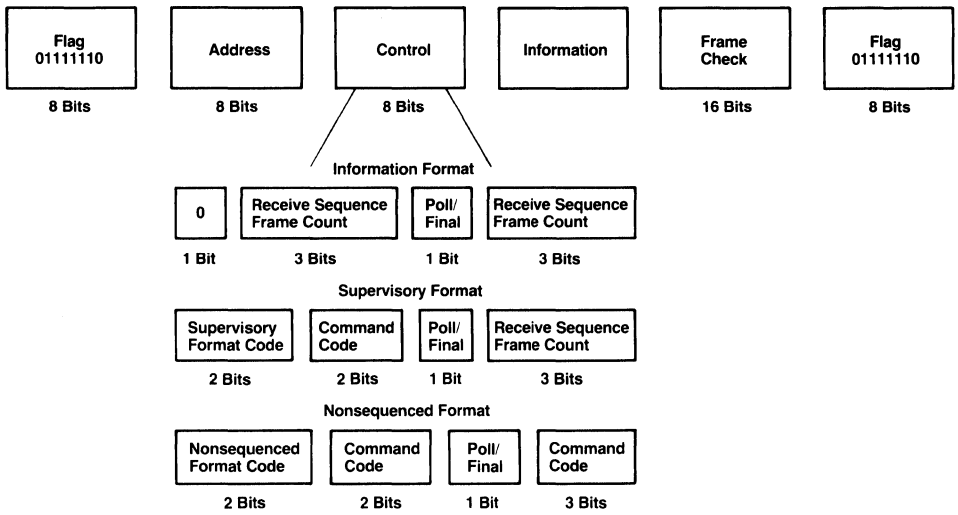


Figure 4. SDLC Frame

Address

The primary station manages a data link by issuing commands to secondary stations that respond. The primary station has no address; only the secondary stations are identified. Each station attached to the data link continuously hunts for a flag followed by an address sequence. A station must recognize its own address in order to get on line. In multipoint operations, for example, a secondary station will transmit an address that tells the primary station which secondary station originated the frame.

Control field

The control sequence can be quite complex and consequently will not be discussed here. (For more information, refer to IBM spec number GA27–3090–0, File No. GENL–09, "IBM Synchronous Data Link Control: General Information.") In general, the control sequence may contain one of three formats: information format, supervisory format and nonsequenced format. These formats contain send and receive information needed by the primary and secondary stations.

Information field

The information field is transparent and is not restricted in its format or its content. In addition, it may be any number of data bytes.

Frame check field

The frame check sequence is two bytes of data that follow the information field. The process in which the frame check field functions is called the CRC. The two CRC bytes are the result of a mathematical computation of all the fields contained between two flags.

For more information refer to the IBM Synchronous Data Link Control General Information bulletin (GA27-3093-0).

Programming the MPSC^2 for SDLC Operations

Control and Status Registers

The MPSC^2 has seven control registers associated with each channel. These control registers are accessed indirectly through the main control register. All control registers except control register 2 channel A (CR2A) are separately maintained for each channel. When initializing the MPSC^2, control register 2A should be programmed first to establish the MPSC^2 bus interface. Control register 4 should be programmed next to establish the mode. The remaining registers may then be programmed in any order.

The register pointer in the main control register specifies which register is to be accessed next. After a software or hardware reset the register pointer will equal zero. Therefore, the first write to the MPSC^2 will be to control register 0. This write can be in the form of a command, register number, or both. As with most rules there are exceptions, and the exception to this rule is that a channel reset command will reset the pointer back to zero. Thus, this command should not be mixed with a register pointer. After a register pointer has been specified, the next write to the MPSC^2 will be to the register specified, after which the pointer will be reset back to zero.

The MPSC^2 also has three status registers associated with each channel. These status registers can be read by the host CPU at any time to determine the MPSC^2's present state. Status register 2 is the only read/write register and one of two registers shared by both channels.

Figure 5 is a block representation of the MPSC^2's registers.

Control Register Number	Control Register Function
0	Frequently used commands; Register pointer
1	Interrupt control
2A	Bus interface specification
2B	Interrupt vector
3	Receiver control
4	Transmit and receive mode control
5	Transmitter control
6	Sync 1
7	Sync 2

Status Register Number	Status Register Function
0	External/phase and buffer status
1	Received character status
2B	Interrupt vector

Figure 5. Block Representation of the MPSC^2's Registers

SDLC (HDLC) BOP Synchronous Mode

Figure 6 is a typical program sequence example of how the MPSC^2 would be initialized to run in the DMA mode using the SDLC protocol. It must be pointed out that this is not the only configuration possible and is only an example to help better understand the MPSC^2.

MPSC² Initialization (INI)

In order to operate in the SDLC configuration the MPSC² must be initialized with the following parameters: character length, clock rate, error mode, I/O mode, and Rx and Tx enables. The parameters are loaded into the correct control register by the INI initialization subroutine. Typical values for these parameters are located in Table 2.

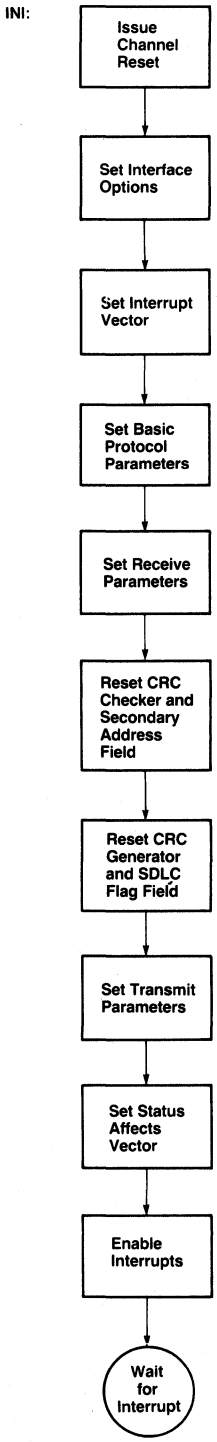


Figure 6. Flow Chart of Initialization

Table 2. Typical WRITE Initialization Commands

Write Register 0

0	0	0	1	1	0	0	0

0 0 0	Register 0	Pointer for the Selection of a Read/Write Register
0 0 1	Register 1	
0 1 0	Register 2	
0 1 1	Register 3	
1 0 0	Register 4	
1 0 1	Register 5	
1 1 0	Register 6	
1 1 1	Register 7	

0 0 0	Null Code
0 0 1	Send Abort (SDLC)
0 1 0	Reset EXT/Status Interrupts
0 1 1	Channel Reset
1 0 0	Enable INT on Next Rx Character
1 0 1	Reset Tx INT/DMA Pending
1 1 0	Error Reset
1 1 1	End of Interrupt (EOI — Channel A only)

0 0	Null Code
0 1	Reset Rx CRC Checker
1 0	Reset Tx CRC Generator
1 1	Reset Tx Underrun/EOM Latch

Channel reset: The reset is the same as a hardware rest except for a specific channel. After a channel reset all registers will be zero (0) and the Tx output will be held in a mark state.

