Using the 8273 SDLC/HDLC Protocol Controller

Contents

INTRODUCTION

The Intel 8273 is a Data Communications Protocol Controller designed for use in systems utilizing either SDLC or HDLC (Synchronous or High-Level Data Link Control) protocols. In addition to the usual features such as full duplex operation, automatic Frame Check Sequence generation and checking, automatic zero bit insertion and deletion, and TTL compatibility found on other single component SDLC controllers; the 8273 features a frame level command structure, a digital phase locked loop, SDLC loop operation, and diagnostics.

The frame level command structure is made possible by the 8273's unique internal dual processor architecture. A high-speed bit processor handles the serial data manipulations and character recognition. A byte processor implements the frame level commands. These dual processors allow the 8273 to control the necessary byte-by-byte operation of the data channel with a minimum of CPU (Central Processing Unit) intervention. For the user this means the CPU has time to take on additional tasks. The digital phase locked loop (DPLL) provides a means of clock recovery from the received data stream on-chip. This feature, along with the frame level commands, makes SDLC loop operation extremely simple and flexible. Diagnostics in the form of both data and clock loopback are available to simplify board debug and link testing. The 8273 is a dedicated function peripheral in the MCS-80/85 Microcomputer family and as such, it interfaces to the 8080/8085 system with a minimum of external hardware.

This application note explains the 8273 as a component and shows its use in a generalized loop configuration and a typical 8085 system. The 8085 system was used to verify the SDLC operation of the 8273 on an actual IBM SDLC data communications link.

The first section of this application note presents an overview of the SDLC/HDLC protocols. It is fairly tutorial in nature and may be skipped by the more knowledgeable reader. The second section describes the 8273 from a functional standpoint with explanation of the block diagram. The software aspects of the 8273, including command examples, are discussed in the third section. The fourth and fifth sections discuss a loop SDLC configuration and the 8085 system respectively.

SDLC/HDLC OVERVIEW

SDLC is a protocol for managing the flow of information on a data communications link. In other words, SDLC can be thought of as an envelope — addressed, stamped, and containing an s.a.s.e. — in which information is transferred from location to location on a data communications link. (Please note that while SDLC is discussed specifically, all comments also apply to HDLC except where noted.) The link may be either point-to-point or multi-point, with the point-to-point configuration being either switched or nonswitched. The information flow may use either full or half duplex exchanges. With this many configurations supported, it is difficult to find a synchronous data communications application where SDLC would not be appropriate.

Aside from supporting a large number of configurations, SDLC offers the potential of a 2× increase in throughput over the presently most prevalent protocol: Bi-Sync. This performance increase is primarily due to two characteristics of SDLC: full duplex operation and the implied acknowledgement of transferred information. The performance increase due to full duplex operation is fairly obvious since, in SDLC, both stations can communicate simultaneously. Bi-Sync supports only half-duplex (two-way alternate) communication. The increase from implied acknowledgement arises from the fact that a station using SDLC may acknowledge previously received information while transmitting different information. Up to 7 messages may be outstanding before an acknowledgement is required. These messages may be acknowledged as a block rather than singly. In Bi-Sync, acknowledgements are unique messages that may not be included with messages containing information and each information message requires a separate acknowledgement. Thus the line efficiency of SDLC is superior to Bi-Sync. On a higher level, the potential of a 2× increase in performance means lower cost per unit of information transferred. Notice that the increase is not due to higher data link speeds (SDLC is actually speed independent), but simply through better line utilization.

Getting down to the more salient characteristics of SDLC; the basic unit of information on an SDLC link is that of the frame. The frame format is shown in Figure 1. Five fields comprise each frame: flag, address, control, information, and frame check sequence. The flag fields (F) form the boundary of the frame and all other fields are positionally related to one of the two flags. All frames start with an opening flag and end with a closing flag. Flags are used for frame synchronization. They also may serve as time-fill characters between frames. (There are no intraframe time-fill characters in SDLC as there are in Bi-Sync.) The opening flag serves as a reference point for the address (A) and control (C) fields. The frame check sequence (FCS) is referenced from the closing flag. All flags have the binary configuration 01111110 (7EH).

SDLC is a bit-oriented protocol, that is, the receiving station must be able to recognize a flag (or any other special character) at any time, not just on an 8-bit boundary. This, of course, implies that a frame may be N-bits in length. (The vast majority of applications tend to use frames which are multiples of 8 bits long, however.)

OPENING FLAG	ADDRESS FIELD (A)	CONTROL FIELD (C)	INFORMATION FIELD (I)	FRAME CHECK SEQUENCE (FCS)	CLOSING FLAG
01111110	8 BITS	8 BITS	ANY LENGTH 0 TO N BITS	16 BITS	01111110

Figure 1. SDLC Frame Format

The fact that the flag has a unique binary pattern would seem to limit the contents of the frame since a flag pattern might inadvertently occur within the frame. This would cause the receiver to think the closing flag was received, invalidating the frame. SDLC handles this situation through a technique called zero bit insertion. This techniques specifies that within a frame a binary 0 be inserted by the transmitter after any succession of five contiguous binary 1s. Thus, no pattern of 01111110 is ever transmitted by chance. On the receiving end, after the opening flag is detected, the receiver removes any 0 following 5 consecutive 1s. The inserted and deleted 0s are not counted for error determination.

Before discussing the address field, an explanation of the roles of an SDLC station is in order. SDLC specifies two types of stations: primary and secondary. The primary is the control station for the data link and thus has responsibility of the overall network. There is only one predetermined primary station, all other stations on the link assume the secondary station role. In general, a secondary station speaks only when spoken to. In other words, the primary polls the secondaries for responses. In order to specify a specific secondary, each secondary is assigned a unique 8-bit address. It is this address that is used in the frame's address field.

When the primary transmits a frame to a specific secondary, the address field contains the secondary's address. When responding, the secondary uses its own address in the address field. The primary is never identified. This ensures that the primary knows which of many secondaries is responding since the primary may have many messages outstanding at various secondary stations. In addition to the specific secondary address, an address common to all secondaries may be used for various purposes. (An all 1s address field is usually used for this "All Parties" address.) Even though the primary may use this common address, the secondaries are expected to respond with their unique address. The address field is always the first 8 bits following the opening flag.

The 8 bits following the address field form the control field. The control field embodies the link-level control of SDLC. A detailed explanation of the commands and responses contained in this field is beyond the scope of this application note. Suffice it to say that it is in the control field that the implied acknowledgement is carried out through the use of frame sequence numbers. None of the currently available SDLC single chip controllers utilize the control field. They simply pass it to the processor for analysis. Readers wishing a more detailed explanation of the control field, or of SDLC in general, should consult the IBM documents referenced on the front page overleaf.

In some types of frames, an information field follows the control field. Frames used strictly for link management may or may not contain one. When an information field is used, it is unrestricted in both content and length. This code transparency is made possible because of the zero bit insertion mentioned earlier and the bit-oriented nature of SDLC. Even main memory core dumps may be transmitted because of this capability. This feature is unique to bit-oriented protocols. Like the control field, the information field is not interpreted by the SDLC device; it is merely transferred to and from memory to be operated on and interpreted by the processor.

The final field is the frame check sequence (FCS). The FCS is the 16 bits immediately preceding the closing flag. This 16-bit field is used for error detection through a Cyclic Redundancy Checkword (CRC). The 16-bit transmitted CRC is the complement of the remainder obtained when the A, C, and I fields are "divided" by a generating polynomial. The receiver accumulates the A, C, and I fields and also the FCS into its internal CRC register. At the closing flag, this register contains one particular number for an error-free reception. If this number is not obtained, the frame was received in error and should be discarded. Discarding the frame causes the station to not update its frame sequence numbering. This results in a retransmission after the station sends an acknowledgement from previous frames. [Unlike all other fields, the FCS is transmitted MSB (Most Significant Bit) first. The A, C, and I fields are transmitted LSB (Least Significant Bit) first.] The details of how the FCS is generated and checked is beyond the scope of this application note and since all single component SDLC controllers handle this function automatically, it is usually sufficient to know only that an error has or has not occurred. The IBM documents contain more detailed information for those readers desiring it.

The closing flag terminates the frame. When the closing flag is received, the receiver knows that the preceding 16 bits constitute the FCS and that any bits between the control field and the FCS constitute the information field.

SDLC does not support an interframe time-fill character such as the SYN character in Bi-Sync. If an unusual condition occurs while transmitting, such as data is not available in time from memory or CTS (Clear-to-Send) is lost from the modem, the transmitter aborts the frame by sending an Abort character to notify the receiver to invalidate the frame. The Abort character consists of eight contiguous 1s sent without zero bit insertion. Intraframe time-fill consists of either flags, Abort characters, or any combination of the two.

While the Abort character protects the receiver from transmitted errors, errors introduced by the transmission medium are discovered at the receiver through the FCS check and a check for invalid frames. Invalid frames are those which are not bounded by flags or are too short, that is, less than 32 bits between flags. All invalid frames are ignored by the receiver.

Although SDLC is a synchronous protocol, it provides an optional feature that allows its use on basically asynchronous data links — NRZI (Non-Return-to-Zero-Inverted) coding. NRZI coding specifies that the signal condition does not change for transmitting a binary 1, while a binary 0 causes a change of state. Figure 2 illustrates NRZI coding compared to the normal NRZ. NRZI coding guarantees that an active line will have a transition at least every 5-bit times; long strings of zeroes cause a transition every bit time, while long strings of 1s are broken up by zero bit insertion. Since asynchronous operation requires that the receiver sampling clock be derived from the received data, NRZI encoding plus zero bit insertion make the design of clock recovery circuitry easier.

All of the previous discussion has applied to SDLC on either point-to-point or multi-point data networks, SDLC (but not HDLC) also includes specification for a loop configuration. Figure 3 compares these three configurations. IBM uses this loop configuration in its 3650 Retail Store System. It consists of a single loop controller station with one or more down-loop secondary stations. Communications on a loop rely on the secondary stations repeating a received message down loop with a delay of one bit time. The reason for the one bit delay will be evident shortly.

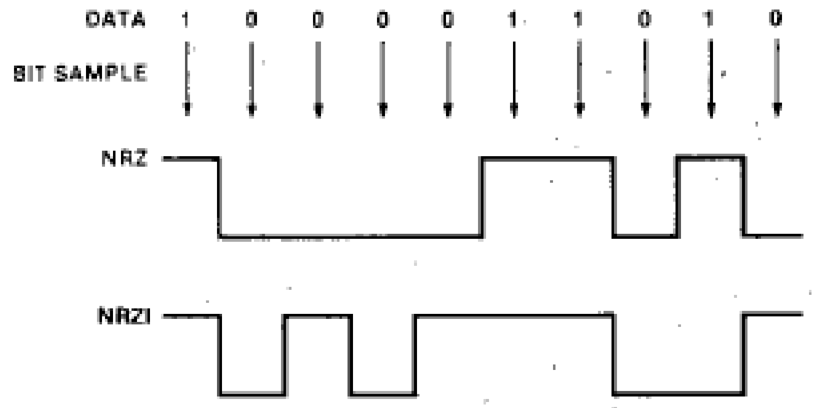

Figure 2. NRZI vs NRZ Encoding

Loop operation defines a new special character: the EOP (End-of-Poll) character which consists of a 0 followed by 7 contiguous, non-zero bit inserted, ones. After the loop controller transmits a message, it idles the line (sends all 1s). The final zero of the closing flag plus the first 7 1s of the idle form an EOP character. While repeating, the secondaries monitor their incoming line for an EOP character. When an EOP is detected, the secondary checks to see if it has a message to transmit. If it does, it changes the seventh 1 to a 0 (the one bit delay allows time for this) and repeats the modified EOP (now alias flag). After this flag is transmitted, the secondary terminates its repeater function and inserts its message (with multiple preceding flags if necessary). After the closing flag, the secondary resumes its one bit delay repeater function. Notice that the final zero of the secondary's closing flag plus the repeated 1s from the controller form an EOP for the next down-loop secondary, allowing it to insert a message if it desires.

One might wonder if the secondary missed any messages from the controller while it was inserting its own message. It does not. Loop operation is basically half-duplex. The controller waits until it receives an EOP before it transmits its next message. The controller's reception of the EOP signifies that the original message has propagated around the loop followed by any messages inserted by the secondaries. Notice that secondaries cannot communicate with one another directly, all secondary-to-secondary communication takes place by way of the controller.

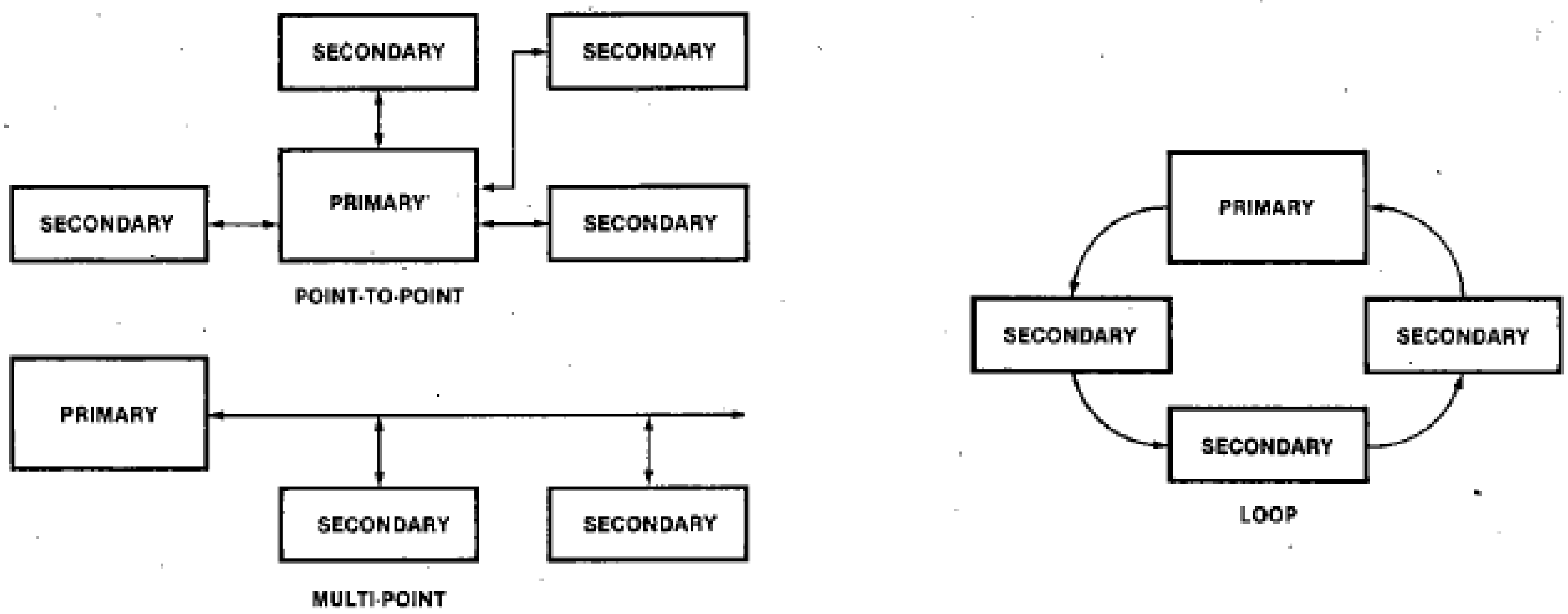

Figure 3. Network Configurations

Loop protocol does not utilize the normal Abort character. Instead, an abort is accomplished by simply transmitting a flag character. Down loop, the receiver sees the abort as a frame which is either too short (if the abort occurred early in the frame) or one with an FCS error. Either results in a discarded frame. For more details on loop operation, please refer to the IBM documents referenced earlier.

Another protocol very similar to SDLC which the 8273 supports is HDLC (High-Level Data Link Control). There are only three basic differences between the two: HDLC offers extended address and control fields, and the HLDC Abort character is 7 contiguous 1s as opposed to SDLC's 8 contiguous 1s.

Extended addressing, beyond the 256 unique addresses possible with SDLC, is provided by using the address field's least significant bit as the extended address modifier. The receiver examines this bit to determine if the octet should be interpreted as the final address octet. As long as the bit is 0, the octet that contains it is considered an extended address. The first time the bit is a 1, the receiver interprets that octet as the final address octet. Thus the address field may be extended to any number of octets. Extended addressing is illustrated in Figure 4a.

A similar technique is used to extend the control field although the extension is limited to only one extra control octet. Figure 4b illustrates control field extension.

Those readers not yet asleep may have noticed the similarity between the SDLC loop EOP character (a 0 followed by 7 1s) and the HDLC Abort (7 1s). This possible incompatibility is neatly handled by the HDLC protocol not specifying a loop configuration.

This completes our brief discussion of the SDLC/HDLC protocols. Now let us turn to the 8273 in particular and discuss its hardware aspects through an explanation of the block diagram and generalized system schematics.

Figure 4a

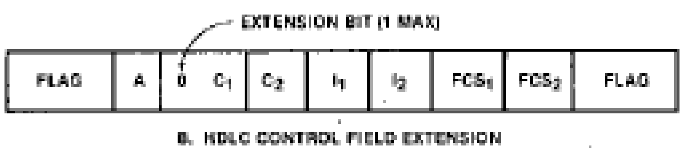

Figure 4b

BASIC 8273 OPERATION

It will be helpful for the following discussions to have some idea of the basic operation of the 8273. Each operation, whether it is a frame transmission, reception or port read, etc., is comprised of three phases: the Command, Execution, and Result phases. Figure 5 shows the sequence of these phases. As an illustration of this sequence, let us look at the transmit operation.

Figure 5. 8273 Operational Phases

When the CPU decides it is time to transmit a frame, the Command phase is entered by the CPU issuing a Transmit Frame command to the 8273. It is not sufficient to just instruct the 8273 to transmit. The frame level command structure sometimes requires more information such as frame length and address and control field content. Once this additional information is supplied, the Command phase is complete and the Execution phase is entered. It is during the Execution phase that the actual operation, in this case a frame transmission, takes place. The 8273 transmits the opening flag, A and C fields, the specified number of I field bytes, inserts the FCS, and closes with the closing flag. Once the closing flag is transmitted, the 8273 leaves the Execution phase and begins the Result phase. During the Result phase the 8273 notifies the CPU of the outcome of the command by supplying interrupt results. In this case, the results would be either that the frame is complete or that some error condition causes the transmission to be aborted. Once the CPU reads all of the results (there is only one for the Transmit Frame command), the Result phase and consequently the operation, is complete. Now that we have a general feeling for the operation of the 8273, let us discuss the 8273 in detail.

HARDWARE ASPECTS OF THE 8273

The 8273 block diagram is shown in Figure 6. It consists of two major interfaces: the CPU module interface and the modem interface. Let's discuss each interface separately.

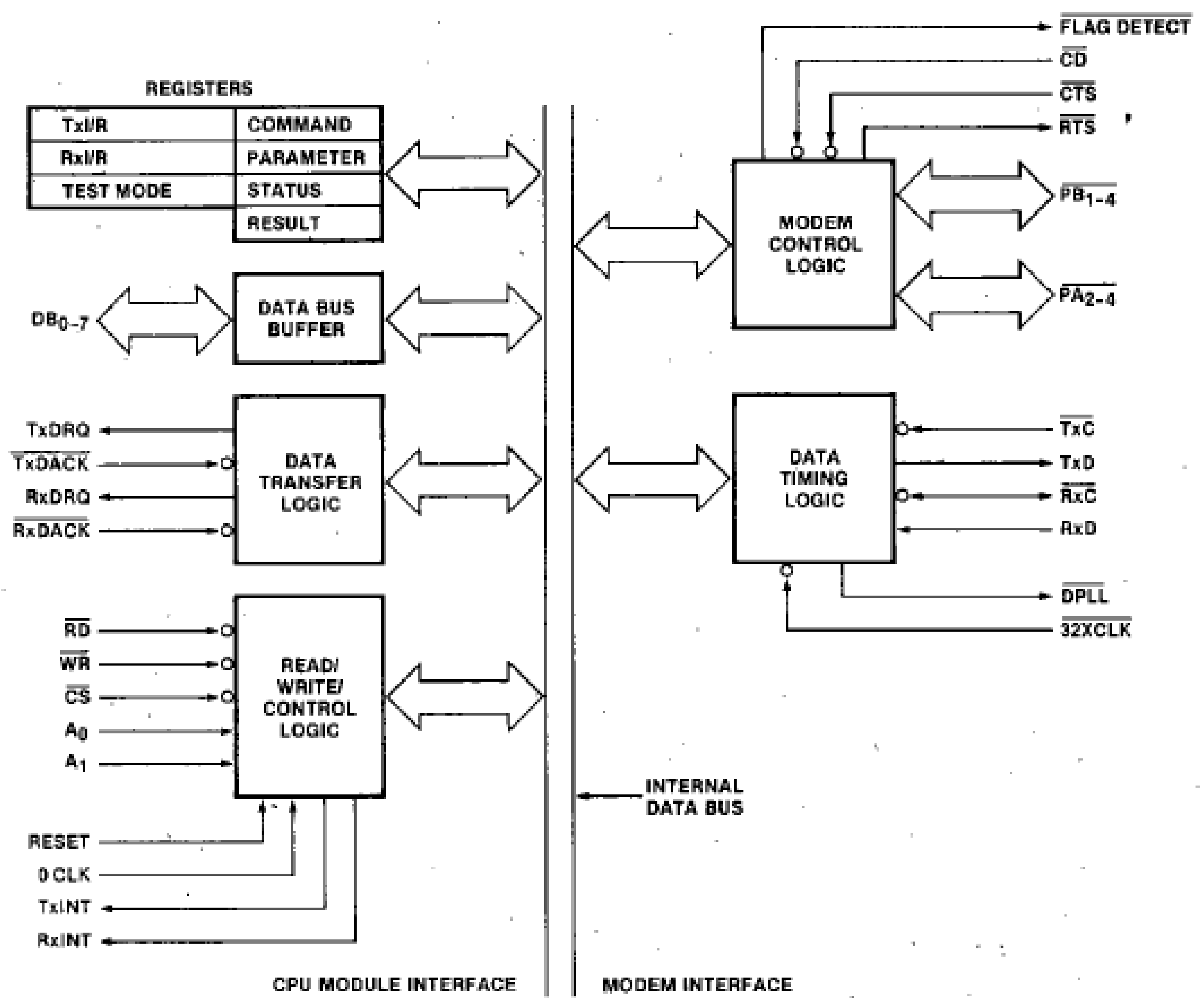

Figure 6. 8273 Block Diagram

CPU Interface

The CPU interface consists of four major blocks: Control/Read/Write logic (C/R/W), internal registers, data transfer logic, and data bus buffers.

The CPU module utilizes the C/R/W logic to issue commands to the 8273. Once the 8273 receives a command and executes it, it returns the results (good/bad completion) of the command by way of the C/R/W logic. The C/R/W logic is supported by seven registers which are addressed via the A_0, A_1, $\overline{RD}$, and $\overline{WR}$ signals, in addition to $\overline{CS}$. The A_0 and A_1 signals are generally derived from the two low order bits of the CPU module address bus while $\overline{RD}$ and $\overline{WR}$ are the normal I/O Read and Write signals found on the system control bus. Figure 7 shows the address of each register using the C/R/W logic. The function of each register is defined as follows:

ADDRESS INPUTS		CONTROL INPUTS	
A_1	A_0	$\overline{CS} \cdot \overline{RD}$	$\overline{CS} \cdot \overline{WR}$
0	0	STATUS	COMMAND
0	1	RESULT	PARAMETER
1	0	TxI/R	TEST MODE
1	1	RxI/R	—

Figure 7. 8273 Register Selection

Command — 8273 operations are initiated by writing the appropriate command byte into this register.

Parameter — Many commands require more information than found in the command itself. This additional information is provided by way of the parameter register.

Immediate Result (Result) — The completion information (results) for commands which execute immediately are provided in this register.

Transmit Interrupt Result (TxI/R) — Results of transmit operations are passed to the CPU in this register.

Receiver Interrupt Result (RxI/R) — Receive operation results are passed to the CPU via this register.

Status — The general status of the 8273 is provided in this register. The Status register supplies the handshaking necessary during various phases of the 8273 operation.

Test Mode — This register provides a software reset function for the 8273.

The commands, parameters, and bit definition of these registers are discussed in the following software section. Notice that there are not specific transmit or receive data registers. This feature is explained in the data transfer logic discussion.

The final elements of the C/R/W logic are the interrupt lines (RxINT and TxINT). These lines notify the CPU module that either the transmitter or the receiver requires service; i.e., results should be read from the appropriate interrupt result register or a data transfer is required. The interrupt request remains active until all the associated interrupt results have been read or the data transfer is performed. Though using the interrupt lines relieves the CPU module of the task of polling the 8273 to check if service is needed, the state of each interrupt line is reflected by a bit in the Status register and non-interrupt driven operation is possible by examing the contents of these bits periodically.

The 8273 supports two independent data interfaces through the data transfer logic; receive data and transmit data. These interfaces are programmable for either DMA or non-DMA data transfers. While the choice of the configuration is up to the system designer, it is based on the intended maximum data rate of the communications channel. Figure 8 illustrates the transfer rate of data bytes that are acquired by the 8273 based on link data rate. Full-duplex data rates above 9600 baud usually require DMA. Slower speeds may or may not require DMA depending on the task load and interrupt response time of the processor.

Figure 9 shows the 8273 in a typical DMA environment. Notice that a separate DMA controller, in this case the Intel 8257, is required. The DMA controller supplies the timing and addresses for the data transfers while the 8273 manages the requesting of transfers and the actual counting of the data block lengths. In this case, elements of the data transfer interface are:

TxDRQ: Transmit DMA Request — Asserted by the 8273, this line requests a DMA transfer from memory to the 8273 for transmit.

TxDACK: Transmit DMA Acknowledge — Returned by the 8257 in response to TxDRQ, this line notifies the 8273 that a request has been granted, and provides access to the transmitter data register.

RxDRQ: Receiver DMA Request — Asserted by the 8273, it requests a DMA transfer from the 8273 to memory for a receive operation.

TxDACK: Receiver DMA Acknowledge — Returned by the 8257, it notifies the 8273 that a receive DMA cycle has been granted, and provides access to the receiver data register.

RD: Read — Supplied by the 8257 to indicate data is to be read from the 8273 and placed in memory.

WR: Write — Supplied by the 8257 to indicate data is to be written to the 8273 from memory.

To request a DMA transfer the 8273 raises the appropriate DMA request line; let us assume it is a transmitter request (TxDRQ). Once the 8257 obtains control of the system bus by way of its HOLD and HLDA (hold acknowledge) lines, it notifies the 8273 that TxDRQ has been granted by returning $\overline{\text{TxDACK}}$ and $\overline{\text{WR}}$. The $\overline{\text{TxDACK}}$ and $\overline{\text{WR}}$ signals transfer data to the 8273 for a transmit, independent of the 8273 chip select pin ($\overline{\text{CS}}$). A similar sequence of events occurs for receiver requests. This "hard select" of data into the transmitter or out of the receiver alleviates the need for the normal transmit and receive data registers addressed by a combination of address lines, CS, and WR or RD. Competitive devices that do not have this "hard select" feature require the use of an external multiplexer to supply the correct inputs for register selection during DMA. (Do not forget that the SDLC controller sees both the addresses and control signals supplied by the DMA controller during DMA cycles.) Let us look at typical frame transmit and frame receive sequences to better see how the 8273 truly manages the DMA data transfer.

