Personal Computer

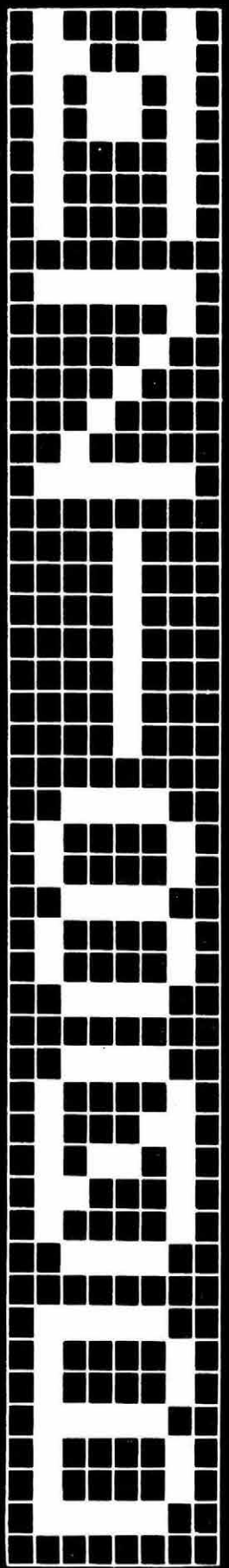
MZ-80B

RS-232C
SERIAL INTERFACE

SHARP

Personal Computer
MZ-80B

RS-232C
Serial Interface Card
Model: MZ-8BI03
Technical Manual



811124-1000-K

Contents

1. General Information

Introduction

There are two methods of data communication between computer and external equipment: 8-bit parallel and bit serial.

The serial interface card MZ-8BI03 (hereinafter referred to as "interface card") permits data communication by the bit serial method. This interface card is manufactured in accordance with EIA RS-232C (the Electronic Industries Association RS-232C), and used for data communication with other equipment having interface based on RS-232C.

Functions of this interface card

The interface card has the following functions.

1. One card has two channels, each of which is capable of transmitting/receiving data independently.
2. One of the ten baud rates can be selected by operating the switch on the card. Baud rates can be set independently for the two channels.
3. Output connector signals to external equipment can be in either terminal mode or modem mode through the operation of the jumper chip.
4. This interface card can be used as 20mA current loop for one channel.

Applications

Equipped with the above-mentioned functions, the interface card has a variety of applications. Some applications of this very versatile serial interface card are shown below.

1. Data communication between computers on telephone line via acoustic couplers
2. Printer
3. Plotter
4. Digitizer
5. Graphic display
6. Card reader
7. Magnetic tape equipment

2. Specifications

Communication mehod:	Asynchronous
Standard:	In compliance with EIA RS-232C
Control LSI:	Z-80SIO/0
Number of channels:	2 (Channel A and Channel B)
20mA current loop:	Changeover is allowed for one channel (Channel B)
Baud rate:	Can be set independently for the two channels (Manual setting using switch)
Number of baud rates:	10 (75, 110, 150, 300, 600, 1200, 1800, 2400, 4800, 9600 baud)
Character length:	5, 6, 7 or 8 bits (Selection by software)
Parity bit:	Odd, none, or even
Stop bit:	1, 1½, or 2
Mode:	Either terminal mode or modem mode can be selected for each channel (through the use of jumper chip).
Interrupt:	Z-80 vector interrupt can be used.
Port address:	Manual setting with switch
Operating temperature:	0°C ~ 50°C
Storage temperature:	−25°C ~ 80°C

(Note) The above-mentioned specifications may be changed in the future for improvement of the product.

3. Cautions for Handling

- This interface card has been especially designed for Sharp personal computer MZ-80B. Do not use this card for other equipment, or destruction of circuit or other troubles may be caused.

- Since the interface has NMOS, CMOS, and other highly integrated ICs incorporated in it, take adequate precautions in handling the card so that no static electricity will be generated. Otherwise, the ICs may be destroyed.

- When pulling the card out of the expansion unit, do NOT pull the signal cable. Otherwise, disconnection of wire may be caused.

- Set the switch, jumper, etc. on the card only when power is turned off.

- Do not turn ON more than one switches at a time for baud rate setting. Otherwise, the ICs may be destroyed.

- The manufacturer will not be held responsible for troubles resulting from the alteration of the circuit by the user.

- For storage, protect this interface card from static electricity, for example, by putting it in an electrically conductive bag.

- Do not leave the interface card in the following places.
  - Place with too much/little humidity
  - Place exposed to direct sunlight
  - Dusty place
  - Excessively hot/cold place
  - Place with a lot of vibration

4. Method of Operation

Arrangement of switches and jumper chips

Before using this card, the switches and jumpers on the card must be set first. Fig. 4-1 shows the positions of the switches and jumpers.

Switch SW-1:	Used for setting port address
Switch SW-2:	Used for setting baud rate for Channel B
Switch SW-3:	Used for setting baud rate for Channel A
Jumper block J-1:	Jumper for mode selection for Channel B
Jumper block J-2:	Jumper for selection of RS-232C/current loop (for channel B)
Jumper block J-3:	Jumper for mode selection for Channel A

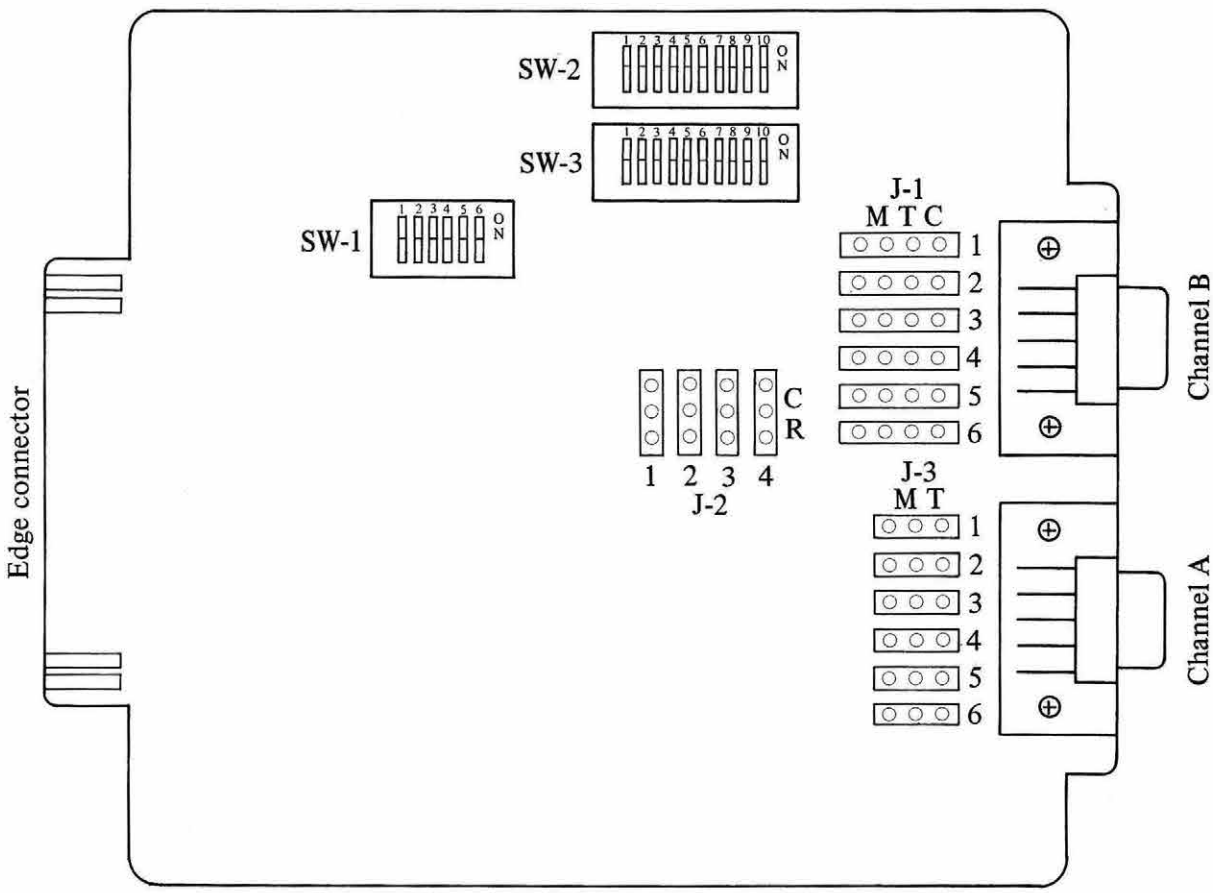


Fig. 4-1. Arrangement of Switches and Jumpers

Setting port address

The Z80-CPU outputs 8-bit port address. Since this card uses 4 sequential port addresses, the port address can be set by setting upper 6 bits of the 8-bit address signal using the switch (SW-1).

As shown in Table 4-1, the upper 6 bits ($A_2 \sim A_7$) of the 8-bit address signal can be selected arbitarily.