Write Register 2 (Channel A)

0	0	0	0	0	0	1	0

0 0	Both Channels Interrupt
0 1	Channel A DMA; Channel B INT
1 0	Both Channels DMA-Internal Priority Mode
1 1	Both Channels DMA-External Priority Mode

0 Priority RxA > TxA > RxB > TxB
1 Priority RxA > RxB > TxA > TxB

0 0	8085 Master Mode
0 1	8085 Slave Mode
1 0	8086/88 Mode
1 1	8085/8259A Slave Mode

1 = Vectored; 0 = Nonvectored

Receive Interrupt Mask

0	RTSB Pin 10
1	SYNCB Pin 10

Set WR2A to indicate the following parameters: Nonvectored interrupts, 8085 Mode 1, priority RxA > TxA, both channels DMA.

Read Register 2 (Channel B)

0	0	0	X	X	X	0	0

V0 ②
V1 ②
V2 ②
V3 ②
V4 ②
V5
V6
V7

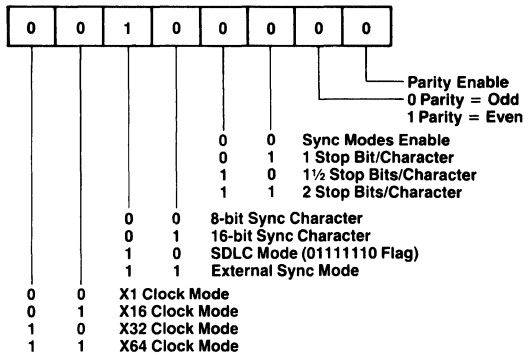
Interrupt Vector

Note: ① Variable if "status affects vector" is programmed.

Set WR2B to indicate the base address of the vector. In the 8085 mode bits D2–D4 are modified using status affects vector. In this case the base address is zero.

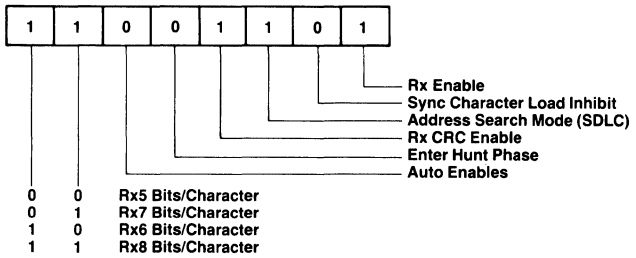
Table 2. Typical WRITE Initialization Commands (Cont.)

Write Register 4



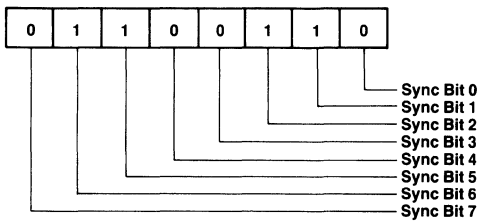
Set-up WR4 with basic protocol parameters: X1 clock and SDLC mode.

Write Register 3



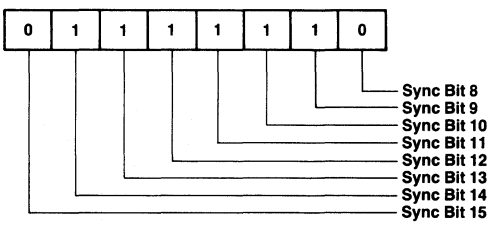
Set receive parameters to indicate the following: 8 bits per character, address search mode, and Rx CRC enabled. When the receiver is enabled the MPSC^2 will automatically enter the hunt phase.

Write Register 6



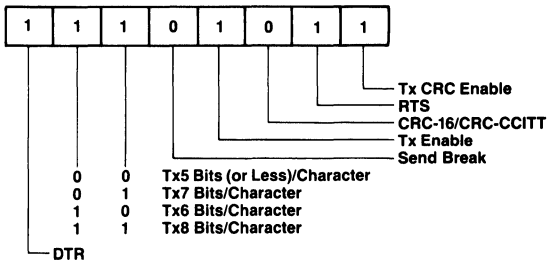
The MPSC^2 will compare the first nonflag byte to WR6. This register should be programmed with the secondary address field.

Write Register 7



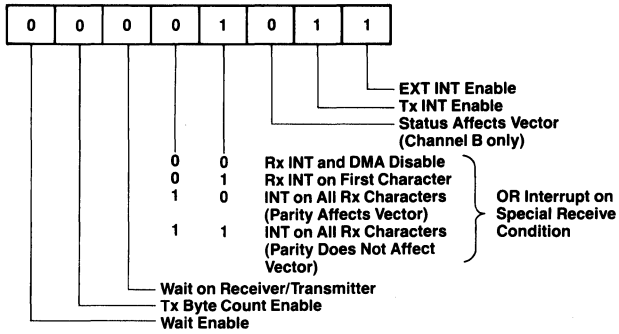
WR7 must be programmed with the SDLC flag (01111110) for comparison. When a flag is detected the sync hunt bit in RR0 will be reset.

Write Register 5



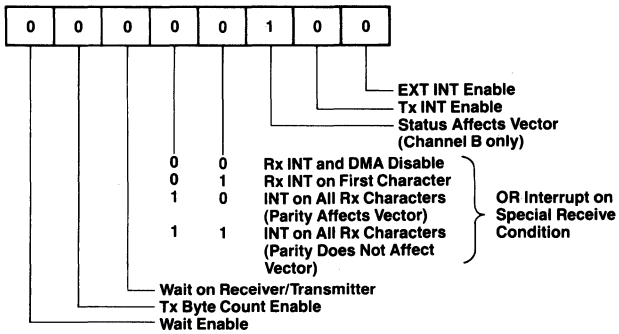
WR5 sets-up the transmit parameters to indicate the following: DTR, 8 bits, CCITT, RTS, and Tx CRC enabled.

Write Register 1A



WR1 is the interrupt control register. Special care should be taken when programming this register. Because this channel is running in the DMA mode Rx interrupt on the first received character was used. The MPSC^2 will interrupt the CPU on the first character only. In this example the CPU will enable the DMA controller when this interrupt occurs.

Write Register 1B



Because some of the commands are shared by both channels it may be necessary to enable a function in an alternate channel. In this case the status affects vector command is programmed in channel B for both channels.

SDLC Receive Mode Operations

When operating the MPSC^2 in the SDLC receive mode, the receiver can be in one of the following phases:

Hunt phase: The receiver compares the incoming data stream to the contents of control register 7 which is programmed with the SDLC flag pattern.