Before a frame can be transmitted, the DMA controller is supplied, by the CPU, the starting address for the desired information field. The 8273 is then commanded to transmit a frame. (Just how this is done is covered later during our software discussion.) After the command, but before transmission begins, the 8273 needs a little more information (parameters). Four parameters are required for the transmit frame command: the address field byte, the control field byte, and two bytes which are the least significant and most significant bytes of the information field byte length. Once all four parameters are loaded, the 8273 makes RTS (Request-to-Send) active and waits for CTS (Clear-to-Send) to go active. Once CTS is active, the 8273 starts the frame transmission. While the 8273 is transmitting the opening flag, address field, and control field; it starts making transmitter DMA requests. These requests continue at character (byte) boundaries until the pre-loaded number of bytes of information field have been transmitted. At this point the requests stop, the FCS and closing flag are transmitted, and the TxINT line is raised, signaling the CPU that the frame transmission is complete. Notice that after the initial command and parameter loading, absolutely no CPU intervention was required (since DMA is used for data transfers) until the entire frame was transmitted. Now let's look at a frame reception.

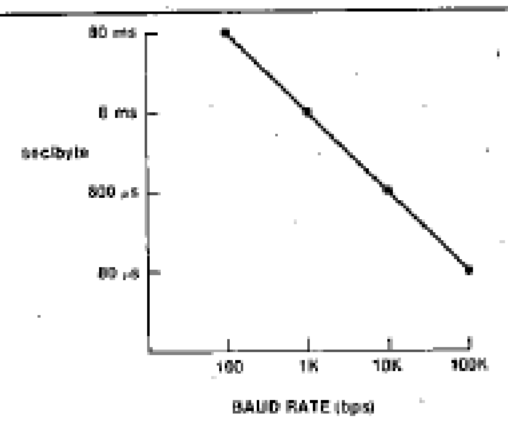

Figure 8. Byte Transfer Rate vs Baud Rate

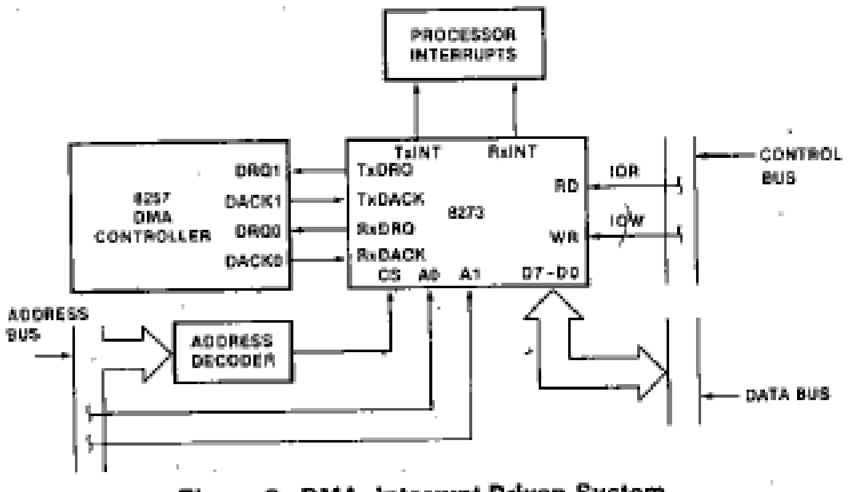

Figure 9. DMA, Interrupt-Driven System

The receiver operation is very similar. Like the initial transmit sequence, the DMA controller is loaded with a starting address for a receiver data buffer and the 8273 is commanded to receive. Unlike the transmitter, there are two different receive commands: General Receive, where all received frames are transferred to memory, and Selective Receive, where only frames having an address field matching one of two preprogrammed 8273 address fields are transferred to memory. Let's assume for now that we want to general receive. After the receive command, two parameters are required before the receiver becomes active: the least significant and most significant bytes of the receiver buffer length. Once these bytes are loaded, the receiver is active and the CPU may return to other tasks. The next frame appearing at the receiver input is transferred to memory using receiver DMA requests. When the closing flag is received, the 8273 checks the FCS and raises its RxINT line. The CPU can then read the results which indicate if the frame was error-free or not. (If the received frame had been longer than the pre-loaded buffer length, the CPU would have been notified of that occurrence earlier with a receiver error interrupt. The command description section contains a complete list of error conditions.) Like the transmit example, after the initial command, the CPU is free for other tasks until a frame is completely received. These examples have illustrated the 8273's management of both the receiver and transmitter DMA channels.

It is possible to use the DMA data transfer interface in a non-DMA interrupt-driven environment. In this case, 4 interrupt levels are used: one each for TxINT and RxINT, and one each for TxDRQ and RxDRQ. This configuration is shown in Figure 10. This configuration offers the advantages that no DMA controller is required and data requests are still separated from result (completion) requests. The disadvantages of the configuration are that 4 interrupt levels are required and that the CPU must actually supply the data transfers. This, of course, reduces the maximum data rate compared to the configuration based strictly on DMA. This system could use an Intel 8259 8-level Priority Interrupt Controller to supply a vectored CALL (subroutine) address based on requests on its inputs. The 8273 transmitter and receiver make data requests by raising the respective DRQ line. The CPU is interrupted by the 8259 and vectored to a data transfer routine. This routine either writes (for transmit) or reads (for receive) the 8273 using the respective TxDACK or RxDACK line. As in the case above, the DACK lines serve as "hard" chip selects into and out of the 8273. ($\overline{TxDACK}$ + $\overline{WR}$ writes data into the 8273 for transmit. $\overline{RxDACK}$ + $\overline{RD}$ reads data from the 8273 for receive.) The CPU is notified of operation completion and results by way of TxINT and RxINT lines. Using the 8273, and the 8259, in this way, provides a very effective, yet simple, interrupt-driven interface.

Figure 11 illustrates a system very similar to that described above. This system utilizes the 8273 in a non-DMA data transfer mode as opposed to the two DMA approaches shown in Figures 9 and 10. In the non-DMA case, data transfer requests are made on the TxINT and RxINT lines. The DRQ lines are not used. Data transfer requests are separated from result requests by a bit in the Status register. Thus, in response to an interrupt, the CPU reads the Status register and branches to either a result or a data transfer routine based on the status of one bit. As before, data transfers are made via using the DACK lines as chip selects to the transmitter and receiver data registers.

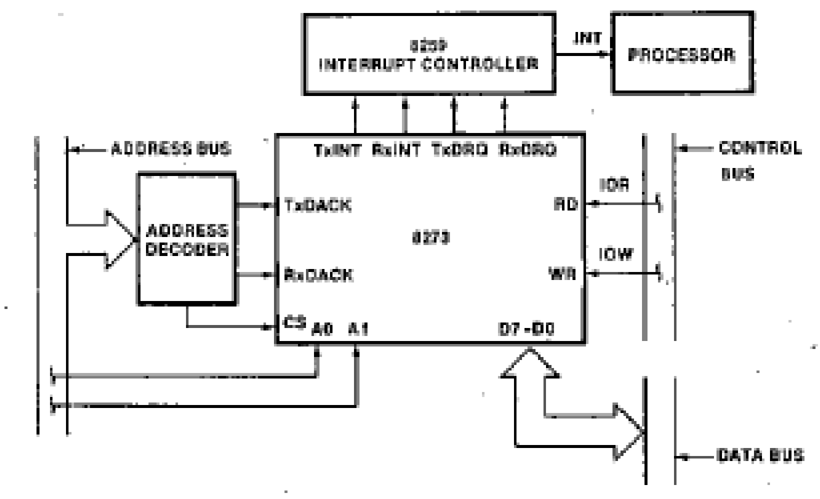

Figure 10. Interrupt-Based DMA System

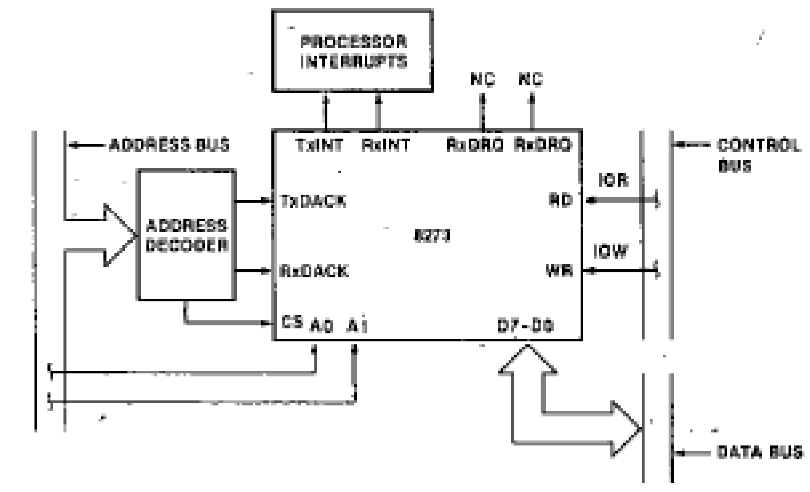

Figure 11. Non-DMA Interrupt-Driven System

Figure 12 illustrates the simplest system of all. This system utilizes polling for all data transfers and results. Since the interrupt pins are reflected in bits in the Status register, the software can read the Status register periodically looking for one of these to be set. If it finds an INT bit set, the appropriate Result Available bit is examined to determine if the "interrupt" is a data transfer or completion result. If a data transfer is called for, the DACK line is used to enter or read the data from the 8273. If the interrupt is a completion result, the appropriate result register is read to determine the good/bad completion of the operation.

The actual selection of either DMA or non-DMA modes is controlled by a command issued during initialization. This command is covered in detail during the software discussion.

The final block of the CPU module interface is the Data Bus Buffer. This block supplies the tri-state, bidirectional data bus interface to allow communication to and from the 8273.

Modem Interface

As the name implies, the modem interface is the modem side of the 8273. It consists of two major blocks: the modem control block and the serial data timing block.

The modem control block provides both dedicated and user-defined modem control functions. All signals supported by this interface are active low so that EIA inverting drivers (MC1488) and inverting receivers (MC1489) may be used to interface to standard modems.

Port A is a modem control input port. Its representation on the data bus is shown in Figure 13. Bits D_0 and D_1 have dedicated functions. D_0 reflects the logical state of the $\overline{CTS}$ (Clear-to-Send) pin. [If $\overline{CTS}$ is active (low), D_0 is a 1.] This signal is used to condition the start of a transmission. The 8273 waits until $\overline{CTS}$ is active before it starts transmitting a frame. While transmitting, if $\overline{CTS}$ goes inactive, the frame is aborted and the CPU is interrupted. When the CPU reads the interrupt result, a $\overline{CTS}$ failure is indicated.

D_1 reflects the logical state of the $\overline{CD}$ (Carrier Detect) pin. $\overline{CD}$ is used to condition the start of a frame reception. $\overline{CD}$ must be active in time for a frame's address field. If $\overline{CD}$ is lost (goes inactive) while receiving a frame, an interrupt is generated with a $\overline{CD}$ failure result. $\overline{CD}$ may go inactive between frames.

Bits D_2 thru D_4 reflect the logical state of the $\overline{PA_2}$ thru $\overline{PA_4}$ pins respectively. These inputs are user defined. The 8273 does not interrogate or manipulate these bits. Bits D_5, D_6, and D_7 are not used and each is read as a 1 for a Read Port A command.

Port B is a modem control output port. Its data bus representation is shown in Figure 14. As in Port A, the bit values represent the logical condition of the pins. D_0 and D_5 are dedicated function outputs. D_0 represents the $\overline{RTS}$ (Request-to-Send) pin. $\overline{RTS}$ is normally used to notify the modem that the 8273 wishes to transmit. This function is handled automatically by the 8273. If $\overline{RTS}$ is inactive (pin is high) when the 8273 is commanded to transmit, the 8273 makes it active and then waits for $\overline{CTS}$ before transmitting the frame. One byte time after the end of the frame, the 8273 returns $\overline{RTS}$ to its inactive state. However, if $\overline{RTS}$ was active when a transmit command is issued, the 8273 leaves it active when the frame is complete.

Bit D_5 reflects the state of the $\overline{\text{Flag Detect}}$ pin. This pin is activated whenever an active receiver sees a flag character. This function is useful to activate a timer for line activity timeout purposes.

Bits D_1 thru D_4 provide four user-defined outputs. Pins $\overline{PB_1}$ thru $\overline{PB_4}$ reflect the logical state of these bits. The 8273 does not interrogate or manipulate these bits. D_6 and D_7 are not used. In addition to being able to output to Port B, Port B may be read using a Read Port B command. All Modem control output pins are forced high on reset. (All commands mentioned in this section are covered in detail later.).

The final block to be covered is the serial data timing block. This block contains two sections: the serial data logic and the digital phase locked loop (DPLL).

Elements of the serial data logic section are the data pins, TxD (transmit data output) and RxD (receive data input), and the respective data clocks, $\overline{TxC}$ and $\overline{RxC}$. The transmit and receive data is synchronized by the $\overline{TxC}$ and $\overline{RxC}$ clocks. Figure 15 shows the timing for these signals. The leading edge (negative transition) of $\overline{TxC}$ generates new transmit data and the trailing edge (positive transition) of $\overline{RxC}$ is used to capture the receive data.

It is possible to reconfigure this section under program control to perform diagnostic functions; both data and clock loopback are available. In data loopback mode, the TxD pin is internally routed to the RxD pin. This allows simple board checkout since the CPU can send an SDLC message to itself. (Note that transmitted data will still appear on the TxD pin.)

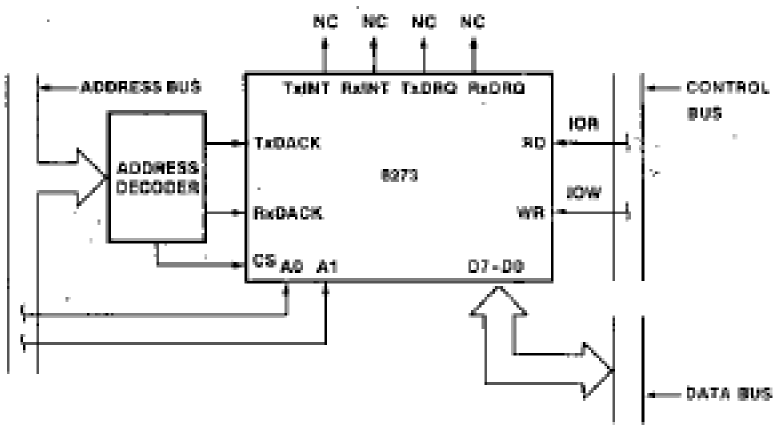

Figure 12. Polled System

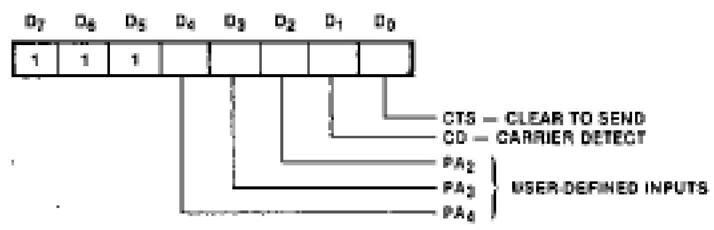

Figure 13. Port A (Input) Bit Definition

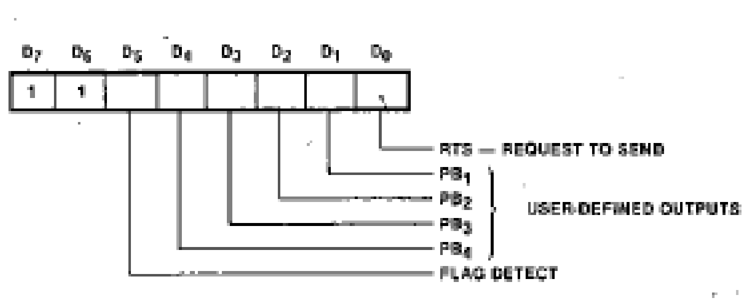

Figure 14. Port B (Output) Bit Definition

When data loopback is utilized, the receiver may be presented incorrect sample timing (RxC) by the external circuitry. Clock loopback overcomes this problem by allowing the internal routing of TxC and RxC. Thus the same clock used to transmit the data is used to receive it. Examination of Figure 15 shows that this method ensures bit synchronism. The final element of the serial data logic is the Digital Phase Locked Loop.

The DPLL provides a means of clock recovery from the received data stream. This feature allows the 8273 to interface without external synchronizing logic to low cost asynchronous modems (modems which do not supply clocks). It also makes the problem of clock timing in loop configurations trivial.

To use the DPLL, a clock at 32 times the required baud rate must be supplied to the 32 × CLK pin. This clock provides the interval that the DPLL samples the received data. The DPLL uses the 32× clock and the received data to generate a pulse at the DPLL output pin. This DPLL pulse is positioned at the nominal center of the received data bit cell. Thus the DPLL output may be wired to RxC and/or TxC to supply the data timing. The exact position of the pulse is varied depending on the line noise and bit distortion of the received data. The adjustment of the DPLL position is determined according to the rules outlined in Figure 16.

Adjustments to the sample phase of DPLL with respect to the received data is made in discrete increments. Referring to Figure 16, following the occurrence of DPLL pulse A, the DPLL counts 32 × CLK pulses and examines the received data for a data edge. Should no edge be detected in 32 pulses, the DPLL positions the next DPLL pulse (B) at 32 clock pulses from pulse A. Since no new phase information is contained in the data stream, the sample phase is assumed to be at nominal 1× baud rate. Now assume a data edge occurs after DPLL pulse B. The distance from B to the next pulse C is influenced according to which quadrant (A_1, B_1, B_2, or A_2) the data edge falls in. (Each quadrant represents 8 32 × CLK times.) For example, if the edge is detected in quadrant A_1, it is apparent that pulse B was too close to the data edge and the time to the next pulse must be shortened. The adjustment for quadrant A_1 is specified as −2. Thus, the next DPLL pulse, pulse C, is positioned 32−2 or 30 32 × CLK pulses following DPLL pulse B. This adjustment moves pulse C closer to the nominal bit center of the next received data cell. A data edge occurring in quadrant B_2 would have caused the adjustment to be small, namely 32+1 or 33 32 × CLK pulses. Using this technique, the DPLL pulse converges to the nominal bit center within 12 data transitions, worse case — 4-bit times adjusting through quadrant A_1 or A_2 and 8-bit times adjusting through B_1 or B_2.

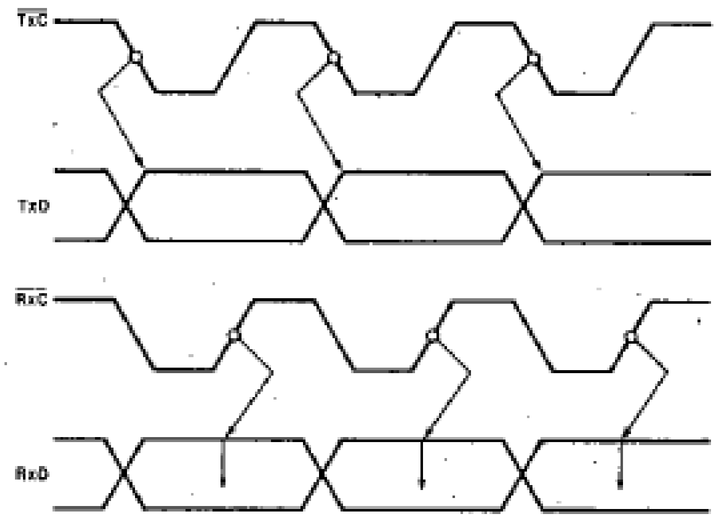

Figure 15. Transmit/Receive Timing

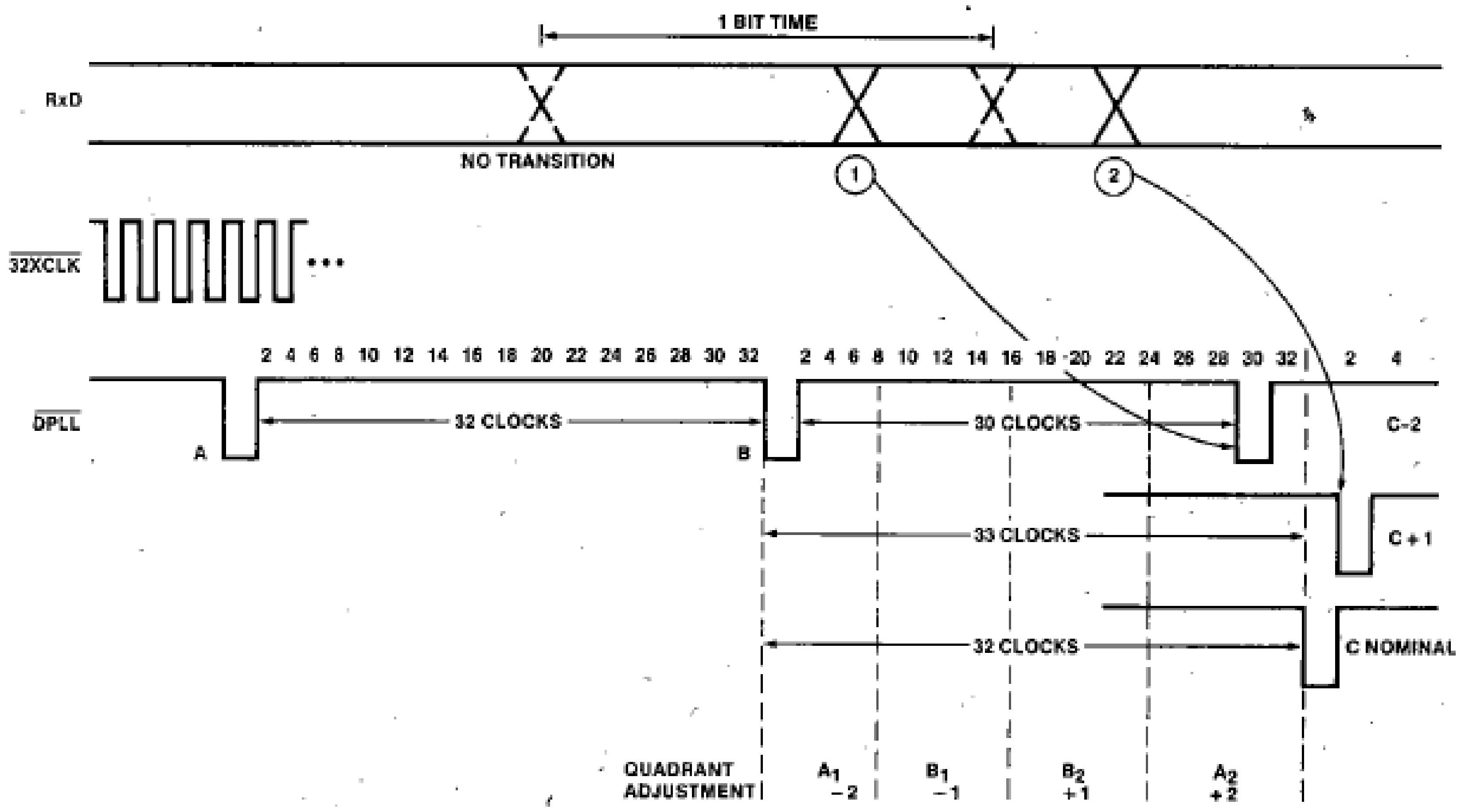

Figure 16. DPLL Phase Adjustments

When the receive data stream goes idle after 15 ones, DPLL pulses are generated at 32 pulse intervals of the 32x CLK. This feature allows the DPLL pulses to be used as both transmitter and receiver clocks.

In order to guarantee sufficient transitions of the received data to enable the $\overline{\text{DPLL}}$ to lock, NRZI encoding of the data is recommended. This ensures that, within a frame, data transitions occur at least every five bit times — the longest sequence of 1s which may be transmitted with zero bit insertion. It is also recommended that frames following a line idle be transmitted with preframe sync characters which provide a minimum of 12 transitions. This ensures that the $\overline{\text{DPLL}}$ is generating $\overline{\text{DPLL}}$ pulses at the nominal bit centers in time for the opening flag. (Two 00H characters meet this requirement by supplying 16 transitions with NRZI encoding. The 8273 contains a mode which supplies such a preframe sync.)

Figure 17 illustrates 8273 clock configurations using either synchronous or asynchronous modems. Notice how the $\overline{\text{DPLL}}$ output is used for both $\overline{\text{TxC}}$ and $\overline{\text{RxC}}$ in the asynchronous case. This feature eliminates the need for external clock generation logic where low cost) asynchronous modems are used and also allows direct connection of 8273s for the ultimate in low cost data links. The configuration for loop, applications is discussed in a following section.

This completes our discussion of the hardware aspects of the 8273. Its software aspects are now discussed.

SOFTWARE ASPECTS OF THE 8273

The software aspects of the 8273 involve the communication of both commands from the CPU to the 8273 and the return of results of those commands from the 8273 to the CPU. Due to the internal processor architecture of the 8273, this CPU-8273 communication is basically a form of interprocessor communication. Such communication usually requires a form of protocol of its own. This protocol is implemented through use of handshaking supplied in the 8273 Status register. The bit definition of this register is shown in Figure 18.

CBSY: Command Busy — CBSY indicates when the 8273 is in the command phase. CBSY is set when the CPU writes a command into the Command register, starting the Command phase. It is reset when the last parameter is deposited in the Parameter register and accepted by the 8273, completing the Command phase.

CBF: Command Buffer Full — When set, this bit indicates that a byte is present in the Command register. This bit is normally not used.

CPBF: Command Parameter Buffer Full — This bit indicates that the Parameter register contains a parameter. It is set when the CPU deposits a parameter in the Parameter register. It is reset when the 8273 accepts the parameter.

CRBF: Command Result Buffer Full — This bit is set when the 8273 places a result from an immediate type command in the Result register. It is reset when the CPU reads the result from the Result register.

RxINT: Receiver Interrupt — The state of the RxINT pin is reflected by this bit. RxINT is set by the 8273 whenever the receiver needs servicing. RxINT is reset when the CPU reads the results or performs the data transfer.

TxINT: Transmitter Interrupt — This bit is identical to RxINT except action is initiated based on transmitter interrupt sources.

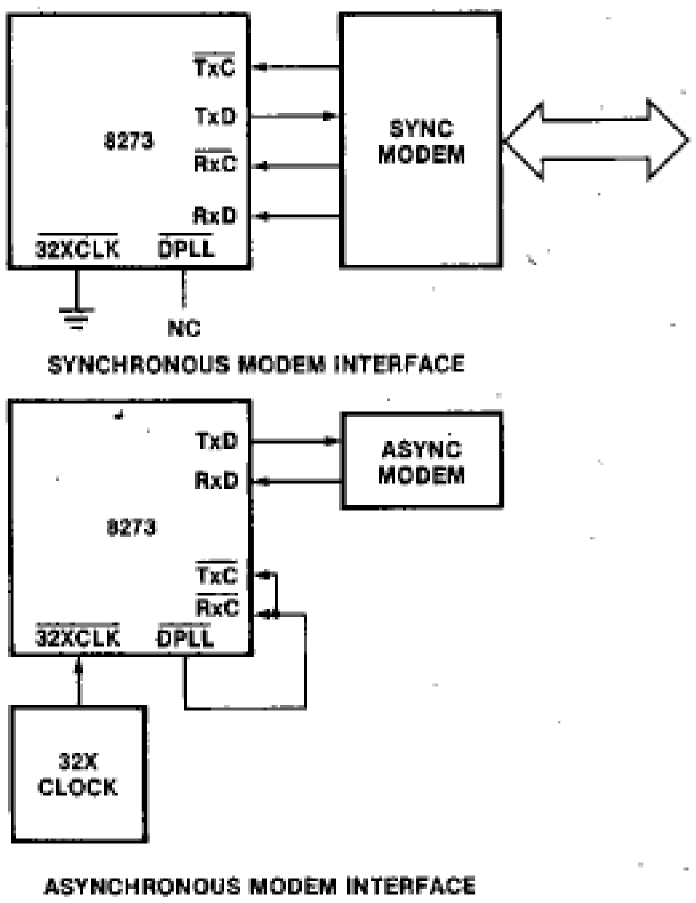

Figure 17. Serial Data Timing Configuration

231311-001

RxIRA: Receiver Interrupt Result Available — RxIRA is set when the 8273 places an interrupt result byte into the RxI/R register. RxIRA is reset when the CPU reads the RxI/R register.

