HARDWARE
SOFTWARE
SIERRA DATA SCIENCES

ZS10 Board

21162 Lorain Avenue   Fairview Park, Ohio 44126   (216) 331-8500

ZSIO

Users & Technical Manual

TABLE OF CONTENTS

Figures and Appendices

SECTION 1.0 Hardware Configuration

The ZSIO Board is shipped with the following configuration:

1) The RS-232 interface is wired such that any dumb terminal can be connected to the ZSIO without the Modem Signal Direction Area being altered. (Pin 2, 3 and 7 connection)

2) Interrupt Priority is (in descending order)
   a. ZSIO chip #1 (Channels A and B)
   b. ZSIO chip #2 (Channels C and D)
   c. CTC chip

3) Channel 0 of the CTC serves as the BAUD rate generator for Serial Channel A. Channel 1 of the CTC serves as the BAUD rate generator for Serial Channel B. Channel 2 of the CTC serves as the BAUD rate generator for Serial Channels C and D.

4) No wait states are inserted during the INTA cycle

This configuration can easily be altered to suit your particular requirements. To facilitate these modifications, seven areas (labeled A through G) and an eight position Dip Switch have been incorporated into the Board.

The following sections detail the functions of each of the eight modifiable items. Section 1.9 describes the procedure for connecting the Real Time Clock to the system.

SECTION 1.1 AREA A; Modem Signal Direction Header (four places)

The configuration of this area determines whether the RS-232 interface will appear as a terminal or as a modem. Examples of the usage of Area A are provided in Figures A1 through A4: **Modem Signal Direction Header.**

The Board is prewired to appear as a modem. The Transmitted Data (T x D) and Received Data (R x D) signals are strapped on the PC board (Figure A2). This configuration will suffice for those applications in which the Board is to communicate with a terminal that does not require any 'handshaking'. Some examples of this type of terminal are the Soroc, Mime IIA and Infoton.

When your application requires it, the modem control signals may be strapped with shorting pins.

Please note that the Board is shipped with Jumper C prewired (Figure A2). This configuration causes the Carrier-Generate signal to be looped back to the Carrier-Detect signal. A detailed explanation of this feature is presented in Section 6.0: **Using Auto Enables.**

To cause a channel to act as a terminal, cut the three traces on the back side of the board and insert the necessary jumpers. In the example that is presented in Figure A3, a channel of the ZSIO is connected to a modem. All of the 'handshaking' signals are being used.

Figure A3 also provides an example of a channel that generates both the receiver and the transmitter clocks, Jumpers K and L. **These jumpers must not be inserted when the channel is configured to accept the clocks from the RS-232 interface (Figure B3).** For example, if Channel A were to generate the receiver and transmitter clocks, then Jumpers K and L would be inserted in the Modem Signal Direction Header for Channel A. Jumpers I and J would not be inserted as in Figure B3. If the clocks were to be generated from the RS-232 interface, then Jumpers I and J would be inserted. Jumpers K and L for the A Channel direction header would not be inserted as in Figure A3.

INTRODUCTION

Description:

The ZSIO Board is a four port RS-232 Serial I/O and Real Time Clock board. By establishing an interface to the powerful Zilog SIO chips, it provides your micro-computer system with a link to the outside world.

The ZSIO Board was designed to meet the needs of multi-user oriented systems as well as systems requiring state-of-the-art data communications. Three design criteria were set forth and achieved:

1) To interface to the IEEE Standard Buss (this is the name given the S-100 buss by the IEEE Standards Board)
2) To be compatible with as many previously designed boards as possible
3) To cause the Zilog SIO and CTC support chips to "think" that they were connected to a Z-80 Processor.

The Board is ideal for CP/M, MPM and Alpha Micro based systems. Applications include Multi-User data processing with high speed data communications, using state-of-the-art communication protocols.

Though the Board can be port polled, maximum system throughput is achieved through the use of the interrupt facilities of the Zilog ZSIO/2 and CTC LSI chips. The RESTART instruction is used for interrupt processing on the 8080 CPU. Interrupt Mode 2 can be used for the Z-80 and a 'free' Vector line is used for Alpha Micro based systems.

Full documentation is included with each board. This includes:

User's Manual
Theory of Operation
Schematic
Application Notes

Advantages:

Can be completely Interrupt Driven, with no additional hardware required to process RESTART Instructions or Interrupt Vectors

Incorporates a REAL TIME CLOCK that is programmable in increments of 1 to 255 times 1/60 second

Can be configured to appear as a Modem or a Terminal with solderless Direction Jumpers

Unique Interrupt Vectors can be generated for the following conditions:

A. Transmitter Buffer Empty
B. External Status Change (DCD or CTS changes)
C. Received Characters Available
D. Special Receive Characters (Parity, Error, Overrun, Framing Error, End of Frame in SDLC mode)

Features:

Baud rates programmable from 75 to 19.2K
Four Independent Full Duplex RS-232 Channels
0-880K Bits/Second
Doubly-buffered Transmitter data registers
Quadruply-buffered Receiver data registers
Program control of Asynchronous Characteristics such as stop bits, bits/character and parity
Program control of Sync Characters in Synchronous Mode
CRC Generation and Checking
HDLC and SDLC Communication Protocols

Figure A4 illustrates the procedure for causing the Modem Signal Direction area to loop back the 'handshaking' signals. This feature is especially useful during testing or when the software to bring up the needed signals is not available.

When operating the ZSIO Board with slow devices, such as printers, the busy signal from the device may be routed to either CTS (RS-232 Pin 5) or DTR (RS-232 Pin 20). The SIO chip can then be programmed to cause the enabling of the transmitter when CTS is TRUE. Additionally, the receiver can be enabled by the use of the Carrier-Detect signal (RS-232 Pin 8). For detailed explanations of these procedures, refer to Section 6.0: **Using Auto Enables.**

SECTION 1.2 AREA B; Receiver and Transmitter Clock Header (one place)

The configuration of this area determines the origin of the Receiver and Transmitter Clocks signals on the SIO chip. Examples of the usage of Area B are provided in Figures B1 through B4: **Receiver and Transmitter Clock Header.**

Figure B1 specifies the hardware configuration for the clock header.

The top portion of this header consists of all of the clock sources. R x CLKA and T x CLKA are Pins 15 and 17, respectively, from the RS-232 interface for Channel A. R x CLKB and T x CLKB are Pins 15 and 17 for Channel B, etc. CTC(0), CTC(1) and CTC(2) are three baud rate generators from the CTC chip.

On the bottom portion of the header are the Receiver and Transmitter inputs to the ZSIO Channels A through D, labeled accordingly.

The Receiver and Transmitter Clock Header is prewired as follows (Figure B2):

Channel A Clocks accept inputs from Channel O of the CTC
Channel B Clocks accept inputs from Channel 1 of the CTC
Channels C and D Clocks accept inputs from Channel 2 of the CTC

Since the CTC has only three outputs, two channels of Serial I/O must be run at the same baud rate if all four channels are running asynchronous. Channel D may receive its clock inputs from any of the CTC Channels (Figure B4).

If it is necessary to clock any of the SIO Channels from the RS-232 interface, then cut the two traces that are prewired for that channel and add the appropriate jumpers. Figure B3 shows all four channels clocked from the RS-232 interface. **As described in Section 1.1, if any of the channels are jumpered, as in Figure B3, then the Modem Signal Direction Header for that channel must not have its clock jumpers inserted.**

Figure B4 shows the clock header configured as follows:

1) CTC(0) serves as the baud rate generator for serial Channels A and D
2) CTC(1) serves as the baud rate generator for serial Channel B
3) Channel C is receiving its clock inputs from the corresponding RS-232 interface

SECTION 1.3 AREA C (one place)

The configuration of this area determines if the Zilog chips are reading the Data-In-Buss during the M1 state. Area C is pertinent only in those applications where the board is used in an interrupt driven environment.

Z-80 Processors execute a special instruction for resetting the interrupt priority logic of their support circuits. If this RETI instruction is used, then you should place the shunt in the M1 column of Area C. If this instruction cannot be used or a processor other than a Z-80 processor is in the system, then you should place the shunt in the INTA column and refer to Section 2.0: **Resetting the Interrupt Priority Logic of the SIO and CTC.**

SECTION 1.4 AREA D (one place, under U32 and U33)

The configuration of this area determines whether or not a Wait State can be inserted by the ZSIO Board and, if so, on which pin it is active.

When used with certain processors, the ZSIO Board requires the insertion of Wait states during Inputs, Outputs, and INTA cycles. If required (jumper inserted), then it is necessary to specify whether it is active on Pin 3 or Pin 72 of the S-100 Buss.

The output is an open collector gate. It should be connected to Pin 72 according to the IEEE specs on the 696 Standard. The board is prewired using this technique. Different systems may require this signal to be placed on Pin 3, so the option is available if needed.

Refer to Section 3.0 for information on wait states for the ZSIO Board.

SECTION 1.5 AREA E (one place, under left SIO chip U14)

This area determines the on-board priority of the three Zilog chips when using the board in an interrupt driven system. Area E has no significance in a port polling environment.

The Board is shipped prewired with the following priorities (in descending order of priority):

1) U14, Channels A and B
2) U16, Channels C and D
3) CTC

Refer to Figure C for information on how this priority allocation may be altered.

SECTION 1.6 AREA F (one place, lower left Pins 4 thru 11 on S-100 buss)

The configuration of this area allows several ZSIO Boards to be 'daisy-chained' together.

The highest priority Board would have nothing connected to the IN pad. Its OUT pad would be connected to one of Pins 4 thru 11 on the S-100 Buss or to any unused pin in a given system.

The next Board in the chain would have this same pin connected to its IN pad. Its OUT pad would be connected to the next desired S-100 pin. This logic would continue through any remaining boards.

Refer to Figure D for a pictorial view of this procedure.

SECTION 1.7 AREA G (one place center of board)

The configuration of this area (jumper) is used to maximize the setup time between IORQ and the System Clock.

This area is prewired in the Z-80 column. When the Board is to be used with a Z-80 or Alpha Micro Processors, this area should not be modified.

When used with an 8080, the following procedures should be performed:

1) Cut the trace on the back side of the Board
2) Insert a jumper in the 8080 column

SECTION 1.8 Eight Position Dip Switch

The configuration of this switch determines the base port address for the Board, the number of Wait States that are to be inserted during the INTA cycle, and whether or not the Board can cause an interrupt.

Switch positions 1 through 4 determine the base port address:

Switch #1 = Address bit 7
Switch #2 = Address bit 6
Switch #3 = Address bit 5
Switch #4 = Address bit 4

A closed switch represents a binary one. For example, if Switch positions one, three, and four were closed then the base port address for the ZSIO board would be BOH.

Switch positions 5, 6 and 7 form a 3-bit code that determines the number of Wait States.

One or more Wait States must be inserted during INTA whenever multiple ZSIO Boards are used in a system. This allows the IEI and IEO daisy-chain signals to stabilize and subsequently address the proper SIO or CTC chip.

The number of Wait States to be inserted is determined by subtracting the 3-position switch setting (read like a binary number with a closed switch representing a 0) from seven. For example, if Switches 5 and 7 are closed, the switch setting is '010' and the number of Wait States inserted during INTA would be 7 minus 2 or 5. If switch positions 6 and 7 were closed, then the number of Wait States inserted would be 7 minus 4 or 3.

If only one ZSIO board is used in the system, then no Wait States are required during INTA and Switches 5, 6 and 7 should be off. If two ZSIO boards are daisy-chained together, then one Wait State should be inserted during INTA. In this case, Switch 7 would be closed (7 - 6 = 1). If three boards are daisy-chained together, two wait states should be inserted during INTA. Switch 6 would be closed (7 - 5 = 2).

Switch position 8 is a hardware Interrupt Enable for the board. If the switch is closed, the board can cause an interrupt; if it is open, the board cannot cause an interrupt.

Note: The above section describes Switch positions 1 through 8, but different Dip Switch manufacturers may number the same positions 0 through 7. In this case, positions 0 through 3 would be the board address etc.

SECTION 1.9 Real Time Clock Connection

There are two methods available for causing the CTC's Channel 3 to create timing intervals.

The first method involves programming the CTC chip to count the number of system clock cycles. This approach results in a time interval that is equal to T x P x TC (Appendix B. page 10) where:

T = System Clock Period
P = Prescaler Constant
TC = Time Constant

For example, if the system is running at 4MHz, the prescaler is set to 255 and the time constant is set to 255, then the time interval would be 16.384 milli-seconds (apprx. 1/60 second).

The second method involves causing the CTC Channel 3 to count the number of clock ticks off of the 60 cycle A.C. line. This approach results in the time intervals being equal to TC x 16.666 msec intervals. Again, TC is equal to the Time Constant register. This method provides the capability of generating time intervals that exceed 4 seconds and can also result in a more accurate time slice.

To use this method, a wire must be connected between the ZSIO Board and the power supply. The Real Time Clock Pin is located in the upper right hand part of the Board. It must be connected to the secondary side of the power supply transformer, ideally to the source for the +8 volt supply. This connection is accomplished as follows:

1) Connect the clip to the transformer tap
2) Insert the wire into the RTC Pin
3) Press the wire into place with the RTC cap, using a Phillips head screwdriver

SECTION 2.0 Resetting Interrupt Priority Logic of SIO and CTC

This section describes the techniques that are available for resetting the Interrupt Priority Logic of the SIO and CTC chips on the ZSIO Board. This information is applicable only to interrupt-driven systems.

SECTION 2.1 Using the RETI Instruction

A special Z-80 Instruction, RETI (ED 4D), is monitored by the SIO and CTC chips. Whenever the chips detect the ED opcode on the buss during the M1 cycle, the IEI and IEO Pins of the SIO and CTC change state. This results in the interrupting chip being addressed. If the subsequent opcode is a 4D, then the Interrupt Priority Logic of the appropriate chip is reset.

Due to the IEI and IEO time delays that are inherent in the Zilog support circuits, only four chips can be daisy-chained together in a 4MHz system that uses Z-80A SIO and Z-80A CTC chips. Eight chips can be chained together in 2 MHz systems. Extra support logic has been incorporated onto the ZSIO Board which allows several boards to be linked together. This feature enables any number of SIO and CTC chips to be daisy-chained.

The following sections describe the procedure for handling the Interrupt Priority Logic in these instances. For additional information on this topic, refer to Appendix A of the SIO Technical Manual (page 41).

If the interrupt priority logic of the SIO and CTC chips is to be reset with the RETI instruction, Area C must have a jumper in the M1 column.

SECTION 2.2 Using the ZSIO RETI port

When the RETI instruction is not available or daisy-chaining problems arise, an alternative method for resetting the Interrupt Priority Logic may be employed.

A ZSIO Board responds to an Output on port FE Hex in such a way that the SIO and CTC chips will "think" that the processor is in an M1 cycle. By writing an ED Hex* value followed by a 4D Hex* to this port, ample time will become available for the IEI and IEO signals to address the proper chip. Note that this port is common to all ZSIO Boards daisy-chained in the system, therefore, any chip located on a ZSIO Board will respond correctly by toggling its IEI and IEO lines.

If the interrupt priority logic is to be reset by writing out ED followed by a 4D to port FEH, then Area C should have a jumper in the INTA column.

* These two codes equal the opcode for an RETI instruction.

SECTION 2.3 Using the RET from Interrupt Command

If only SIO chips are being activated in a system, then the Interrupt Priority Logic of these chips can be reset with the RET from Interrupt Command. Refer to Appendix A of the SIO Technical Manual for additional information. **In this situation, Area C is not pertinent.**

SECTION 3.0 Wait States During INTA, INPUT and OUTPUT Cycles

The Z-80 processor automatically inserts one Wait State during Input and Output cycles, However, as several Z-80 processor cards inhibit this characteristic, it is often required that the ZSIO Board insert an additional Wait State during Input and Output Instructions. The Board is shipped wired to do just that. Since this single Wait State is only inserted when the ZSIO Board is performing an Input or Output cycle, the system's throughput is not hampered.

On a Z-80 processor, there exists a time period during which all Z-80 support circuits are allowed to compete for service. A Z-80 inserts two Wait States during an INTA. The two wait states allow sufficient time for the IEI and IEO ripple signals to stabilize and identify which I/O device must insert the response vector. This time interval can be extended by use of Switch positions 5, 6 and 7, as described in Section 1.8. Such an extension would assure that ripple time delays do not cause erroneous vectors to be placed onto the buss.

Wait States are also inserted by the ZSIO Board whenever the user is using the Wait function of the SIO chip. Details on the procedure for programming the SIO in this manner are presented in Appendix A (Z-80 Programming; Write Register 1). While in this mode, a block I/O instruction can be issued to the ZSIO Board. The Wait line will become active whenever the CPU attempts to perform one of the following activities:

1) Write data while the transmit buffer is full
2) Read data that has not yet been received

If the Wait/Ready function of the SIO chip is to be used, then a wait jumper must be inserted (Section 1.4)

Software Configuration

Sections 5, 6 and 7 that follow describe how to program the SIO and CTC. Additional information on the procedure for programming the SIO is provided in Appendix A, The SIO Technical Manual, and Appendix C, The SIO Application Manual. CTC programming techniques may be found in Appendix B, The CTC Technical Manual.

SECTION 5.0 Using the CTC Channels

The CTC chip has four channels, referenced in Appendix B of the CTC Technical Manual as Channels 0 through 3. As it has only three Outputs, the maximum number of unique baud rates that it can generate is also three. If all four channels are running asynchronously, then two channels must process at the same baud rate.

The board is shipped with Area B (Section 1.2) prewired as follows:

Channel A connected to CTC Channel 0
Channel B connected to CTC Channel 1
Channels C and D connected to CTC Channel 2

The baud rate of each channel is determined by using the following formula:

Baud Rate = F / (CTC x SIOCLK)

Where F is equal to 921.6KHz, CTC is the HEX number programmed into the CTC time constant register and SIOCLK is the SIO chip dividing factor.

The following table details the values to be used in programming the CTC time constant register and SIO for the commonly used baud rates.

CTC Time Constant REG.	Dividing Factor	Baud Rate
03H	X16	19.2K
06H	X16	9600
08H	X16	7200
0CH	X16	4800
10H	X16	3600
18H	X16	2400
20H	X16	1800
30H	X16	1200
60H	X16	600
C0H	X16	300
C0H	X32	150
83H	X64	110
C0H	X64	75

Before the time-constant register can be programmed, the CTC channel control register must be programmed with a 55H. Refer to Appendix B for information on how this value was obtained.

Channel 3 of the CTC is used as a general purpose timer. It may be used in either a counting or a timer mode, as described in Appendix B of the CTC technical manual.

When in counting mode, the CTC counts the number of system clock cycles. This can result in a short time interval. The longest possible period is obtained from the following formula: 255 x 255 x T, where T represents the system clock period (250 nsec. running at 4MHz).

SECTION 4.0 Port Definitions

The ZSIO Board is I/O mapped and occupies a total of 14 ports. The base address of each board is determined by Switches 1 through 4, as described in Section 1.8. The following text refers to this port base address as XXXX. The ZSIO Board's Serial Channels are referred to as Channels A through D. In Appendix A of the SIO Technical Manual, the chip is divided into two channels that are also referenced as Channel A and Channel B.

Please note that the Board's Channels C and D correspond to the SIO Manual's Channels A and B, respectively.

The actual board addresses are as follows:

XXXX0000 = X0H - Board Channel A Data Input and Output Port
XXXX0001 = X1H - Board Channel A Status and Control Port
XXXX0010 = X2H - Board Channel B Data Input and Output Port
XXXX0011 = X3H - Board Channel B Status and Control Port
XXXX0100 = X4H - Board Channel C Data Input and Output Port
XXXX0101 = X5H - Board Channel C Status and Control Port
XXXX0110 = X6H - Board Channel D Data Input and Output Port
XXXX0111 = X7H - Board Channel D Status and Control Port

XXXX1000 = X8H - CTC Channel 0 = Baud Rate Generator Channel A
XXXX1001 = X9H - CTC Channel 2 = Baud Rate Generator Channel B
XXXX1010 = XAH - CTC Channel 2 = Baud Rate Generator Channels C and D
XXXX1011 = XBH - CTC Channel 3 = Real Time Clock Channel

XXXX1100 = XCH - Unused; can be used by any other S-100 board
XXXX1101 = XDH - Unused; can be used by any other S-100 board

XXXX1110 = XEH - Output=Carrier Generator for each Channel
Writing a 1 to this port turns on Carrier
Writing a 0 to this port turns off Carrier

Bit 0 = Carrier for Channel A
Bit 1 = Carrier for Channel B
Bit 2 = Carrier for Channel C
Bit 3 = Carrier for Channel D

Input=DSR from each Channel
DSR is on when a 0 is read from this port
DSR is off when a 1 is read from this port

Bit 0 = DSR from Channel A
Bit 1 = DSR from Channel B
Bit 2 = DSR from Channel C
Bit 3 = DSR from Channel D

XXXX1111 = XFH - Same as XEH

11111110 = FEH - Port used to reset Interrupt Priority Logic of Sio and CTC chips (Section 2.2).

When in timer mode the clock input may be a 60 cycle clock. By programming the time constant register with a number from 1 to 255, a time period from 1/60 to 255/60 seconds is obtainable.

Both of the aforementioned techniques may be used for timing-out one of the serial channels or for time-slicing data processing.

Note that any of the CTC channels not used for baud rate generators are available for generating different time intervals.

SECTION 6.0 Using Auto Enables

One of the special features of the SIO chip is its ability to enable the transmitter with the CTS signal input and the receiver with the DCD signal input. This feature is extremely useful in those applications where your system must communicate with slower devices, such as printers and low speed modems.

This function of the SIO chip is enabled by writing a 1 to Bit 5 in Write Register 3.

For example, a Texas Instrument 810 or 820 dot matrix printer has its Busy signal coming out of pin 11 of its RS-232 interface. When the buffer of the printer becomes full this line goes false. This signal can be wired to pin 5 (CTS) or pin 20 (DTR) on the computer side of the interface. (If this busy signal was connected to pin 5, Jumper G, in Figure A3, would be inserted.) When the printer cannot accept any more data, the SIO CTS input will be forced low and will automatically disable the transmitter of that particular channel. As soon as the T.I. clears out its character buffer, the CTS signal will reverse states and the SIO channel will again be able to transmit data.

If a Receive Only terminal is connected to a channel that is programmed for auto enable, then the receiver of the SIO chip will never require enabling. Only the CTS interface signal will have to be jumpered in the Modem Signal Direction Header.

When interfacing to a device that is both sending and receiving there may be a circumstance where the transmitter is enabled with CTS but the receiver cannot be enabled with DCD. It is for this reason that Jumper C in Figure A2 was prewired on the ZSIO circuit board. By writing out to port XEH, the Carrier-Generate port, you are able to activate DCD. Jumper C then loops the signal back to the DCD input of the SIO chip, thereby enabling the channel receiver.

The following is an example of a solution to a software problem that my arise when interfacing different devices to the ZSIO board. Consider a system which has a slow printer and three high speed terminals. The printer is a receive-only model and the terminals do not require any 'handshaking' signals. Since it is desirable to initialize all four channels with the same initialization driver, all four are programmed with Auto Enable ON. This is done in order to accomodate the slow printer. The three terminals must have both their CTS and their DCD signals True to operate. For this reason, the appropriate DCD-generated bits are turned on by writing out to port XEH. The board is prewired to have DCD generate loop back to DCD detect. This results in the receivers for the three terminals being enabled. Bit 1 in Write Register 5 is turned ON to bring up RTS. This signal is then looped back to CTS in the Modem Signal Direction Header with a shunt and the transmitters are also enabled.

SECTION 7.0 Programming Examples and Tests

You may find the following sections to contain the most helpful parts of the ZSIO Manual. They contain several programming examples which illustrate:

How to set up the ZSIO board for simple asynchronous operation.
How to program the CTC channels for generating baud rates and timing intervals.
How to set up the ZSIO board for specific systems.

By referring to these sections and the Technical Manuals, programming the ZSIO Board should prove to be a relatively simple matter. The short programs contained here can also be used to test various functions of the ZSIO board. Before trying to use the ZSIO board in a certain system, it is highly recommended that these tests be executed. This will guarantee hardware compatibility as well as provide a means by which you can gain familiarity with the Board

SECTION 7.1 Program to Test Receiver Interrupts

The following program tests a channel of the ZSIO board for receiver interrupts. With one ZSIO board in the system, turn on Dip Switches 1, 3, 4 and 8. This sets the base address at B0H.

This program simply enables the first channel of the ZSIO board for receiver interrupts. Any character typed on a terminal connected to channel A will be echoed back.

```
                                  •••••••••••••••••••••••••••••••
                                   Set up CTC for 9600 baud. Init SP
                                    Set up Interrupt Vector as RST 0
                                  •••••••••••••••••••••••••••••••

                                  ORG 0100H
0100        31 00 10              LXI SP,1000H              ;Set up Stack pointer
0103        3E 02                 MVI A,2                   ;Set A to WR2 pointer
0105        D3 B3                 OUT 0B3H                  ;Send A to B side of Chip 1
0107        3E C7                 MVI A,0C7H
0109        D3 B3                 OUT 0B3H                  ;Set WR2 to a RESTART 0
010B        3E 55                 MVI A,55H                 ;CTC(0) to Counting Mode and TC is
010D        D3 B8                 OUT 0B8H                  ;next word to be written to this port
010F        3E 06                 MVI A,6                   ;Time Constant equals 6 which
0111        D3 B8                 OUT 0B8H                  ;sets CTC(0) to 9600 baud
                            •••••••••••••••••••••••••••••••••••••••••••
                             Now program the SIO chip for Receiver Interrupts
                            •••••••••••••••••••••••••••••••••••••••••••

0113        3E 14                 MVI A,14H             ;Reset Ext/Starus Int, and
0115        D3 B1                 OUT 0B1H              ;Set register Pointer to WR4
0117        3E 4C                 MVI A,4CH             ;X16 clock mode, no parity
0119        D3 B1                 OUT 0B1H              ;2 Stop Bits
011B        3E 03                 MVI A,3
011D        D3 B1                 OUT 0B1H              ;Point to WR3
011F        3E 41                 MVI A,41H             ;Set Receiver for 7 bits/char.
0121        D3 B1                 OUT 0B1H              ;and Enable the Receiver
0123        3E 05                 MVI A,5
0125        D3 B1                 OUT 0B1H              ;Point to WR5
0127        3E AA                 MVI A,0AAH            ;Turn on DTR and RTS, Transmitter set to
0129        D3 B1                 OUT 0B1H              ;7 bits/Char., Transmitter enabled
012B        3E 01                 MVI A,1
012D        D3 B1                 OUT 0B1H              ;Set pointer to WR1
012F        3E 18                 MVI A,18H             ;Set SIO to Interrupt on all
0131        D3 B1                 OUT 0B1H              ;Received Characters
                                 ••••••••••••••••••••••••••••••••••
                                  SIO is Setup for Receiver Interrupts
                                 ••••••••••••••••••••••••••••••••••

0133        FB              LOOP: EI                    ;Enable Interrupts
0134        76                    HLT                   ;Don't do anything until Rec. Int. Occurs
0135        C3 33 01              JMP LOOP              ;After you echo character jump back
```

```
                                  ••••••••••••••••••••••••
                                   Interupt Service Routine
                                  ••••••••••••••••••••••••
                           ORG 0
0000        DB B0          IN 0B0H               ;Grab Character that caused Int.
0002        D3 B0          OUT 0B0H              ;Echo it back
0004        AF             XRA                   ;Clear A to 0
0005        D3 B1          OUT 0B1H              ;Make sure WR pointer is at WR0
0007        3E 38          MVI A,38H             ;Reset Int. Pending
0009        D3 B1          OUT 0B1H              ;so that next char. can cause an Int.
000B        C9             RET                   ;Go back to program
```

SECTION 7.2 Program to test Receiver Interrupts (Z-80 system)

The program below is very similar to the one of Section 7.1. In this example, some of the Z-80 expanded instructions are used. Note how much "cleaner" the ZSIO channel is initialized and how the initialization values are organized within a table. Set Dip Switches 1, 3, 4 and 8 ON. This establishes B0H as the base address and enables interrupts. Place the jumper in area C in the M1 column and the jumper in area G in the Z-80 column.

Interrupt Mode 2 is used in which the I Register is the high byte and vector placed on the bus during INTA is the low-order byte of the routine's address.

```
                                 ••••••••••••••••••••••••••
                                  Set up table with Init. Values
                                 ••••••••••••••••••••••••••
                           ORG 0140
0140        14             DB 14H                ;Reset Ext/Status Int. and point to WR4
0141        4C             DB 4CH                ;X16 Clock mode, no parity, 2 stop bits
0142        03             DB 03                 ;Point to WR3
0143        41             DB 41H                ;Set Rec. for 7 bits/char., Enable Rec.
0144        05             DB 05                 ;Point to WR5
0145        AA             DB 0AAH               ;Turn on RTS and DTR, Trans 7 bits/char.
                                                 ;transmitter enabled
0146        01             DB 01                 ;Point to WR1
0147        18             DB 18H                ;Set SIO to Int. on all Received Char.
0148        02             DB 02                 ;Point to WR2 (B channel only)
0149        80             DB 80H                ;Set Vector to 80H
014A        55             DB 55H                ;Set CTC Channel 0 to counting Mode
014B        06             DB 06                 ;Time Constant for 9600 baud

                           ORG 0100H
                                  •••••••••••••••••••••••••
                                   Main program is put here
                                  •••••••••••••••••••••••••
0100        31 00 10       LXI SP,1000H          ;Set up Stack Pointer just in case
0103        ED 5E          IM2                   ;Set Z-80 for Interrupt Mode 2
0105        3E 02          MVI A,02              ;Put 02 into Reg. A
0107        ED 47          MOV I,A               ;Set I Reg. to 02
0109        21 40 01       LXI H,0140            ;Set H pointer to beginning of table
010C        06 08          MVI B,8               ;Set B to length of data table
10E         0E B1          MVI C,0B1H            ;Set C to port Address for Channel A
0110        ED B3          OTIR                  ;Do a block Output to port B1H
0112        06 02          MVI B,02              ;Set B to length of 2
0114        0E B3          MVI C,0B3H            ;Set C to port Address for Channel B
0116        ED B3          OTIR                  ;Do a block Output to Port B3H
0118        06 02          MVI B,2               ;Set length to 2 again
011A        0E B8          MVI C,0B8H            ;Set C to block Address for CTC(0)
011C        ED B3          OTIR                  ;Do a block Output to CTC(0)
```

```
                    ••••••••••••••••••••••••••••••••••••••••
                     The SIO and CTC are now programmed
                    ••••••••••••••••••••••••••••••••••••••••
011E      FB        LOOP:EI                  ;Enable Interrupts
011F      76             HLT                 ;Halt until Rec. gets a character
0120      C3 1E 01       JMP LOOP            ;After Char. Echoed, jump back
                  ••••••••••••••••••••••••••••••••••••••••••••
                  Interrupt Service Routine Note ORG Statement
                  ••••••••••••••••••••••••••••••••••••••••••••
                         ORG 0280H           ;I Reg + Vector in WR2
0280      00             DB 00               ;Address of Routine is 0300
0281      03             DB03

                         ORG 0300H           ;Finally got there
0300      DB B0          IN 0B0H             ;Get Character that was typed
0302      D3 B0          OUT 0B0H            ;Echo it back
0304      ED 4D          RETI                ;Reset Int. Pending Status and Return
```

SECTION 7.3 Program to use the CTC as a Time Slice Divider

The next program is a simple test that determines if the RTC wire has been installed properly. It also indicates the procedure for programming the CTC(3) channel for interrupts.

Install the RTC wire as described in Section 1.9. Set Dip Switches 1,3,4 and 8 to the on or closed position (Base address equals B0H and Interrupts Enabled). Set the jumper in area C in the M1 column and the jumper in area G to the Z-80 column. Again Z-80 mode 2 is used to process the Interrupt.

This program will simply output a letter A to a specified port. In this example the output goes to port 02 (Super Serial Port A) but the user can choose whatever port he normally uses for the console. Since the CTC(3) Time Constant is set to 1, a letter will appear every 1/60 sec.

```
                  ••••••••••••••••••••••••••••••••••••••••••••
                   Set up CTC Channel 3 Interrupts every 1/60 sec.
                  ••••••••••••••••••••••••••••••••••••••••••••
                        ORG 0100H
          31 00 10      LXI SP,1000H         ;Set up stack pointer just in case
                                             ;the cops are coming
0103      ED 5E         IM2                  ;Set Z-80 to Interrupt Mode 2
0105      3E 02         MVI A,2              ;Set A Reg. to 02
0107      ED 47         MOV I,A              ;I Reg. set to 02
0109      AF            XRA                  ;Get a 00 in A
010A      D3 B8         OUT 0B8H             ;Set Interrupt Vector in CTC to 00
010C      3E D5         MVIA,0D5             ;CTC Channel 3 Interrupts Enabled.
010E      D3 BB         OUT 0BBH             ;Counting Mode, TC is next byte
0110      3E 01         MVI A,1              ;This value equals the number of
0112      D3 BB         OUT 0BBH             ;ticks that go by between interrupts
0114      FB       LOOP:EI                   ;Enable Interrupts
0115      76            HLT                  ;RTC will give us an Interrupt
0116      C3 14 01      JMP LOOP             ;After we send an "A", jmp to LOOP
                    ••••••••••••••••••••••••••••••••
                    Interrupt Routine Note ORG Value
                    ••••••••••••••••••••••••••••••••
                      ORG 0206H              ;I Reg. + CTC Int. Vector + Channel
                                             ;offset (00000110=Int Vector)
0206      00          DB 00                  ;This is address of Int. Routine
0207      03          DB 03                  ;which is 0300H

                      ORG 0300H              ;Finally got there
0300      3E 41       MVI A,'A               ;Get ASCII "A" in A Reg.
0302      D3 02       OUT 02                 ;Output "A" to your favorite port
0304      ED 4D       RETI                   ;Reset Int. Pending on CTC and Ret.
```

SECTION 8.0 Theory of Operation

The Board can be sub-divided into several simple modules. These are port decoding, data buss interface, status decoding, special function logic and RS-232 interface circuits.

SECTION 8.1 Port Decoding

The ZSIO Board occupies a total of (Ax14)-1 I/O mapped ports, where A is the number of boards in the system. Port FEH is common to all ZSIO boards in the system. The high order nybble of the Address buss during an I/O operation is decoded by the first four dip switch positions and an exclusive-or chip.

A closed switch position places a logic 0 at the input of one of the four gates and an open position places a logic 1 at the input. When the opposite state is placed at the other input of the gate, the output of that gate will attempt to set U24, Pin 9 to a logic 1. If any of the four outputs of the exclusive-or circuit are forced to a logic 0, U24 Pin 9 will be at a low. This in turn, forces U17 Pin 15 to a high, disabling the two-to-four decoder.

The decoder will be enabled only if the correct address bits, A7 through A4, are on the buss. Address bits A3 and A2 are two inputs to the decoder and determine one of the four port blocks.

If both A3 and A2 are at logic 0, U17 Pin 12 will go low and be the CE signal to SIO chip one.

If A3 is a low and A2 is a high, then U17 Pin 11 goes low and is the CE of SIO chip two.

If A3 is high and A2 is low, U17 Pin 10 goes low and is the CE of the CTC chip

If both A3 and A2 are high, the last four port block is enabled with U17 Pin 9. This signal is ANDed with A1 and used to enable the second two to four decoder at U17 Pin 1. Only two of the four ports in this block are actually used to enable the buss transceivers U36 and U35.

The FEH port is decoded by U34 and U24 Pin 6. The output of the eight input NAND gate is fed to U18 Pin 3. U18 Pin 6 is the board enable during I/O and is true when port addresses X0 through XBH, addresses XEH and XFH, and address FEH are TRUE. (X = Board base address).

SECTION 8.2 Data Buss Interface

The ZSIO board interfaces to the 696 Standard Buss through U35 and U36 Tri-Directional Buss Transceivers. The four possible buss transfers that the ZSIO internal buss has with the external IEEE 696 Standard Buss are the following:

1) I/O Read
2) I/O Write
3) Interrupt Acknowledge
4) M1 Read

Both I/O Read and INTA transfers are from the ZSIO Board's internal buss to the 696 Standard Input Buss. The I/O Write transfer is from the 696 Standard output buss to the ZSIO internal buss. The M1 read transfer is from the 696 Input Buss to the ZSIO internal buss.

Complete flexibility in busss transfers is achieved by the six control inputs of the 74LS442 busss transceivers.

The chip enables of the transceivers. U35 and U36 Pin 1 are activated when U18 Pin 6 (board I/O enable) is true. These transceivers are also enabled during every M1 cycle (jumper area C in the M1 position) or whenever the processor is in an INTA cycle (jumper area C in the INTA position). Once the transceivers are enabled. then S0, S1, GA, GB and GC must be in the correct state to accomplish the appropriate busss transfer. The following table illustrates how the six control signals of the buss transceivers are used to control data flow:

Buss Transfer	CE	GA	GB	GC	SO	S1
I/O Read	L	H	H	L	L	L
I/O Write	L	H	L	L	L	H
INTA	L	H	H	L	L	L
M1 Read	L	H	H	L	H	L

From this table the following functions are derived:

CE = X0 thru XB + XE + XF + FE + M1
GA = logic 1 always
GB = SOUT
GC = logic 0 always
SO = SOUT + (IEI•IEO•INTA'•DBIN) + (DBIN•SINP)
S1 = (DBIN•SINP) + (IEI•IEO•INTA'•DBIN) + (M1•INTA *)
Where INTA' is INTA delayed by wait states if inserted and an asterisk is used to denote a low true condition.

When Jumper Area C is in the INTA position, CE and S1 are not true during M1 cycles and are true during INTA.

Jumper area C is used to prevent the ZSIO support circuits from reading the IEEE Standard Data Input Buss when the Interrupt Priority Logic of the SIO and CTC is to be reset by writing out to Port FEH and EDH followed by a 4DH.

After the SIO and CTC detect an EDH during M1, the Interrupt-Priority Logic (IPL) will be reset only if the next M1 cycle has a 4DH on the buss. consider the following set of instructions used to reset the IPL.

```
0100        3E ED       MVI A,EDH
0102        D3 FE       OUT OFEH
0104        3E 4D       MVI A,4DH
0106        D3 FE       OUT OFEH
```

If jumper area C were left in the M1 position, the SIO and CTC would see ED during the first OUT instruction. However, the chips would see 3E during the next M1 cycle. Because the chips would not see ED followed by 4D in successive M1 cycles the chips would not see a RETI instruction on the buss and the IPL would not be reset. By placing jumper area C in the INTA position the SIO and CTC will have its M1 signal true only during INTA and the above program can reset the IPL properly.

SECTION 8.3 Status Decoding

Status Decoding can best be understood by again describing the four types of buss transfers to which the ZSIO will respond.

During an I/O Read, DBIN (buffered through U25) and SINP (buffered with a gate from U29) are ANDed together with U19. The output (U19, Pin 8) is fed to the buss transceivers and also to IORQ and RD of the SIO and CTC chips. When the proper port address is on the buss, the CE signal of the SIO and CTC chips will be low and data from the specific chip will be transferred to the IEEE 696 Data Input Bus.

DBIN and SINP are also ANDed with U25 (Pins 4, 5 and 6) and fed to the B input of the two-to-four decoder. Since the other input to the decoder is a write strobe, it will be false and force the output of the decoder Y2 low. This is used as the buffer enable to U29 and DSR status for the four RS-232 channels is placed onto the Input Buss. This transfer will take place only if the G input to the decoder U17, Pin 1 is low. This will be the case only when the correct port is addressed.

During INTA, DBIN is ANDed with several signals. The first is INTA', which is INTA delayed by a preset number of wait states (determined by switch positions 5, 6 and 7). This delayed signal is U31 Pin 7 and is used to give the SIO and CTC chips sufficient time to compete for the Interrupt when more than one chip has an Interrupt pending.

When using more than one ZSIO board, several boards can attempt to place an Interrupt vector on the buss during INTA. For this reason, IEI and IEO are ANDed together with INTA. This signal, U18, Pin 12, is finally ANDed with DBIN at U19. This output is routed to the buss transceivers and to the IORQ Pin of the SIO and CTC chips. Thus, in a multiple board system, only the board with the highest priority can insert an Interrupt Vector on the buss.

When the processor writes data out to the ZSIO board, PWR and SOUT are ANDed together and the output, U20 Pin 13, is fed to U21 Pin 12. This flip flop is used to guarantee enough time between IORQ and the rising edge of the system clock. This setup time is only activated when jumper area G has the shunt in the 8080 column. If the shunt is in the Z-80 column, then U20 Pin 13 is connected directly to the IORQ logic.

The absence of the RD signal to the SIO and CTC chips forces the chips into an I/O write state. PWR ANDed with SOUT is also fed to the two-to-four decoder U17 Pin 2. Since the B input will be false during an I/O write, Y1 of the decoder will strobe data into the quad latch, U23. The outputs of this latch are fed to the four RS-232 channels as the Data Carrier Generate signals.

The M1 read state is only enabled when jumper area C has a shunt in the M1 column. When so jumpered, the SIO and CTC chips are able to read data off the Data Input Buss during instruction fetches. This function is activated through U37 Pin 2 and is fed to the M1 input of the SIO and to the CTC chips through U28 Pin 8.

During the M1 fetch cycle the chips can look for an RETI instruction. This instruction is used to reset the Interrupt Priority Logic of the chips. If the shunt in jumper area C is in the INTA column, the M1 signal to the SIO and CTC chips will only be TRUE during INTA. This is a requirement for enabling Interrupt Vectors to be read from the Zilog chips.

SECTION 8.4 Special Function Logic

Most of the special function logic of the ZSIO Board was added to the design in order to enable any number of boards to be daisy-chained together. As mentioned in previous sections, problems can arise when attempting to use more than four Zilog support circuits in an interrupt environment. See Appendix A, The SIO Technical Manual, page 41 and 42 for details.

To extend the amount of time that the SIO and CTC chips can have to compete for an Interrupt Vector response, U31, and its associated logic, places the processor into a wait state. The length of this period is determined by switches 5, 6 and 7 (see Section 1.8).

If all three switches are open, then U31 is loaded with a binary 7 at the time the INTA cycle begins. The first rising edge of the system clock increments this counter to 8 and the INTA circuitry forces IORQ low without any time delay. If any of the switches are closed, a binary count less than 7 will be loaded at the beginning of INTA and the processor will be forced into a wait state until the count of U31 reaches 10.

The first several clock cycles during an INTA are not used by the processor for examining the wait line. Because of this, the number of wait states is equal to the number loaded into U31 (at the beginning of the INTA cycle) subtracted from 7. A load of seven results in regular INTA timing. The wait line is also forced low when the processor is executing an Input or Output cycle to a ZSIO Board. This is due to the setup of the IORQ to rising edge of the system clock.

Finally, the ZSIO board will put the processor into a wait state when the wait function of the ZSIO chip is enabled. This can prove very useful when doing block I/O. **In Summary, the wait line is forced low during the extended INTA cycles, during Input and Output instructions to the ZSIO Board, and if the Wait function of the SIO is enabled.** Since these added wait cycles are added only during the above special conditions, total processor throughput is not degregated.

Sufficient IORQ-to-rising-edge setup time is assured by a set of flip-flops, 1/2 of U21 and 1/2 of U30. SOUT and PWR are ANDed together by U20 Pin 13 and are then fed to the D input of U21.

The rising edge of the system clock transfers this signal to the IORQ circuitry and the setup time is set at 250nsec. when running at 4MHz. The rising edge of PWR then sets U30 Pin 5 low which in turn resets U21 Pin 9 to a low state. M1 during the next instruction fetch sets U30 Pin 5 back to a high state and prepares U21 for the next ZSIO write cycle. Note that this special timing is only needed during Write cycles to the ZSIO Board. This is due to the fact that Input timing does not create setup time problems. If this timing is not needed (as is the case with Z-80 processors), jumper area G provides a means whereby SOUT and PWR ANDed by U20 Pin 13 can be fed directly to the IORQ logic.

The 18.432MHz crystal is divided by U30 and U38 by a factor of 20. The primary reason for this is to provide a frequency that can easily be sub-divided into common baud rates. The second reason for the divide by 20 logic is so that the frequency going into the CTC is less than 1MHz. This is desirable because of the fact that the maximum rate at which the CTC can be clocked is 1/2 the system clock rate (in many cases 2MHz).

SECTION 8.5 RS-232 Interface Circuitry

The RS-232 Circuitry is basically four indentical sets of line drivers and receivers organized in such a way as to allow ease of configuration. Of primary importance in its design was enabling the user the ability to create any desired RS-232 configuration with a solderless Modem Signal Direction Area. Each line driver and receiver can be jumpered in one or two ways. In the case of CTS, this jumpering can be performed in one of three ways.

Since the ZSIO does not have a DSR input, this signal was routed to a 74LS367 and can be polled through Port XEH or XFH.

When interfacing to several peripherals and modems it is desirable to generate a carrier for the RS-232 interface. It was for this reason that U23 (a four bit register) was incorporated in the board. By writing out to this port, DCD can be software controlled.

Figures and Appendices

Figures A1 - A4; Modem Signal Direction Header

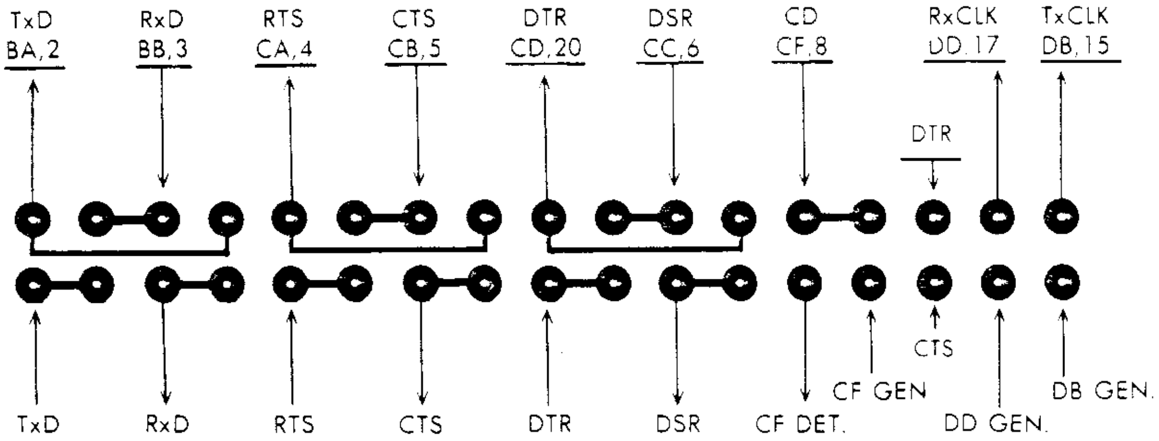


Figure A1 - Hardware configuration for one of four modem signal direction headers shown without any jumpers. Note that each of the SIO signals can be connected to one of two places with the exception of CTS which can be connected to one of three places.

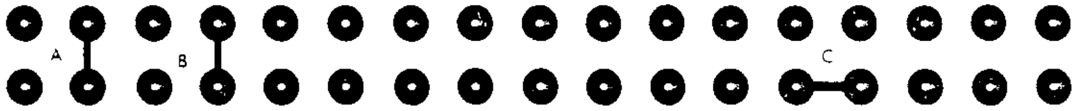


Figure A2 - Modem signal direction header showing the three jumpers **etched into the P.C. board** (A,B, and C). Jumpers A and B are all that are needed when communicating to terminals not requiring any handshaking signals. Jumper C is used to enable the receiver of the channel when the **Auto Enable** function is used. See text for details on how to use Auto Enables.

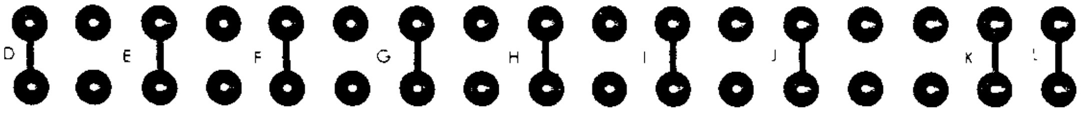


Figure A3 - In this configuration jumpers A,B and C have been removed, jumpers D through L have been added, and the ZSIO channel is acting like a terminal talking to a modem. The channel is excepting CTS (G), DSR (I), and CD (J) from the modem while generating RTS (F), DTR (H), Receiver Clock (K), and Transmitter Clock (L). Usually when the ZSIO is operating in Synchronous mode the modem will supply the two clock signals, therefore jumpers K and L would not be inserted. See figure B3.

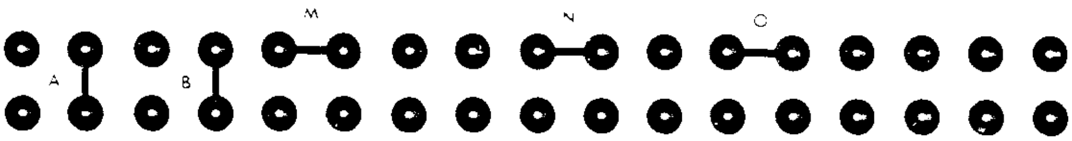


Figure A4 - This configuration can be used to make a terminal act as though it is connected to a modem when handshaking signals are needed such as Diablo or Necwriter printers. The printer's RTS is looped back to its CTS (M) and the printer's DTR is looped to both DSR (N) and CD (O).

Figures B1 - B4; Receiver and Transmitter Clock Header

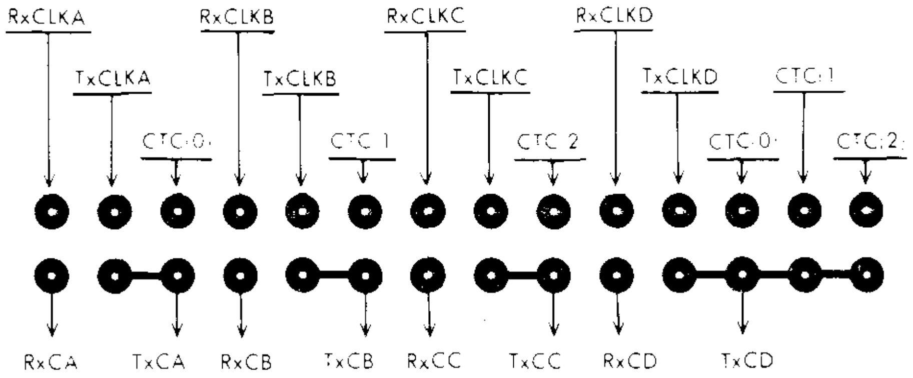


Figure B1 - Hardware configuration for the Receiver and Transmitter Clock Header. Each channel can be jumpered such that the receiver and transmitter clocks are generated from the RS-232 interface or a CTC channel.

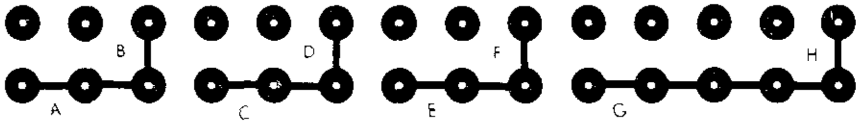


Figure B2 - This figure shows the jumpers which are **etched into the P.C. board.** Jumpers A,C,E, and G are connecting the receiver and transmitter clocks together for channels A,B,C,and D respectively. Jumper B connects CTC channel 0 to serial channel A, jumper D connects CTC channel 1 to serial channel B, and jumpers F and H connect CTC channel 2 to serial channels C **and** D. These jumpers can be cut with a sharp knife when necessary.

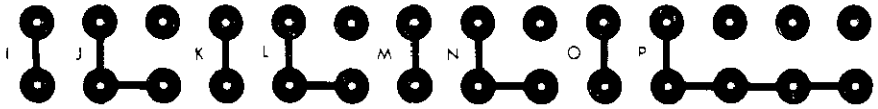


Figure B3 - In this configuration jumpers A through H have been removed and jumpers I through P have been installed. All four channels are being clocked from the RS-232 interface and the CTC channels are not needed to generate baud rates. A channel would normally be connected as shown when it is communicating with a **Synchronous** modem.

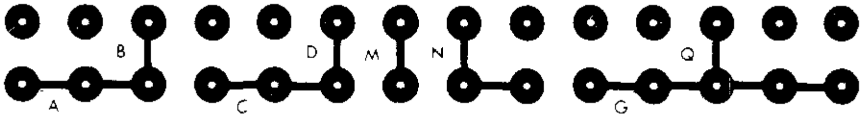


Figure B4 - In this configuration channel A and D clocks are generated from CTC channel 0, channel B clocks are generated from CTC channel 1, and channel C clocks are generated from the RS-232 interface.