Table 4-1. Switch for Setting Port Address

Address bit	Segment of switch SW-1	Manufacturer's setting	Comment
A_7	6	OFF	These bits can be set arbitarily.
A_6	5	ON	
A_5	4	OFF	
A_4	3	OFF	
A_3	2	ON	
A_2	1	ON	
A_1	–	–	Channel select
A_0	–	–	Control or Data Select

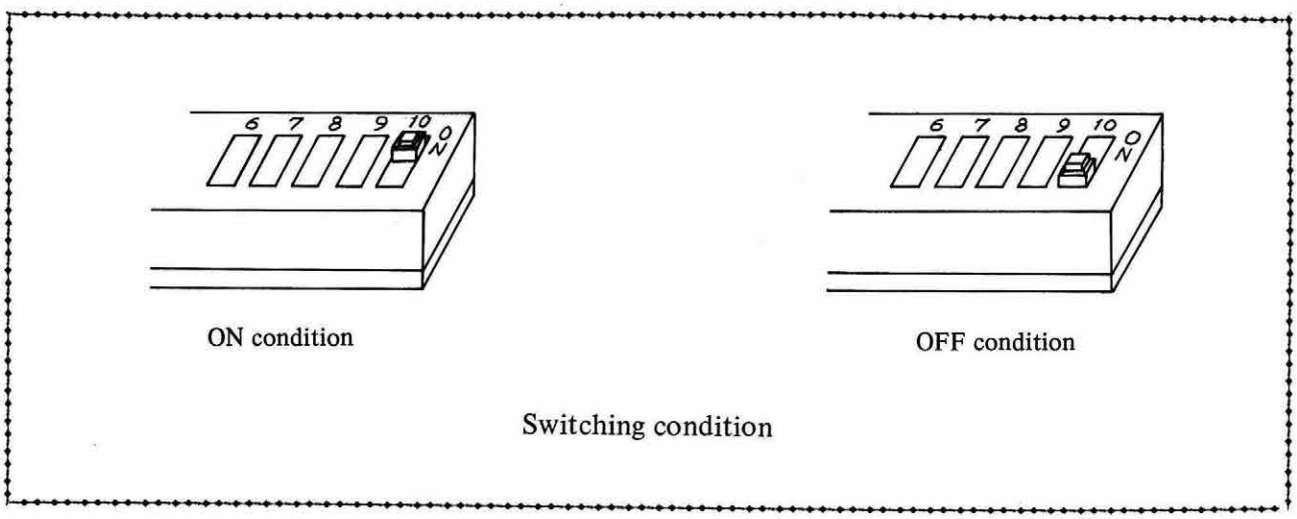
The correspondence between the switching positions of the switch SW-1 and logic levels is as shown below. The address of this interface card is the port address whose logic level coincides with that of the switch SW-1.

Switching position	Logic level
ON	0
OFF	1

Table 4-1 shows the state of the switch SW-1 as set by the manufacturer for delivery. In this case, the port addresses are B0H, B1H, B2H, and B3H.

Address bit A_1 is assigned for selecting channels for Z80-SIO whereas address bit A_0 is assigned for selection of control word or data.

A_1	A_0	Selection
0	0	Channel A/Data
0	1	Channel A/Control word
1	0	Channel B/Data
1	1	Channel B/Control word



Setting baud rate for each channel

The interface card has two channels (Channel A and Channel B), for each of which baud rate can be set independently. This operation can be carried out by selecting the switches SW-2 and SW-3 on the card. One out of the 10 baud rates can be selected.

Switch SW-2	For setting baud rate for Channel B
Switch SW-3	For setting baud rate for Channel A

The setting methods are the same for both channels. Turn on the switch corresponding to the desired baud rate and turn off the other switches. The correspondence between the segments of the switches and baud rates is shown in Table 4-2.

Table 4-2. Setting Baud Rates

Baud Rate	Segments of switches SW-2 and SW-3									
	1	2	3	4	5	6	7	8	9	10
75 baud	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
110	OFF	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
150	OFF	OFF	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF
300	OFF	OFF	OFF	ON	OFF	OFF	OFF	OFF	OFF	OFF
600	OFF	OFF	OFF	OFF	ON	OFF	OFF	OFF	OFF	OFF
1200	OFF	OFF	OFF	OFF	OFF	ON	OFF	OFF	OFF	OFF
1800	OFF	OFF	OFF	OFF	OFF	OFF	ON	OFF	OFF	OFF
2400	OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON	OFF	OFF
4800	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON	OFF
9600	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON ← Factory Setting

(Caution) ***In setting baud rate, only one of the 10 segments of the switch SW-2 or SW-3 must be turned ON. Turning ON 2 or more segments will cause destruction of the ICs. After setting, confirm that only one segment of each switch is turned ON.***

Modes of connector signals

This interface card has two 9-pin connectors as shown in Fig. 4-2.

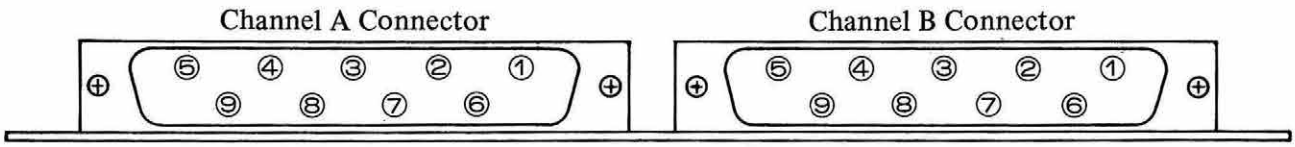


Fig. 4-2. Connector Pin Configuration

By changing the wiring of the jumper blocks J-1, J-2, J-3 on the card, signals for each connector pin can be changed. Through this operation, the channels can be set in three different states: terminal mode, modem mode, and current loop.

Terminal and modem modes are in compliance with RS-232C standard. In both modes, binary signals are transmitted by voltage levels.

Current loop is not included in the RS-232C standard, but is adopted in ASR-33 manufactured by Teletype Corporation. By this method, binary signals are transmitted with intermittent 20mA current.

The modes available for each channel are given in Table 4-3.

Table 4-3. Channel Mode

Mode \ Channel	Channel A	Channel B
Terminal mode	Available	Available
Modem mode	Available	Available
Current loop	Not available	Available

The correspondence between connector pins and signals in different modes is shown in Table 4-4. For connector pin numbers, refer to Fig. 4-2.

Table 4-4. Signal Configuration

Connector pin No.	Terminal mode	Modem mode	Current loop
1	Security grounding	Security grounding	Security grounding
2	Transmit data TxD	Receive data RxD	20mA data TxD
3	Receive data RxD	Transmit data TxD	20mA data return
4	Request to send RTS	Clear to send CTS (Transmitter enable)	20mA receive RxD
5	Clear to send CTS (Transmitter enable)	Request to send RTS	20mA receive return
6	Data terminal ready DTR	Data carrier detect DCD (Receiver enable)	TTY TAPE control return DTR
7	Data carrier detect DCD (Receiver enable)	Data terminal ready DTR	TTY TAPE control return
8	Grounding for signals	Grounding for signals	Grounding for signals
9	NC	NC	NC

In the terminal mode, transmit data are connected to pin No. 2. However, in the modem mode, receive data are connected to pin No. 2, thus creating a reverse flow of signals. The directions of each connector pin are opposite in the terminal and modem modes. This feature can be utilized in the following way.

- Terminal mode is selected for connection with acoustic coupler
- Modem mode is selected for connection with printers, plotters, etc. equipped with RS-232C interface.

Of course, there are some exceptions. Therefore, read the instruction manuals of individual equipment.

Setting Channel A

The mode of Channel A can be set by changing jumper wiring of the jumper block J-3. Terminal mode can be selected by short-circuiting all pairs of terminals marked "T" by inserting jumper chips between them as illustrated in Fig. 4-3. Modem mode can be selected by doing the same for pairs of terminals marked "M".