TxIRA: Transmitter Interrupt Result Available — TxIRA is the corresponding Result Available bit for the transmitter. It is set when the 8273 places an interrupt result byte in the TxI/R register and reset when the CPU reads the register.

The significance of each of these bits will be evident shortly. Since the software requirements of each 8273 phase are essentially independent, each phase is covered separately.

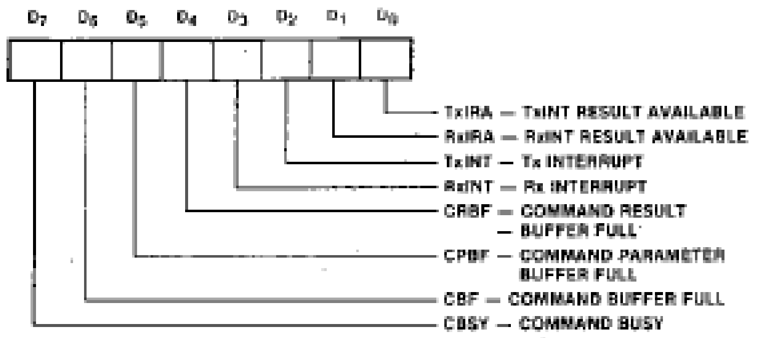

Figure 18. Status Register Format

Command Phase Software

Recalling the Command phase description in an earlier section, the CPU starts the Command phase by writing a command byte into the 8273 Command register. If further information about the command is required by the 8273, the CPU writes this information into the Parameter register. Figure 19 is a flowchart of the Command phase. Notice that the CBSY and CPBF bits of the Status register are used to handshake the command and parameter bytes. Also note that the chart shows that a command may not be issued if the Status register indicates the 8273 is busy (CBSY = 1). If a command is issued while CBSY = 1, the original command is overwritten and lost. (Remember that CBSY signifies the command phase is in progress and not the actual execution of the command.) The flowchart also includes a Parameter buffer full check. The CPU must wait until CPBF = 0 before writing a parameter to the Parameter register. If a parameter is issued while CPBF = 1, the previous parameter is overwritten and lost. An example of command output assembly language software is provided in Figure 20a. This software assumes that a command buffer exists in memory. The buffer is pointed at by the HL register. Figure 20b shows the command buffer structure.

The 8273 is a full duplex device, i.e., both the transmitter and receiver may be executing commands or passing interrupt results at any given time. (Separate Rx and Tx interrupt pins and result registers are provided for this reason.) However, there is only one Command register. Thus, the Command register must be used for only one command sequence at a time and the transmitter and receiver may never be simultaneously in a command phase. A detailed description of the commands and their parameters is presented in a following section.

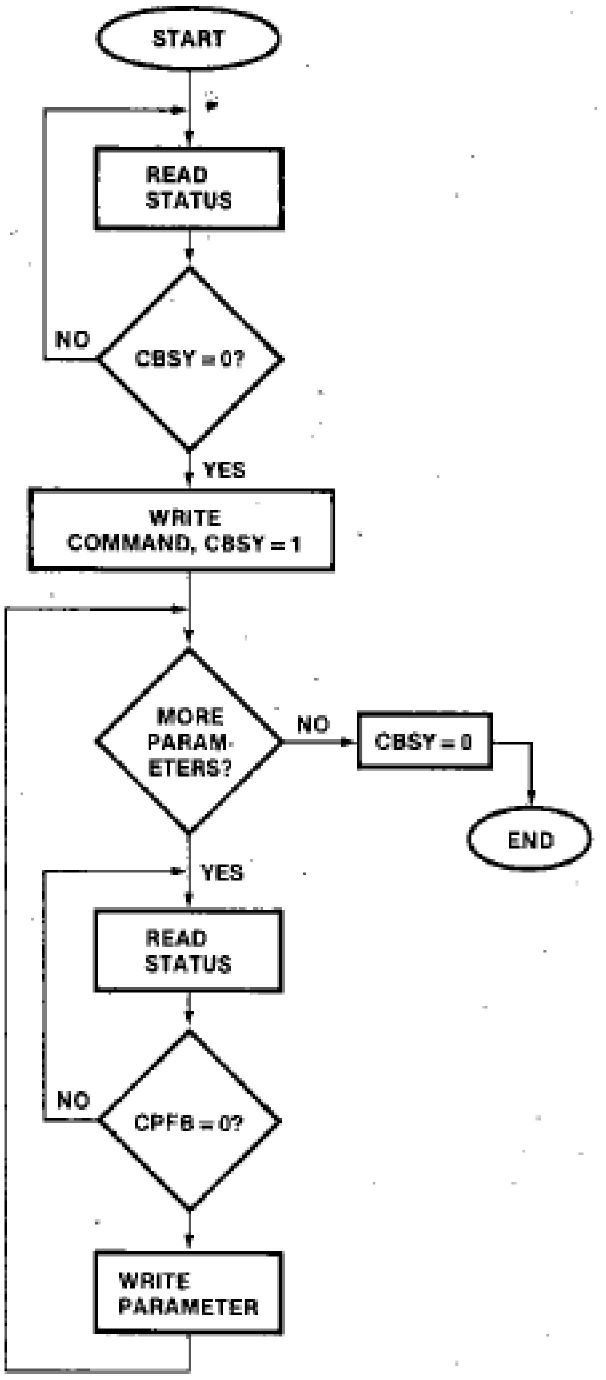

Figure 19. Command Phase Flowchart

```
;FUNCTION: COMMAND DISPATCHER
;INPUTS: HL - COMMAND BUFFER ADDRESS
;OUTPUTS: NONE
;CALLS: NONE
;DESTROYS: A,B,H,L,F/F'S
;DESCRIPTION: CMDOUT ISSUES THE COMMAND + PARAMETERS
;IN THE COMMAND BUFFER POINTED AT BY HL
;
CMDOUT: LXI     H,CMDBUF;POINT HL AT BUFFER
        MOV     B,M     ;1ST ENTRY IS PAR. COUNT
        INX     H       ;POINT AT COMMAND BYTE
CMD1:   IN      STAT73  ;READ 8273 STATUS
        RLC     ;ROTATE CBSY INTO CARRY
        JC      CMD1    ;WAIT UNTIL CBSY=0
        MOV     A,M     ;MOVE COMMAND BYTE TO A
        OUT     COMM73  ;PUT COMMAND IN COMMAND REG
CMD2:   MOV     A,B     ;GET PARAMETER COUNT
        ANA     A       ;TEST IF ZERO
        RZ      ;IF 0 THEN DONE
        INX     H       ;NOT DONE, SO POINT AT NEXT PAR
        DCR     B       ;DEC PARAMETER COUNT
CMD3:   IN      STAT73  ;READ 8273 STATUS
        ANI     CPBF    ;TEST CPBF BIT
        JNZ     CMD3    ;WAIT UNTIL CPBF IS 0
        MOV     A,M     ;GET PARAMETER FROM BUFFER
        OUT     PARM73  ;OUTPUT PAR TO PARAMETER REG
        JMP     CMD2    ;CHECK IF MORE PARAMETERS
```

Figure 20A. Command Phase Software

+4	PARAMETER 3	
+3	PARAMETER 2	
+2	PARAMETER 1	
+1	COMMAND	
CMDBUF:	PARAMETER COUNT	← HL

Figure 20B. Command Buffer Format

Execution Phase Software

During the Execution phase, the operation specified by the Command phase is performed. If the system utilizes DMA for data transfers, there is no CPU involvement during this phase, so no software is required. If non-DMA data transfers are used, either interrupts or polling is used to signal a data transfer request.

For interrupt-driven transfers the 8273 raises the appropriate INT pin. When responding to the interrupt, the CPU must determine whether it is a data transfer request or an interrupt signaling that an operation is complete and results are available. The CPU determines the cause by reading the Status register and interrogating the associated IRA (Interrupt Result Available) bit (TxIRA for TxINT and RxIRA for RxINT). If the IRA = 0, the interrupt is a data transfer request. If the IRA = 1, an operation is complete and the associated Interrupt Result register must be read to determine the completion status (good/bad/etc.). A software interrupt handler implementing the above sequence is presented as part of the Result phase software.

When polling is used to determine when data transfers are required, the polling routine reads the Status register looking for one of the INT bits to be set. When a set INT bit is found, the corresponding IRA bit is examined. Like in the interrupt-driven case, if the IRA = 0, a data transfer is required. If IRA = 1, an operation is complete and the Interrupt Result register needs to be read. Again, example polling software is presented in the next section.

Result Phase Software

During the Result phase the 8273 notifies the CPU of the outcome of a command. The Result phase is initiated by either a successful completion of an operation or an error detected during execution. Some commands such as reading or writing the I/O ports provide immediate results, that is, there is essentially no delay from the issuing of the command and when the result is available. Other commands such as frame transmit, take time to complete so their result is not available immediately. Separate result registers are provided to distinguish these two types of commands and to avoid interrupt handling for simple results.

Immediate results are provided in the Result register. Validity of information in this register is indicated to the CPU by way of the CRBF bit in the Status register. When the CPU completes the Command phase of an immediate command, it polls the Status register waiting until CRBF = 1. When this occurs, the CPU may read the Result register to obtain the immediate result. The Result register provides only the results from immediate commands.

Example software for handling immediate results is shown in Figure 21. The routine returns with the result in the accumulator. The CPU then uses the result as is appropriate.

All non-immediate commands deal with either the transmitter or receiver. Results from these commands are provided in the TxI/R (Transmit Interrupt Result) and RxI/R (Receive Interrupt Result) registers respectively. Results in these registers are conveyed to the CPU by the TxIRA and RxIRA bits of the Status register. Results of non-immediate commands consist of one byte result interrupt code indicating the condition for the interrupt and, if required, one or more bytes supplying additional information. The interrupt codes and the meaning of the additional results are covered following the detailed command description.

Non-immediate results are passed to the CPU in response to either interrupts or polling of the Status register. Figure 22 illustrates an interrupt-driven result handler. (Please note that all of the software presented in this application note is not optimized for either speed or code efficiency. They are provided as a guide and to illustrate concepts.) This handler provides for interrupt-driven data transfers as was promised in the last section. Users employing DMA-based transfers do not need the lines where the IRA bit is tested for zero. (These lines are denoted by an asterisk in the comments column.) Note that the INT bit is used to determine when all results have been read. All results must be read. Otherwise, the INT bit (and pin) will remain high and further interrupts may be missed. These routines place the results in a result buffer pointed at by RCRBUF and TxRBUF.

A typical result handler for systems utilizing polling is shown in Figure 23. Data transfers are also handled by this routine. This routine utilizes the routines of Figure 22 to handle the results.

At this point, the reader should have a good conceptual feel about how the 8273 operates. It is now time for the particulars of each command to be discussed.

```
;FUNCTION: IMDRLT
;INPUTS: NONE
;OUTPUTS: RESULT REGISTER IN A
;CALLS: NONE
;DESTROYS: A,F/F'S
;DESCRIPTION: IMDRLT IS CALLED AFTER A CMDOUT FOR AN
;IMMEDIATE COMMAND TO READ THE RESULT REGISTER
;
IMDRLT: IN      STAT73  ;READ 8273 STATUS
        ANI     CRBF    ;TEST IF RESULT REG READY
        JZ      IMDRLT  ;WAIT IF CRBF=0
        IN      RESL73  ;READ RESULT REGISTER
        RET     ;RETURN
```

Figure 21. Immediate Result Handler

```
;FUNCTION: RXI - INTERRUPT DRIVEN RESULT/DATA HANDLER
;INPUTS: RCRBUF, RCVPNT
;CALLS: NONE
;OUTPUTS: RCRBUF, RCVPNT
;DESTROYS: NOTHING
;DESCRIPTION: RXI IS ENTERED AT A RECEIVER INTERRUPT.
;THE INTERRUPT IS TESTED FOR DATA TRANSFER (IRA=0)
;OR RESULT (IRA=1).  FOR DATA TRANSFER, THE DATA IS
;PLACED IN A BUFFER AT RCVPNT. RESULTS ARE PLACED IN
;A BUFFER AT RCRBUF.
;A FLAG(RXFLAG) IS SET IF THE INTERRUPT WAS A RESULT.
;(DATA TRANSFER INSTRUCTIONS ARE DENOTED BY (*) AND
;MAYBE ELIMINATED BY USERS USING DMA.
;
RXI:    PUSH    H       ;SAVE HL
        PUSH    PSW     ;SAVE PSW
        PUSH    B       ;SAVE B
        IN      STAT73  ; (*) READ 8273 STATUS
        ANI     RXIRA   ; (*) TEST IRA BIT
        JZ      RXI2    ; (*) IF 0, DATA TRANSFER NEEDED
RXI1:   LHLD    RCRBUF  ;GET RESULT BUFFER POINTER
        IN      STAT73  ;READ 8273 STATUS AGAIN
        ANI     RXINT   ;TEST INT BIT
        JZ      RXI4    ;IF 0, THEN DONE
        IN      STAT73  ;READ 8273 STATUS AGAIN
        ANI     RXIRA   ;TEST IRA AGAIN
        JZ      RXI1    ;LOOP UNTIL RESULT IS READY
        IN      RXIR73  ;READY, READ RXI/R
        MOV     M,A     ;STORE RESULT IN BUFFER
        INX     H       ;BUMP RESULT POINTER
        SHLD    RCRBUF  ;RESTORE BUFFER POINTER
        JMP     RXI1    ;GO BACK TO SEE IF MORE
RXI2:   SHLD    RCVPNT  ; (*) GET DATA BUFFER POINTER
        IN      RCVDAT  ; (*) READ DATA VIA RXDACK
        MOV     M,A     ; (*) STORE DATA IN BUFFER
        INX     H       ; (*) BUMP DATA POINTER
        JMP     RXI3    ; (*) DONE
RXI4:   MVI     A,01H   ;SET RX FLAG TO SHOW COMPLETION
        STA     RXFLAG  ;COMPLETION
RXI3:   POP     B       ;RESTORE BC
        POP     PSW     ;RESTORE PSW
        POP     H       ;RESTORE HL
        EI      ;ENABLE INTERRUPTS
        RET     ;DONE

;FUNCTION: TXI - INTERRUPT DRIVEN RESULT/DATA HANDLER
;INPUTS: TXRBUF, TXPNT, TXFLAG
;OUTPUTS: TXRBUF, TXPNT, TXFLAG
;CALLS: NONE
;DESTROYS: NOTHING
;DESCRIPTION: TXI IS ENTERED AT A TRANSMITTER INTERRUPT.
;THE INTERRUPT IS TESTED BY WAY OF THE IRA BIT TO SEE
;IF A DATA TRANSFER OR RESULT COMPLETION HAS OCCURED.
;FOR DATA TRANSFERS (IRA=0), THE DATA IS OBTAINED FROM
;A BUFFER LOCATION POINTED AT BY TXPNT.  FOR COMPLETION,
;(IRA=1), THE RESULTS ARE READ AND PLACED AT A RESULT
;BUFFER POINTED AT BY TXRBUF, AND THE TXFLAG IS SET
;TO INDICATE TO THE MAIN PROGRAM THAT A OPERATION IS
;COMPLETE. TX OPERATIONS HAVE ONLY ONE RESULT.
;DATA TRANSFER INSTRUCTIONS ARE DENOTED BY (*). THESE
;MAYBE REMOVED BY USERS USING DMA.
;
TXI:    PUSH    H       ;SAVE HL
        PUSH    PSW     ;SAVE PSW
        IN      STAT73  ; (*) READ 8273 STATUS
        ANI     TXIRA   ; (*) TEST TXIRA BIT
        JZ      TXI2    ; (*) IF 0, DATA TRANSFER
        IN      TXIR73  ;1, THEN READ TXIR
        LHLD    TXRBUF  ;GET RESULT BUFFER POINTER
        MOV     M,A     ;STORE RESULT IN BUFFER
        INX     H       ;BUMP RESULT POINTER
        SHLD    TXRBUF  ;RESTORE RESULT POINTER
        MVI     A,01H   ;SET TXFLAG TO SHOW COMPLETION
        STA     TXFLAG  ;SET FLAG
TXI1:   POP     PSW     ;RESTORE PSW
        POP     H       ;RESTORE HL
        EI      ;ENABLE INTERRUPTS
        RET     ;DONE
TXI2:   LHLD    TXPNT   ; (*) GET DATA POINTER
        MOV     A,M     ; (*) GET DATA FROM BUFFER
        OUT     TXDATA  ; (*) OUTPUT TO 8273 VIA TXDACK
        INX     H       ; (*) BUMP DATA POINTER
        SHLD    TXPNT   ; (*) RESTORE POINTER
        JMP     TXI1    ; (*) RETURN AFTER RESTORE
```

Figure 22. Interrupt-Driven Result Handlers with Non-DMA Data Transfers

```
;FUNCTION: POLOP
;INPUTS: NONE
;OUTPUTS: C=0 (NO STATUS), =1 (RX COMPLETION),
;    =2 (TX COMPLETION), =3 (BOTH)
;CALLS: TXI, RXI
;DESTROYS: B,C
;DESCRIPTION: POLOP IS CALLED TO POLL THE 8273 FOR
;DATA TRANSFERS AND COMPLETION RESULTS.  THE
;ROUTINES TXI AND RXI ARE USED FOR THE ACTUAL
;TRANSFERS AND BUFFER WORK.  POLOP RETURNS
;THE STATUS OF THEIR ACTION.
;
POLOP:  PUSH    PSW     ;SAVE PSW
        MVI     C,00H   ;CLEAR C
POLOP1: IN      STAT73  ;READ 8273 STATUS
        ANI     INT     ;ARE TXINT OR RXINT SET?
        JZ      PEXIT   ;NO, EXIT
        IN      STAT73  ;READ 8273 STATUS
        ANI     RXINT   ;TEST RX INT
        JNZ     RXIC    ;YES, GO SERVICE RX
        CALL    TXI     ;MUST BE TX, GO SERVICE IT
        LDA     TXFLAG  ;GET TX FLAG
        CPI     01H     ;WAS IT A COMPLETION? (01)
        JNZ     PEXIT   ;NO, SO JUST EXIT
        INR     C       ;YES, UPDATE C
        INR     C
        JMP     POLOP1  ;TRY AGAIN
;
RXIC:   CALL    RXI     ;GO SERVICE RX
        LDA     RXFLAG  ;GET RX FLAG
        CPI     01H     ;WAS IT A COMPLETION? (01)
        JNZ     PEXIT   ;NO, SO JUST EXIT
        INR     C       ;YES, UPDATE C
        JMP     POLOP1  ;TRY AGAIN
;
PEXIT:  POP     PSW     ;RESTORE PSW
        RET     ;RETURN WITH COMP. STATUS IN C
```

Figure 23. Polling Result Handler

8273 COMMAND DESCRIPTION

In this section, each command is discussed in detail. In order to shorten the notation, please refer to the command key in Table 1. The 8273 utilizes five different command types: Initialization/Configuration, Receive, Transmit, Reset, and Modem Control.

Initialization/Configuration Commands

The Initialization/Configuration commands manipulate registers internal to the 8273 that define the various operating modes. These commands either set or reset specified bits in the registers depending on the type of command. One parameter is required. Set commands perform a logical OR operation of the parameter (mask) and the internal register. This mask contains 1s where register bits are to be set. A 0 in the mask causes no change in the corresponding register bit. Reset commands perform a logical AND operation of the parameter (mask) and the internal register, i.e., the mask is 0 to reset a register bit and a 1 to cause no change. Before presenting the commands, the register bit definitions are discussed.

TABLE 1. COMMAND SUMMARY KEY

B_0, B_1	— LSB AND MSB OF RECEIVE BUFFER LENGTH
R_0, R_1	— LSB AND MSB OF RECEIVED FRAME LENGTH
L_0, L_1	— LSB AND MSB OF TRANSMIT FRAME LENGTH
A_1, A_2	— MATCH ADDRESSES FOR SELECTIVE RECEIVE
RIC	— RECEIVER INTERRUPT RESULT CODE
TIC	— TRANSMITTER INTERRUPT RESULT CODE
A	— ADDRESS FIELD OF RECEIVED FRAME
C	— CONTROL FIELD OF RECEIVED FRAME

Operating Mode Register (Figure 24)

D_7-D_6: *Not Used* — These bits must not be manipulated by any command; i.e., D_7-D_6 must be 0 for the Set command and 1 for the Reset command.

D_5: *HDLC Abort* — When this bit is set, the 8273 will interrupt when 7 1s (HDLC Abort) are received by an active receiver. When reset, an SDLC Abort (8 1s) will cause an interrupt.

D_4: *EOP Interrupt* — Reception of an EOP character (0 followed by 7 1s) will cause the 8273 to interrupt the CPU when this bit is set. Loop controller stations use this mode as a signal that a polling frame has completed the loop. No EOP interrupt is generated when this bit is reset.

D_3: *Early Tx Interrupt* — This bit specifies when the transmitter should generate an end of frame interrupt. If this bit is set, an interrupt is generated when the last data character has been passed to the 8273. If the user software issues another transmit command within two byte times, the final flag interrupt does not occur and the new frame is transmitted with only one flag of separation. If this restriction is not met, more than one flag will separate the frames and a frame complete interrupt is generated after the closing flag. If the bit is reset, only the frame complete interrupt occurs. This bit, when set, allows a single flag to separate consecutive frames.

D_2: *Buffered Address and Control* — When set, the address and control fields of received frames are buffered in the 8273 and passed to the CPU as results after a received frame interrupt (they are not transferred to memory with the information field). On transmit, the A and C fields are passed to the 8273 as parameters. This mode simplifies buffer management. When this bit is reset, the A and C fields are passed to and from memory as the first two data transfers.

D_1: *Preframe Sync* — When set, the 8273 prefaces each transmitted frame with two characters before the opening flag. These two characters provide 16 transitions to allow synchronization of the opposing receiver. To guarantee 16 transitions, the two characters are 55H-55H for non-NRZI mode (see Serial I/O Register description) or 00H-00H for NRZI mode. When reset, no preframe characters are transmitted.

D_0: *Flag Stream* — When set, the transmitter will start sending flag characters as soon as it is idle; i.e., immediately if idle when the command is issued or after a transmission if the transmitter is active when this bit is set. When reset, the transmitter starts sending Idle characters on the next character boundary if idle already, or at the end of a transmission if active.

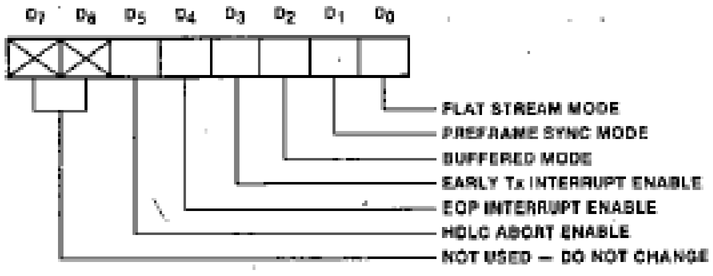

Figure 24. Operating Mode Register

Serial I/O Mode Register (Figure 25)

D_7-D_3: *Not Used* — These bits must be 0 for the Set command and 1 for the Reset command.

D_2: *Data Loopback*— When set, transmitted data (TxD) is internally routed to the receive data circuitry. When reset, TxD and RxD are independent.

D_1: *Clock Loopback* — When set, $\overline{TxC}$ is internally routed to $\overline{RxC}$. When reset, the clocks are independent.

D_0: *NRZI (Non-Return to Zero Inverted)* — When set, the 8273 assumes the received data is NRZI encoded, and NRZI encodes the transmitted data. When reset, the received and transmitted data are treated as a normal positive logic bit stream.

Data Transfer Mode Register (Figure 26)

D_7-D_1: *Not Used* — These bits must be 0 for the Set command and 1 for the Reset command.

D_0: *Interrupt Data Transfer* — When set, the 8273 will interrupt the CPU when data transfers are required (the corresponding IRA Status register bit will be 0 to signify a data transfer interrupt rather than a Result phase interrupt). When reset, 8273 data transfers are performed through DMA requests on the DRQ pins without interrupting the CPU.

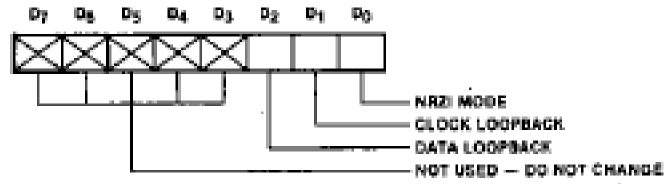

Figure 25. Serial I/O Mode Register

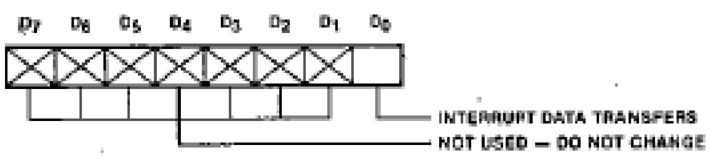

Figure 26. Data Transfer Mode Register

One Bit Delay Register (Figure 27)

D_7: *One Bit Delay* — When set, the 8273 retransmits the received data stream one bit delayed. This mode is entered and exited at a received character boundary. When reset, the transmitted and received data are independent. This mode is utilized for loop operation and is discussed in a later section.

D_6-D_0: *Not Used* — These bits must be 0 for the Set command and 1 for the Reset command.

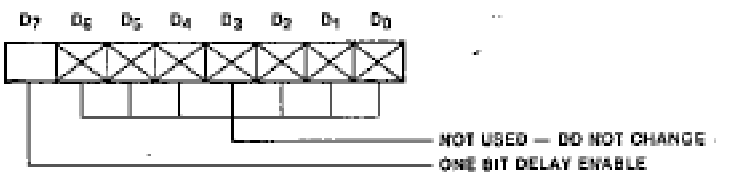

Figure 27. One Bit Delay Mode Register

Figure 28 shows the Set and Reset commands associated with the above registers. The mask which sets or resets the desired bits is treated as a single parameter. These commands do not interrupt nor provide results during the Result phase. After reset, the 8273 defaults to all of these bits reset.

REGISTER	COMMAND	HEX CODE	PARAMETER
ONE BIT DELAY MODE	SET	A4	SET MASK
	RESET	64	RESET MASK
DATA TRANSFER MODE	SET	97	SET MASK
	RESET	57	RESET MASK
OPERATING MODE	SET	91	SET MASK
	RESET	51	RESET MASK
SERIAL I/O MODE	SET	A0	SET MASK
	RESET	60	RESET MASK

Figure 28. Initialization/Configuration Command Summary

Receive Commands

The 8273 supports three receive commands plus a receiver disable function.