Figures C1 - C3; Interrupt Priority Jumper Area

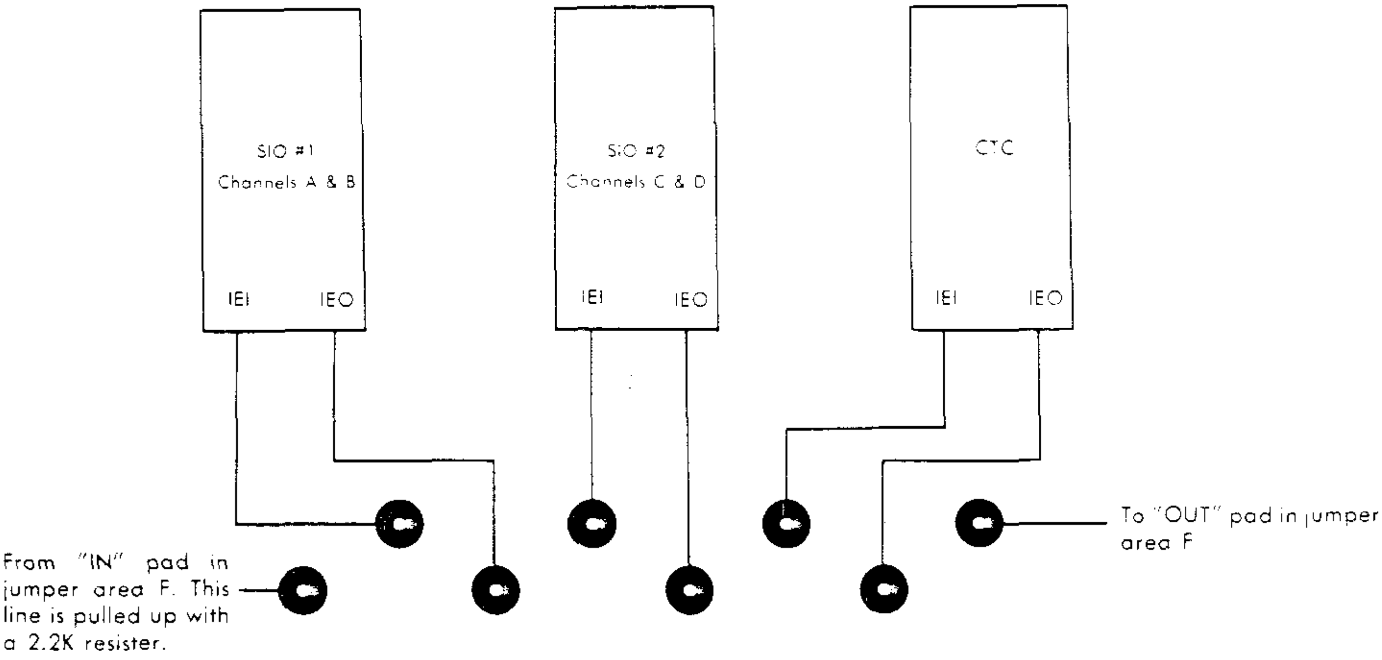

SIO #1
Channels A & B
IEI
IEO
SIO #2
Channels C & D
IEI
IEO
CTC
IEI
IEO
To "OUT" pad in jumper area F
From "IN" pad in jumper area F. This line is pulled up with a 2.2K resister.

Figure C1 - Interrupt Priority jumper area shown without any jumpers.

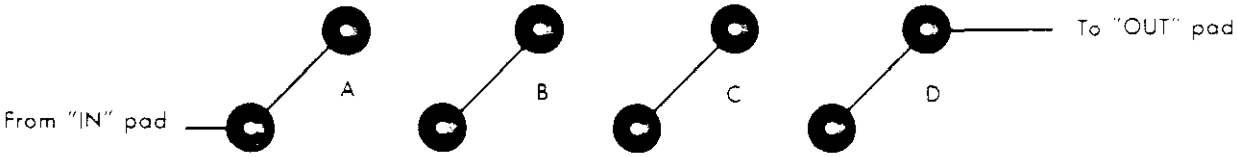


Figure C2 - Interrupt Priority area showing the four jumpers **etched into the P.C. board**. This configuration allows SIO #1 to have the highest priority in the daisy chain. SIO #2 is next and the CTC has the lowest priority.

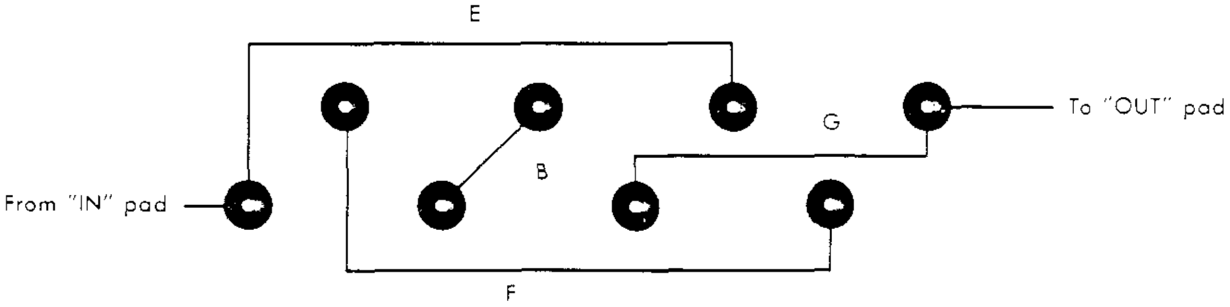


Figure C3 - In this example jumpers A,C and D have been cut away with a sharp knife and jumpers E,F and G have been installed. The CTC now has highest priority in the daisy chain followed by SIO #1 then SIO #2.

Figures D1 and D2 - Multiple Board Daisy Chain Jumper Area

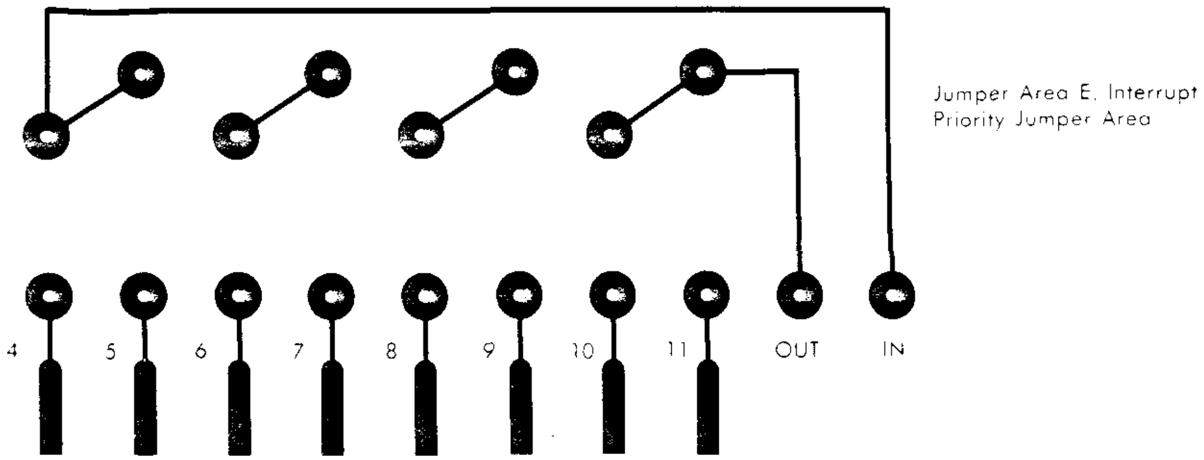


Figure D1 - Jumper area F shown without any jumpers. The **on board** daisy chain jumper area E is shown in this figure to clarify the relationship between the two areas.

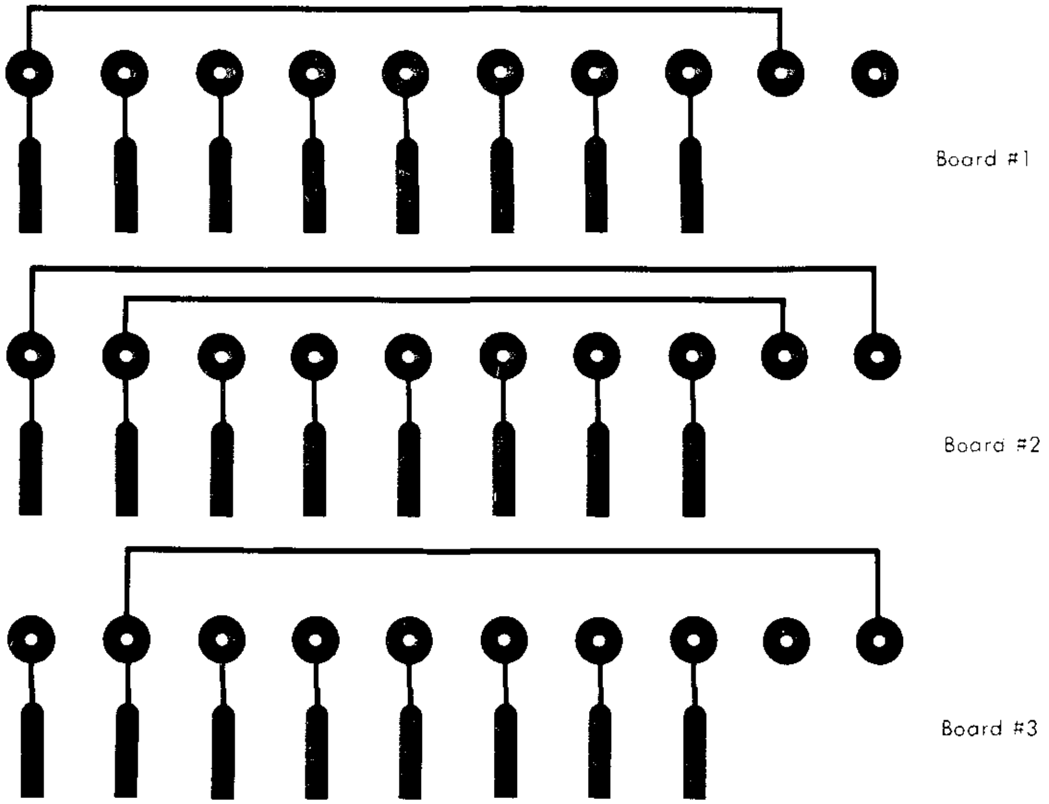


Figure D2 - This example shows how to daisy chain three ZSIO boards in a system. Jumper Area F is configured such that the daisy chain communication is transmitted through S-100 buss lines 4 through 11 which are the vector interrupt lines. If these lines are being used in a given system, then any free lines can be used for this purpose.

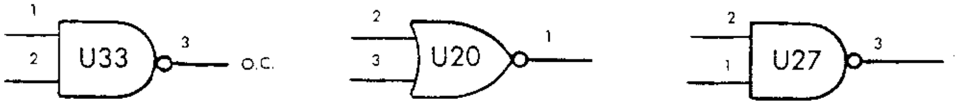
Board #1 would have nothing connected to its "IN" pad, which is pulled up with a 2.2K resister. This board would have the highest priority in the three board daisy chain. The "OUT" pad of this first board would be connected to a free line on the S-100 buss, in this example pin 4 is used. The second board would take pin 4 and connect it to its "IN" pad and the "OUT" pad would be connected to the next free line, in this example pin 5 is used. Finally, board #3 would have pin 5 connected to its "IN" pad which completes the daisy chain.

It should be noted that if other hardware in the system is generating interrupts such as disk controllers, then the interrupt from this board should be incorporated in the daisy chain if there is a possibility that they can generate an interrupt response vector at the same time. In this way more than one vector could not be placed on the data buss during interrupt acknowledge.

ZSIO PARTS LIST

U1	75189	U23	74LS175	VR1	7805
U2	75189	U24	75LS04	VR2	7805
U3	75188	U25	75LS08	VR3	7812
U4	75189	U26	74LS04	VR4	7912
U5	75188	U27	74LS00	RP1	1k Res. Pack
U6	75188	U28	74LS08	R1	2.2k
U7	75189	U29	74LS367	R2	18k
U8	74LS367	U30	74LS74	R3	1k
U9	75188	U31	74LS191	2	Heat Sinks
U10	75189	U32	74LS136	4	2x8 Right Angle Headers
U11	75188	U34	74LS30	4	2x17 Straight Headers
U12	75188	U34	74LS30	1	2x14 Straight Header
U13	75189	U35	74LS442	2	2x2 Straight Headers
U14	Z80A-SIO/2	U36	74LS442	1	RTC Terminal
U15	Z80A-CTC	U37	74LS14	1	RTC Cap
U16	Z80A-SIO/2	U38	7493	1	RTC Wire
U17	74LS139		18.432 Mhz Crystal	3	Metal Bolts
U18	74LS11		8 Position Dip Switch	1	Nylon Bolt
U19	74LS51		C1 thru C5 - 16uf, 35V	4	Washers
U20	74LS02		C6 thru C23 - .1uf	4	Nuts
U21	74LS74		CR1 - 1N914	4	Cable Assm.
U22	74LS02		Q1 - 2N3904	4	Screw Lock Assm.

UNUSED HARDWARE



USER NOTES

Appendix A

Z80® SIO

Technical Manual

Z80-SIO Technical Manual

Contents

General Information

The Z80-SIO (Serial Input/Output) is a dual-channel multi-function peripheral component designed to satisfy a wide variety of serial data communications requirements in microcomputer systems. Its basic function is a serial-to-parallel, parallel-to-serial converter/controller, but—within that role—it is configurable by systems software so its "personality" can be optimized for a given serial data communications application.

The Z80-SIO is capable of handling asynchronous and synchronous byte-oriented protocols such as IBM Bisync, and synchronous bit-oriented protocols such as HDLC and IBM SDLC. This versatile device can also be used to support virtually any other serial protocol for applications other than data communications (cassette or floppy disk interfaces, for example).

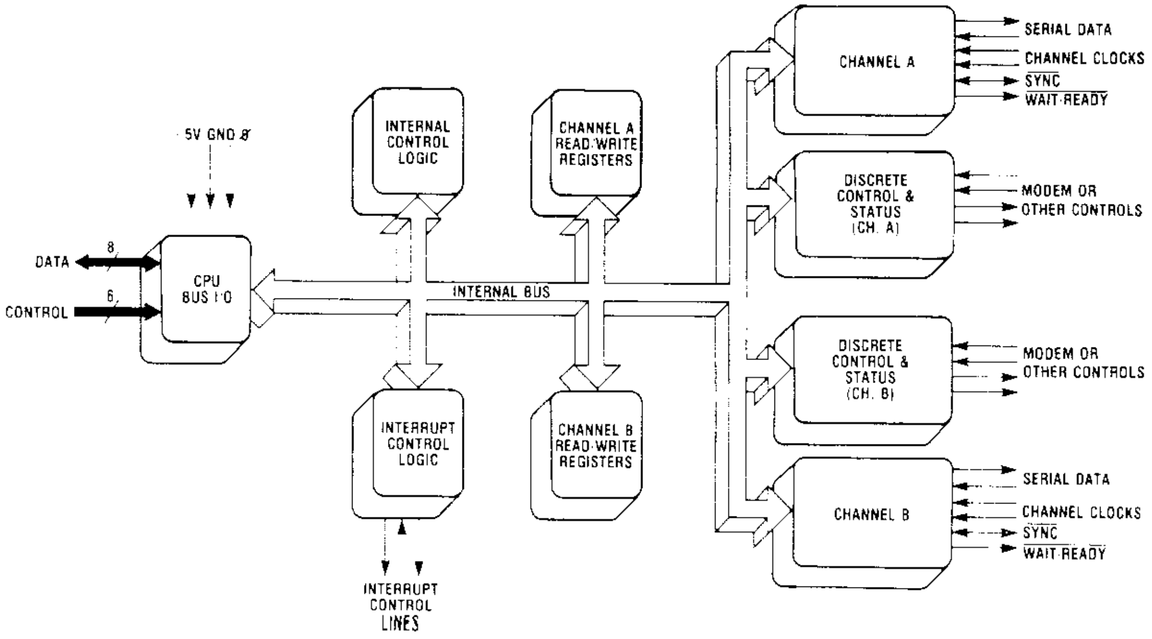
The Z80-SIO can generate and check CRC codes in any synchronous mode and can be programmed to check data integrity in various modes. The device also has facilities for modem controls in both channels. In applications where these controls are not needed, the modem controls can be used for general-purpose I/O.

STRUCTURE

- N-channel silicon-gate depletion-load technology
- 40-pin DIP
- Single 5 V power supply
- Single-phase 5 V clock
- All inputs and outputs TTL compatible

FEATURES

- Two independent full-duplex channels
- Data rates in synchronous or isosynchronous modes:
  - 0–550K bits/second with 2.5 MHz system clock rate
  - 0–880K bits/second with 4.0 MHz system clock rate
- Receiver data registers quadruply buffered; transmitter doubly buffered.
- Asynchronous features:
  - 5, 6, 7 or 8 bits/character
  - 1, 1½ or 2 stop bits
  - Even, odd or no parity
  - ×1, ×16, ×32 and ×64 clock modes
  - Break generation and detection
  - Parity, overrun and framing error detection



Z80-SIO BLOCK DIAGRAM

- Binary synchronous features:
  - Internal or external character synchronization
  - One or two sync characters in separate registers
  - Automatic sync character insertion
  - CRC generation and checking
- HDLC and IBM SDLC features:
  - Abort sequence generation and detection
  - Automatic zero insertion and deletion
  - Automatic flag insertion between messages
  - Address field recognition
  - I-field residue handling
  - Valid receive messages protected from overrun
  - CRC generation and checking
- Separate modem control inputs and outputs for both channels
- CRC-16 or CRC-CCITT block check
- Daisy-chain priority interrupt logic provides automatic interrupt vectoring without external logic
- Modem status can be monitored

Pin Description

D_0-D_7. *System Data Bus* (bidirectional, 3-state). The system data bus transfers data and commands between the CPU and the Z80-SIO. D_0 is the least significant bit.

$B/\overline{A}$. *Channel A Or B Select* (input, High selects Channel B). This input defines which channel is accessed during a data transfer between the CPU and the Z80-SIO. Address bit A_0 from the CPU is often used for the selection function.

$C/\overline{D}$. *Control Or Data Select* (input, High selects Control). This input defines the type of information transfer performed between the CPU and the Z80-SIO. A High at this input during a CPU write to the Z80-SIO causes the information on the data bus to be interpreted as a command for the channel selected by $B/\overline{A}$. A Low at $C/\overline{D}$ means that the information on the data bus is data. Address bit A_1 is often used for this function.

$\overline{CE}$. *Chip Enable* (input, active Low). A Low level at this input enables the Z80-SIO to accept command or data inputs from the CPU during a write cycle, or to transmit data to the CPU during a read cycle.

ϕ. *System Clock* (input). The Z80-SIO uses the standard Z80A System Clock to synchronize internal signals. This is a single-phase clock.

$\overline{M1}$. *Machine Cycle One* (input from Z80-CPU, active Low). When $\overline{M1}$ is active and $\overline{RD}$ is also active, the Z80-CPU is fetching an instruction from memory; when $\overline{M1}$ is active while $\overline{IORQ}$ is active, the Z80-SIO accepts $\overline{M1}$ and $\overline{IORQ}$ as an interrupt acknowledge if the Z80-SIO is the highest priority device that has interrupted the Z80-CPU.

$\overline{IORQ}$. *Input/Output Request* (input from CPU, active Low). $\overline{IORQ}$ is used in conjunction with $B/\overline{A}$, $C/\overline{D}$, $\overline{CE}$ and $\overline{RD}$ to transfer commands and data between the CPU and the Z80-SIO. When $\overline{CE}$, $\overline{RD}$ and $\overline{IORQ}$ are all active,

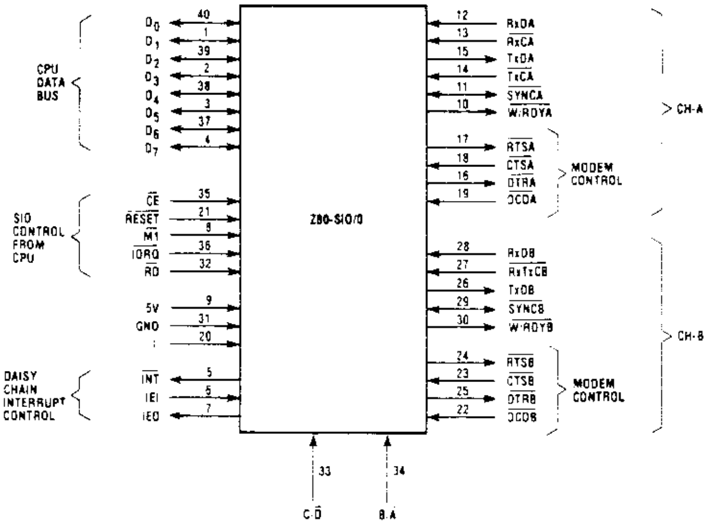


Figure 1. Z80-SIO/0 Pin Configuration

the channel selected by B/A transfers data to the CPU (a read operation). When CE and IORQ are active, but RD is inactive, the channel selected by B/A is written to by the CPU with either data or control information as specified by C/D. As mentioned previously, if IORQ and M1 are active simultaneously, the CPU is acknowledging an interrupt and the Z80-SIO automatically places its interrupt vector on the CPU data bus if it is the highest priority device requesting an interrupt.

RD. *Read Cycle Status.* (input from CPU, active Low). If RD is active, a memory or I/O read operation is in progress. RD is used with B/A, CE and IORQ to transfer data from the Z80-SIO to the CPU.

RESET. *Reset* (input, active Low). A Low RESET disables both receivers and transmitters, forces TxDA and TxDB marking, forces the modem controls High and disables all interrupts. The control registers must be rewritten after the Z80-SIO is reset and before data is transmitted or received.

IEI. *Interrupt Enable In* (input, active High). This signal is used with IEO to form a priority daisy chain when there is more than one interrupt-driven device. A High on this line indicates that no other device of higher priority is being serviced by a CPU interrupt service routine.

IEO. *Interrupt Enable Out* (output, active High). IEO is High only if IEI is High and the CPU is not servicing an interrupt from this Z80-SIO. Thus, this signal blocks lower priority devices from interrupting while a higher priority device is being serviced by its CPU interrupt service routine.

INT. *Interrupt Request* (output, open drain, active Low). When the Z80-SIO is requesting an interrupt, it pulls INT Low.

W/RDYA, W/RDYB. *Wait/Ready A, Wait/Ready B* (outputs, open drain when programmed for Wait function, driven High and Low when programmed for Ready function). These dual-purpose outputs may be programmed as Ready lines for a DMA controller or as Wait lines that synchronize the CPU to the Z80-SIO data rate. The reset state is open drain.

CTSA, CTSB. *Clear To Send* (inputs, active Low). When programmed as Auto Enables, a Low on these inputs enables the respective transmitter. If not programmed as Auto Enables, these inputs may be programmed as general-purpose inputs. Both inputs are Schmitt-trigger buffered to accommodate slow-risetime inputs. The Z80-SIO detects pulses on these inputs and interrupts the CPU on both logic level transitions. The Schmitt-trigger inputs do not guarantee a specified noise-level margin.

DCDA, DCDB. *Data Carrier Detect* (inputs, active Low). These signals are similar to the CTS inputs, except they can be used as receiver enables.

RxDA, RxDB. *Receive Data* (inputs, active High).

TxDA, TxDB. *Transmit Data* (outputs, active High).

RxCA, RxCB.* *Receiver Clocks* (inputs). See the following section on bonding options. The Receive Clocks may be 1, 16, 32 or 64 times the data rate in asynchronous modes. Receive data is sampled on the rising edge of RxC.

*See footnote on next page.

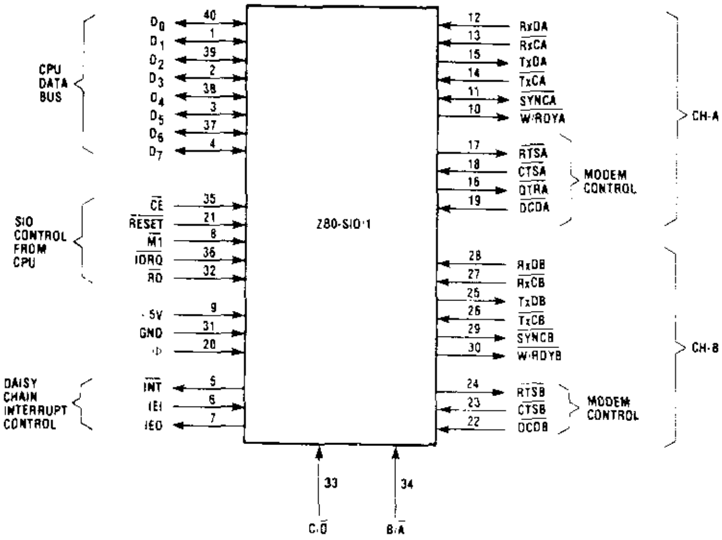


Figure 2. Z80-SIO/1 Pin Configuration

$\overline{\text{TxCA}}$, $\overline{\text{TxCB}}$.* *Transmitter Clocks* (inputs). See section on bonding options. In asynchronous modes, the Transmitter clocks may be 1, 16, 32 or 64 times the data rate. The multiplier for the transmitter and the receiver must be the same. Both the $\overline{\text{TxC}}$ and $\overline{\text{RxC}}$ inputs are Schmitt-trigger buffered for relaxed rise- and fall-time requirements (no noise margin is specified). TxD changes on the falling edge of $\overline{\text{TxC}}$.

$\overline{\text{RTSA}}$, $\overline{\text{RTSB}}$. *Request To Send* (outputs, active Low). When the RTS bit is set, the $\overline{\text{RTS}}$ output goes Low. When the RTS bit is reset in the Asynchronous mode, the output goes High after the transmitter is empty. In Synchronous modes, the $\overline{\text{RTS}}$ pin strictly follows the state of the RTS bit. Both pins can be used as general-purpose outputs.

$\overline{\text{DTRA}}$, $\overline{\text{DTRB}}$. *Data Terminal Ready* (outputs, active Low). See note on bonding options. These outputs follow the state programmed into the DTR bit. They can also be programmed as general-purpose outputs.

$\overline{\text{SYNC A}}$, $\overline{\text{SYNC B}}$. *Synchronization* (inputs/outputs, active Low). These pins can act either as inputs or outputs. In the Asynchronous Receive mode, they are inputs similar to $\overline{\text{CTS}}$ and $\overline{\text{DCD}}$. In this mode, the transitions on these lines affect the state of the Sync/Hunt status bits in RR0. In the External Sync mode, these lines also act as inputs. When external synchronization is achieved, $\overline{\text{SYNC}}$ must be driven Low on the second rising edge of $\overline{\text{RxC}}$ after that rising edge of $\overline{\text{RxC}}$ on which the last bit of the sync character was received. In other words, after the sync pattern is detected, the external logic must wait for two full Receive Clock cycles to activate the $\overline{\text{SYNC}}$ input. Once $\overline{\text{SYNC}}$ is forced Low, it is wise to keep it Low until the CPU informs the external sync logic that synchronization has been lost or a new message is about to start. Character assembly begins on the rising edge of $\overline{\text{RxC}}$ that immediately precedes the falling edge of $\overline{\text{SYNC}}$ in the External Sync mode.

In the Internal Synchronization mode (Monosync and Bisync), these pins act as outputs that are active during the part of the receive clock ($\overline{\text{RxC}}$) cycle in which sync characters are recognized. The sync condition is not latched, so these outputs are active each time a sync pattern is recognized, regardless of character boundaries.

BONDING OPTIONS

The constraints of a 40-pin package make it impossible to bring out the Receive Clock, Transmit Clock, Data Terminal Ready and Sync signals for both channels. Therefore, Channel B must sacrifice a signal or have two signals bonded together. Since user requirements vary, three bondings options are offered:

- Z80-SIO/0 has all four signals, but $\overline{\text{TxCB}}$ and $\overline{\text{RxCB}}$ are bonded together (Fig. 1).
- Z80-SIO/1 sacrifices $\overline{\text{DTRB}}$ and keeps $\overline{\text{TxCB}}$, $\overline{\text{RxCB}}$ and $\overline{\text{SYNCB}}$ (Fig. 2).
- Z80-SIO/2 sacrifices $\overline{\text{SYNCB}}$ and keeps $\overline{\text{TxCB}}$, $\overline{\text{RxCB}}$ and $\overline{\text{DTRB}}$ (Fig. 3).

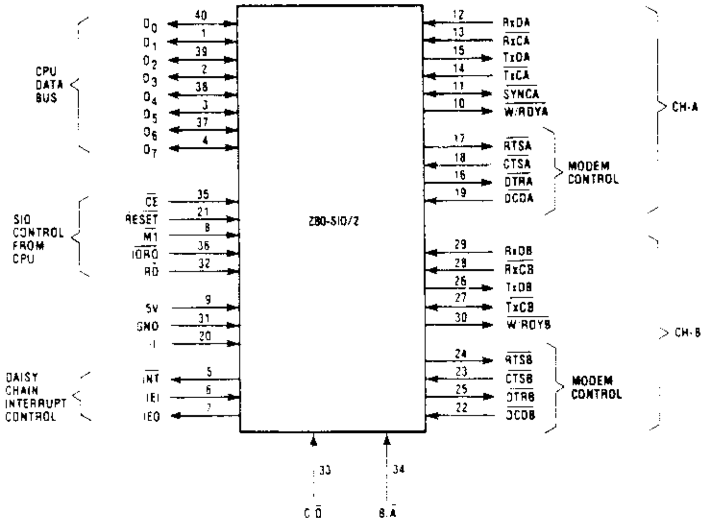


Figure 3. Z80-SIO/2 Pin Configuration

*These clocks may be directly driven by the Z80-CTC (Counter Timer Circuit) for fully programmable baud rate generation.

Architecture

The device internal structure includes a Z80-CPU interface, internal control and interrupt logic, and two full-duplex channels. Associated with each channel are read and write registers, and discrete control and status logic that provides the interface to modems or other external devices.

The read and write register group includes five 8-bit control registers, two sync-character registers and two status registers. The interrupt vector is written into an additional 8-bit register (Write Register 2) in Channel B that may be read through Read Register 2 in Channel B. The registers for both channels are designated in the text as follows:

WR0-WR7 — Write Registers 0 through 7
RR0-RR2 — Read Registers 0 through 2

The bit assignment and functional grouping of each register is configured to simplify and organize the programming process. Table 1 illustrates the functions assigned to each read or write register.

WR0	Register pointers, CRC initialize, initialization commands for the various modes, etc.
WR1	Transmit/Receive interrupt and data transfer mode definition.
WR2	Interrupt vector (Channel B only)
WR3	Receive parameters and controls
WR4	Transmit/Receive miscellaneous parameters and modes
WR5	Transmit parameters and controls
WR6	Sync character or SDLC address field
WR7	Sync character or SDLC flag

(a) Write Register Functions

RR0	Transmit/Receive buffer status, interrupt status and external status
RR1	Special Receive Condition status
RR2	Modified interrupt vector (Channel B only)

(b) Read Register Functions

Table 1. Functional Assignments of Read and Write Registers

The logic for both channels provides formats, synchronization and validation for data transferred to and from the channel interface. The modem control inputs Clear to Send ($\overline{\text{CTS}}$) and Data Carrier Detect ($\overline{\text{DCD}}$) are monitored by the discrete control logic under program control. All the modem control signals are general purpose in nature and can be used for functions other than modem control.

For automatic interrupt vectoring, the interrupt control logic determines which channel and which device within the channel has the highest priority. Priority is fixed with Channel A assigned a higher priority than Channel B; Receive, Transmit and External/ Status interrupts are prioritized in that order within each channel.

Data Path

The transmit and receive data path for each channel is shown in Figure 4. The receiver has three 8-bit buffer registers in a FIFO arrangement (to provide a 3-byte delay) in addition to the 8-bit receive shift register. This arrangement creates additional time for the CPU to service an interrupt at the beginning of a block of high-speed data. The receive error FIFO stores parity and framing errors and other types of status information for each of the three bytes in the receive data FIFO.

Incoming data is routed through one of several paths depending on the mode and character length. In the Asynchronous mode, serial data is entered in the 3-bit buffer if it has a character length of seven or eight bits, or is entered in the 8-bit receive shift register if it has a length of five or six bits.

In the Synchronous mode, however, the data path is determined by the phase of the receive process currently in operation. A Synchronous Receive operation begins with the receiver in the Hunt phase, during which the receiver searches the incoming data stream for a bit pattern that matches the preprogrammed sync characters (or flags in the SDLC mode). If the device is programmed for Monosync Hunt, a match is made with a single sync character stored in WR7. In Bisync Hunt, a match is made with dual sync characters stored in WR6 and WR7.

In either case the incoming data passes through the receive sync register, and is compared against the programmed sync character in WR6 or WR7. In the Monosync mode, a match between the sync character programmed into WR7 and the character assembled in the receive sync register establishes synchronization.

In the Bisync mode, however, incoming data is shifted to the receive shift register while the next eight bits of the message are assembled in the receive sync register. The match between the assembled character in the receive sync registers with the programmed sync character in WR6 and WR7 establishes synchronization. Once synchronization is established, incoming data by-

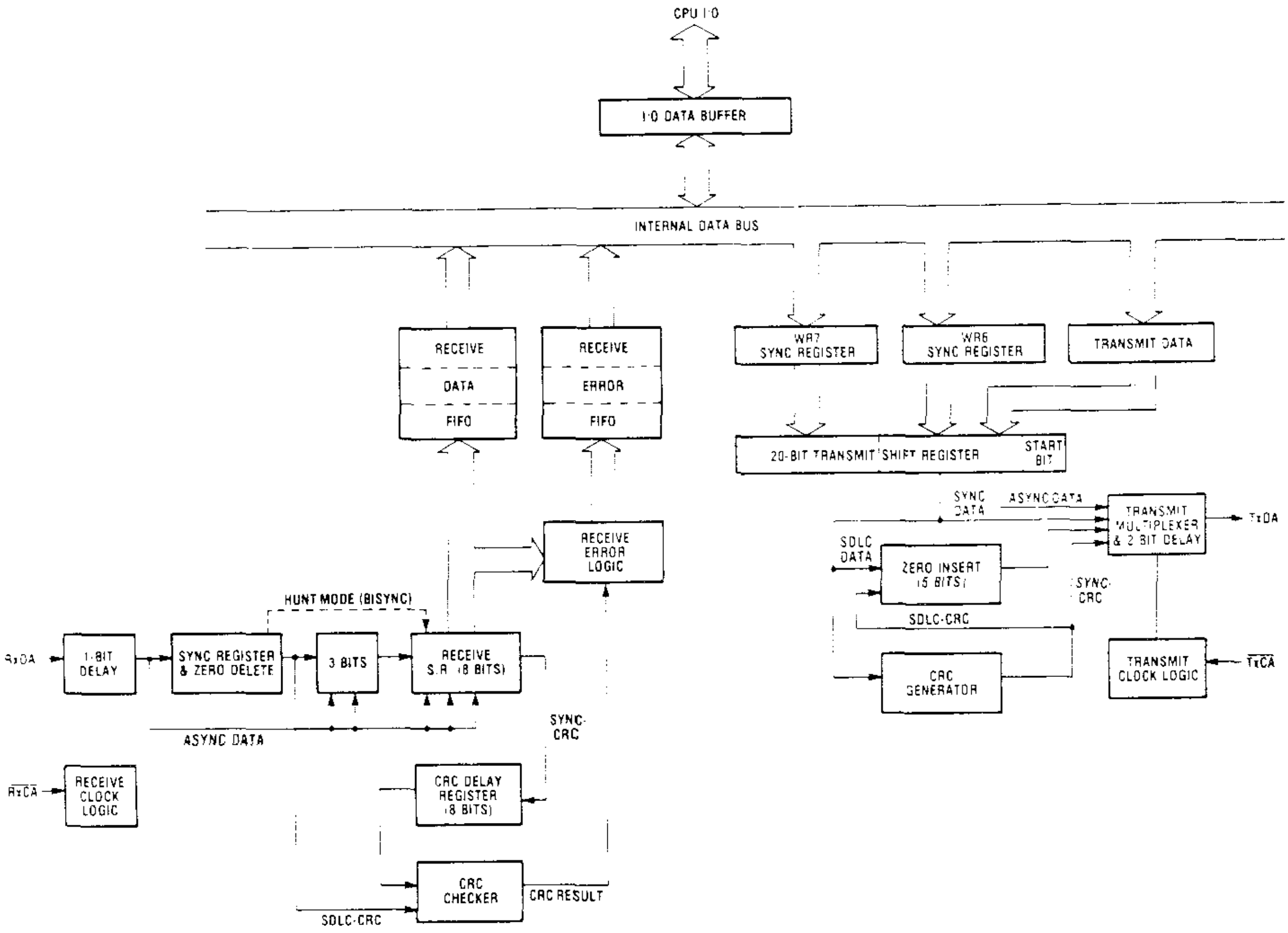


Figure 4. Transmit and Receive Data Path

passes the receive sync register and directly enters the 3-bit buffer.

In the SDLC mode, incoming data first passes through the receive sync register, which continuously monitors the receive data stream and performs zero deletion when indicated. Upon receiving five contiguous 1's, the sixth bit is inspected. If the sixth bit is a 0, it is deleted from the data stream. If the sixth bit is a 1, the seventh bit is inspected. If that bit is a 0, a Flag sequence has been received; if it is a 1, an Abort sequence has been received.

The reformatted data enters the 3-bit buffer and is transferred to the receive shift register. Note that the SDLC receive operation also begins in the Hunt phase, during which the Z80-SIO tries to match the assembled character in the receive shift register with the flag pattern in WR7. Once the first flag character is recognized, all subsequent data is routed through the same path, regardless of character length.

Although the same CRC checker is used for both SDLC and synchronous data, the data path taken for each mode is different. In Bisync protocol, a byte-oriented operation requires that the CPU decide to include the data character in CRC. To allow the CPU ample time to make this decision, the Z80-SIO provides an 8-bit delay for synchronous data. In the SDLC mode, no delay is provided since the Z80-SIO contains logic that determines the bytes on which CRC is calculated.

The transmitter has an 8-bit transmit data register that is loaded from the internal data bus and a 20-bit transmit shift register that can be loaded from WR6, WR7 and the transmit data register. WR6 and WR7 contain sync characters in the Monosync or Bisync modes, or address field (one character long) and flag respectively in the SDLC mode. During Synchronous modes, information contained in WR6 and WR7 is loaded into the transmit shift register at the beginning of the message and, as a time filler, in the middle of the message if a Transmit Underrun condition occurs. In the SDLC mode, the flags are loaded into the transmit shift register at the beginning and end of message.

Asynchronous data in the transmit shift register is formatted with start and stop bits and is shifted out to the transmit multiplexer at the selected clock rate. Synchronous (Monosync or Bisync) data is shifted out to the transmit multiplexer and also to the CRC generator at the × 1 clock rate.

SDLC/HDLC data is shifted out through the zero insertion logic, which is disabled while the flags are being sent. For all other fields (address, control and frame check) a 0 is inserted following five contiguous 1's in the data stream. The CRC generator result for SDLC data is also routed through the zero insertion logic.

Functional Description

The functional capabilities of the Z80-SIO can be described from two different points of view: as a data communications device, it transmits and receives serial data, and meets the requirements of various data communications protocols; as a Z80 family peripheral, it interacts with the Z80-CPU and other Z80 peripheral circuits, and shares their data, address and control busses, as well as being a part of the Z80 interrupt structure. As a peripheral to other microprocessors, the Z80-SIO offers valuable features such as non-vectored interrupts, polling and simple handshake capabilities.

The first part of the following functional description describes the interaction between the CPU and Z80-SIO; the second part introduces its data communications capabilities.

I/O CAPABILITIES

The Z80-SIO offers the choice of Polling, Interrupt (vectored or non-vectored) and Block Transfer modes to transfer data, status and control information to and from the CPU. The Block Transfer mode can be implemented under CPU or DMA control.

Polling. The Polled mode avoids interrupts. Status registers RR0 and RR1 are updated at appropriate times for each function being performed (for example, CRC Error status valid at the end of the message). All the interrupt modes of the Z80-SIO must be disabled to operate the device in a polled environment.

While in its Polling sequence, the CPU examines the status contained in RR0 for each channel; the RR0 status bits serve as an acknowledge to the Poll inquiry. The two RR0 status bits D_0 and D_2 indicate that a receive or transmit data transfer is needed. The status also indicates Error or other special status conditions (see "Z80-SIO Programming"). The Special Receive Condition status contained in RR1 does not have to be read in a Polling sequence because the status bits in *RR1 are* accompanied by a Receive Character Available status in RR0.

Interrupts. The Z80-SIO offers an elaborate interrupt scheme to provide fast interrupt response in real-time applications. As mentioned earlier, Channel B registers WR2 and RR2 contain the interrupt vector that points to an interrupt service routine in the memory. To service operations in both channels and to eliminate the necessity of writing a status analysis routine, the Z80-SIO can modify the interrupt vector in RR2 so it points directly to one of eight interrupt service routines. This is done under program_control by setting a program bit (WR1, D_2) in Channel B called "Status Affects Vector." When this bit is set, the interrupt vector in WR2 is modified according to the assigned priority of the various interrupting conditions. The table in the Write Register 1 description (Z80-SIO Programming section) shows the modification details.

Transmit interrupts, Receive interrupts and External/Status interrupts are the main sources of interrupts (Figure 5). Each interrupt source is enabled under program control with Channel A having a higher priority than Channel B, and with Receiver, Transmit and External/Status interrupts prioritized in that order within each channel. When the Transmit interrupt is enabled, the CPU is interrupted by the transmit buffer *becoming* empty. (This implies that the transmitter must have had a data character written into it so it can become empty.) When enabled, the receiver can interrupt the CPU in one of three ways:

- Interrupt on first receive character
- Interrupt on all receive characters
- Interrupt on a Special Receive condition

Interrupt On First Character is typically used with the Block Transfer mode. Interrupt On All Receive Characters has the option of modifying the interrupt vector in the event of a parity error. The Special Receive Condition interrupt can occur on a character or message basis (End Of Frame interrupt in SDLC, for example). The Special Receive condition can cause an interrupt only if the Interrupt On First Receive Character or Interrupt On All Receive Characters mode is selected. In Interrupt On First Receive Character, an interrupt can occur from Special Receive conditions (except Parity Error) after the first receive character interrupt (example: Receive Overrun interrupt).

The main function of the External/Status interrupt is to monitor the signal transitions of the $\overline{CTS}$, $\overline{DCD}$ and $\overline{SYNC}$ pins; however, an External/Status interrupt is also caused by a Transmit Underrun condition or by the detection of a Break (Asynchronous mode) or Abort (SDLC mode) sequence in the data stream. The interrupt caused by the Break/Abort sequence has a special feature that allows the Z80-SIO to interrupt when the Break/Abort sequence is detected or terminated. This feature facilitates the proper termination of the current message, correct initialization of the next message, and the accurate timing of the Break/Abort condition in external logic.

CPU/DMA Block Transfer. The Z80-SIO provides a Block Transfer mode to accommodate CPU block transfer functions and DMA controllers (Z80-DMA or other designs). The Block Transfer mode uses the $\overline{\text{WAIT}}/\overline{\text{READY}}$ output in conjunction with the Wait/Ready bits of Write Register 1. The $\overline{\text{WAIT}}/\overline{\text{READY}}$ output can be defined under software control as a WAIT line in the CPU Block Transfer mode or as a READY line in the DMA Block Transfer mode.

To a DMA controller, the Z80-SIO READY output indicates that the Z80-SIO is ready to transfer data to or from memory. To the CPU, the WAIT output indicates that the Z80-SIO is not ready to transfer data, thereby requesting the CPU to extend the I/O cycle. The programming of bits 5, 6 and 7 of Write Register 1 and the logic states of the $\overline{\text{WAIT}}/\overline{\text{READY}}$ line are defined in the Write Register 1 description (Z80-SIO Programming section.)

DATA COMMUNICATIONS CAPABILITIES

In addition to the I/O capabilities previously discussed, the Z80-SIO provides two independent full-duplex channels as well as Asynchronous, Synchronous and SDLC (HDLC) operational modes. These modes facilitate the implementation of commonly used data communications protocols.

The specific features of these modes are described in the following sections. To preserve the independence and completeness of each section, some information common to all modes is repeated.

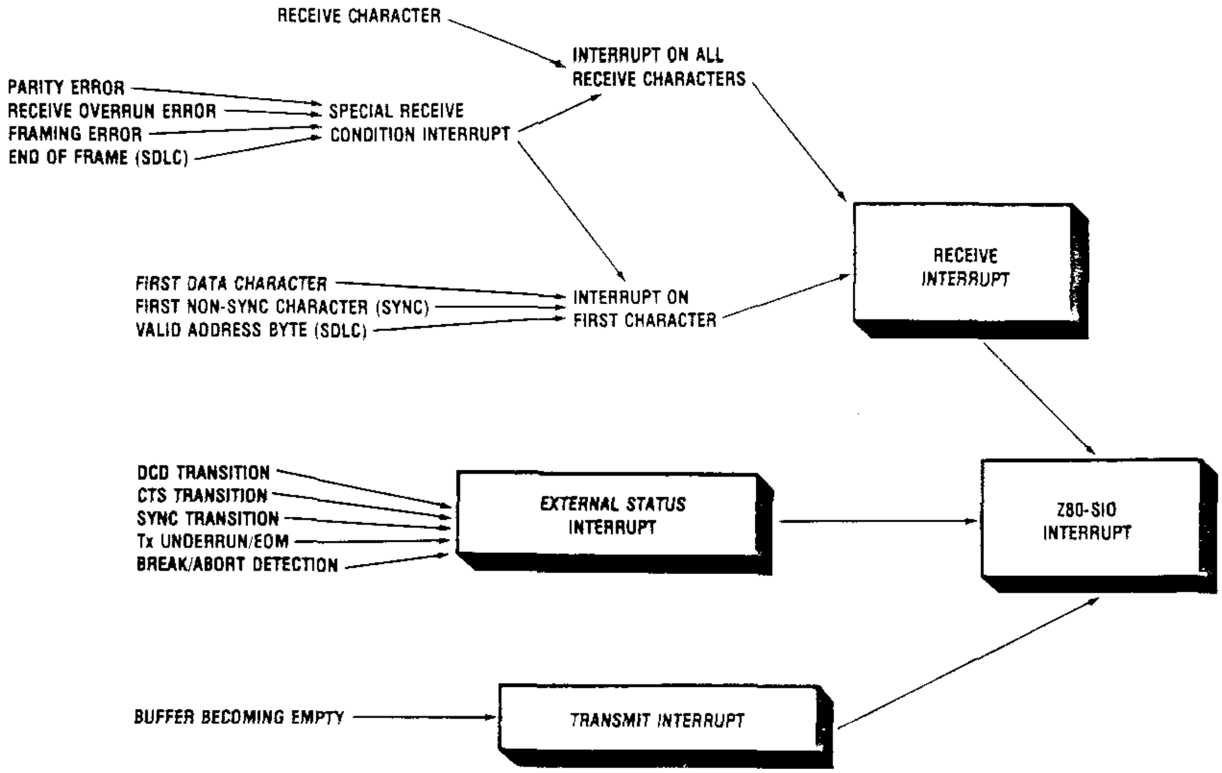


Figure 5. Interrupt Structure

Asynchronous Operation

To receive or transmit data in the Asynchronous mode, the Z80-SIO must be initialized with the following parameters: character length, clock rate, number of stop bits, even or odd parity, interrupt mode, and receiver or transmitter enable. The parameters are loaded into the appropriate write registers by the system program. WR4 parameters must be issued before WR1, WR3 and WR5 parameters or commands.

If the data is transmitted over a modem or RS232C interface, the $\overline{\text{REQUEST TO SEND}}$ ($\overline{\text{RTS}}$) and $\overline{\text{DATA TERMINAL READY}}$ ($\overline{\text{DTR}}$) outputs must be set along with the Transmit Enable bit. Transmission cannot begin until the Transmit Enable bit is set.

The Auto Enables feature allows the programmer to send the first data character of the message to the Z80-SIO without waiting for $\overline{\text{CTS}}$. If the Auto Enables bit is set, the Z80-SIO will wait for the $\overline{\text{CTS}}$ pin to go Low before it begins data transmission. $\overline{\text{CTS}}$, $\overline{\text{DCD}}$ and $\overline{\text{SYNC}}$ are general-purpose I/O lines that may be used for functions other than their labeled purposes. If $\overline{\text{CTS}}$ is used for another purpose, the Auto Enables Bit must be programmed to 0.

Figure 6 illustrates asynchronous message formats; Table 2 shows WR3, WR4 and WR5 with bits set to indicate the applicable modes, parameters and commands in asynchronous modes. WR2 (Channel B only) stores the interrupt vector; WR1 defines the interrupt modes and data transfer modes. WR6 and WR7 are not used in asynchronous modes. Table 3 shows the typical program steps that implement a full-duplex receive/transmit operation in either channel.

Asynchronous Transmit

The Transmit Data output (TxD) is held marking (High) when the transmitter has no data to send. Under program control, the Send Break (WR5, D_4) command can be issued to hold TxD spacing (Low) until the command is cleared.

The Z80-SIO automatically adds the start bit, the programmed parity bit (odd, even or no parity) and the programmed number of stop bits to the data character to be transmitted. When the character length is six or seven bits, the unused bits are automatically ignored by the Z80-SIO. If the character length is five bits or less, refer to the table in the Write Register 5 description (Z80-SIO Programming section) for the data format.

Serial data is shifted from TxD at a rate equal to 1, 1/16th, 1/32nd or 1/64th of the clock rate supplied to the Transmit Clock input ($\overline{\text{TxC}}$). Serial data is shifted out on the falling edge of ($\overline{\text{TxC}}$).

If set, the External/Status Interrupt mode monitors the status of $\overline{\text{DCD}}$, $\overline{\text{CTS}}$ and $\overline{\text{SYNC}}$ throughout the transmission of the message. If these inputs change for a period of time greater than the minimum specified pulse width, the interrupt is generated. In a transmit operation, this feature is used to monitor the modem control signal $\overline{\text{CTS}}$.

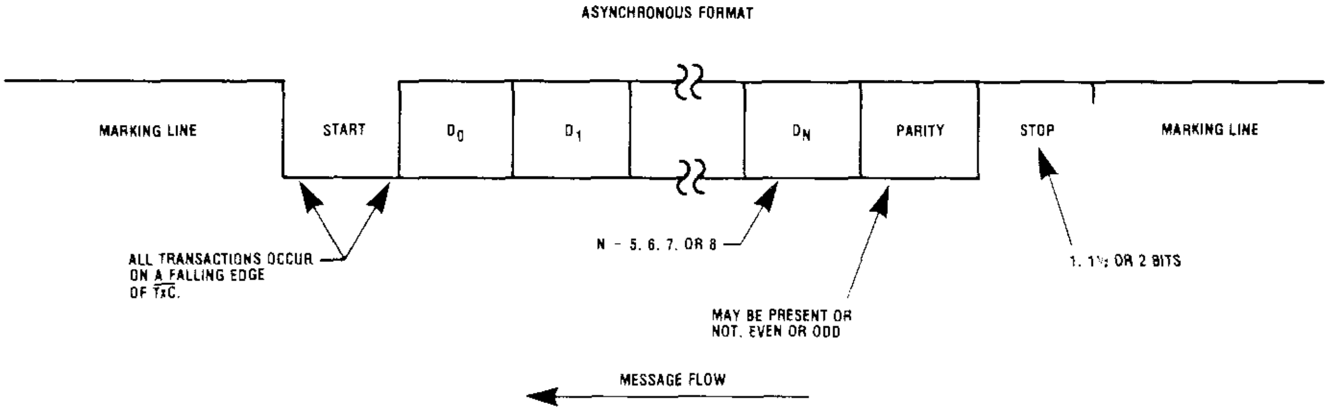


Figure 6. Asynchronous Message Format

Asynchronous Receive

An Asynchronous Receive operation begins when the Receive Enable bit is set. If the Auto Enables option is selected, $\overline{DCD}$ must be Low as well. A Low (spacing) condition on the Receive Data input (RxD) indicates a start bit. If this Low persists for at least one-half of a bit time, the start bit is assumed to be valid and the data input is then sampled at mid-bit time until the entire character is assembled. This method of detecting a start bit improves error rejection when noise spikes exist on *an otherwise marking line.*

If the × 1 clock mode is selected, bit synchronization must be accomplished externally. Receive data is sampled on the rising edge of RxC. The receiver inserts 1's when a character length of other than eight bits is used. If parity is enabled, the parity bit is not stripped from the assembled character for character lengths other than eight bits. For lengths other than eight bits, the receiver assembles a character length of the required number of data bits, plus a parity bit and 1's for any unused bits. For example, the receiver assembles a 5-bit character with the following format: $11\ P\ D_4\ D_3\ D_2\ D_1\ D_0$.

Since the receiver is buffered by three 8-bit registers in addition to the receive shift register, the CPU has enough time to service an interrupt and to accept the data character assembled by the Z80-SIO. The receiver also has three buffers that store error flags for each data character in the receive buffer. These error flags are loaded at the same time as the data characters.

After a character is received, it is checked for the following error conditions:

- When parity is enabled, the Parity Error bit (RR1, D_4) is set whenever the parity bit of the character does not match with the programmed parity. Once this bit is set, it remains set until the Error Reset Command (WR0) is given.

- The Framing Error bit (RR1, D_6) is set if the character is assembled without any stop bits (that is, a Low level detected for a stop bit). Unlike the Parity Error bit, this bit is set (and not latched) only for the character on which it occurred. Detection of framing error adds an additional one-half of a bit time to the character time so the framing error is not interpreted as a new start bit.

- If the CPU fails to read a data character while more than three characters have been received, the Receive Overrun bit (RR1, D_5) is set. When this occurs, the fourth character assembled replaces the third character in the receive buffers. With this arrangement, only the character that has been written over is flagged with the Receive Overrun Error bit. Like Parity Error, this bit can only be reset by the Error Reset command from the CPU. Both the Framing Error and Receive Overrun Error cause an interrupt with the interrupt vector indicating a Special Receive condition (if Status Affects Vector is selected).