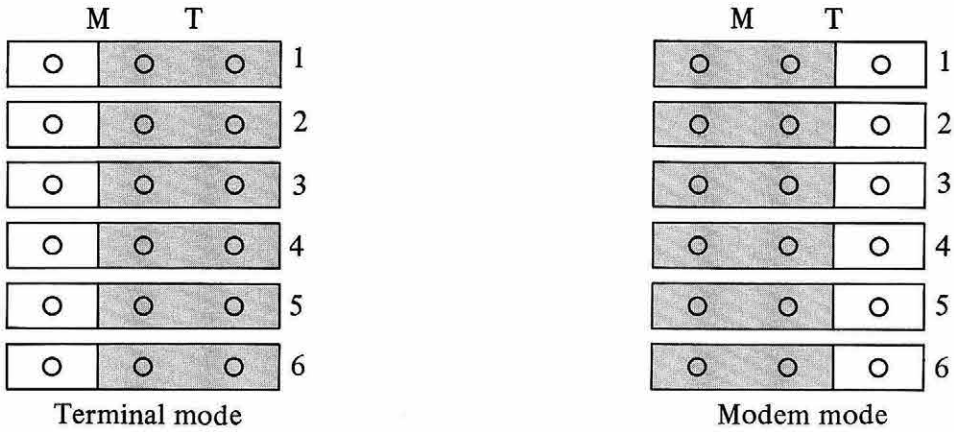
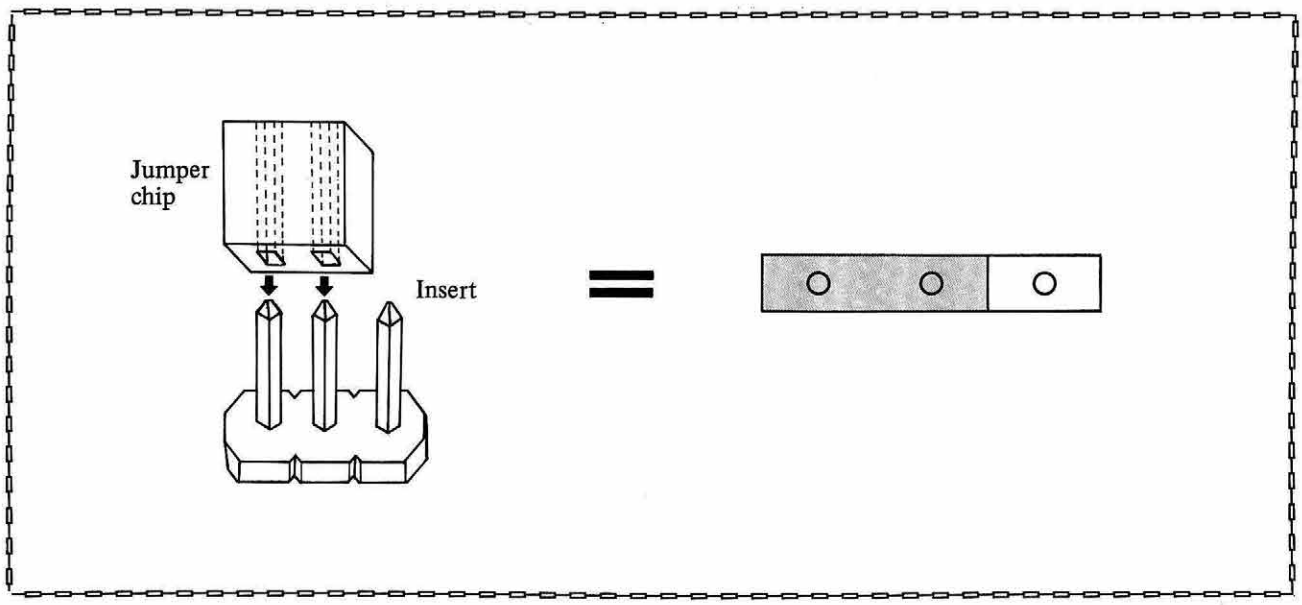


Fig. 4-3 Setting Modes for Channel A (Jumper block J-3)

Incidentally, Channel A is set at terminal mode for delivery.



Setting Channel B

Either RS-232C or current loop must be selected for Channel B. For this purpose, change the jumper wiring of the jumper block J-2. Current loop can be selected by short-circuiting all pairs of terminals marked "C" with the use of jumper chips, and RS-232C can be selected by doing the same for all apirs of terminals marked "R", as shown in Fig. 4-4.

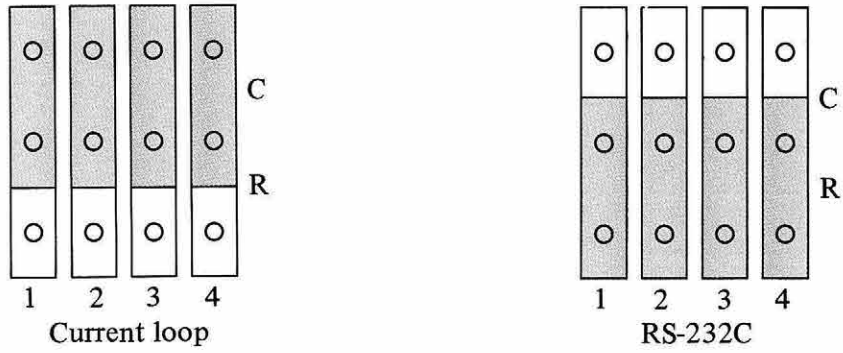


Fig. 4-4 Jumper Block J-2

Incidentally, the jumper block J-2 is set at RS-232C side for delivery.

Next, Channel B can be set at terminal mode, modem mode, or current loop by manipulating the jumper block J-1. The wiring of jumper chips is illustrated in Fig. 4-5.

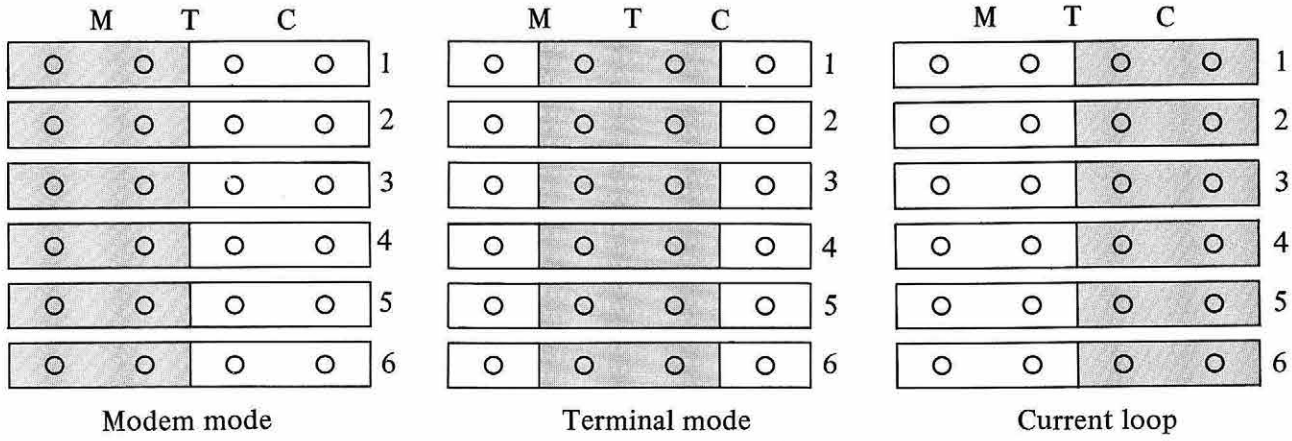


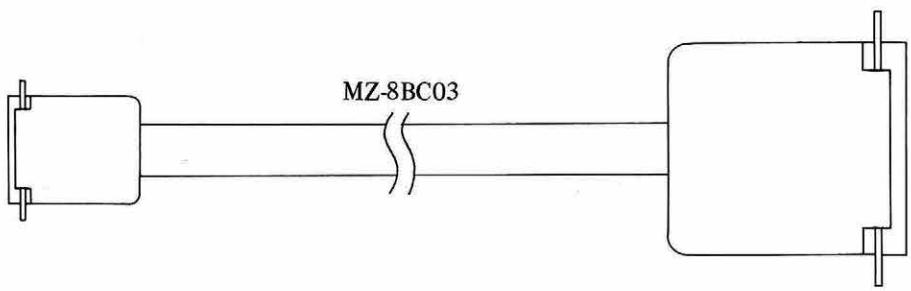
Fig. 4-5 Jumper Block J-1

The jumper block J-1 is set at modem mode at the time of delivery.

Installation of the interface card

This card is designed for installation in Sharp personal computer MZ-80B. Install the card in the following procedure.

1. Set port address, baud rate, and mode of each channel. For this purpose, set the switches and jumper blocks on the card.
2. Insert this card into the slot of the MZ-80B expansion unit. There are 6 slots in the expansion unit. Use any one of slot Nos. 1, 2, 4, and 5. For the method of insertion and the installation of expansion unit, refer to the Owner's Manual for MZ-80B.
3. Connect the card with other equipment, using signal cable. Signal cable MZ-8BC03 to be exclusively used for this card is sold separately. Fix both ends of the connector connection on the card side, using screws.