General Receive

When commanded to General Receive, the 8273 passes all frames either to memory (DMA mode) or to the CPU (non-DMA mode) regardless of the contents of the frame's address field. This command is used for primary and loop controller stations. Two parameters are required: B_0 and B_1. These parameters are the LSB and MSB of the receiver buffer size. Giving the 8273 this extra information alleviates the CPU of the burden of checking for buffer overflow. The 8273 will interrupt the CPU if the received frame attempts to overfill the allotted buffer space.

Selective Receive

In Selective Receive, two additional parameters besides B_0 and B_1 are required: A_1 and A_2. These parameters are two address match bytes. When commanded to Selective Receive, the 8273 passes to memory or the CPU only those frames having an address field matching either A_1 or A_2. This command is usually used for secondary stations with A_1 being the secondary address and A_2 is the "All Parties" address. If only one match byte is needed, A_1 and A_2 should be equal. As in General Receive, the 8273 counts the incoming data bytes and interrupts the CPU if B_0, B_1 is exceeded.

Selective Loop Receive

This command is very similar in operation to Selective Receive except that One Bit Delay mode must be set and that the loop is captured by placing transmitter in Flag Stream mode automatically after an EOP character is detected following a selectively received frame. The details of using the 8273 in loop configurations is discussed in a later section so please hold questions until then.

The handling of interrupt results is common among the three commands. When a frame is received without error, i.e., the FCS is correct and $\overline{CD}$ (Carrier Detect) was active throughout the frame or no attempt was made to overfill the buffer; the 8273 interrupts the CPU following the closing flag to pass the completion results. These results, in order, are the receiver interrupt result code (RIC), and the byte length of the information field of the received frame (R_0, R_1). If Buffered mode is selected, the address and control fields are passed as two additional results. If Buffered mode is not selected, the address and control fields are passed as the first two data transfers and R_0, R_1 reflect the information field length plus two.

Receive Disable

The receiver may also be disabled using the Receive Disable command. This command terminates any receive operation immediately. No parameters are required and no results are returned.

The details for the Receive command are shown in Figure 29. The interrupt result code key is shown in Figure 30. Some explanation of these result codes is appropriate.

The interrupt result code is the first byte passed to the CPU in the RxI/R register during the Result phase. Bits D_4-D_0 define the cause of the receiver interrupt. Since each result code has specific implications, they are discussed separately below.

COMMAND	HEX CODE	PARAMETERS	RESULTS* RxI/R
GENERAL RECEIVE	C0	B_0, B_1	RIC, R_0, R_1, A, C
SELECTIVE RECEIVE	C1	B_0, B_1, A_1, A_2	RIC, R_0, R_1, A, C
SELECTIVE LOOP RECEIVE	C2	B_0, B_1, A_1, A_2	RIC, R_0, R_1, A, C
DISABLE RECEIVER	C5	NONE	NONE

*A AND C ARE PASSED AS RESULTS ONLY IN BUFFERED MODE.

Figure 29. Receiver Command Summary

RIC D_7-D_0	RECEIVER INTERRUPT RESULT CODE	Rx STATUS AFTER INT
* 00000	A_1 MATCH OR GENERAL RECEIVE	ACTIVE
* 00001	A_2 MATCH	ACTIVE
000 00011	CRC ERROR	ACTIVE
000 00100	ABORT DETECTED	ACTIVE
000 00101	IDLE DETECTED	DISABLED
000 00110	EOP DETECTED	DISABLED
000 00111	FRAME < 32 BITS	ACTIVE
000 01000	DMA OVERRUN	DISABLED
000 01001	MEMORY BUFFER OVERFLOW	DISABLED
000 01010	CARRIER DETECT FAILURE	DISABLED
000 01011	RECEIVER INTERRUPT OVERRUN	DISABLED

*D_7-D_5	PARTIAL BYTE RECEIVED
111	ALL 8 BITS OF LAST BYTE
000	D_0
100	D_1-D_0
010	D_2-D_0
110	D_3-D_0
001	D_4D_{-0}
101	D_5-D_0
011	D_6-D_0

Figure 30. Receiver Interrupt Result Codes (RIC)

The first two result codes result from the error-free reception of a frame. If the frame is received correctly after a General Receive command, the first result is returned. If either Selective Receive command was used (normal or loop), a match with A_1 generates the first result code and a match with A_2 generates the second. In either case, the receiver remains active after the interrupt; however, the internal buffer size counters are not reset. That is, if the receive command indicated 100 bytes were allocated to the receive buffer (B_0, B_1) and an 80-byte frame was received correctly, the maximum next frame size that could be received without recommanding the receiver (resetting B_0 and B_1) is 20 bytes. Thus, it is common practice to recommand the receiver after each frame reception. DMA and/or memory pointers are usually updated at this time. (Note that users who do not wish to take advantage of the 8273's buffer management features may simply use B_0, B_1 = 0FFH for each receive command. Then frames of 65K bytes may be received without buffer overflow errors.)

The third result code is a CRC error. This indicates that a frame was received in the correct format (flags, etc.); however, the received FCS did not check with the internally generated FCS. The frame should be discarded. The receiver remains active. (Do not forget that even though an error condition has been detected, all frame information up until that error has either been transferred to memory or passed to the CPU. This information should be invalidated. This applies to all receiver error conditions.) Note that the FCS, either transmitted or received, is never available to the CPU.

The Abort Detect result occurs whenever the receiver sees either an SDLC (8 1s) or an HDLC (7 1s), depending on the Operating Mode register. However, the intervening Abort character between a closing flag and an Idle does not generate an interrupt. If an Abort character (seen by an active receiver within a frame) is not preceded by a flag and is followed by an Idle, an interrupt will be generated for the Abort, followed by an Idle interrupt one character time later. The Idle Detect result occurs whenever 15 consecutive 1s are received. After the Abort Detect interrupt, the receiver remains active. After the Idle Detect interrupt, the receiver is disabled and must be recommanded before further frames may be received.

If the EOP Interrupt bit is set in the Operating Mode register, the EOP Detect result is returned whenever an EOP character is received. The receiver is disabled, so the Idle following the EOP does not generate an Idle Detect interrupt.

The minimum number of bits in a valid frame between the flags is 32. Fewer than 32 bits indicates an error. If Buffered mode is selected, such frames are ignored, i.e., no data transfers or interrupts are generated. In non-Buffered mode, a < 32-bit frame generates an interrupt with the < 32-bit Frame result since data transfers may already have disturbed the 8257 or interrupt handler. The receiver remains active.

The DMA Overrun result results from the $\overline{DMA}$ controller being too slow in extracting data from the 8273, i.e., the $\overline{RxDACK}$ signal is not returned before the next received byte is ready for transfer. The receiver is disabled if this error condition occurs.

The Memory Buffer Overflow result occurs when the number of received bytes exceeds the receiver buffer length supplied by the B_0 and B_1 parameters in the receive command. The receiver is disabled.

The Carrier Detect Failure result occurs when the $\overline{CD}$ pin goes high (inactive) during reception of a frame. The $\overline{CD}$ pin is used to qualify reception and must be active by the time the address field starts to be received. If $\overline{CD}$ is lost during the frame, a $\overline{CD}$ Failure interrupt is generated and the receiver is disabled. No interrupt is generated if $\overline{CD}$ goes inactive between frames.

If a condition occurs requiring an interrupt be generated before the CPU has finished reading the previous interrupt results, the second interrupt is generated after the current Result phase is complete (the RxINT pin and status bit go low then high). However, the interrupt result for this second interrupt will be a Receive Interrupt Overrun. The actual cause of the second interrupt is lost. One case where this may occur is at the end of a received frame where the line goes idle. The 8273 generates a received frame interrupt after the closing flag and then 15-bit times later, generates an Idle Detect interrupt. If the interrupt service routine is slow in reading the first interrupt's results, the internal RxI/R register still contains result information when the Idle Detect interrupt occurs. Rather than wiping out the previous results, the 8273 adds a Receive Interrupt Overrun result as an extra result. If the system's interrupt structure is such that the second interrupt is not acknowledged (interrupts are still disabled from the first interrupt), the Receive Interrupt Overrun result is read as an extra result, after those from the first interrupt. If the second interrupt is serviced, the Receive Interrupt Overrun is returned as a single result. (Note that the INT pins supply the necessary transitions to support a Programmable Interrupt Controller such as the Intel 8259. Each interrupt generates a positive-going edge on the appropriate INT pin and the high level is held until the interrupt is completely serviced.) In general, it is possible to have interrupts occurring at one character time intervals. Thus the interrupt handling software must have at least that much response and service time.

The occurrence of Receive Interrupt Overruns is an indication of marginal software design; the system's interrupt response and servicing time is not sufficient for the data rates being attempted. It is advisable to configure the interrupt handling software to simply read the interrupt results, place them into a buffer, and clear the interrupt as quickly as possible. The software can then examine the buffer for new results at its leisure, and take appropriate action. This can easily be accomplished by using a result buffer flag that indicates when new results are available. The interrupt handler sets the flag and the main program resets it once the results are retrieved.

Both SDLC and HDLC allow frames which are of arbitrary length (>32 bits). The 8273 handles this N-bit reception through the high order bits (D_7–D_5) of the result code. These bits code the number of valid received bits in the last received information field byte. This coding is shown in Figure 30. The high order bits of the received partial btye are indeterminate. [The address, control, and information fields are transmitted least significant bit (A_0) first. The FCS is complemented and transmitted most significant bit first.]

Transmit Commands

The 8273 transmitter is supported by three Transmit commands and three corresponding Abort commands.

Transmit Frame

The Transmit Frame command simply transmits a frame. Four parameters are required when Buffered mode is selected and two when it is not. In either case, the first two parameters are the least and the most significant bytes of the desired frame length (L_0, L_1). In Buffered mode, L_0 and L_1 equal the length in bytes of the desired information field, while in the non-Buffered mode, L_0 and L_1 must be specified as the information field length plus two. (L_0 and L_1 specify the number of data transfers to be performed.) In Buffered mode, the address and control fields are presented to the transmitter as the third and fourth parameters respectively. In non-Buffered mode, the A and C fields must be passed as the first two data transfers.

When the Transmit Frame command is issued, the 8273 makes $\overline{RTS}$ (Request-to-Send) active (pin low) if it was not already. It then waits until $\overline{CTS}$ (Clear-to-Send) goes active (pin low) before starting the frame. If the Preframe Sync bit in the Operting Mode register is set, the transmitter prefaces two characters (16 transitions) before the opening flag. If the Flag Stream bit is set in the Operating Mode register, the frame (including Preframe Sync if selected) is started on a flag boundary. Otherwise the frame starts on a character boundary.

At the end of the frame, the transmitter interrupts the CPU (the interrupt results are discussed shortly) and returns to either Idle or Flag Stream, depending on the Flag Stream bit of the Operating Mode register. If $\overline{RTS}$ was active before the transmit command, the 8273 does not change it. If it was inactive, the 8273 will deactivate it within one character time.

Loop Transmit

Loop Transmit is similar to Frame Transmit (the parameter definition is the same). But since it deals with loop configurations, One Bit Delay mode must be selected.

If the transmitter is not in Flag Stream mode when this command is issued, the transmitter waits until after a received EOP character has been converted to a flag (this is done automatically) before transmitting. (The one bit delay is, of course, suspended during transmit.) If the transmitter is already in Flag Stream mode as a result of a selectively received frame during a Selective Loop Receive command, transmission will begin at the next flag boundary for Buffered mode or at the third flag boundary for non-Buffered mode. This discrepancy is to allow time for enough data transfers to occur to fill up the internal transmit buffer. At the end of a Loop Transmit, the One Bit Delay mode is re-entered and the flag stream mode is reset. More detailed loop operation is covered later.

Transmit Transparent

The Transmit Transparent command enables the 8273 to transmit a block of raw data. This data is without SDLC protocol, i.e., no zero bit insertion, flags, or FCS. Thus it is possible to construct and transmit a Bi-Sync message for front-end processor switching or to construct and transmit an SDLC message with incorrect FCS for diagnostic purposes. Only the L_0 and L_1 parameters are used since there are not fields in this mode. (the 8273 does not support a Receive Transparent command.)

Abort Commands

Each of the above transmit commands has an associated Abort command. The Abort Frame Transmit command causes the transmitter to send eight contiguous ones (no zero bit insertion) immediately and then revert to either idle or flag streaming based on the Flag Stream bit. (The 8 1s as an Abort character is compatible with both SDLC and HDLC.)

For Loop Transmit, the Abort Loop Transmit command causes the transmitter to send one flag and then revert to one bit delay. Loop protocol depends upon FCS errors to detect aborted frames.

The Abort Transmit Transparent simply causes the transmitter to revert to either idles or flags as a function of the Flag Stream mode specified.

The Abort commands require no parameters, however, they do generate an interrupt and return a result when complete.

A summary of the Transmit commands is shown in Figure 31. Figure 32 shows the various transmit interrupt result codes. As in the receiver operation, the transmitter generates interrupts based on either good completion of an operation or an error condition to start the Result phase.

The Early Transmit Interrupt result occurs after the last data transfer to the 8273 if the Early Transmit Interrupt bit is set in the Operating Mode register. If the 8273 is commanded to transmit again within two character times, a single flag will separate the frames. (Buffered mode must be used for a single flag to separate the frames. If non-Buffered mode is selected, three flags will separate the frames.) If this time constraint is not met, another interrupt is generated and multiple flags or idles will separate the frames. The second interrupt is the normal Frame Transmit Complete interrupt. The Frame Transmit Complete result occurs at the closing flag to signify a good completion.

The DMA Underrun result is analogous to the DMA Overrun result in the receiver. Since SDLC does not support intraframe time fill, if the DMA controller or CPU does not supply the data in time, the frame must be aborted. The action taken by the transmitter on this error is automatic. It aborts the frame just as if an Abort command had been issued.

Clear-to-Send Error result is generated if $\overline{CTS}$ goes inactive during a frame transmission. The frame is aborted as above.

The Abort Complete result is self-explanatory. Please note however that no Abort Complete interrupt is generated when an automatic abort occurs. The next command type consists of only one command.

COMMAND	HEX CODE	PARAMETERS*	RESULTS TxI/R
TRANSMIT FRAME	C8	L_0, L_1, A, C	TIC
ABORT	CC	NONE	TIC
LOOP TRANSMIT	CA	L_0, L_1, A, C	TIC
ABORT	CE	NONE	TIC
TRANSMIT TRANSPARENT	C9	L_0, L_1	TIC
ABORT	CD	NONE	TIC

*A AND C ARE PASSED AS PARAMETERS IN BUFFERED MODE ONLY.

Figure 31. Transmitter Command Summary

TIC D_7-D_0	TRANSMITTER INTERRUPT RESULT CODE	Tx STATUS AFTER INT
000 01100	EARLY Tx INTERRUPT	ACTIVE
000 01101	FRAME Tx COMPLETE	IDLE OR FLAGS
000 01110	DMA UNDERRUN	ABORT
000 01111	CLEAR TO SEND ERROR	ABORT
000 10000	ABORT COMPLETE	IDLE OR FLAGS

Figure 32. Transmitter Interrupt Result Codes

Reset Command

The Reset command provides a software reset function for the 8273. It is a special case and does not utilize the normal command interface. The reset facility is provided in the Test Mode register. The 8273 is reset by simply outputting a 01H followed by a 00H to the Test Mode register. Writing the 01 followed by the 00 mimicks the action required by the hardware reset. Since the 8273 requires time to process the reset internally, at least 10 cycles of the ØCLK clock must occur between the writing of the 01 and the 00. The action taken is the same as if a hardware reset is performed, namely:

1. The modem control outputs are forced high inactive).
2. The 8273 Status register is cleared.
3. Any commands in progress cease.
4. The 8273 enters an idle state until the next command is issued.

Modem Control Commands

The modem control ports were discussed earlier in the Hardware section. The commands used to manipulate these ports are shown in Figure 33. The Read Port A and Read Port B commands are immediate. The bit definition for the returned byte is shown in Figures 13 and 14. Do not forget that the returned value represents the logical condition of the pin, i.e., pin active (low) = bit set.

PORT	COMMAND	HEX CODE	PARAMETER	REG RESULT
A INPUT	READ	22	NONE	PORT VALUE
B OUTPUT	READ	23	NONE	PORT VALUE
	SET	A3	SET MASK	NONE
	RESET	63	RESET MASK	NONE

Figure 33. Modem Control Command Summary

The Set and Reset Port B commands are similar to the initialization commands in that they use a mask parameter which defines the bits to be changed. Set Port B utilizes a logical OR mask and Reset Port B uses a logical AND mask. Setting a bit makes the pin active (low). Resetting the bit deactivates the pin (high).

To help clarify the numerous timing relationships that occur and their consequences, Figures 34 and 35 are provided as an illustration of several typical sequences. It is suggested that the reader go over these diagrams and re-read the appropriate part of the previous sections if necessary.

HLDC CONSIDERATIONS

The 8273 supports HDLC as well as SDLC. Let's discuss how the 8273 handles the three basic HDLC/SDLC differences: extended addressing, extended control, and the 7 1s Abort character.

Recalling Figure 4A, HDLC supports an address field of indefinite length. The actual amount of extension used is determined by the least significant bit of the characters immediately following the opening flag. If the LSB is 0, more address field bytes follow. If the LSB is 1, this byte is the final address field byte. Software must be used to determine this extension.

If non-Buffered mode is used, the A, C, and I fields are in memory. The software must examine the initial characters to find the extent of the address field. If Buffered mode is used, the characters corresponding to the SDLC A and C fields are transferred to the CPU as interrupt results. Buffered mode assumes the two characters following the opening flag are to be transferred as interrupt results regardless of content or meaning. (The 8273 does not know whether it is being used in an SDLC or an HDLC environment.) In SDLC, these characters are necessarily the A and C field bytes, however in HDLC, their meaning may change depending on the amount of extension used. The software must recognize this and examine the transferred results as possible address field extensions.

Frames may still be selectively received as is needed for secondary stations. The Selective Receive command is still used. This command qualifies a frame reception on the first byte following the opening flag matching either of the A_1 or A_2 match byte parameters. While this does not allow qualification over the complete range of HDLC addresses, it does perform a qualification on the first address byte. The remaining address field bytes, if any, are then examined via software to completely qualify the frame.

Once the extent of the address field is found, the following bytes form the control field. The same LSB test used for the address field is applied to these bytes to determine the control field extension, up to two bytes maximum. The remaining frame bytes in memory represent the information field.

The Abort character difference is handled in the Operating Mode register. If the HDLC Abort Enable bit is set, the reception of seven contiguous ones by an active receiver will generate an Abort Detect interrupt rather than eight ones. (Note that both the HDLC Abort Enable bit and the EOP Interrupt bit must not be set simultaneously.)

Now let's move on to the SDLC loop configuration discussion.

LOOP CONFIGURATION

Aside from use in the normal data link applications, the 8273 is extremely attractive in loop configuration due to the special frame-level loop commands and the Digital Phase Locked Loop. Toward this end, this section details the hardware and software considerations when using the 8273 in a loop application.

The loop configuration offers a simple, low-cost solution for systems with multiple stations within a small physical location, i.e., retail stores and banks. There are two primary reasons to consider a loop configuration. The interconnect cost is lower for a loop over a multipoint configuration since only one twisted pair or fiber optic cable is used. (The loop configuration does not support the passing of distinct clock signals from station to station.) In addition, loop stations do not need the intelligence of a multi-point station since the loop protocal is simpler. The most difficult aspects of loop station design are clock recovery and implementation of one bit delay (both are handled neatly by the 8273).

Figure 36 illustrates a typical loop configuration with one controller and two down-loop secondaries. Each station must derive its own data timing from the received data stream. Recalling our earlier discussion of the DPLL, notice that $\overline{TxC}$ and $\overline{RxC}$ clocks are provided by the DPLL output. The only clock required in the secondaries is a simple, non-synchronized clock at 32 times the desired baud rate. The controller requires both 32× and 1× clocks. (The 1× is usually implemented by dividing the 32× clock with a 5-bit divider. However, there is no synchronism requirement between these clocks so any convenient implementation may be used.)

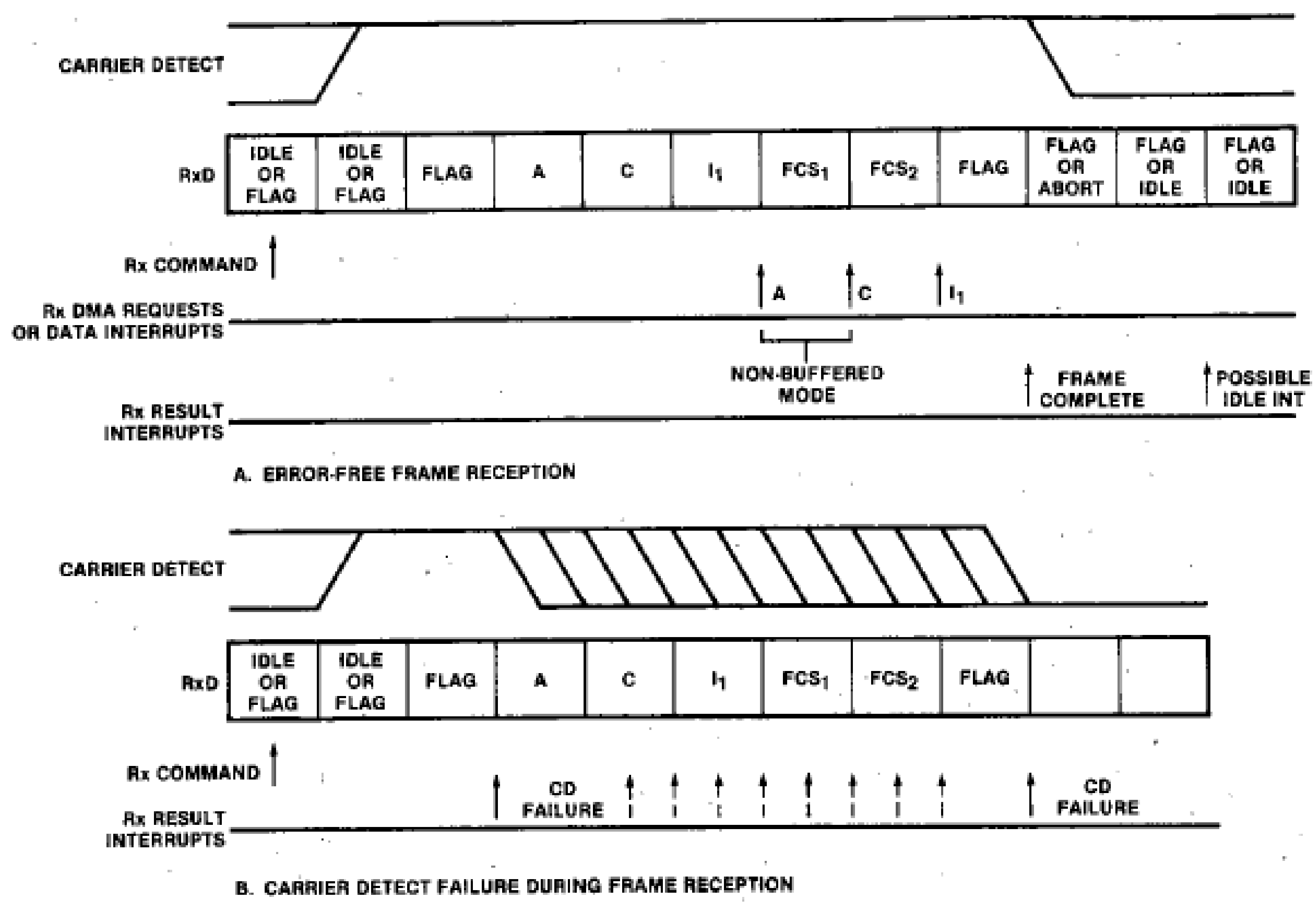

Figure 34. Sample Receiver Timing Diagrams

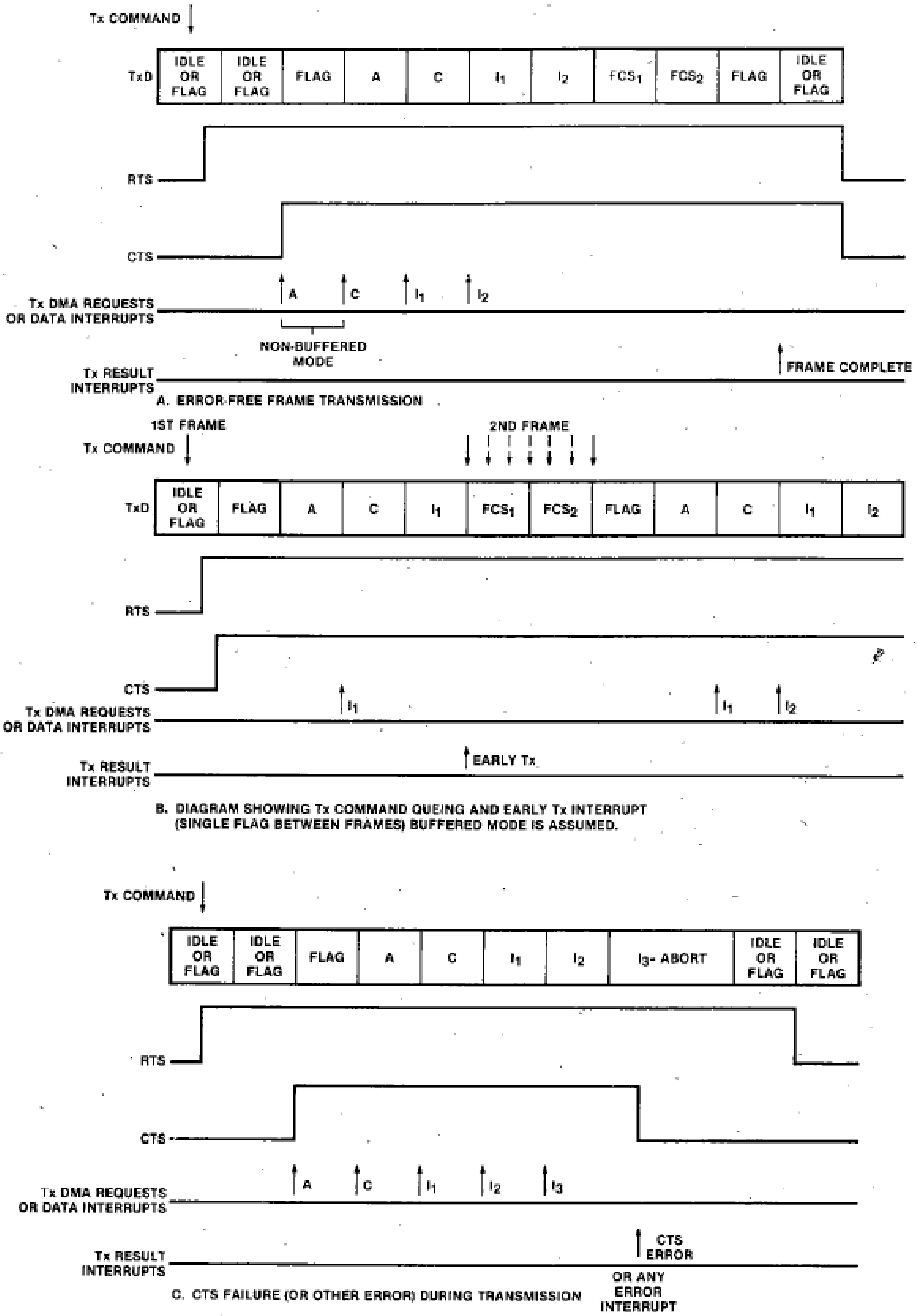

Figure 35. Sample Transmitter Timing Diagrams

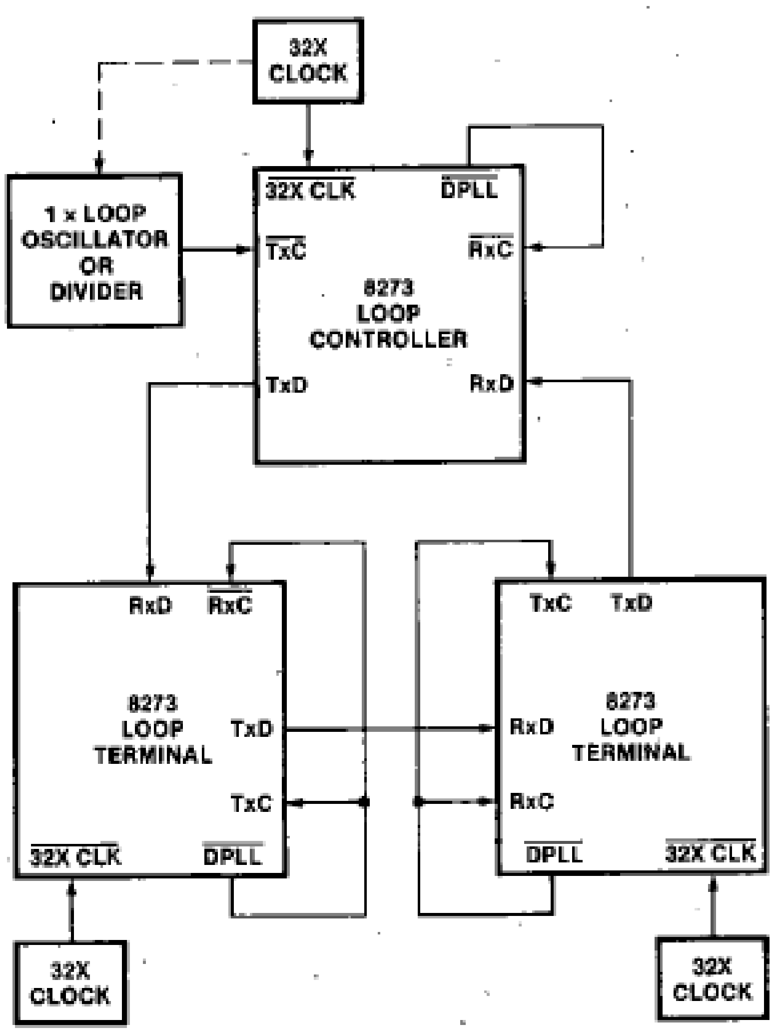

Figure 36. SDLC Loop Application

A quick review of loop protocol is appropriate. All communication on the loop is controlled by the loop controller. When the controller wishes to allow the secondaries to transmit, it sends a polling frame (the control field contains a poll code) followed by an EOP (End-of-Poll) character. The secondaries use the EOP character to capture the loop and insert a response frame as will be discussed shortly.