Address search phase: The receiver compares the incoming data stream to the contents of control register 6 which is programmed with the secondary station address.

Data phase: The information sequence is received.

Termination phase: The SDLC end of frame sequence has been received and the block check flag (CRC status flag) is now valid.

Once the MPSC^2 has been initialized and the receiver enabled the receiver will enter the hunt phase. When the leading flag is detected the receiver leaves the hunt phase (setting the external/status change flag) and, if the address search mode (CR3-D2) was selected during initialization, the receiver will enter the address search phase. At this point the receiver will compare the first 8-bit nonflag character received with the contents of control register 6 which contains the SDLC address sequence. If the two values match, or the received character is the global address (11111111), the receiver will enter the data phase and begin assembling characters. On the other hand, if an address match is not found the receiver will return to the hunt phase until another flag is found at which time the process will begin again. Since all SDLC messages are framed with flag characters, a message may be skipped by setting the enter hunt phase bit (D4) in control register 3. This may be particularly useful for extended addressing protocols.

Once the data phase has been entered the receiver will assemble characters according to the number of bits per character until the end of frame sequence is received. On encountering the end of frame sequence (SDLC flag) the receiver enters the termination phase and sets a special receive condition status bit in RR1 notifying the CPU that the CRC calculation is complete and the error status has been passed to status register 1 along with the residue code.

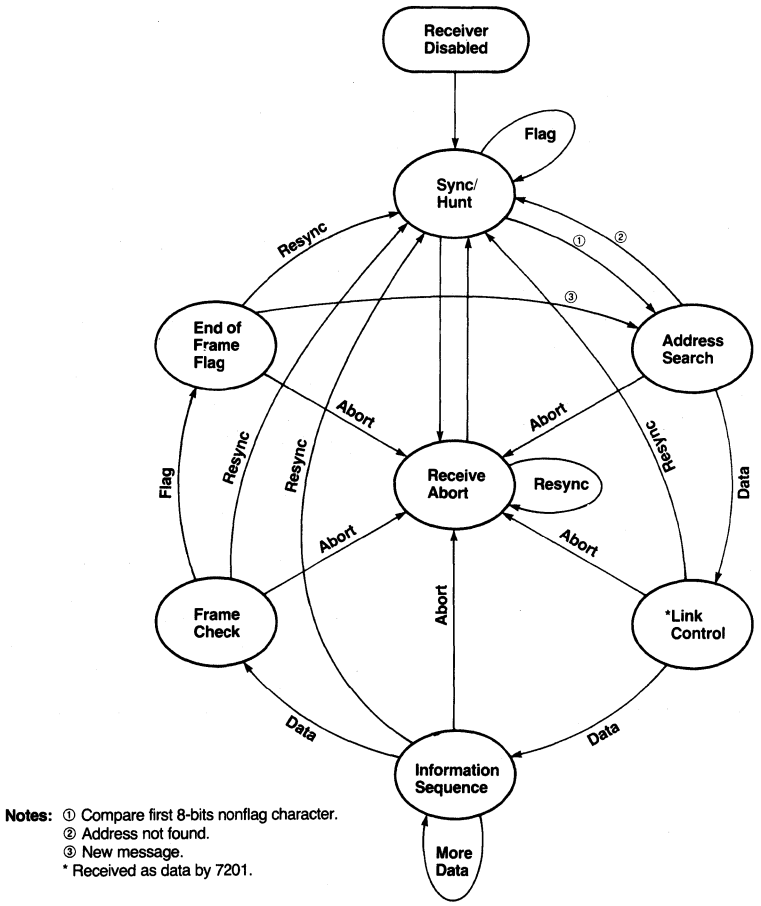


Figure 7. 7201 Bit-synchronous State Diagram (Receiver)

SDLC Transmit Operations

When operating in the SDLC mode the MPSC^2's transmitter (Tx) may be in any one of the following phases:

Disabled phase: The transmit enable is off (CR5, D3 = 0) or $\overline{CTS}$ is high when the auto enable function has been selected.

Idle phase: Flag characters are being sent.

Data phase: The information sequence is being sent.

Termination phase: When CRC is being sent (if used) and/or ending flag is being sent.

After selecting the desired initialization parameters, the transmitter will be in the disabled phase. The transmit data output (TxD) will be high or marking until the transmit enable bit has been set. Once set the transmitter enters the idle phase which will cause the transmitter to start sending flag characters. This phase will continue until the first byte of data (address field) has been written to the transmit buffer. After a byte of data has been loaded into the Tx data buffer and at least one flag sent, the MPSC^2 will enter the data phase. Entering this phase allows the data byte in the Tx buffer to be passed to the Tx shift register. This action will set the Tx buffer empty indicator notifying the CPU that another byte of data may be transferred to the Tx data buffer. This double buffering allows the processor one full character time from the Tx buffer indication to respond with the next

byte of data. It should be noted that it is the transfer of a data byte from the Tx data buffer to the shift register rather than the empty condition itself that causes the Tx buffer empty condition. The contents of the Tx shift register are sent least significant bit first after being passed through the zero insertion logic. The zero insertion logic inserts a 0 bit after detecting five contiguous ones in a data stream. This is done so that data can be distinguished from a flag or abort sequence.

The MPSC^2 is capable of generating an SDLC abort sequence by means of a command in CR0. Issuing this command causes at least 8 but less than 14 ones (abort and up to five ones allowed by the zero insertion logic from the previous character) to be transmitted. Issuing the abort command will destroy any data in the Tx shift register and the Tx data buffer. After a send abort command has been issued the transmitter will reenter the idle phase.

During the data phase, the transmitter may enter the termination phase for one of two reasons:

1. The processor is busy and is not able to provide the next data character in time within a message (transmit underrun).
2. The information field portion of the message is complete and it is time to send the frame check sequence.

The MPSC^2 automatically handles both of these conditions through a mechanism called the transmit underrun/EOM latch. The state of this latch can be read at any time from status register 0 (bit D6). Entering the termination phase sets the transmit underrun/EOM latch which, in turn, sets the external/status change flag indicating that the transmitter is sending the frame check sequence.