Male connector (9 pin)
JAE DE-9P or equivalent

Male connector (25 pin)
JAE DB-25P or equivalent

Fig. 4-6 Signal Cable

Setting the card for delivery

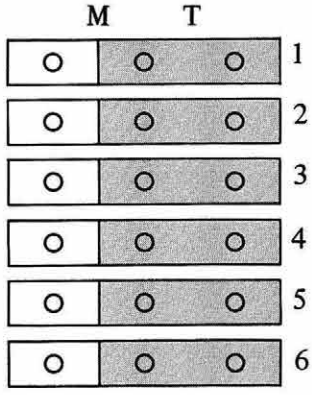
- Port address (SW-1): B0H, B1H, B2H, B3H

Switch segment	1	2	3	4	5	6
Switch position	ON	ON	OFF	OFF	ON	OFF

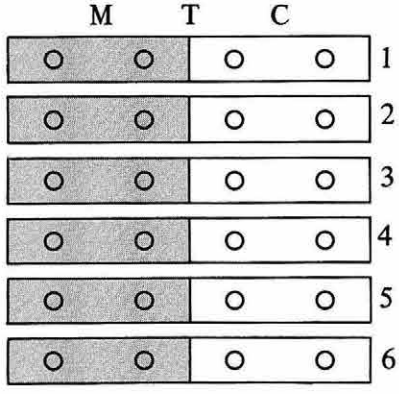
- Baud rate (SW-2, SW-3): 9600 baud

Switch segment	1	2	3	4	5	6	7	8	9	10
Switch position	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON

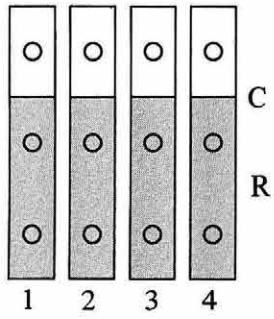
- Terminal mode for Channel A (J-3): Terminal mode



- Terminal mode for Channel B (J-1): Modem mode

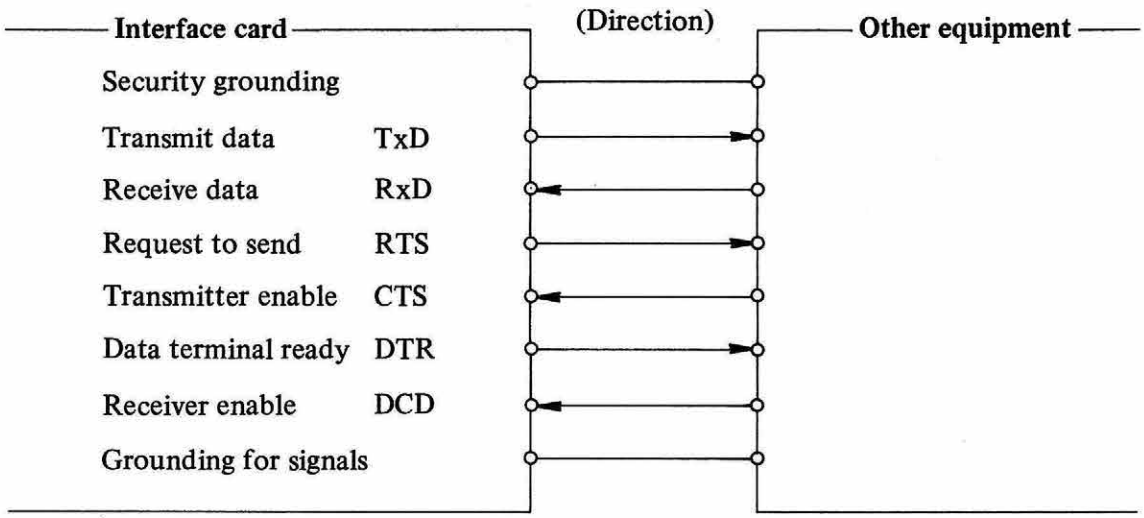


- RS-232C/current loop selection for Channel B (J-2): RS-232C

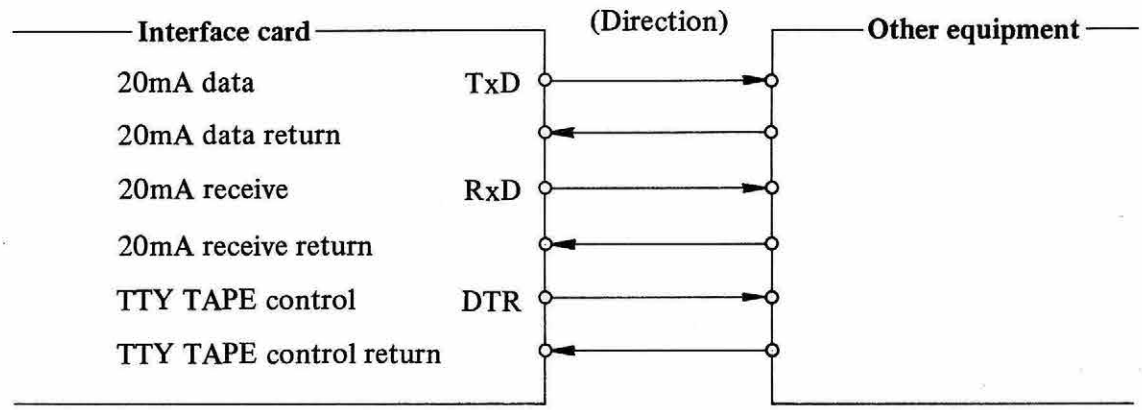


5. Electrical Characteristics of Signals

Direction of signals for RS-232C

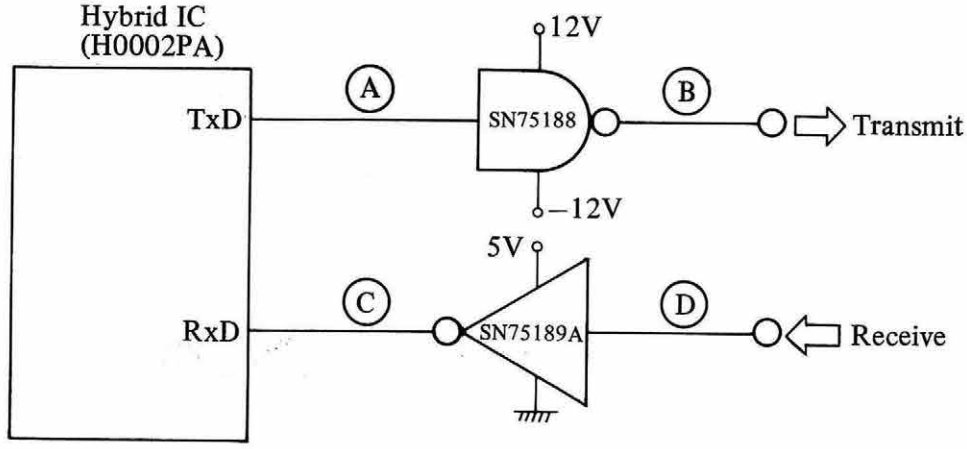


Direction of signals for current loop



Signal level with RS-232C interface

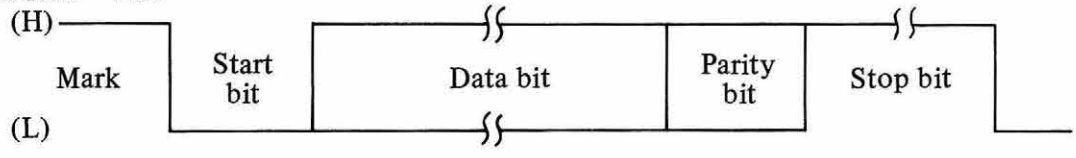
The signal levels (voltage levels) and polarities of transmit data (TxD) and receive data (RxD) with RS-232C interface are as follows.



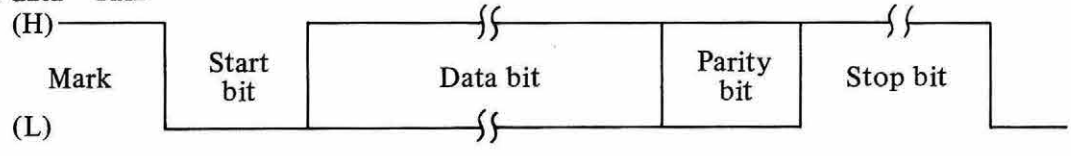
Transmit/Receive Line

The signals on the lines (A) and (C) in Fig. 5-3 are on TTL level.

- **Transmit data TxD**

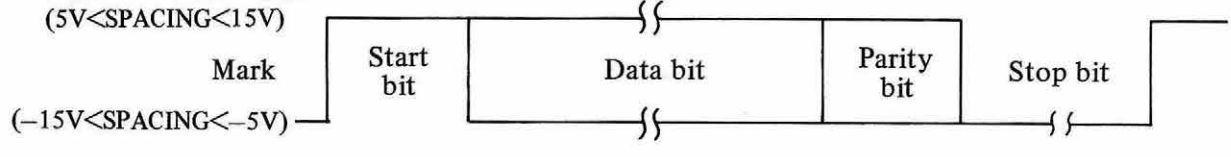


- **Receive data RxD**

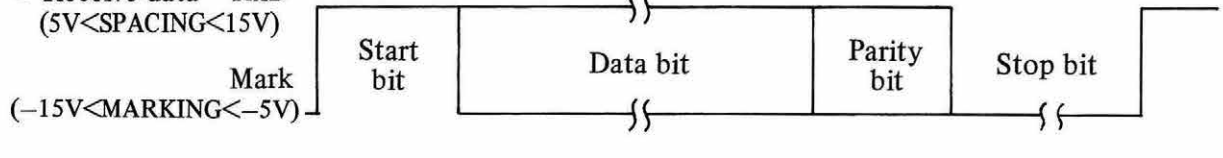


The signals on the lines (B) and (D) in Fig. 5-3 are on RS-232C level.