Since the Parity Error and Receive Overrun Error flags are latched, the error status that is read reflects an error in the current word in the receive buffer plus any Parity or Overrun Errors received since the last Error Reset command. To keep correspondence between the state of the error buffers and the contents of the receive data buffers, the error status register must be read before the data. This is easily accomplished if vectored

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
WR3	00 = Rx 5 BITS/CHAR 10 = Rx 6 BITS/CHAR 01 = Rx 7 BITS/CHAR 11 = Rx 8 BITS/CHAR		AUTO ENABLES	0	0	0	0	Rx ENABLE
WR4	00 = × 1 CLOCK MODE 01 = × 16 CLOCK MODE 10 = × 32 CLOCK MODE 11 = × 64 CLOCK MODE		0	0	00 = NOT USED 01 = 1 STOP BIT/CHAR 10 = 1½ STOP BITS/CHAR 11 = 2 STOP BITS/CHAR		EVEN/$\overline{ODD}$ PARITY	PARITY ENABLE
WR5	DTR	00 = Tx 5 BITS (OR LESS)/CHAR 10 = Tx 6 BITS/CHAR 01 = Tx 7 BITS/CHAR 11 = Tx 8 BITS/CHAR		SEND BREAK	Tx ENABLE	0	RTS	0

Table 2. Contents of Write Registers 3, 4 and 5 in Asynchronous Modes

FUNCTION	TYPICAL PROGRAM STEPS		COMMENTS
	REGISTER:	*INFORMATION LOADED:*	
INITIALIZE	WR0	CHANNEL RESET	Reset SIO
	WR0	POINTER 2	
	WR2	INTERRUPT VECTOR	Channel B only
	WR0	POINTER 4. RESET EXTERNAL/STATUS INTERRUPT	
	WR4	ASYNCHRONOUS MODE. PARITY INFORMATION. STOP BITS INFORMATION. CLOCK RATE INFORMATION	Issue parameters
	WR0	POINTER 3	
	WR3	RECEIVE ENABLE. AUTO ENABLES, RECEIVE CHARACTER LENGTH	
	WR0	POINTER 5	
	WR5	REQUEST TO SEND, TRANSMIT ENABLE. TRANSMIT CHARACTER LENGTH, DATA TERMINAL READY	Receive and Transmit both fully initialized. Auto Enables will enable Transmitter if $\overline{CTS}$ is active and Receiver if $\overline{DCD}$ is active.
	WR0	POINTER 1, RESET EXTERNAL/STATUS INTERRUPT	
	WR1	TRANSMIT INTERRUPT ENABLE, STATUS AFFECTS VECTOR, INTERRUPT ON ALL RECEIVE CHARACTERS, DISABLE WAIT/READY FUNCTION, EXTERNAL INTERRUPT ENABLE	Transmit/Receive interrupt mode selected. External Interrupt monitors the status of the $\overline{CTS}$. $\overline{DCD}$ and $\overline{SYNC}$ inputs and detects the Break sequence. Status Affects Vector in Channel B only.
	TRANSFER FIRST DATA BYTE TO SIO		This data byte must be transferred or no transmit interrupts will occur.
IDLE MODE	EXECUTE HALT INSTRUCTION OR SOME OTHER PROGRAM		Program is waiting for an interrupt from the SIO.
DATA TRANSFER AND ERROR MONITORING	Z80 INTERRUPT ACKNOWLEDGE CYCLE TRANSFERS RR2 TO CPU *IF A CHARACTER IS RECEIVED:* • TRANSFER DATA CHARACTER TO CPU • UPDATE POINTERS AND PARAMETERS • RETURN FROM INTERRUPT		When the interrupt occurs, the interrupt vector is modified by: 1. Receive Character Available; 2. Transmit Buffer Empty; 3. External/Status change; and 4. Special Receive condition.
	IF TRANSMITTER BUFFER IS EMPTY: • TRANSFER DATA CHARACTER TO SIO • UPDATE POINTERS AND PARAMETERS • RETURN FROM INTERRUPT		Program control is transferred to one of the eight interrupt service routines.
	IF EXTERNAL STATUS CHANGES: • TRANSFER RR0 TO CPU • PERFORM ERROR ROUTINES (INCLUDE BREAK DETECTION) • RETURN FROM INTERRUPT		If used with processors other than the Z80, the modified interrupt vector (RR2) should be returned to the CPU in the Interrupt Acknowledge sequence.
	IF SPECIAL RECEIVE CONDITION OCCURS: • TRANSFER RR1 TO CPU • DO SPECIAL ERROR (E.G. FRAMING ERROR) ROUTINE • RETURN FROM INTERRUPT		
TERMINATION	REDEFINE RECEIVE/TRANSMIT INTERRUPT MODES		When transmit or receive data transfer is complete.
	DISABLE TRANSMIT/RECEIVE MODES		
	UPDATE MODEM CONTROL OUTPUTS (E.G. RTS OFF)		In Transmit, the All Sent status bit indicates transmission is complete.

Table 3. Asynchronous Mode

interrupts are used, because a special interrupt vector is generated for these conditions.

While the External/Status interrupt is enabled, break detection causes an interrupt and the Break Detected status bit (RR0, D_7) is set. The Break Detected interrupt should be handled by issuing the Reset External/Status Interrupt command to the Z80-SIO in response to the first Break Detected interrupt that has a Break status of 1 (RR0, D_7). The Z80-SIO monitors the Receive Data input and waits for the Break sequence to terminate, at which point the Z80-SIO interrupts the CPU with the Break status set to 0. The CPU must again issue the Reset External/Status Interrupt command in its interrupt service routine to reinitialize the break detection logic.

The External/Status interrupt also monitors the status of $\overline{\text{DCD}}$. If the $\overline{\text{DCD}}$ pin becomes inactive for a period greater than the minimum specified pulse width, an interrupt is generated with the DCD status bit (RR0, D_3) set to 1. Note that the $\overline{\text{DCD}}$ input is inverted in the RR0 status register.

If the status is read after the data, the error data for the next word is also included if it has been stacked in the buffer. If operations are performed rapidly enough so the next character is not yet received, the status register remains valid. An exception occurs when the Interrupt On First Character Only mode is selected. A special interrupt in this mode holds the error data and the character itself (even if read from the buffer) until the Error Reset command is issued. This prevents further data from becoming available in the receiver until the Reset command is issued, and allows CPU intervention on the character with the error even if DMA or block transfer techniques are being used.

If Interrupt On Every Character is selected, the interrupt vector is different if there is an error status in RR1. If a Receiver Overrun occurs, the most recent character received is loaded into the buffer; the character preceding it is lost. When the character that has been written over the other characters is read, the Receive Overrun bit is set and the Special Receive Condition vector is returned if Status Affects Vector is enabled.

In a polled environment, the Receive Character Available bit (RR0, D_0) must be monitored so the Z80-CPU can know when to read a character. This bit is automatically reset when the receive buffers are read. To prevent overwriting data in polled operations, the transmit buffer status must be checked before writing into the transmitter. The Transmit Buffer Empty bit is set to 1 whenever the transmit buffer is empty.

Synchronous Operation

Before describing synchronous transmission and reception, the three types of character synchronization—Monosync, Bisync and External Sync—require some explanation. These modes use the ×1 clock for both Transmit and Receive operations. Data is sampled on the rising edge of the Receive Clock input ($\overline{\text{RxC}}$). Transmitter data transitions occur on the falling edge of the Transmit Clock input ($\overline{\text{TxC}}$).

The differences between Monosync, Bisync and External Sync are in the manner in which initial character synchronization is achieved. The mode of operation must be selected before sync characters are loaded, because the registers are used differently in the various modes. Figure 7 shows the formats for all three of these synchronous modes.

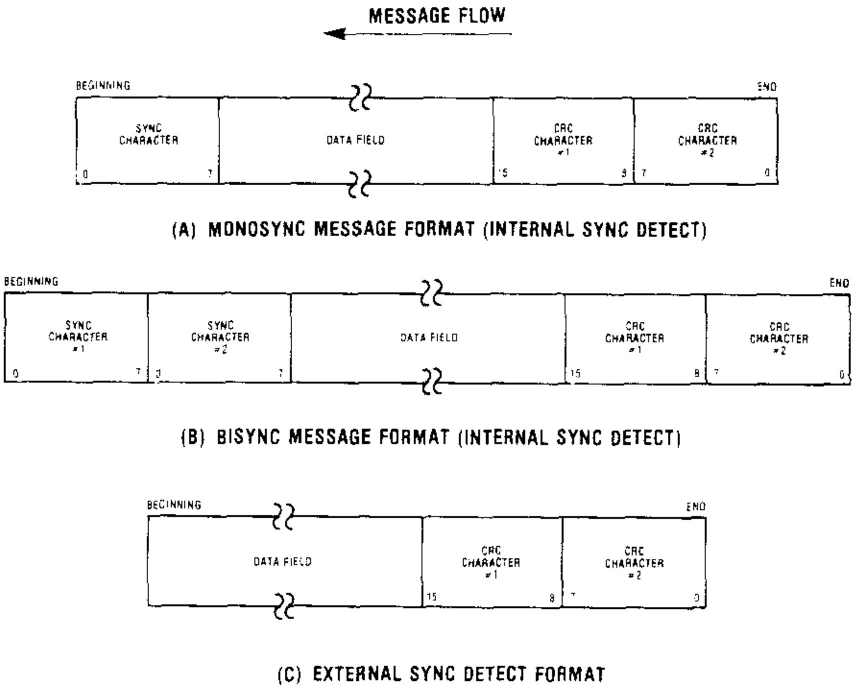
Monosync. In a Receive operation, matching a single sync character (8-bit sync mode) with the programmed sync character stored in WR7 implies character synchronization and enables data transfer.

Bisync. Matching two contiguous sync characters (16-bit sync mode) with the programmed sync characters stored in WR6 and WR7 implies character synchronization. In both the Monosync and Bisync modes, $\overline{\text{SYNC}}$ is used as an output, and is active for the part of the receive clock that detects the sync character.

External Sync. In this mode, character synchronization is established externally; $\overline{\text{SYNC}}$ is an input that indicates external character synchronization has been achieved. After the sync pattern is detected, the external logic must wait for two full Receive Clock cycles to activate the $\overline{\text{SYNC}}$ input. The SYNC input must be held Low until character synchronization is lost. Character assembly begins on the rising edge of $\overline{\text{RxC}}$ that precedes the falling edge of $\overline{\text{SYNC}}$.

In all cases after a reset, the receiver is in the Hunt phase, during which the Z80-SIO looks for character synchronization. The hunt can begin only when the receiver is enabled, and data transfer can begin only when character synchronization has been achieved. If character synchronization is lost, the Hunt phase can be re-entered by writing a control word with the Enter Hunt Phase bit set (WR3, D_4). In the Transmit mode, the transmitter always sends the programmed number of sync bits (8 or 16). In the Monosync mode, the transmitter transmits from WR6; the receiver compares against WR7.

In the Monosync, Bisync and External Sync modes, assembly of received data continues until the Z80-SIO is reset, or until the receiver is disabled (by command or by $\overline{\text{DCD}}$ in the Auto Enables mode), or until the CPU sets the Enter Hunt Phase bit.



(A) MONOSYNC MESSAGE FORMAT (INTERNAL SYNC DETECT)

(B) BISYNC MESSAGE FORMAT (INTERNAL SYNC DETECT)

(C) EXTERNAL SYNC DETECT FORMAT

Figure 7. Synchronous Formats

After initial synchronization has been achieved, the operation of the Monosync, Bisync and External Sync modes is quite similar. Any differences are specified in the following text.

Table 4 shows how WR3, WR4 and WR5 are used in synchronous receive and transmit operations. WR0 points to other registers and issues various commands, WR1 defines the interrupt modes, WR2 stores the interrupt vector, and WR6 and WR7 store sync characters. Table 5 illustrates the typical program steps that implement a half-duplex Bisync transmit operation.

Synchronous Transmit

INITIALIZATION

The system program must initialize the transmitter with the following parameters: odd or even parity, × 1 clock mode, 8- or 16-bit sync character(s), CRC polynomial, Transmitter Enables, Request To Send, Data Terminal Ready, interrupt modes and transmit character length. WR4 parameters must be issued before WR1, WR3, WR5, WR6 and WR7 parameters or commands.

One of two polynomials—CRC-16 ($X^{16}+X^{15}+X^{2}+1$) or SDLC ($X^{16}+X^{12}+X^{5}+1$)—may be used with synchronous modes. In either case (SDLC mode not selected), the CRC generator and checker are reset to all 0's. In the transmit initialization process, the CRC generator is initialized by setting the Reset Transmit CRC Generator command bits (WR0). Both the transmitter and the receiver use the same polynomial.

Transmit Interrupt Enable or Wait/Ready Enable can be selected to transfer the data. The External/Status interrupt mode is used to monitor the status of the $\overline{\text{CLEAR TO SEND}}$ input as well as the Transmit Underrun/EOM latch. Optionally, the Auto Enables feature can be used to enable the transmitter when $\overline{\text{CTS}}$ is active. The first data transfer to the Z80-SIO can begin when the External/Status interrupt occurs (CTS status bit set) or immediately following the Transmit Enable command (if the Auto Enables modes is set).

Transmit data is held marking after reset or if the transmitter is not enabled. Break may be programmed to generate a spacing line that begins as soon as the Send Break bit is set. With the transmitter fully initialized and enabled, the default condition is continuous transmission of the 8- or 16-bit sync character.

DATA TRANSFER AND STATUS MONITORING

In this phase, there are several combinations of interrupts and Wait/Ready.

Data Transfer Using Interrupts. If the Transmit Interrupt Enable bit (WR1, D_1) is set, an interrupt is generated each time the transmit buffer becomes empty. The interrupt can be satisfied either by writing another character into the transmitter or by resetting the Transmitter Interrupt Pending latch with a Reset Transmitter Pending command (WR0, CMD_5). If the interrupt is satisfied with this command and nothing more is written into the transmitter, there can be no further Transmit Buffer Empty interrupts, because it is the process of the buffer becoming empty that causes the interrupts and the buffer cannot become empty when it is already empty. This situation does cause a Transmit Underrun condition, which is explained in the "Bisync Transmit Underrun" section.

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
WR3	00 = Rx 5 BITS/CHAR 10 = Rx 6 BITS/CHAR 01 = Rx 7 BITS/CHAR 11 = Rx 8 BITS/CHAR		AUTO ENABLES	ENTER HUNT MODE	Rx CRC ENABLE	0	SYNC CHAR LOAD INHIBIT	RX ENABLE
WR4	0	0	00 = 8-BIT SYNC CHAR 01 = 16-BIT SYNC CHAR 10 = SDLC MODE 11 = EXT SYNC MODE		0 SELECTS SYNC MODES	0	EVEN/$\overline{\text{ODD}}$ PARITY	PARITY ENABLE
WR5	DTR	00 = Tx 5 BITS (OR LESS)/CHAR 10 = Tx 6 BITS/CHAR 01 = Tx 7 BITS/CHAR 11 = Tx 8 BITS/CHAR		SEND BREAK	Tx ENABLE	1 SELECTS CRC-16	RTS	Tx CRC ENABLE

Table 4. Contents of Write Registers 3, 4 and 5 in Synchronous Modes

Data Transfer Using WAIT/READY. To the CPU, the activation of $\overline{\text{WAIT}}$ indicates that the Z80-SIO is not ready to accept data and that the CPU must extend the output cycle. To a DMA controller, READY indicates that the transmit buffer is empty and that the Z80-SIO is ready to accept the next data character. If the data character is not loaded into the Z80-SIO by the time the transmit shift register is empty, the Z80-SIO enters the Transmit Underrun condition.

Bisync Transmit Underrun. In Bisync protocol, filler characters are inserted to maintain synchronization when the transmitter has no data to send (Transmit Underrun condition). The Z80-SIO has two programmable options for solving this situation: it can insert sync characters, or it can send the CRC characters generated so far, followed by sync characters.

These options are under the control of the Reset Transmit Underrun/EOM command in WR0. Following a chip or channel reset, the Transmit Underrun/EOM status bit (RR0, D_6) is in a set condition and allows the insertion of sync characters when there is no data to send. CRC is not calculated on the automatically inserted sync characters. When the CPU detects the end of message, a Reset Transmit Underrun/EOM command can be issued. This allows CRC to be sent when the transmitter has no data. In this case, the Z80-SIO sends CRC, followed by sync characters, to terminate the message.

There is no restriction as to when in the message the Transmit Underrun/EOM bit can be reset. If Reset is issued after the first data character has been loaded the 16-bit CRC is sent and followed by sync characters the first time the transmitter has no data to send. Because of the Transmit Underrun condition, an External/Status interrupt is generated whenever the Transmit Underrun/EOM bit becomes set.

In the case of sync insertion, an interrupt is generated only after the first automatically inserted sync character has been loaded. The status indicates the Transmit Underrun/EOM bit and the Transmit Buffer Empty bit are set.

In the case of CRC insertion, the Transmit Underrun/EOM bit is set and the Transmit Buffer Empty bit is reset while CRC is being sent. When CRC has been completely sent, the Transmit Buffer Empty status bit is set and an interrupt is generated to indicate to the CPU that another message can begin (this interrupt occurs because CRC has been sent and sync has been loaded). If no more messages are to be sent, the program can terminate transmission by resetting RTS, and disabling the transmitter (WR5, D_3).

Pad characters may be sent by setting the Z80-SIO to 8 bits/transmit character and writing FF to the transmitter while CRC is being sent. Alternatively, the sync characters can be redefined as pad characters during this time. The following example is included to clarify this point.

The Z80-SIO interrupts with the Transmit Buffer Empty bit set.

The CPU recognizes that the last character (ETX) of the message has already been sent to the Z80-SIO by examining the internal program status.

To force the Z80-SIO to send CRC, the CPU issues the Reset Transmit Underrun/EOM Latch command (WR0) and satisfies the interrupt with the Reset Transmit Interrupt Pending command. (This command prevents the Z80-SIO from requesting more data.) Because of the transmit underrun caused by this command, the Z80-SIO starts sending CRC. The Z80-SIO also causes an External/Status interrupt with the Transmit Underrun/EOM latch set.

The CPU satisfies this interrupt by loading pad characters into the transmit buffer and issuing the Reset External/Status Interrupt command.

With this sequence, CRC is followed by a pad character instead of a sync character. Note that the Z80-SIO will interrupt with a Transmit Buffer Empty interrupt when CRC is completely sent and that the pad character is loaded into the transmit shift register.

From this point on the CPU can send more pad characters or sync characters.

Bisync CRC Generation. Setting the Transmit CRC enable bit (WR5, D_0) initiates CRC accumulation when the program sends the first data character to the Z80-SIO. Although the Z80-SIO automatically transmits up to two sync characters (16-bit sync), it is wise to send a few more sync characters ahead of the message (before enabling Transmit CRC) to ensure synchronization at the receiving end.

The transmit CRC Enable bit can be changed on the fly any time in the message to include or exclude a particular data character from CRC accumulation. The Transmit CRC Enable bit should be in the desired state when the data character is loaded from the transmit data buffer into the transmit shift register. To ensure this bit is in the proper state, the Transmit CRC Enable bit must be issued before sending the data character to the Z80-SIO.

Transmit Transparent Mode. Transparent mode (Bisync protocol) operation is made possible by the ability to change Transmit CRC Enable on the fly and by the additional capability of inserting 16-bit sync characters. Exclusion of DLE characters from CRC calculation can be achieved by disabling CRC calculation immediately preceding the DLE character transfer to the Z80-SIO.

In the case of a Transmit Underrun condition in the Transparent mode, a pair of DLE-SYN characters are sent. The Z80-SIO can be programmed to send the DLE-SYN sequence by loading a DLE character into WR6 and a sync character into WR7.

Transmit Termination. The Z80-SIO is equipped with a special termination feature that maintains data integrity and validity. If the transmitter is disabled while a data or sync character is being sent, that character is sent as usual, but is followed by a marking line rather than CRC or sync characters. When the transmitter is disabled, a

FUNCTION	TYPICAL PROGRAM STEPS		COMMENTS
	REGISTER:	*INFORMATION LOADED:*	
INITIALIZE	WR0	CHANNEL RESET. RESET TRANSMIT CRC GENERATOR	Reset SIO. initilize CRC generator.
	WR0	POINTER 2	
	WR2	INTERRUPT VECTOR	Channel B only
	WR0	POINTER 3	
	WR3	AUTO ENABLES	Transmission begins only after $\overline{CTS}$ is *detected.*
	WR0	POINTER 4	
	WR4	PARITY INFORMATION. SYNC MODES INFORMATION. ×1 CLOCK MODE	Issue transmit parameters.
	WR0	POINTER 6	
	WR6	SYNC CHARACTER 1	
	WR0	*POINTER 7 RESET EXTERNAL/STATUS INTERRUPTS*	
	WR7	SYNC CHARACTER 2	
	WR0	POINTER 1, RESET EXTERNAL/STATUS INTERRUPTS	
	WR1	STATUS AFFECTS VECTOR. EXTERNAL INTERRUPT ENABLE. TRANSMIT INTERRUPT ENABLE OR WAIT/READY MODE ENABLE	External Interrupt mode monitors the status of $\overline{CTS}$ and $\overline{DCD}$ input pins as well as the status of Tx Underrun/EOM latch Transmit Interrupt Enable interrupts when the Transmit buffer becomes empty; the Wait/Ready mode can be used to transfer data using DMA or CPU Block Transfer.
	WR0	POINTER 5	*Status Affects Vector* (Channel B only).
	WR5	REQUEST TO SEND. TRANSMIT ENABLE. BISYNC CRC. TRANSMIT CHARACTER LENGTH	Transmit CRC Enable should be set when first non-sync data is sent to Z80-SIO.
	FIRST SYNC BYTE TO SIO		Need several sync characters in the beginning of message. Transmitter is fully initialized.
IDLE MODE	EXECUTE HALT INSTRUCTION OR SOME OTHER PROGRAM		Waiting for interrupt or Wait/Ready output to transfer data.
DATA TRANSFER AND STATUS MONITORING	*WHEN INTERRUPT (WAIT/READY) OCCURS:* • INCLUDE/EXECLUDE DATA BYTE FROM CRC ACCUMULATION (IN SIO). • TRANSFER DATA BYTE FROM CPU (OR MEMORY) TO SIO. • DETECT AND SET APPROPRIATE FLAGS FOR CONTROL CHARACTERS (IN CPU). • RESET Tx UNDERRUN/EOM LATCH (WR0) IF LAST CHARACTER OF MESSAGE IS DETECTED. • UPDATE POINTERS AND PARAMETERS (CPU). • *RETURN FROM INTERRUPT.*		Interrupt occurs (Wait/Ready becomes active) when first data byte is being sent. Wait mode allows CPU block transfer from memory to SIO; Ready mode allows DMA block transfer from memory to SIO. The DMA chip can be programmed to capture special control characters (by examining only the bits that specify ASCII or EBCDIC control characters), and interrupt CPU.
	IF ERROR CONDITION OR STATUS CHANGE OCCURS: • TRANSFER RR0 TO CPU. • EXECUTE ERROR ROUTINE • RETURN FROM INTERRUPT.		Tx Underrun/EOM indicates either transmit underrun (sync character being sent) or end of message (CRC-16 being sent).
TERMINATION	REDEFINE INTERRUPT MODES.		
	UPDATE MODEM CONTROL OUTPUTS (E.G., TURN OFF RTS).		Program should gracefully terminate message.
	DISABLE TRANSMIT MODE		

Table 5. Bisync Transmit Mode

character in the buffer remains in the buffer. If the transmitter is disabled while CRC is being sent, the 16-bit transmission is completed, but sync is sent instead of CRC.

A programmed break is effective as soon as it is written into the control register; characters in the transmit buffer and shift register are lost.

In all modes, characters are sent with the least significant bits first. This requires right-hand justification of transmitted data if the word length is less than eight bits. If the word length is five bits or less, the special technique described in the Write Register 5 discussion (Z80-SIO Programming section) must be used for the data format. The states of any unused bits in a data character are irrelevant, except when in the Five Bits Or Less mode.

If the External/Status Interrupt Enable bit is set, transmitter conditions such as "starting to send CRC characters," "starting to send sync characters," and $\overline{\text{CTS}}$ changing state cause interrupts that have a unique vector if Status Affects Vector is set. This interrupt mode may be used during block transfers.

All interrupts may be disabled for operation in a Polled mode, or to avoid interrupts at inappropriate times during the execution of a program.

Synchronous Receive

INITIALIZATION

The system program initiates the Synchronous Receive operation with the following parameters: odd or even parity, 8- or 16-bit sync characters, ×1 clock mode, CRC polynomial, receive character length, etc. Sync characters must be loaded into registers WR6 and WR7. The receivers can be enabled only after all receive parameters are set. WR4 parameters must be issued before WR1, WR3, WR5, WR6 and WR7 parameters or commands.

After this is done, the receiver is in the Hunt phase. It remains in this phase until character synchronization is achieved. Note that, under program control, all the leading sync characters of the message can be inhibited from loading the receive buffers by setting the Sync Character Load Inhibit bit in WR3.

DATA TRANSFER AND STATUS MONITORING

After character synchronization is achieved, the assembled characters are transferred to the receive data FIFO. The following four interrupt modes are available to transfer the data and its associated status to the CPU.

No Interrupts Enabled. This mode is used for a purely polled operation or for off-line conditions.

Interrupt On First Character Only. This mode is normally used to start a polling loop or a Block Transfer instruction using $\overline{\text{WAIT/READY}}$ to synchronize the CPU or the DMA device to the incoming data rate. In this mode, the Z80-SIO interrupts on the first character and thereafter interrupts only if Special Receive conditions are detected. The mode is reinitialized with the Enable Interrupt On Next Receive Character command to allow the next character received to generate an interrupt. Parity errors do not cause interrupts in this mode, but End Of Frame (SDLC mode) and Receive Overrun do.

If External/Status interrupts are enabled, they may interrupt any time $\overline{\text{DCD}}$ changes state.

Interrupt On Every Character. Whenever a character enters the receive buffer, an interrupt is generated. Error and Special Receive conditions generate a special vector if Status Affects Vector is selected. Optionally, a Parity Error may be directed not to generate the special interrupt vector.

Special Receive Condition Interrupts. The Special Receive Condition interrupt can occur only if either the Receive Interrupt On First Character Only or Interrupt On Every Receive Character modes is also set. The Special Receive Condition interrupt is caused by the Receive Overrun error condition. Since the Receive Overrun and Parity error status bits are latched, the error status—when read—reflects an error in the current word in the receive buffer in addition to any Parity or Overrun errors received since the last Error Reset command. These status bits can only be reset by the Error reset command.

CRC Error Checking and Termination. A CRC error check on the receive message can be performed on a per character basis under program control. The Receive CRC Enable bit (WR3, D_3) must be set/reset by the program before the next character is transferred from the receive shift register into the receive buffer register. This ensures proper inclusion or exclusion of data characters in the CRC check.

To allow the CPU ample time to enable or disable the CRC check on a particular character, the Z80-SIO calculates CRC eight bit times after the character has been transferred to the receive buffer. If CRC is enabled before the next character is transferred, CRC is calculated on the transferred character. If CRC is disabled before the time of the next transfer, calculation proceeds on the word in progress, but the word just transferred to the buffer is not included. When these requirements are satisfied, the 3-byte receive data buffer is, in effect, unusable in Bisync operation. CRC may be enabled and disabled as many times as necessary for a given calculation.

In the Monosync, Bisync and External Sync modes, the CRC/Framing Error bit (RR1, D_6) contains the comparison result of the CRC checker 16 bit times (eight bits delay and eight shifts for CRC) after the character has been transferred from the receive shift register to the buffer. The result should be zero, indicating an error-free transmission. (Note that the result is valid only at the end of CRC calculation. If the result is examined before this time, it usually indicates an error.) The comparison is made with each transfer and is valid only as long as the character remains in the receive FIFO.

Following is an example of the CRC checking operation when four characters (A, B, C and D) are received in that order.

Character A loaded into buffer
Character B loaded into buffer

If CRC is disabled before C is in the buffer, CRC is not calculated on B.

Character C loaded into buffer

After C is loaded, the CRC/Framing Error bit shows the result of the comparison through character A.

Character D loaded into buffer

After D is in the buffer, the CRC Error bit shows the result of the comparison through character B whether or not B was included in the CRC calculations.

Due to the serial nature of CRC calculation, the Receive Clock ($\overline{\text{RxC}}$) must cycle 16 times (8-bit delay plus 8-bit CRC shift) after the second CRC character has been loaded into the receive buffer, or 20 times (the previous 16 plus 3-bit buffer delay and 1-bit input delay) after the last bit is at the RxD input, before CRC calculation is complete. A faster external clock can be gated into the Receive Clock input to supply the required 16 cycles. The Transmit and Receive Data Path diagram (Figure 4) illustrates the various points of delay in the CRC path.

The typical program steps that implement a half-duplex Bisync Receive mode are illustrated in Table 6. The complete set of command and status bit definitions are explained under "Z80-SIO Programming."

FUNCTION	TYPICAL PROGRAM STEPS		COMMENTS
	REGISTER:	*INFORMATION LOADED*	
	WR0	CHANNEL RESET, RESET RECEIVE CRC CHECKER	Reset SIO; initialize Receive CRC checker.
	WR0	POINTER 2	
	WR2	INTERRUPT VECTOR	Channel B only
	WR0	POINTER 4	
	WR4	PARITY INFORMATION, SYNC MODES INFORMATION, ×1 CLOCK MODE	Issue receive parameters.
	WR0	POINTER 5, RESET EXTERNAL STATUS INTERRUPT	
	WR5	BISYNC CRC-16, DATA TERMINAL READY	
	WR0	POINTER 3	
INITIALIZE	WR3	SYNC CHARACTER LOAD INHIBIT, RECEIVE CRC ENABLE; ENTER HUNT MODE, AUTO ENABLES, RECEIVE CHARACTER LENGTH	Sync character load inhibit strips all the leading sync characters at the beginning of the message. Auto Enables enables the receiver to accept data only after the $\overline{\text{DCD}}$ input is active.
	WR0	POINTER 6	
	WR6	SYNC CHARACTER 1	
	WR0	POINTER 7	
	WR7	SYNC CHARACTER 2	
	WR0	POINTER 1, RESET EXTERNAL/STATUS INTERRUPT	
	WR1	STATUS AFFECTS VECTOR, EXTERNAL INTERRUPT ENABLE, RECEIVE INTERRUPT ON FIRST CHARACTER ONLY	In this interrupt mode, only the first non-sync data character is transferred to the CPU. All subsequent data is transferred on a DMA basis; however Special Receive Condition interrupts will interrupt the CPU. Status Affects Vector used in Channel B only.

Table 6. Bisync Receive Mode

FUNCTION	TYPICAL PROGRAM STEPS	COMMENTS
INITIALIZE (CONTINUED)	WR0 POINTER 3. ENABLE INTERRUPT ON NEXT RECEIVE CHARACTER	Resetting this interrupt mode provides simple program loopback entry for the next transaction.
	WR3 RECEIVE ENABLE. SYNC CHARACTER LOAD INHIBIT. ENTER HUNT MODE, AUTO ENABLE. RECEIVE WORD LENGTH	WR3 is reissued to enable receiver. Receive CRC Enable must be set after receiving SOH or STX character.
IDLE MODE	EXECUTE HALT INSTRUCTION OR SOME OTHER PROGRAM	Receive mode is fully initialized and the system is waiting for interrupt on first character.
DATA TRANSFER AND STATUS MONITORING	*WHEN INTERRUPT ON FIRST CHARACTER OCCURS, THE CPU DOES THE FOLLOWING:* • TRANSFERS DATA BYTE TO CPU • DETECTS AND SETS APPROPRIATE FLAGS FOR CONTROL CHARACTERS (IN CPU) • INCLUDES/EXCLUDES DATA BYTE IN CRC CHECKER • UPDATES POINTERS AND OTHER PARAMETERS • ENABLES WAIT/READY FOR DMA OPERATION • ENABLES DMA CONTROLLER • RETURNS FROM INTERRUPT	During the Hunt mode, the SIO detects two contiguous characters to establish synchronization. The CPU establishes the DMA mode and all subsequent data characters are transferred by the DMA controller. The controller is also programmed to capture special characters (by examining only the bits that specify ASCII or EBCDIC control characters) and interrupt the CPU upon detection. In response, the CPU examines the status or control characters and takes appropriate action (e.g. CRC Enable Update).
	WHEN WAIT/READY BECOMES ACTIVE, THE DMA CONTROLLER DOES THE FOLLOWING: • TRANSFERS DATA BYTE TO MEMORY • INTERRUPTS CPU IF A SPECIAL CHARACTER IS CAPTURED BY THE DMA CONTROLLER • INTERRUPTS THE CPU IF THE LAST CHARACTER OF THE MESSAGE IS DETECTED	
	FOR MESSAGE TERMINATION, THE CPU DOES THE FOLLOWING: • TRANSFERS RR1 TO THE CPU • SETS ACK/NAK REPLY FLAG BASED ON CRC RESULT • UPDATES POINTERS AND PARAMETERS • RETURNS FROM INTERRUPT	The SIO interrupts the CPU for error condition, and the error routine aborts the present message, clears the error condition, and repeats the operation.
TERMINATION	REDEFINE INTERRUPT MODES AND SYNC MODES UPDATE MODEM CONTROLS DISABLES RECEIVE MODE	

Table 6. Bisync Receive Mode (Continued)

SDLC (HDLC) Operation

The Z80-SIO is capable of handling both High-level Synchronous Data Link Control (HDLC) and IBM Synchronous Data Link Control (SDLC) protocols. In the following text, only SDLC is referred to because of the high degree of similarity between SDLC and HDLC.

The SDLC mode is considerably different than Synchronous Bisync protocol because it is bit oriented rather than character oriented and, therefore, can naturally handle transparent operation. Bit orientation makes SDLC a flexible protocol in terms of message length and bit patterns. The Z80-SIO has several built-in features to handle variable message length. Detailed information concerning SDLC protocol can be found in literature published on this subject, such as IBM document GA27-3093.

The SDLC message, called the frame (Figure 8), is opened and closed by flags that are similar to the sync characters in Bisync protocol. The Z80-SIO handles the transmission and recognition of the flag characters that mark the beginning and end of the frame. Note that the Z80-SIO can receive shared-zero flags, but cannot transmit them. The 8-bit address field of an SDLC frame contains the secondary station address. The Z80-SIO has an Address Search mode that recognizes the secondary station address so it can accept or reject the frame.

Since the control field of the SDLC frame is transparent to the Z80-SIO, it is simply transferred to the CPU. The Z80-SIO handles the Frame Check sequence in a manner that simplifies the program by incorporating features such as initializing the CRC generator to all 1's, resetting the CRC checker when the opening flag is detected in the Receive mode, and sending the Frame Check/Flag sequence in the Transmit mode. Controller hardware is simplified by automatic zero insertion and deletion logic contained in the Z80-SIO.

Table 7 shows the contents of WR3, WR4 and WR5 during SDLC Receive and Transmit modes. WR0 points to other registers and issues various commands. WR1 defines the interrupt modes. WR2 stores the interrupt vector. WR7 stores the flag character and WR6 the secondary address.

SDLC Transmit

INITIALIZATION

Like Synchronous operation, the SDLC Transmit mode must be initialized with the following parameters: SDLC mode, SDLC polynomial, Request To Send, Data Terminal Ready, transmit character length, transmit interrupt modes (or Wait/Ready function), Transmit Enable, Auto Enables and External/Status interrupt.

Selecting the SDLC mode and the SDLC polynomial enables the Z80-SIO to initialize the CRC Generator to all 1's. This is accomplished by issuing the Reset Transmit CRC Generator command (WR0). Refer to the Synchronous Operation section for more details on the interrupt modes.

After reset, or when the transmitter is not enabled, the Transmit Data output is held marking. Break may be programmed to generate a spacing line. With the transmitter fully initialized and enabled, continuous flags are transmitted on the Transmit Data output.

An abort sequence may be sent by issuing the Send Abort command (WR0, CMD_1). This causes at least eight, but less than fourteen, 1's to be sent before the line reverts to continuous flags. It is possible that the Abort sequence (eight 1's) could follow up to five continuous 1 bits (allowed by the zero insertion logic) and thus cause up to thirteen 1's to be sent. Any data being transmitted and any data in the transmit buffer is lost when an abort is issued.

When required, an extra 0 is automatically inserted when there are five contiguous 1's in the data stream. This does not apply to flags or aborts.

DATA TRANSFER AND STATUS MONITORING

There are several combinations of interrupts and the Wait/Ready function in the SLDC mode.

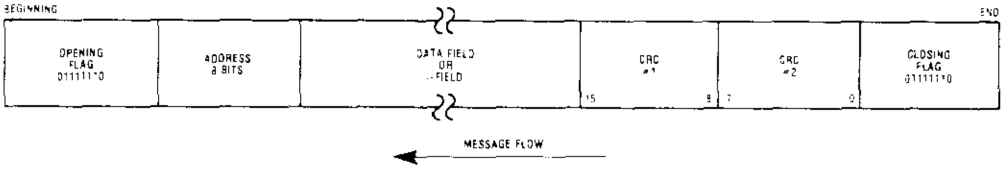


Figure 8. Transmit/Receive SDLC/HDLC Message Format

Data Transfer Using Interrupts. If the Transmit Interrupt Enable bit is set, an interrupt is generated each time the buffer becomes empty. The interrupt may be satisfied either by writing another character into the transmitter or by resetting the Transmit Interrupt Pending latch with a Reset Transmitter Pending command (WR0, CMD$_5$). If the interrupt is satisfied with this command and nothing more is written into the transmitter, there are no further transmitter interrupts. The result is a Transmit Underrun condition. When another character is written and sent out, the transmitter can again become empty and interrupt the CPU. Following the flags in an SDLC operation, the 8-bit address field, control field and information field may be sent to the Z80-SIO using the Transmit Interrupt mode. The Z80-SIO transmits the Frame Check sequence using the Transmit Underrun feature.

When the transmitter is first enabled, it is already empty and obviously cannot then become empty. Therefore, no Transmit Buffer Empty interrupts can occur until after the first data character is written.

When the transmitter is first enabled, it is already empty and cannot then become empty. Therefore, no Transmit Buffer Empty interrupts can occur until after the first data character is written.

Data Transfer Using Wait/Ready. If the Wait/Ready function has been selected, WAIT indicates to the CPU that the Z80-SIO is not ready to accept the data and the CPU must extend the I/O cycle. To a DMA controller, READY indicates that the transmitter buffer is empty and that the Z80-SIO is ready to accept the next character. If the data character is not loaded into the Z80-SIO by the time the transmit shift register is empty, the Z80-SIO enters the Transmit Underrun condition. Address, control and information fields may be transferred to the Z80-SIO with this mode using the Wait/Ready function. The Z80-SIO transmits the Frame Check sequence using the Transmit Underrun feature.

SDLC Transmit Underrun/End Of Message. SDLC-like protocols do not have provisions for fill characters within a message. The Z80-SIO therefore automatically terminates an SDLC frame when the transmit data buffer and output shift register have no more bits to send. It does this by first sending the two bytes of CRC and following these with one or more flags. This technique allows very high-speed transmissions under DMA or CPU block I/O control without requiring the CPU to respond quickly to the end of message situation.

The action that the Z80-SIO takes in the underrun situation depends on the state of the Transmit Underrun/EOM command. Following a reset, the Transmit Underrun/EOM status bit is in the set state and prevents the insertion of CRC characters during the time there is no data to send. Consequently, flag characters are sent. The Z80-SIO begins to send the frame as data is written into the transmit buffer. Between the time the first data byte is written and the end of the message, the Reset Transmit Underrun/EOM command must be issued. Thus the Transmit Underrun/EOM status bit is in the reset state at the end of the message (when underrun occurs), which automatically sends the CRC characters. The sending of CRC again sets the Transmit/Underrun/EOM status bit.

Although there is no restriction as to when the Transmit Underrun/EOM bit can be reset within a message, it is usually reset after the first data character (secondary address) is sent to the Z80-SIO. Resetting this bit allows CRC and flags to be sent when there is no data to send which gives additional time to the CPU for recognizing the fault and responding with an abort command. By resetting it early in the message, the entire message has the maximum amount of CPU response time in an unintentional transmit underrun situation.

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
WR3	00 = Rx 5 BITS/CHAR 10 = Rx 6 BITS/CHAR 01 = Rx 7 BITS/CHAR 11 = Rx 8 BITS/CHAR		AUTO ENABLES	ENTER HUNT MODE (IF INCOMING DATA NOT NEEDED)	Rx CRC ENABLE	ADDRESS SEARCH MODE	0	Rx ENABLE
WR4	0	0	1 SELECTS SDLC MODE	0	0	0	0	0
WR5	DTR	00 = Tx 5 BITS (OR LESS) CHAR 10 = Tx 6 BITS CHAR 01 = Tx 7 BITS/CHAR 11 = Tx 8 BITS/CHAR		0	Tx ENABLE	0 SELECTS SDLC CRC	RTS	Tx CRC ENABLE

Table 7. Contents of Write Registers 3, 4 and 5 in SDLC Modes

When the External/Status interrupt is set and while CRC is being sent, the Transmit Underrun/EOM bit is set and the Transmit Buffer Empty bit is reset to indicate that the transmit register is full of CRC data. When CRC has been completely sent, the Transmit Buffer Empty status bit is set and an interrupt is generated to indicate to the CPU that another message can begin. This interrupt occurs because CRC has been sent and the flag has been loaded. If no more messages are to be sent, the program can terminate transmission by resetting $\overline{\text{RTS}}$, and disabling the transmitter.

In the SDLC mode, it is good practice to reset the Transmit Underrun/EOM status bit immediately after the first character is sent to the Z80-SIO. When the Transmit Underrun is detected, this ensures that the transmission time is filled by CRC characters, giving the CPU enough time to issue the Send Abort command. This also stops the flags from going on the line prematurely and eliminates the possibility of the receiver accepting the frame as valid data. The situation can happen because it is possible that—at the receiving end—the data pattern immediately preceding the automatic flag insertion could match the CRC checker, giving a false CRC check result. The External/Status interrupt is generated whenever the Transmit Underrun/EOM bit is set because of the Transmit Underrun condition.

The transmit underrun logic provides additional protection against premature flag insertion if the proper response is given to the Z80-SIO by the CPU interrupt service routine. The following example is given to clarify this point:

The Z80-SIO raises an interrupt with the Transmit Buffer Empty status bit set.

The CPU does not respond in time and causes a Transmit Underrun condition.

The Z80-SIO starts sending CRC characters (two bytes).

The CPU eventually satisfies the Transmit Buffer Empty interrupt with a data character that follows the CRC character being transmitted.

The Z80-SIO sets the External/Status interrupt with the Transmit Underrun/EOM status bit set.

The CPU recognizes the Transmit Underrun/EOM status and determines from its internal program status that the interrupt is not for "end of message".

The CPU immediately issues a Send Abort Command (WR0) to the Z80-SIO.

The Z80-SIO sends the Abort sequence by destroying whatever data (CRC, data or flag) is being sent.

This sequence illustrates that the CPU has a protection of 22 minimum and 30 maximum transmit clock cycles.

SDLC CRC Generation. The CRC generator must be reset to all 1's at the beginning of each frame before CRC accumulation can begin. Actual accumulation begins when the program sends the address field (eight bits) to the Z80-SIO. Although the Z80-SIO automatically transmits one flag character following the Transmit Enable, it may be wise to send a few more flag characters ahead of the message to ensure character synchronization at the receiving end. This can be done by externally timing out after enabling the transmitter and before loading the first character.

The Transmit CRC Enable (WR5, D_0) should be enabled prior to sending the address field. In the SDLC mode all the characters between the opening and closing flags are included in CRC accumulation, and the CRC generated in the Z80-SIO transmitter is inverted before it is sent on the line.

Transmit Termination. If the transmitter is disabled while a character is being sent, that character (data or flag) is sent in the normal fashion, but is followed by a marking line rather than CRC or flag characters.

A character in the buffer when the transmitter is disabled remains in the buffer; however, a programmed Abort sequence is effective as soon as it is written into the control register. Characters being transmitted, if any, are lost. In the case of CRC, the 16-bit transmission is completed if the transmitter is disabled; however, flags are sent in place of CRC.

In all modes, characters are sent with the least-significant bits first. This requires right-hand justification of data to be transmitted if the word length is less than eight bits. If the word length is five bits or less, the special technique described in the Write Register 5 section ("Z80-SIO Programming" chapter; "Write Registers" section) must be used.

Since the number of bits/character can be changed on the fly, the data field can be filled with any number of bits. When used in conjunction with the Receiver Residue codes, the Z80-SIO can receive a message that has a variable I-field and retransmit it exactly as received with no previous information about the character structure of the I-field (if any). A change in the number of bits does not affect the character in the process of being shifted out. Characters are sent with the number of bits programmed at the time that the character is loaded from the transmit buffer to the transmitter.

If the External/Status Interrupt Enable is set, transmitter conditions such as "starting to send CRC characters," "starting to send flag characters," and $\overline{\text{CTS}}$ changing state cause interrupts that have a unique vector if Status Affects Vector is set. All interrupts can be disabled for operation in a polled mode.

Table 8 shows the typical program steps that implement the half-duplex SDLC Transmit mode.

FUNCTION		TYPICAL PROGRAM STEPS	COMMENTS
INITIALIZE	*REGISTER:*	*INFORMATION LOADED:*	
	WR0	CHANNEL RESET	Reset SIO.
	WR0	POINTER 2	
	WR2	INTERRUPT VECTOR	Channel B only
	WR0	POINTER 3	
	WR3	AUTO ENABLES	Transmitter sends data only after $\overline{CTS}$ is detected.
	WR0	POINTER 4, RESET EXTERNAL STATUS INTERRUPTS	
	WR4	PARITY INFORMATION, SDLC MODE, ×1 CLOCK MODE	
	WR0	*POINTER 1, RESET EXTERNAL/STATUS INTERRUPTS*	
	WR1	EXTERNAL INTERRUPT ENABLE, STATUS AFFECTS VECTOR, TRANSMIT INTERRUPT ENABLE *OR* WAIT/READY MODE ENABLE	The External Interrupt mode monitors the status of the $\overline{CTS}$ and $\overline{DCD}$ inputs, as well *as the status of Tx Underrun/EOM latch.* Transmit Interrupt interrupts when the Transmit buffer becomes empty; the Wait/Ready mode can be used to transfer data on a DMA or Block Transfer basis. The first interrupt occurs when $\overline{CTS}$ becomes active, at which point flags are transmitted by the Z80-SIO. The first data byte (address field) can be loaded in the Z80-SIO after this interrupt. Flags cannot be sent to the Z80-SIO as data. Status Affects Vector used in Channel B only.
	WR0	POINTER 5	
	WR5	TRANSMIT CRC ENABLE, REQUEST TO SEND, SDLC-CRC, TRANSMIT ENABLE, TRANSMIT WORD LENGTH, DATA TERMINAL READY	SDLC-CRC mode *must be defined before* initializing transmit CRC generator.
	WR0	RESET TRANSMIT CRC GENERATOR	Initialize CRC generator to all 1's.
IDLE MODE		*EXECUTE HALT INSTRUCTION OR SOME OTHER PROGRAM*	Waiting for Interrupt or Wait/Ready output to transfer data.
DATA TRANSFER AND STATUS MONITORING		*WHEN INTERRUPT (WAIT/READY) OCCURS, THE CPU DOES THE FOLLOWING:* • CHANGES TRANSMIT WORD LENGTH (IF NECESSARY) • TRANSFERS DATA BYTE FROM CPU (MEMORY) TO SIO • RESETS Tx UNDERRUN/EOM LATCH (WR0)	Flags are transmitted by the SIO as soon as Transmit Enable is set and $\overline{CTS}$ becomes active. The $\overline{CTS}$ status change is the first interrupt that occurs and is followed by transmit buffer empty for subsequent transfers.
		IF LAST CHARACTER OF THE I-FIELD IS SENT, THE SIO DOES THE FOLLOWING: • SENDS CRC • SENDS CLOSING FLAG • INTERRUPTS CPU WITH BUFFER EMPTY STATUS *CPU DOES THE FOLLOWING:* • ISSUES RESET Tx INTERRUPT PENDING COMMAND TO THE Z80-SIO • UPDATES NS COUNT • REPEATS THE PROCESS FOR NEXT MESSAGE, ETC. *IF THE VECTOR INDICATES AN ERROR, THE CPU DOES THE FOLLOWING:* • SENDS ABORT • EXECUTES ERROR ROUTINE • UPDATES PARAMETERS, MODES, ETC. • RETURNS FROM INTERRUPT	Word length can be changed "on the fly" *for variable* I-field length. The data byte can contain address, control, or I-field information (never a flag). It is a good practice to reset Tx Underrun/EOM latch *in the beginning of the message to avoid a* false end-of-frame detection at the receiving end. This ensures that, when underrun occurs, CRC is transmitted and underrun interrupt (Tx Underrun/EOM latch active) occurs. Note that "Send Abort" can be issued to the SIO in response to any interrupting continuing to abort the transmission.
TERMINATION		REDEFINE INTERRUPT MODES	Terminate gracefully.
		UPDATE MODEM CONTROL OUTPUTS	
		DISABLE TRANSMIT MODE	

Table 8. SDLC Transmit Mode

SDLC Receive

INITIALIZATION

The SDLC Receive mode is initialized by the system with the following parameters: SDLC mode, × 1 clock mode, SDLC polynomial, receive word length, etc. The flag characters must also be loaded in WR7 and the secondary address field loaded in WR6. The receiver is enabled only after all the receive parameters have been set. After all this has been done, the receiver is in the Hunt phase and remains in this phase until the first flag is received. While in the SDLC mode, the receiver never re-enters the Hunt phase, unless specifically instructed to do so by the program. The WR4 parameters must be issued prior to the WR1, WR3, WR5, WR6 and WR7 parameters.

Under program control, the receiver can enter the Address Search mode. If the Address Search bit (WR3, D_2) is set, a character following the flag (first non-flag character) is compared against the programmed address in WR6 and the hardwired global address (11111111). If the SDLC frame address field matches either address, data transfer begins.

Since the Z80-SIO is capable of matching only one address character, extended address field recognition must be done by the CPU. In this case, the Z80-SIO simply transfers the additional address bytes to the CPU as if they were data characters. If the CPU determines that the frame does not have the correct address field, it can set the Hunt bit, and the Z80-SIO suspends reception and searches for a new message headed by a flag. Since the control field of the frame is transparent to the Z80-SIO, it is transferred to the CPU as a data character. Extra zeros inserted in the data stream are automatically deleted; flags are not transferred to the CPU.

DATA TRANSFER AND STATUS MONITORING

After receipt of a valid flag, the assembled characters are transferred to the receive data FIFO. The following four interrupt modes are available to transfer this data and its associated status.

No Interrupts Enabled. This mode is used for purely polled operations or for off-line conditions.

Interrupt On First Character Only. This mode is normally used to start a software polling loop or a Block Transfer instruction using $\overline{\text{WAIT}}/\text{READY}$ to synchronize the CPU or DMA device to the incoming data rate. In this mode, the Z80-SIO interrupts on the first character and thereafter only interrupts if Special Receive conditions are detected. The mode is reinitialized with the Enable Interrupt On Next Receive Character Command.

The first character received after this command is issued causes an interrupt. If External/Status interrupts are enabled, they may interrupt any time the $\overline{\text{DCD}}$ input changes state. Special Receive conditions such as End Of Frame and Receiver Overrun also cause interrupts. The End Of Frame interrupt can be used to exit the Block Transfer mode.

Interrupt On Every Character. An interrupt is generated whenever the receive FIFO contains a character. Error and Special Receive conditions generate a special vector if Status Affects Vector is selected.

Special Receive Condition Interrupts. The Special Receive Condition interrupt is not, as such, a separate interrupt mode. Before the Special Receive condition can cause an interrupt, either Interrupt On First Receive Character Only or Interrupt On Every Character must be selected. The Special Receive Condition interrupt is caused by a Receive Overrun or End Of Frame detection. Since the Receive Overrun status bit is latched, the error status read reflects an error in the current word in the receive buffer in addition to any errors received since the last Error Reset command. The Receive Overrun status bit can only be reset by the Error Reset command. The End Of Frame status bit indicates that a valid ending flag has been received and that the CRC Error and Residue codes are also valid.

Character length may be changed on the fly. If the address and control bytes are processed as 8-bit characters, the receiver may be switched to a shorter character length during the time that the first information character is being assembled. This change must be made fast enough so it is effective before the number of bits specified for the character length have been assembled. For example, if the change is to be from the 8-bit control field to a 7-bit information field, the change must be made *before* the first seven bits of the I-field are assembled.

SDLC Receive CRC Checking. Control of the receive CRC checker is automatic. It is reset by the leading flag and CRC is calculated up to the final flag. The byte that has the End Of Frame bit set is the byte that contains the result of the CRC check. If the CRC/Framing Error bit is not set, the CRC indicates a valid message. A special check sequence is used for the SDLC check because the transmitted CRC check is inverted. The final check must be 0001110100001111. The 2-byte CRC check characters must be read by the CPU and discarded because the Z80-SIO, while using them for CRC checking, treats them as ordinary data.