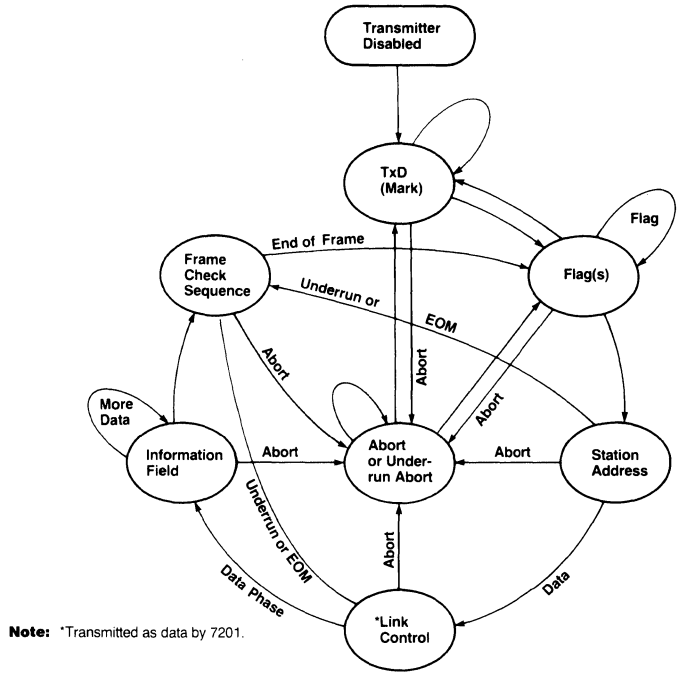


Figure 8. 7201 Bit-synchronous State Diagram (Transmit)

Servicing a Receive Frame

When data carrier detect ($\overline{DCD}$) becomes active it will cause an external/status change. This sends the CPU through the same procedure to determine the cause of the service request. In this example when $\overline{DCD}$ goes active, the receiver is enabled. At this point the receiver enters the hunt phase. If the receiver input RxD is at a mark (high), an external/status interrupt will occur indicating an SDLC abort has been received. The next interrupt that occurs is the sync/hunt status change indicating the reception of an SDLC flag. This is also an external status interrupt and is handled by the external status interrupt routine. The 7201 is now in the address search mode. The receiver will compare this first nonflag character to the one programmed in the address register. When an address match is made the receiver will start assembling data in the receive shift register. From this point on, the 7201 will issue a DMA request for each character received in the receive FIFO. The next interrupt to occur (providing no abort sequence was received) is a special receive condition interrupt indicating the ending flag has been received.

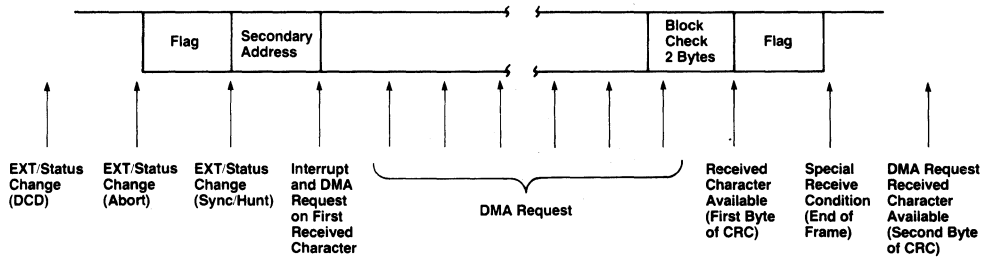


Figure 9. Receive Frame Interrupts and DMA Requests

Servicing a Transmit Frame

In reference to the transmit frame the first interrupt is initiated by a clear to send ($\overline{CTS}$) going active. This causes the CPU to poll the 7201 and jump to the external status interrupt routine. The CPU determines the cause of the service request to be $\overline{CTS}$ and enables the transmitter. This routine also primes the transmitter by loading the first byte (secondary address) to the transmit buffer. From this point on the 7201 will issue a DMA request (DRQn will become active) each time the transmit buffer becomes empty. Sometime later (depending on the byte count of the DMA controller) the 7201 will again make another external status service request that indicates transmit underrun or end of message status. The CPU determines from the DMA controller's byte count that the end of message has occurred (byte count = 0). The 7201 automatically sends two bytes of CRC when the transmit underrun (end of message) occurs followed by a flag(s).

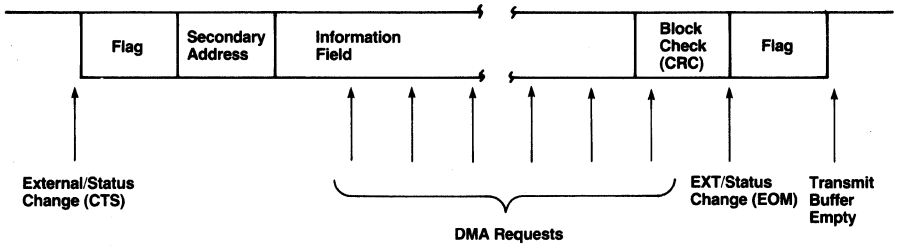
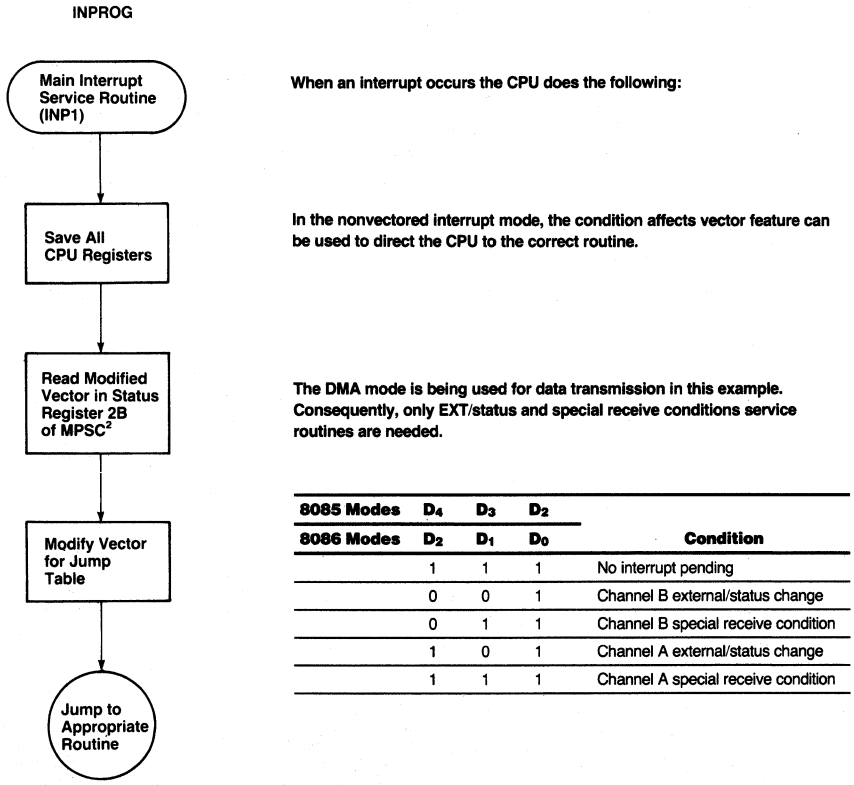


Figure 10. Transmit Frame Interrupts and DMA Requests

Interrupts Service Routine (ISR)

The MPSC^2 offers an elaborate interrupt scheme to provide fast interrupt response in real-time applications. The interrupt structure of the MPSC^2 allows it to work with the 8080/85 or 8088/86 with or without an 8259A interrupt controller. Channel B control register 2 (CR2) contains the interrupt vector that points to an interrupt service routine when operating in the nonvector mode. To service requests for both channels and to eliminate the necessity of analyzing four status registers to determine the cause of an interrupt, the status affects vector can be used (CR1, D2 channel B only). Enabling this option allows the internal interrupt controller to modify the interrupt vector in status register 2B. The interrupt controller will also assign priorities to the various interrupt conditions.