- **Terminal Data TxD**
  (5V<SPACING<15V)

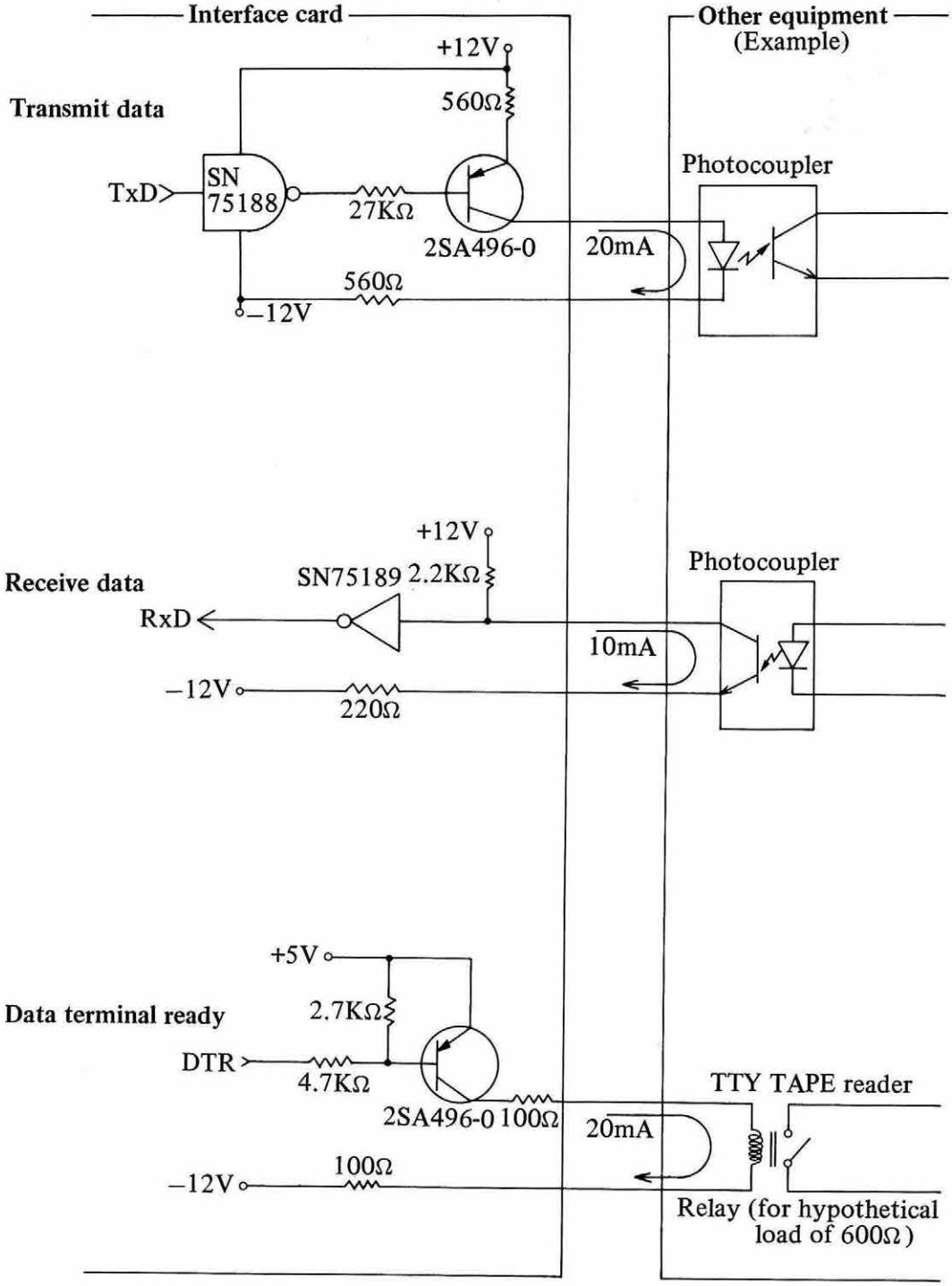


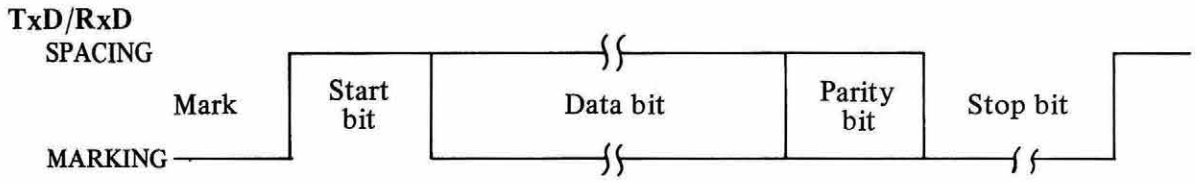
- **Receive data RxD**
  (5V<SPACING<15V)



Current loop

Current loop permits transmission/receiving of signals on logic levels 1 and 0, depending on the presence/absence of current. The direction of current flow is illustrated below.





	Current
MARKING	20mA
SPACING	0mA

Note: Since current value varies depending on load, typical values are shown here.

6. Programming

The interface card transmits data by the asynchronous transmission method. In the asynchronous data communication, a datum is composed of 4 parts: start bit, character length, parity bit, and stop bit, as shown in Fig. 6-1.

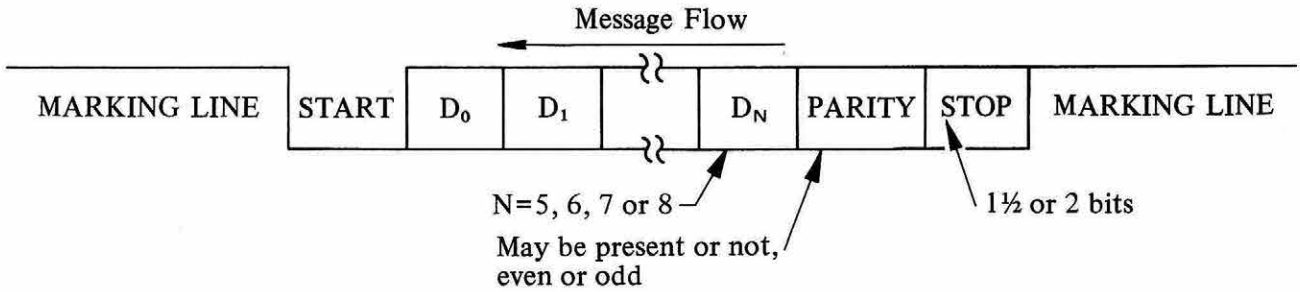


Fig. 6-1 Asynchronous Message Format

The hybrid IC (H0002PA) converts 8-bit parallel data into the message format as shown in Fig. 6-1 to transmit the data according to the predetermined baud rate, and, conversely, receives data, while checking the contents of errors, and convert them into 8-bit parallel data.

The hybrid IC consists of Z80A-SIO/0 logic, interrupt vector logic, bus interface logic, etc., and can be directly connected with Z80A-CPU bus. Further, the direct connection eliminates the necessity of a logic to return interrupt vector in the interrupt mode 2 of Z80-CPU. Fig. 6-2 shows the block diagram of the hybrid IC.

Therefore programming of the hybrid IC can be conceived in the same way as in the case of Z80A-SIO programming. The hybrid IC has two channels (Channel A and Channel B), each of which has a read register and a write register for control of the channel. Table 6-1 shows the relationship between port addresses and transmit/receive data/register, and Table 6-2 shows the registers of each channel.