The secondaries normally operate in the repeater mode, retransmitting received data with one bit time of delay. All received frames are repeated. The secondary uses the one bit time of delay to capture the loop.

When the loop is idle (no frames), the controller transmits continuous flag characters. This keeps transitions on the loop for the sake of down-loop phase locked loops. When the controller has a non-polling frame to transmit, it simply transmits the frame and continues to send flags. The non-polling frame is then repeated around the loop and the controller receives it to signify a complete traversal of the loop. At the particular secondary addressed by the frame, the data is transferred to memory while being repeated. Other secondaries simply repeat it.

If the controller wants to poll the secondaries, it transmits a polling frame followed by all 1s (no zero bit insertion). The final zero of the closing frame plus the first seven 1s form an EOP. While repeating, the secondaries monitor their incoming line for an EOP. When an EOP is received, the secondary checks if it has any response for the controller. If not, it simply continues repeating. If the secondary has a response, it changes the seventh EOP one into a zero (the one bit time of delay allows time for this) and repeats it, forming a flag for the down-loop stations. After this flag is transmitted, the secondary terminates its repeater function and inserts its response frame (with multiple preceding flags if necessary). After the closing flag of the response, the secondary re-enters its repeater function, repeating the up-loop controller 1s. Notice that the final zero of the response's closing flag plus the repeated 1s from the controller form a new EOP for the next down-loop secondary. This new EOP allows the next secondary to insert a response if it desires. This gives each secondary a chance to respond.

Back at the controller, after the polling frame has been transmitted and the continuous 1s started, the controller waits until it receives an EOP. Receiving an EOP signifies to the controller that the original frame has propagated around the loop followed by any responses inserted by the secondaries. At this point, the controller may either send flags to idle the loop or transmit the next frame. Let's assume that the loop is implemented completely with the 8273s and describe the command flows for a typical controller and secondary.

The loop controller is initialized with commands which specify that the NRZI, Preframe Sync, Flag Stream, and EOP Interrupt modes are set. Thus, the controller encodes and decodes all data using NRZI format. Preframe Sync mode specifies that all transmitted frames be prefaced with 16 line transitions. This ensures that the minimum of 12 transitions needed by the DPLLs to lock after an all 1s line have occurred by the time the secondary sees a frame's opening flag. Setting the Flag Stream mode starts the transmitter sending flags which idles the loop. And the EOP Interrupt mode specifies that the controller processor will be interrupted whenever the active receiver sees an EOP, indicating the completion of a poll cycle.

When the controller wishes to transmit a non-polling frame, it simply executes a Frame Transmit command. Since the Flag Stream mode is set, no EOP is formed after the closing flag. When a polling frame is to be transmitted, a General Receive command is executed first. This enables the receiver and allows reception of all incoming frames; namely, the original polling frame plus any response frames inserted by the secondaries. After the General Receive command, the frame is transmitted with a Frame Transmit command. When the frame is complete, a transmitter interrupt is generated. The loop controller processor uses this interrupt to reset Flag Stream mode. This causes the transmitter to start sending all 1s. An EOP is formed by the last flag and the first 7 1s. This completes the loop controller transmit sequence.

At any time following the start of the polling frame transmission the loop controller receiver will start receiving frames. (The exact time difference depends, of course, on the number of down-loop secondaries due to each inserting one bit time of delay.) The first received frame is simply the original polling frame. However, any additional frames are those inserted by the secondaries. The loop controller processor knows all frames have been received when it sees an EOP Interrupt. This interrupt is generated by the 8273 since the EOP Interrupt mode was set during initialization. At this point, the transmitter may be commanded either to enter Flag

Stream mode, idling the loop, or to transmit the next frame. A flowchart of the above sequence is shown in Figure 37.

The secondaries are initialized with the NRZI and One Bit Delay modes set. This puts the 8273 into the repeater mode with the transmitter repeating the received data with one bit time of delay. Since a loop station cannot transmit until it sees and EOP character, any transmit command is queued until an EOP is received. Thus whenever the secondary wishes to transmit a response, a Loop Transmit command is issued. The 8273 then waits until it receives an EOP. At this point, the receiver changes the EOP into a flag, repeats it, resets One Bit Delay mode stopping the repeater function, and sets the transmitter into Flag Stream mode. This captures the loop. The transmitter now inserts its message. At the closing flag, Flag Stream mode is reset, and One Bit Delay mode is set, returning the 8273 to repeater function and forming an EOP for the next down-loop station. These actions happen automatically after a Loop Transmit command is issued.

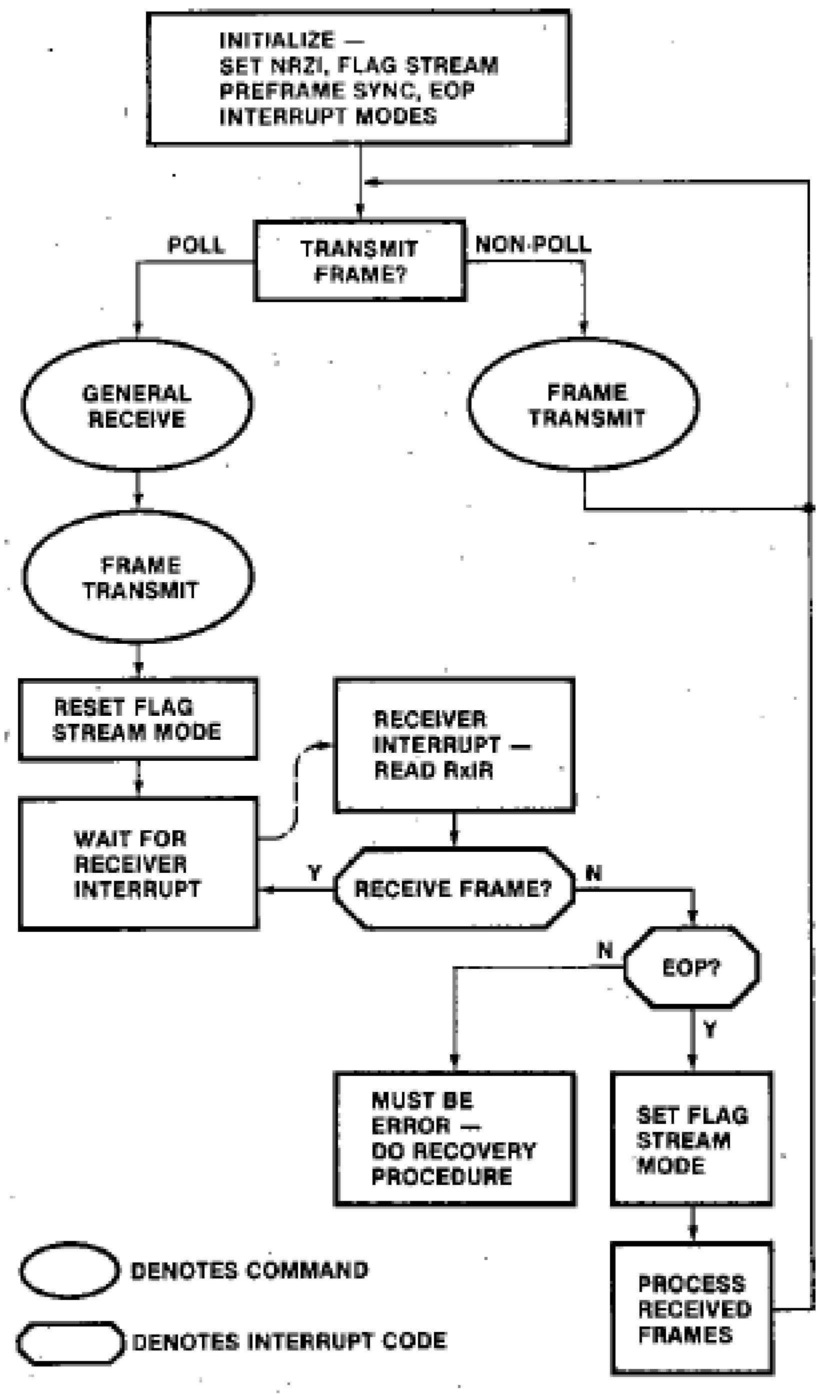

Figure 37. Loop Controller Flowchart

When the secondary wants its receiver enabled, a Selective Loop Receive command is issued. The receiver then looks for a frame having a match in the Address field. Once such a frame is received, repeated, and transferred to memory, the secondary's processor is interrupted with the appropriate Match interrupt result and the 8273 continues with the repeater function until an EOP is received, at which point the loop is captured as above. The processor should use the interrupt to determine if it has a message for the controller. If it does, it simply issues a Loop Transmit command and things progress as above. If the processor has no message, the software must reset the Flag Stream mode bit in the Operating Mode register. This will inhibit the 8273 from capturing the loop at the EOP. (The match frame and the EOP may be separated in time by several frames depending on how many up-loop stations inserted messages of their own.) If the timing is such that the receiver has already captured the loop when the Flag Stream mode bit is reset, the mode is exited on a flag boundary and the frame just appears to have extra closing flags before the EOP. Notice that the 8273 handles the queuing of the transmit commands and the setting and resetting of the mode bits automatically. Figure 38 illustrates the major points of the secondary command sequence.

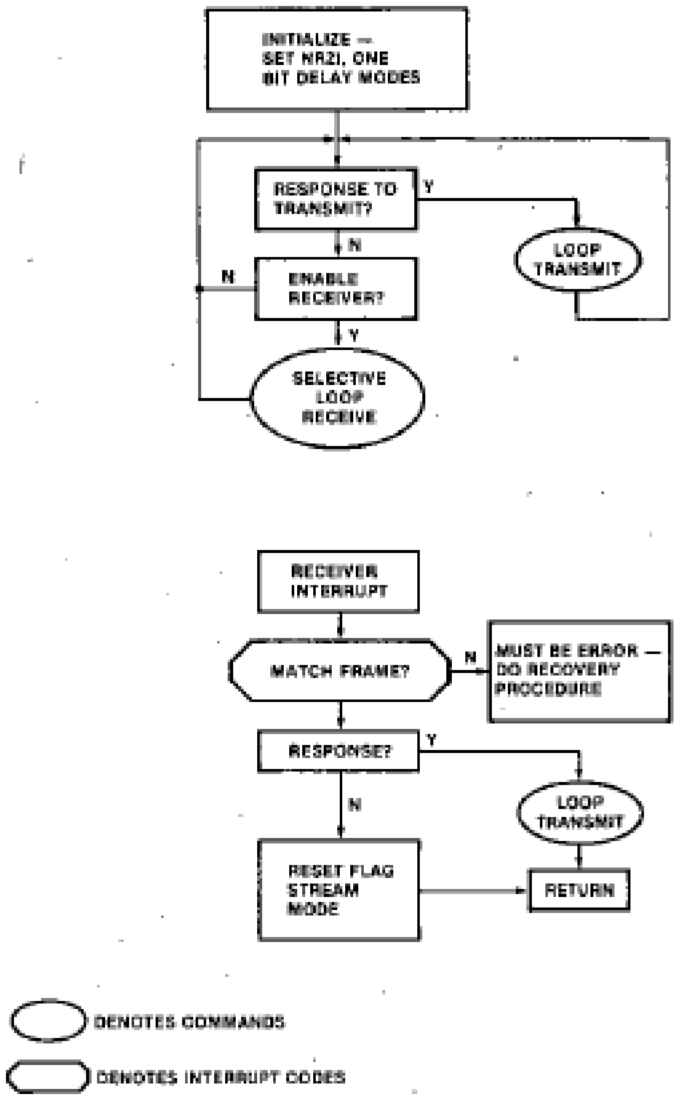

Figure 38. Loop Secondary Flowchart

When an off-line secondary wishes to come on-line, it must do so in a manner which does not disturb data on the loop. Figure 39 shows a typical hardware interface. The line labeled Port could be one of the 8273 Port B outputs and is assumed to be high (1) initially. Thus up-loop data is simply passed down-loop with no delay; however, the receiver may still monitor data on the loop. To come on-line, the secondary is initialized with only the EOP Interrupt mode set. The up-loop data is then monitored until an EOP occurs. At this point, the secondary's CPU is interrupted with an EOP interrupt. This signals the CPU to set One Bit Delay mode in the 8273 and then to set Port low (active). These actions switch the secondary's one bit delay into the loop. Since after the EOP only 1s are traversing the loop, no loop disturbance occurs. The secondary now waits for the next EOP, captures the loop, and inserts a "new on-line" message. This signals the controller that a new secondary exists and must be acknowledged. After the secondary receives its acknowledgement, the normal command flow is used.

It is hopefully evident from the above discussion that the 8273 offers a very simple and easy to implement solution for designing loop stations whether they are controllers or down-loop secondaries.

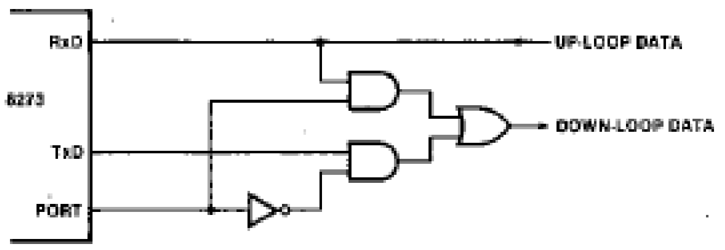

Figure 39. Loop Interface

APPLICATION EXAMPLE

This section describes the hardware and software of the 8273/8085 system used to verify the 8273 implementation of SDLC on an actual IBM SDLC Link. This IBM link was gratefully volunteered by Raytheon Data Systems in Norwood, Mass. and I wish to thank them for their generous cooperation. The IBM system consisted of a 370 Mainframe, a 3705 Communications Processor, and a 3271 Terminal Controller. A Comlink II Modem supplied the modem interface and all communications took place at 4800 baud. In addition to observing correct responses, a Spectron D601B Datascope was used to verify the data exchanges. A block diagram of the system is shown in Figure 40. The actual verification was accomplished by the 8273 system receiving and responding to polls from the 3705. This method was used on both point-to-point and multi-point configurations. No attempt was made to implement any higher protocol software over that of the poll and poll responses since such software would not affect the verification of the 8273 implementation. As testimony to the ease of use of the 8273, the system worked on the first try.

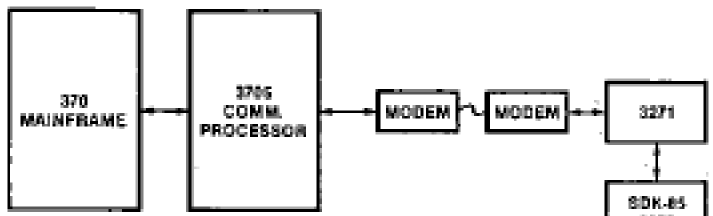

Figure 40. Raytheon Block Diagram

An SDK-85 (System Design Kit) was used as the core 8085 system. This system provides up to 4K bytes of ROM/EPROM, 512 bytes of RAM, 76 I/O pins, plus two timers as provided in two 8755 Combination EPROM/I/O devices and two 8155 Combination RAM/I/O/Timer devices. In addition, 5 interrupt inputs are supplied on the 8085. The address, data, and control buses are buffered by the 8212 and 8216 latches and bidirectional bus drivers. Although it was not used in this application, an 8279 Display Driver/Keyboard Encoder is included to interface the on-board display and keyboard. A block diagram of the SDK-85 is shown in Figure 41. The 8273 and associated circuitry was constructed on the ample wire-wrap area provided for the user.

The example 8273/8085 system is interrupt-driven and uses DMA for all data transfers supervised by an 8257 DMA Controller. A 2400 baud asynchronous line, implemented with an 8251A USART, provides communication between the software and the user. 8253 Programmable Interval Timer is used to supply the baud rate clocks for the 8251A and 8273. (The 8273 baud rate clocks were used only during initial system debug. In actual operation, the modem supplied these clocks via the RS-232 interface.) Two 2142 1K × 4 RAMs provided 512 bytes of transmitter and 512 bytes of receiver buffer memory. (Command and result buffers, plus miscellaneous variables are stored in the 8155s.) The RS-232 interface utilized MC1488 and MC1489 RS-232 drivers and receivers. The schematic of the system is shown in Figure 42.

One detail to note is the DMA and interrupt structure of the transmit and receive channels. In both cases, the receiver is always given the higher priority (8257 DMA channel 0 has priority over the remaining channels and the 8085 RST 7.5 interrupt input has priority over the RST 6.5 input.) Although the choice is arbitrary, this technique minimizes the chance that received data could be lost due to other processor or DMA commitments.

Also note that only one 8205 Decoder is used for both the peripherals' and the memorys' Chip Selects. This was done to eliminate separate memory and I/O decoders since it was known beforehand that neither address space would be completely filled.

The 4 MHz crystal and 8224 Clock Generator were used only to verify that the 8273 operates correctly at that maximum spec speed. In a normal system, the 3.072 MHz clock from the 8085 would be sufficient. (This fact was verified during initial checkout.)

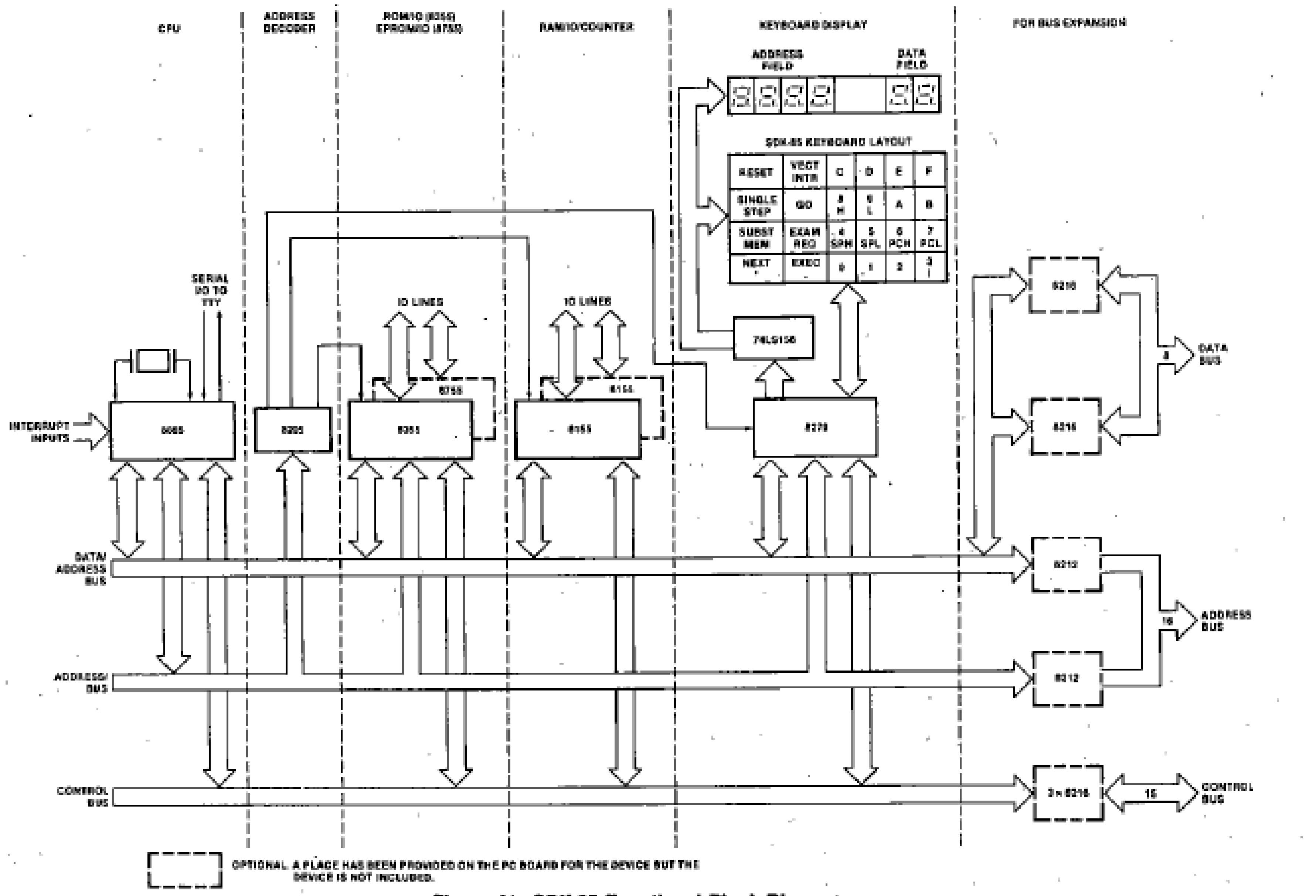

Figure 41. SDK-85 Functional Block Diagram

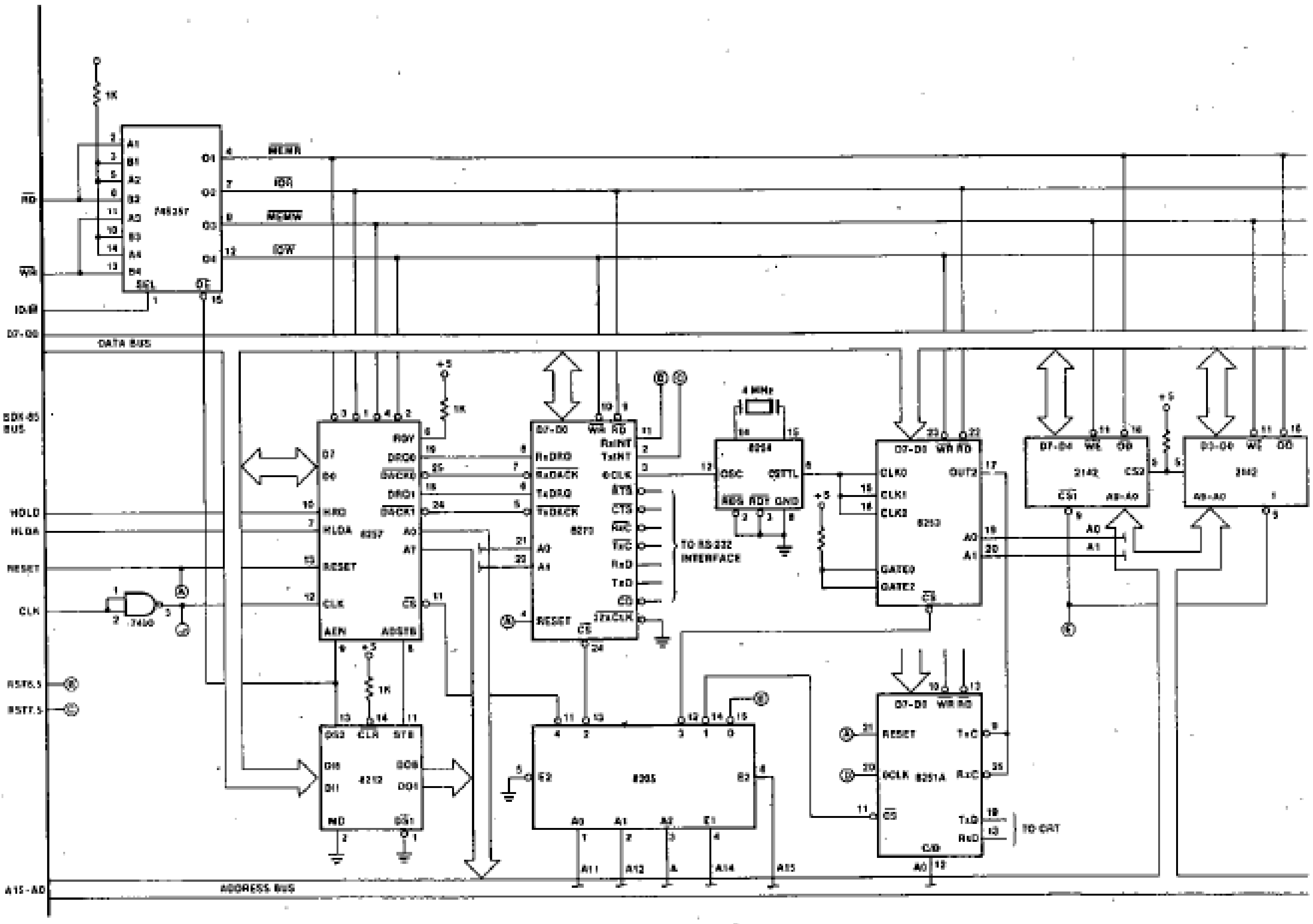

Figure 42. 8273/SDK-85 System

The software consists of the normal monitor program supplied with the SDK-85 and a program to input commands to the 8273 and to display results. The SDK-85 monitor allows the user to read and write on-board RAM, start execution at any memory location, to single-step through a program, and to examine any of the 8085's internal registers. The monitor drives either the on-board keyboard/LED display or a serial TTY interface. This monitor was modified slightly in order to use the 8251A with a 2400 baud CRT as opposed to the 110 baud normally used. The 8273 program implements monitor-like user interface. 8273 commands are entered by a two-character code followed by any parameters required by that command. When 8273 interrupts occur, the source of the interrupt is displayed along with any results associated with it. To gain a flavor of how the user/program interface operates, a sample output is shown in Figure 43. The 8273 program prompt character is a "–" and user inputs are underlined.