SDLC Receive Termination. If enabled, a special vector is generated when the closing flag is received. This signals that the byte with the End Of Frame bit set has been received. In addition to the results of the CRC check, RR1 has three bits of Residue code valid at this time. For those cases in which the number of bits in the I-field is not an integral multiple of the character length used, these bits indicate the boundary between the CRC check bits and the I-field bits. For a detailed description of the meaning of these bits, see the description of the residue codes in RR1 under "Z80-SIO Programming."

Any frame can be prematurely aborted by an Abort sequence. Aborts are detected if seven or more 1's occur and cause an External/Status interrupt (if enabled) with the Break/Abort bit in RR0 set. After the Reset External/Status interrupts command has been issued a second interrupt occurs when the continuous 1's condition has been cleared. This can be used to distinguish between the Abort and Idle line conditions.

Unlike the synchronous mode, CRC calculation in SDLC does not have an 8-bit delay since all the characters are included in CRC calculation. When the second CRC character is loaded into the receive buffer, CRC calculation is complete.

Table 9 shows the typical steps required to implement a half-duplex SDLC receive mode. The complete set of command and status bit definitions is found in the next section.

FUNCTION		TYPICAL PROGRAM STEPS	COMMENTS
INITIALIZE	*REGISTER:*	*INFORMATION LOADED:*	
	WR0	CHANNEL 2	Reset SIO
	WR0	POINTER 2	
	WR2	INTERRUPT VECTOR	Channel B only
	WR0	POINTER 4	
	WR4	PARITY INFORMATION, SYNC MODE, SDLC MODE, x1 CLOCK MODE	
	WR0	POINTER 5, RESET EXTERNAL/STATUS INTERRUPTS	
	WR5	SDLC-CRC, DATA TERMINAL READY	
	WR0	POINTER 3	
	WR3	RECEIVE CRC ENABLE, ENTER HUNT MODE, AUTO ENABLES, RECEIVE CHARACTER LENGTH, ADDRESS SEARCH MODE	'Auto Enables' enables the receiver to accept data only after $\overline{DCD}$ becomes active. Address Search Mode enables SIO to match the message address with the programmed address or the global address.
	WR0	POINTER 6	
	WR6	SECONDARY ADDRESS FIELD	This address is matched against the message address in an SDLC poll operation.
	WR0	POINTER 7	
	WR7	SDLC FLAG 01111110	This flag detects the start and end of frame in an SDLC operation.
	WR0	POINTER 1, RESET EXTERNAL/STATUS INTERRUPTS	In this interrupt mode, only the Address Field (1 character only) is transferred to the CPU. All subsequent fields (Control, Information, etc.) are transferred on a DMA basis. Status Affects Vector in Channel B only.
	WR1	STATUS AFFECTS VECTOR, EXTERNAL INTERRUPT ENABLE, RECEIVE INTERRUPT ON FIRST CHARACTER ONLY.	
	WR0	POINTER 3, ENABLE INTERRUPT ON NEXT RECEIVE CHARACTER	Used to provide simple loop-back entry point for next transaction.
	WR3	RECEIVE ENABLE, RECEIVE CRC ENABLE, ENTER HUNT MODE, AUTO ENABLES, RECEIVER CHARACTER LENGTH, ADDRESS SEARCH MODE	WR3 reissued to enable receiver.
IDLE MODE		EXECUTE HALT INSTRUCTION OR SOME OTHER PROGRAM	SDLC Receive Mode is fully initialized and SIO is waiting for the opening flag followed by a matching address field to interrupt the CPU.

Table 9. SDLC Receive Mode

FUNCTION	TYPICAL PROGRAM STEPS	COMMENTS
	WHEN INTERRUPT ON FIRST CHARACTER OCCURS, THE CPU DOES THE FOLLOWING: • TRANSFERS DATA BYTE (ADDRESS BYTE) TO CPU • DETECTS AND SETS APPROPRIATE FLAG FOR EXTENDED ADDRESS FIELD • UPDATES POINTERS AND PARAMETERS • ENABLES DMA CONTROLLER • ENABLES WAIT/READY FUNCTION IN SIO • RETURNS FROM INTERRUPT	During the Hunt phase, the SIO interrupts when the programmed address matches the message address. The CPU establishes the DMA mode and all subsequent data characters are transferred by the DMA controller to memory.
	WHEN THE READY OUTPUT BECOMES ACTIVE, THE DMA CONTROLLER DOES THE FOLLOWING: • TRANSFERS THE DATA BYTE TO MEMORY • UPDATES THE POINTERS	During the DMA operation, the SIO monitors the $\overline{DCD}$ input and the Abort sequence in the data stream to interrupt the CPU with External Status error. The Special Receive condition interrupt is caused by Receive Overrun error.
DATA TRANSFER AND STATUS MONITORING	*WHEN END OF FRAME INTERRUPT OCCURS, THE CPU DOES THE FOLLOWING:* • EXITS DMA MODE (DISABLES WAIT/READY) • TRANSFERS RR1 TO THE CPU • CHECKS THE CRC ERROR BIT STATUS AND RESIDUE CODES • UPDATES NR COUNT • ISSUES 'ERROR RESET' COMMAND TO SIO	Detection of End of Frame (Flag) causes interrupt and deactivates the Wait/Ready function. Residue codes indicate the bit structure of the last two bytes of the message, which were transferred to memory under DMA. 'Error Reset' is issued to clear the special condition.
	WHEN 'ABORT SEQUENCE DETECTED' INTERRUPT OCCURS, THE CPU DOES THE FOLLOWING: • TRANSFERS RR0 TO THE CPU • EXITS DMA MODE • ISSUES THE RESET EXTERNAL STATUS INTERRUPT COMMAND TO THE SIO	Abort sequence is detected when seven or more 1's are found in the data stream.
	• ENTERS THE IDLE MODE	CPU is waiting for Abort Sequence to terminate. Termination clears the Break/Abort status bit and causes interrupt.
	WHEN THE SECOND ABORT SEQUENCE INTERRUPT OCCURS, THE CPU DOES THE FOLLOWING: • ISSUES THE RESET EXTERNAL STATUS INTERRUPT COMMAND TO THE SIO	At this point, the program proceeds to terminate this message.
TERMINATION	REDEFINE INTERRUPT MODES, SYNC MODE AND SDLC MODES DISABLE RECEIVE MODE	

Table 9. SDLC Receive Mode (Continued)

Z80-SIO Programming

To program the Z80-SIO, the system program first issues a series of commands that initialize the basic mode of operation and then other commands that qualify conditions within the selected mode. For example, the Asynchronous mode, character length, clock rate, number of stop bits, even or odd parity are first set, then the interrupt mode and, finally, receiver or transmitter enable. The WR4 parameters must be issued before any other parameters are issued in the initialization routine.

Both channels contain command registers that must be programmed via the system program prior to operation. The Channel Select input (B/$\overline{A}$) and the Control/Data input (C/$\overline{D}$) are the command structure addressing controls, and are normally controlled by the CPU address bus. Figure 14 illustrates the timing relationships for programming the write registers, and transferring data and status.

C/$\overline{D}$	B/$\overline{A}$	Function
0	0	Channel A Data
0	1	Channel B Data
1	0	Channel A Commands/Status
1	1	Channel B Commands/Status

Write Registers

The Z80-SIO contains eight registers (WR0-WR7) in each channel that are programmed separately by the system program to configure the functional personality of the channels. With the exception of WR0, programming the write registers requires two bytes. The first byte contains three bits (D_0-D_2) that point to the selected register; the second byte is the actual control word that is written into the register to configure the Z80-SIO.

Note that the programmer has complete freedom, after pointing to the selected register, of either reading to test the read register or writing to initialize the write register. By designing software to initialize the Z80-SIO in a modular and structured fashion, the programmer can use powerful block I/O instructions.

WR0 is a special case in that all the basic commands (CMD_0-CMD_2) can be accessed with a single byte. Reset (internal or external) initializes the pointer bits D_0-D_2 to point to WR0.

The basic commands (CMD_0-CMD_2) and the CRC controls (CRC_0, CRC_1) are contained in the first byte of any write register access. This maintains maximum flexibility and system control. Each channel contains the following control registers. These registers are addressed as commands (not data).

WRITE REGISTER 0

WR0 is the command register; however, it is also used for CRC reset codes and to point to the other registers.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CRC Reset Code 1	CRC Reset Code 0	CMD 2	CMD 1	CMD 0	PTR 2	PTR 1	PTR 0

Pointer Bits (D_0-D_2). Bits D_0-D_2 are pointer bits that determine which other write register the next byte is to be written into or which read register the next byte is to be read from. The first byte written into each channel after a reset (either by a Reset command or by the external reset input) goes into WR0. Following a read or write to any register (except WR0), the pointer will point to WR0.

Command Bits (D_3-D_5). Three bits, D_3-D_5, are encoded to issue the seven basic Z80-SIO commands.

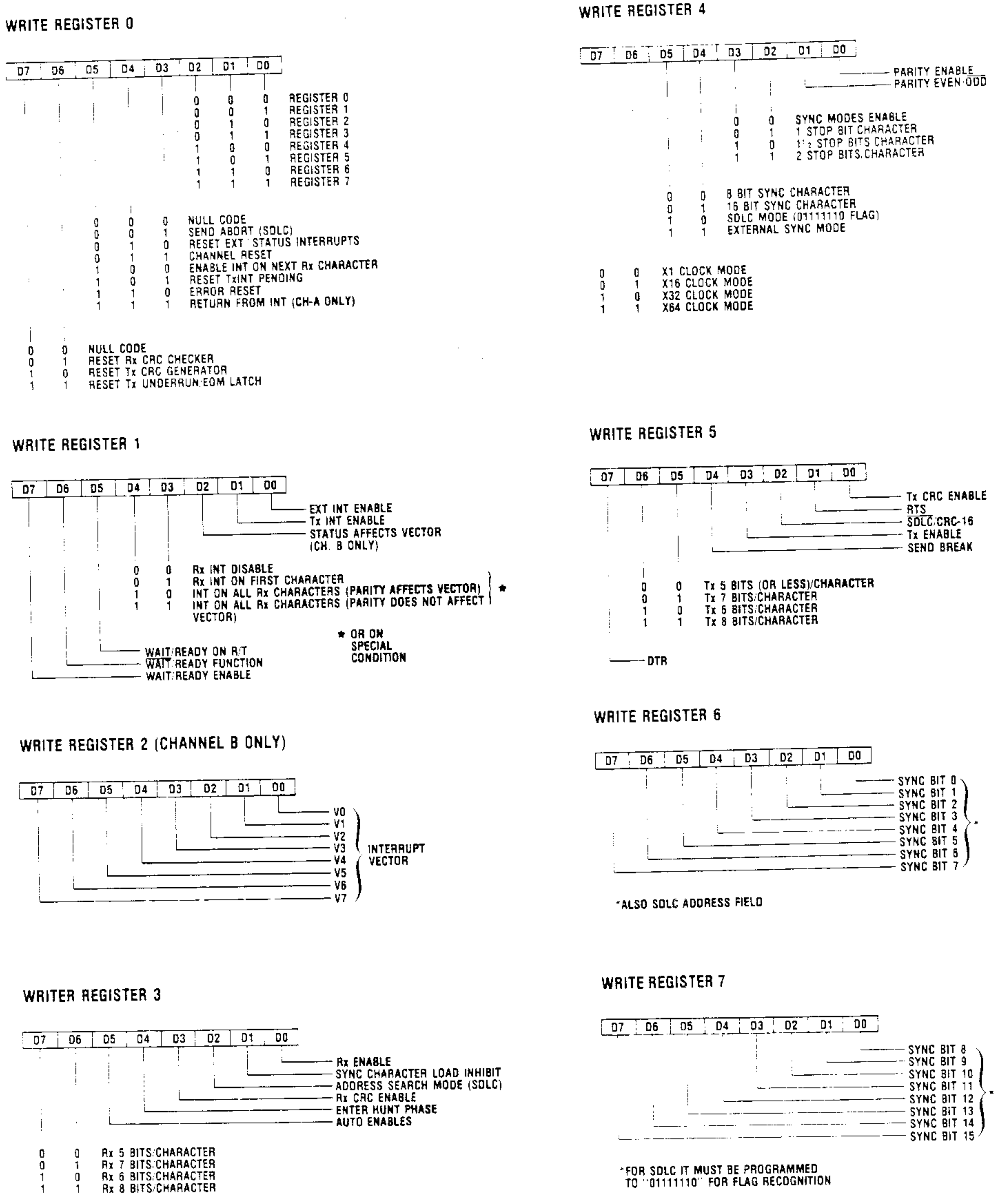
Command	CMD_2	CMD_1	CMD_0	
0	0	0	0	Null Command (no effect)
1	0	0	1	Send Abort (SDLC Mode)
2	0	1	0	Reset External/Status Interrupts
3	0	1	1	Channel Reset
4	1	0	0	Enable Interrupt on next Rx Character
5	1	0	1	Reset Transmitter Interrupt Pending
6	1	1	0	Error Reset (latches)
7	1	1	1	Return from Interrupt (Channel A)

Command 0 (Null). The Null command has no effect. Its normal use is to cause the Z80-SIO to do nothing while the pointers are set for the following byte.

Command 1 (Send Abort). This command is used only with the SDLC mode to generate a sequence of eight to thirteen 1's.

Command 2 (Reset External/Status Interrupts). After an External/Status interrupt (a change on a modem line or a break condition, for example), the status bits of RR0 are latched. This command re-enables them and allows interrupts to occur again. Latching the status bits captures short pulses until the CPU has time to read the change.

Command 3 (Channel Reset). This command performs the same function as an External Reset, but only on a single channel. Channel A Reset also resets the interrupt prioritization logic. All control registers for the channel must be rewritten after a Channel Reset command.



WRITE REGISTER 0

D7 D6 D5 D4 D3 D2 D1 D0

D2	D1	D0	
0	0	0	REGISTER 0
0	0	1	REGISTER 1
0	1	0	REGISTER 2
0	1	1	REGISTER 3
1	0	0	REGISTER 4
1	0	1	REGISTER 5
1	1	0	REGISTER 6
1	1	1	REGISTER 7

D5	D4	D3	
0	0	0	NULL CODE
0	0	1	SEND ABORT (SDLC)
0	1	0	RESET EXT/STATUS INTERRUPTS
0	1	1	CHANNEL RESET
1	0	0	ENABLE INT ON NEXT Rx CHARACTER
1	0	1	RESET TxINT PENDING
1	1	0	ERROR RESET
1	1	1	RETURN FROM INT (CH-A ONLY)

D7	D6	
0	0	NULL CODE
0	1	RESET Rx CRC CHECKER
1	0	RESET Tx CRC GENERATOR
1	1	RESET Tx UNDERRUN/EOM LATCH

WRITE REGISTER 1

D7 D6 D5 D4 D3 D2 D1 D0

- D0 — EXT INT ENABLE
- D1 — Tx INT ENABLE
- D2 — STATUS AFFECTS VECTOR (CH. B ONLY)

D4	D3	
0	0	Rx INT DISABLE
0	1	Rx INT ON FIRST CHARACTER
1	0	INT ON ALL Rx CHARACTERS (PARITY AFFECTS VECTOR) *
1	1	INT ON ALL Rx CHARACTERS (PARITY DOES NOT AFFECT VECTOR) *

* OR ON SPECIAL CONDITION

- D5 — WAIT/READY ON R/T
- D6 — $\overline{\text{WAIT}}$/READY FUNCTION
- D7 — WAIT/READY ENABLE

WRITE REGISTER 2 (CHANNEL B ONLY)

D7 D6 D5 D4 D3 D2 D1 D0

- D0 — V0
- D1 — V1
- D2 — V2
- D3 — V3
- D4 — V4
- D5 — V5
- D6 — V6
- D7 — V7

V0–V7: INTERRUPT VECTOR

WRITER REGISTER 3

D7 D6 D5 D4 D3 D2 D1 D0

- D0 — Rx ENABLE
- D1 — SYNC CHARACTER LOAD INHIBIT
- D2 — ADDRESS SEARCH MODE (SDLC)
- D3 — Rx CRC ENABLE
- D4 — ENTER HUNT PHASE
- D5 — AUTO ENABLES

D7	D6	
0	0	Rx 5 BITS/CHARACTER
0	1	Rx 7 BITS/CHARACTER
1	0	Rx 6 BITS/CHARACTER
1	1	Rx 8 BITS/CHARACTER

WRITE REGISTER 4

D7 D6 D5 D4 D3 D2 D1 D0

- D0 — PARITY ENABLE
- D1 — PARITY EVEN/$\overline{\text{ODD}}$

D3	D2	
0	0	SYNC MODES ENABLE
0	1	1 STOP BIT/CHARACTER
1	0	1½ STOP BITS/CHARACTER
1	1	2 STOP BITS/CHARACTER

D5	D4	
0	0	8 BIT SYNC CHARACTER
0	1	16 BIT SYNC CHARACTER
1	0	SDLC MODE (01111110 FLAG)
1	1	EXTERNAL SYNC MODE

D7	D6	
0	0	X1 CLOCK MODE
0	1	X16 CLOCK MODE
1	0	X32 CLOCK MODE
1	1	X64 CLOCK MODE

WRITE REGISTER 5

D7 D6 D5 D4 D3 D2 D1 D0

- D0 — Tx CRC ENABLE
- D1 — $\overline{\text{RTS}}$
- D2 — $\overline{\text{SDLC}}$/CRC-16
- D3 — Tx ENABLE
- D4 — SEND BREAK

D6	D5	
0	0	Tx 5 BITS (OR LESS)/CHARACTER
0	1	Tx 7 BITS/CHARACTER
1	0	Tx 6 BITS/CHARACTER
1	1	Tx 8 BITS/CHARACTER

- D7 — DTR

WRITE REGISTER 6

D7 D6 D5 D4 D3 D2 D1 D0

- D0 — SYNC BIT 0
- D1 — SYNC BIT 1
- D2 — SYNC BIT 2
- D3 — SYNC BIT 3
- D4 — SYNC BIT 4
- D5 — SYNC BIT 5
- D6 — SYNC BIT 6
- D7 — SYNC BIT 7

SYNC BIT 0–7: *

*ALSO SDLC ADDRESS FIELD

WRITE REGISTER 7

D7 D6 D5 D4 D3 D2 D1 D0

- D0 — SYNC BIT 8
- D1 — SYNC BIT 9
- D2 — SYNC BIT 10
- D3 — SYNC BIT 11
- D4 — SYNC BIT 12
- D5 — SYNC BIT 13
- D6 — SYNC BIT 14
- D7 — SYNC BIT 15

SYNC BIT 8–15: *

*FOR SDLC IT MUST BE PROGRAMMED TO "01111110" FOR FLAG RECOGNITION

Figure 9. Write Register Bit Functions

After a Channel Reset, four extra system clock cycles should be allowed for Z80-SIO reset time before any additional commands or controls are written into that channel. This can normally be the time used by the CPU to fetch the next op code.

Command 4 (Enable Interrupt On Next Receive Character). If the Interrupt On First Receive Character mode is selected, this command reactivates that mode after each complete message is received to prepare the Z80-SIO for the next message.

Command 5 (Reset Transmitter Interrupt Pending). The transmitter interrupts when the transmit buffer becomes empty if the Transmit Interrupt Enable mode is selected. In those cases where there are no more characters to be sent (at the end of message, for example), issuing this command prevents further transmitter interrupts until after the next character has been loaded into the transmit buffer or until CRC has been completely sent.

Command 6 (Error Reset). This command resets the error latches. Parity and Overrun errors are latched in RR1 until they are reset with this command. With this scheme, parity errors occurring in block transfers can be examined at the end of the block.

Command 7 (Return From Interrupt). This command must be issued in Channel A and is interpreted by the Z80-SIO in exactly the same way it would interpret an RETI command on the data bus. It resets the interrupt-under-service latch of the highest-priority internal device under service and thus allows lower priority devices to interrupt via the daisy chain. This command allows use of the internal daisy chain even in systems with no external daisy chain or RETI command.

CRC Reset Codes 0 and 1 (D_6 and D_7). Together, these bits select one of the three following reset commands:

CRC Reset Code 1	CRC Reset Code 0	
0	0	Null Code (no affect)
0	1	Reset Receive CRC Checker
1	0	Reset Transmit CRC Generator
1	1	Reset Tx Underrun/End Of Message latch

The Reset Transmit CRC Generator command normally initializes the CRC generator to all 0's. If the SDLC mode is selected, this command initializes the CRC generator to all 1's. The Receive CRC checker is also initialized to all 1's for the SDLC mode.

WRITE REGISTER 1

WR1 contains the control bits for the various interrupt and Wait/Ready modes.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Wait/Ready Enable	$\overline{\text{Wait}}$ Or Ready Function	Wait/Ready On Receive/ Transmit	Receive Interrupt Mode 1	Receive Interrupt Mode 0	Status Affects Vector	Transmit Interrupt Enable	External Interrupts Enable

External/Status Interrupt Enable (D_0). The External/Status Interrupt Enable allows interrupts to occur as a result of transitions on the $\overline{\text{DCD}}$, $\overline{\text{CTS}}$ or $\overline{\text{SYNC}}$ inputs, as a result of a Break/Abort detection and termination, or at the beginning of CRC or sync character transmission when the Transmit Underrun/EOM latch becomes set.

Transmitter Interrupt Enable (D_1). If enabled, interrupts occur whenever the transmitter buffer becomes empty.

Status Affects Vector (D_2). This bit is active in Channel B only. If this bit is not set, the fixed vector programmed in WR2 is returned from an interrupt acknowledge sequence. If this bit is set, the vector returned from an interrupt acknowledge is variable according to the following interrupt conditions:

	V_3	V_2	V_1	
Ch B	0	0	0	Ch B Transmit Buffer Empty
	0	0	1	Ch B External/Status Change
	0	1	0	Ch B Receive Character Available
	0	1	1	Ch B Special Receive Condition*
Ch A	1	0	0	Ch A Transmit Buffer Empty
	1	0	1	Ch A External/Status Change
	1	1	0	Ch A Receive Character Available
	1	1	1	Ch A Special Receive Condition*

*Special Receive Conditions: Parity Error, Rx Overrun Error, Framing Error, End Of Frame (SDLC).

Receive Interrupt Modes 0 and 1 (D_3 and D_4). Together these two bits specify the various character-available conditions. In Receive Interrupt modes 1, 2 and 3, a Special Receive Condition can cause an interrupt and modify the interrupt vector.

D_4 Receive Interrupt Mode 1	D_3 Receive Interrupt Mode 0	
0	0	0. Receive Interrupts Disabled
0	1	1. Receive Interrupt On First Character Only
1	0	2. Interrupt On All Receive Characters—parity error is a Special Receive condition
1	1	3. Interrupt On All Receive Characters—parity error is not a Special Receive condition

Wait/Ready Function Selection (D_5-D_7). The Wait and Ready functions are selected by controlling D_5, D_6, and D_7. Wait/Ready function is enabled by setting Wait/Ready Enable (WR1, D_7) to 1. The Ready function is selected by setting D_6 (Wait/Ready function) to 1. If this bit is 1, the $\overline{\text{WAIT/READY}}$ output switches from High to Low when the Z80-SIO is ready to transfer data. The Wait function is selected by setting D_6 to 0. If this bit is

0, the $\overline{\text{WAIT}}/\overline{\text{READY}}$ output is in the open-drain state and goes Low when active.

Both the Wait and Ready functions can be used in either the Transmit or Receive modes, but not both simultaneously. If D_5 (Wait/Ready on Receive/Transmit) is set to 1, the Wait/Ready function responds to the condition of the receive buffer (empty or full). If D_5 is set to 0, the Wait/Ready function responds to the condition of the transmit buffer (empty or full).

The logic states of the $\overline{\text{WAIT}}/\overline{\text{READY}}$ output when active or inactive depend on the combination of modes selected. Following is a summary of these combinations:

If $D_7 = 0$

And $D_6 = 1$	And $D_6 = 0$
$\overline{\text{READY}}$ is High	$\overline{\text{WAIT}}$ is floating

If $D_7 = 1$

And $D_5 = 0$		And $D_5 = 1$	
$\overline{\text{READY}}$	Is High when transmit buffer is full.	$\overline{\text{READY}}$	Is High when receive buffer is empty.
$\overline{\text{WAIT}}$	Is Low when transmit buffer is full and an SIO data port is selected.	$\overline{\text{WAIT}}$	Is Low when receive buffer is empty and an SIO data port is selected.
$\overline{\text{READY}}$	Is Low when transmit buffer is empty.	$\overline{\text{READY}}$	Is Low when receive buffer is full.
$\overline{\text{WAIT}}$	Is floating when transmit buffer is empty.	$\overline{\text{WAIT}}$	Is floating when receive buffer is full.

The $\overline{\text{WAIT}}$ output High-to-Low transition occurs with the delay time $t_{DIC(WR)}$ after the I/O request. The Low-to-High transition occurs with the delay $t_{DH\phi(WR)}$ from the falling edge of ϕ. The $\overline{\text{READY}}$ output High-to-Low transition occurs with the delay $t_{DL\phi(WR)}$ from the rising edge of ϕ. The $\overline{\text{READY}}$ output Low-to-High transition occurs with the delay $t_{DIC(WR)}$ after $\overline{\text{IORQ}}$ falls.

The Ready function can occur any time the Z80-SIO is not selected. When the $\overline{\text{READY}}$ output becomes active (Low), the DMA controller issues $\overline{\text{IORQ}}$ and the corresponding $B/\overline{A}$ and $C/\overline{D}$ inputs to the Z80-SIO to transfer data. The $\overline{\text{READY}}$ output becomes inactive as soon as $\overline{\text{IORQ}}$ and $\overline{\text{CS}}$ become active. Since the Ready function can occur internally in the Z80-SIO whether it is addressed or not, the $\overline{\text{READY}}$ output becomes inactive when any CPU data or command transfer takes place. This does not cause problems because the DMA controller is not enabled when the CPU transfer takes place.

The Wait function—on the other hand—is active only if the CPU attempts to read Z80-SIO data that has not yet been received, which occurs frequently when block transfer instructions are used. The Wait function can also become active (under program control) if the CPU tries to write data while the transmit buffer is still full. The fact that the $\overline{\text{WAIT}}$ output for either channel can become active when the opposite channel is addressed (because the Z80-SIO is addressed) does not affect operation of software loops or block move instructions.

WRITE REGISTER 2

WR2 is the interrupt vector register; it exists in Channel B only. V_4-V_7 and V_0 are always returned exactly as written; V_1-V_3 are returned as written if the Status Affects Vector (WR1, D_2) control bit is 0. If this bit is 1, they are modified as explained in the previous section.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
V_7	V_6	V_5	V_4	V_3	V_2	V_1	V_0

WRITE REGISTER 3

WR3 contains receiver logic control bits and parameters.

D_7	D_6	D_5	D_4
Receiver Bits/ Char 1	Receiver Bits/ Char 0	Auto Enables	Enter Hunt Phase

D_3	D_2	D_1	D_0
Receiver CRC Enable	Address Search Mode	Sync Char Load Inhibit	Receiver Enable

Receiver Enable (D_0). A 1 programmed into this bit allows receive operations to begin. This bit should be set only after all other receive parameters are set and receiver is completely initialized.

Sync Character Load Inhibit (D_1). Sync characters preceding the message (leading sync characters) are not loaded into the receive buffers if this option is selected. Because CRC calculations are not stopped by sync character stripping, this feature should be enabled only at the beginning of the message.

Address Search Mode (D_2). If SDLC is selected, setting this mode causes messages with addresses not matching the programmed address in WR6 or the global (11111111) address to be rejected. In other words, no receive interrupts can occur in the Address Search mode unless there is an address match.

Receiver CRC Enable (D_3). If this bit is set, CRC calculation starts (or restarts) at the beginning of the last character transferred from the receive shift register to the buffer stack, regardless of the number of characters in the stack. See "SDLC Receive CRC Checking" (SDLC Receive section) and "CRC Error Checking" (Synchronous Receive section) for details regarding when this bit should be set.

Enter Hunt Phase (D_4). The Z80-SIO automatically enters the Hunt phase after a reset; however, it can be re-entered if character synchronization is lost for any reason (Synchronous mode) or if the contents of an incoming message are not needed (SDLC mode). The Hunt phase is re-entered by writing a 1 into bit D_4. This sets the Sync/Hunt bit (D_4) in RR0.

Auto Enables (D_5). If this mode is selected, $\overline{\text{DCD}}$ and $\overline{\text{CTS}}$ become the receiver and transmitter enables, respectively. If this bit is not set, $\overline{\text{DCD}}$ and $\overline{\text{CTS}}$ are simply inputs to their corresponding status bits in RR0.

Receiver Bits/Characters 1 and 0 (D_7 and D_6). Together, these bits determine the number of serial receive bits assembled to form a character. Both bits may be changed during the time that a character is being assembled, but they must be changed before the number of bits currently programmed is reached.

D_7	D_6	Bits/Character
0	0	5
0	1	7
1	0	6
1	1	8

WRITE REGISTER 4

WR4 contains the control bits that affect both the receiver and transmitter. In the transmit and receive initialization routine, these bits should be set before issuing WR1, WR3, WR5, WR6, and WR7.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Clock Rate 1	Clock Rate 0	Sync Modes 1	Sync Modes 0	Stop Bits 1	Stop Bits 0	Parity Even/$\overline{\text{Odd}}$	Parity

Parity (D_0). If this bit is set, an additional bit position (in addition to those specified in the bits/character control) is added to transmitted data and is expected in receive data. In the Receive mode, the parity bit received is transferred to the CPU as part of the character, unless 8 bits/character is selected.

Parity Even/$\overline{\text{Odd}}$ (D_1). If parity is specified, this bit determines whether it is sent and checked as even or odd (1 = even).

Stop Bits 0 and 1 (D_2 and D_3). These bits determine the number of stop bits added to each asynchronous character sent. The receiver always checks for one stop bit. A special mode (00) signifies that a synchronous mode is to be selected.

D_3 Stop Bits 1	D_2 Stop Bits 0	
0	0	Sync modes
0	1	1 stop bit per character
1	0	1½ stop bits per character
1	1	2 stop bits per character

Sync Modes 0 and 1 (D_4 and D_5). These bits select the various options for character synchronization.

Sync Mode 1	Sync Mode 0	
0	0	8-bit programmed sync
0	1	16-bit programmed sync
1	0	SDLC mode (01111110 flag pattern)
1	1	External Sync mode

Clock Rate 0 and 1 (D_6 and D_7). These bits specify the multiplier between the clock ($\overline{\text{TxC}}$ and $\overline{\text{RxC}}$) and data rates. For synchronous modes, the × 1 clock rate must be specified. Any rate may be specified for asynchronous modes; however, the same rate must be used for both the receiver and transmitter. The system clock in all modes must be at least 4.5 times the data rate. If the × 1 clock rate is selected, bit synchronization must be accomplished externally.

Clock Rate 1	Clock Rate 0	
0	0	Data Rate × 1 = Clock Rate
0	1	Data Rate × 16 = Clock Rate
1	0	Data Rate × 32 = Clock Rate
1	1	Data Rate × 64 = Clock Rate

WRITE REGISTER 5

WR5 contains control bits that affect the operation of transmitter, with the exception of D_2, which affects the transmitter and receiver.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
DTR	Tx Bits/ Char 1	Tx Bits/ Char 0	Send Break	Tx Enable	CRC-16/ $\overline{\text{SDLC}}$	RTS	Tx CRC Enable

Transmit CRC Enable (D_0). This bit determines if CRC is calculated on a particular transmit character. If it is set at the time the character is loaded from the transmit buffer into the transmit shift register, CRC is calculated on the character. CRC is not automatically sent unless this bit is set when the Transmit Underrun condition exists.

Request To Send (D_1). This is the control bit for the $\overline{\text{RTS}}$ pin. When the $\overline{\text{RTS}}$ bit is set, the $\overline{\text{RTS}}$ pin goes Low; when reset, $\overline{\text{RTS}}$ goes High. In the Asynchronous mode, $\overline{\text{RTS}}$ goes High only after all the bits of the character are transmitted and the transmitter buffer is empty. In Synchronous modes, the pin directly follows the state of the bit.

CRC-16/$\overline{\text{SDLC}}$ (D_2). This bit selects the CRC polynomial used by both the transmitter and receiver. When set, the CRC-16 polynomial ($X^{16} + X^{15} + X^2 + 1$) is used; when reset the SDLC polynomial ($X^{16} + X^{12} + X^5 + 1$) is used. If the SDLC mode is selected, the CRC generator and checker are preset to all 1's and a special check sequence is used. The SDLC CRC polnomial must be selected when the SDLC mode is selected. If the SDLC mode is not selected, the CRC generator and checker are preset to all 0's (for both polynomials).

Transmit Enable (D_3). Data is not transmitted until this bit is set, and the Transmit Data output is held marking. Data or sync characters in the process of being transmitted are completely sent if this bit is reset after transmission has started. If the transmitter is disabled during the transmission of a CRC character, sync or flag characters are sent instead of CRC.

Send Break (D_4). When set, this bit immediately forces the Transmit Data output to the spacing condition, regardless of any data being transmitted. When reset, TxD returns to marking.

Transmit Bits/Characters 0 and 1 (D_5 and D_6). Together, D_6 and D_5 control the number of bits in each byte transferred to the transmit buffer.

D_6 Transmit Bits/ Character 1	D_5 Transmit Bits/ Character 0	Bits/Character
0	0	Five or less
0	1	7
1	0	6
1	1	8

Bits to be sent must be right justified, least-significant bits first. The Five Or Less mode allows transmission of one to five bits per character; however, the CPU should format the data character as shown in the following table.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
1	1	1	1	0	0	0	D	Sends one data bit
1	1	1	0	0	0	D	D	Sends two data bits
1	1	0	0	0	D	D	D	Sends three data bits
1	0	0	0	D	D	D	D	Sends four data bits
0	0	0	D	D	D	D	D	Sends five data bits

Data Terminal Ready (D_7). This is the control bit for the $\overline{DTR}$ pin. When set, $\overline{DTR}$ is active (Low); when reset, $\overline{DTR}$ is inactive (High).

WRITE REGISTER 6

This register is programmed to contain the transmit sync character in the Monosync mode, the first eight bits of a 16-bit sync character in the Bisync mode, or a transmit sync character in the External Sync mode. In the SDLC mode, it is programmed to contain the secondary address field used to compare against the address field of the SDLC frame.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Sync 7	Sync 6	Sync 5	Sync 4	Sync 3	Sync 2	Sync 1	Sync 0

WRITE REGISTER 7

This register is programmed to contain the receive sync character in the Monosync mode, a second byte (last eight bits) of a 16-bit sync character in the Bisync mode, or a flag character (01111110) in the SDLC mode. WR7 is not used in the External Sync mode.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Sync 15	Sync 14	Sync 13	Sync 12	Sync 11	Sync 10	Sync 9	Sync 8

Read Registers

The Z80-SIO contains three registers, RR0-RR2 (Figure 10), that can be read to obtain the status information for each channel (except for RR2—Channel B only). The status information includes error conditions, interrupt vector and standard communications-interface signals.

To read the contents of a selected read register other than RR0, the system program must first write the pointer byte to WR0 in exactly the same way as a write register operation. Then, by executing an input instruction, the contents of the addressed read register can be read by the CPU.

The status bits of RR0 and RR1 are carefully grouped to simplify status monitoring. For example, when the interrupt vector indicates that a Special Receive Condition interrupt has occurred, all the appropriate error bits can be read from a single register (RR1).

READ REGISTER 0

This register contains the status of the receive and transmit buffers; the $\overline{DCD}$, $\overline{CTS}$ and $\overline{SYNC}$ inputs; the Transmit Underrun/EOM latch; and the Break/Abort latch.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Break/ Abort	Transmit Underrun/ EOM	CTS	Sync/ Hunt	DCD	Transmit Buffer Empty	Interrupt Pending (Ch. A only)	Receive Character Available

Receive Character Available (D_0). This bit is set when at least one character is available in the receive buffer; it is reset when the receive FIFO is completely empty.

Interrupt Pending (D_1). Any interrupting condition in the Z80-SIO causes this bit to be set; however, it is readable only in Channel A. This bit is mainly used in applications that do not have vectored interrupts available. During the interrupt service routine in these applications, this bit indicates if any interrupt conditions are present in the Z80-SIO. This eliminates the need for analyzing all the bits of RR0 in both Channels A and B. Bit D_1 is reset when all the interrupting conditions are satisfied. This bit is always 0 in Channel B.

Transmit Buffer Empty (D_2). This bit is set whenever the transmit buffer becomes empty, except when a CRC character is being sent in a synchronous or SDLC mode. The bit is reset when a character is loaded into the transmit buffer. This bit is in the set condition after a reset.

Data Carrier Detect (D_3). The DCD bit shows the state of the $\overline{DCD}$ input at the time of the last change of any of the five External/Status bits (DCD, $\overline{CTS}$, Sync/Hunt, Break/Abort or Transmit Underrun/EOM). Any transition of the $\overline{DCD}$ input causes the DCD bit to be latched and causes an External/Status interrupt. To read the current state of the DCD bit, this bit must be read immediately following a Reset External/Status Interrupt command.

Sync/Hunt (D_4). Since this bit is controlled differently in the Asynchronous, Synchronous and SDLC modes, its operation is somewhat more complex than that of the other bits and therefore requires more explanation.

In asynchronous modes, the operation of this bit is similar to the DCD status bit, except that Sync/Hunt shows the state of the $\overline{\text{SYNC}}$ input. Any High-to-Low transition on the $\overline{\text{SYNC}}$ pin sets this bit and causes an External/Status interrupt (if enabled). The Reset External/Status Interrupt command is issued to clear the interrupt. A Low-to-High transition clears this bit and sets the External/Status interrupt. When the External/Status interrupt is set by the change in state of any other input or condition, this bit shows the inverted state of the $\overline{\text{SYNC}}$ pin at the time of the change. This bit must be read immediately following a Reset External/Status Interrupt command to read the current state of the $\overline{\text{SYNC}}$ input.

In the External Sync mode, the Sync/Hunt bit operates in a fashion similar to the Asynchronous mode, except the Enter Hunt Mode control bit enables the external sync detection logic. When the External Sync Mode and Enter Hunt Mode bits are set (for example, when the receiver is enabled following a reset), the $\overline{\text{SYNC}}$ input must be held High by the external logic until external character synchronization is achieved. A High at the $\overline{\text{SYNC}}$ input holds the Sync/Hunt status bit in the reset condition.

When external synchronization is achieved, $\overline{\text{SYNC}}$ must be driven Low on the second rising edge of $\overline{\text{RxC}}$ after that rising edge of $\overline{\text{RxC}}$ on which the last bit of the sync character was received. In other words, after the sync pattern is detected, the external logic must wait for two full Receive Clock cycles to activate the $\overline{\text{SYNC}}$ input. Once $\overline{\text{SYNC}}$ is forced Low, it is a good practice to keep it Low until the CPU informs the external sync logic that synchronization has been lost or a new message is about to start. Refer to Figure 18 for timing details. The High-to-Low transition of the $\overline{\text{SYNC}}$ input sets the Sync/Hunt bit, which—in turn—sets the External/Status interrupt. The CPU must clear the interrupt by issuing the Reset External/Status Interrupt command.

When the $\overline{\text{SYNC}}$ input goes High again, another External/Status interrupt is generated that must also be cleared. The Enter Hunt Mode control bit is set whenever character synchronization is lost or the end of message is detected. In this case, the Z80-SIO again looks for a High-to-Low transition on the $\overline{\text{SYNC}}$ input and the operation repeats as explained previously. This implies the CPU should also inform the external logic that character synchronization has been lost and that the Z80-SIO is waiting for $\overline{\text{SYNC}}$ to become active.

In the Monosync and Bisync Receive modes, the *Sync/Hunt status bit is initially set to 1 by the Enter Hunt Mode bit. The Sync/Hunt bit is reset when the* Z80-SIO establishes character synchronization. The

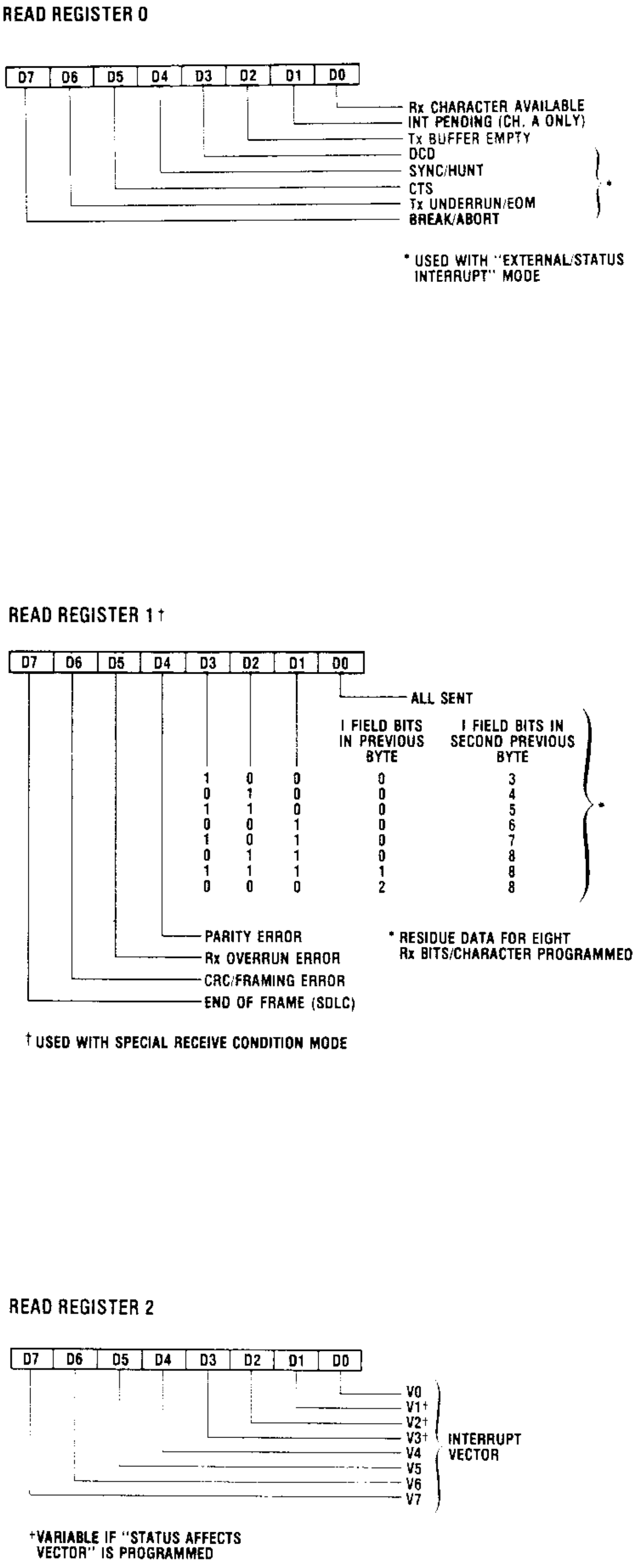


Figure 10. Read Register Bit Functions

High-to-Low transition of the Sync/Hunt bit causes an External/Status interrupt that must be cleared by the CPU issuing the Reset External/Status Interrupt command. This enables the Z80-SIO to detect the next transition of other External/Status bits.

When the CPU detects the end of message or that character synchronization is lost, it sets the Enter Hunt Mode control bit, which—in turn—sets the Sync/Hunt bit to 1. The Low-to-High transition of the Sync/Hunt bit sets the External/Status interrupt, which must also be cleared by the Reset External/Status Interrupt command. Note that the $\overline{\text{SYNC}}$ pin acts as an output in this mode and goes Low every time a sync pattern is detected in the data stream.

In the SDLC mode, the Sync/Hunt bit is initially set by the Enter Hunt mode bit, or when the receiver is disabled. In any case, it is reset to 0 when the opening flag of the first frame is detected by the Z80-SIO. The External/Status interrupt is also generated, and should be handled as discussed previously.

Unlike the Monosync and Bisync modes, once the Sync/Hunt bit is reset in the SDLC mode, it does not need to be set when the end of message is detected. The Z80-SIO automatically maintains synchronization. The only way the Sync/Hunt bit can be set again is by the Enter Hunt Mode bit, or by disabling the receiver.

Clear To Send (D_5). This bit is similar to the DCD bit, except that it shows the inverted state of the $\overline{\text{CTS}}$ pin.

Transmit Underrun/End Of Message (D_6). This bit is in a set condition following a reset (internal or external). The only command that can reset this bit is the Reset Transmit Underrun/EOM Latch command (WR0, D_6 and D_7). When the Transmit Underrun condition occurs, this bit is set; its becoming set causes the External/Status interrupt, which must be reset by issuing the Reset External/Status Interrupt command bits (WR0). This status bit plays an important role in conjunction with other control bits in controlling a transmit operation. Refer to "Bisync Transmit Underrun" and "SDLC Transmit Underrun" for additional details.

Break/Abort (D_7). In the Asynchronous Receive mode, this bit is set when a Break sequence (null character plus framing error) is detected in the data stream. The External/Status interrupt, if enabled, is set when Break is detected. The interrupt service routine must issue the Reset External/Status Interrupt command (WR0, CMD_2) to the break detection logic so the Break sequence termination can be recognized.

The Break/Abort bit is reset when the termination of the Break sequence is detected in the incoming data stream. The termination of the Break sequence also causes the External/Status interrupt to be set. The Reset External/Status Interrupt command must be issued to enable the break detection logic to look for the next Break sequence. A single extraneous null character is present in the receiver after the termination of a break; it should be read and discarded.

In the SDLC Receive mode, this status bit is set by the detection of an Abort sequence (seven or more 1's). The External/Status interrupt is handled the same way as in the case of a Break. The Break/Abort bit is not used in the Synchronous Receive mode.

READ REGISTER 1

This register contains the Special Receive condition status bits and Residue codes for the I-field in the SDLC Receive Mode.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
End Of Frame (SDLC)	CRC/ Framing Error	Receiver Overrun Error	Parity Error	Residue Code 2	Residue Code 1	Residue Code 0	All Sent

All Sent (D_0). In asynchronous modes, this bit is set when all the characters have completely cleared the transmitter. Transitions of this bit do not cause interrupts. It is always set in synchronous modes.

Residue Codes 0, 1 and 2 (D_1-D_3). In those cases of the SDLC receive mode where the I-field is not an integral multiple of the character length, these three bits indicate the length of the I-field. These codes are meaningful only for the transfer in which the End Of Frame bit is set (SDLC). For a receive character length of eight bits per character, the codes signify the following:

Residue Code 2	Residue Code 1	Residue Code 0	I-Field Bits In Previous Byte	I-Field Bits In Second Previous Byte
1	0	0	0	3
0	1	0	0	4
1	1	0	0	5
0	0	1	0	6
1	0	1	0	7
0	1	1	0	8
1	1	1	1	8
0	0	0	2	8

I-Field bits are right-justified in all cases.

If a receive character length different from eight bits is used for the I-field, a table similar to the previous one may be constructed for each different character length. For no residue (that is, the last character boundary coincides with the boundary of the I-field and CRC field), the Residue codes are:

Bits per Character	Residue Code 2	Residue Code 1	Residue Code 0
8 Bits per Character	0	1	1
7 Bits per Character	0	0	0
6 Bits per Character	0	1	0
5 Bits per Character	0	0	1

Parity Error (D_4). When parity is enabled, this bit is set for those characters whose parity does not match the programmed sense (even/odd). The bit is latched, so once an error occurs, it remains set until the Error Reset command (WR0) is given.

Receive Overrun Error (D_5). This bit indicates that more than three characters have been received without a read from the CPU. Only the character that has been written over is flagged with this error, but when this character is read, the error condition is latched until reset by the Error Reset command. If Status Affects Vector is enabled, the character that has been overrun interrupts with a Special Receive Condition vector.

CRC/Framing Error (D_6). If a Framing Error occurs (asynchronous modes), this bit is set (and not latched) for the receive character in which the Framing Error occurred. Detection of a Framing Error adds an additional one-half of a bit time to the character time so the Framing Error is not interpreted as a new start bit. In synchronous and SDLC modes, this bit indicates the result of comparing the CRC checker to the appropriate check value. This bit is reset by issuing an Error Reset command. The bit is not latched, so it is always updated when the next character is received. When used for CRC error and status in synchronous modes, it is usually set since most bit combinations result in a non-zero CRC except for a correctly completed message.

End Of Frame (D_7). This bit is used only with the SDLC mode and indicates that a valid ending flag has been received and that the CRC Error and Residue codes are also valid. This bit can be reset by issuing the Error Reset command. It is also updated by the first character of the following frame.

READ REGISTER 2 (Ch. B Only)

This register contains the interrupt vector written into WR2 if the Status Affects Vector control bit is not set. If the control bit is set, it contains the modified vector shown in the Status Affects Vector paragraph of the Write Register 1 section. When this register is read, the vector returned is modified by the highest priority interrupting condition at the time of the read. If no interrupts are pending, the vector is modified with $V_3=0$, $V_2=1$ and $V_1=1$. This register may be read only through Channel B.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
V_7	V_6	V_5	V_4	V_3	V_2	V_1	V_0
					Variable if Status Affects Vector is enabled		

Applications

The flexibility and versatility of the Z80-SIO make it useful for numerous applications, a few of which are included here. These examples show several applications that combine the Z80-SIO with other members of the Z80 family.

Figure 11 shows simple processor-to-processor communication over a direct line. Both remote processors in this system can communicate to the Z80-CPU with different protocols and data rates. Depending on the complexity of the application, other Z80 peripheral circuits (Z80-CTC, for example) may be required. The unused channel of the Z80-SIO can be used to control other peripherals or they can be connected to other remote processors.

Figure 12 illustrates how both channels of a single Z80-SIO are used with modems that have primary and secondary, or reverse channel options. Alternatively, two modems without these options can be connected to the Z80-SIO. A suitable baud-rate generator (Z80-CTC) must be used for asynchronous modems.

Figure 13 shows the Z80-SIO in a data concentrator, a relatively complex application that uses two Z80-SIOs to perform a variety of functions. The data concentrator can be used to collect data from many terminals over low-speed lines and transmit it over a single high-speed line after editing and reformatting.

The Z80-DMA controller circuit is used with Z80-SIO #2 to transmit the reformatted data at high speed with the required protocol. The high-speed modem provides the transmit clock for this channel. The Z80-CTC counter-timer circuit supplies the transmit and receive clocks for the low-speed lines and is also used as a time-out counter for various functions.

Z80-SIO #1 controls local or remote terminals. A single intelligent terminal is shown within the dashed lines. The terminal employs a Z80-SIO to communicate to the data concentrator on one channel while providing the interface to a line printer over its second channel. The intelligent terminal shown could be designed to operate interactively with the operator.

Depending on the software and hardware capabilities built into this system, the data concentrator can employ store-and-forward or hold-and-forward methods for regulating information traffic between slow terminals and the high-speed remote processor. If the high-speed channel is provided with a dial-out option, the channel can be connected to a number of remote processors over a switched line.