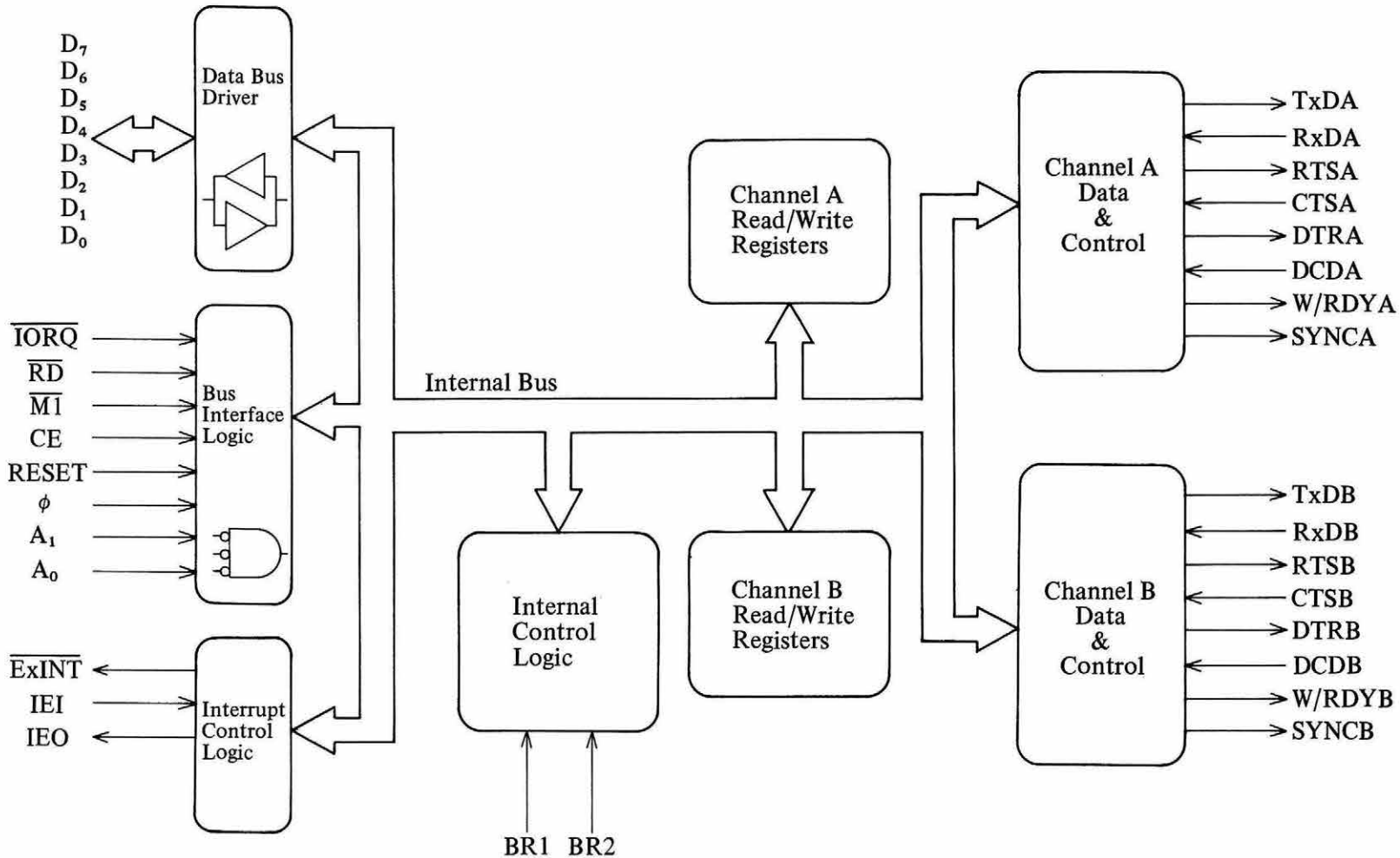


Fig. 6-2 Block Diagram of Hybrid IC

Table 6-1.

Z80-CPU command	Address		Function
	A_1	A_0	
IN	L	L	Channel A receive data read
IN	L	H	Channel A register read
IN	H	L	Channel B receive data read
IN	H	H	Channel B register read
OUT	L	L	Channel A transmit data write
OUT	L	H	Channel A register write
OUT	H	L	Channel B transmit data write
OUT	H	H	Channel B register write

Table 6-2. Read/Write Registers

Channel A	Write register	WR0	WR1	–	WR3	WR4	WR5	WR6	WR7
	Read register	RR0	RR1						
Channel B	Write register	WR0	WR1	WR2	WR3	WR4	WR5	WR6	WR7
	Read register	RR0	RR1	RR2					

In order to carry out transmission/receiving of data via this interface card, the following three programs (mode setting, input, and output) are required basically. For details, refer to the description of the registers which will be given later.

Mode setting program

Particulars of asynchronous data communication must be set in the first place. Mode setting includes the following.

- Magnification of clock rate (Set at x 16)
- Number of stop bits (1, 1½, or 2)
- Presence/absence of parity bit
- Designation of odd or even parity, if any
- Transmit/receive character length (5, 6, 7, or 8 bits)

Input program

Program to read the data received by the hybrid IC.

Error check: Checks D_6, D_5, and D_4 of read register PR1.
Data read: Checks D_0 of read register RR0, and reads data if it is 1.

Output program

Program to write data to be transmitted to the hybrid IC. This program checks D_9 of the register RR0 and writes data it is 1. After that, start bit, parity bit, and stop bit are automatically added for transmission.

Example of control program

A list by assembler language is shown as an example of control program using this interface card.

Set port address

Channel A/data:	B0H
Channel A/control:	B1H
Channel B/data:	B2H
Channel B/control:	B3H

Mode setting parameters

Clock rate:	x 16
Stop bit:	2 bits
Parity:	Present
Odd/even of parity:	Even
Transmit/receive character length:	8 bits
DTR, RTS:	Set

```
   **  Z80 ASSEMBLER SB-7201  <SIF-ROUTINE>  PAGE 01

01 0000                    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
02 0000                    ;
03 0000                    ; Serial I/F Subroutine for MZ-8BI03 on MZ-80B
04 0000                    ;
05 0000                    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
06 0000                    ;
07 F000                            ORG   F000H
08 F000                    ;
09 F000                    ;***   Port Address Equation   ***
10 F000                    ;
11 F000  P                 CHADT:  EQU   B0H              ; channel A data port
12 F000  P                 CHACT:  EQU   B1H              ; channel A control port
13 F000  P                 CHBDT:  EQU   B2H              ; channel B data port
14 F000  P                 CHBCT:  EQU   B3H              ; channel B control port
15 F000                    ;
16 F000                    ;***  Parameter  ***
17 F000                    ;
18 F000                    WR:     ENT
19 F000 18                         DEFB  18H              ; channel reset
20 F001 10                         DEFB  10H              ; EXT/STATUS reset
21 F002 10                         DEFB  10H              ; EXT/STATUS reset
22 F003 04                 WR4:    DEFB  4                ; register NO.
23 F004 4F                         DEFB  4FH              ; x16,2stop,EV,PE
24 F005 05                 WR5:    DEFB  5                ; register NO.
25 F006 EA                         DEFB  EAH              ; DTR,TxDT=8,TxEN,RTS
26 F007 03                 WR3:    DEFB  3                ; register NO.
27 F008 C0                         DEFB  C0H              ; RxDT=8
28 F009                    ;
29 F009                    MODE:   ENT                    ; mode set
30 F009 0EB1                       LD    C,CHACT          ; CH.A port adr. load
31 F00B 0609                       LD    B,9
32 F00D 2100F0                     LD    HL,WR
33 F010 EDB3                       OTIR                   ; parameter transfer
34 F012 0EB3                       LD    C,CHBCT          ; CH.B port adr. load
35 F014 0609                       LD    B,9
36 F016 2100F0                     LD    HL,WR
37 F019 EDB3                       OTIR                   ; parameter transfer
38 F01B 3E03                       LD    A,3              ; CH.A RxEN
39 F01D D3B1                       OUT   (CHACT),A        ;
40 F01F 3A08F0                     LD    A,(WR3+1)
41 F022 F601                       OR    1
42 F024 D3B1                       OUT   (CHACT),A
43 F026 3E03                       LD    A,3              ; CH.B RxEN
44 F028 D3B3                       OUT   (CHBCT),A
45 F02A 3A08F0                     LD    A,(WR3+1)
46 F02D F601                       OR    1
47 F02F D3B3                       OUT   (CHBCT),A
48 F031 C9                         RET
49 F032                            SKP   H
```

```
   **  Z80 ASSEMBLER SB-7201  <SIF-ROUTINE>  PAGE 02

01 F032                  ;
02 F032                  ;     INPUT ROUTINE
03 F032                  ;       CF=0 : NO DATA
04 F032                  ;       CF=1 : ACC DATA
05 F032                  ;      (INER@)=0 : NO ERROR      (ZF=1)
06 F032                  ;        bit4=1 : PARITY ERROR   (ZF=0)
07 F032                  ;        bit5=1 : OVERRUN ERROR  (ZF=0)
08 F032                  ;        bit6=1 : FRAMING ERROR  (ZF=0)
09 F032                  ;
10 F032                  INER@:  ENT                     ; error code buffer
11 F032                          DEFS   1
12 F033                  ;
13 F033                  CHAIN:  ENT                     ; channel A input
14 F033 DBB1                     IN     A,(CHACT)        ; receive check
15 F035 0F                       RRCA
16 F036 D0                       RET    NC
17 F037 3E01                     LD     A,1
18 F039 D3B1                     OUT    (CHACT),A
19 F03B DBB1                     IN     A,(CHACT)        ; error code input
20 F03D E670                     AND    70H
21 F03F 3232F0                   LD     (INER@),A        ; error code store
22 F042 DBB0                     IN     A,(CHADT)        ; character input
23 F044 37                       SCF
24 F045 C9                       RET
25 F046                  ;
26 F046                  CHBIN:  ENT                     ; channel B input
27 F046 DBB3                     IN     A,(CHBCT)        ; receive check
28 F048 0F                       RRCA
29 F049 D0                       RET    NC
30 F04A 3E01                     LD     A,1
31 F04C D3B3                     OUT    (CHBCT),A
32 F04E DBB3                     IN     A,(CHBCT)        ; error code input
33 F050 E670                     AND    70H
34 F052 3232F0                   LD     (INER@),A        ; error code store
35 F055 DBB2                     IN     A,(CHBDT)        ; character input
36 F057 37                       SCF
37 F058 C9                       RET
38 F059                  ;
39 F059                  ;   OUTPUT ROUTINE
40 F059                  ;      (ACC) = DATA
41 F059                  ;
42 F059                  CHAOUT: ENT                     ; channel A outoput
43 F059 F5                       PUSH   AF
44 F05A DBB1             AOUT1:  IN     A,(CHACT)        ; buffer empty check
45 F05C CB57                     BIT    2,A
46 F05E 28FA                     JR     Z,AOUT1
47 F060 F1                       POP    AF
48 F061 D3B0                     OUT    (CHADT),A        ; character output
49 F063 C9                       RET
50 F064                  ;
51 F064                  CHBOUT: ENT                     ; channel B output
52 F064 F5                       PUSH   AF
53 F065 DBB3             BOUT1:  IN     A,(CHBCT)        ; buffer empty check
54 F067 CB57                     BIT    2,A
55 F069 28FA                     JR     Z,BOUT1
56 F06B F1                       POP    AF
57 F06C D3B2                     OUT    (CHBDT),A        ; character output
58 F06E C9                       RET
59 F06F                          END
```

Interrupts

The following paragraphs describe the use of interrupt between Z80-CPU and the interface card.