The "SO 05" implements the Set Operating Mode command with a parameter of 05H. This sets the Buffer and Flag Stream modes. "SS 01" sets the 8273 in NRZI mode using the Set Serial I/O Mode command. The next command specifies General Receiver with a receiver buffer size of 0100H bytes ($B_0 = 00$, $B_1 = 01$). The "TF" command causes the 8273 to transmit a frame containing an address field of C2H and control field of 11H. The information field is 001122. The "TF" command has a special format. The L_0 and L_1 parameters are computed from the number of information field bytes entered.

After the TF command is entered, the 8273 transmits the frame (assuming that the modem protocol is observed). After the closing flag, the 8273 interrupts the 8085. The 8085 reads the interrupt results and places them in a buffer. The software examines this buffer for new results and if new results exist, the source of the interrupt is displayed along with the results.

In this example, the 0DH result indicates a Frame Complete interrupt. There is only one result for a transmitter interrupt, the interrupt's trailing zero results were included to simplify programming.

The next event is a frame reception. The interrupt results are displayed in the order read from the 8273. The E0H indicates a General Receive interrupt with the last byte of the information field received on an 8-bit boundary. The 03 00 (R_0, R_1) results show that there are 3H bytes of information field received. The remaining two results indicate that the received frame had a C2H address field and a 34H control field. The 3 bytes of information field are displayed on the next line.

```
8273 MONITOR V1.2

— SO 05
— SS 01
— GR 00 01
— TF C2 11 00 11 22
—
TxINT — 0D 00 00 00 00
—
RxINT — E0 03 00 C2 34
FF EE DD
—
```

Figure 43. Sample 8273 Monitor I/O

Figures 44 through 51 show the flowcharts used for the 8273 program development. The actual program listing is included as Appendix A. Figure 44 is the main status poll loop. After all devices are initialized and a prompt character displayed, a loop is entered at LOOPIT. This loop checks for a change of status in the result buffer or if a keyboard character has been received by the 8251 or if a poll frame has been received. If any of these conditions are met, the program branches to the appropriate routine. Otherwise, the loop is traversed again.

The result buffer is implemented as a 255-byte circular buffer with two pointers: CNADR and LDADR. CNADR is the console pointer. It points to the next result to be displayed LDADR is the load pointer. It points to the next empty position in the buffer into which the interrupt handler places the next result. The same buffer is used for both transmitter and receiver results. LOOPIT examines these pointers to detect when CNADR is not equal to LDADR indicating that the buffer contains results which have not been displayed. When this occurs, the program branches to the DISPLY routine.

DISPLY determines the source of the undisplayed results by testing the first result. This first result is necessarily the interrupt result code. If this result is 0CH or greater, the result is from a transmitter interrupt. Otherwise it is from a receiver source. The source of the result code is then displayed on the console along with the next four results from the buffer. If the source was a transmitter interrupt, the routine merely repoints the pointer CNADR and returns to LOOPIT. For a receiver source, the receiver data buffer is displayed in addition to the receiver interrupt results before returning to LOOPIT.

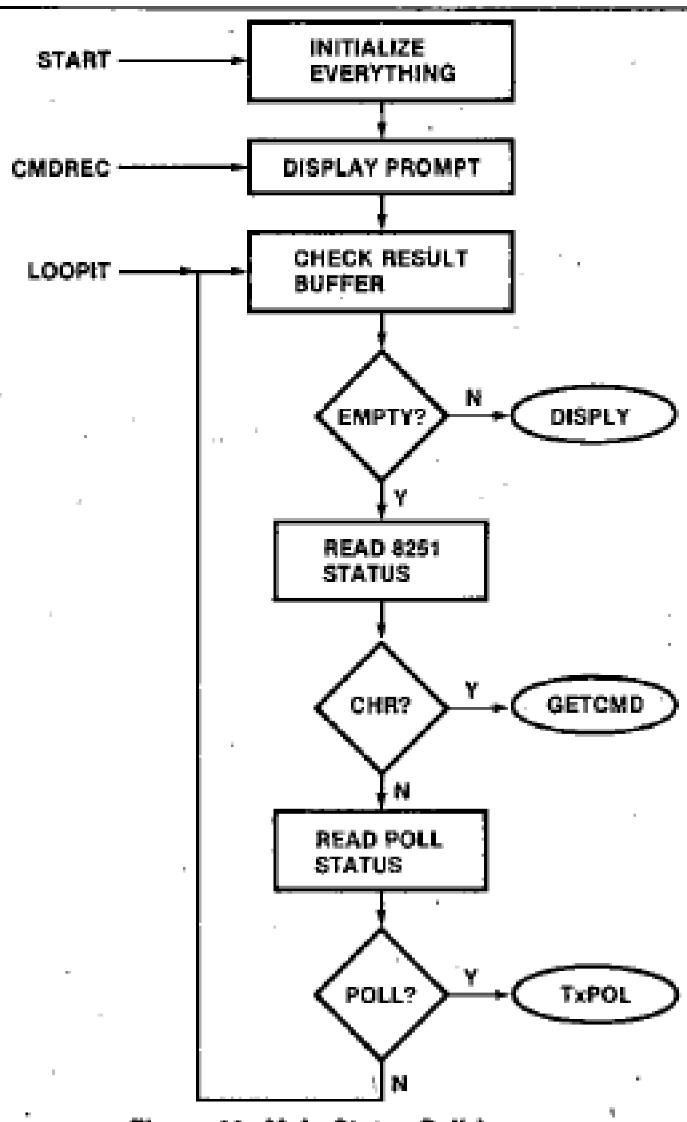

Figure 44. Main Status Poll Loop

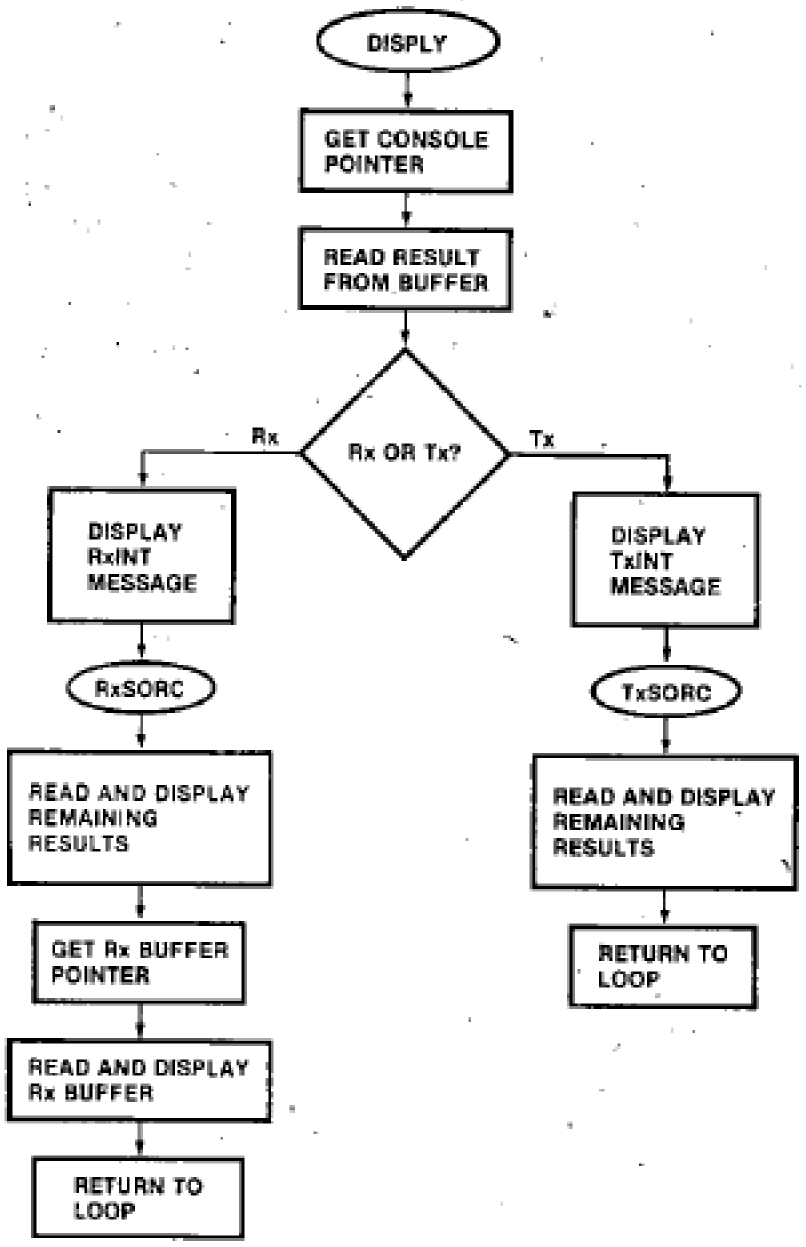

Figure 45. DISPLY Subroutine

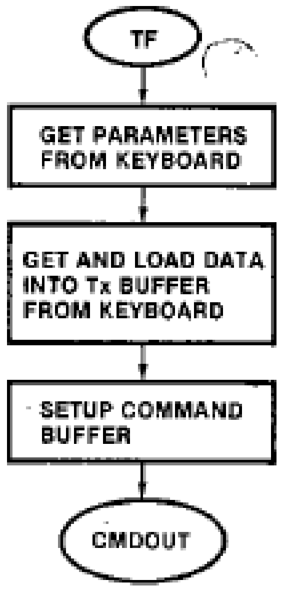

Figure 47. TF Subroutine

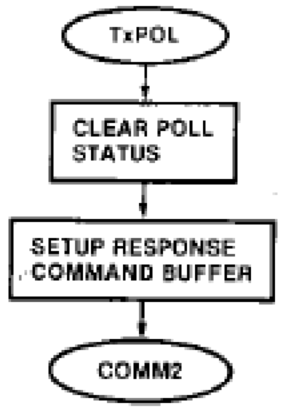

Figure 48. TxPOL Subroutine

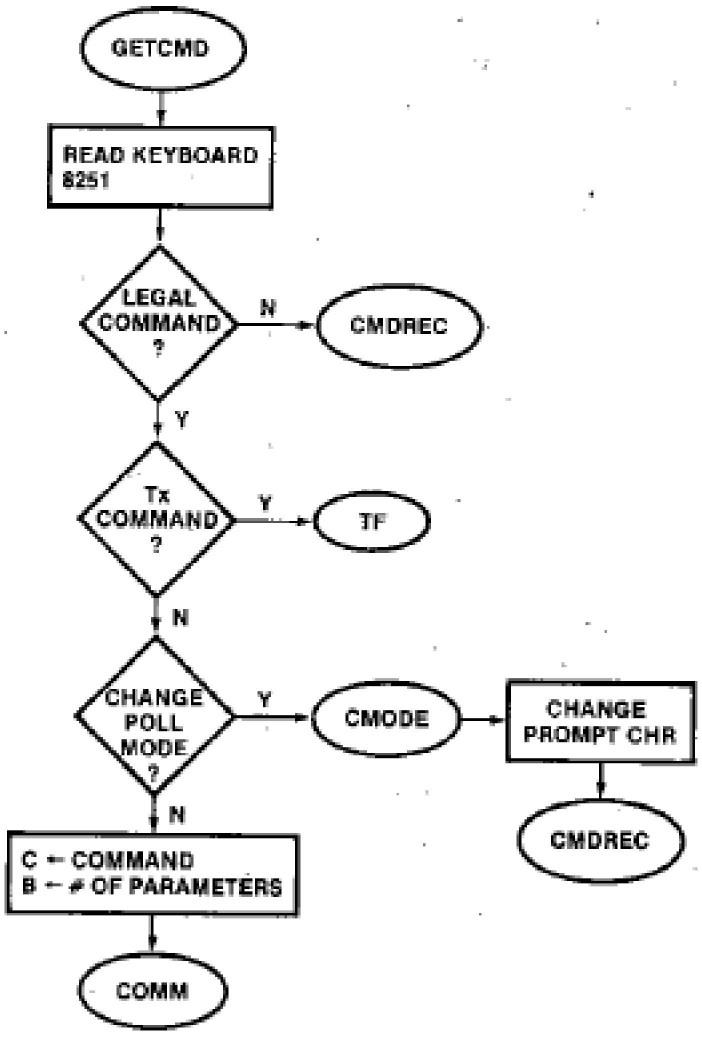

Figure 46. GETCMD Subroutine

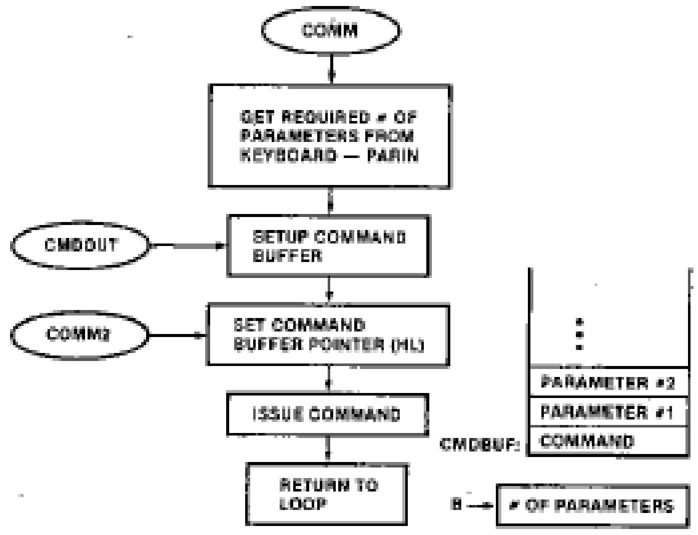

Figure 49. COMM Subroutine with Command Buffer Format

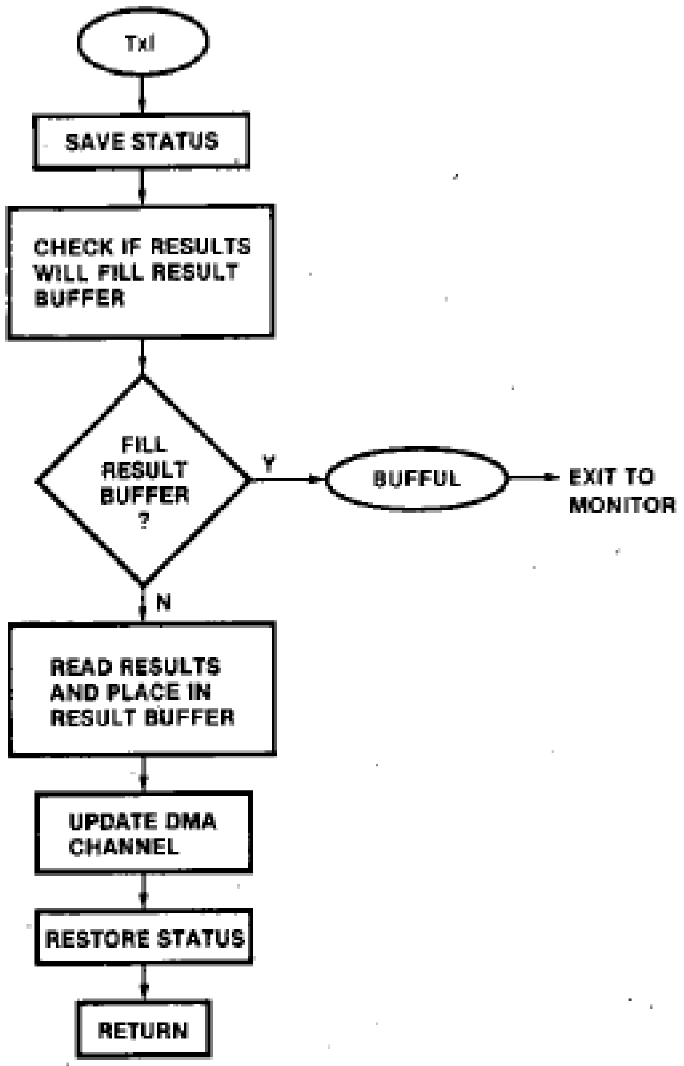

Figure 50. TxI (Transmitter Interrupt) Routine

If the result buffer pointers indicate an empty buffer, the 8251A is polled for a keyboard character. If the 8251 has a character, GETCMD is called. There the character is read and checked if legal. Illegal characters simply cause a reprompt. Legal characters indicate the start of a command input. Most commands are organized as two characters signifying the command action; i.e., GR — General Receive. The software recognizes the two character command code and takes the appropriate action. For non-Transmit type commands, the hex equivalent of the command is placed in the C register and the number of parameters associated with that command is placed in the B register. The program then branches to the COMM routine.

The COMM routine builds the command buffer by reading the required number of parameters from the keyboard and placing them at the buffer pointed at by CMDBUF. The routine at COMM2 then issues this command buffer to the 8273.

If a Transmit type command is specified, the command buffer is set up similarly to the the COMM routine; however, since the information field data is entered from the keyboard, an intermediate routine, TF, is called. TF loads the transmit data buffer pointed at by TxBUF. It counts the number of data bytes entered and loads this number into the command buffer as L_0, L_1. The command is then issued to the 8273 by jumping to CMDOUT.

One command does not directly result in a command being issued to the 8273. This command, Z, operates a software flip-flop which selects whether the software will respond automatically to received polling frames. If the Poll-Response mode is selected, the prompt character is changed to a '+'. If a frame is received which contains a prearranged poll control field, the memory location POLIN is made nonzero by the receiver interrupt handler. LOOPIT examines this location and if it is nonzero, causes a branch to the TxPOL routine. The TxPOL routine clears POLIN, sets a pointer to a special command buffer at CMDBUF1, and issues the command by way of the COMM2 entry in the COMM routine. The special command buffer contains the appropriate response frame for the poll frame received. These actions only occur when the Z command has changed the prompt to a '+'. If the prompt is normal '−', polling frames are displayed as normal frames and no response is transmitted. The Poll-Response mode was used during the IBM tests.

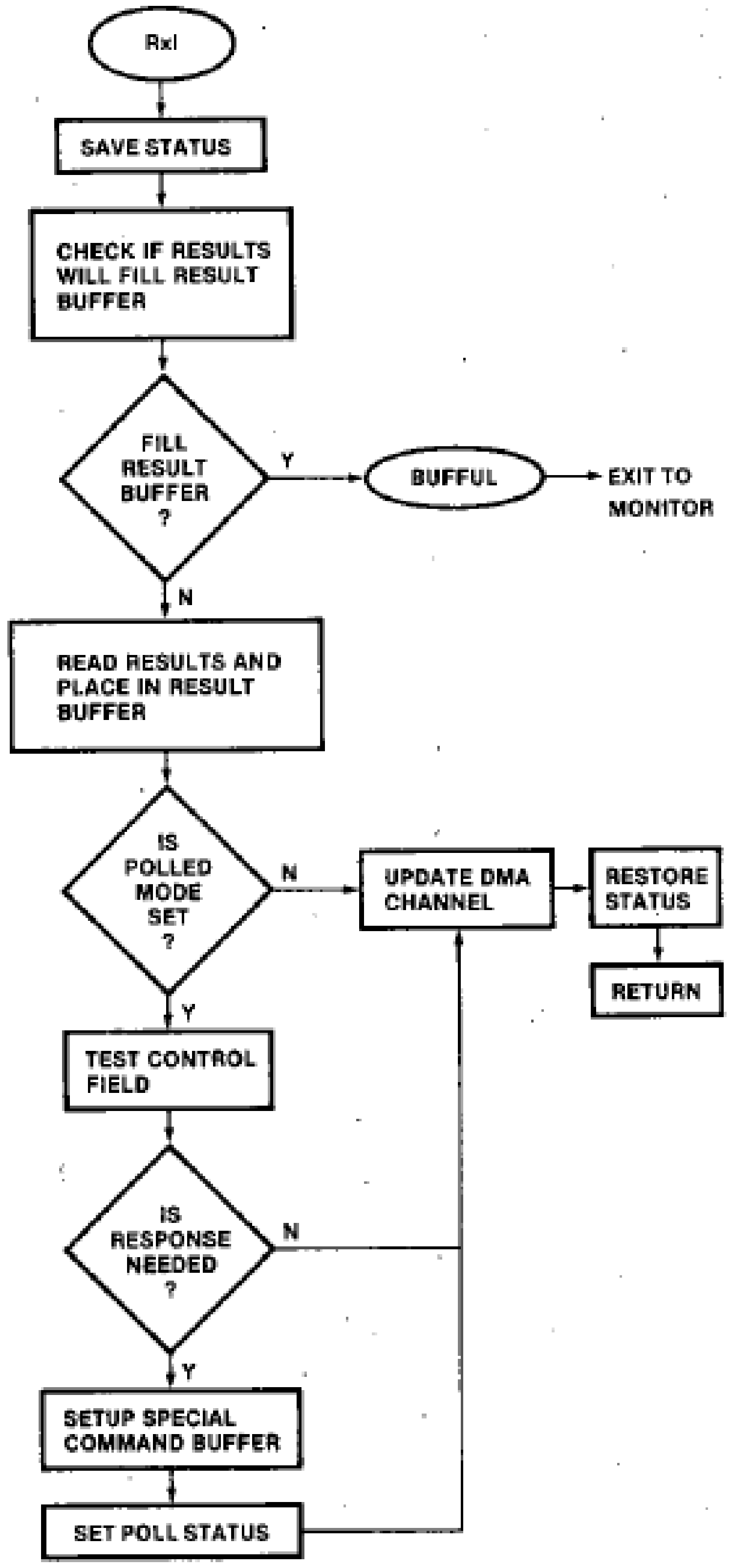

Figure 51. RxI (Receiver Interrupt) Routine

APPENDIX A

The final two software routines are the transmitter and receiver interrupt handlers. The transmit interrupt handler, TxI, simply saves the registers on the stack and checks if loading the result buffer will fill it. If the result buffer will overfill, the program is exited and control is passed to the SDK-85 monitor. If not, the results are read from the TxI/R register and placed in the result buffer at LDADR. The DMA pointers are then reset, the registers restored, and interrupts enabled. Execution then returns to the pre-interrupt location.

The receiver interrupt handler, RxI, is only slightly more complex. As in TxI, the registers are saved and the possibility of overfilling the result buffer is examined. If the result buffer is not full, the results are read from RxI/R and placed in the buffer. At this point the prompt character is examined to see if the Poll-Response mode is selected. If so, the control field is compared with two possible polling control fields. If there is a match, the special command buffer is loaded and the poll indicator, POLIN, is made nonzero. If no match occurred, no action is taken. Finally, the receiver DMA buffer pointers are reset, the processor status restored, and interrupts are enabled. The RET instruction returns execution to the pre-interrupt location.

This completes the discussion of the 8273/8085 system design.

CONCLUSION

This application note has covered the 8273 in some detail. The simple and low cost loop configuration was explored. And an 8273/8085 system was presented as a sample design illustrating the DMA/interrupt-driven interface. It is hoped that the major features of the 8273, namely the frame-level command structure and the Digital Phase Locked Loop, have been shown to be a valuable asset in an SDLC system design.