8085 Modes	D_4	D_3	D_2	
8086 Modes	D_2	D_1	D_0	Condition
	1	1	1	No interrupt pending
	0	0	1	Channel B external/status change
	0	1	1	Channel B special receive condition
	1	0	1	Channel A external/status change
	1	1	1	Channel A special receive condition

Figure 11. Flowchart Example

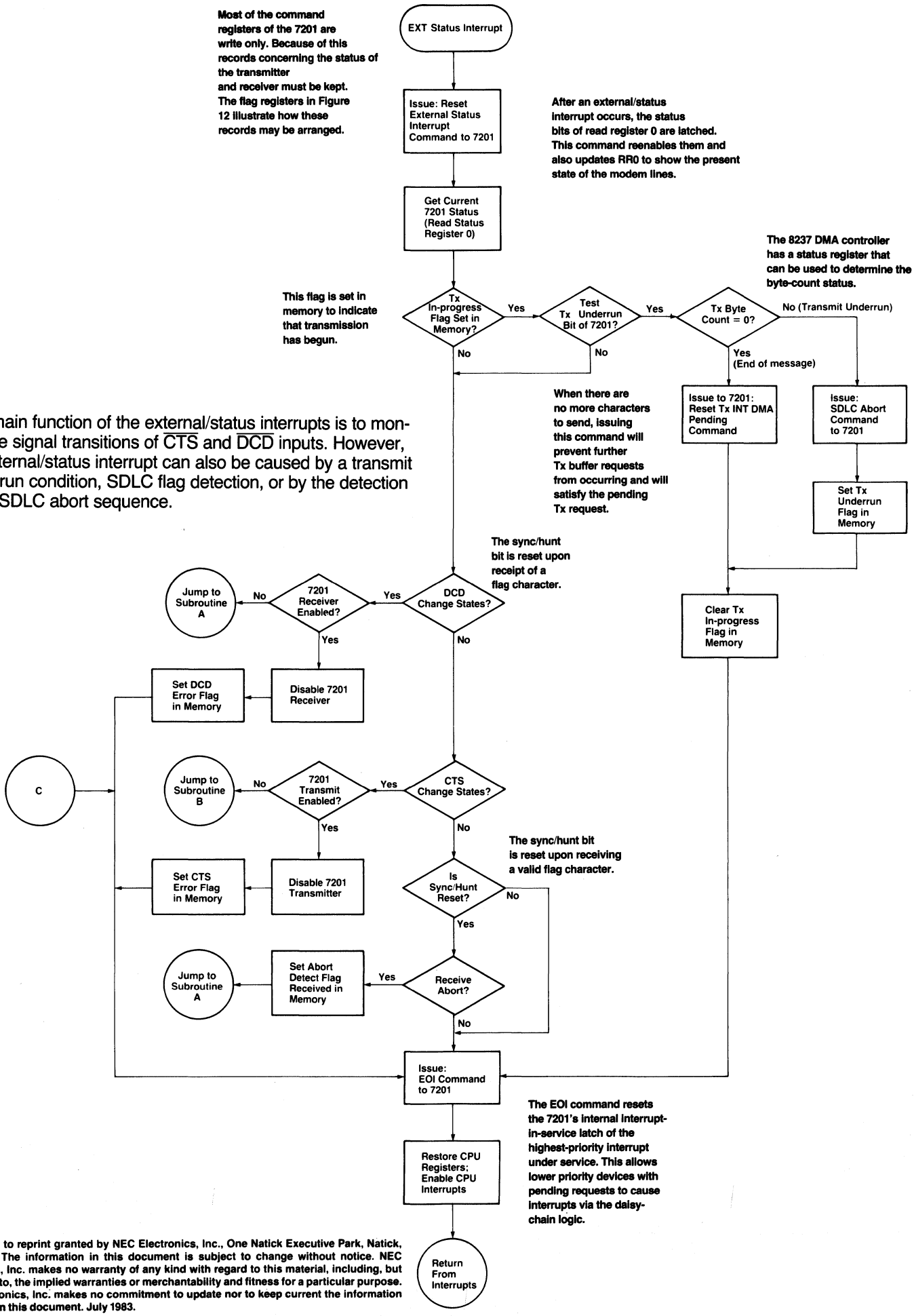


Figure 11. Flowchart Example (Cont.)