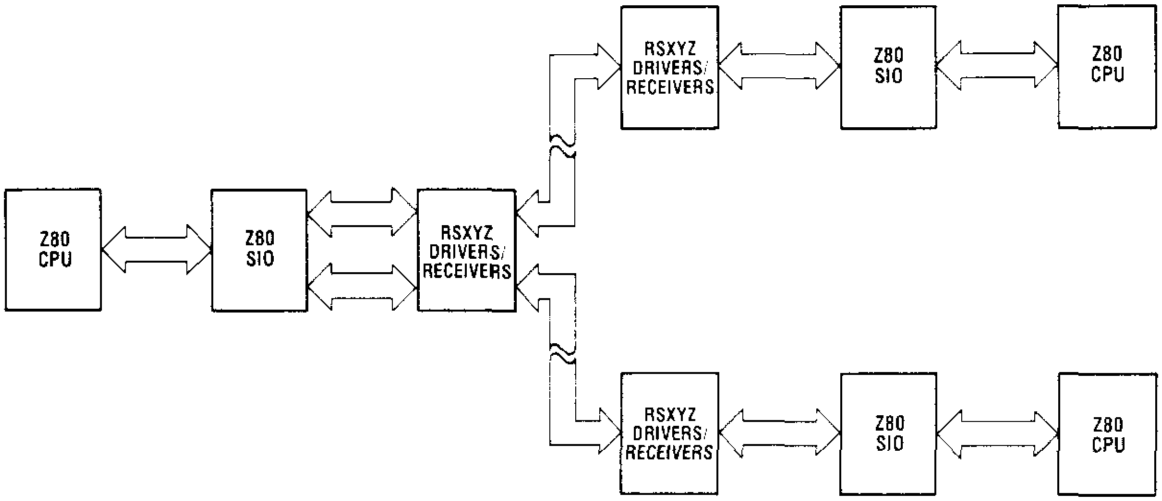


Figure 11. Synchronous/Asynchronous Processor-to-Processor Communcation (Direct Wire to Two Remote Locations)

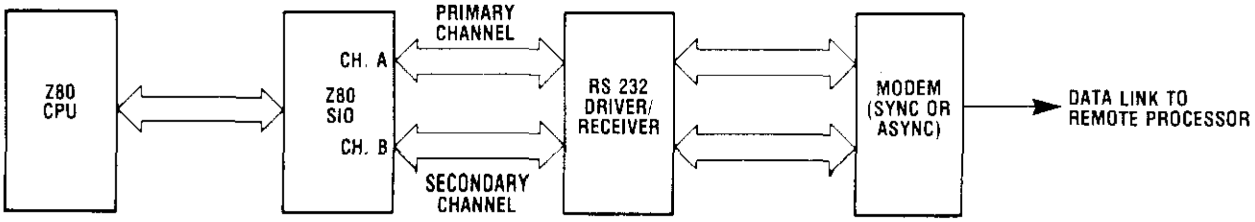


Figure 12. Synchronous/Asynchronous Processor-to-Processor Communication (Using Telephone Line)

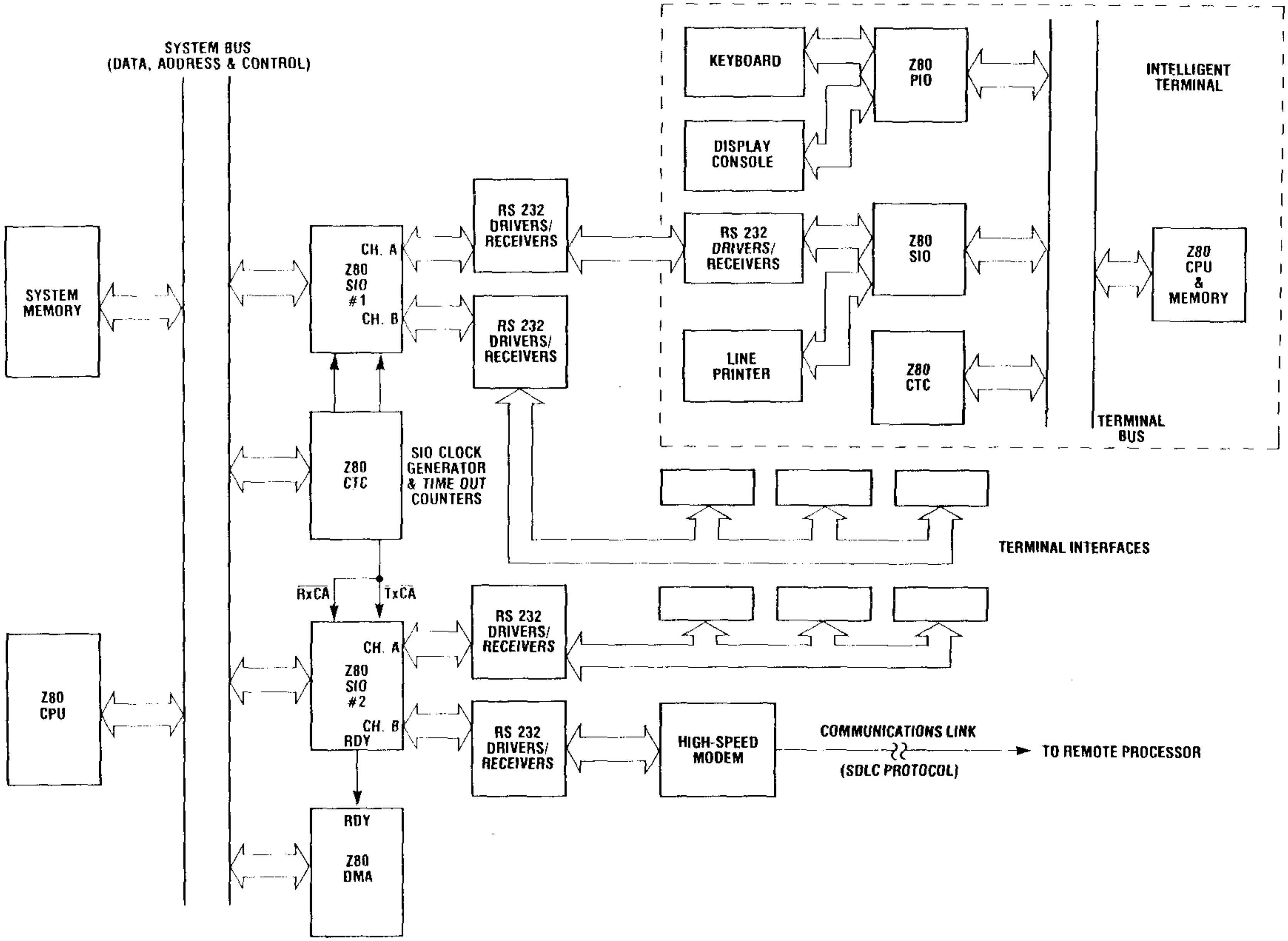


Figure 13. Data Concentrator

Timing

READ CYCLE

The timing signals generated by a Z80-CPU input instruction to read a Data or Status byte from the Z80-SIO are illustrated in Figure 14a.

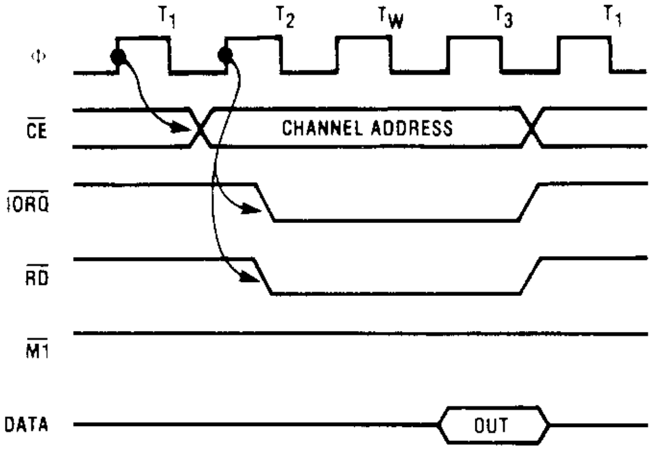


Figure 14a. Read Cycle

INTERRUPT ACKNOWLEDGE CYCLE

After receiving an Interrupt Request signal ($\overline{\text{INT}}$ pulled Low), the Z80-CPU sends an Interrupt Acknowledge signal ($\overline{\text{M1}}$ and $\overline{\text{IORQ}}$ both Low). The daisy-chained interrupt circuits determine the highest priority interrupt requestor. The IEI of the highest priority peripheral is terminated High. For any peripheral that has no interrupt pending or under service, IEO = IEI. Any peripheral that does have an interrupt pending or under service forces its IEO Low.

To insure stable conditions in the daisy chain, all interrupt status signals are prevented from changing while $\overline{\text{M1}}$ is Low. When $\overline{\text{IORQ}}$ is Low, the highest priority interrupt requestor (the one with IEI High) places its interrupt vector on the data bus and sets its internal interrupt-under-service latch.

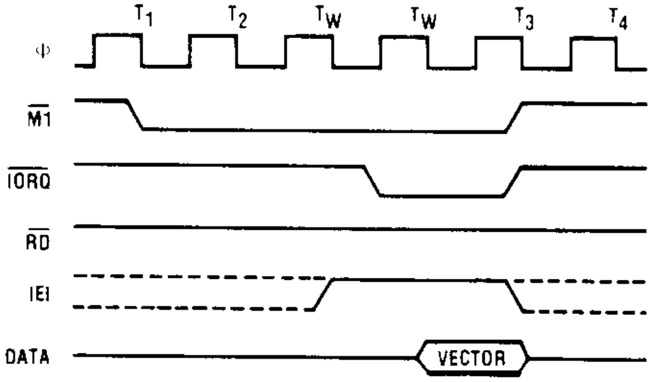


Figure 14c. Interrupt Acknowledge Cycle

WRITE CYCLE

Figure 14b illustrates the timing and data signals generated by a Z80-CPU output instruction to write a Data or Control bvte into the Z80-SIO.

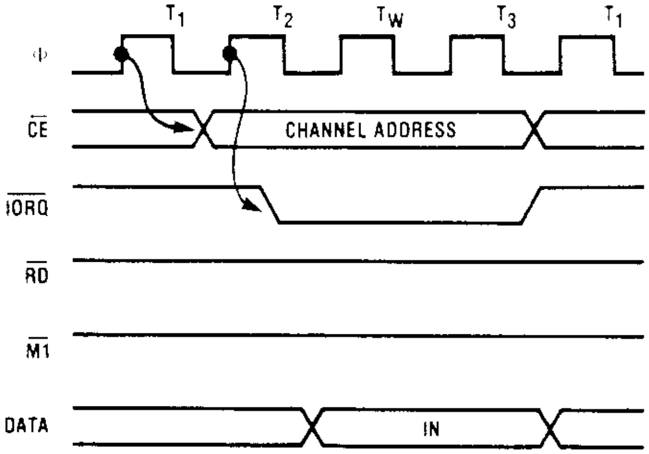


Figure 14b. Write Cycle

RETURN FROM INTERRUPT CYCLE

Normally, the Z80-CPU issues a RETI (RETurn from Interrupt) instruction at the end of an interrupt service routine. RETI is a 2-byte opcode (ED-4D) that resets the interrupt-under-service latch to terminate the interrupt that has just been processed. This is accomplished by manipulating the daisy chain in the following way.

The normal daisy chain operation can be used to detect a pending interrupt; however, it cannot distinguish between an interrupt under service and a pending unacknowledged interrupt of a higher priority. Whenever "ED" is decoded, the daisy chain is modified by forcing High the IEO of any interrupt that has not yet been acknowledged. Thus the daisy chain identifies the device presently under service as the only one with an IEI High and an IEO Low. If the next opcode byte is "4D," the interrupt-under-service latch is reset.

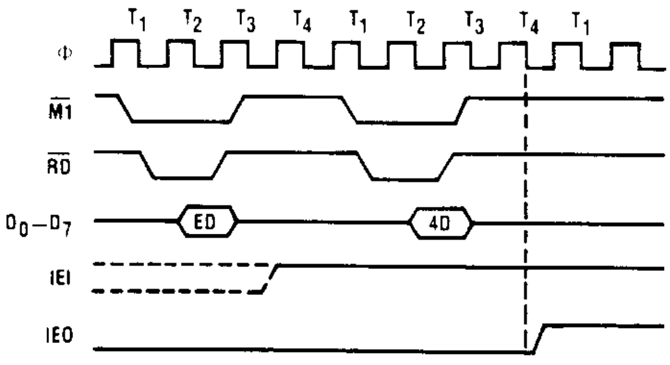


Figure 14d. Return from Interrupt Cycle

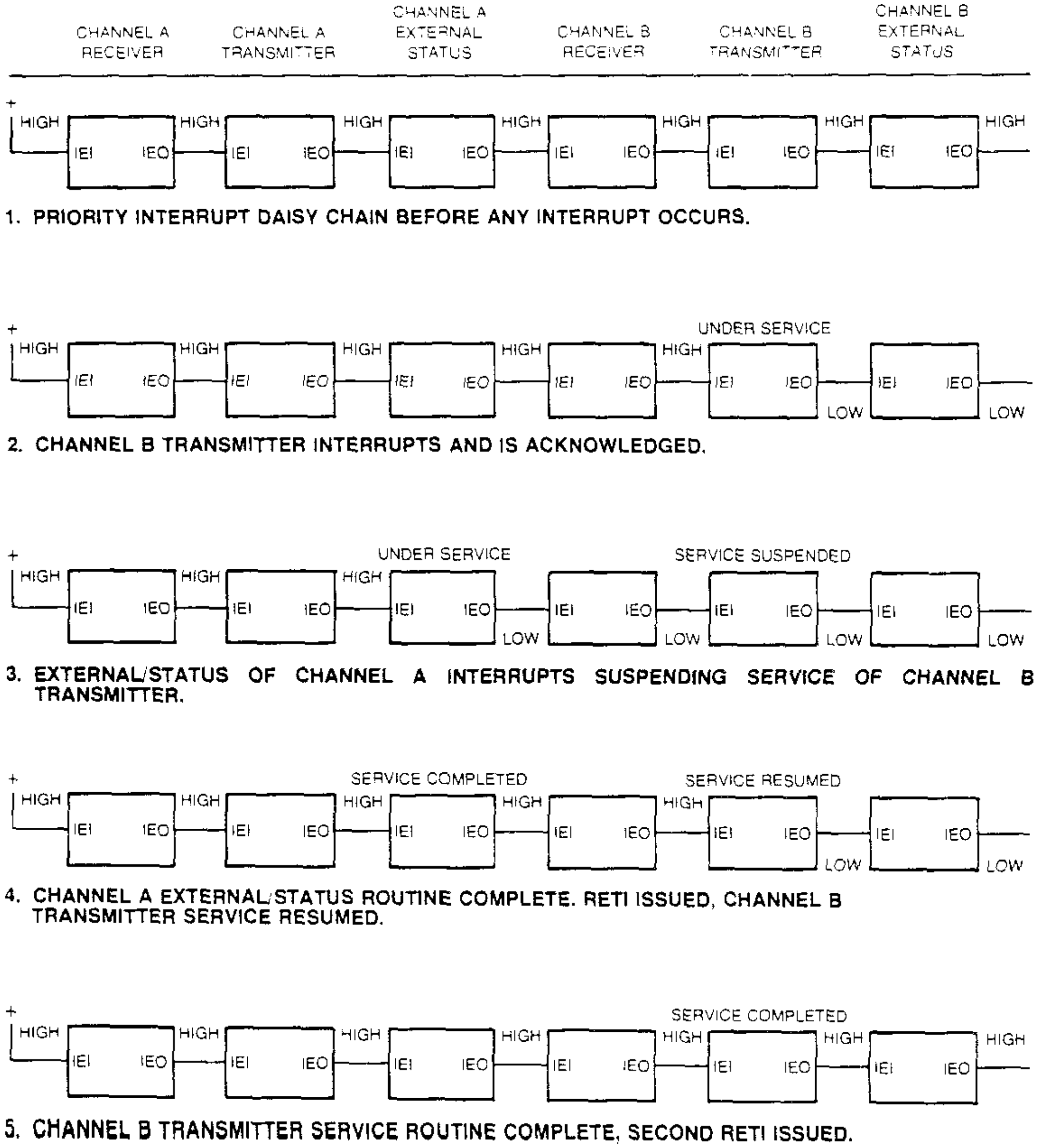
The ripple time of the interrupt daisy chain (both the High-to-Low and the Low-to-High transitions) limits the number of devices that can be placed in the daisy chain. Ripple time can be improved with carry-look-ahead, or by extending the interrupt acknowledge cycle. For further information about techniques for increasing the number of daisy-chained devices, refer to Zilog Application Note 03-0041-01 (*The Z80 Family Program Interrupt Structure*).

DAISY CHAIN INTERRUPT NESTING

Figure 15 illustrates the daisy chain configuration of interrupt circuits and their behavior with nested interrupts (an interrupt that is interrupted by another with a higher priority).

Each box in the illustration could be a separate external Z80 peripheral circuit with a user-defined order of interrupt priorities. However, a similar daisy chain structure also exists inside the Z80-SIO, which has six interrupt levels with a fixed order of priorities.

The case illustrated occurs when the transmitter of Channel B interrupts and is granted service. While this interrupt is being serviced, it is interrupted by a higher priority interrupt from Channel A. The second interrupt is serviced and—upon completion—a RETI instruction is executed or a RETI command is written into the Z80-SIO, resetting the interrupt-under-service latch of the Channel A interrupt. At this time, the service routine for Channel B is resumed. When it is completed, another RETI instruction is executed to complete the interrupt service.



1. PRIORITY INTERRUPT DAISY CHAIN BEFORE ANY INTERRUPT OCCURS.

2. CHANNEL B TRANSMITTER INTERRUPTS AND IS ACKNOWLEDGED.

3. EXTERNAL/STATUS OF CHANNEL A INTERRUPTS SUSPENDING SERVICE OF CHANNEL B TRANSMITTER.

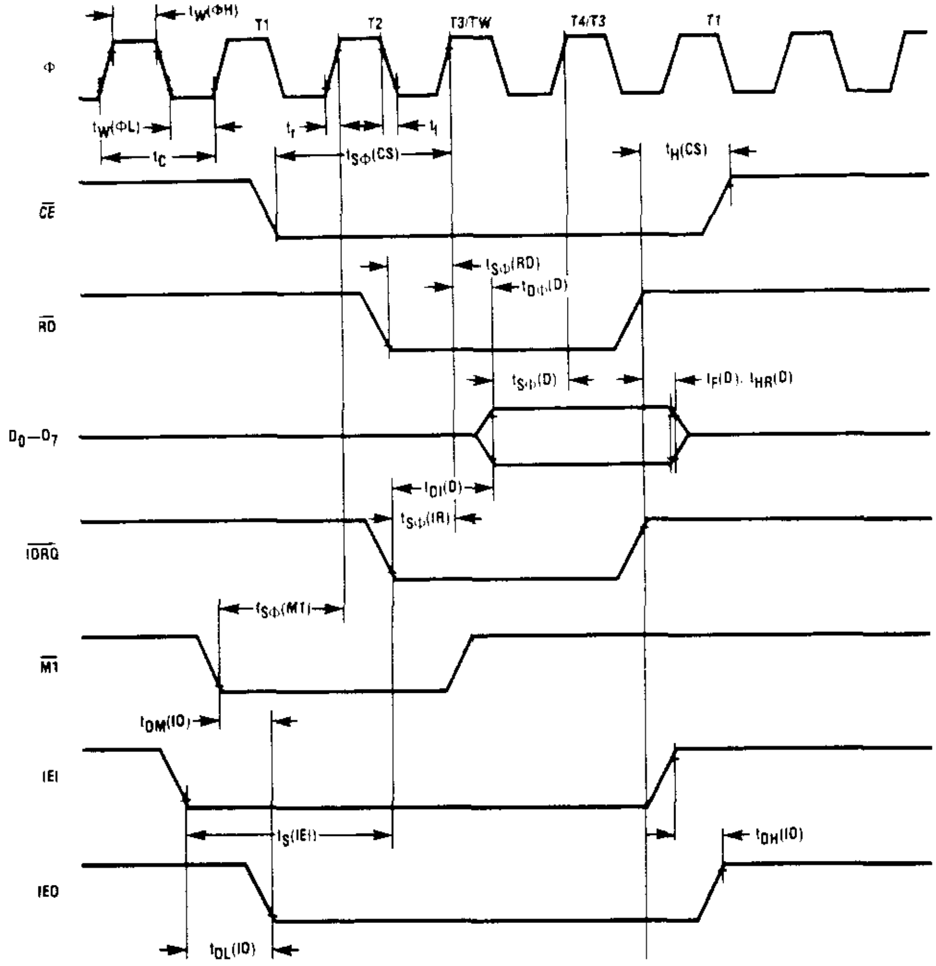
4. CHANNEL A EXTERNAL/STATUS ROUTINE COMPLETE. RETI ISSUED, CHANNEL B TRANSMITTER SERVICE RESUMED.

5. CHANNEL B TRANSMITTER SERVICE ROUTINE COMPLETE, SECOND RETI ISSUED.

Figure 15. Typical Interrupt Sequence

AC Characteristics

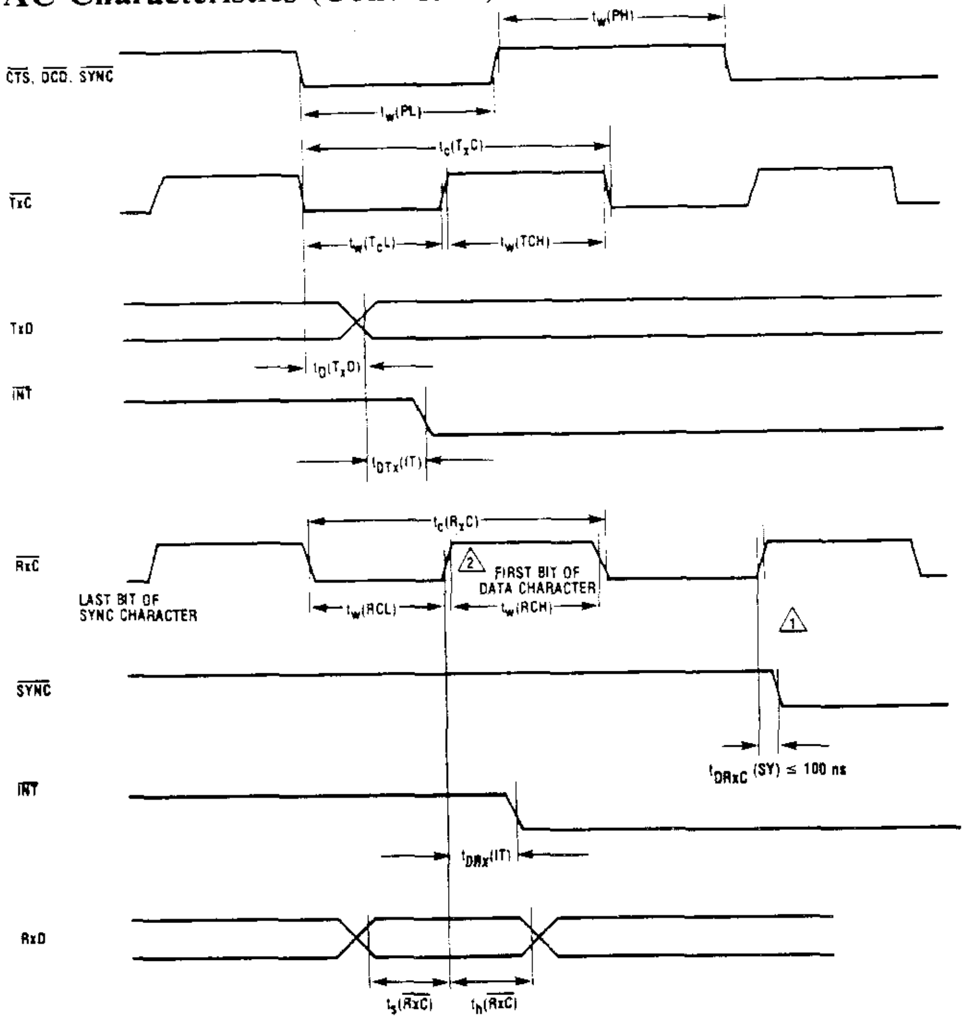
$T_A = 0°C$ to $70°C$, $V_{CC} = +5V$, ±5%



Signal	Symbol	Parameter	Z80-SIO Min	Z80-SIO Max	Z80A-SIO Min	Z80A-SIO Max	Unit
ϕ	$t_c(\phi)$	Clock Period	400	4000	250	4000	ns
	$t_w(\phi H)$	Clock Pulse Width, clock HIGH	170	2000	105	2000	ns
	$t_w(\phi L)$	Clock Pulse Width, clock LOW	170	2000	105	2000	ns
	t_r, t_f	Clock Rise and Fall Times	0	30	0	30	ns
	t_H	Any Unspecified Hold Time for setup times specified below	0		0		ns
$\overline{CE}$, $\overline{B}/A$ $\overline{C}/D$, $\overline{IORQ}$	$t_{S\phi}(CS)$	Control Signal Setup Time to rising edge of ϕ during Read or Write Cycle	160		145		ns
D_0-D_7	$t_{D\phi}(D)$	Data Output Delay from rising edge of ϕ during Read Cycle		240		220	ns
	$t_{S\phi}(D)$	Data Setup Time to rising edge of ϕ during Write or M1 Cycle	50		50		ns
	$t_{DI}(D)$	Data Output Delay from falling edge of $\overline{IORQ}$ during INTA Cycle		340		160	ns
	$t_F(D)$	Delay to Floating Bus (output buffer disable time)		230		110	ns
IEI	$t_S(IEI)$	IEI Setup Time to falling edge of $\overline{IORQ}$ during INTA Cycle	200		140		ns
IEO	$t_{DH}(IO)$	IEO Delay Time from rising edge of IEI (after 'ED' decode)		150		100	ns
	$t_{DL}(IO)$	IEO Delay Time from falling edge of IEI		150		100	ns
	$t_{DM}(IO)$	IEO Delay Time from falling edge of $\overline{M1}$ (interrupt occurring just prior to M1)		300		190	ns
M1	$t_{S\phi}(M1)$	$\overline{M1}$ Setup Time to rising edge of ϕ during INTA or M1 Cycle	210		90		ns
$\overline{RD}$	$t_{S\phi}(RD)$	$\overline{RD}$ Setup Time to rising edge of ϕ during Read or M1 Cycle	240		115		ns

***If WAIT from the SIO is to be used, $\overline{CE}$, $\overline{IORQ}$, C/$\overline{D}$ and $\overline{M1}$ must be valid for as long as the Wait condition is to persist.**

AC Characteristics (Continued)



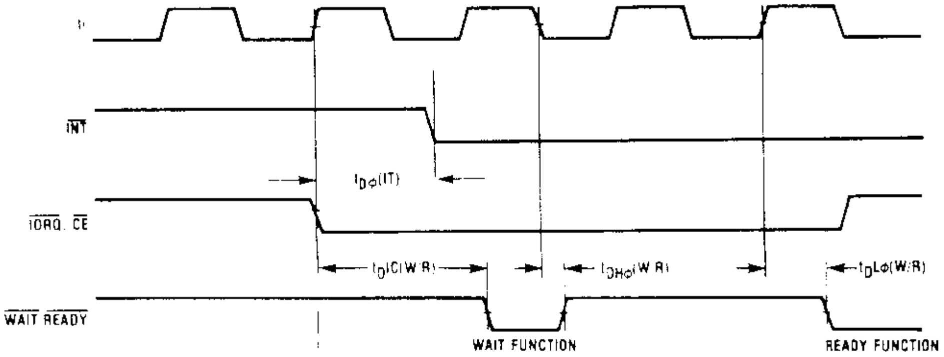
NOTES:

1. The $\overline{SYNC}$ input must be driven Low on the rising edge of $\overline{RxC}$ delayed two complete clock cycles from the last bit of the sync character.
2. Data character assembly begins on the next Receive Clock cycle after the last bit of the sync character is received.

Signal	Symbol	Parameter	Z80-SIO Min	Z80-SIO Max	Z80A-SIO Min	Z80A-SIO Max	Unit
$\overline{INT}$	$t_{DRx}(IT)$	INT Delay Time from rising edge of RxC	10	13	10	13	ϕ periods
	$t_{DTx}(IT)$	$\overline{INT}$ Delay Time from transition of Xmit Data Bit	5	9	5	9	ϕ periods
$\overline{CTSA}$, $\overline{CTSB}$, $\overline{DCDA}$, $\overline{DCDB}$, $\overline{SYNCA}$, $\overline{SYNCB}$	$t_w(PH)$	Minimum HIGH Pulse Width for latching state into register and generating interrupt	200		200		ns
	$t_w(PL)$	Minimum LOW Pulse Width for latching state into register and generating interrupt	200		200		ns
$\overline{SYNCA}$, $\overline{SYNCB}$	$t_{DL}(SY)$	Sync Pulse Delay Time from rising edge of $\overline{RxC}$, Output Modes	4	7	4	7	ϕ periods
	$t_{DRxC}(SY)$	Sync Pulse Delay Time from rising edge of $\overline{RxC}$ External Sync Mode		100		100	ns
$\overline{TxCA}$, $\overline{TxCB}$	$t_c(TxC)$	Transmit Clock Period	400	∞	400	∞	ns
	$t_w(TCH)$	Transmit Clock Pulse Width, clock HIGH	180	∞	180	∞	ns
	$t_w(TCL)$	Transmit Clock Pulse Width, clock LOW	180	∞	180	∞	ns
TxDA, TxDB†	$t_D(TxD)$	TxD Output Delay from falling Edge of $\overline{TxC}$ (x1 Clock Mode)		400		300	ns
$\overline{RxCA}$, $\overline{RxCB}$	$t_c(RxC)$	Receive Clock Period	400	∞	400	∞	ns
	$t_w(RCH)$	Receive Clock Pulse Width, clock HIGH	180	∞	180	∞	ns
	$t_w(RCL)$	Receive Clock Pulse Width, clock LOW	180	∞	180	∞	ns
RxDA, RxDB†	$t_s(RxC)$	Setup Time to rising edge of $\overline{RxC, x1}$ mode	0		0		ns
	$t_h(RxC)$	Hold Time from rising edge of $\overline{RxC, x1}$ mode	140		140		ns

†In all modes, the system clock (ϕ) rate must be at least 4.5 times the maximum data rate. $\overline{RESET}$ must be active a minimum of one complete ϕ cycle.

AC Characteristics (Continued)



Signal	Symbol	Parameter	Z80-SIO Min	Z80-SIO Max	Z80A-SIO Min	Z80A-SIO Max	Unit
$\overline{\text{INT}}$	$t_D\phi(IT)$	$\overline{\text{INT}}$ Delay Time from rising edge of ϕ		200		200	ns
$\overline{\text{WAIT}}/\text{READY}$	$t_DIC(W/R)$	$\overline{\text{WAIT}}/\text{READY}$ Delay Time from $\overline{\text{IORQ}}$ or $\overline{\text{CE}}$ in Wait Mode		180		130	ns
	$t_DH\phi(W/R)$	$\overline{\text{WAIT}}/\text{READY}$ Delay Time from falling edge of ϕ. $\overline{\text{WAIT}}/\text{READY}$ HIGH, Wait Mode		150		130	ns
	$t_DRx(W/R)$	$\overline{\text{WAIT}}/\text{READY}$ Delay Time from rising edge of $\overline{\text{RxC}}$ Data Bit, Ready Mode	10	13	10	13	ϕ periods
	$t_DTx(W/R)$	$\overline{\text{WAIT}}/\text{READY}$ Delay Time from center of Transmit Data Bit, Ready Mode	5	9	5	9	ϕ periods
	$t_DL\phi(W/R)$	$\overline{\text{WAIT}}/\text{READY}$ Delay Time from rising edge of ϕ. $\overline{\text{WAIT}}/\text{READY}$ LOW Ready Mode		120		120	ns

DC Characteristics

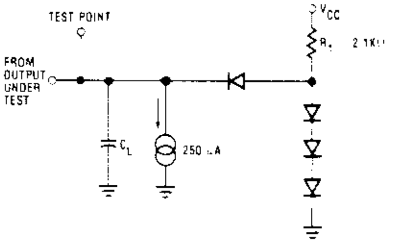
$T_A = 0°C$ to $70°C$, $V_{CC} = +5V, \pm 5\%$

Symbol	Parameter	Min.	Max.	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	−0.3	+0.45	V	
V_{IHC}	Clock Input High Voltage	$V_{CC}-0.6$	+5.5	V	
V_{IL}	Input Low Voltage	−0.3	+0.8	V	
V_{IH}	Input High Voltage	+2.0	+5.5	V	
V_{OL}	Output Low Voltage		+0.4	V	$I_{OL} = 2.0$ mA
V_{OH}	Output High Voltage	+2.4		V	$I_{OH} = -250\ \mu A$
I_{LI}	Input Leakage Current	−10	+10	μA	$0 \leq V_{IN} \leq V_{CC}$
I_Z	3-State Output/Data Bus Input Leakage Current	−10	+10	μA	$0 \leq V_{IN} \leq V_{CC}$
I_{LSY}	$\overline{\text{SYNC}}$ Pin Leakage Current	−40	+10	μA	
I_{CC}	Power Supply Current		100	mA	

Capacitance

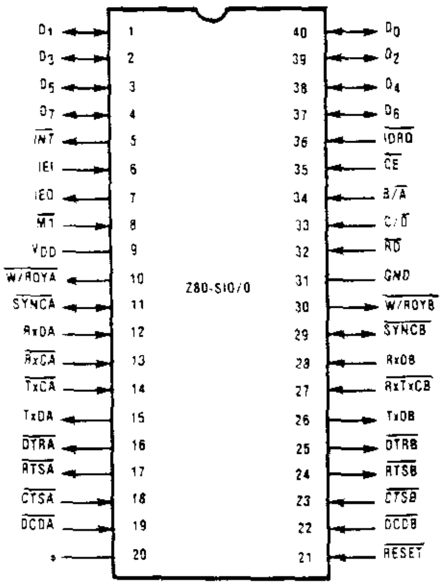
$T_A = 25°C$, $f = 1$ MHz

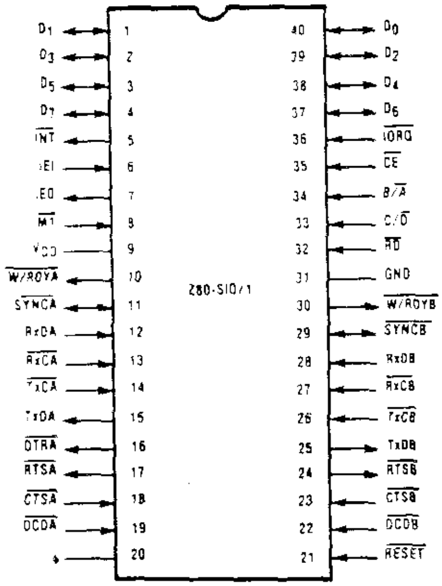
Symbol	Parameter	Min.	Max.	Unit	Test Condition
C	Clock Capacitance		40	pF	Unmeasured pins returned to ground
C_{IN}	Input Capacitance		5	pF	
C_{OUT}	Output Capacitance		10	pF	

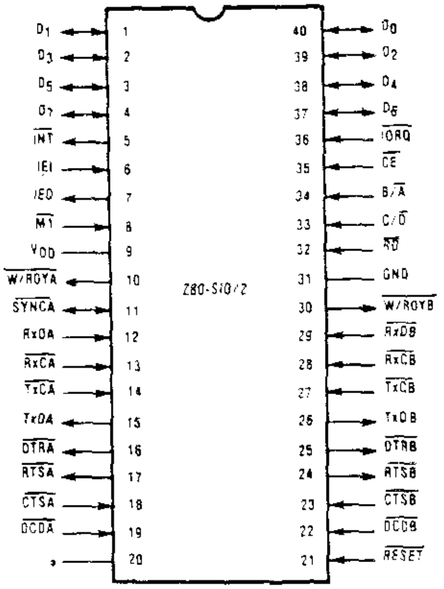


$C_L = 50$ pF. Increase delay by 10 ns for each 50 pF increase in C_L, up to 200 pF maximum.

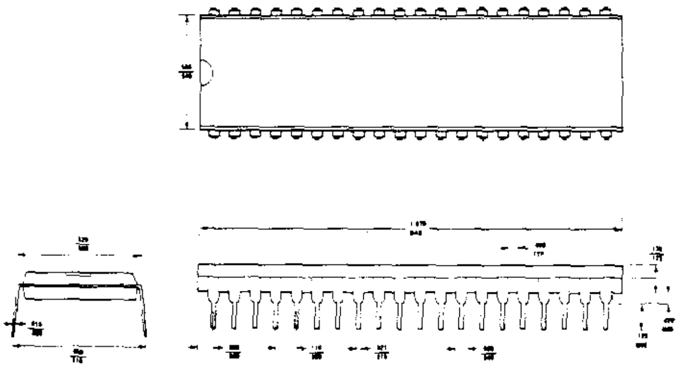
Package Configurations



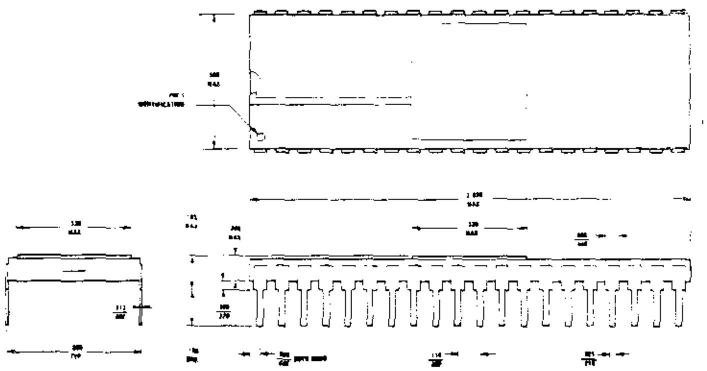




Package Outlines



40-Pin Plastic



40-Pin Ceramic

Ordering Information

C — Ceramic
P — Plastic
S — Standard 5V ±5%, 0° to 70°C
E — Extended 5V ±5%, −40° to 85°C
M — Military 5V ±10%, −55° to 125°C
/0 — Type 0 Bonding
/1 — Type 1 Bonding
/2 — Type 2 Bonding

Example:

Z80-SIO/1 CS (Ceramic—Standard Range—Type 1 Bonding)

Z80-SIO/0 PS (Plastic — Standard Range — Type 0 Bonding)

Appendix B

Z80-CTC

Technical Manual

TABLE OF CONTENTS

1.0 INTRODUCTION

The Z80-Counter Timer Circuit (CTC) is a progammable component with four independent channels that provide counting and timing functions for microcomputer systems based on the Z80-CPU. The CPU can configure the CTC channels to operate under various modes and conditions as required to interface with a wide range of devices. In most applications, little or no external logic is required. The Z80-CTC utilizes N-channel silicon gate depletion load technology and is packaged in a 28-pin DIP. The Z80-CTC requires only a single 5 volt supply and a one-phase 5 volt clock. Major features of the Z80-CTC include:

- All inputs and outputs fully TTL compatible.
- Each channel may be selected to operate in either Counter Mode or Timer Mode.
- Used in either mode, a CPU-readable Down Counter indicates number of counts-to-go until zero.
- A Time Constant Register can automatically reload the Down Counter at Count Zero in Counter and Timer Mode.
- Selectable positive or negative trigger initiates time operation in Timer Mode. The same input is monitored for event counts in Counter Mode.
- Three channels have Zero Count/Timeout outputs capable of driving Darlington transistors.
- Interrupts may be programmed to occur on the zero count condition in any channel.
- Daisy chain priority interrupt logic included to provide for automatic interrupt vectoring without external logic.

2.0 CTC ARCHITECTURE

2.1 OVERVIEW

A block diagram of the Z80-CTC is shown in figure 2.0-1. The internal structure of the Z80-CTC consists of a Z80-CPU bus interface, Internal Control Logic, four sets of Counter/Timer Channel Logic, and Interrupt Control Logic. The four independent counter/timer channels are identified by sequential numbers from 0 to 3. The CTC has the capability of generating a unique interrupt vector for each separate channel (for automatic vectoring to an interrupt service routine). The 4 channels can be connected into four contiguous slots in the standard Z80 priority chain with channel number 0 having the highest priority. The CPU bus interface logic allows the CTC device to interface directly to the CPU with no other external logic. However, port address decoders and/or line buffers may be required for large systems.

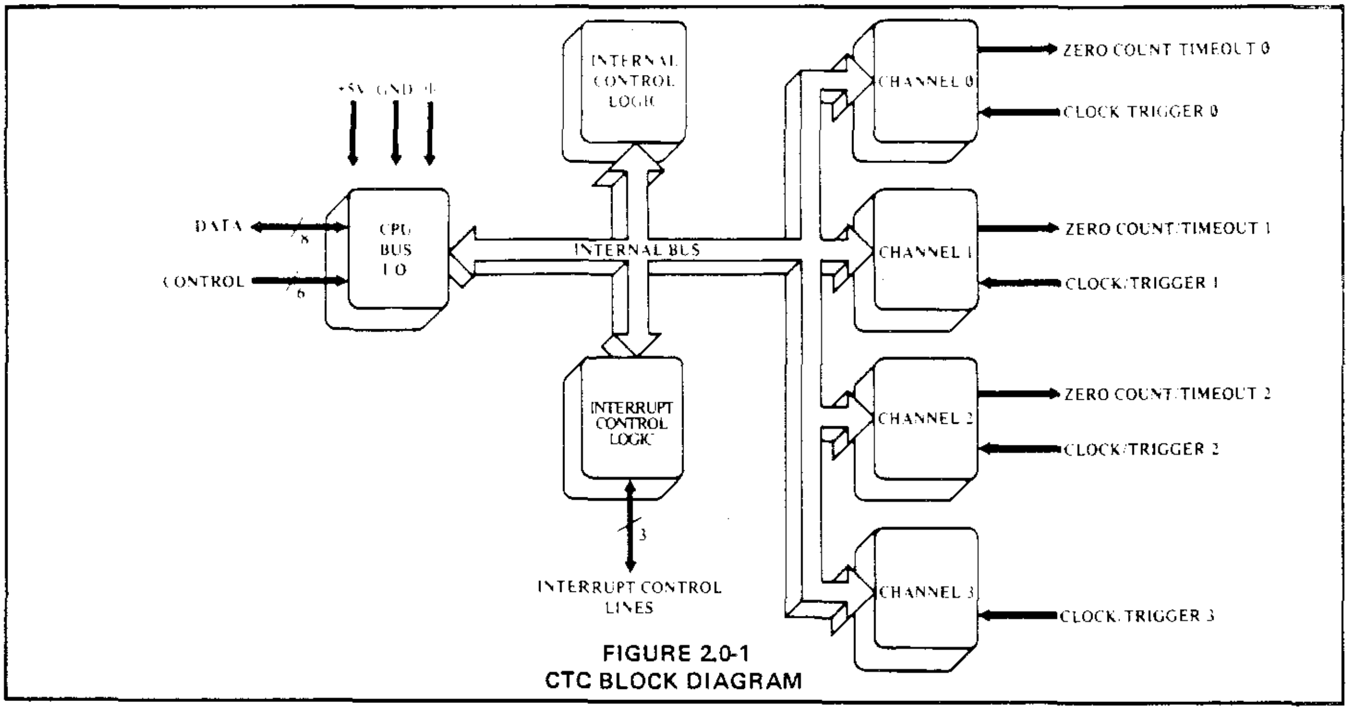


FIGURE 2.0-1
CTC BLOCK DIAGRAM

2.2 STRUCTURE OF CHANNEL LOGIC

The structure of one of the four sets of Counter/Timer Channel Logic is shown in figure 2.0-2. This logic is composed of 2 registers, 2 counters and control logic. The registers are an 8-bit Time Constant Register and an 8-bit Channel Control Register. The counters are an 8-bit CPU-readable Down Counter and an 8-bit Prescaler.

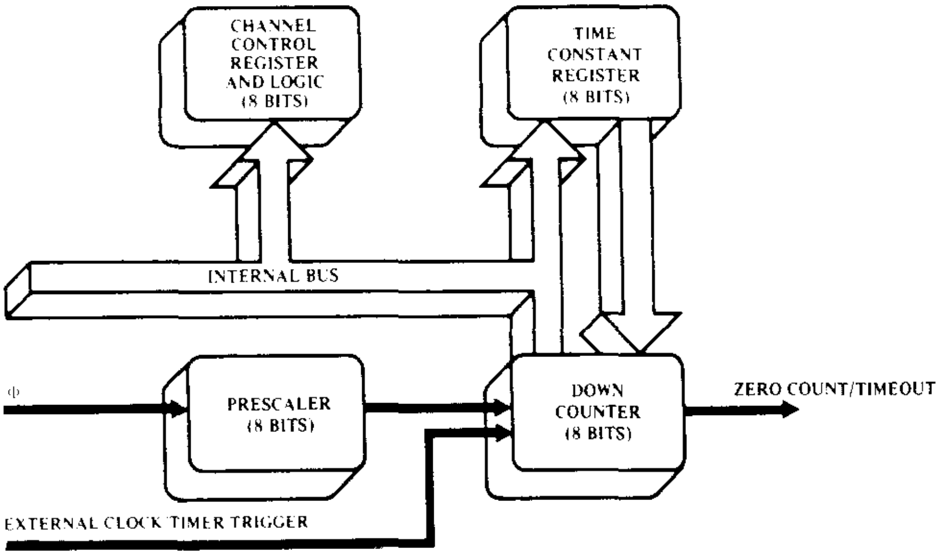

CHANNEL CONTROL REGISTER AND LOGIC (8 BITS)
TIME CONSTANT REGISTER (8 BITS)
INTERNAL BUS
Φ
PRESCALER (8 BITS)
DOWN COUNTER (8 BITS)
ZERO COUNT/TIMEOUT
EXTERNAL CLOCK TIMER TRIGGER

FIGURE 2.0-2
CHANNEL BLOCK DIAGRAM

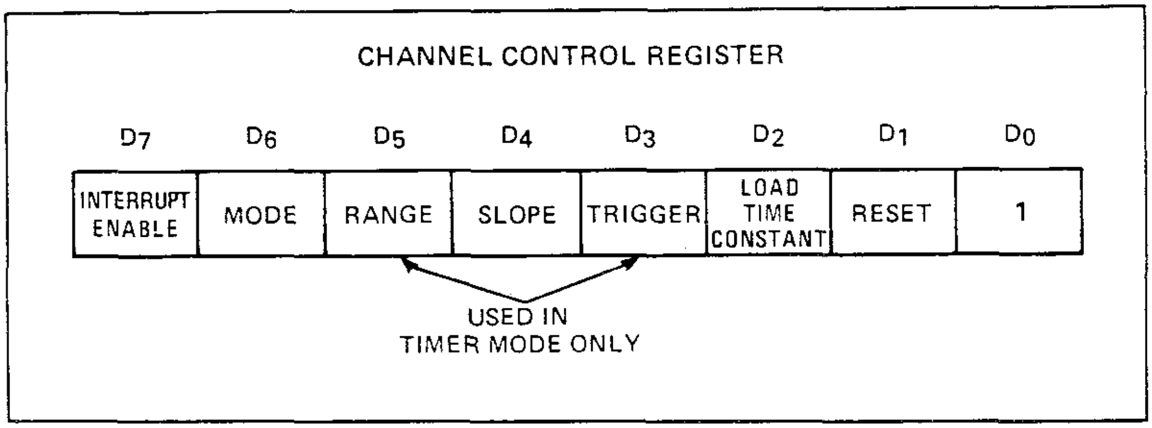
2.2.1 THE CHANNEL CONTROL REGISTER AND LOGIC

The Channel Control Register (8-bit) and Logic is written to by the CPU to select the modes and parameters of the channel. Within the entire CTC device there are four such registers, corresponding to the four Counter/Timer Channels. Which of the four is being written to depends on the encoding of two channel select input pins: CS0 and CS1 (usually attached to A0 and A1 of the CPU address bus). This is illustrated in the truth table below:

	CS1	CS0
Ch 0	0	0
Ch 1	0	1
Ch 2	1	0
Ch 3	1	1

2.2.1 CONTINUED

In the control word written to program each Channel Control Register, bit 0 is always set, and the other 7 bits are programmed to select alternatives on the channel's operating modes and parameters, as shown in the diagram below. (For a more complete discussion see section 4.0: "CTC Operating Modes" and section 5.0: "CTC Programming.")



CHANNEL CONTROL REGISTER

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
INTERRUPT ENABLE	MODE	RANGE	SLOPE	TRIGGER	LOAD TIME CONSTANT	RESET	1

2.2.2 THE PRESCALER

Used in the Timer Mode only, the Prescaler is an 8-bit device which can be programmed by the CPU via the Channel Control Register to divide its input, the System Clock (Φ), by 16 or 256. The output of the Prescaler is then fed as an input to clock the Down Counter, which initially, and every time it clocks down to zero, is reloaded automatically with the contents of the Time Constant Register. In effect this again divides the System Clock by an additional factor of the time constant. Every time the Down Counter counts down to zero, its output, Zero Count/Timeout (ZC/TO), is pulsed high.

2.2.3 THE TIME CONSTANT REGISTER

The Time Constant Register is an 8-bit register, used in both Counter Mode and Timer Mode, programmed by the CPU just after the Channel Control Word with an integer time constant value of 1 through 256. This register loads the programmed value into the Down Counter when the CTC is first initialized and reloads the same value into the Down Counter automatically whenever it counts down thereafter to zero. If a new time constant is loaded into the Time Constant Register while a channel is counting or timing, the present down count will be completed before the new time constant is loaded into the Down Counter. (For details of how a time constant is written to a CTC channel, see section 5.0: "CTC Programming.")

2.2.4 THE DOWN COUNTER

The Down Counter is an 8-bit register, used in both Counter Mode and Timer Mode, loaded initially, and later when it counts down to zero, by the Time Constant Register. The Down Counter is decremented by each external clock edge in the Counter Mode, or in the Timer Mode, by the clock output of the Prescaler. At any time, by performing a simple I/O Read at the port address assigned to the selected CTC channel, the CPU can access the contents of this register and obtain the number of counts-to-zero. Any CTC channel may be programmed to generate an interrupt request sequence each time the zero count is reached.

In channels 0, 1, and 2, when the zero count condition is reached, a signal pulse appears at the corresponding ZC/TO pin. Due to package pin limitations, however, channel 3 does not have this pin and so may be used only in applications where this output pulse is not required.

2.3 INTERRUPT CONTROL LOGIC

The Interrupt Control Logic insures that the CTC acts in accordance with Z80 system interrupt protocol for nested priority interrupting and return from interrupt. The priority of any system device is determined by its physical location in a daisy chain configuration. Two signal lines (IEI and IEO) are provided in CTC devices to form this system daisy chain. The device closest to the CPU has the highest priority: within the CTC, interrupt priority is predetermined by channel number, with channel 0 having highest priority down to channel 3 which has the lowest priority. The purpose of a CTC-generated interrupt, as with any other peripheral device, is to force the CPU to execute an interrupt service routine. According to Z80 system interrupt protocol, lower priority devices or channels may not interrupt higher priority devices or channels that have already interrupted and have not had their interrupt service routines completed. However, high priority devices or channels may interrupt the servicing of lower priority devices or channels.

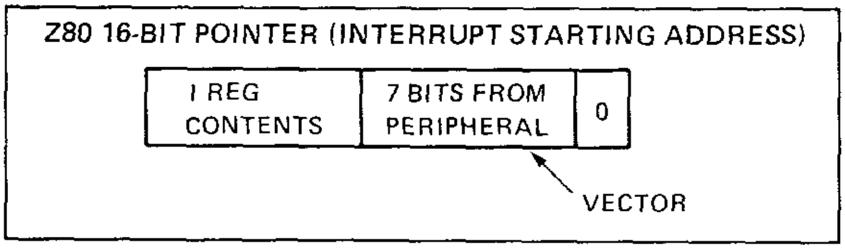
A CTC channel may be programmed to request an interrupt every time its Down Counter reaches a count of zero. (To utilize this feature requires that the CPU be programmed for interrupt mode 2.) Some time after the interrupt request, the CPU will send out an interrupt acknowledge, and the CTC's Interrupt Control Logic will determine the highest-priority channel which is requesting an interrupt within the CTC device. Then if the CTC's IEI input is active, indicating that it has priority within the system daisy chain, it will place an 8-bit Interrupt Vector on the system data bus. The high-order 5 bits of this vector will have been written to the CTC earlier as part of the CTC initial programming process; the next two bits will be provided by the CTC's Interrupt Control Logic as a binary code corresponding to the highest-priority channel requesting an interrupt; finally the low-order bit of the vector will always be zero according to a convention described below.

INTERRUPT VECTOR

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
V_7	V_6	V_5	V_4	V_3	X	X	0

D_2	D_1	
0	0	CHANNEL 0
0	1	CHANNEL 1
1	0	CHANNEL 2
1	1	CHANNEL 3

This interrupt vector is used to form a pointer to a location in memory where the address of the interrupt service routine is stored in a table. The vector represents the least significant 8 bits, while the CPU reads the contents of the I register to provide the most significant 8-bits of the 16-bit pointer. The address in memory pointed to will contain the low-order byte, and the next highest address will contain the high-order byte of an address which in turn contains the first opcode of the interrupt service routine. Thus in mode 2, a single 8-bit vector stored in an interrupting CTC can result in an indirect call to any memory location.



There is a Z80 system convention that all addresses in the interrupt service routine table should have their low-order byte in an even location in memory, and their high-order byte in the next highest location in memory, which will always be odd so that the least significant bit of any interrupt vector will always be even. Hence the least significant bit of any interrupt vector will always be zero.

The RETI instruction is used at the end of any interrupt service routine to initialize the daisy chain enable line IEO for proper control of nested priority interrupt handing. The CTC monitors the system data bus and decodes this instruction when it occurs. Thus the CTC channel control logic will know when the CPU has completed servicing an interrupt, without any further communication with the CPU being necessary.

3.0 CTC PIN DESCRIPTION

A diagram of the Z80-CTC pin configuration is shown in figure 3.0-1. This section describes the function of each pin.

D7 - D0

Z80-CPU Data Bus (bi-directional, tri-state)

This bus is used to transfer all data and command words between the Z80-CPU and the Z80-CTC. There are 8 bits on this bus, of which D0 is the least significant.

CS1 – CS0

Channel Select (input, active high)

These pins form a 2-bit binary address code for selecting one of the four independent CTC channels for an I/O Write or Read. (See truth table below.)

	CS1	CS0
Ch 0	0	0
Ch 1	0	1
Ch 2	1	0
Ch 3	1	1

$\overline{\text{CE}}$

Chip Enable (input, active low)

A low level on this pin enables the CTC to accept control words, Interrupt Vectors, or time constant data words from the Z80 Data Bus during an I/O Write cycle, or to transmit the contents of the Down Counter to the CPU during an I/O Read cycle. In most applications this signal is decoded from the 8 least significant bits of the address bus for any of the four I/O port addresses that are mapped to the four Counter/Timer Channels.

Clock (Φ)

System Clock (input)

This single-phase clock is used by the CTC to synchronize certain signals internally.