APPENDIX A

```
ASM80 :F1:RAYT73.SRC


ISIS-II 8080/8085 MACRO ASSEMBLER, X108        MODULE   PAGE    1


 LOC  OBJ         SEQ         SOURCE STATEMENT

                        $NOPAGING MOD85  NOCOND
 0000                   TRUE    EQU     00H             ;00 FOR RAYTHEON
                        ;                               ;FF FOR SELF-TEST
 0000                   TRUE1   EQU     00H             ;00 FOR NORMAL RESPONSE
                        ;                               ;FF FOR LOOP RESPONSE
 0000                   DEM     EQU     00H             ;00 FOR NO DEMO
                        ;                               ;FF FOR DEMO
                        ;
                        ;
                        ;GENERAL 8273 MONITOR WITH RAYTHEON POLL MODE ADDED
                        ;
                        ;
                        ;
                        ;COMMAND SUPPORTED ARE: RS - RESET SERIAL I/O MODE
                        ;                       SS - SET SERIAL I/O MODE
                        ;                       RO - RESET OPERATING MODE
                        ;                       SO - SET OPERATING MODE
                        ;                       RD - RECEIVER DISABLE
                        ;                       GR - GENERAL RECEIVE
                        ;                       SR - SELECTIVE RECEIVE
                        ;                       TF - TRANSMIT FRAME
                        ;                       AF - ABORT FRAME
                        ;                       SP - SET PORT B
                        ;                       RP - RESET PORT B
                        ;                       RB - RESET 0NE BIT DELAY (PAR = 7F)
                        ;                       SB - SET ONE BIT DELAY (PAR = 80)
                        ;                       SL - SELECTIVE LOOP RECEIVE
                        ;                       TL - TRANSMIT LOOP
                        ;                       Z  - CHANGE MODES FLIP/FLOP
                        ;
                        ;**************************************************************************
                        ;
                        ;NOTE:   'SET' COMMANDS IMPLEMENT LOGICAL 'OR' FUNCTIONS
                        ;        'RESET' COMMANDS IMPLEMENT LOGICAL 'AND' FUNCTIONS
                        ;
                        ;**************************************************************************
                        ;
                        ;BUFFERED MODE MUST BE SELECTED WHEN SELECTIVE RECEIVE IS USED.
                        ;
                        ;COMMAND FORMAT IS: 'COMMAND (2 LTRS)' 'PAR.#1' 'PAR.#2' ETC.
                        ;
                        ;THE TRANSMIT FRAME COMMAND FORMAT IS: 'TF' 'A' 'C' 'BUFFER CONTENTS'.
                        ;        NO LENGTH COUNT IS NEEDED.  BUFFER CONTENTS IS ENDED WITH A CR.
                        ;
                        ;**************************************************************************
                        ;
                        ;POLLED MODE:   WHEN POLLED MODE IS SELECTED (DENOTED BY A '+' PROMPT), IF
```

```
                ;               A SNRM-P OR RR(0)-P IS RECEIVED, A RESPONSE FRAME OF NSA-F
                ;               OR RR(0)-F IS TRANSMITTED.  OTHER COMMANDS OPERATE NORMALLY.
                ;
                ;*****************************************************************************
                ;
                ;8273 EQUATES
                ;
0090            STAT73  EQU     90H             ;STATUS REGISTER
0090            COMM73  EQU     90H             ;COMMAND REGISTER
0091            PARM73  EQU     91H             ;PARAMETER REGISTER
0091            RESL73  EQU     91H             ;RESULT REGISTER
0092            TXIR73  EQU     92H             ;TX INTERRUPT RESULT REGISTER
0093            RXIR73  EQU     93H             ;RX INTERRUPT RESULT REGISTER
0092            TEST73  EQU     92H             ;TEST MODE REGISTER
0020            CPBF    EQU     20H             ;PARAMETER BUFFER FULL BIT
0004            TXINT   EQU     04H             ;TX INTERRUPT BIT IN STATUS REGISTER
0008            RXINT   EQU     08H             ;RX INTERRUPT BIT IN STATUS REGISTER
0001            TXIRA   EQU     01H             ;TX INT RESULT AVAILABLE BIT
0002            RXIRA   EQU     02H             ;RX INT RESULT AVAILABLE BIT
                ;
                ;8253 EQUATES
                ;
009B            MODE53  EQU     9BH             ;8253 MODE WORD REGISTER
009C            CNT053  EQU     9CH             ;COUNTER 0 REGISTER
009D            CNT153  EQU     9DH             ;COUNTER 1 REGISTER
009E            CNT253  EQU     9EH             ;COUNTER 2 REGISTER
000C            COBR    EQU     000CH           ;CONSOLE BAUD RATE (2400)
0036            MDCNT0  EQU     36H             ;MODE FOR COUNTER 0
00B6            MDCNT2  EQU     0B6H            ;MODE FOR COUNTER 2
2017            LKBR1   EQU     2017H           ;8273 BAUD RATE LSB ADR
2018            LKBR2   EQU     2018H           ;8273 BAUD RATE MSB ADR
                ;
                ;BAUD RATE TABLE:        BAUD RATE       LKBR1   LKBR2
                ;                        *********       *****   *****
                ;                          9600           2E      00
                ;                          4800           5C      00
                ;                          2400           B9      00
                ;                          1200           72      01
                ;                           600           E5      02
                ;                           300           C9      05
                ;
                ;
                ;8257 EQUATES
                ;
00A8            MODE57  EQU     0A8H            ;8257 MODE PORT
00A0            CH0ADR  EQU     0A0H            ;CH0 (RX) ADR REGISTER
00A1            CH0TC   EQU     0A1H            ;CH0 TERMINAL COUNT REGISTER
00A2            CH1ADR  EQU     0A2H            ;CH1 (TX) ADR REGISTER
00A3            CH1TC   EQU     0A3H            ;CH1 TERMINAL COUNT REGISTER
00A8            STAT57  EQU     0A8H            ;STATUS REGISTER
8200            RXBUF   EQU     8200H           ;RX BUFFER START ADDRESS
8000            TXBUF   EQU     8000H           ;TX BUFFER START ADDRESS
0062            DRDMA   EQU     62H             ;DISABLE RX DMA CHANNEL, TX STILL ON
41FF            RXTC    EQU     41FFH           ;TERMINAL COUNT AND MODE FOR RX CHANNEL
0063            ENDMA   EQU     63H             ;ENABLE BOTH TX AND RX CHANNELS-EXT. WR, TX STOP
0061            DTDMA   EQU     61H             ;DISABLE TX DMA CHANNEL, RX STILL ON
81FF            TXTC    EQU     81FFH           ;TERMINAL COUNT AND MODE FOR TX CHANNEL
                ;
```

```
              ;8251A EQUATES
              ;
0089          CNTL51  EQU     89H         ;CONTROL WORD REGISTER
0089          STAT51  EQU     89H         ;STATUS REGISTER
0088          TXD51   EQU     88H         ;TX DATA REGISTER
0088          RXD51   EQU     88H         ;RX DATA REGISTER
00CE          MDE51   EQU     0CEH        ;MODE 16X,2 STOP,NO PARITY
0027          CMD51   EQU     27H         ;COMMAND, ENABLE TX&RX
0002          RDY     EQU     02H         ;RXRDY BIT
              ;
              ;MONITOR SUBROUTINE EQUATES
              ;
061F          GETCH   EQU     061FH       ;GET CHR FROM KEYBOARD, ASCII IN CH
05F8          ECHO    EQU     05F8H       ;ECHO CHR TO DISPLAY
075E          VALDG   EQU     075EH       ;CHECK IF VALID DIGIT, CARRY SET IF VALID
05BB          CNVBN   EQU     05BBH       ;CONVERTS ASCII TO HEX
05EB          CRLF    EQU     05EBH       ;DISPLAY CR, HENCE LF TOO
06C7          NMOUT   EQU     06C7H       ;CONVERT BYTE TO 2 ASCII CHR AND DISPLAY
              ;
              ;MISC EQUATES
              ;
20C0          STKSRT  EQU     20C0H       ;STACK START
0003          CNTLC   EQU     03H         ;CNTL-C EQUIVALENT
0008          MONTOR  EQU     0008H       ;MONITOR
2000          CMDBUF  EQU     2000H       ;START OF COMMAND BUFFER
2020          CMDBF1  EQU     2020H       ;POLL MODE SPECIAL TX COMMAND BUFFER
000D          CR      EQU     0DH         ;ASCII CR
000A          LF      EQU     0AH         ;ASCII LF
20D4          RST75   EQU     20D4H       ;RST7.5 JUMP ADDRESS
20CE          RST65   EQU     20CEH       ;RST6.5 JUMP ADDRESS
2010          LDADR   EQU     2010H       ;RESULT BUFFER LOAD POINTER STORAGE
2013          CNADR   EQU     2013H       ;RESULT BUFFER CONSOLE POINTER STORAGE
2800          RESBUF  EQU     2800H       ;RESULT BUFFER START - 255 BYTES
0093          SNRMP   EQU     93H         ;SNRM-P CONTROL CODE
0011          RR0P    EQU     11H         ;RR(0)-P CONTROL CODE
0073          NSAF    EQU     73H         ;NSA-F CONTROL CODE
0011          RR0F    EQU     11H         ;RR(0)-F CONTROL CODE
2015          PRMPT   EQU     2015H       ;PRMPT STORAGE
2016          POLIN   EQU     2016H       ;POLL MODE SELECTION INDICATOR
2027          DEMODE  EQU     2027H       ;DEMO MODE INDICATOR
              ;
              ;**************************************************************************
              ;
              ;RAM STORAGE DEFINITIONS:
              ;       LOC             DEF
              ;       ---             ---
              ;       2000-200F       COMMAND BUFFER
              ;       2010-2011       RESULT BUFFER LOAD POINTER
              ;       2013-2014       RESULT BUFFER CONSOLE POINTER
              ;       2015            PROMPT CHARACTER STORAGE
              ;       2016            POLL MODE INDICATOR
              ;       2017            BAUD RATE LSB FOR SELF-TEST
              ;       2018            BAUD RATE MSB FOR SELF-TEST
              ;       2019            SPARE
              ;       2020-2026       RESPONSE COMMAND BUFFER FOR POLL MODE
              ;       2800-28FF       RESULT BUFFER
              ;
              ;**************************************************************************
```

```
                ;
                ;PROGRAM START
                ;
                ;INITIALIZE 8253, 8257, 8251A, AND RESET 8273.
                ;ALSO SET NORMAL MODE, AND PRINT SIGNON MESSAGE
                ;
0800                    ORG     800H

0800 31C020     START:  LXI     SP,STKSRT       ;INITIALIZE SP
0803 3E36               MVI     A,MDCNT0        ;8253 MODE SET
0805 D39B               OUT     MODE53          ;8253 MODE PORT
0807 3A1720             LDA     LKBR1           ;GET 8273 BAUD RATE LSB
080A D39C               OUT     CNT053          ;USING COUNTER 0 AS BAUD RATE GEN
080C 3A1820             LDA     LKBR2           ;GET 8273 BUAD RATE MSB
080F D39C               OUT     CNT053          ;COUNTER 0
0811 CD1A0B             CALL    RXDMA           ;INITIALIZE 8257 RX DMA CHANNEL
0814 CD350B             CALL    TXDMA           ;INITIALIZE 8257 TX DMA CHANNEL
0817 3E01               MVI     A,01H           ;OUTPUT 1 FOLLOWED BY A 0
0819 D392               OUT     TEST73          ;TO TEST MODE REGISTER
081B 3E00               MVI     A,00H           ;TO RESET THE 8273
081D D392               OUT     TEST73
081F 3E2D               MVI     A,'-'           ;NORMAL MODE PROMPT CHR
0821 321520             STA     PRMPT           ;PUT IN STORAGE
0824 3E00               MVI     A,00H           ;TX POLL RESPONSE INDICATOR
0826 321620             STA     POLIN           ;0 MEANS NO SPECIAL TX
0829 322720             STA     DEMODE          ;CLEAR DEMO MODE
082C 21A30C             LXI     H,SIGNON                ;SIGNON MESSAGE ADR
082F CD920C             CALL    TYMSG           ;DISPLAY SIGNON
                ;
                ;MONITOR USES JUMPS IN RAM TO DIRECT INTERRUPTS
                ;
0832 21D420             LXI     H,RST75         ;RST7.5 JUMP LOCATION USED BY MONITOR
0835 01000C             LXI     B,RXI           ;ADDRESS OF RX INT ROUTINE
0838 36C3               MVI     M,0C3H          ;LOAD 'JMP' OPCODE
083A 23                 INX     H               ;INC POINTER
083B 71                 MOV     M,C             ;LOAD RXI LSB
083C 23                 INX     H               ;INC POINTER
083D 70                 MOV     M,B             ;LOAD RXI MSB
083E 21CE20             LXI     H,RST65         ;RST6.5 JUMP LOCATION USED BY MONITOR
0841 01CE0C             LXI     B,TXI           ;ADDRESS OF TX INT ROUTINE
0844 36C3               MVI     M,0C3H          ;LOAD 'JMP' OPCODE
0846 23                 INX     H               ;INC POINTER
0847 71                 MOV     M,C             ;LOAD TXI LSB
0848 23                 INX     H               ;INC POINTER
0849 70                 MOV     M,B             ;LOAD TXI MSB
084A 3E18               MVI     A,18H           ;GET SET TO RESET INTERRUPTS
084C 30                 SIM                     ;RESET INTERRUPTS
084D FB                 EI                      ;ENABLE INTERRUPTS
                ;
                ;INITIALIZE BUFFER POINTER
                ;
                ;
084E 210028             LXI     H,RESBUF        ;SET RESULT BUFFER POINTERS
0851 221320             SHLD    CNADR           ;RESULT CONSOLE POINTER
0854 221020             SHLD    LDADR           ;RESULT LOAD POINTER
                ;
                ;MAIN PROGRAM LOOP - CHECKS FOR CHANGE IN RESULT POINTERS, USART STATUS,
                ;       OR POLL STATUS
```

```
                   ;
0857 CDEB05        CMDREC: CALL   CRLF        ;DISPLAY CR
085A 3A1520                LDA    PRMPT       ;GET CURRENT PROMPT CHR
085D 4F                    MOV    C,A         ;MOVE TO C
085E CDF805                CALL   ECHO        ;DISPLAY IT
0861 2A1320        LOOPIT: LHLD   CNADR       ;GET CONSOLE POINTER
0864 7D                    MOV    A,L         ;SAVE POINTER LSB
0865 2A1020                LHLD   LDADR       ;GET LOAD POINTER
0868 BD                    CMP    L           ;SAME LSB?
0869 C2390A                JNZ    DISPY       ;NO, RESULTS NEED DISPLAYING
086C DB89                  IN     STAT51      ;YES, CHECK KEYBOARD
086E E602                  ANI    RDY         ;CHR RECEIVED?
0870 C27D08                JNZ    GETCMD      ;MUST BE CHR SO GO GET IT
0873 3A1620                LDA    POLIN       ;GET POLL MODE STATUS
0876 A7                    ANA    A           ;IS IT 0?
0877 C24C09                JNZ    TXPOL       ;NO, THEN POLL OCCURRED
087A C36108                JMP    LOOPIT      ;YES, TRY AGAIN
                   ;
                   ;
                   ;COMMAND RECOGNIZER ROUTINE
                   ;
                   ;
087D CD1F06        GETCMD: CALL   GETCH       ;GET CHR
0880 CDF805                CALL   ECHO        ;ECHO IT
0883 79                    MOV    A,C         ;SETUP FOR COMPARE
0884 FE52                  CPI    'R'         ;R?
0886 CAAF08                JZ     RDWN        ;GET MORE
0889 FE53                  CPI    'S'         ;S?
088B CAD708                JZ     SDWN        ;GET MORE
088E FE47                  CPI    'G'         ;G?
0890 CAFF08                JZ     GDWN        ;GET MORE
0893 FE54                  CPI    'T'         ;T?
0895 CA0E09                JZ     TDWN        ;GET MORE
0898 FE41                  CPI    'A'         ;A?
089A CA2209                JZ     ADWN        ;GET MORE
089D FE5A                  CPI    'Z'         ;Z?
089F CA3109                JZ     CMODE       ;YES, GO CHANGE MODE
08A2 FE03                  CPI    CNTLC       ;CNTL-C?
08A4 CA0800                JZ     MONTOR      ;EXIT TO MONITOR
08A7 0E3F          ILLEG:  MVI    C,'?'       ;PRINT ?
08A9 CDF805                CALL   ECHO        ;DISPLAY IT
08AC C35708                JMP    CMDREC      ;LOOP FOR COMMAND

08AF CD1F06        RDWN:   CALL   GETCH       ;GET NEXT CHR
08B2 CDF805                CALL   ECHO        ;ECHO IT
08B5 79                    MOV    A,C         ;SETUP FOR COMPARE
08B6 FE4F                  CPI    'O'         ;O?
08B8 CA5D09                JZ     ROCMD       ;RO COMMAND
08BB FE53                  CPI    'S'         ;S?
08BD CA6709                JZ     RSCMD       ;RS COMMAND
08C0 FE44                  CPI    'D'         ;D?
08C2 CA7109                JZ     RDCMD       ;RD COMMAND
08C5 FE50                  CPI    'P'         ;P?
08C7 CAD809                JZ     RPCMD       ;RP COMMAND
08CA FE52                  CPI    'R'         ;R?
08CC CA0008                JZ     START       ;START OVER
08CF FE42                  CPI    'B'         ;B?
08D1 CA7B09                JZ     RBCMD       ;RB COMMAND
```

```
08D4 C3A708          JMP     ILLEG          ;ILLEGAL, TRY AGAIN

08D7 CD1F06    SDWN: CALL    GETCH          ;GET NEXT CHR
08DA CDF805          CALL    ECHO           ;ECHO IT
08DD 78              MOV     A,B            ;SETUP FOR COMPARE
08DE FE4F            CPI     'O'            ;O?
08E0 CAA609          JZ      SOCMD          ;SO COMMAND
08E3 FE53            CPI     'S'            ;S?
08E5 CAB009          JZ      SSCMD          ;SS COMMAND
08E8 FE52            CPI     'R'            ;R?
08EA CABA09          JZ      SRCMD          ;SR COMMAND
08ED FE50            CPI     'P'            ;P?
08EF CAE209          JZ      SPCMD          ;SP COMMAND
08F2 FE42            CPI     'B'            ;B?
08F4 CA8509          JZ      SBCMD          ;SB COMMAND
08F7 FE4C            CPI     'L'            ;L?
08F9 CA6F09          JZ      SLCMD          ;SL COMMAND
08FC C3A708          JMP     ILLEG          ;ILLEGAL, TRY AGAIN

08FF CD1F06    GDWN: CALL    GETCH          ;GET NEXT CHR
0902 CDF805          CALL    ECHO           ;ECHO IT
0905 78              MOV     A,B            ;SETUP FOR COMPARE
0906 FE52            CPI     'R'            ;R?
0908 CAC409          JZ      GRCMD          ;GR COMMAND
090B C3A708          JMP     ILLEG          ;ILLEGAL, TRY AGAIN

090E CD1F06    TDWN: CALL    GETCH          ;GET NEXT CHR
0911 CDF805          CALL    ECHO           ;ECHO IT
0914 78              MOV     A,B            ;SETUP FOR COMPARE
0915 FE46            CPI     'F'            ;F?
0917 CAEC09          JZ      TFCMD          ;TF COMMAND
091A FE4C            CPI     'L'            ;L?
091C CA9909          JZ      TLCMD          ;TL COMMAND
091F C3A708          JMP     ILLEG          ;ILLEGAL, TRY AGAIN

0922 CD1F06    ADWN: CALL    GETCH          ;GET NEXT CHR
0925 CDF805          CALL    ECHO           ;ECHO IT
0928 78              MOV     A,B            ;SETUP FOR COMPARE
0929 FE46            CPI     'F'            ;F?
092B CACE09          JZ      AFCMD          ;AF COMMAND
092E C3A708          JMP     ILLEG          ;ILLEGAL, TRY AGAIN
               ;
               ;RESET POLL MODE RESPONSE - CHANGE PROMPT CHR AS INDICATOR
               ;
0931 F3        CMODE: DI                    ;DISABLE INTERRUPTS
0932 3A1520          LDA     PRMPT          ;GET CURRENT PROMPT
0935 FE2D            CPI     '-'            ;NORMAL MODE?
0937 C24309          JNZ     SW             ;NO, CHANGE IT
093A 3E2B            MVI     A,'+'          ;NEW PROMPT
093C 321520          STA     PRMPT          ;STORE NEW PROMPT
093F FB              EI                     ;ENABLE INTERRUPTS
0940 C35708          JMP     CMDREC         ;RETURN TO LOOP
0943 3E2D      SW:   MVI     A,'-'          ;NEW PROMPT CHR
0945 321520          STA     PRMPT          ;STORE IT
0948 FB              EI                     ;ENABLE INTERRUPTS
0949 C35708          JMP     CMDREC         ;RETURN TO LOOP
               ;
               ;
```

```
                    373 ;TRANSMIT ANSWER TO POLL SETUP
                    374 ;
094C 3E00           382 TXPOL:  MVI     A,00H           ;CLEAR POLL INDICATOR
094E 321620         384         STA     POLIN           ;INDICATOR ADR
0951 216108         385         LXI     H,LOOPIT        ;SETUP STACK FOR COMMAND OUTPUT
0954 E5             386         PUSH    H               ;PUT RETURN TO CMDREC ON STACK
0955 0604           387         MVI     B,04H           ;GET # OF PARAMETERS READY
0957 212020         388         LXI     H,CMDBF1        ;POINT TO SPECIAL BUFFER
095A C3FF0A         389         JMP     COMM2           ;JUMP TO COMMAND OUTPUTER
                    390 ;
                    391 ;
                    392 ;
                    393 ;COMMAND IMPLEMENTING ROUTINES
                    394 ;
                    395 ;
                    396 ;RO - RESET OPERATING MODE
                    397 ;
095D 0601           398 ROCMD:  MVI     B,01H           ;# OF PARAMETERS
095F 0E51           399         MVI     C,51H           ;COMMAND
0961 CDE50A         400         CALL    COMM            ;GET PARAMETERS AND ISSUE COMMAND
0964 C35708         401         JMP     CMDREC          ;GET NEXT COMMAND
                    402 ;
                    403 ;RS - RESET SERIAL I/O MODE COMMAND
                    404 ;
0967 0601           405 RSCMD:  MVI     B,01H           ;# OF PARAMETERS
0969 0E60           406         MVI     C,60H           ;COMMAND
096B CDE50A         407         CALL    COMM            ;GET PARAMETERS AND ISSUE COMMAND
096E C35708         408         JMP     CMDREC          ;GET NEXT COMMAND
                    409 ;
                    410 ;RD - RECEIVER DISABLE COMMAND
                    411 ;
0971 0600           412 RDCMD:  MVI     B,00H           ;# OF PARAMETERS
0973 0EC5           413         MVI     C,0C5H          ;COMMAND
0975 CDE50A         414         CALL    COMM            ;ISSUE COMMAND
0978 C35708         415         JMP     CMDREC          ;GET NEXT COMMAND
                    416 ;
                    417 ;RB - RESET ONE BIT DELAY COMMAND
                    418 ;
097B 0601           419 RBCMD:  MVI     B,01H           ;# OF PARAMETERS
097D 0E64           420         MVI     C,64H           ;COMMAND
097F CDE50A         421         CALL    COMM            ;GET PARAMETER AND ISSUE COMMAND
0982 C35708         422         JMP     CMDREC          ;GET NEXT COMMAND
                    423 ;
                    424 ;SB - SET ONE BIT DELAY COMMAND
                    425 ;
0985 0601           426 SBCMD:  MVI     B,01H           ;# OF PARAMETERS
0987 0EA4           427         MVI     C,0A4H          ;COMMAND
0989 CDE50A         428         CALL    COMM            ;GET PARAMETER AND ISSUE COMMAND
098C C35708         429         JMP     CMDREC          ;GET NEXT COMMAND
                    430 ;
                    431 ;SL - SELECTIVE LOOP RECEIVE COMMAND
                    432 ;
098F 0604           433 SLCMD:  MVI     B,04H           ;# OF PARAMETERES
0991 0EC2           434         MVI     C,0C2H          ;COMMAND
0993 CDE50A         435         CALL    COMM            ;GET PARAMETERS AND ISSUE COMMAND
0996 C35708         436         JMP     CMDREC          ;GET NEXT COMMAND
                    437 ;
                    438 ;TL - TRANSMIT LOOP COMMAND
```

```
                   ;
0999 210020          TLCMD:  LXI    H,CMDBUF     ;SET COMMAND BUFFER POINTER
099C 0602                    MVI    B,02H        ;LOAD PARAMETER COUNTER
099E 36CA                    MVI    M,0CAH       ;LOAD COMMAND INTO BUFFER
09A0 210220                  LXI    H,CMDBUF+2   ;POINT AT ADR AND CNTL POSITIONS
09A3 C3F609                  JMP    TFCMD1       ;FINISH OFF COMMAND IN TF ROUTINE
                   ;
                   ;SO - SET OPERATING MODE COMMAND
                   ;
09A6 0601            SOCMD:  MVI    B,01H        ;# OF PARAMETERS
09A8 0E91                    MVI    C,91H        ;COMMAND
09AA CDE50A                  CALL   COMM         ;GET PARAMETER AND ISSUE COMMAND
09AD C35708                  JMP    CMDREC       ;GET NEXT COMMAND
                   ;
                   ;SS - SET SERIAL I/O COMMAND
                   ;
09B0 0601            SSCMD:  MVI    B,01H        ;# OF PARAMETERS
09B2 0EA0                    MVI    C,0A0H       ;COMMAND
09B4 CDE50A                  CALL   COMM         ;GET PARAMETER AND ISSUE COMMAND
09B7 C35708                  JMP    CMDREC       ;GET NEXT COMMAND
                   ;
                   ;SR - SELECTIVE RECEIVE COMMAND
                   ;
09BA 0604            SRCMD:  MVI    B,04H        ;# OF PARAMETERS
09BC 0EC1                    MVI    C,0C1H       ;COMMAND
09BE CDE50A                  CALL   COMM         ;GET PARAMETERS AND ISSUE COMMAND
09C1 C35708                  JMP    CMDREC       ;GET NEXT COMMAND
                   ;
                   ;GR - GENERAL RECEIVE COMMAND
                   ;
09C4 0602            GRCMD:  MVI    B,02H        ;NO PARAMETERS
09C6 0EC0                    MVI    C,0C0H       ;COMMAND
09C8 CDE50A                  CALL   COMM         ;ISSUE COMMAND
09CB C35708                  JMP    CMDREC       ;GET NEXT COMMAND
                   ;
                   ;AF - ABORT FRAME COMMAND
                   ;
09CE 0600            AFCMD:  MVI    B,00H        ;NO PARAMETERS
09D0 0ECC                    MVI    C,0CCH       ;COMMAND
09D2 CDE50A                  CALL   COMM         ;ISSUE COMMAND
09D5 C35708                  JMP    CMDREC       ;GET NEXT COMMAND
                   ;
                   ;RP - RESET PORT COMMAND
                   ;
09D8 0601            RPCMD:  MVI    B,01H        ;# OF PARAMETERS
09DA 0E63                    MVI    C,63H        ;COMMAND
09DC CDE50A                  CALL   COMM         ;GET PARAMETER AND ISSUE COMMAND
09DF C35708                  JMP    CMDREC       ;GET NEXT COMMAND
                   ;
                   ;SP - SET PORT COMMAND
                   ;
09E2 0601            SPCMD:  MVI    B,01H        ;# OF PARAMETERS
09E4 0EA3                    MVI    C,0A3H       ;COMMAND
09E6 CDE50A                  CALL   COMM         ;GET PARAMETER AND ISSUE COMMAND
09E9 C35708                  JMP    CMDREC       ;GET NEX COMMAND
                   ;
                   ;TF - TRANSMIT FRAME COMMAND