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.

Transmit interrupts, Receive interrupts and External/Status interrupts are the main sources of interrupts (Figure 6-3). 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 of 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 $\overline{\text{CTS}}$, $\overline{\text{DCD}}$ and $\overline{\text{SYNC}}$ 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.

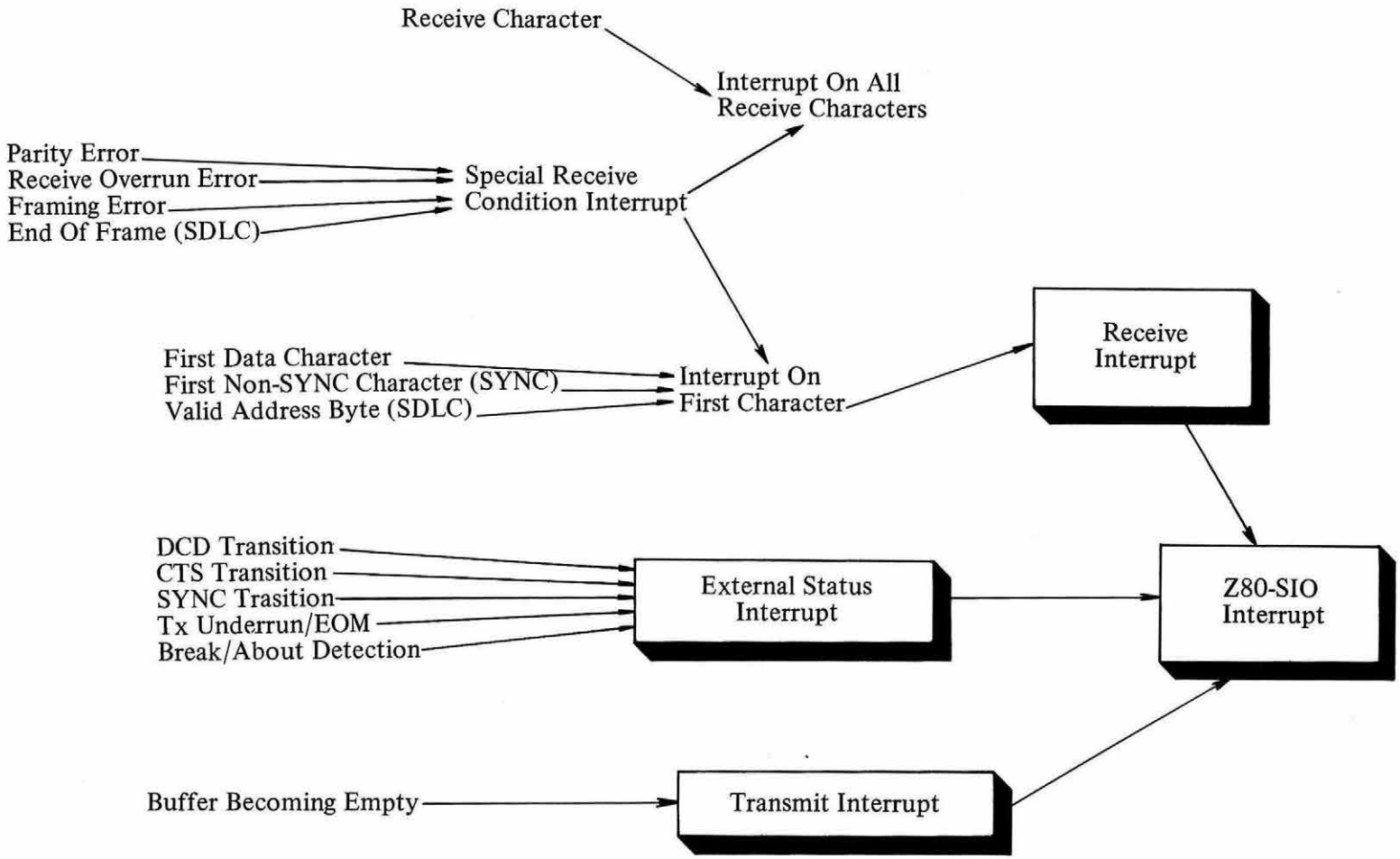


Fig. 6-3 Interrupt Structure

7. Z80-SIO Registers

As mentioned earlier, Z80-SIO incorporated in the hybrid IC has a write register and a read register for each channel.

The bit assignment and functional grouping of each register is configured to simplify and organize the programming process. Table 7-1 illustrates the functions assigned to each read or write register.

Table 7-1. Functional Assignments of Read and Write Registers

Register	Function
WR0	Resister 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 of SDLC address field
WR7	Sync character or SDLC flag

(a) Write Register

Register	Function
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

Sinch this interface card is designed exclusively for asynchrounous data communication, it cannot be applied to synchrounous communication.

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 reigster access. This maintains maximum flexibility and system control. Each channel contains the following control registers. These registers are addressed as commands (not data).

(1) 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	PRT 0
		Control Word			Register Pointer		

① 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.

② 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.

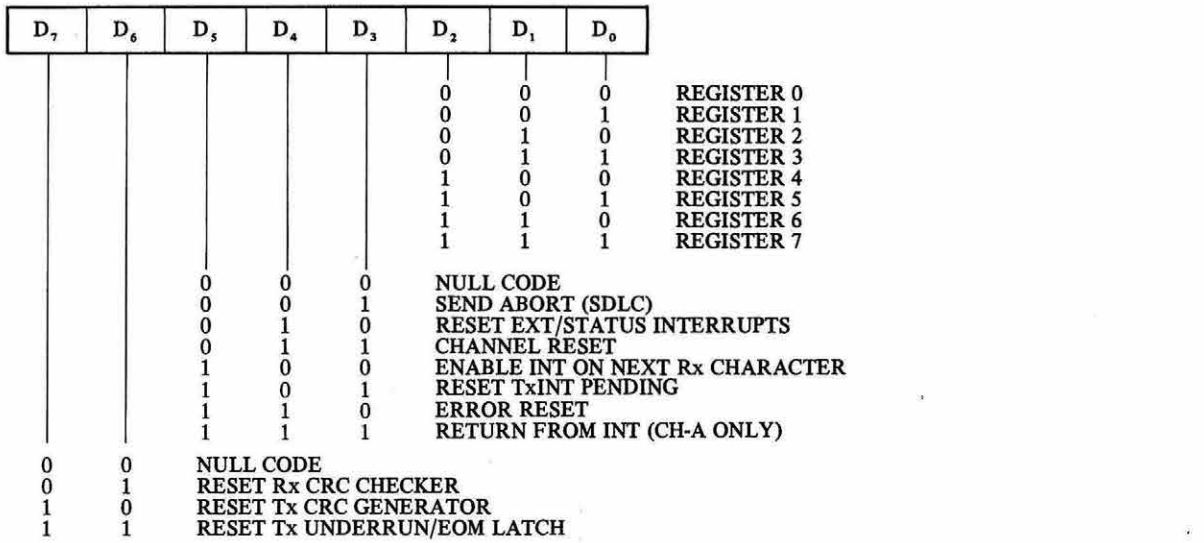
Command 2 (Reset External/Status Interrupts)

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.

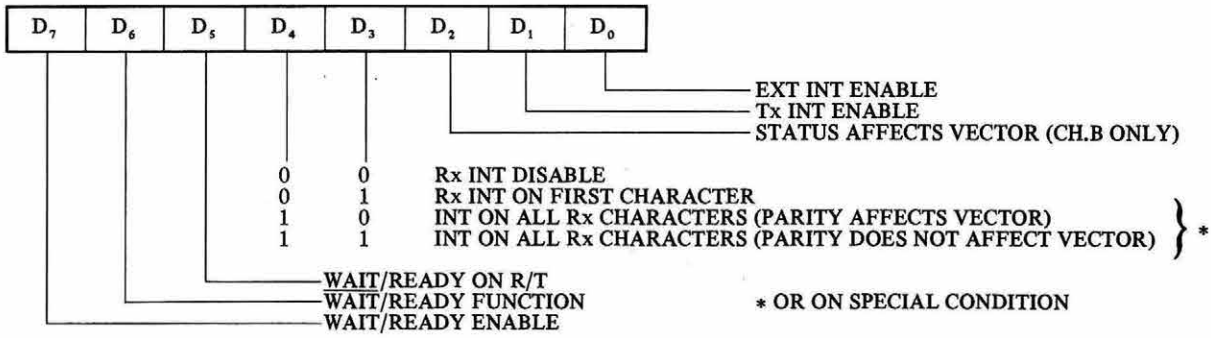
Command 3 (Channel Reset)

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.