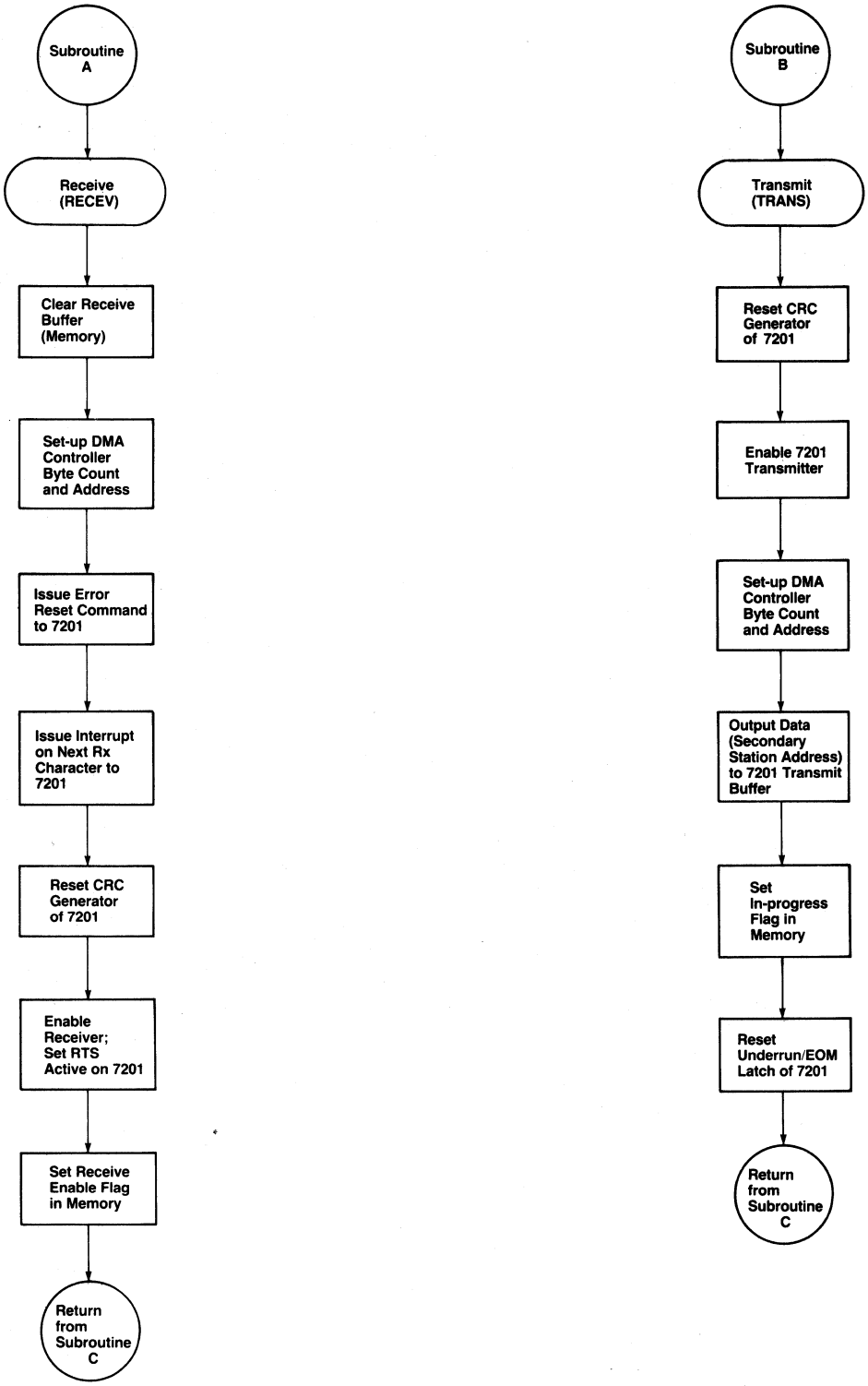


Figure 11. Flowchart Example (Cont.)

Special Receive Conditions

The special receive condition interrupt is not a separate interrupt mode as such, but rather an extension of the receive character modes. Special receive conditions must be enabled by selecting either an Rx interrupt on a first character mode or an interrupt on an all character mode.

In the SDLC example, a special receive condition can be caused by a receive overrun or end of frame detection. Since the receive overrun status bit is latched in RR1 at the time of the FIFO overrun, the error status reflects an error in the current word of the receive FIFO. This is, of course, in addition to any errors received since the last error reset command. The end of frame (EOF) status bit indicates that a valid ending flag has been received and that the CRC error status and residue codes are valid. A special receive condition must be reset by the error reset command which resides in CR0.

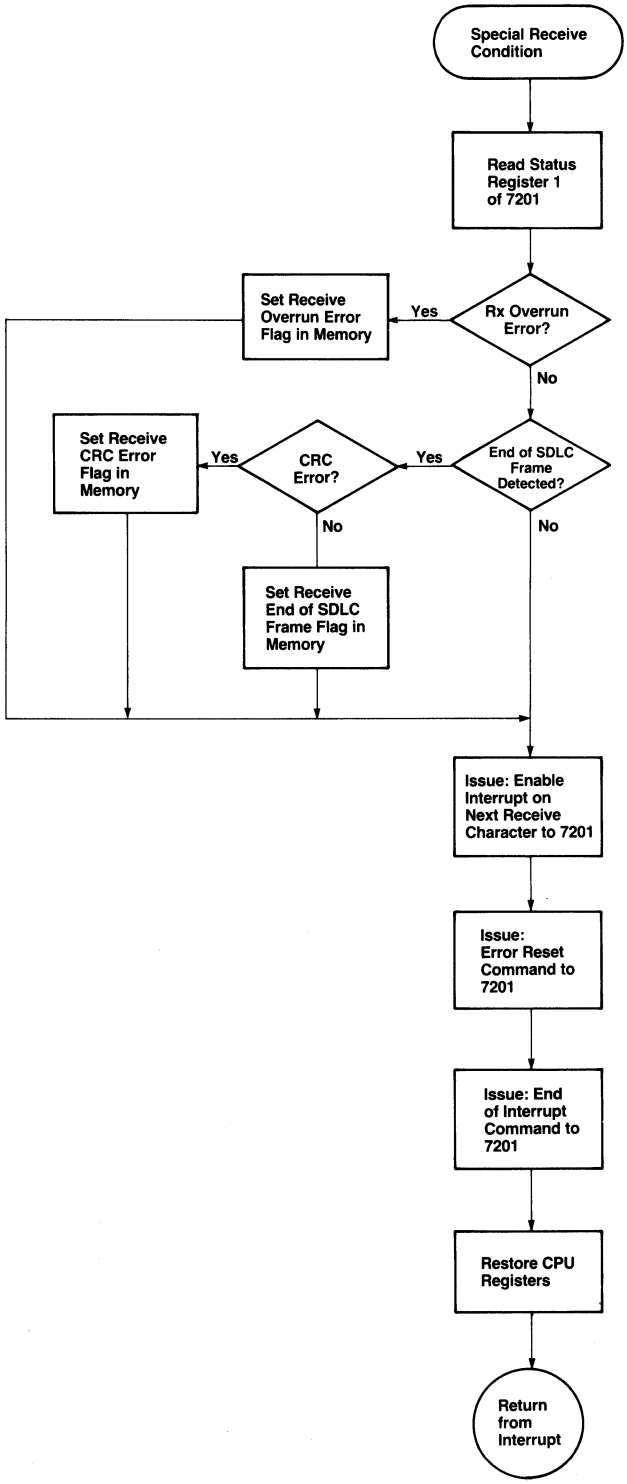


Figure 11. Flowchart Example (Cont.)

NEC
NEC Electronics, Inc.

One Natick Executive Park
Natick, Massachusetts 01760
TEL 617-655-8833
TWX 710-386-2110

	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Flag Register 0	Receive Abort Flag	Transmit Underrun End of Message Flag	Clear to Send Flag (CTS)	Sync/ Hunt Flag	Data Carrier Detect Flag	Not Used	Not Used	Not Used
	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Transmit Flag Register	Transmit In-progress Flag (TxBEG)	Transmit Underrun Flag	End of Transmit Flag	Transmit Enabled Flag	Clear to Send Error Flag	Not Used	Not Used	Tx Error
	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Receive Flag Registers	Receive Overrun Flag	End of SDLC Frame Flag	Receive CRC Error Flag	Data Carrier Detect Error Flag	Receive Abort Error Flag	Receive Enabled Flag	Not Used	Rx Error

Figure 12. Memory Flag Registers

Printed in U.S.A.
AN28-6-83-8K

PRODUCT

B·U·L·L·E·T·I·N

COMPLEX PERIPHERALS

PB# 30
November 1982

QUESTIONS ON μPD7201 MPSC²

The following document answers a few of the most asked questions on the μPD7201 MULTI-PROTOCOL SERIAL CONTROLLER (MPSC²). These questions and answers should help you when talking to customers. Any additional questions you may have should be sent to Randy Spears and he will answer and publish them on a monthly basis.