$\overline{\text{M1}}$

Machine Cycle One Signal from CPU (input, active low)

When $\overline{\text{M1}}$ is active and the $\overline{\text{RD}}$ signal is active, the CPU is fetching an instruction from memory. When $\overline{\text{M1}}$ is active and the $\overline{\text{IORQ}}$ signal is active, the CPU is acknowledging an interrupt, alerting the CTC to place an Interrupt Vector on the Z80 Data Bus if it has daisy chain priority and one of its channels has requested an interrupt

$\overline{\text{IORQ}}$

Input/Output Request from CPU (input, active low)

The $\overline{\text{IORQ}}$ signal is used in conjunction with the $\overline{\text{CE}}$ and $\overline{\text{RD}}$ signals to transfer data and Channel Control Words between the Z80-CPU and the CTC. During a CTC Write Cycle, $\overline{\text{IORQ}}$ and $\overline{\text{CE}}$ must be true and $\overline{\text{RD}}$ false. The CTC does not receive a specific write signal, instead generating its own internally from the inverse of a valid $\overline{\text{RD}}$ signal. In a CTC Read Cycle, $\overline{\text{IORQ}}$, $\overline{\text{CE}}$ and $\overline{\text{RD}}$ must be active to place the contents of the Down Counter on the Z80 Data Bus. If $\overline{\text{IORQ}}$ and $\overline{\text{M1}}$ are both true, the CPU is acknowledging an interrupt request, and the highest-priority interrupting channel will place its Interrupt Vector on the Z80 Data Bus.

3.0 CTC PIN DESCRIPTION (CONT'D)

$\overline{\text{RD}}$

Read Cycle Status from the CPU (input, active low)

The $\overline{\text{RD}}$ signal is used in conjunction with the $\overline{\text{IORQ}}$ and $\overline{\text{CE}}$ signals to transfer data and Channel Control Words between the Z80-CPU and the CTC. During a CTC Write Cycle. $\overline{\text{IORQ}}$ and $\overline{\text{CE}}$ must be true and $\overline{\text{RD}}$ false. The CTC does not receive a specific write signal, instead generating its own internally from the inverse of a valid $\overline{\text{RD}}$ signal. In a CTC Read Cycle, $\overline{\text{IORQ}}$, $\overline{\text{CE}}$ and $\overline{\text{RD}}$ must be active to place the contents of the Down Counter on the Z80 Data Bus.

IEI

Interrupt Enable In (input, active high)

This signal is used to help form a system-wide interrupt daisy chain which establishes priorities when more than one peripheral device in the system has interrupting capability. A high level on this pin indicates that no other interrupting devices of higher priority in the daisy chain are being serviced by the Z80-CPU.

IEO

Interrupt Enable Out (output, active high)

The IEO signal, in conjunction with IEI, is used to form a system-wide interrupt priority daisy chain. IEO is high only if IEI is high and the CPU is not servicing an interrupt from any CTC channel. Thus this signal blocks lower priority devices from interrupting while a higher priority interrupting device is being serviced by the CPU.

$\overline{\text{INT}}$

Interrupt Request (output, open drain, active low)

This signal goes true when any CTC channel which has been programmed to enable interrupts has a zero-count condition in its Down Counter.

$\overline{\text{RESET}}$

Reset (input, active low)

This signal stops all channels from counting and resets channel interrupt enable bits in all control registers, thereby disabling CTC-generated interrupts. The ZC/TO and $\overline{\text{INT}}$ outputs go to their inactive states, IEO reflects IEI, and the CTC's data bus output drivers go to the high impedance state.

CLK/TRG3 – CLK/TRG0

External Clock/Timer Trigger (input, user-selectable active high or low)

There are four CLK/TRG pins, corresponding to the four independent CTC channels. In the Counter Mode, every active edge on this pin decrements the Down Counter. In the Timer Mode, an active edge on this pin initiates the timing function. The user may select the active edge to be either rising or falling.

ZC/TO2 – AC/TO0

Zero Count/Timeout (output, active high)

There are three ZC/TO pins, corresponding to CTC channels 2 through 0. (Due to package pin limitations channel 3 has no ZC/TO pin.) In either Counter Mode or Timer Mode, when the Down Counter decrements to zero an active high going pulse appears at this pin.

3.0 CTC PIN DESCRIPTION

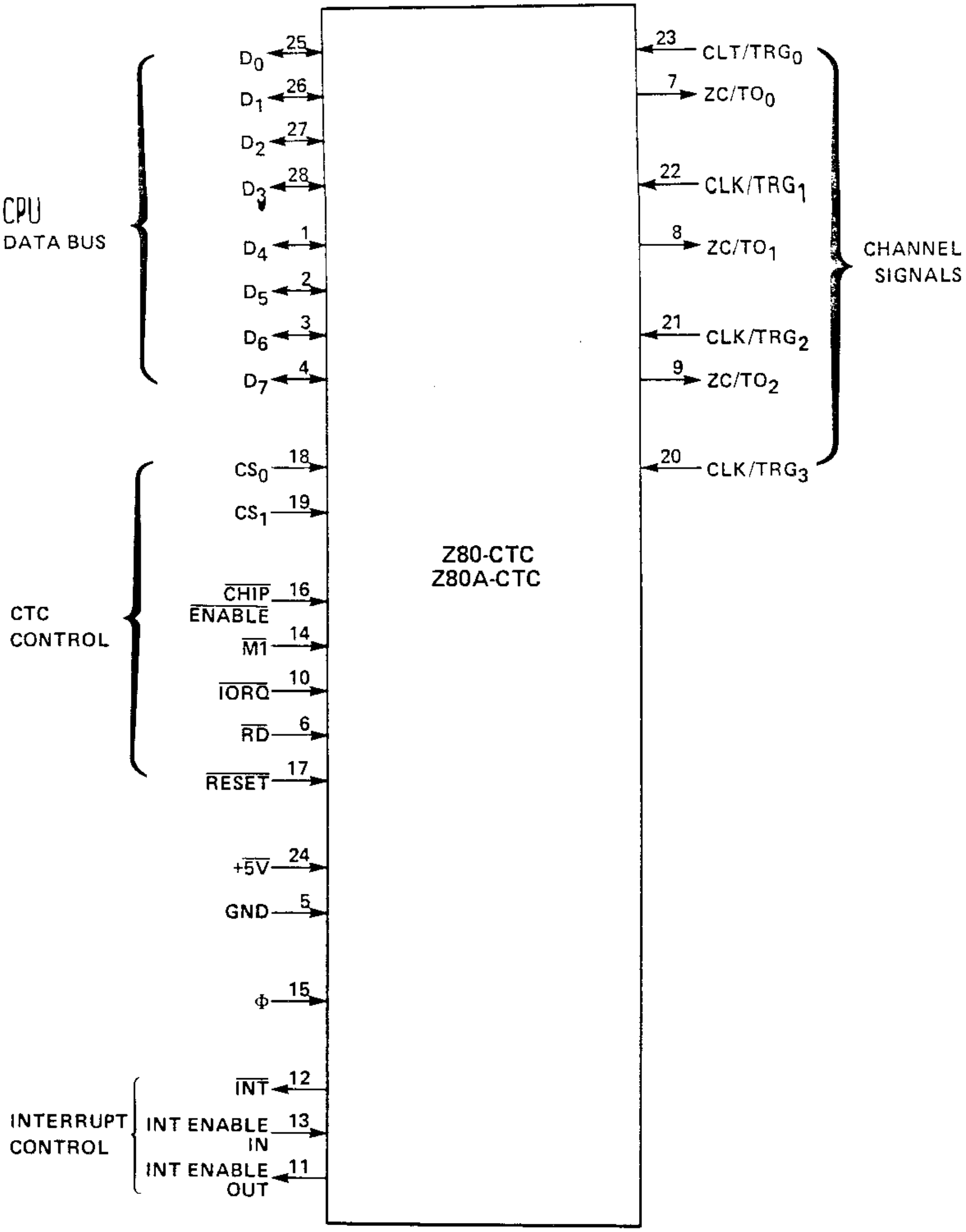


FIGURE 3.0-1
CTC PIN CONFIGURATION

4.0 CTC OPERATING MODES

At power-on, the Z80-CTC state is undefined. Asserting $\overline{\text{RESET}}$ puts the CTC in a known state. Before any channel can begin counting or timing, a Channel Control Word and a time constant data word must be written to the appropriate registers of that channel. Further, if any channel has been programmed to enable interrupts, an Interrupt Vector word must be written to the CTC's Interrupt Control Logic. (For further details, refer to section 5.0: "CTC Programming.") When the CPU has written all of these words to the CTC all active channels will be programmed for immediate operation in either the Counter Mode or the Timer Mode.

4.1 CTC COUNTER MODE

In this mode the CTC counts edges of the CLK/TRG input. The Counter Mode is programmed for a channel when its Channel Control Word is written with bit 6 set. The Channel's External Clock (CLK/TRG) input is monitored for a series of triggering edges; after each, in synchronization with the next rising edge of Φ (the System Clock), the Down Counter (which was initialized with the time constant data word at the start of any sequence of down-counting) is decremented. Although there is no set-up time requirement between the triggering edge of the External Clock and the rising edge of Φ, (Clock), the Down Counter will not be decremented until the following Φ pulse. (See the parameter ts(CK) in section 8.3: "A.C. Characteristics.") A channels's External Clock input is pre-programmed by bit 4 of the Channel Control Word to trigger the decrementing sequence with either a high or a low going edge.

In any of Channels 0, 1, or 2, when the Down Counter is successively decremented from the original time constant until finally it reaches zero, the Zero Count (ZC/TO) output pin for that channel will be pulsed active (high). (However, due to package pin limitations, channel 3 does not have this pin and so may only be used in applications where this output pulse is not required.) Further, if the channel has been so pre-programmed by bit 7 of the Channel Control Word, an interrupt request sequence will be generated. (For more details, see section 7.0: "CTC Interrupt Servicing.")

As the above sequence is proceeding, the zero count condition also results in the automatic reload of the Down Counter with the original time constant data word in the Time Constant Register. There is no interruption in the sequence of continued down-counting. If the Time Constant Register is written to with a new time constant data word while the Down Counter is decrementing, the present count will be completed before the new time constant will be loaded into the Down Counter.

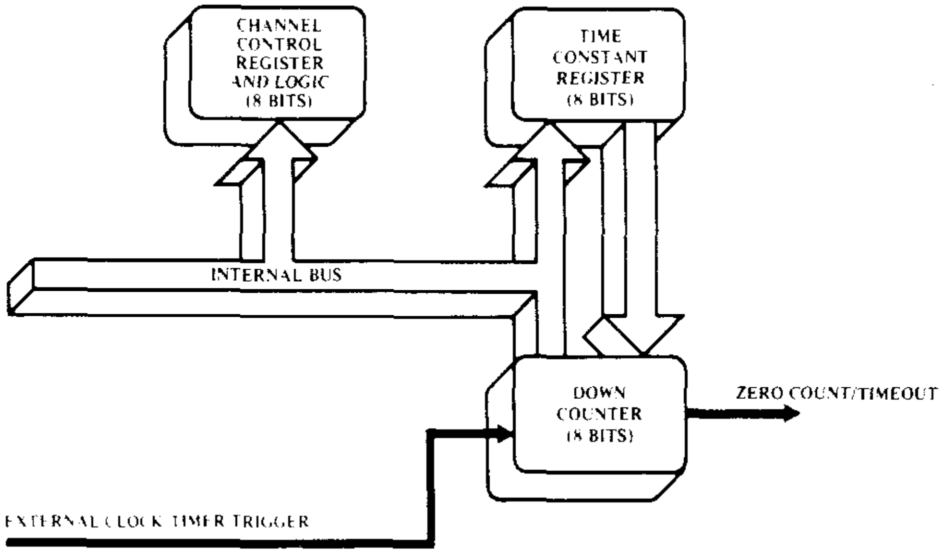

CHANNEL CONTROL REGISTER AND LOGIC (8 BITS)
TIME CONSTANT REGISTER (8 BITS)
INTERNAL BUS
DOWN COUNTER (8 BITS)
ZERO COUNT/TIMEOUT
EXTERNAL CLOCK TIMER TRIGGER

FIGURE 4.1-0
CHANNEL—COUNTER MODE

4.2 CTC TIMER MODE

In this mode the CTC generates timing intervals that are an integer value of the system clock period. The Timer Mode is programmed for a channel when its Channel Control Word is written with bit 6 reset. The channel then may be used to measure intervals of time based on the System Clock period. The System Clock is fed through two successive counters, the Prescaler and the Down Counter. Depending on the pre-programmed bit 5 in the Channel Control Word, the Prescaler divides the System Clock by a factor of either 16 or 256. The output of the Prescaler is then used as a clock to decrement the Down Counter, which may be pre-programmed with any time constant integer between 1 and 256. As in the Counter Mode, the time constant is automatically reloaded into the Down Counter at each zero-count condition, and counting continues. Also at zero-count, the channel's Time Out (ZC/TO) output (which is the output of the Down Counter) is pulsed, resulting in a uniform pulse train of precise period given by the product.

$$t_c * P * TC$$

where t_c is the System Clock period, P is the Prescaler factor of 16 or 256 and TC is the pre-programmed time constant.

Bit 3 of the Channel Control Word is pre-programmed to select whether timing will be automatically initiated, or whether it will be initiated with a triggering edge at the channel's Timer Trigger (CLK/TRG) input. If bit 3 is reset the timer automatically begins operation at the start of the CPU cycle following the I/O Write machine cycle that loads the time constant data word to the channel. If bit 3 is set the timer begins operation on the second succeeding rising edge of Φ after the Timer Trigger edge following the loading of the time constant data word. If no time constant data word is to follow then the timer begins operation on the second succeeding rising edge of Φ after the Timer Trigger edge following the control word write cycle. Bit 4 of the Channel Control Word is pre-programmed to select whether the Timer Trigger will be sensitive to a rising or falling edge. Although there is no set-up requirement between the active edge of the Timer Trigger and the next rising edge of Φ. If the Timer Trigger edge occurs closer than a specified minimum set-up time to the rising edge of Φ, the Down Counter will not begin decrementing until the following rising edge of Φ. (See the parameter ts(TR) in section 8.3: "A.C. Characteristics".)

If bit 7 in the Channel Control Word is set, the zero-count condition in the Down Counter, besides causing a pulse at the channel's Time Out pin, will be used to initiate an interrupt request sequence. (For more details, see section 7.0: "CTC Interrupt Servicing.".)

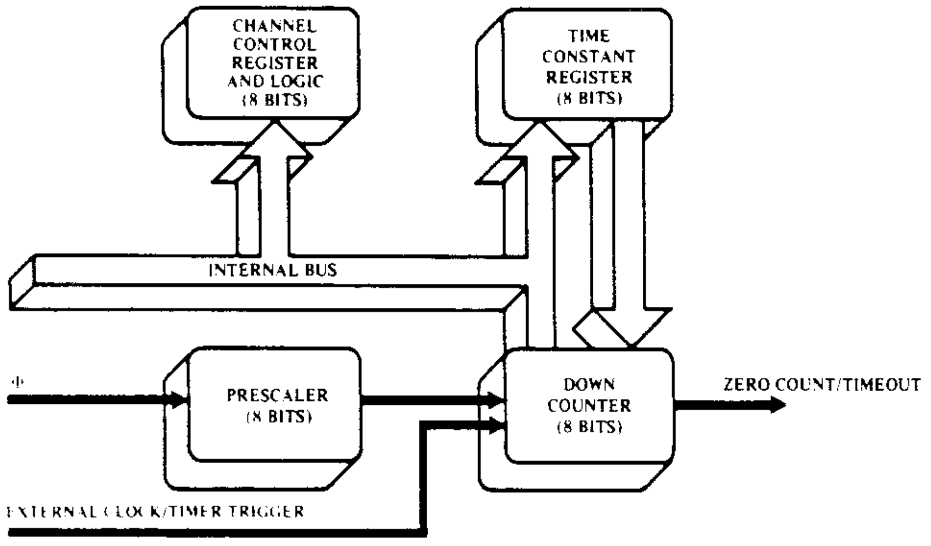

CHANNEL CONTROL REGISTER AND LOGIC (8 BITS)
TIME CONSTANT REGISTER (8 BITS)
INTERNAL BUS
Φ
PRESCALER (8 BITS)
DOWN COUNTER (8 BITS)
ZERO COUNT/TIMEOUT
EXTERNAL CLOCK/TIMER TRIGGER

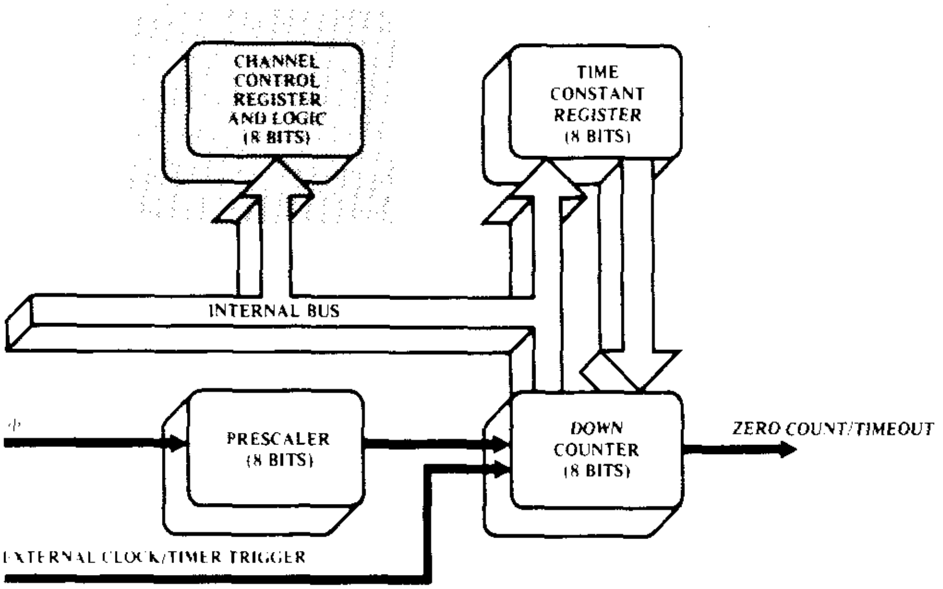
FIGURE 4.2-0
CHANNEL–TIMER MODE

5.0 CTC PROGRAMMING

Before a Z80-CTC channel can begin counting or timing operations, a Channel Control Word and a Time Constant data word must be written to it by the CPU. These words will be stored in the Channel Control Register and the Time Constant Register of that channel. In addition, if any of the four channels have been programmed with bit 7 of their Channel Control Words to enable interrupts, an Interrupt Vector must be written to the appropriate register in the CTC. Due to automatic features in the Interrupt Control Logic, one pre-programmed Interrupt Vector suffices for all four channels.

5.1 LOADING THE CHANNEL CONTROL REGISTER

To load a Channel Control Word, the CPU performs a normal I/O Write sequence to the port address corresponding to the desired CTC channel. Two CTC input pins, namely CS0 and CS1, are used to form a 2-bit binary address to select one of four channels within the device. (For a truth table, see section 2.2.1: "The Channel Control Register and Logic".) In many system architectures, these two input pins are connected to Address Bus lines A0 and A1, respectively, so that the four channels in a CTC device will occupy contiguous I/O port addresses. A word written to a CTC channel will be interpreted as a Channel Control Word, and loaded into the Channel Control Register, its bit 0 is a logic 1. The other seven bits of this word select operating modes and conditions as indicated in the diagram below. Following the diagram the meaning of each bit will be discussed in detail.


CHANNEL CONTROL REGISTER AND LOGIC (8 BITS)
TIME CONSTANT REGISTER (8 BITS)
INTERNAL BUS
φ
PRESCALER (8 BITS)
DOWN COUNTER (8 BITS)
ZERO COUNT/TIMEOUT
EXTERNAL CLOCK/TIMER TRIGGER

5.1 LOADING THE CHANNEL CONTROL REGISTER (CONT'D)

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
INTERRUPT ENABLE	MODE	RANGE	SLOPE	TRIGGER	LOAD TIME CONSTANT	RESET	1

RANGE: USED IN TIMER MODE ONLY

TRIGGER: USED IN TIMER MODE ONLY

Bit 7 = 1

The channel is enabled to generate an interrupt request sequence every time the Down Counter reaches a zero-count condition. To set this bit to 1 in any of the four Channel Control Registers necessitates that an Interrupt Vector also be written to the CTC before operation begins. Channel interrupts may be programmed in either Counter Mode or Timer Mode. If an updated Channel Control Word is written to a channel already in operation, with bit 7 set, the interrupt enable selection will not be retroactive to a preceding zero-count condition.

Bit 7 = 0

Channel interrupts disabled.

Bit 6 = 1

Counter Mode selected. The Down Counter is decremented by each triggering edge of the External Clock (CLK/TRG) input. The Prescaler is not used.

Bit 6 = 0

Timer Mode selected. The Prescaler is clocked by the System Clock Φ, and the output of the Prescaler in turn clocks the Down Counter. The output of the Down Counter (the channel's ZC/TO output) is a uniform pulse train of period given by the product

$$t_c * P * TC$$

where t_c is the period of System Clock Φ, P is the Prescaler factor of 16 or 256, and TC is the time constant data word.

Bit 5 = 1

(Defined for Timer Mode only.) Prescaler factor is 256.

Bit 5 = 0

(Defined for Timer Mode only.) Prescaler factor is 16.

5.1 LOADING THE CHANNEL CONTROL REGISTER (CONT'D)

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
INTERRUPT ENABLE	MODE	RANGE	SLOPE	TRIGGER	LOAD TIME CONSTANT	RESET	1
		USED IN TIMER MODE ONLY		USED IN TIMER MODE ONLY			

Bit 4 = 1

TIMER MODE – positive edge trigger starts timer operation.

COUNTER MODE – positive edge decrements the down counter.

Bit 4 = 0

TIMER MODE – negative edge trigger starts timer operation.

COUNTER MODE – negative edge decrements the down counter.

Bit 3 = 1

Timer Mode Only – External trigger is valid for starting timer operation after rising edge of T_2 of the machine cycle following the one that loads the time constant. The Prescaler is decremented 2 clock cycles later if the setup time is met, otherwise 3 clock cycles.

Bit 3 = 0

Timer Mode Only – Timer begins operation on the rising edge of T_2 of the machine cycle following the one that loads the time constant.

Bit 2 = 1

The time constant data word for the Time Constant Register will be the next word written to this channel. If an updated Channel Control Word and time constant data word are written to a channel while it is already in operation, the Down Counter will continue decrementing to zero before the new time constant is loaded into it.

Bit 2 = 0

No time constant data word for the Time Constant Register should be expected to follow. To program bit 2 to this state implies that this Channel Control Word is intended to update the status of a channel already in operation, since a channel will not operate without a correctly programmed data word in the Time Constant Register, and a set bit 2 in this Channel Control Word provides the only way of writing to the Time Constant Register.

Bit 1 = 1

Reset channel. Channel stops counting or timing. This is not a stored condition. Upon writing into this bit a reset pulse discontinues current channel operation, however, none of the bits in the channel control register are changed. If both bit 2 = 1 and bit 1 = 1 the channel will resume operation upon loading a time constant.

Bit 1 = 0

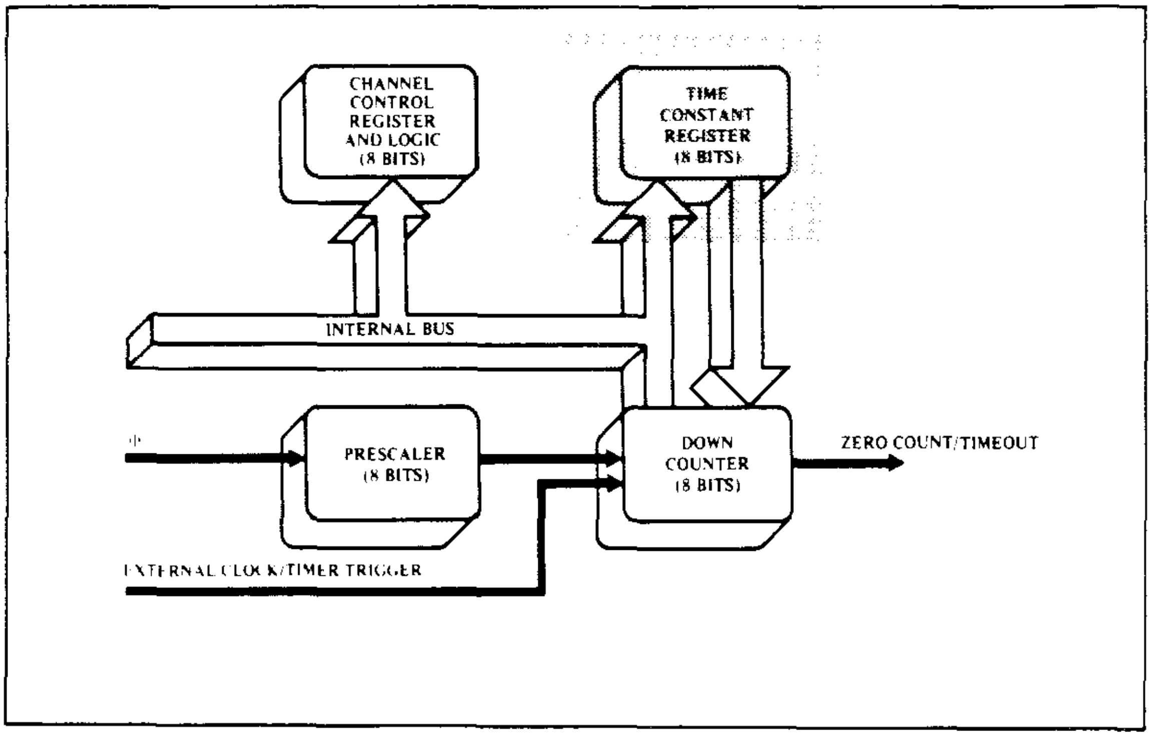
Channel continues current operation.

5.2 LOADING THE TIME CONSTANT REGISTER

A channel may not begin operation in either Timer Mode or Counter Mode unless a time constant data word is written into the Time Constant Register by the CPU. This data word will be expected on the next I/O Write to this channel following the I/O Write of the Channel Control Word, provided that bit 2 of the Channel Control Word is set. The time constant data word may be any integer value in the range 1-256. If all eight bits in this word are zero, it is interpreted as 256. If a time constant data word is loaded to a channel already in operation, the Down Counter will continue decrementing to zero before the new time constant is loaded from the Time Constant Register to the Down Counter.

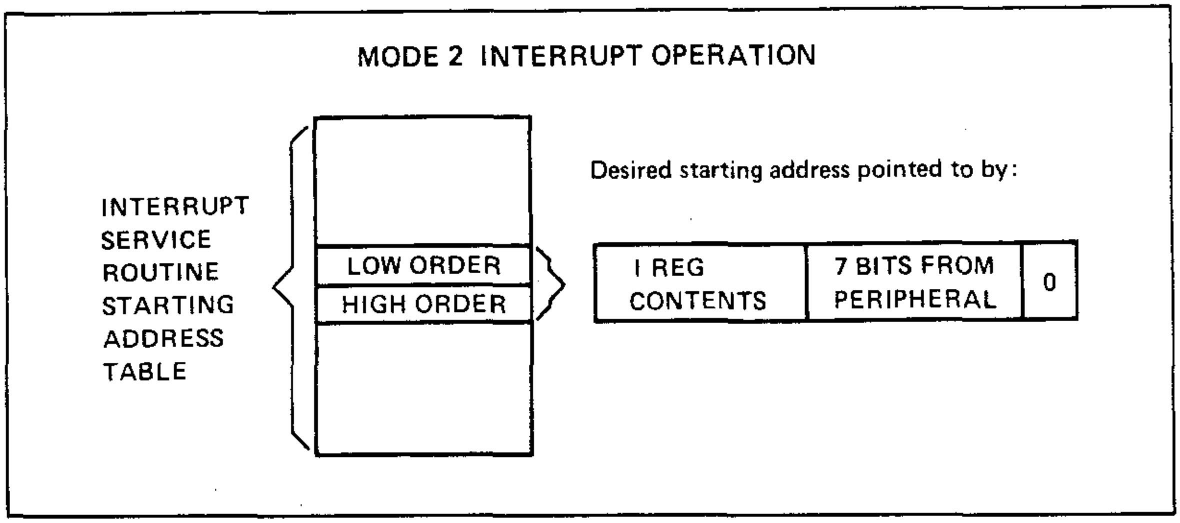
TIME CONSTANT REGISTER

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
TC_7	TC_6	TC_5	TC_4	TC_3	TC_2	TC_1	TC_0
MSB							LSB

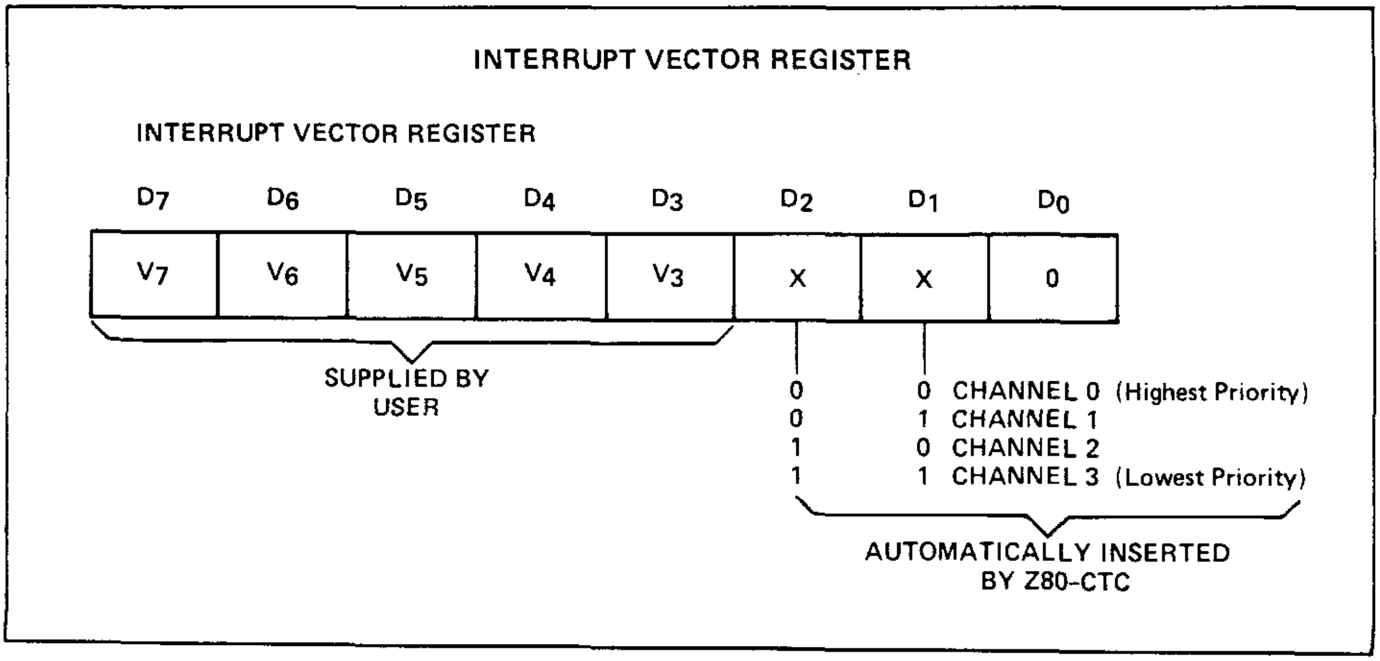

CHANNEL CONTROL REGISTER AND LOGIC (8 BITS)
TIME CONSTANT REGISTER (8 BITS)
INTERNAL BUS
φ
PRESCALER (8 BITS)
DOWN COUNTER (8 BITS)
ZERO COUNT/TIMEOUT
EXTERNAL CLOCK/TIMER TRIGGER

5.3 LOADING THE INTERRUPT VECTOR REGISTER

The Z80-CTC has been designed to operate with the Z80-CPU programmed for mode 2 interrupt response. Under the requirements of this mode, when a CTC channel requests an interrupt and is acknowledged, a 16-bit pointer must be formed to obtain a corresponding interrupt service routine starting address from a table in memory. The upper 8 bits of this pointer are provided by the CPU's I register, and the lower 8 bits of the pointer are provided by the CTC in the form of an Interrupt Vector unique to the particular channel that requested the interrupt. (For further details, see section 7.0: "CTC Interrupt Servicing".)


MODE 2 INTERRUPT OPERATION
INTERRUPT SERVICE ROUTINE STARTING ADDRESS TABLE
LOW ORDER
HIGH ORDER
Desired starting address pointed to by:
I REG CONTENTS
7 BITS FROM PERIPHERAL
0

The high order 5 bits of this Interrupt Vector must be written to the CTC in advance as part of the initial programming sequence. To do so, the CPU must write to the I/O port address corresponding to the CTC channel 0, just as it would if a Channel Control Word were being written to that channel, except that bit 0 of the word being written must contain a 0. (As explained above in section 5.1, if bit 0 of a word written to a channel were set to 1, the word would be interpreted as a Channel Control Word, so a 0 in bit 0 signals the CTC to load the incoming word into the Interrupt Vector Register.) Bits 1 and 2, however, are not used when loading this vector. At the time when the interrupting channel must place the Interrupt Vector on the Z80 Data Bus, the Interrupt Control Logic of the CTC automatically supplies a binary code in bits 1 and 2 identifying which of the four CTC channels is to be serviced.


INTERRUPT VECTOR REGISTER
INTERRUPT VECTOR REGISTER
D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0
V_7 V_6 V_5 V_4 V_3 X X 0
SUPPLIED BY USER
0 0 CHANNEL 0 (Highest Priority)
0 1 CHANNEL 1
1 0 CHANNEL 2
1 1 CHANNEL 3 (Lowest Priority)
AUTOMATICALLY INSERTED BY Z80-CTC

6.0 CTC TIMING

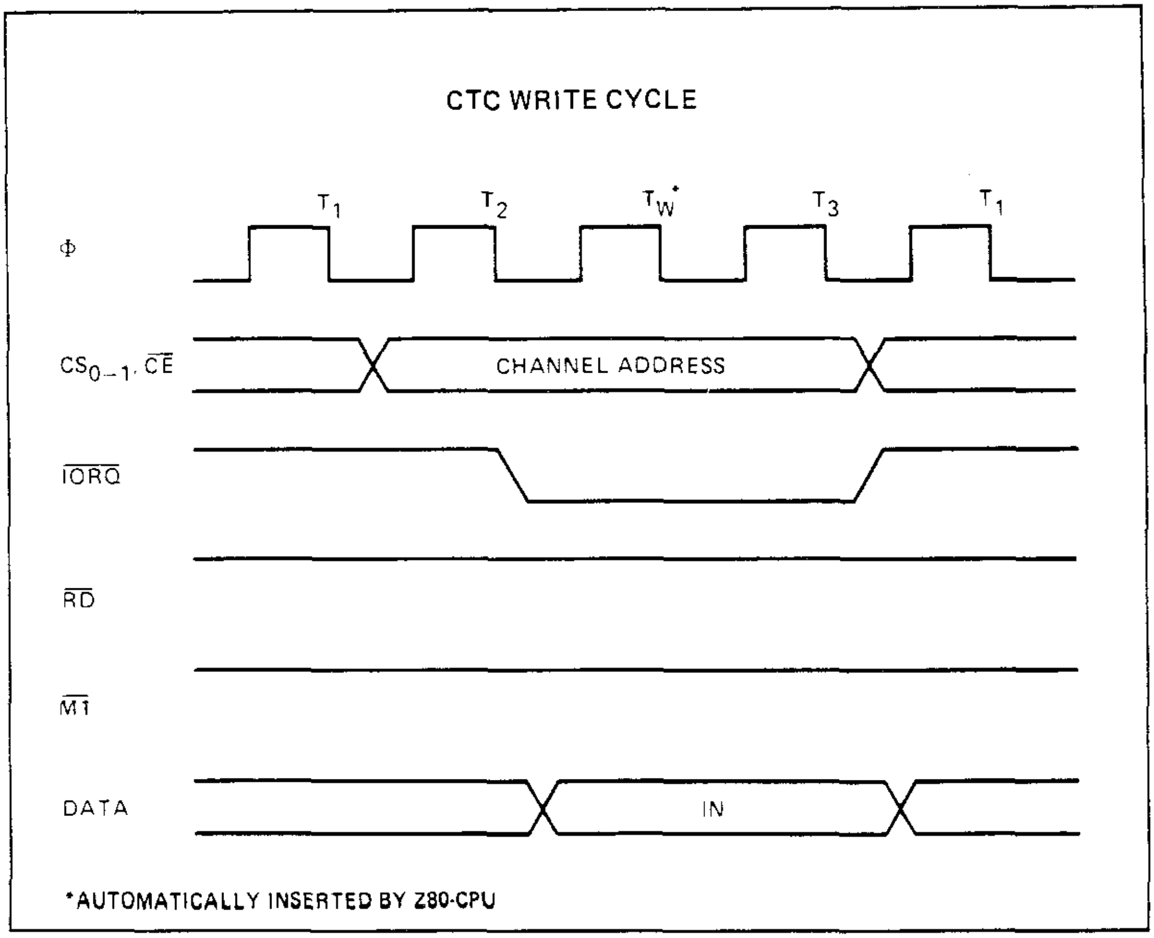
This section illustrates the timing relationships of the relevant CTC pins for the following types of operation: writing a word to the CTC, reading a word from the CTC, counting, and timing. Elsewhere in this manual may be found timing diagrams relating to interrupt servicing (section 7.0) and an A.C. Timing Diagram which quantitatively specifies the timing relationships (section 8.4).

6.1 CTC WRITE CYCLE

Figure 6.0-1 illustrates the timing associated with the CTC Write Cycle. This sequence is applicable to loading either a Channel Control Word, an Interrupt Vector, or a time constant data word.

In the sequence shown, during clock cycle T_1, the Z80-CPU prepares for the Write Cycle with a false (high) signal at CTC input pin $\overline{RD}$ (Read). Since the CTC has no separate Write signal input, it generates its own internally from the false $\overline{RD}$ input. Later, during clock cycle T_2, the Z80-CPU initiates the Write Cycle with true (low) signals at CTC input pins $\overline{IORQ}$ (I/O Request) and $\overline{CE}$ (Chip Enable). (Note: $\overline{M1}$ must be false to distinguish the cycle from an interrupt acknowledge.) Also at this time a 2-bit binary code appears at CTC inputs CS1 and CS0 (Channel Select 1 and 0), specifying which of the four CTC channels is being written to, and the word being written appears on the Z80 Data Bus. Now everything is ready for the word to be latched into the appropriate CTC internal register in synchronization with the rising edge beginning clock cycle T_3. No additional wait states are allowed.

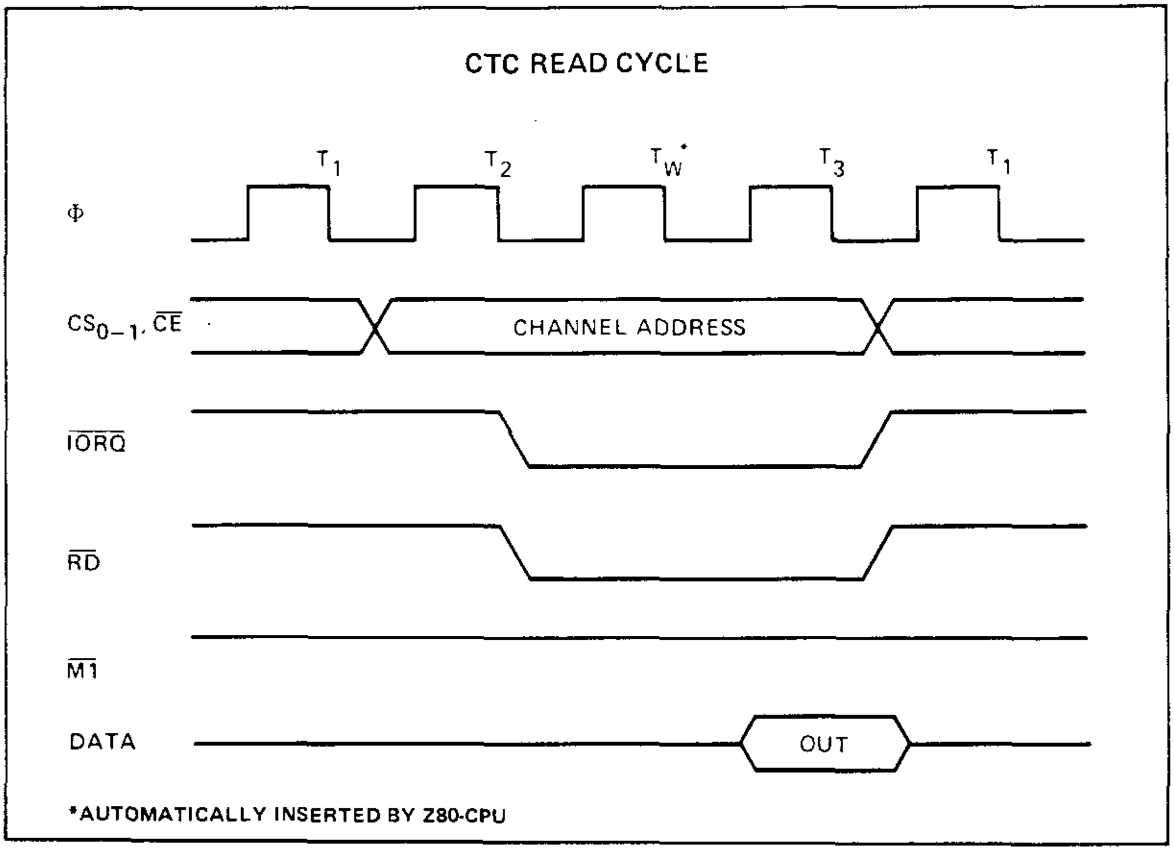
CTC WRITE CYCLE



*AUTOMATICALLY INSERTED BY Z80-CPU

6.2 CTC READ CYCLE

Figure 6.0-2 illustrates the timing associated with the CTC Read Cycle. This sequence is used any time the CPU reads the current contents of the Down Counter. During clock cycle T_2, the Z80-CPU initiates the Read Cycle with true signals at input pins RD (Read), IORQ (I/O Request), and CE (Chip Enable). Also at this time a 2-bit binary code appears at CTC inputs CS1 and CS0 (Channel Select 1 and 0), specifying which of the four CTC channels is being read from. (Note: M1 must be false to distinguish the cycle from an interrupt acknowledge.) On the rising edge of the cycle T_3 the valid contents of the Down Counter as of the rising edge of cycle T_2 will be available on the Z80 Data Bus. No additional wait states are allowed.



CTC READ CYCLE

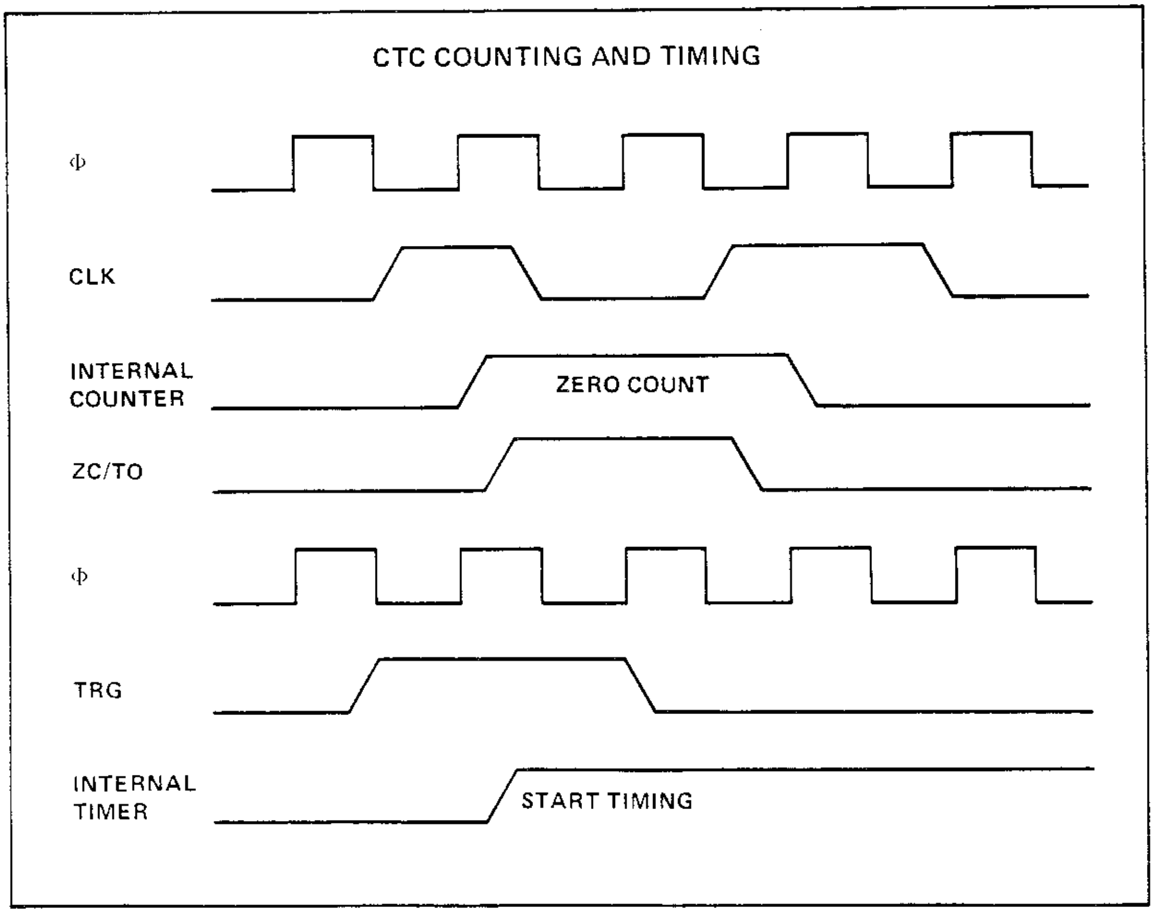
*AUTOMATICALLY INSERTED BY Z80-CPU

6.3 CTC COUNTING AND TIMING

Figure 6.0-3 illustrates the timing diagram for the CTC Counting and Timing Modes.

In the Counter Mode, the edge (rising edge is active in this example) from the external hardware connected to pin CLK/TRG decrements the Down Counter in synchronization with the System Clock Φ. As specified in the A.C. Characteristics (Section 9.1) this CLK/TRG pulse must have a minimum width and the minimum period must not be less than twice the system clock period. Although there is no set-up time requirement between the active edge of the CLK/TRG and the rising edge of Φ if the CLK/TRG edge occurs closer than a specified minimum time, the decrement of the Down Counter will be delayed one cycle of Φ. Immediately after the decrement of the Down Counter, 1 to 0, the ZC/TO output is pulsed true.

In the Timer Mode, a pulse trigger (user-selectable as either active high or active low) at the CLK/TRG pin enables timing function on the second succeeding rising edge of Φ. As in the Counter Mode, the triggering pulse is detected asynchronously and must have a minimum width. The timing function is initiated in syncronization with Φ, and a minimum set-up time is required between the active edge of the CLK/TRG and the next rising edge of Φ. If the CLK/TRG edge occurs closer than this, the initiation of the timer function will be delayed one cycle of Φ.


CTC COUNTING AND TIMING

7.0 CTC INTERRUPT SERVICING

Each CTC channel may be individually programmed to request an interrupt every time its Down Counter reaches a count of zero. The purpose of a CTC-generated interrupt, as for any other peripheral device, is to force the CPU to execute an interrupt service routine. To utilize this feature the Z80-CPU must be programmed for mode 2 interrupt response. Under the requirements of this mode, when a CTC channel requests an interrupt and is acknowledged, a 16-bit pointer must be formed to obtain a corresponding interrupt service routine starting address from a table in memory. The lower 8 bits of the pointer are provided by the CTC in the form of an Interrupt Vector unique to the particular channel that requested the interrupt. (For further details, refer to chapter 8.0 of the Z80-CPU Technical Manual.)

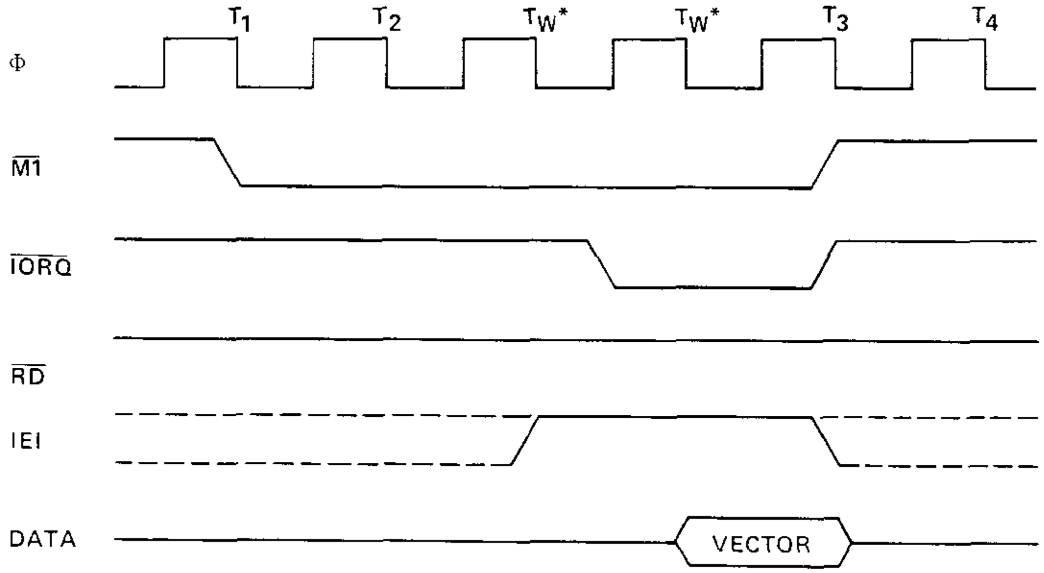
The CTC's Interrupt Control Logic insures that it acts in accordance with Z80 system interrupt protocol for nested priority interrupt and proper return from interrupt. The priority of any system device is determined by its physical location in a daisy chain configuration. Two signal lines (IEI and IEO) are provided in the CTC and all Z80 peripheral devices to form the system daisy chain. The device closest to the CPU has the highest priority; within the CTC, interrupt priority is predetermined by channel number, with channel 0 having highest priority. According to Z80 system interrupt protocol, low priority devices or channels may not interrupt higher priority devices or channels that have already interrupted and not had their interrupt service routines completed. However, high priority devices or channels may interrupt the servicing of lower priority devices or channels. (For further details, see section 2.3: "Interrupt Control Logic".)

Sections 7.1 and 7.2 below describe the nominal timing relationships of the relevant CTC pins for the Interrupt Acknowledge Cycle and the Return from Interrupt Cycle. Section 7.3 below discusses a typical example of daisy chain interrupt servicing.

7.1 INTERRUPT ACKNOWLEDGE CYCLE

Figure 7.0-1 illustrates the timing associated with the Interrupt Acknowledge Cycle. Some time after an interrupt is requested by the CTC, the CPU will send out an interrupt acknowledge ($\overline{M1}$ and $\overline{IORQ}$). To insure that the daisy chain enable lines stabilize, channels are inhibited from changing their interrupt request status when $\overline{M1}$ is active. $\overline{M1}$ is active about two clock cycles earlier than $\overline{IORQ}$, and $\overline{RD}$ is false to distinguish the cycle from an instruction fetch. During this time the interrupt logic of the CTC will determine the highest priority channel requesting an interrupt. If the CTC Interrupt Enable Input (IEI) is active, then the highest priority interrupting channel within the CTC places its Interrupt Vector onto the Data Bus when $\overline{IORQ}$ goes active. Two wait states (T_W*) are automatically inserted at this time to allow the daisy chain to stabilize. Additional wait states may be added.

INTERRUPT ACKNOWLEDGE CYCLE



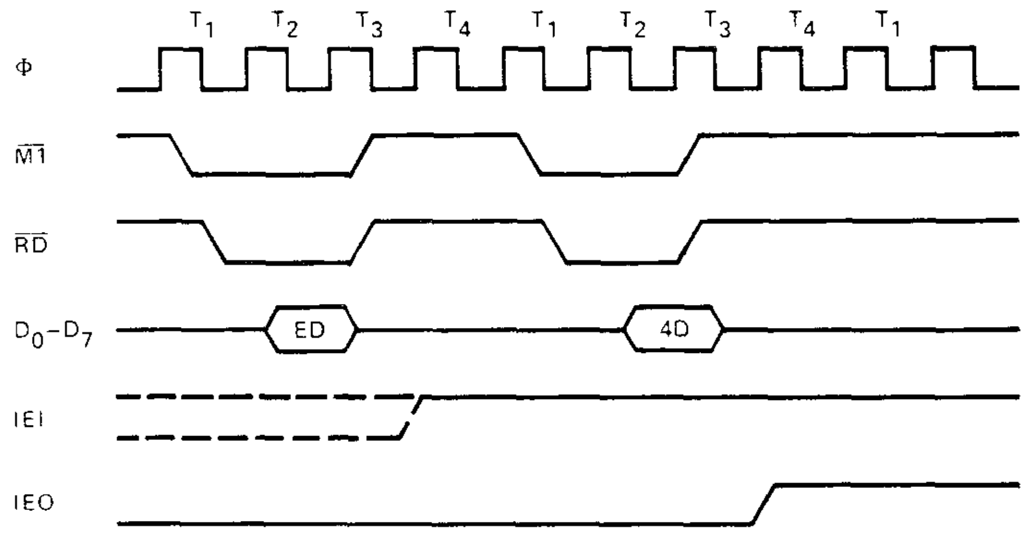
*AUTOMATICALLY INSERTED BY Z80-CPU

7.2 RETURN FROM INTERRUPT CYCLE

Figure 7.0-2 illustrates the timing associated with the RETI Instruction. This instruction is used at the end of an interrupt service routine to initialize the daisy chain enable lines for proper control of nested priority interrupt handling. The CTC decodes the two-byte RETI code internally and determines whether it is intended for a channel being serviced.