                   ;
```

```
09EC 210020        TFCMD:  LXI   H,CMDBUF       ;SET COMMAND BUFFER POINTER
09EF 0602                  MVI   B,02H          ;LOAD PARAMETER COUNTER
09F1 36C8                  MVI   M,0C8H         ;LOAD COMMAND INTO BUFFER
09F3 210220                LXI   H,CMDBUF+2     ;POINT AT ADR AND CNTL POSITIONS
09F6 78            TFCMD1: MOV   A,B            ;TEST PARAMETER COUNT
09F7 A7                    ANA   A              ;IS IT 0?
09F8 CA070A                JZ    TBUFL          ;YES, LOAD TX DATA BUFFER
09FB CDAD0A                CALL  PARIN          ;GET PARAMETER
09FE DAA708                JC    ILLEG          ;ILLEGAL CHR RETURNED
0A01 23                    INX   H              ;INC COMMAND BUFFER POINTER
0A02 05                    DCR   B              ;DEC PARAMETER COUNTER
0A03 77                    MOV   M,A            ;LOAD PARAMETER INTO COMMAND BUFFER
0A04 C3F609                JMP   TFCMD1         ;GET NEXT PARAMETER

0A07 210080        TBUFL:  LXI   H,TXBUF        ;LOAD TX DATA BUFFER POINTER
0A0A 010000                LXI   B,0000H        ;CLEAR BC - BYTE COUNTER
0A0D C5            TBUFL1: PUSH  B              ;SAVE BYTE COUNTER
0A0E CDAD0A                CALL  PARIN          ;GET DATA, ALIAS PARAMETER
0A11 DA1B0A                JC    ENDCHK         ;MAYBE END IF ILLEGAL
0A14 77                    MOV   M,A            ;LOAD DATA BYTE INTO BUFFER
0A15 23                    INX   H              ;INC BUFFER POINTER
0A16 C1                    POP   B              ;RESTORE BYTE COUNTER
0A17 03                    INX   B              ;INC BYTE COUNTER
0A18 C30D0A                JMP   TBUFL1         ;GET NEXT DATA
0A1B FE0D          ENDCHK: CPI   CR             ;RETURNED ILLEGAL CHR CR?
0A1D CA240A                JZ    TBUFFL         ;YES, THEN TX BUFFER FULL
0A20 C1                    POP   B              ;RESTORE B TO SAVE STACK
0A21 C3A708                JMP   ILLEG          ;ILLEGAL CHR
0A24 C1            TBUFFL: POP   B              ;RESTORE BYTE COUNTER
0A25 210120                LXI   H,CMDBUF+1     ;POINT INTO COMMAND BUFFER
0A28 71                    MOV   M,C            ;STORE BYTE COUNT LSB
0A29 23                    INX   H              ;INC POINTER
0A2A 70                    MOV   M,B            ;STORE BYTE COUNT MSB
0A2B 0604                  MVI   B,04H          ;LOAD PARAMETER COUNT INTO B
0A2D 21360A                LXI   H,TFRET        ;GET RETURN ADR FOR THIS ROUTINE
0A30 C5                    PUSH  B              ;PUSH ONCE
0A31 E3                    XTHL                 ;PUT RETURN ON STACK
0A32 C5                    PUSH  B              ;PUSH IT SO CMDOUT CAN USE IT
0A33 C3FB0A                JMP   CMDOUT         ;ISSUE COMMAND
0A36 C35708        TFRET:  JMP   CMDREC         ;GET NEXT COMMAND
                   ;
                   ;
                   ;ROUTINE TO DISPLAY RESULT IN RESULT BUFFER WHEN LOAD AND CONSOLE
                   ;POINTERS ARE DIFFERENT.
                   ;
                   ;
0A39 1605          DISPY:  MVI   D,05H          ;D IS RESULT COUNTER
0A3B 2A1320                LHLD  CNADR          ;GET CONSOLE POINTER
0A3E E5                    PUSH  H              ;SAVE IT
0A3F 7E                    MOV   A,M            ;GET RESULT IC
0A40 E61F                  ANI   1FH            ;LIMIT TO RESULT CODE
0A42 FE0C                  CPI   0CH            ;TEST IF RX OR TX SOURCE
0A44 DA620A                JC    RXSORC         ;CARRY, THEN RX SOURCE
0A47 21C30C        TXSORC: LXI   H,TXIMSG       ;TX INT MESSAGE
0A4A CD920C                CALL  TYMSG          ;DISPLAY IT
0A4D E1            DISPY2: POP   H              ;RESTORE CONSOLE POINTER
0A4E 7E            DISPY1: MOV   A,M            ;GET RESULT
0A4F CDC706                CALL  NMOUT          ;CONVERT AND DISPLAY
```

```
0A52 0E20                  MVI    C,' '       ;SP CHR
0A54 CDF805                CALL   ECHO        ;DISPLAY IT
0A57 2C                    INR    L           ;INC BUFFER POINTER
0A58 15                    DCR    D           ;DEC RESULT COUNTER
0A59 C24E0A                JNZ    DISPY1      ;NOT DONE
0A5C 221320                SHLD   CNADR       ;UPDATE CONSOLE POINTER
0A5F C35708                JMP    CMDREC      ;RETURN TO LOOP
                     ;
                     ;
                     ;RECEIVER SOURCE - DISPLAY RESULTS AND RECEVIE BUFFER CONTENTS
                     ;
                     ;
0A62 21B80C          RXSORC: LXI  H,RXIMSG    ;RX INT MESSAGE ADR
0A65 CD920C                CALL   TYMSG       ;DISPLAY MESSAGE
0A68 E1                    POP    H           ;RESTORE CONSOLE POINTER
0A69 7E              RXS1:  MOV   A,M         ;RETRIEVE RESULT FROM BUFFER
0A6A CDC706                CALL   NMOUT       ;CONVERT AND DISPLAY IT
0A6D 0E20                  MVI    C,' '       ;ASCII SP
0A6F CDF805                CALL   ECHO        ;DISPLAY IT
0A72 2C                    INR    L           ;INC CONSOLE POINTER
0A73 15                    DCR    D           ;DEC RESULT COUNTER
0A74 7A                    MOV    A,D         ;GET SET TO TEST COUNTER
0A75 FE04                  CPI    04H         ;IS THE RESULT R0?
0A77 CAA20A                JZ     R0PT        ;YES, GO SAVE IT
0A7A FE03                  CPI    03H         ;IS THE RESULT R1?
0A7C CAA70A                JZ     R1PT        ;YES, GO SAVE IT
0A7F A7              RXS2:  ANA   A           ;TEST RESULT COUNTER
0A80 C2690A                JNZ    RXS1        ;NOT DONE YET, GET NEXT RESULT
0A83 221320                SHLD   CNADR       ;DONE, SO UPDATE CONSOLE POINTER
0A86 CDEB05                CALL   CRLF        ;DISPLAY CR
0A89 210082                LXI    H,RXBUF     ;POINT AT RX BUFFER
0A8C C1                    POP    B           ;RETRIEVE RECEIVED COUNT
0A8D 78              RXS3:  MOV   A,B         ;IS COUNT 0?
0A8E B1                    ORA    C           ;
0A8F CA5708                JZ     CMDREC      ;YES, GO BACK TO LOOP
0A92 7E                    MOV    A,M         ;NO, GET CHR
0A93 C5                    PUSH   B           ;SAVE BC
0A94 CDC706                CALL   NMOUT       ;CONVERT AND DISPLAY CHR
0A97 0E20                  MVI    C,' '       ;ASCII SP
0A99 CDF805                CALL   ECHO        ;DISPLAY IT TO SEPARATE DATA
0A9C C1                    POP    B           ;RESTORE BC
0A9D 0B                    DCX    B           ;DEC COUNT
0A9E 23                    INX    H           ;INC POINTER
0A9F C38D0A                JMP    RXS3        ;GET NEXT CHR

0AA2 4E              R0PT:  MOV   C,M         ;GET R0 FOR RESULT BUFFER
0AA3 C5                    PUSH   B           ;SAVE IT
0AA4 C37F0A                JMP    RXS2        ;RETURN

0AA7 C1              R1PT:  POP   B           ;GET R0
0AA8 46                    MOV    B,M         ;GET R1 FOR RESULT BUFFER
0AA9 C5                    PUSH   B           ;SAVE IT
0AAA C37F0A                JMP    RXS2
                     ;
                     ;
                     ;
                     ;PARAMETER INPUT - PARAMETER RETURNED IN E REGISTER
                     ;
```

```
                ;
0AAD C5         PARIN:  PUSH    B               ;SAVE BC
0AAE 1601               MVI     D,01H           ;SET CHR COUNTER
0AB0 CD1F06             CALL    GETCH           ;GET CHR
0AB3 CDF805             CALL    ECHO            ;ECHO IT
0AB6 79                 MOV     A,C             ;PUT CHR IN A
0AB7 FE20               CPI     ' '             ;SP?
0AB9 C2E00A             JNZ     PARIN1          ;NO, ILLEGAL, TRY AGAIN
0ABC CD1F06     PARIN3: CALL    GETCH           ;GET CHR OF PARAMETER
0ABF CDF805             CALL    ECHO            ;ECHO IT
0AC2 CD5E07             CALL    VALDG           ;IS IT A VALID CHR?
0AC5 D2E00A             JNC     PARIN1          ;NO, TRY AGAIN
0AC8 CDBB05             CALL    CNVBN           ;CONVERT IT TO HEX
0ACB 4F                 MOV     C,A             ;SAVE IT IN C
0ACC 7A                 MOV     A,D             ;GET CHR COUNTER
0ACD A7                 ANA     A               ;IS IT 0?
0ACE CADC0A             JZ      PARIN2          ;YES,DONE WITH THIS PARAMETER
0AD1 15                 DCR     D               ;DEC CHR COUNTER
0AD2 AF                 XRA     A               ;CLEAR CARRY
0AD3 79                 MOV     A,C             ;RECOVER 1ST CHR
0AD4 17                 RAL                     ;ROTATE LEFT 4 PLACES
0AD5 17                 RAL
0AD6 17                 RAL
0AD7 17                 RAL
0AD8 5F                 MOV     E,A             ;SAVE IT IN E
0AD9 C3BC0A             JMP     PARIN3          ;GET NEXT CHR
0ADC 79         PARIN2: MOV     A,C             ;2ND CHR IN A
0ADD B3                 ORA     E               ;COMBINE BOTH CHRS
0ADE C1                 POP     B               ;RESTORE BC
0ADF C9                 RET                     ;RETURN TO CALLING PROGRAM
0AE0 79         PARIN1: MOV     A,C             ;PUT ILLEGAL CHR IN A
0AE1 37                 STC                     ;SET CARRY AS ILLEGAL STATUS
0AE2 C1                 POP     B               ;RESTORE BC
0AE3 C9                 RET                     ;RETURN TO CALLING PROGRAM
                ;
                ;
                ;JUMP HERE IF BUFFER FULL
                ;
0AE4 CF         BUFFUL: DB      0CFH            ;EXIT TO MONITOR
                ;
                ;
                ;COMMAND DISPATCHER
                ;
                ;
0AE5 210020     COMM:   LXI     H,CMDBUF        ;SET POINTER
0AE8 C5                 PUSH    B               ;SAVE BC
0AE9 71                 MOV     M,C             ;LOAD COMMAND INTO BUFFER
0AEA 78         COMM1:  MOV     A,B             ;CHECK PARAMETER COUNTER
0AEB A7                 ANA     A               ;IS IT 0?
0AEC CAFB0A             JZ      CMDOUT          ;YES, GO ISSUE COMMAND
0AEF CDAD0A             CALL    PARIN           ;GET PARAMETER
0AF2 DAA708             JC      ILLEG           ;ILLEGAL CHR RETURNED
0AF5 23                 INX     H               ;INC BUFFER POINTER
0AF6 05                 DCR     B               ;DEC PARAMETER COUNTER
0AF7 77                 MOV     M,A             ;PARAMETER TO BUFFER
0AF8 C3EA0A             JMP     COMM1           ;GET NEXT PARAMETER
0AFB 210020     CMDOUT: LXI     H,CMDBUF        ;REPOINT POINTER
0AFE C1                 POP     B               ;RESTORE PARAMETER COUNT
```

```
0AFF DB90          COMM2:  IN      STAT73          ;READ 8273 STATUS
0B01 07                    RLC                     ;ROTATE CBSY INTO CARRY
0B02 DAFF0A                JC      COMM2           ;WAIT FOR OK
0B05 7E                    MOV     A,M             ;OK, MOVE COMMAND INTO A
0B06 D390                  OUT     COMM73          ;OUTPUT COMMAND
0B08 78            PAR1:   MOV     A,B             ;GET PARAMETER COUNT
0B09 A7                    ANA     A               ;IS IT 0?
0B0A C8                    RZ                      ;YES, DONE, RETURN
0B0B 23                    INX     H               ;INC COMMAND BUFFER POINTER
0B0C 05                    DCR     B               ;DEC PARAMETER COUNT
0B0D DB90          PAR2:   IN      STAT73          ;READ STATUS
0B0F E620                  ANI     CPBF            ;IS CPBF BIT SET?
0B11 C20D0B                JNZ     PAR2            ;WAIT TIL ITS 0
0B14 7E                    MOV     A,M             ;OK, GET PARAMETER FROM BUFFER
0B15 D391                  OUT     PARM73          ;OUTPUT PARAMETER
0B17 C3080B                JMP     PAR1            ;GET NEXT PARAMETER
                   ;
                   ;
                   ;INITIALIZE AND ENABLE RX DMA CHANNEL
                   ;
                   ;
0B1A 3E62          RXDMA:  MVI     A,DRDMA         ;DISABLE RX DMA CHANNEL
0B1C D3A8                  OUT     MODE57          ;8257 MODE PORT
0B1E 010082                LXI     B,RXBUF         ;RX BUFFER START ADDRESS
0B21 79                    MOV     A,C             ;RX BUFFER LSB
0B22 D3A0                  OUT     CH0ADR          ;CH0 ADR PORT
0B24 78                    MOV     A,B             ;RX BUFFER MSB
0B25 D3A0                  OUT     CH0ADR          ;CH0 ADR PORT
0B27 01FF41                LXI     B,RXTC          ;RX CH TEERMINAL COUNT
0B2A 79                    MOV     A,C             ;RX TERMINAL COUNT LSB
0B2B D3A1                  OUT     CH0TC           ;CH0 TC PORT
0B2D 78                    MOV     A,B             ;RX TERMINAL COUNT MSB
0B2E D3A1                  OUT     CH0TC           ;CH0 TC PORT
0B30 3E63                  MVI     A,ENDMA         ;ENABLE DMA WORD
0B32 D3A8                  OUT     MODE57          ;8257 MODE PORT
0B34 C9                    RET                     ;RETURN
                   ;
                   ;
                   ;INITIALIZE AND ENABLE TX DMA CHANNEL
                   ;
                   ;
0B35 3E61          TXDMA:  MVI     A,DTDMA         ;DISABLE TX DMA CHANNEL
0B37 D3A8                  OUT     MODE57          ;8257 MODE PORT
0B39 010080                LXI     B,TXBUF         ;TX BUFFER START ADDRESS
0B3C 79                    MOV     A,C             ;TX BUFFER LSB
0B3D D3A2                  OUT     CH1ADR          ;CH1 ADR PORT
0B3F 78                    MOV     A,B             ;TX BUFFER MSB
0B40 D3A2                  OUT     CH1ADR          ;CH1 ADR PORT
0B42 01FF81        TXDMA1: LXI     B,TXTC          ;TX CH TERMINAL COUNT
0B45 79                    MOV     A,C             ;TX TERMINAL COUNT LSB
0B46 D3A3                  OUT     CH1TC           ;CH1 TC PORT
0B48 78                    MOV     A,B             ;TX TERMINAL COUNT MSB
0B49 D3A3                  OUT     CH1TC           ;CH1 TC PORT
0B4B 3E63                  MVI     A,ENDMA         ;ENABLE DMA WORD
0B4D D3A8                  OUT     MODE57          ;8257 MODE PORT
0B4F C9                    RET                     ;RETURN
                   ;
                   ;
```

```
                   ;INERRUPT PROCESSING SECTION
                   ;
0C00                         ORG     0C00H
                   ;
                   ;
                   ;RECEIVER INTERRUPT - RST 7.5 (LOC 3CH)
                   ;
0C00 E5            RXI:      PUSH    H              ;SAVE HL
0C01 F5                      PUSH    PSW            ;SAVE PSW
0C02 C5                      PUSH    B              ;SAVE BC
0C03 D5                      PUSH    D              ;SAVE DE
0C04 3E62                    MVI     A,DRDMA        ;DISABLE RX DMA
0C06 D3A8                    OUT     MODE57         ;8257 MODE PORT
0C08 3E18                    MVI     A,18H          ;RESET RST7.5 F/F
0C0A 30                      SIM
0C0B 1604                    MVI     D,04H          ;D IS RESULT COUNTER
0C0D 2A1020                  LHLD    LDADR          ;GET LOAD POINTER
0C10 E5                      PUSH    H              ;SAVE IT
0C11 E5                      PUSH    H              ;SAVE IT AGAIN
0C12 45                      MOV     B,L            ;SAVE LSB
0C13 2A1320                  LHLD    CNADR          ;GET CONSOLE POINTER
0C16 04            RXI1:     INR     B              ;BUMP LOAD POINTER LSB
0C17 78                      MOV     A,B            ;GET SET TO TEST
0C18 BD                      CMP     L              ;LOAD=CONSOLE?
0C19 CAE40A                  JZ      BUFFUL         ;YES, BUFFER FULL
0C1C 15                      DCR     D              ;DEC COUNTER
0C1D C2160C                  JNZ     RXI1           ;NOT DONE, TRY AGAIN
0C20 1605                    MVI     D,05H          ;RESET COUNTER
0C22 E1                      POP     H              ;RESTORE LOAD POINTER
0C23 DB90          RXI2:     IN      STAT73         ;READ STATUS
0C25 E608                    ANI     RXINT          ;TEST RX INT BIT
0C27 CA390C                  JZ      RXI3           ;DONE, GO FINISH UP
0C2A DB90                    IN      STAT73         ;READ STATUS AGAIN
0C2C E602                    ANI     RXIRA          ;IS RESULT READY?
0C2E CA230C                  JZ      RXI2           ;NO, TEST AGAIN
0C31 DB93                    IN      RXIR73         ;YES, READ RESULT
0C33 77                      MOV     M,A            ;STORE IN BUFFER
0C34 2C                      INR     L              ;INC BUFFER POINTER
0C35 15                      DCR     D              ;DEC COUNTER
0C36 C3230C                  JMP     RXI2           ;GET MORE RESULTS
0C39 7A            RXI3:     MOV     A,D            ;GET SET TO TEST
0C3A A7                      ANA     A              ;ALL RESULTS?
0C3B CA450C                  JZ      RXI4           ;YES, SO FINISH UP
0C3E 3600                    MVI     M,00H          ;NO, LOAD 0 TIL DONE
0C40 2C                      INR     L              ;BUMP POINTER
0C41 15                      DCR     D              ;DEC COUNTER
0C42 C3390C                  JMP     RXI3           ;GO AGAIN
0C45 221020        RXI4:     SHLD    LDADR          ;UPDATE LOAD POINTER
0C48 3A1520                  LDA     PRMPT          ;GET MODE INDICATOR
0C4B FE2D                    CPI     '-'            ;NORMAL MODE?
0C4D CA850C                  JZ      RXI6           ;YES, CLEAN UP BEFORE RETURN
                   ;
                   ;         POLL MODE SO CHECK CONTROL BYTE
                   ;         IF CONTROL IS A POLL, SET UP SPECIAL TX COMMAND BUFFER
                   ;         AND RETURN WITH POLL INDICATOR NOT 0
                   ;
0C50 E1                      POP     H              ;GET PREVIOUS LOAD ADR POINTER
0C51 7E                      MOV     A,M            ;GET IC BYTE FROM BUFFER
```

```
0C52 E61E                 ANI    1EH         ;LOOK AT GOOD FRAME BITS
0C54 C2890C               JNZ    RXI5        ;IF NOT 0, INTERRUPT WASN'T FROM A GOOD FRAME
0C57 2C                   INR    L           ;BYPASS R0 AND R1 IN BUFFER
0C58 2C                   INR    L
0C59 2C                   INR    L
0C5A 56                   MOV    D,M         ;GET ADR BYTE AND SAVE IT IN D
0C5B 2C                   INR    L
0C5C 7E                   MOV    A,M         ;GET CNTL BYTE FROM BUFFER
0C5D FE93                 CPI    SNRMP       ;WAS IT SNRM-P?
0C5F CA6C0C               JZ     T1          ;YES, GO SET RESPONSE
0C62 FE11                 CPI    RR0P        ;WAS IT RR(0)-P?
0C64 C2890C               JNZ    RXI5        ;YES, GO SET RESPONSE, OTHERWISE RETURN
0C67 1E11                 MVI    E,RR0F      ;RR(0)-P SO SET RESPONSE TO RR(0)-F
0C69 C36E0C               JMP    TXRET       ;GO FINISH LOADING SPECIAL BUFFER
0C6C 1E73         T1:     MVI    E,NSAF      ;SNRM-P SO SET RESPONSE TO NSA-F
0C6E 212020       TXRET:  LXI    H,CMDBF1    ;SPECIAL BUFFER ADR
0C71 36C8                 MVI    M,0C8H      ;LOAD TX FRAME COMMAND
0C73 23                   INX    H           ;INC POINTER
0C74 3600                 MVI    M,00H       ;L0=0
0C76 23                   INX    H           ;INC POINTER
0C77 3600                 MVI    M,00H       ;L1=0
0C79 23                   INX    H           ;INC POINTER
0C7A 72                   MOV    M,D         ;LOAD RCVD ADR BYTE
0C7B 23                   INX    H           ;INC POINTER
0C7C 73                   MOV    M,E         ;LOAD RESPONSE CNTL BYTE
0C7D 3E01                 MVI    A,01H       ;SET POLL INDICATOR NOT 0
0C7F 321620               STA    POLIN       ;LOAD POLL INDICATOR
0C82 C3890C               JMP    RXI5        ;RETURN

0C85 E1           RXI6:   POP    H           ;CLEAN UP STACK IF NORMAL MODE
0C86 C3890C               JMP    RXI5        ;RETURN

0C89 CD1A0B       RXI5:   CALL   RXDMA       ;RESET DMA CHANNEL
0C8C D1                   POP    D           ;RESTORE REGISTERS
0C8D C1                   POP    B
0C8E F1                   POP    PSW
0C8F E1                   POP    H
0C90 FB                   EI                 ;ENABLE INTERRUPTS
0C91 C9                   RET                ;RETURN
                  ;
                  ;
                  ;MESSAGE TYPER - ASSUMES MESSAGE STARTS AT HL
                  ;
                  ;
0C92 C5           TYMSG:  PUSH   B           ;SAVE BC
0C93 7E           TYMSG2: MOV    A,M         ;GET ASCII CHR
0C94 23                   INX    H           ;INC POINTER
0C95 FEFF                 CPI    0FFH        ;STOP?
0C97 CAA10C               JZ     TYMSG1      ;YES, GET SET FOR EXIT
0C9A 4F                   MOV    C,A         ;SET UP FOR DISPLAY
0C9B CDF805               CALL   ECHO        ;DISPLAY CHR
0C9E C3930C               JMP    TYMSG2      ;GET NEXT CHR
0CA1 C1           TYMSG1: POP    B           ;RESTORE BC
0CA2 C9                   RET                ;RETURN
                  ;
                  ;
                  ;SIGNON MESSAGE
                  ;
```

```
0CA3 0D            SIGNON: DB      CR,'8273 MONITOR  V1.1',CR,0FFH
0CA4 38323733
0CA8 204D4F4E
0CAC 49544F52
0CB0 20205631
0CB4 2E31
0CB6 0D
0CB7 FF
                   ;
                   ;
                   ;
                   ;RECEIVER INTERRUPT MESSAGES
                   ;
                   ;
0CB8 0D            RXIMSG: DB      CR,'RX INT - ',0FFH
0CB9 52582049
0CBD 4E54202D
0CC1 20
0CC2 FF
                   ;
                   ;TRANSMITTER INTERRUPT MESSAGES
                   ;
0CC3 0D            TXIMSG: DB      CR,'TX INT - ',0FFH
0CC4 54582049
0CC8 4E54202D
0CCC 20
0CCD FF
                   ;
                   ;
                   ;TRANSMITTER INTERRUPT ROUTINE
                   ;
0CCE E5            TXI:    PUSH    H               ;SAVE HL
0CCF F5                    PUSH    PSW             ;SAVE PSW
0CD0 C5                    PUSH    B               ;SAVE BC
0CD1 D5                    PUSH    D               ;SAVE DE
0CD2 3E61                  MVI     A,DTDMA         ;DISABLE TX DMA
0CD4 D3A8                  OUT     MODE57          ;8257 MODE PORT
0CD6 1604                  MVI     D,04H           ;SET COUNTER
0CD8 2A1020                LHLD    LDADR           ;GET LOAD POINTER
0CDB E5                    PUSH    H               ;SAVE IT
0CDC 45                    MOV     B,L             ;SAVE LSB IN B
0CDD 2A1320                LHLD    CNADR           ;GET CONSOLE POINTER
0CE0 04            TXI1:   INR     B               ;INC POINTER
0CE1 78                    MOV     A,B             ;GET SET TO TEST
0CE2 BD                    CMP     L               ;LOAD=CONSOLE?
0CE3 CAE40A                JZ      BUFFUL          ;YES, BUFFER FULL
0CE6 15                    DCR     D               ;NO, TEST NEXT LOCATION
0CE7 C2E00C                JNZ     TXI1            ;TRY AGAIN
0CEA E1                    POP     H               ;RESTORE LOAD POINTER
0CEB DB92                  IN      TXIR73          ;READ RESULT
0CED 77                    MOV     M,A             ;STORE IN BUFFER
0CEE 2C                    INR     L               ;INR POINTER
0CEF 3600                  MVI     M,00H           ;EXTRA RESULT SPOTS 0
0CF1 2C                    INR     L
0CF2 3600                  MVI     M,00H
0CF4 2C                    INR     L
0CF5 3600                  MVI     M,00H
0CF7 2C                    INR     L
```

```
0CF8 3600      892     MVI    M,00H
0CFA 2C        893     INR    L
0CFB 221020    894     SHLD   LDADR         ;UPDATE LOAD POINTER
0CFE CD350B    899     CALL   TXDMA         ;RESET DMA CHANNEL
0D01 D1        900     POP    D             ;RESTORE DE
0D02 C1        901     POP    B             ;RESTORE BC
0D03 F1        902     POP    PSW           ;RESTORE PSW
0D04 E1        903     POP    H             ;RESTORE HL
0D05 FB        904     EI                   ;ENABLE INTERRUPTS
0D06 C9        905     RET                  ;RETURN
               906 ;
               907 ;
               952 ;
               953 ;
               954     END

PUBLIC SYMBOLS

EXTERNAL SYMBOLS

USER SYMBOLS
ADWN   A 0922  AFCMD  A 09CE  BUFFUL A 0AE4  CH0ADR A 00A0  CH0TC  A 00A1  CH1ADR A 00A2  CH1TC  A 00A3
CMD51  A 0027  CMDBF1 A 2020  CMDBUF A 2000  CMDOUT A 0AFB  CMDREC A 0857  CMODE  A 0931  CNADR  A 2013
CNT053 A 009C  CNT153 A 009D  CNT253 A 009E  CNTL51 A 0089  CNTLC  A 0003  CNVBN  A 05B8  COBR   A 000C
COMM   A 0AE5  COMM1  A 0AEA  COMM2  A 0AFF  COMM73 A 0090  CPBF   A 0020  CR     A 000D  CRLF   A 05EB
DEM    A 0000  DEMODE A 2027  DISPY  A 0A39  DISPY1 A 0A4E  DISPY2 A 0A4D  DRDMA  A 0062  DTDMA  A 0061
ECHO   A 05F8  ENDCHK A 0A1B  ENDMA  A 0063  GDWN   A 08FF  GETCH  A 061F  GETCMD A 087D  GRCMD  A 09C4
ILLEG  A 08A7  LDADR  A 2010  LF     A 000A  LKBR1  A 2017  LKBR2  A 2018  LOOPIT A 0861  MDCNT0 A 0036
MDCNT2 A 00B6  MDE51  A 00CE  MODE53 A 009B  MODE57 A 00A8  MONTOR A 0008  NMOUT  A 06C7  NSAF   A 0073
PAR1   A 0B08  PAR2   A 0B0D  PARIN  A 0AAD  PARIN1 A 0AE0  PARIN2 A 0ADC  PARIN3 A 0ABC  PARM73 A 0091
POLIN  A 2016  PRMPT  A 2015  R0PT   A 0AA2  R1PT   A 0AA7  RBCMD  A 097B  RDCMD  A 0971  RDWN   A 08AF
RDY    A 0002  RESBUF A 2800  RESL73 A 0091  ROCMD  A 095D  RPCMD  A 09D8  RR0F   A 0011  RR0P   A 0011
RSCMD  A 0967  RST65  A 20CE  RST75  A 20D4  RXBUF  A 8200  RXD51  A 0088  RXDMA  A 081A  RXI    A 0C00
RXI1   A 0C16  RXI2   A 0C23  RXI3   A 0C39  RXI4   A 0C45  RXI5   A 0C89  RXI6   A 0C85  RXIMSG A 0CB8
RXINT  A 0006  RXIR73 A 0093  RXIRA  A 0002  RXS1   A 0A69  RXS2   A 0A7F  RXS3   A 0A8D  RXSORC A 0A62
RXTC   A 41FF  SBCMD  A 0985  SDWN   A 08D7  SIGNON A 0CA3  SLCMD  A 098F  SNRMP  A 0093  SOCMD  A 09A6
SPCMD  A 09E2  SRCMD  A 09BA  SSCMD  A 09B0  START  A 0800  STAT51 A 0089  STAT57 A 00A8  STAT73 A 0090
STKSRT A 20C0  SW     A 0943  T1     A 0C6C  TBUFFL A 0A24  TBUFL  A 0A07  TBUFL1 A 0A0D  TDWN   A 090E
TEST73 A 0092  TFCMD  A 09EC  TFCMD1 A 09F6  TFRET  A 0A36  TLCMD  A 0999  TRUE   A 0000  TRUE1  A 0000
TXBUF  A 8000  TXD51  A 0088  TXDMA  A 0B35  TXDMA1 A 0B42  TXI    A 0CCE  TXI1   A 0CE0  TXIMSG A 0CC3
TXINT  A 0004  TXIR73 A 0092  TXIRA  A 0001  TXPOL  A 094C  TXRET  A 0C6E  TXSORC A 0A47  TXTC   A 81FF
TYMSG  A 0C92  TYMSG1 A 0CA1  TYMSG2 A 0C93  VALDG  A 075E

ASSEMBLY COMPLETE,  NO ERRORS
```