Q: *Could you tell me how the transmit end of message termination is done, using the SDLC protocol in the DMA mode?*

A: The following is an example of a typical termination of a transmit SDLC frame. The termination of an SDLC frame occurs because of either a transmit underrun (for some reason the DMA controller did not send the MPSC² a byte of data in time) or because the End of Message occurred (the DMA controller has reached terminal count).

Example:

The MPSC² raises one of its two transmit DMA Request pins (DRQTxA pin 11 or DRQTxB pin 30) notifying the DMA controller that one of the transmit buffers has become empty. The DMA controller does not respond and causes a transmit underrun condition. The MPSC² satisifies the transmit buffer empty condition by sending two 8-bit CRC characters. While the first CRC character is being sent the MPSC² sets the external/status interrupt latch causing the external interrupt pin ($\overline{INT}$, pin 28) to go low (if external status interrupts are enabled).

The CPU recognizes the transmit Underrun End of Message status and determines from its internal program status whether or not the underrun condition occurred because of a data underrun or End of Message situation. If it is determined to be a Transmit Underrun, the CPU will immediately issue a Send Abort command. The Send Abort command will destroy whatever data, CRC, or flag is being sent and generates a sequence of 8 to 13 ones. However, if the underrun condition is determined to be caused by the end of a message, the

NEC
NEC Electronics U.S.A. Inc.
Microcomputer Division One Natick Executive Park, Natick, Massachusetts 01760 TEL 617-655-8833 TWX 710-386-2110

CPU can either reinitialize the DMA controller for a new message and transfer the first byte of data to the MPSC^2 (this data byte will satisfy the pending DMA request) or issue a Transmit Interrupt DMA Pending command which will inhibit the MPSC^2 from making any more requests.

Q: *From the above explanation, is there a problem if Channel B's transmitter has a pending DMA request when an underrun condition occurs on Channel A?*

A: Yes, the problem occurs when the transmitter on Channel A underruns with an active DMA request pending on Channel B (B has a lower priority). The following explanation should help you understand the problem.

The MPSC^2 will accept DMA requests, from the internal elements of both Channels A and B at the following times:

1. During system clock low, when $\overline{HAI}$ is inactive (high).

2. During the leading (falling) edge of $\overline{RD}$ or $\overline{WR}$ when $\overline{HAI}$ is active (low).

If the DMA request is internally generated during these acceptable times, they are latched into the DMA Request Register (DRR) and the corresponding external DMA request line will become active immediately, independently of priority order, and regardless of any higher priority DMA request that may have occurred previously. The MPSC^2 also sets a DMA In-Service (DIS) latch for each internal DMA request made. When a DMA cycle has been accepted by the DMA controller and CPU (which the MPSC^2 will determine by receiving $\overline{HAI}$, pin 26, low), the MPSC^2 will determine which of the DIS latches will be cleared and will remain cleared until after the DMA transfer has taken place ($\overline{RD}$ or $\overline{WR}$ going low). With the highest priority DIS latch set, and all other DIS latches frozen, in the cleared state, the MPSC^2 will disconnect internally C/$\overline{D}$, B/$\overline{A}$, and $\overline{CS}$ from the external bus and set its internal bus logic to $\overline{CS}$ = 0, C/$\overline{D}$ = 0, and, depending on if the highest priority DIS latch was set for Channel A or Channel B, would set B/$\overline{A}$ accordingly. It is for this reason that the priority problem occurs. If the underrun condition occurs on Channel A for any reason, the MPSC^2 has no way of knowing that there will be no more data transferred by the DMA controller to that channel, thus leaving Channel A's transmit DIS latch active. If the transmitter on Channel B has an active DIS latch and gets serviced by the DMA controller, the data that was meant for Channel B will inadvertently go to the

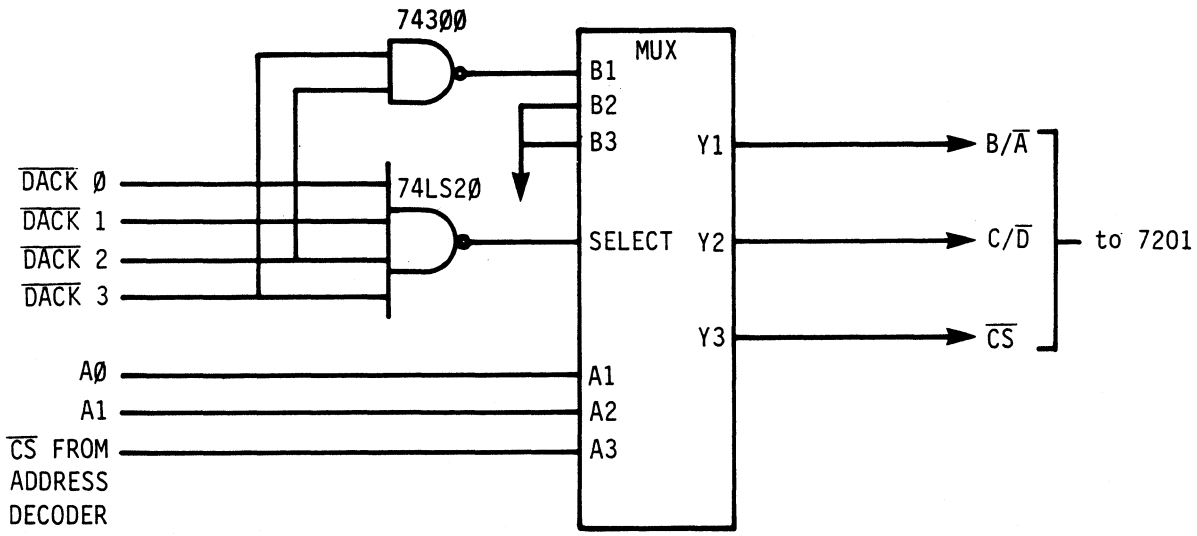
transmitter on Channel A because it has the highest priority DIS latch set.

Q: What do you suggest for correcting the above problem?