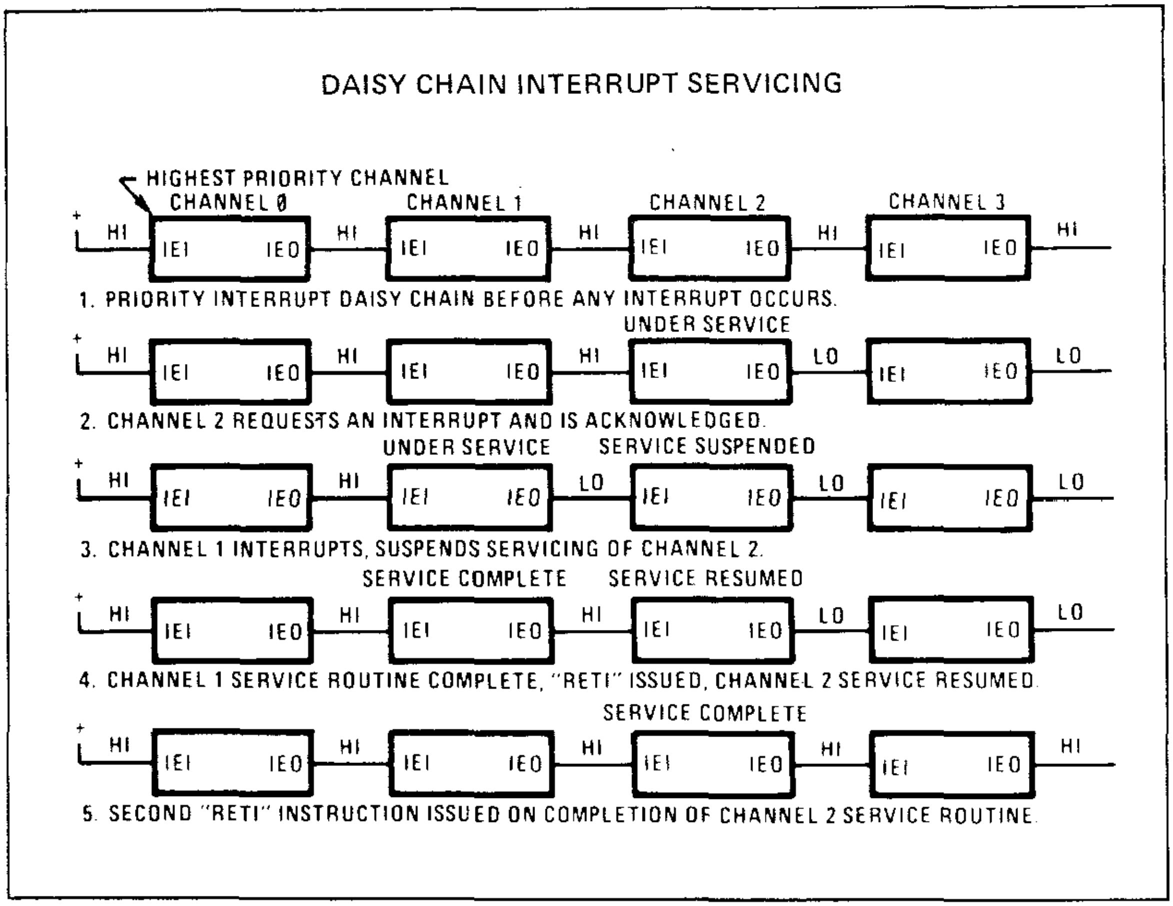
When several Z80 peripheral chips are in the daisy chain IEI will become active on the chip currently under service when an EDH opcode is decoded. If the following opcode is 4DH, the peripheral being serviced will be re-initialized and its IEO will become active. Additional wait states are allowed.

RETURN FROM INTERRUPT CYCLE



7.3 DAISY CHAIN INTERRUPT SERVICING

Figure 7.0-3 illustrates a typical nested interrupt sequence which may occur in the CTC. In this example, channel 2 interrupts and is granted service. While this channel is being serviced, higher priority channel 1 interrupts and is granted service. The service routine for the higher priority channel is completed, and a RETI instruction (see section 7.2 for further details) is executed to signal the channel that its routine is complete. At this time, the service routine of the lower priority channel 2 is resumed and completed.



1. PRIORITY INTERRUPT DAISY CHAIN BEFORE ANY INTERRUPT OCCURS.

2. CHANNEL 2 REQUESTS AN INTERRUPT AND IS ACKNOWLEDGED.

3. CHANNEL 1 INTERRUPTS, SUSPENDS SERVICING OF CHANNEL 2.

4. CHANNEL 1 SERVICE ROUTINE COMPLETE, "RETI" ISSUED, CHANNEL 2 SERVICE RESUMED.

5. SECOND "RETI" INSTRUCTION ISSUED ON COMPLETION OF CHANNEL 2 SERVICE ROUTINE.

8.0 ABSOLUTE MAXIMUM RATINGS

Temperature Under Bias	0° C to 70° C
Storage Temperature	-65° C to + 150° C
Voltage On Any Pin With Respect To Ground	-0.3 V to +7 V
Power Dissipation	0.8 W

***Comment**

Stresses above those listed under "Absolute Maximum Rating" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating *conditions for extended periods may affect device reliability.*

8.1 D.C. CHARACTERISTICS

TA = 0°C to 70°C, Vcc = 5V ± 5% unless otherwise specified

Z80–CTC

Symbol	Parameter	Min	Max	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	–0.3	.45	V	
V_{IHC}	Clock Input High Voltage [1]	V_{CC} - .6	V_{CC} + .3	V	
V_{IL}	Input Low Voltage	–0.3	0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC}	V	
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = 2 mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = -250 μA
I_{CC}	Power Supply Current		120	mA	T_C = 400 nsec
I_{LI}	Input Leakage Current		10	μA	V_{IN} = 0 to V_{CC}
I_{LOH}	Tri-State Output Leakage Current in Float		10	μA	V_{OUT} = 2.4 to V_{CC}
I_{LOL}	Tri-State Output Leakage Current in Float		-10	μA	V_{OUT} = 0.4V
I_{OHD}	*Darlington Drive Current*	-1.5		mA	V_{OH} = 1.5V R_{EXT} = 390Ω

Z80A–CTC

Symbol	Parameter	Min	Max	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	–0.3	.45	V	
V_{IHC}	Clock Input High Voltage [1]	V_{CC} - .6	V_{CC} + .3	V	
V_{IL}	Input Low Voltage	–0.3	0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC}	V	
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = 2 mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = –250 μA
I_{CC}	Power Supply Current		120	mA	T_C = 250 nsec
I_{LI}	Input Leakage Current		10	μA	V_{IN} = 0 to V_{CC}
I_{LOH}	Tri-State Output Leakage Current in Float		10	μA	V_{OUT} = 2.4 to V_{CC}
I_{LOL}	Tri-State Output Leakage Current in Float		-10	μA	V_{OUT} = 0.4V
I_{OHD}	Darlington Drive Current	-1.5		mA	V_{OH} = 1.5V R_{EXT} = 390Ω

8.2 CAPACITANCE

TA = 25° C, f = 1 MHz

Symbol	Parameter	Max.	Unit	Test Condition
C_{Φ}	Clock Capacitance	20	pF	Unmeasured Pins Returned to Ground
C_{IN}	Input Capacitance	5	pF	
C_{OUT}	*Output Capacitance*	10	pF	

8.3 A.C. CHARACTERISTICS Z80-CTC

TA = 0° C to 70° C, Vcc = +5 V ± 5%, unless otherwise noted

Signal	Symbol	Parameter	Min	Max	Unit	Comments
Φ	t_C	Clock Period	400	[1]	ns	
	$t_W(\Phi H)$	Clock Pulse Width, Clock High	170	2000	ns	
	$t_W(\Phi L)$	Clock Pulse Width, Clock Low	170	2000	ns	
	t_r, t_f	Clock Rise and Fall Times		30	ns	
	t_H	Any Hold Time for Specified Setup Time	0		ns	
CS, $\overline{CE}$, etc.	$t_{S\Phi}(CS)$	Control Signal Setup Time to Rising Edge of Φ During Read or Write Cycle	160		ns	
D_0-D_7	$t_{DR}(D)$	Data Output Delay from Rising Edge of $\overline{RD}$ During Read Cycle		480	ns	[2]
	$t_{S\Phi}(D)$	Data Setup Time to Rising Edge of Φ During Write or M1 Cycle	60		ns	
	$t_{DI}(D)$	Data Output Delay from Falling Edge of IORQ During INTA Cycle		340	ns	[2]
	$t_F(D)$	Delay to Floating Bus (Output Buffer Disable Time)		230	ns	
IEI	$t_S(IEI)$	IEI Setup Time to Falling Edge of $\overline{IORQ}$ During INTA Cycle	200		ns	
IEO	$t_{DH}(IO)$	IEO Delay Time from Rising Edge of IEI		220	ns	[3]
	$t_{DL}(IO)$	IEO Delay Time from Falling Edge of IEI		190	ns	[3]
	$t_{DM}(IO)$	IEO Delay from Falling Edge of $\overline{M1}$ (Interrupt Occurring just Prior to $\overline{M1}$)		300	ns	[3]
$\overline{IORQ}$	$t_{S\Phi}(IR)$	$\overline{IORQ}$ Setup Time to Rising Edge of Φ During Read or Write Cycle	250		ns	
$\overline{M1}$	$t_{S\Phi}(M1)$	$\overline{M1}$ Setup Time to Rising Edge of Φ During INTA or M1 Cycle	210		ns	
$\overline{RD}$	$t_{S\Phi}(RD)$	$\overline{RD}$ Setup Time to Rising Edge of Φ During Read or M1 Cycle	240		ns	
$\overline{INT}$	$t_{DCK}(IT)$	$\overline{INT}$ Delay Time from Rising Edge of CLK/TRG		$2t_C(\Phi) + 200$		Counter Mode
	$t_{D\Phi}(IT)$	$\overline{INT}$ Delay Time from Rising Edge of Φ		$t_C(\Phi) + 200$		Timer Mode
CLK/TRG_{0-3}	$t_C(CK)$	Clock Period	$2t_C(\Phi)$			Counter Mode
	t_r, t_f	Clock and Trigger Rise and Fall Times		50		
	$t_S(CK)$	Clock Setup Time to Rising Edge of Φ for Immediate Count	210			Counter Mode
	$t_S(TR)$	Trigger Setup Time to Rising Edge of Φ for Enabling of Prescaler on Following Rising Edge of Φ	210			Timer Mode
	$t_W(CTH)$	Clock and Trigger High Pulse Width	200			Counter and Timer Modes
	$t_W(CTL)$	Clock and Trigger Low Pulse Width	200			Counter and Timer Modes
ZC/TO_{0-2}	$t_{DH}(ZC)$	ZC/TO Delay Time from Rising Edge of Φ, ZC/TO High		190		Counter and Timer Modes
	$t_{DL}(ZC)$	ZC/TO Delay Time from Falling Edge of Φ, ZC/TO Low		190		Counter and Timer Modes

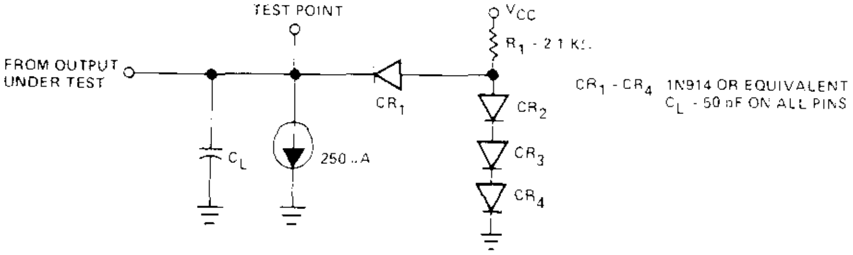
Notes: [1] $t_C = t_W(\Phi H) + t_W(\Phi L) + t_r + t_f$.

[2] Increase delay by 10 nsec for each 50 pF increase in loading, 200 pF maximum for data lines and 100 pF for control lines.

[3] Increase delay by 2 nsec for each 10 pF increase in loading, 100 pF maximum

[4] $\overline{RESET}$ must be active for a minimum of 3 clock cycles.

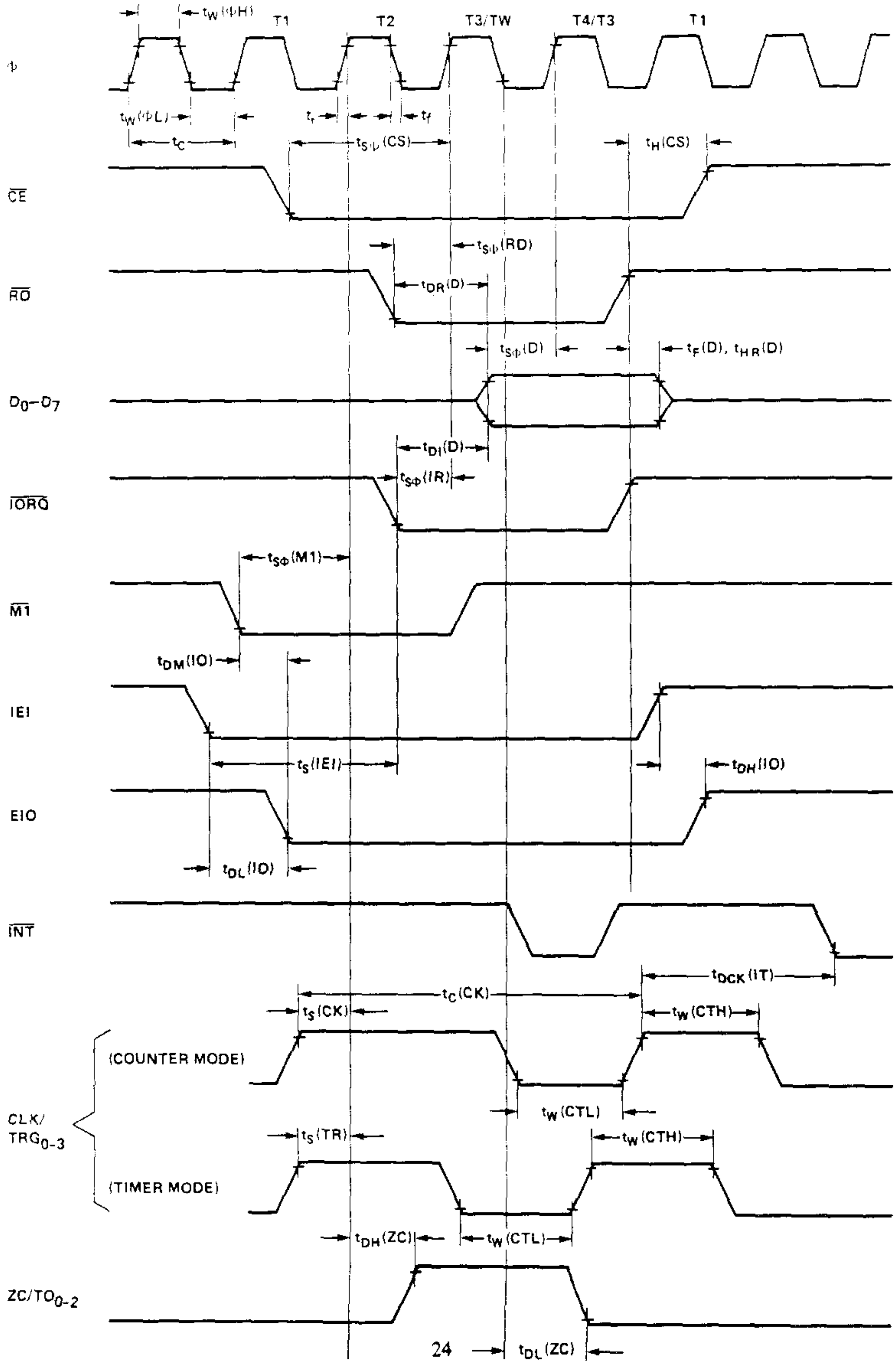
OUTPUT LOAD CIRCUIT



8.4 A.C. TIMING DIAGRAM

Timing measurements are made at the following voltages, unless otherwise specified:

	"1"	"0"
CLOCK	V_{CC} - .6V	.45V
OUTPUT	2.0V	.8V
INPUT	2.0V	.8V
FLOAT	ΔV	±0.5V



8.5 A.C. CHARACTERISTICS Z80A-CTC

TA = 0° C to 70° C, Vcc = +5 V ± 5%, unless otherwise noted

Signal	Symbol	Parameter	Min	Max	Unit	Comments
Φ	t_C	Clock Period	250	[1]	ns	
	$t_W(\Phi H)$	Clock Pulse Width, Clock High	105	2000	ns	
	$t_W(\Phi L)$	Clock Pulse Width, Clock Low	105	2000	ns	
	t_r, t_f	Clock Rise and Fall Times		30	ns	
	t_H	Any Hold Time for Specified Setup Time	0		ns	
CS, $\overline{CE}$, etc	$t_{S\Phi}(CS)$	Control Signal Setup Time to Rising Edge of Φ During Read or Write Cycle	60		ns	
D_0–D_7	$t_{DR}(D)$	Data Output Delay from Falling Edge of $\overline{RD}$ During Read Cycle		380	ns	[2]
	$t_{S\Phi}(D)$	Data Setup Time to Rising Edge of Φ During Write or M1 Cycle	50		ns	
	$t_{DI}(D)$	Data Output Delay from Falling Edge of IORG During INTA Cycle		160	ns	[2]
	$t_F(D)$	Delay to Floating Bus (Output Buffer Disable Time)		110	ns	
IEI	$t_S(IEI)$	IEI Setup Time to Falling Edge of $\overline{IORQ}$ During INTA Cycle	140		ns	
IEO	$t_{DH}(IO)$	IEO Delay Time from Rising Edge of IEI		160	ns	[3]
	$t_{DL}(IO)$	IEO Delay Time from Falling Edge of IEI		130	ns	[3]
	$t_{DM}(IO)$	IEO Delay from Falling Edge of $\overline{M1}$ (Interrupt Occurring just Prior to $\overline{M1}$)		190	ns	[3]
$\overline{IORQ}$	$t_{S\Phi}(IR)$	$\overline{IORQ}$ Setup Time to Rising Edge of Φ During Read or Write Cycle	115		ns	
$\overline{M1}$	$t_{S\Phi}(M1)$	$\overline{M1}$ Setup Time to Rising Edge of Φ During INTA or M1 Cycle	90		ns	
$\overline{RD}$	$t_{S\Phi}(RD)$	$\overline{RD}$ Setup Time to Rising Edge of Φ During Read or M1 Cycle	115		ns	
$\overline{INT}$	$t_{DCK}(IT)$	$\overline{INT}$ Delay Time from Rising Edge of CLK/TRG		$2t_C(\Phi) + 140$		Counter Mode
	$t_{D\Phi}(IT)$	$\overline{INT}$ Delay Time from Rising Edge of Φ		$t_C(\Phi) + 140$		Timer Mode
CLK/TRG$_{0-3}$	$t_C(CK)$	Clock Period	$2t_C(\Phi)$			Counter Mode
	t_r, t_f	Clock and Trigger Rise and Fall Times		30		
	$t_S(CK)$	Clock Setup Time to Rising Edge of Φ for Immediate Count	130			Counter Mode
	$t_S(TR)$	Trigger Setup Time to Rising Edge of Φ for enabling of Prescaler on Following Rising Edge of Φ	130			Timer Mode
	$t_W(CTH)$	Clock and Trigger High Pulse Width	120			Counter and Timer Modes
	$t_W(CTL)$	Clock and Trigger Low Pulse Width	120			Counter and Timer Modes
ZC/TO$_{0-2}$	$t_{DH}(ZC)$	ZC/TO Delay Time from Rising Edge of Φ, ZC/TO High		120		Counter and Timer Modes
	$t_{DL}(ZC)$	ZC/TO Delay Time from Rising Edge of Φ, ZC/TO Low		120		Counter and Timer Modes

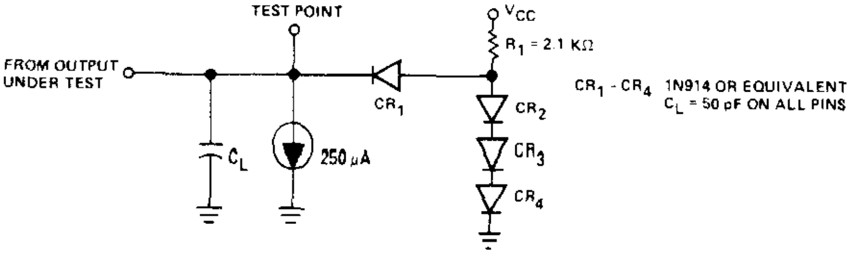
Notes: [1] $t_C = t_W(\Phi H) + t_W(\Phi L) + t_r + t_f$.

[2] Increase delay by 10 nsec for each 50 pF increase in loading, 200 pF maximum for data lines and 100 pF for control lines.

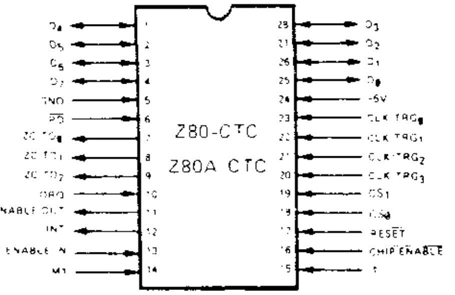
[3] Increase delay by 2 nsec for each 10 pF increase in loading, 100 pF maximum.

[4] $\overline{\text{RESET}}$ must be active for a minimum of 3 clock cycles.

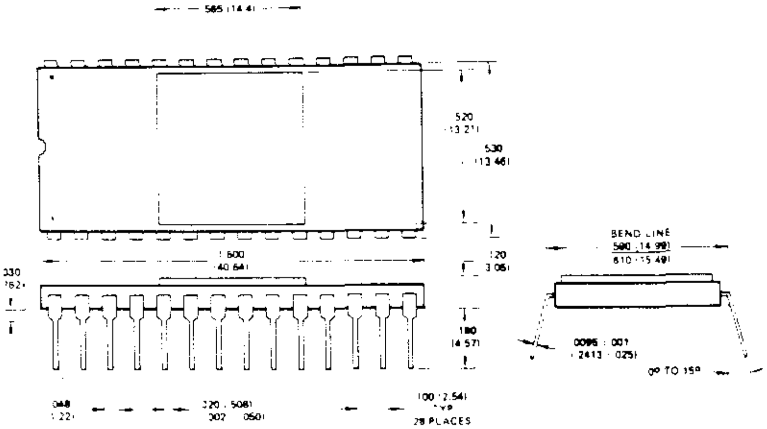
OUTPUT LOAD CIRCUIT



8.6 PACKAGE CONFIGURATION



PACKAGE OUTLINE



* DIMENSIONS FOR METRIC SYSTEM IN PARENTHESES (mm)

Ordering Information

C – Ceramic
P – Plastic
S – Standard 5V±5%, 0° to 70°C
E – Extended 5V±5%, -40° to 85°C
M – Military 5V±10%, -55° to 125°C

Example:

Z80-CTC CS (Ceramic–Standard Range)
Z80A-CTC PS (Plastic–Standard Range, 4 MHz)

ZILOG Z80 MICROCOMPUTER SYSTEM COMPONENT FAMILY

● Z80, Z80A-CPU	CENTRAL PROCESSOR UNIT
● Z80, Z80A-PIO	PARALLEL I/O
● Z80, Z80A-CTC	COUNTER/TIMER CIRCUIT
● Z80, Z80A-DMA	DIRECT MEMORY ACCESS
● Z80, Z80A-SIO	SERIAL I/O
● Z6104	4K x 1 STATIC RAM
● Z6116	16K x 1 DYNAMIC RAM

Appendix C

Using the Z80 SIO In Asynchronous Communications

Contents

Using the Z80® SIO In Asynchronous Communications

Zilog

Application Note

July 1980

SECTION 1 Introduction.

The Z80 Serial Input/Output (SIO) controller is designed for use in a wide variety of serial-to-parallel input and parallel-to-serial output applications. In this application note, only asynchronous applications are considered. The emphasis is almost completely on software implementation, with only modest reference to hardware considerations.

While reference is made only to the Z80 SIO, the entire text also applies to the Z80 DART, which is functionally identical to the Z80 SIO in asynchronous applications.

Protocol

Communication, either on an external data link or to a local peripheral, occurs in one of two basic formats: synchronous or asynchronous. In synchronous communication, a message is sent as a continuous string of characters where the string is preceded and terminated by control characters; the preceding control characters are used by the receiving device to synchronize its clock with the transmitter's clock. In asynchronous communication, which is described in this application note, there is no attempt at synchronizing the clocks on the transmitting and receiving devices. Instead, each fixed-length character (rather than character string) is preceded and terminated by "framing bits" that identify the beginning and end of the character. The time between bits within a character is approximately constant, since the clocks or "baud rates" in the transmitter and receiver are selected to be the same, but the time between characters can vary.

Thus, in asynchronous communication, each character to be transmitted is preceded by a "start" framing bit and followed by one or more "stop" framing bits. A start bit is a logical 0 and a stop bit is a logical 1. The receiver will look for a start bit, assemble the character up to the number of bits the SIO has been programmed for, and then expect to find a stop bit. The time between the start and stop bits is approximately constant, but the time between characters can vary. When one character ends, the receiving device will wait idly for the start of the next character while the transmitter continues to send stop or "marking" bits (both the stop bits and the marking bits are logical 1). Figure 1 illustrates this. A very common application of asynchronous communication is with keyboard devices, where the time between the operator's keystrokes can vary considerably.

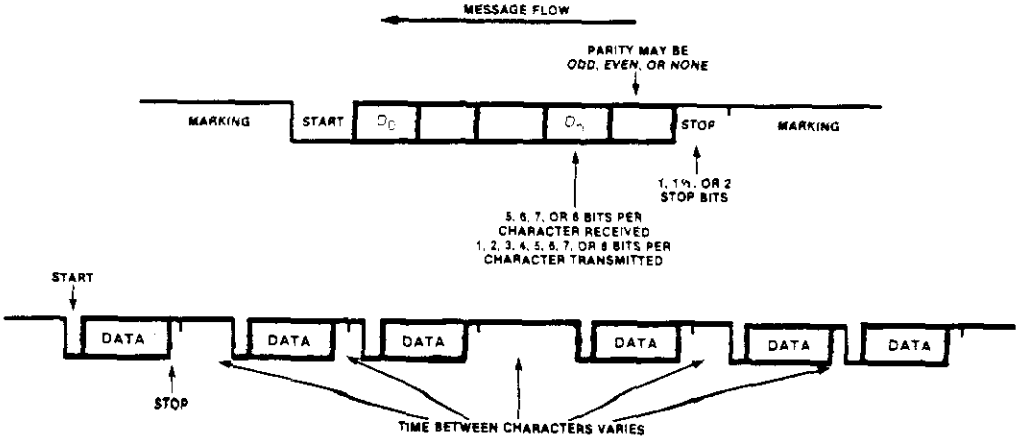

MESSAGE FLOW
PARITY MAY BE ODD, EVEN, OR NONE
MARKING
START
D_0
D_n
STOP
MARKING
1, 1½, OR 2 STOP BITS
5, 6, 7, OR 8 BITS PER CHARACTER RECEIVED 1, 2, 3, 4, 5, 6, 7, OR 8 BITS PER CHARACTER TRANSMITTED
START
DATA
DATA
DATA
DATA
DATA
DATA
STOP
TIME BETWEEN CHARACTERS VARIES

Figure 1. Asynchronous Data Format

Protocol (Continued)

If the transmitter's clock is slightly faster than the receiver's clock, the transmitter can be programmed to send additional stop bits, which will allow the receiver to catch up. If the receiver runs slightly faster than the transmitter, then the receiver will see somewhat larger gaps between characters than the transmitter does, but the characters will normally still be received properly. This tolerance of minor frequency deviations is an important advantage of using asynchronous I/O. Note however that errors, called "framing errors," can still occur if the transmitter and receiver differ substantially in speed, since data bits may then be erroneously treated as start or stop bits.

Modes

The SIO may be used in one of three modes: Polled, Interrupt, or Block Transfer, depending on the capabilities of the CPU. In Polled mode the CPU reads a status register in the SIO periodically to determine if a data character has been received or is ready for transmission. When the SIO is ready, the CPU handles the transfer within its main program.

In Interrupt mode, which is far more common, the SIO informs the CPU via an interrupt signal that a single-character transfer is required. To accomplish this, the CPU must be able to check for the presence of interrupt signals (or "interrupt requests") at the end of most instruction cycles. When the CPU detects an interrupt it branches to an interrupt service routine which handles the single-character transfer. The beginning memory address of this interrupt service routine can be derived, in part, from an "interrupt vector" (8-bit byte) supplied by the SIO during the interrupt acknowledge cycle.

In Block Transfer mode, the SIO is used in conjunction with a DMA (direct memory access) controller or with the Z80 or Z8000 CPU block transfer instructions for very fast transfers. The SIO interrupts the CPU or DMA only when the first character of a message becomes available, and thereafter the SIO uses only its Wait/Ready output pin to signal its readiness for subsequent character transfers. Due to the faster transfer speeds achievable, Block Transfer mode is most commonly used in synchronous communication and only rarely in asynchronous formats. It is therefore not treated with specific examples in this application note.

Since Polled mode requires CPU overhead regardless of whether or not an I/O device desires attention, Interrupt mode is usually the preferred alternative when it is supported by the CPU. Note that the choice of Polled or Interrupt mode is independent of the choice of synchronous or asynchronous I/O. This latter choice is usually determined by the type of device to which the system is communicating.

SIO Configurations

The SIO comes in four different 40-pin configurations: SIO/0, SIO/1, SIO/2, and SIO/9. The first three of these support two independent full-duplex channels, each with separate control and status registers used by the CPU to write control bytes and read status bytes. The SIO/9 differs from the first three versions in that it supports only one full-duplex channel. The product specifications for these versions explain this in full.

There are 41 different signals needed for complete two-channel implementation in the SIO/0, SIO/1, and SIO/2, but only 40 pins are available. Therefore, the versions differ by either omitting one signal or bonding two signals together. The dual-channel asynchronous-only Z80 DART has the same pin configuration as the SIO/0.

SECTION 2

Operational Considerations.

All of the SIO options to be discussed here are software controllable and are set by the CPU. Thus, use of the SIO begins with an initialization phase where the various options are set by writing control bytes. These options are established separately for each of the two channels supported by the SIO if both channels are used. *Before giving an overview of* how initialization is done, we will describe some of the basic characteristics of SIO operations that are common to both the Polled and Interrupt-driven modes.

Addressing the SIO

The CPU must have a means to identify any specific I/O device, including any attached SIO. In a Z80 CPU environment, this is done by using the lower 8 bits of the address bus (A_0-A_7). Typically, the A_0 bit is wired to the SIO's B/$\overline{A}$ input pin for selecting access to Channel A or Channel B, and the A_1 bit is wired to the SIO's C/$\overline{D}$ input pin for selecting the use of the data bus as an avenue for transferring control/status information (C) or *actual data messages* (D). The remaining bits of the address bus, A_2-A_7, contain a port *address that uniquely identifies the SIO* device. These latter six lines are usually wired to an external decoding chip which activates that SIO's Chip Enable ($\overline{CE}$) input pin when its address appears on A_2-A_7 of the address bus.

The bar notation drawn above the names of certain signal lines, such as B/$\overline{A}$ and C/$\overline{D}$, refer to signals which are interpreted as active when their logic sense—and voltage level—is Low. For example, the B/$\overline{A}$ pin specifies Channel B of the SIO when it carries a logic 1 (high voltage) and it specifies Channel A when it carries a logic 0 (low voltage).

Asynchronous Format Operations

Bits per Character. The SIO *can receive* or transmit 5, 6, 7, or 8 bits per character. This *can be different for transmission and reception*, and different for each channel. ASCII characters, *for example, are usually transmitted* as 7 bits. The SIO can in fact transmit fewer than 5 bits per character when set to the 5-bit mode; this is discussed further in the section entitled "Questions and Answers."

Parity. A parity bit is an additional bit added to a character for error checking. The parity bit is set to 0 or 1 in order to make the total number of 1s in the character (including parity bit) even or odd, depending on whether even or odd parity is selected. The SIO can be set *either to add an optional parity bit* to the "bits per character" described above, or not to add *such a bit. When a parity bit is included,* either even or odd parity can be chosen. This selection can be made independently for each channel.

Start and Stop Bits. There are two types of framing bits for each character: start and stop. When transmitting asynchronously, the SIO automatically inserts one start bit (logic 0) at the beginning of each character transmitted. The SIO can be programmed to set the number of stop bits inserted at the end of each character to either 1, 1½, or 2. The receiver always checks for 1 stop bit. Stop bits refer to the length of time that the stop value, a logic 1, will be transmitted; thus 1½ stop bits means that a 1 will be transmitted for the length of clock time that 1½ bits would normally take up. A logic 1 level that continues after the specified number of stop bits is called a "marking" condition or "mark bits."

CPU-SIO Character Transfers

The SIO always passes 8-bit bytes to the CPU for each character received, no matter how many "bits per character" are specified in the SIO initialization phase. If the number of "bits per character" is less than eight, parity and/or stop bits will be included in the byte sent to the CPU. The received character starts with the least-significant bit (D_0) and continues to the most-significant bit; it is immediately followed by the parity bit (if parity is enabled) *and by the stop bit, which will be logic 1* unless there is a framing error. The remainder of the byte, *if space is still available, is filled* with logic 1s (marking). If the "bits per character" is eight, then the byte sent to the CPU will contain only the data bits. In all cases, the start bit is stripped off by the SIO and is not transmitted to the CPU.

Clock Divider

The SIO has five input pins for clock signals. One of these inputs (CLK) is used only for internal timing and does not affect transmission or reception rates. The other four clock inputs ($\overline{RxCA}$, $\overline{TxCA}$, $\overline{RxCB}$, and $\overline{TxCB}$) are used for timing the reception and transmission rates in Channels A and B. Only these last four are involved in "clock dividing." A clock divider within the SIO can be programmed to cause reception/transmission clocking at the actual input clock rate or at 1/16, 1/32, or 1/64 of the input clock rate. The receiver and transmitter clock divisions within a given channel must be the same, although their input clock rates can be different. The x1 clock rate can be used only if the transitions of the Receive clock are synchronized to occur during valid data bit times.

Auto Enables

The SIO has an Auto Enables feature that *allows automatic SIO response and telephone* answering. When Auto Enables is set for a particular channel, a transition to logical 0 (Low input level) on the respective Data Carrier Detect ($\overline{\text{DCD}}$) input will enable reception, and *a transition to logical 0 on the respective Clear* To Send ($\overline{\text{CTS}}$) input will enable transmission. This is described below under the heading "Modem Control."

Special Receive Conditions

There are three error conditions that can occur when the SIO is receiving data. Each of these will cause a status bit to be set, and if operating in Interrupt mode, the SIO can optionally be programmed to interrupt the CPU on such an error. The error conditions are called "special receive conditions" and they include:

- **Framing error.** If a stop bit is not detected in its correct location after the parity bit (if used) or after the most-significant data bit (if parity is not used), a framing error will result. The start bit preceding the character's data bits is not considered in determining a framing error, although character assembly will not begin until a start bit is detected.
- **Parity error.** If parity bits are attached by the external I/O device and checked by the SIO while receiving characters, a parity error will occur whenever the number of logic 1 data bits in the character (including the parity bit) does not correspond to the odd/even setting of the parity-checking function.
- **Receiver overrun error.** SIO buffers can hold up to three characters. If a character is received when the buffers are full (i.e., characters have not been read by the CPU), an SIO receiver overrun error will result. In this case, the most recently received character overwrites the next most recently received character.

Modem Control

Five signal lines on the SIO are provided for optional modem control, although these lines can also be used for other general-purpose control functions. They are:

RTS (Request To Send). An output from the SIO to tell its modem that the SIO is ready to transmit data.

$\overline{\text{DTR}}$ (Data Terminal Ready). An output from the SIO to tell its modem that the SIO is ready to receive data.

$\overline{\text{CTS}}$ (Clear To Send). An input to the SIO from its modem that enables SIO transmission if the Auto Enables function is used.

$\overline{\text{DCD}}$ (Data Carrier Detect). An input to the SIO from its modem that enables SIO reception if the Auto Enables function is used.

SYNC (Synchronization). A spare input to the SIO in asynchronous applications. This input may be used for the Ring Indicator function, if necessary, or for general-purpose inputs.

In most applications of asynchronous I/O that use modems, the $\overline{\text{RTS}}$ and $\overline{\text{DTR}}$ control lines and the Auto Enables function are activated during the initialization sequence, and they are left active until no further I/O is expected. This causes the SIO to tell its modem continuously that the SIO is ready to transmit and receive data, and it allows the modem to enable automatically the SIO's transmission and reception of data. Figure 3 illustrates this.

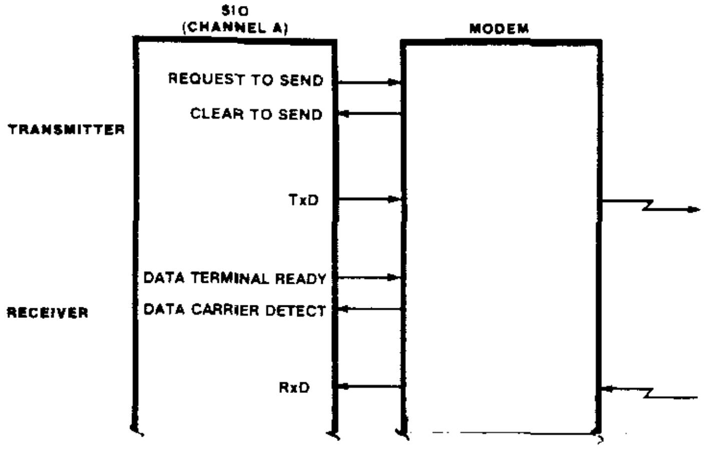


Figure 3. Modem Control (Single Channel)

External/Status Interrupts

A change in the status of certain external inputs to the SIO will cause status bits in the SIO to be set. In the Polled Mode, these status bits can be read by the CPU. In the Interrupt mode, the SIO can also be programmed to interrupt the CPU when the change occurs. There are three such "external/status" conditions that can cause these events:

- **DCD.** Reflects the value of the $\overline{DCD}$ input.
- **CTS.** Reflects the value of the $\overline{CTS}$ input.
- **Break.** *A series of logic 0 or "spacing" bits.*

Note that the DCD and CTS status bits are the inverse of the $\overline{SIO}$ lines, i.e., the DCD bit will be 1 when the $\overline{DCD}$ line is Low.

Any transition in any direction (i.e., to logic 0 or to logic 1) on any of these inputs to the SIO will cause the related status bit to be latched and (optionally) cause an interrupt. The SIO status bits are latched after a transition on any one of them. The status must be reset (using an SIO command) before new transitions can be reflected in the status bits.

Initialization

The SIO contains eight write registers for Channel B (WR0-WR7) and seven write registers for Channel A (all except write register WR2). These are described fully in the *Z80 SIO Technical Manual* and are summarized in Appendix B. The registers are programmed separately for each channel to configure the functional personality of the channel. WR2 exists only in the Channel B register set and contains the interrupt vector for both channels. Bits in each register are named D_7 (most significant) through D_0. With the exception of WR0, programming the write registers requires two bytes: the first byte is to WR0 and contains pointer bits for selection of one of the other registers; the second byte is written to the register selected. WR0 is a special case in that all of the basic commands can be written to it with a single byte.

There are also three read registers, named RR0 through RR2, from which status results of operations can be read by the CPU (see Appendix B). Both channels have a set of read registers, but register RR2 exists only in Channel B.

Let us now look at the typical sequence of write registers that are loaded to initialize the SIO for either Polled or Interrupt-driven asynchronous I/O. Figure 4 illustrates the sequence. Except for step E, this loading is done for each channel when both are used. Steps E and F are described further in the section on "Interrupt-Driven Environments."

Registers WR6 and WR7 are not used in asynchronous I/O. They apply only to synchronous communication.

The related publications on the SIO should be referred to at this point. They will be necessary in following the discussion of functions. In particular, the following material should be reviewed:

Z80 SIO Technical Manual, pages 9-12 ("Asynchronous Operation")

Z80 SIO Technical Manual, pages 29-37 ("Z80 SIO Programming")

A. Load WR0 This is done to reset the SIO.

B. Load WR4 This specifies the clock divider, number of stop bits, and parity selection. Since register WR4 establishes the general form of I/O for which the SIO is to be used, it is best to set WR4 values first.

C. Load WR3 This specifies the number of receive bits per character, Auto Enable selection, and turns on the receiver enabling bit.

D. Load WR5 This specifies the number of transmit bits per character, turns off the bit that transmits the Break signal, turns on the bits indicating Data Terminal Ready and Request To Send, and turns on the transmitter enabling bit.

E. Load WR2 (*Interrupt mode only and Channel B only.*) This specifies the interrupt vector.

F. Load WR1 (Interrupt mode only.) This specifies various interrupt-handling options that will be explained later.

NOTES:
Steps A through F are performed in sequence.
*Channel B only.
†Interrupt mode only. Polling mode begins I/O after step D.

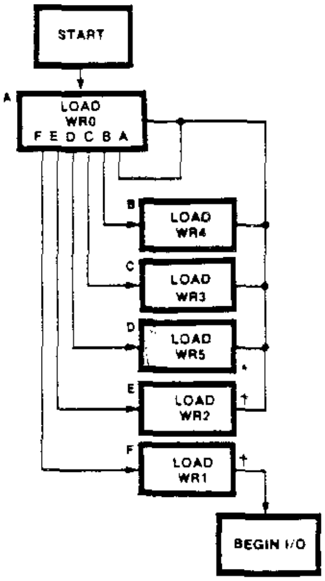


Figure 4. Typical Initialization Sequence (One Channel)

SECTION 3

Polled Environments.

In a typical Polled environment, the SIO is initialized and then periodically checked for completion of an I/O operation. Of course, if *the checking is not frequent enough*, *received* characters may be lost or the transmitter may be operated at a slower data rate than that of which it is capable. Initialization for Polled I/O follows the general outline described in the last section. We now give an overview of routines necessary for the CPU to check whether a character has been received by the SIO or whether the SIO is ready to transmit a character.

Character Reception

To check whether a character has been received, and to obtain a received character if one is available, the sequence illustrated in Figure 5 should be followed after the SIO is initialized. We assume that reception was enabled during initialization; if it was not, the Rx Enable bit in register WR3 must be turned on before reception can occur. This must be done for each channel to be checked.

Bit D_0 of register RR0 is set to 1 by the SIO if there is at least one character available to be received. The SIO contains a three-character input buffer for each channel, so more than one character may be available to be received. Removing the last available character from the read buffer for a particular channel turns off bit D_0.

If bit D_0 of register RR0 is 0, then no character is available to be received. In this case it is recommended that checks be made of bit D_7 to determine if a Break sequence (null character plus a framing error) has been received. If so, a Reset External/Status Interrupts command should be given; this will set the External/Status bits in register RR0 to the values of the signals currently being received. Thus, if the Break sequence has terminated, the next check of bit D_7 will so indicate. It may also be desirable to check bit 3 of register RR0 which reports the value of the Data Carrier Detect (DCD) bit.

In any case, if bit D_0 of register RR0 is 0, polled receive processing terminates with no *character to receive*. Depending on the facilities of the associated CPU, this step may be repeated until a character is available (or possibly a time-out occurs), or the CPU may return to other tasks and repeat this process later.

If bit D_0 of register RR0 is 1, then *at least* one character is available to be read. In this case, the value of register RR1 should first be read and stored to avoid losing any error information (the manner in which it is read is explained later). The character in the data register is then read. Note that the character must be read to clear the buffer even if there is an error found.

Finally, it is necessary to check the value stored from register RR1 to determine if the character received was valid. Up to three bits need to be checked: bit 6 is set to 1 for a framing error, bit 5 is set to 1 for a receiver overrun error (which occurs when the receive buffers are overwritten, i.e., no character has been removed and more than three characters have been received), and bit 4 is set to 1 for a parity error (if parity is enabled at initialization time). In case of a receiver overrun or parity error, an Error Reset command must be given to reset the bits.

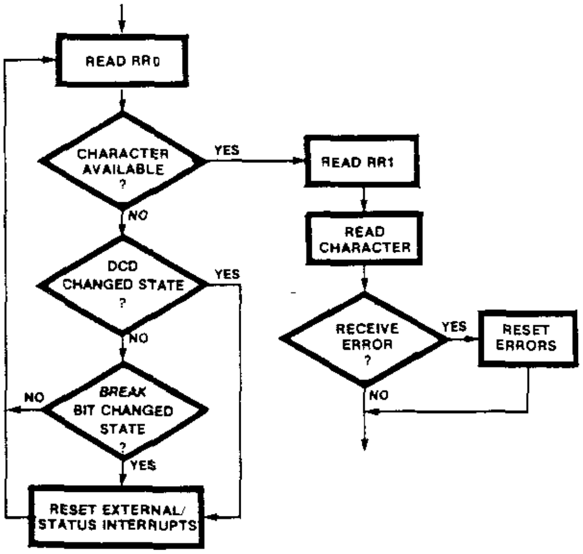


Figure 5. Polled Receive Routine

Character Transmission

To check that an initialized SIO is ready to transmit a character on a channel, and if so to transmit the character, the steps illustrated in Figure 6 should be followed. We assume that the Request To Send (RTS) bit in WR5, if required by the external receiving device, and the Transmit (Tx) Enable bit were set at initialization.

Depending on the external receiving device, the following bits in register RR0 should be checked: bit 3 (DCD), to determine if a data carrier has been detected; bit 5 (CTS), to determine if the device has signalled that it is clear to send; and bit 7 (Break), to determine if a Break sequence has been received. If any of these situations have occurred, the bits in register RR0 must be reset by sending the Reset External/Status Interrupts command, and the transmit sequence must be started again.

Next, bit 2 of register RR0 is checked. If this bit is 0, then the transmit buffer is not empty and a new character cannot yet be transmitted. Depending on the capabilities of the CPU, this is repeated until a character can be transmitted (or a timeout occurs), or the CPU may return to other tasks and start again later.

If bit 2 of register RR0 is 1, then the transmit buffer is empty and the CPU may pass the character to be transmitted to the SIO, completing the transmit processing. On the Z80 CPU, this is done with an OUT instruction to the SIO data port.

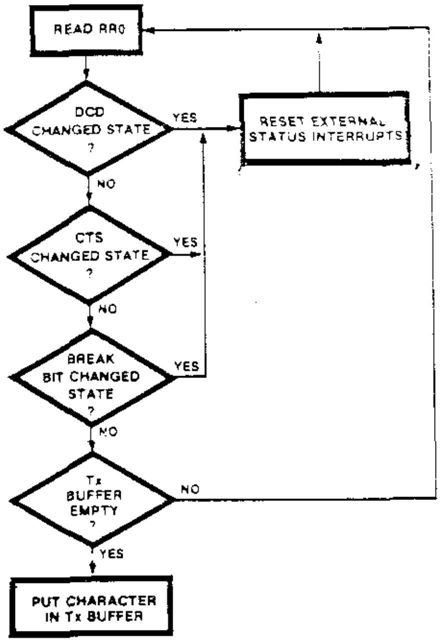

READ RR0
DCD CHANGED STATE ?
YES
RESET EXTERNAL STATUS INTERRUPTS
NO
CTS CHANGED STATE ?
YES
NO
BREAK BIT CHANGED STATE ?
YES
NO
Tx BUFFER EMPTY ?
NO
YES
PUT CHARACTER IN Tx BUFFER

Figure 6. Polled Transmit

Assumptions for an Example

Now let us consider some examples in more detail. We assume we are given an external device to which we will input and output 8-bit characters, with odd parity, using the Auto Enables feature. We will support this device with I/O polling routines following the patterns illustrated in Figures 5 and 6. We assume that the CPU will provide space to receive characters from the SIO as fast as the characters are received by the SIO, and that the CPU will transfer characters as fast as the output can be accomplished by the SIO.

We specify this example by giving the control bytes (commands) written to the SIO and the status bytes that must be read from the SIO. Recall that to write a command to a register, except register WR0, the number of the register to be written is first sent to register WR0; the following byte will be sent to the named register. Similarly, to read a register other than RR0 (the default), the number of the register to be read is sent to register WR0; the following byte will return the register named.

Initialization

We begin with the initialization code for the SIO. This follows the outline illustrated in Figure 4. In the following sample code, each time register WR0 is changed to point to another register, the Reset External/Status Interrupts command is given simultaneously. Whenever a transition on any of the external lines occurs, the bits reporting such a transition are latched until the Reset External/Status Interrupts command is given. Up to two transitions can be remembered by the SIO. Therefore, it is desirable to do at least two different Reset External/Status Interrupts commands as late as possible in the initialization so that the status bits reflect the most recent information. Since it doesn't hurt, we include these commands each time WR0 is changed to point to another register. This is an easy way to code the initialization to insure that the appropriate resets occur.

In the example below, the logic states on the $C/\overline{D}$ control line and the system data bus (D_7-D_0) are illustrated, together with comments.

Initialization
(Continued)

	Bits sent to the SIO								
C/$\overline{D}$	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	**Effects and Comments**
1	0	0	0	1	1	0	0	0	Channel Reset command sent to register WR0 (D_5-D_3).
1	0	0	0	1	0	1	0	0	Point WR0 to WR4 (D_2-D_0) and issue a Reset External Status Interrupts command (D_5-D_3). Throughout the initialization, whenever we point WR0 to another register, we will also issue this command for the reasons noted above.
1	1	1	0	0	1	1	0	1	Set WR4 to indicate the following parameters (from left to right): A. Run at 1/64 the input clock rate (D_7-D_6). B. Disable the sync bits and send out 2 stop bits per character (D_3-D_2). C. Enable odd parity (D_1-D_0).
1	0	0	0	1	0	0	1	1	Point WR0 to WR3.
1	1	1	1	0	0	0	0	1	Set WR3 to indicate the following: A. 8-bit characters to be received (D_7-D_6). B. Auto Enables on (D_5). C. Receive (Rx) Enable on (D_0).
1	0	0	0	1	0	1	0	1	Point WR0 to WR5.
1	1	1	1	0	1	0	1	0	Set WR5 to indicate the following: A. Data Terminal Ready (DTR) on (D_7). B. 8-bit characters to be transmitted (D_6-D_5). C. Break not to be transmitted (D_4). D. Transmit (Tx) Enable on (D_3). E. Request To Send (RTS) on (D_1).

Reset and Error Sequences

In the receive and transmit routines that follow, we treat errors such as a transition on the Data Carrier Detect line by calling for a "reset sequence" to set the values in read register RR0 to reflect the current values found at the pins. This sequence consists of giving the Reset External/Status Interrupts command and beginning the driver over again. The command takes the form of a write to register WR0:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	0	1	0	0	0	0

Permits the status bits in RR0 to reflect current status.

This command does not turn off the latches for such things as parity errors stored in bits 4-6 of register RR1. When such an error occurs and the latches must be reset, we will call for an "error sequence." This sequence consists of giving the Error Reset command and beginning the driver over again. The command also takes the form of a write to register WR0:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	1	1	0	0	0	0

Resets the latches in register RR1.

When specifying the result of reading register RR0 or RR1 or specifying data, we will indicate the values read as follows:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
D	D	D	D	D	D	D	D

Read a byte from the designated register..

Receive and Transmit Routines

Now we will first give an example of the receive routine. This parallels the preceding discussion of "Character Reception."

The framing error in this routine is reported on a character-by-character basis and it is not necessary to execute an "error sequence" if it is the only error received. However, it is not harmful to do so.

Next, we give an example of transmission code that parallels the above discussion on "Character Transmission."

Receive and Transmit Routines (Continued)

	Bits sent and received								
$C/\overline{D}$	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	**Effects and Comments (Receive Routine)**
1	D	D	D	D	D	D	D	D	Read a byte from RR0 (the default read register): if $D_0 = 0$ then no character is ready to be received. In this case, if D_7 (Break) or D_3 (Data Carrier Detect) have changed state, then execute a "reset sequence." If $D_0 = 0$ and D_7 and D_3 have not changed state, then no character is ready to be received; either loop on this read or try again later
1	0	0	0	0	0	0	0	1	Point WR0 to read from RR1 (we will now check for errors in the character read). Note that Reset External Status Interrupt Commands are not done normally to avoid losing a line-status change.
1	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Read a byte from RR1: if either bit $D_6 = 1$ (framing error), $D_5 = 1$ (receive overrun error), or $D_4 = 1$ (parity error), the character is invalid and an "error sequence" should be executed after the following step.
0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Read in the data byte received. This must be done to clear the SIO buffer even if an error is detected

	Bits sent and received								
$C/\overline{D}$	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	**Effects and Comments (Transmit Routine)**
1	D	D	D	D	D	D	D	D	Read a byte from RR0; if either bit D_3 (Data Carrier Detect), D_5 (Clear To Send) or D_7 (Break) have changed state, a "reset sequence" should be executed. If D_3, D_5 and D_7 have not changed state, then if $D_2 = 0$, the transmit buffer is not yet empty and a transmit cannot take place; either loop, reading RR0, or try again later.
0	D	D	D	D	D	D	D	D	Send the data byte to be transmitted.