A: The above problem can be corrected by allowing the DMA controller to determine the DMA request priorities instead of the MPSC^2. Figure 1 shows a priority resolution scheme that allows the DMA controller to determine the highest priority channel of the MPSC^2. The $\overline{DACK}$ signals from the DMA controller are encoded and multiplexed with A1 and $\overline{CS}$ to determine what channel the DMA controller is addressing. If the MUX is used, $\overline{HAI}$ must be tied high (inactive state) to enable the μPD7201 bus logic C/$\overline{D}$, B/$\overline{A}$, and $\overline{CS}$. There is one timing parameter that is affected when $\overline{HAI}$ is high (inactive) during an active DMA request. TCQ (request hold time from $\overline{WR}$) is longer than specified (150ns), resulting in a TCQ of approximately 300ns to 600ns. Care must be taken when choosing the DMA controller mode, so that the DMA controller does not respond twice to the same DMA request. In the case of the μPD8237 DMA controller, the Single Byte Transfer mode should be used. This mode takes the DMA controller a longer period of time to test the DMA request lines without greatly affecting the throughput.

Figure 1

Multiplex Scheme for Determining External MPSC^2 Priorities



Q: *If a read operation occurs (such as a status read) during an active receive DMA request when $\overline{HAI}$ is tied to the inactive state, will this cause any improper result from the MPSC²?*

A: Yes, if the $\overline{CS}$ signal is activated for a data read operation for a channel that has an active receive DMA request, the external DMA request will be cleared leaving the internal DMA request pending. To avoid this problem, the following should be remembered:

1) When operating in the DMA mode, status should not be read during a receive data frame.

2) A hardware solution is shown in Figure 2 which will keep the DMA request active until the DMA controller has acknowledged the DMA request.

Q: *How is the Auto Enable feature used in the Asynchronous and Synchronous modes?*

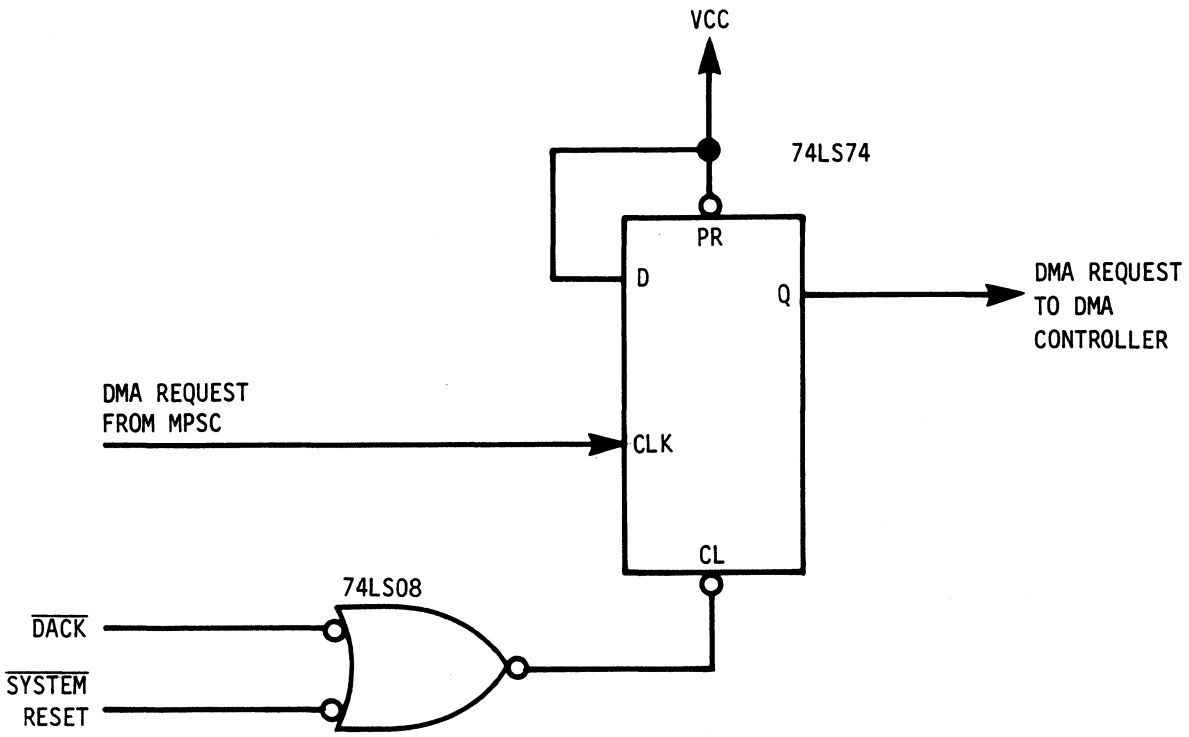
A: Let me explain how the Auto Enable function works, and its advantages using different protocols.

If Auto Enable is selected, $\overline{DCD}$ and $\overline{CTS}$ become the receiver and transmitter enables, respectively.

When using modems this becomes very useful for automatic response and telephone answering. When Auto Enable is used in the Asynchronous mode, the transmit buffer can be loaded with a character which will not be sent until $\overline{CTS}$ on that channel becomes active. If the interrupt or DMA mode is selected, after $\overline{CTS}$ becomes active the MPSC² will automatically generate a Transmit Interrupt or DMA Request respectively, and transmit the character that was previously loaded into the transmit buffer.

In the Synchronous mode the Auto Enable feature works the same as the Asynchronous mode, except that a character should not be loaded into the buffer before $\overline{CTS}$ becomes active. Doing so will cause the MPSC² to send the first character instead of the beginning flag.

Figure 2 DMA Request Latch



The following are corrections to the μPD7201 Technical Manual.

1) Page 32, paragraph 2 reads:

   At the end of the second $\overline{\text{INTA}}$ pulse, the $\overline{\text{INT}}$...

   It should read:

   At the beginning of the second $\overline{\text{INTA}}$ pulse, the $\overline{\text{INT}}$...

2) Page 85, Figure 6.4:

   The INTA generator for the Z-80 or μPD780 art work is incomplete and should be as in Figure 3 of this document.

μPD780-μPD7201 INTERFACE CIRCUIT DESCRIPTION

In order to use the μPD780 with the μPD7201, two items must be taken into account. First, three $\overline{\text{INTA}}$ pulses must be generated and second, memory must be inhibited from responding to the memory read cycles which immediately follow the Interrupt Acknowledge cycle. Both of these may be accomplished with the circuit shown in Figure 3. The J-K flip-flop is set by the occurrence of an IACK ($\overline{\text{MI}}$ and $\overline{\text{IORQ}}$ together) and reset by any WRITE operation. When the flip-flop is set, $\overline{\text{READ}}$ pulses from the CPU are inhibited from reaching the system (memory read is inhibited) and are steered instead to the $\overline{\text{INTA}}$ line, thus producing the second and third $\overline{\text{INTA}}$ pulses. The program counter is not incremented during the $\overline{\text{INTA}}$ cycle and therefore contains the correct address when the Return instruction is encountered. (Note - the μPD780 is operating in mode 0).

Figure 3 μPD780-μPD7201 Interface Schematic