SECTION 4

Interrupt-Driven Environments.

In a typical interrupt-driven environment, the SIO is initialized and the first transmission, if any, is begun. Thereafter, further I/O is interrupt driven. When action by the CPU is needed, an SIO interrupt causes the CPU to branch to an interrupt service routine after the CPU first saves state information.

In common usage, if I/O is interrupt driven, all interrupts are enabled and each different type of interrupt is used to cause a CPU branch to a different memory address. There is perhaps one frequent exception to this: parity errors are sometimes checked only at the end of a sequence of characters. The SIO facilitates this kind of operation since the parity error bit in read register RR1 is latched; once the bit is set it is not reset until an explicit reset operation is done. Thus, if a parity error *has occurred on any character since last reset,* bit 4 in register RR1 will be set. It is then possible to set register WR1 so that parity errors do not cause an error interrupt when a character is received. The user then has the obligation to poll for the value of the parity bit upon completion of the sequence.

SIO initialization for Interrupt mode normally requires two steps not used in Polled mode: an interrupt vector (if used) must be stored in write register WR2 of Channel B and write register WR1 must be initialized to specify the form of interrupt handling. It is preferable to initialize the interrupt vector in WR2 first. In this way an interrupt that arrives after the enabling bits are set in WR1 will cause proper interrupt servicing.

Interrupt Vectors

The interrupt vector, register WR2 of Channel B, is an 8-bit memory address. When an interrupt occurs (and note that an interrupt can only occur after interrupts have been enabled by writing to register WR1) the interrupt vector is normally taken as one byte of an address used by the CPU to find the location of the interrupt service routine. It is also possible to cause the particular type of interrupt condition to modify the address vector in WR2 before branching, resulting in a branch to a different memory location for each interrupt condition. This is a very useful construct; it permits short, special-purpose interrupt routines. The alternative, to have one general-purpose interrupt routine which must determine the situation before proceeding, can be quite inefficient. This is usually undesirable since the speed of interrupt-service routines is often a critical factor in determining system performance.

Interrupt Vectors (Continued)

There are at most eight different types of interrupts that the SIO may cause, four for each of the two channels. If bit 1 in register WR1 of Channel B has been turned on so that an interrupt will modify the interrupt vector, the three bits (1-3) of the vector will be changed to reflect the particular type of interrupt. These interrupts follow a hardware-set priority as follows, starting with the highest priority:

Channel A Special Receive Condition sets bits 3-1 of WR1 to 111,

Channel A Character Received sets bits 3-1 to 110,

Channel A Transmit Buffer Empty sets bits 3-1 to 100,

Channel A External/Status Transition sets bits 3-1 to 101.

Channel B Special Receive Condition sets bits 3-1 to 011,

Channel B Character Received sets bits 3-1 to 010,

Channel B Transmit Buffer Empty sets bits 3-1 to 000,

Channel B External/Status Transition sets bits 3-1 to 001.

For example, suppose that the interrupt vector had the value 11110001 and the Status Affects Vector bit is enabled, along with all interrupt-enable bits. When an External/Status transition occurs in Channel A, the three zeros (bits 3-1) would be modified to 101, yielding an interrupt vector of 11111011. The value of the interrupt vector, as modified, may be tained by reading register RR2 in Channel B.

Note that when a character is received, either the Special Receive Condition or Rx Character Available interrupt will occur, depending on whether or not an error occurred; the two will never occur simultaneously. Therefore, these two interrupts have equal priority. Note also that you can select not to be interrupted on some of the eight conditions; in this case, the presence of a particular condition for which interrupts are not desired can be determined by polling.

Suppose that interrupts have been enabled for all possible cases, and that the Status Affects Vector bit has also been enabled, allowing a different routine to handle each possible interrupt. As each interrupt causes a branch to a location only two bytes higher than the last interrupt, it is not possible to place a routine directly at the location where the vectored interrupt branches. In a Z80 CPU environment, these addresses refer to a table in memory which contains the actual starting location of the interrupt service routine. Also, since the state information saved by a CPU is rarely all of the information necessary to properly preserve a computation state, a typical interrupt service routine will begin by saving additional information and end by restoring that information. This is shown briefly in the examples of code in Appendix A.

It is possible to connect several SIOs using the interrupt mechanism and the IEI and IEO lines on the SIO to determine a priority for interrupt service. This mechanism is discussed on page 42 of the *Z80 SIO Technical Manual* and in the *Z80 Family Program Interrupt Structure Manual.* We do not go into it further in this application note.

Initialization

In general, the initialization procedure illustrated in Figure 4 can still be followed. All six steps (A through F) are required here. After completing the first four steps, which are the same as initialization for polled I/O, it is necessary to load an interrupt vector into WR2 of Channel B. Information is then written into register WR1 specifying which interrupts are to be enabled and whether a specific kind of interrupt should modify the interrupt vector.

Now let us give an example. As in the polled example, we assume that we are given a device to which we will input and output 8-bit characters, with odd parity, using the Auto Enables feature. We also assume the CPU will provide space to store characters as received.

We do not discuss the SIO commands and registers in detail. This is done in the *Z80 SIO Technical Manual.* A summary of the register bit assignments taken from the *Z80 SIO Serial Input/Output Product Specification* is included at the end of this note. Recall that to write a register other than register WR0, the number of the register to be written is first sent to register WR0, and the following byte will be sent to the named register. Similarly, to read a register other than RR0 (the default), the number of the register to be read is first written to register WR0 and the next byte read will return the contents of the register named.

In our example below, each time register WR0 is changed to point to another register, the Reset External/Status Interrupts command is also given. Whenever a transition on any of the external/status lines occurs, the bits reporting the transition are latched until the Reset External/Status Interrupts command is given. Up to two transitions can be remembered by the internal logic of the SIO. Therefore, it is desirable to do at least two different Reset External/Status Interrupt commands as late as possible in the initialization so that the status bits reflect the most recent information. Since it doesn't hurt, we give these commands each

Initialization (Continued)

time WR0 is changed to point to another register. *This is an easy way to code the initialization* to assure that the appropriate resets occur.

The columns below show the logic states on the $C/\overline{D}$ control line and the system data bus (D_7-D_0), together with comments.

	Bits sent to the SIO								
$C/\overline{D}$	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Effects and Comments
1	0	0	0	1	1	0	0	0	Channel Reset command sent to register WR0 (D_5-D_3).
1	0	0	0	1	0	1	0	0	Point WR0 to WR4 (D_2-D_0), and issue a Reset External/Status Interrupts command (D_5-D_3). Throughout the initialization, whenever we point WR0 to another register we will also issue a Reset External Status Interrupts command for the reasons noted above.
1	1	1	0	0	1	1	0	1	Set WR4 to indicate the following parameters (from left to right): A. Run at 1/64 the clock rate (D_7-D_6). B. Disable the sync bits and send out 2 stop bits per character (D_5-D_2). C. Enable odd parity (D_1-D_0).
1	0	0	0	1	0	0	1	1	Point WR0 to WR3.
1	1	1	1	0	0	0	0	1	Set WR3 to indicate the following: A. 8-bit characters to be received (D_7-D_6). B. Auto Enables on (D_5). C. Rx Enable on (D_0).
1	0	0	0	1	0	1	0	1	Point WR0 to WR5.
1	1	1	1	0	1	0	1	0	Set WR5 to indicate the following: A. Data Terminal Ready (DTR) on (D_7). B. 8-bit characters to be transmitted (D_6-D_5). C. Break not to be transmitted (D_4). D. Tx Enable on (D_3). E. *Request To Send* (RTS) on (D_1).
1	0	0	0	1	0	0	1	0	Point WR0 to WR2 (Channel B only).
1	1	1	1	0	0	0	0	0	Set the interrupt vector to point to address 11100000 (which is hex E0 and decimal 224). Once interrupts are enabled, they will cause a branch to this memory location, modified as described above if the Status Affects Vector bit is turned on (which it will be here). This vector is only set for Channel B, but it applies to both channels. *It has no effect when set in* Channel A.
1	0	0	0	1	0	0	0	1	Point WR0 to WR1.
1	0	0	0	1	0	1	1	1	Set WR1 to indicate the following: A. Cause interrupts on all characters received, treating a parity error as a Special Receive Condition interrupt (D_4-D_3). B. Turn on the Status Affects Vector feature, causing interrupts to modify the status vector—*meaningful* only on Channel B, but will not hurt if set for Channel A (D_2). C. Enable interrupts due to transmit buffer being empty (D_1). D. Enable External/Status interrupts (D_0).

Special Receive Condition Interrupts

A Special Receive Condition interrupt occurs (a) if a parity error has occurred, (b) if there is a receiver overrun error (data is being overwritten because the channel's three-byte receiver buffer is full and a new character is being received), or (c) if there is a framing error. The processing in this case is the following:

1. Issue an Error Reset command (to register WR0) to reset the latches in register RR1.
2. Read the character from the read buffer and discard it to empty the buffer.

It may be desirable to read and store the value of register RR1 to gather statistics on performance or determine whether to accept the character. In some applications, a character may still be acceptable if received with a framing error.

In specifying the result of reading register RR0, RR1, or specifying data, we will indicate the values as follows:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
D	D	D	D	D	D	D	D

Read a byte from the designated register.

We now present an example of processing a Special Receive Condition interrupt.

$C/\overline{D}$	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Effects and Comments
	Bits sent and received								
1	0	0	0	0	0	0	0	1	If we need to know what kind of error occurred, we point WR0 to read from RR1. Note that the Reset External/Status Interrups command is not used. This avoids losing a valid interrupt.
1	D	D	D	D	D	D	D	D	Read a byte from RR1: one or more of bit D_6 (framing error), D_5 (receive overrun error), or D_4 (parity error) will be 1 to indicate the specific error.
1	0	0	1	1	0	0	0	0	Give an Error Reset command to reset all the error latches.
0	D	D	D	D	D	D	D	D	Read in the data byte received. This must be done to clear the receiver buffer, but the character will generally be disregarded.

Received (Rx) Character Interrupts

When an Rx Character Available interrupt occurs, the character need only be read from the read buffer and stored. If parity is enabled with character lengths of 5, 6, or 7 bits, the received parity bit will be transferred with the character. Any unused bits will be 1s.

External/ Status Interrupts

To respond to an External/Status Interrupt, all that is necessary is to send a Reset External/Status Interrupts command. However, if you wish to find the specific cause of the interrupt, it is necessary to read register RR0. In this case, the complete processing takes the following form:

$C/\overline{D}$	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Effects and Comments
	Bits sent and received								
1	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Read register RR0; bit D_7 (Break), D_5 (Clear To Send), or D_3 (Data Carrier Detect) will have had a transition to indicate the cause of the interrupt.
1	0	0	0	1	0	0	0	0	Give a Reset External/Status Interrupts command to set the latches in RR0 to their current values and stop External/Status Interrupts until another transition occurs.

Transmit (Tx) Buffer Empty Interrupts

The final kind of interrupt is a Tx Buffer Empty interrupt. If another character is ready to be transmitted on this channel, a Tx Buffer Empty interrupt indicates that it is time to do so. To respond to this interrupt, you need only send the next character. If no other character is ready to transmit, it may be desirable to mark the availability of the transmit mechanism for future use. In addition, you should send a Reset Tx Interrupt Pending command. This command prevents further transmitter interrupts *until the next character has been loaded* into the transmitter buffer.

The Reset Tx Interrupt Pending command to WR0 takes the following form:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	1	0	1	0	0	0

Reset Tx Interrupt Pending command; no Tx Empty Interrupts will be given until after the next character has been placed in the transmit buffer.

Z80 Assembler Code

To take these examples further, let us use Z80 Assembler code to implement the routines for a single channel. We assume that the location stored in register WR2 points to the appropriate interrupt service routine. We also assume that the following constants have already been defined:

SIOctrl. The address of the SIO's Channel B control port (we assume Channel B in order to include code to initialize the interrupt vector).

SIOdata. The address of the SIO's Channel B data port.

X. An address pointing to locations in memory that will be used to store various values.

We will write data as binary constants; the "B" suffix indicates this. In most cases, binary constants will be referred to by the command names. We begin with the initialization routine:

```
INIT:       LD     C,SIOctrl       ;place the address of the SIO in the C register for
                                   ; use in subsequent output
            LD     A,00011000B     ;load Channel Reset command in A register
            OUT    (C),A           ;give Channel Reset command

            LD     A,00010100B     ;write to register WR0 pointing it to register WR4
            OUT    (C),A
            LD     A,11001101B     ;output basic I/O parameters to WR4
            OUT    (C),A

            LD     A,00010011B     ;write to register WR0 pointing it to register WR3
            OUT    (C),A
            LD     A,11000001B     ;output receive parameters to WR3
            OUT    (C),A

            LD     A,00010101B     ;write to register WR0 pointing it to register WR5
            OUT    (C),A
            LD     A,11101010B     ;output transmit parameters to WR5
            OUT    (C),A

            LD     A,00010010B     ;write to register WR0 pointing it to register WR2
                                   ; (Channel B only)
            OUT    (C),A
            LD     A,11100000B     ;output the interrupt vector to WR2; in this case it is
                                   ; decimal location 224
            OUT    (C),A

            LD     A,00010001B     ;write to register WR0 pointing it to register WR1
            OUT    (C),A
            LD     A,00010111B     ;output interrupt parameters to WR1
            OUT    (C),A

            RET                    ;return from initialization routine
```

Now let us look first at some sample codes for the Special Receive Condition interrupt routine, following the example above.

This is followed by a simple receive interrupt routine that will fetch the character received and store it in a temporary location.

```
SIOspecint:  PUSH   AF              ;save registers which will be used in this routine

             LD     A,00000001B     ;write to register WR0 pointing it to register RR1
             OUT    (SIOctrl),A
             IN     A,(SIOctrl)     ;fetch register RR1
             LD     (X),A           ;store result for later error analysis

             LD     A,00110000B     ;send an Error Reset command to reset device
                                    ; latches
             OUT    (SIOctrl),A

             IN     A,(SIOdata)     ;fetch the character received—we will discard this
                                    ; character since an error occurred during its
                                    ; reception

             POP    AF              ;restore saved registers
             EI                     ;enable interrupts

             RETI                   ;return from interrupt
```

Z80 Assembler Code (Continued)

```
SIOrecint:    PUSH   AF            ;save registers which will be used in this routine
              IN     A,(SIOdata)   ;fetch the character received
              LD     (X),A         ;store result for later use
              POP    AF            ;restore saved registers
              EI                   ;enable interrupts
              RETI                 ;return from interrupt
```

Of course, this last routine is probably far too simple to be useful. It is more likely that an interrupt routine will fill up a buffer of characters. A more complex example of a receive interrupt routine is contained in the chapter entitled "A Longer Example."

We now give a simple interrupt routine for an External/Status Interrupt, again assuming that the status contents of SIO register RR0 are stored in temporary location X:

```
SIOextint:    PUSH   AF            ;save registers which will be used in this routine
              LD     A,00010000B   ;send a Reset External/Status Interrupts command
              OUT    (SIOctrl),A
              IN     A,(SIOctrl)   ;fetch register RR0
              LD     (X),A         ;store result for later analysis
              POP    AF            ;restore saved registers
              EI                   ;enable interrupts
              RETI                 ;return from interrupt
```

Finally, we give the processing for a transmit interrupt routine in the case where no more characters are to be transmitted.

It is likely that this code would just be a portion of a more general transmit interrupt routine which would transmit a buffer-full of information at a time. A more complex example is included in the section entitled "A Longer Example."

```
SIOtrnint:    PUSH   AF            ;save registers which will be used in this routine
              LD     A,00101000B   ;send a Reset Tx Interrupt Pending command
              OUT    (SIOctrl),A
              POP    AF            ;restore saved registers
              EI                   ;Enable Interrupts
              RETI                 ;Return From Interrupt
```

SECTION 5

Hardware Considerations

Questions and Answers.

Q: Can a sloppy system clock cause problems in SIO operation?

A: Yes: the specifications for the system clock are very tight and must be met closely to prevent SIO malfunction. The clock high voltage must be greater than $V_{CC} - 0.6V$ but less than +5.5V. The clock low voltage must be greater than −0.3V but less than +0.45V. The transitions between these two levels must be made in less than 30 ns. This does not apply to the $\overline{\text{RxC}}$ and $\overline{\text{TxC}}$ inputs which are standard TTL levels.

Q: When is a received character available to be read?

A: Data will be available a maximum of 13 system clock cycles from the rising edge of the $\overline{\text{RxC}}$ signal which samples the last bit of the data.

Q: What is the maximum time between character-insertion for transmission and next-character transmission?

A: This will vary depending on the speed of the line over which the character is being transmitted.

Q: Are the control lines to the SIO synchronous with the system clock so that noise may exist on the buses any time before setup requirements are satisfied?

A: Yes.

Q: In asynchronous use must receiver and transmitter clock rates be the same?

A: No, the SIO allows receive and transmit for each channel to use a different clock (thus up to four different clocks for receiving and transmitting data can be used on each SIO). However, the clock multiplier for each channel must be the same.

Q: Do Wait states have to be added when using the SIO with other processors other than the Z80 CPU?

A: No, provided that setup times specified for the SIO are met.

Q: If the Auto Enables bit in register WR3 is set, will a change in state on the $\overline{\text{DCD}}$ (Data Carrier Detect) or $\overline{\text{CTS}}$ (Clear To Send) lines still cause an interrupt?

A: Yes, provided that External Status Interrupts are enabled (bit 0 in register WR1).

Q: Is the $\overline{\text{M1}}$ line used by the SIO if no interrupts are enabled?

A: No, and in this case the $\overline{\text{M1}}$ input should be tied high.

Q: Will the SIO continue to interrupt for a condition if the condition persists and the interrupt remains enabled?

A: Yes.

Q: What is the maximum data rate of the SIO?

A: It is 1/5 the rate of the system clock (CLK). For example, if the system clock operates at 4 MHz, the SIO's maximum transfer rate is 800K bits (100K bytes) per second.

Q: What pins are edge sensitive and should be strapped to avoid strange interrupts?

A: The external synchronization ($\overline{\text{SYNC}}$) pins and any other external status pins that are not used, including $\overline{\text{CTS}}$, and $\overline{\text{DCD}}$.

Q: What happens if the transmitter or receiver is disabled, while processing a character, by turning off its associated enable bit (bit 3 in register WR5 for transmit or bit 0 in register WR3 for receive)?

A: The transmitter will complete the character transmission in an orderly fashion. The receiver, however, will not finish. It will lose the character being received and no interrupt will occur.

Register Contents

Q: Does the Tx Buffer Empty (bit 2 in register RR0 get set when the last byte in the buffer is in the process of being shifted out?

A: No. The bit is set when the transmit buffer has already become empty. Similarly, the Tx Buffer Empty interrupt will not occur until the buffer is empty. The same is true for reception: the Rx Character Available bit (bit 0 in register RR0) is not set until the entire character is in the receive buffer, and the Rx Character Available interrupt will not occur until the entire character has been moved into the buffer.

Q: If an Rx Overrun error occurs (and bit 5 of register RR1 becomes latched on) because a new character has arrived, which character gets lost?

A: The most recently received character overwrites the next most recently received character.

Q: Does the Reset External/Status Interrupts command reset any of the status bits in register RR0?

A: No. However, when a transition occurs on any of the five External/Status bits in register RR0, all of the status bits are latched in their current position until a Reset External/Status Interrupts command is issued. Thus, the command does permit the appropriate bits of register RR0 to reflect the current signal values and should be done immediately after processing each transition on the channel.

Special Uses

Q: If the CPU does not have the return from interrupt sequence (RETI instruction on the Z80 CPU), how may the SIO be informed of the completion of interrupt handling?

A: This may be done by writing the Return From Interrupt command (binary, 00111000) to WR0 in Channel A of the SIO.

Q. If the CPU can be interrupted but cannot be used with vectored interrupts, how should processing be done?

A: Immediately after being interrupted, proceed in a manner similar to polling the SIO for both receive and transmit. Alternatively, the Status Affects Vector bit (bit 2 in register WR1) may be set and a 0 byte placed into the interrupt vector (register WR2 in Channel B). Then, the contents of the interrupt vector can be used to determine the cause of the interrupt and the channel on which the interrupt occurred. This can be queried by reading register RR1 of Channel B. Also, $\overline{\text{M1}}$ should be tied High and no equivalent to an interrupt acknowledge should be issued.

Q: How can the Wait/Ready $(\overline{\text{W/RDY}})$ signal be used by the CPU in asynchronous I/O?

A: The $\overline{\text{W/RDY}}$ signal is most commonly used in Block Transfer Mode with a DMA, and this use is described in the *Z80 DMA Technical Manual*. However, $\overline{\text{W/RDY}}$ may be directly connected to the Z80 CPU $\overline{\text{WAIT}}$ line in order to use the block I/O instructions OTDR, OTIR, INDR, and INIR. In this case, the SIO can be used for block transfer reception. To do this, the SIO is configured to interrupt on the first character received only (by settings bits 4 and 3 of register WR1 to 01) and additional characters are sensed using the $\overline{\text{W/RDY}}$ line. The block I/O instructions decrement a byte counter to determine when I/O is complete.

Q: Can the $\overline{\text{SYNC}}$ pin have any use in asynchronous I/O?

A: It may be used as a general-purpose input. For example, by connecting it to a modem ring indicator, the status of that ring indicator can be monitored by the CPU.

Q: How can the SIO be used to transmit characters containing fewer than 5 bits?

A: First, set bits 6 and 5 in register WR5 to indicate that five or fewer bits per character will be transmitted. The SIO then determines the number of bits to actually transmit from the data byte itself. The data byte should consist of zero or more 1s, three 0s, and the data to be transmitted. Thus, beginning the data byte with 1111000l will cause only the last bit to be transmitted:

Contents of data byte
(d = arbitrary value)

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
1	1	1	1	0	0	0	d	1
1	1	1	0	0	0	d	d	2
1	1	0	0	0	d	d	d	3
1	0	0	0	d	d	d	d	4
0	0	0	d	d	d	d	d	5

*The rightmost number of bits indicated will be transmitted.

Q: Can a Break sequence be sent for a fixed number of character periods?

A: Yes. Break is continuously transmitted as logic 0 by setting bit 4 of register WR5. You can then send characters to the transmitter as long as the Break level is desired to persist. A Break signal, rather than the characters sent, will actually be transmitted, but each bit of each character sent will be clocked as if it were transmitted. The All Sent bit, bit 0 of register RR1, is set to 1 when the last bit of a character is clocked for transmission, and this may be used to determine when to reset bit 4 of register WR5 and stop the Break signal.

Q: If a Break sequence is initiated by setting bit 4 of register WR5, will any character in the process of being transmitted be completed?

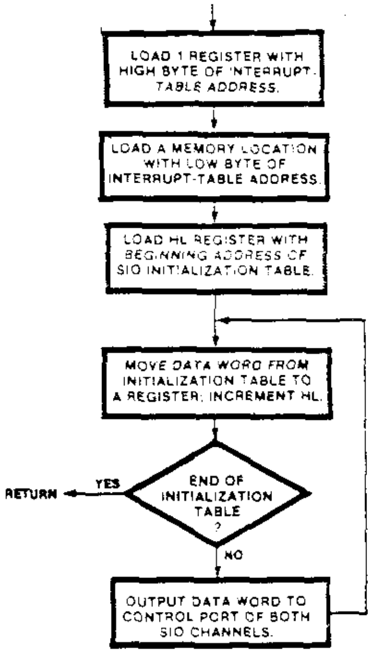
A: No. Break is effective immediately when bit 4 of WR5 is set. The "all sent" bit in register RR1 should be monitored to determine when it is safe to initiate a Break sequence.

A Longer Example.

In this section, we give a longer example of asynchronous interrupt-driven full-duplex I/O using the SIO. The code for this example is contained in Appendix A, and the basic routines are flow charted in Figures 7-12.

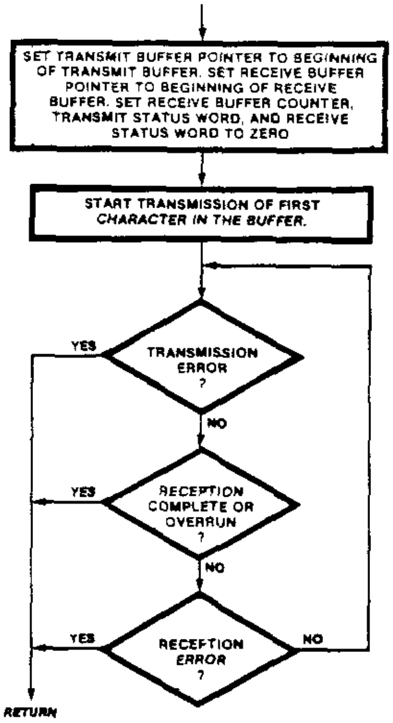
The example includes code for initialization of the SIO, initialization of a receive buffer interrupt routine, and a transfer routine which causes a buffer of up to 80 characters of information to be transmitted on Channel A and a buffer of up to 80 characters of information to be received from Channel A. The transfer *routine stops when either all data is received* or an error occurs. Completion of an operation *on a buffer for both receive and transmit* is indicated by a carriage return character. Additional routines (not included in this example) would be needed to call the initialization code and initiate the transfer routine. Therefore, we do not present a complete example; that would only be possible when all details of a particular communication environment and operating system were known.

The code begins by defining the value of the SIO control and data channels, followed by location definitions for the interrupt vector. There is then a series of constant definitions of the various fields in each register of the SIO. This is followed by a table-driven SIO initiali*zation routine called "SIO_init," shown in* Figure 7, which uses the table beginning at the location *"SIOItable." The SIO_Init routine* initializes the SIO with exactly the same



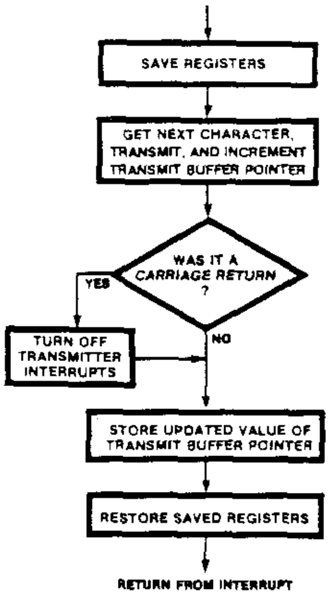
LOAD 1 REGISTER WITH HIGH BYTE OF INTERRUPT-TABLE ADDRESS.
LOAD A MEMORY LOCATION WITH LOW BYTE OF INTERRUPT-TABLE ADDRESS.
LOAD HL REGISTER WITH BEGINNING ADDRESS OF SIO INITIALIZATION TABLE.
MOVE DATA WORD FROM INITIALIZATION TABLE TO A REGISTER; INCREMENT HL.
END OF INITIALIZATION TABLE ?
YES
RETURN
NO
OUTPUT DATA WORD TO CONTROL PORT OF BOTH SIO CHANNELS.

Figure 7. Interrupt-Driven Initialization Routine



SET TRANSMIT BUFFER POINTER TO BEGINNING OF TRANSMIT BUFFER. SET RECEIVE BUFFER POINTER TO BEGINNING OF RECEIVE BUFFER. SET RECEIVE BUFFER COUNTER, TRANSMIT STATUS WORD, AND RECEIVE STATUS WORD TO ZERO
START TRANSMISSION OF FIRST CHARACTER IN THE BUFFER.
TRANSMISSION ERROR ?
YES
NO
RECEPTION COMPLETE OR OVERRUN ?
YES
NO
RECEPTION ERROR ?
YES
NO
RETURN

Figure 8. Interrupt-Driven Transmit Routine



SAVE REGISTERS
GET NEXT CHARACTER, TRANSMIT, AND INCREMENT TRANSMIT BUFFER POINTER
WAS IT A CARRIAGE RETURN ?
YES
NO
TURN OFF TRANSMITTER INTERRUPTS
STORE UPDATED VALUE OF TRANSMIT BUFFER POINTER
RESTORE SAVED REGISTERS
RETURN FROM INTERRUPT

Figure 9. Transmitter Buffer Empty Interrupt Routine

A Longer Example (Continued)

parameters as the interrupt-driven example in the previous section. The table-driven version is presented simply as an alternative means of coding this material.

A short routine for filling the receive buffer with "FF" (hex) characters and buffer definitions follows the SIO_Init routine. This in turn is followed by the transfer routine, Figure 8, which begins transmitting on Channel A; transmission and reception is thereafter directed by the interrupt routines. After the transfer routine begins output, it checks for various error conditions and loops until there is either completion or an error.

Then the four interrupt routines follow: TxBEmpty, Figure 9, is called on a transmit buffer interrupt; it begins transmission of the next character in the buffer. A carriage return stops transmission. RecvChar, Figure 10, is called on a normal receive interrupt; it places the received character in the buffer if the buffer is not full and updates receive counters. The routines SpRecvChar, Figure 11, and ExtStatus, Figure 12, are error interrupts; they update information to indicate the nature of the error.

The code of this example can be used in a situation where data is being sent to a device which echoes the data sent. In such a case, the transmit and receive buffers could be compared upon completion for line or transmission errors.

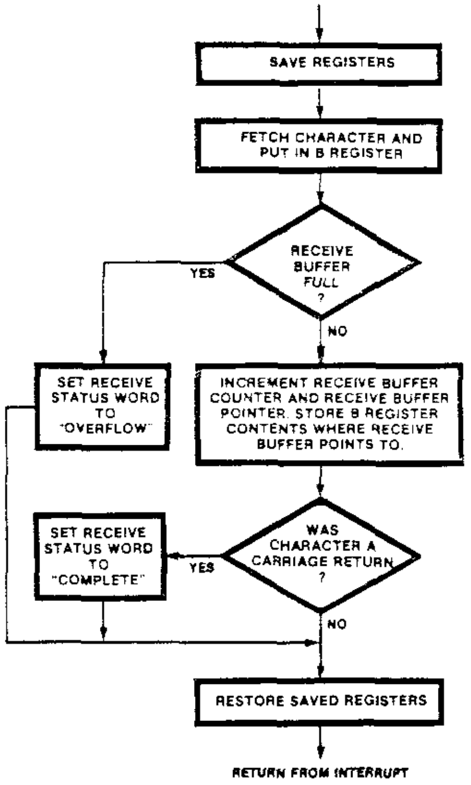

SAVE REGISTERS
FETCH CHARACTER AND PUT IN B REGISTER
RECEIVE BUFFER FULL ?
YES
NO
SET RECEIVE STATUS WORD TO "OVERFLOW"
INCREMENT RECEIVE BUFFER COUNTER AND RECEIVE BUFFER POINTER. STORE B REGISTER CONTENTS WHERE RECEIVE BUFFER POINTS TO.
WAS CHARACTER A CARRIAGE RETURN ?
YES
NO
SET RECEIVE STATUS WORD TO "COMPLETE"
RESTORE SAVED REGISTERS
RETURN FROM INTERRUPT

Figure 10. Receive Character **Interrupt Routine**

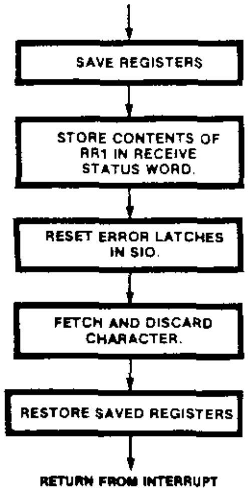

SAVE REGISTERS
STORE CONTENTS OF RR1 IN RECEIVE STATUS WORD.
RESET ERROR LATCHES IN SIO.
FETCH AND DISCARD CHARACTER.
RESTORE SAVED REGISTERS
RETURN FROM INTERRUPT

Figure 11. Special Receive Condition **Interrupt Routine**

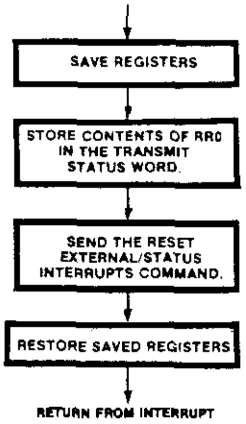

SAVE REGISTERS
STORE CONTENTS OF RR0 IN THE TRANSMIT STATUS WORD.
SEND THE RESET EXTERNAL/STATUS INTERRUPTS COMMAND.
RESTORE SAVED REGISTERS
RETURN FROM INTERRUPT

Figure 12. External/Status **Interrupt Routine**

Appendix A

Interrupt-Driven Code Example

SIO Port Identifiers and System Address Bus Addresses

```
SIO:        EQU   40H

SIOAData:   EQU   SIO + 1
SIOACtrl:   EQU   SIO + 2
SIOBData:   EQU   SIO + 3
SIOBCtrl:   EQU   SIO + 4
```

Table of Interrupt Vectors

The table (Int_Tab) starts at the lowest priority vector, which should be dddd000d.

```
            ORG   0D0H        ;starts at address with low
                              ; byte = 11010000
Int_Tab:    DEFW  TxBEmpty    ;;interrupt types for Channel B
            DEFW  ExtStat
            DEFW  RxChar
            DEFW  SpRxCond

            DEFW  TxBEmpty    ;;interrupt types for Channel A
            DEFW  ExtStat
            DEFW  RxChar
            DEFW  SpRxCond
```

Command Identifiers and Values

Includes all control bytes for asynchronous and synchronous I/O.

WR0 Commands

```
R0:         EQU   00H         ;SIO register pointers
R1:         EQU   01H
R2:         EQU   02H
R3:         EQU   03H
R4:         EQU   04H
R5:         EQU   05H
R6:         EQU   06H
R7:         EQU   07H

NC:         EQU   00H         ;Null Code
SA:         EQU   08H         ;Send Abort (SDLC)
RESI:       EQU   10H         ;Reset Ext/Stat Int
CHRST:      EQU   18H         ;Channel Reset
EIONRC:     EQU   20H         ;Enable Int On Next Rx Char
RTIP:       EQU   28H         ;Reset Tx Int Pending
ER:         EQU   30H         ;Error Reset
RFI:        EQU   38H         ;Return From Int
RRCC:       EQU   40H         ;Reset Rx CRC Checker
RTCG:       EQU   80H         ;Reset Tx CRC Generator
RTUEL:      EQU   0C0H        ;Reset Tx Under/EOM Latch
```

WR1 Commands

```
WAIT:       EQU   00H         ;Wait function
DRCVRI:     EQU   00H         ;Disable Receive interrupts
EXTIE:      EQU   01H         ;External interrupt enable
XMTRIE:     EQU   02H         ;Transmit interrupt enable
SAVECT:     EQU   04H         ;Status affects vector
FIRSTC:     EQU   08H         ;Rx interrupt on first character
PAVECT:     EQU   10H         ;Rx interrupt on all characters
                              ; (parity affects vector)
PDAVCT:     EQU   18H         ;Rx interrupt on all characters
                              ; (parity doesn't affect vector)
WRONRT:     EQU   20H         ;Wait/Ready on receive
RDY:        EQU   40H         ;Ready function
WRDYEN:     EQU   80H         ;Wait/Ready enable
```

WR2 Commands

```
IV:         EQU   00H
```

WR3 Commands

```
B5:         EQU   00H         ;Receive 5 bits/character
RENABL:     EQU   01H         ;Receiver enable
ENRCVR:     EQU   01H         ;Receiver enable
SCLINH:     EQU   02H         ;Sync character load inhibit
ADSRCH:     EQU   04H         ;Address search mode
RCRCEN:     EQU   08H         ;Receive CRC enable
HUNT:       EQU   10H         ;Enter hunt mode
AUTOEN:     EQU   20H         ;Auto enables
B7:         EQU   40H         ;Receive 7 bits/character
B6:         EQU   80H         ;Receive 6 bits/character
B8:         EQU   0C0H        ;Receive 8 bits/character
```

WR4 Commands

```
SYNC:       EQU   00H         ;Sync modes enable
NOPRTY:     EQU   00H         ;Disable parity
ODD:        EQU   00H         ;Odd parity
MONO:       EQU   00H         ;8 bit sync character
C1:         EQU   00H         ;X1 clock mode
PARITY:     EQU   01H         ;Enable parity
EVEN:       EQU   02H         ;Even parity
S1:         EQU   04H         ;1 stop bit/character
S1HALF:     EQU   08H         ;1 and a half stop bits/character
S2:         EQU   0CH         ;2 stop bits/character
BISYNC:     EQU   10H         ;16 bit sync character
SDLC:       EQU   20H         ;SDLC mode
ESYNC:      EQU   30H         ;External sync mode
C16:        EQU   40H         ;X16 clock mode
C32:        EQU   80H         ;X32 clock mode
C64:        EQU   0C0H        ;X64 clock mode
```

WR5 Commands

```
T5:         EQU   00H         ;Transmit 5 bits/character
XCRCEN:     EQU   01H         ;Transmit CRC enable
RTS:        EQU   02H         ;Request to send
SELCRC:     EQU   04H         ;Select CRC-16 polynomial
XENABL:     EQU   08H         ;Transmitter enable
BREAK:      EQU   10H         ;Send break
T7:         EQU   20H         ;Transmit 7 bits/character
T6:         EQU   40H         ;Transmit 6 bits/character
T8:         EQU   60H         ;Transmit 8 bits/character
DTR:        EQU   80H         ;Data terminal ready
```

Initialization

```
SIO_Init:   LD    HL, Int_Tab
            LD    A,H
            LD    I,A
            LD    A,L
            LD    (I_Loc),A
            LD    HL, SIOItable

Init_Loop:  LD    A,(HL)            ;loop for initialization
            INC   HL
            CP    0
            RET   Z
            OUT   (SIOACtrl),A
            OUT   (SIOBCtrl),A
            JR    Init_Loop

SIOItable:  DEFB  CR                ;table for initialization
            DEFB  R4 + RESI
            DEFB  C64 + ODD + PARITY + S2
            DEFB  R3 + RESI
            DEFB  B8 +  AUTOEN + ENRCVR
            DEFB  R5 + RESI
            DEFB  DTR + RTS + T8 + XENABL
            DEFB  R2 + RESI

I_Loc:      DEFS  1                 ;location of int table
            DEFB  R1 + RESI         ;address
            DEFB  EXTIE + XMTRIE + SAVECT + PAVECT
            DEFB  0
```

Receiver Buffer Initialization

```
Buf_Init:   LD      A,BufLength   ;fill receiver buffer
            LD      B,A           ; with FF characters
            LD      HL,RBuffer    ; to detect errors
            LD      A,0FFH

Buf_I:      LD      (HL),A        ;a loop for Buf_Init
            INC     HL
            DJNZ    Buf_I
            RET

BufLength:  EQU     80            ;buffer length
XBuffer:    DEFS    BufLength     ;Tx buffer starting location
RBuffer:    DEFS    BufLength     ;Rx buffer starting location

XBufPtr:    DEFS    2             ;Tx pointer
RBufPtr:    DEFS    2             ;Rx pointer
RBufCtr:    DEFS    1             ;Rx counter
```

Transmit Routine [illegible]

Initiates transmission of a buffer full of data and terminates when an error is detected or a complete buffer has been received

```
RxStat:     DEFS    1             ;Receive Status Word
TxStat:     DEFS    1             ;Transmit Status Word

Complete:   EQU     1
CR:         EQU     0DH
Break:      EQU     80H
EOM:        EQU     80H
Overflow:   EQU     0FFH

Transfer:   LD      HL,XBuffer    ;setup to begin Tx
            INC     HL
            LD      (XBufPtr),HL
            LD      HL,RBuffer
            LD      (RBufPtr),HL
            XOR     A             ;A = 0
            LD      (RBufCtr),A
            LD      (TxStat),A
            LD      (RxStat),A

            LD      A,SIOAData    ;start Tx task
            LD      C,A
            LD      HL,(XBuffer)  ;first character
            LD      A,(HL)
            OUT     (C),A

Tloop:      LD      A,(TxStat)    ;await Tx completion or error
            CP      0
            RET     NZ
            LD      A,(RxStat)
            CP      Overflow
            RET     Z
            CP      Complete
            RET     Z
            JR      NZ,Tloop
            RET
```

Transmitter Buffer Empty Routine see Figure [illegible]

```
TxBEmpty    PUSH    AF
            PUSH    BC
            PUSH    HL

            LD      HL,(XBufPtr)
            LD      A,SIOAData
            LD      C,A
            LD      A,(HL)
            OUTI
            CP      CR
            JR      NZ, TxBExit   ;last character?

            LD      A,RTIP        ;Reset Tx Int Pending
            INC     C
            OUT     (C),A         ;to control port

TxBExit:    LD      (XBufPtr),HL  ;save pointer
            POP     HL
            POP     BC
            POP     AF
            EI
            RETI
```

Receive Character Routine see Figure [illegible]

```
RxChar:     PUSH    AF
            PUSH    BC

            LD      A,SIOAData
            LD      C,A
            IN      A,(C)         ;get character
            LD      B,A
            LD      A,(RBufCtr)
            CP      BufLength
            JR      Z,Over

            INC     A             ;bump counter
            LD      (RBufCtr),A
            LD      A,B
            LD      HL,(RBufPtr)  ;bump pointer
            LD      (HL),A
            INC     HL
            LD      (RBufPtr),HL
            CP      CR
            JR      NZ,RxExit

            LD      A,Complete
            LD      (RxStat),A
            JR      RxExit

Over:       LD      A,Overflow    ;indicate error
            LD      (RxStat),A

RxExit:     POP     BC
            POP     AF
            EI
            RETI
```

Special Receive Condition Routine see Figure [illegible]

```
SpRxCond:   PUSH    AF
            PUSH    BC

            LD      A,SIOAData
            LD      C,A
            LD      A,R1          ;get RR1
            INC     C
            OUT     (C),A
            IN      A,(C)
            LD      (RxStat),A    ;save status
            LD      A,ER          ;Reset Errors
            DEC     C
            OUT     (C),A
            DEC     C
            IN      A,(C)         ;get character

            POP     BC
            POP     AF
            EI
            RETI
```

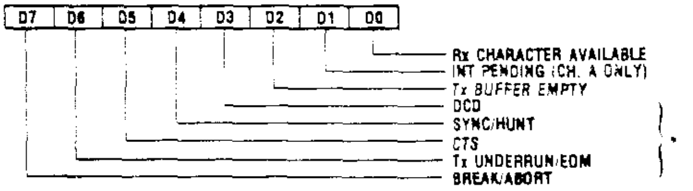
External/Status Routine see Figure 12

```
ExtStatus:  PUSH    AF
            PUSH    BC

            LD      A,SIOACtrl
            LD      C,A
            IN      A,(C)         ;get RR0
            LD      (TxStat),A
            LD      A,RESI        ;Reset Ext Stat Int
            OUT     (C),A

            POP     BC
            POP     AF
            EI
            RETI
END
```

Appendix B

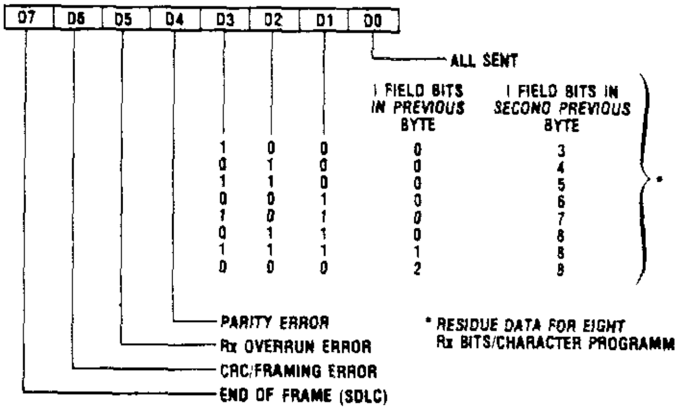
Read Register Bit Functions

READ REGISTER 0



* USED WITH "EXTERNAL/STATUS INTERRUPT" MODE

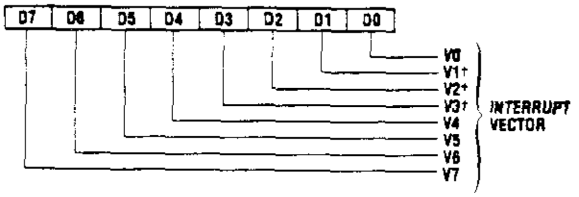
READ REGISTER 1†



* RESIDUE DATA FOR EIGHT Rx BITS/CHARACTER PROGRAMMED

† USED WITH SPECIAL RECEIVE CONDITION MODE

READ REGISTER 2

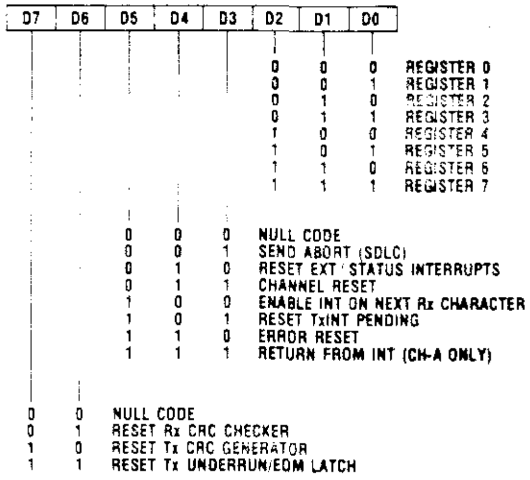


†VARIABLE IF "STATUS AFFECTS VECTOR" IS PROGRAMMED

Appendix C

Write Register Bit Functions

WRITE REGISTER 0



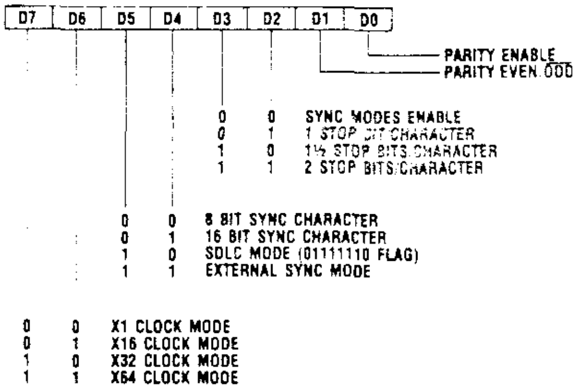
D7	D6	D5	D4	D3	D2	D1	D0

0	0	0	REGISTER 0
0	0	1	REGISTER 1
0	1	0	REGISTER 2
0	1	1	REGISTER 3
1	0	0	REGISTER 4
1	0	1	REGISTER 5
1	1	0	REGISTER 6
1	1	1	REGISTER 7

0	0	0	NULL CODE
0	0	1	SEND ABORT (SDLC)
0	1	0	RESET EXT/STATUS INTERRUPTS
0	1	1	CHANNEL RESET
1	0	0	ENABLE INT ON NEXT Rx CHARACTER
1	0	1	RESET TxINT PENDING
1	1	0	ERROR RESET
1	1	1	RETURN FROM INT (CH-A ONLY)

0	0	NULL CODE
0	1	RESET Rx CRC CHECKER
1	0	RESET Tx CRC GENERATOR
1	1	RESET Tx UNDERRUN/EOM LATCH

WRITE REGISTER 4



D7	D6	D5	D4	D3	D2	D1	D0

PARITY ENABLE

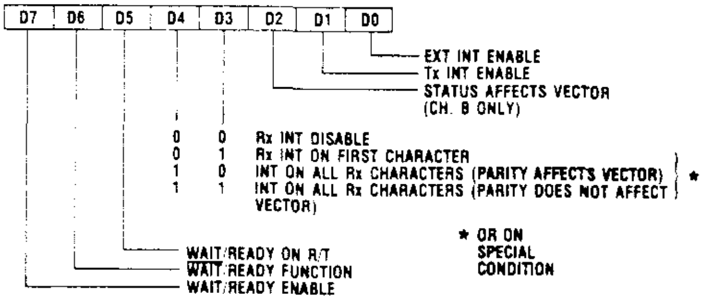
PARITY EVEN/ODD

0	0	SYNC MODES ENABLE
0	1	1 STOP BIT/CHARACTER
1	0	1½ STOP BITS/CHARACTER
1	1	2 STOP BITS/CHARACTER

0	0	8 BIT SYNC CHARACTER
0	1	16 BIT SYNC CHARACTER
1	0	SDLC MODE (01111110 FLAG)
1	1	EXTERNAL SYNC MODE

0	0	X1 CLOCK MODE
0	1	X16 CLOCK MODE
1	0	X32 CLOCK MODE
1	1	X64 CLOCK MODE

WRITE REGISTER 1



D7	D6	D5	D4	D3	D2	D1	D0

EXT INT ENABLE

Tx INT ENABLE

STATUS AFFECTS VECTOR (CH. B ONLY)

0	0	Rx INT DISABLE
0	1	Rx INT ON FIRST CHARACTER
1	0	INT ON ALL Rx CHARACTERS (PARITY AFFECTS VECTOR) *
1	1	INT ON ALL Rx CHARACTERS (PARITY DOES NOT AFFECT VECTOR) *

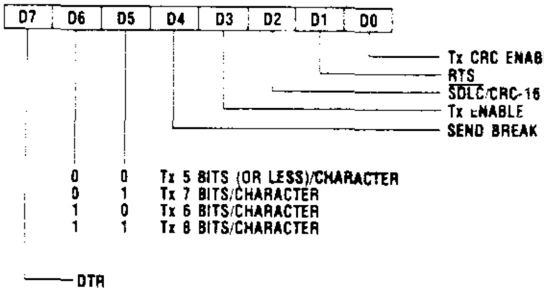
* OR ON SPECIAL CONDITION

WAIT/READY ON R/T

$\overline{\text{WAIT}}$/READY FUNCTION

WAIT/READY ENABLE

WRITE REGISTER 5



D7	D6	D5	D4	D3	D2	D1	D0

Tx CRC ENABLE

RTS

$\overline{\text{SDLC}}$/CRC-16

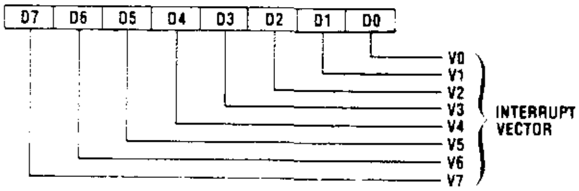
Tx ENABLE

SEND BREAK

0	0	Tx 5 BITS (OR LESS)/CHARACTER
0	1	Tx 7 BITS/CHARACTER
1	0	Tx 6 BITS/CHARACTER
1	1	Tx 8 BITS/CHARACTER

DTR

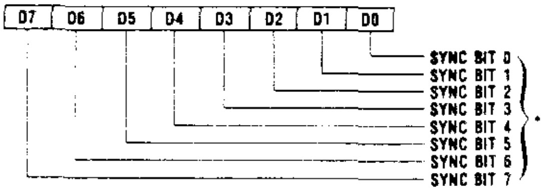
WRITE REGISTER 2 (CHANNEL B ONLY)



D7	D6	D5	D4	D3	D2	D1	D0
V7	V6	V5	V4	V3	V2	V1	V0

INTERRUPT VECTOR

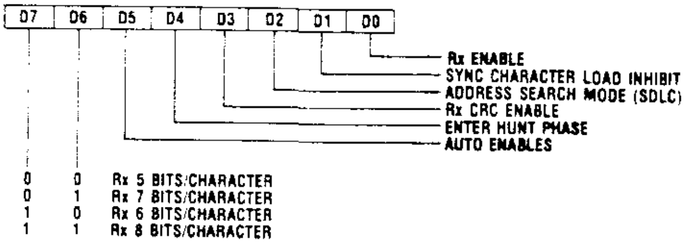
WRITE REGISTER 6



D7	D6	D5	D4	D3	D2	D1	D0
SYNC BIT 7	SYNC BIT 6	SYNC BIT 5	SYNC BIT 4	SYNC BIT 3	SYNC BIT 2	SYNC BIT 1	SYNC BIT 0

*ALSO SDLC ADDRESS FIELD

WRITER REGISTER 3



D7	D6	D5	D4	D3	D2	D1	D0

Rx ENABLE

SYNC CHARACTER LOAD INHIBIT

ADDRESS SEARCH MODE (SDLC)

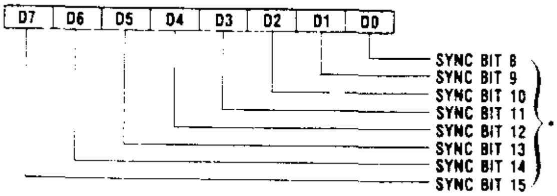
Rx CRC ENABLE

ENTER HUNT PHASE

AUTO ENABLES

0	0	Rx 5 BITS/CHARACTER
0	1	Rx 7 BITS/CHARACTER
1	0	Rx 6 BITS/CHARACTER
1	1	Rx 8 BITS/CHARACTER

WRITE REGISTER 7



D7	D6	D5	D4	D3	D2	D1	D0
SYNC BIT 15	SYNC BIT 14	SYNC BIT 13	SYNC BIT 12	SYNC BIT 11	SYNC BIT 10	SYNC BIT 9	SYNC BIT 8

*FOR SDLC IT MUST BE PROGRAMMED TO "01111110" FOR FLAG RECOGNITION