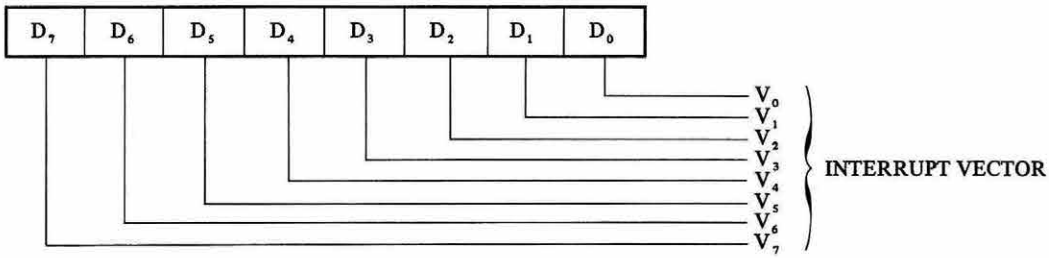
WRITE REGISTER 0



WRITE REGISTER 1



WRITE REGISTER 2 (CHANNEL B ONLY)



WRITE REGISTER 3

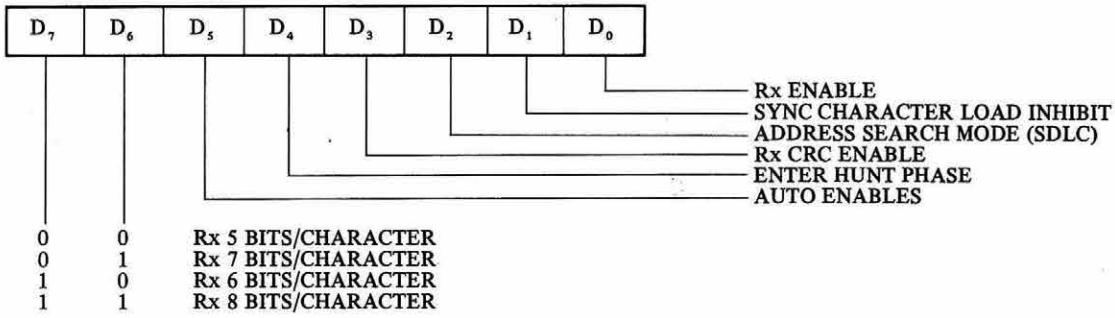
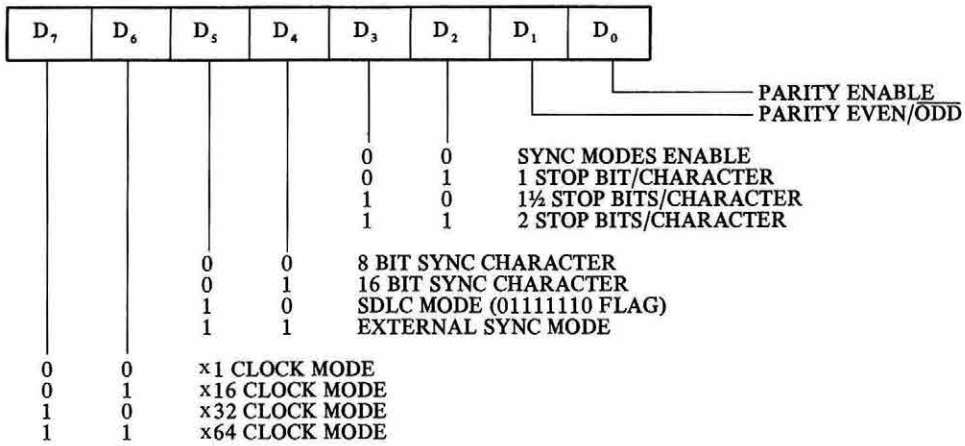
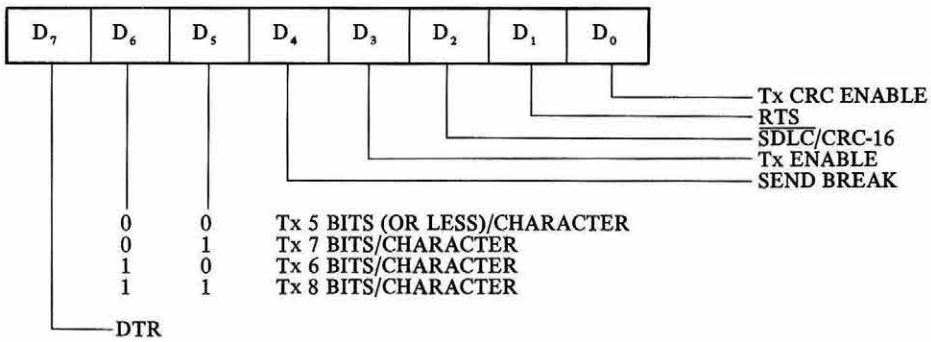


Figure 7-2. Write Register Bit Functions (1)

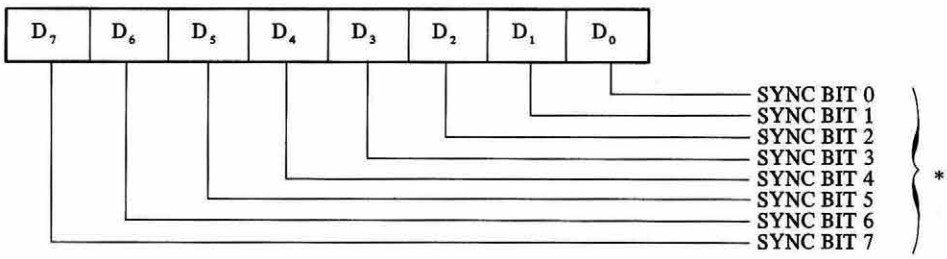
WRITE REGISTER 4



WRITE REGISTER 5

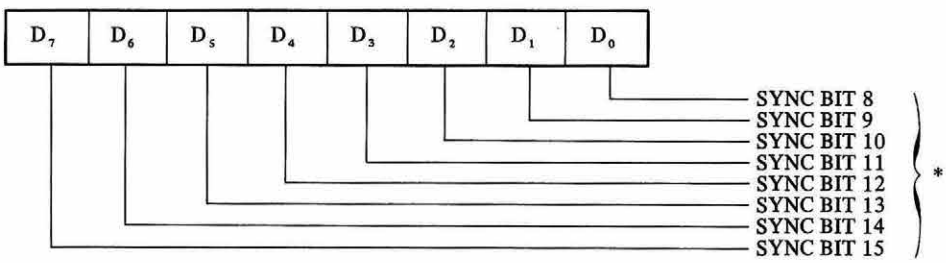


WRITE REGISTER 6



* ALSO SDLC ADDRESS FIELD

WRITE REGISTER 7



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

Fig. 7-2 Write Register Bit Functions (II)

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 these are no more characters to be sent (at the end of message, for example), issueing 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.

③ CRC Reset Codes 0 and 1 (D_6 and D_7)

Together, these bits select one of the three following reset command:

Code 1 (D_7)	Code 0 (D_6)	Reset Command
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.

(2) 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

① 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.

② Transmitter Interrupt Enable (D_1)

If enabled, interrupts occur whenever the transmitter buffer becomes empty.

③ 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:

Channel	V_3	V_2	V_1	Interrupt Conditions
B	0	0	0	Transmit Buffer Empty
	0	0	1	External/Satus Change
	0	1	0	Receive Character Available
	0	1	1	Special Receive Condition *
A	1	0	0	Transmit Buffer Empty
	1	0	1	External/Status Change
	1	1	0	Receive Character Available
	1	1	1	Special Receive Condition *
*Special Receive Conditions: Parity Error, Rx Overrun Error, Framing Error, End Of Frame (SDLC).				

④ 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.

Mode	D_4 Receive Interrupt Mode 1	D_3 Receive Interrupt Mode 0	Description
0	0	0	Receive Interrupts Disabled
1	0	1	Receive Interrupt On First Character Only
2	1	0	Interrupt On All Receive Characters (Parity error is a Special Receive condition)
3	1	1	Interrupt On All Receive Characters (Parity error is not a Special Receive condition)

⑤ 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 $\overline{\text{WAIT/READY}}$ 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 $\overline{\text{WAIT/READY}}$ output when active or inactive depend on the combination of modes selected. Following is a summary of these combinations:

D_7	D_5	D_6	Selected Functions	Logic States of $\overline{\text{WAIT/READY}}$ Output
0	—	0	$\overline{\text{WAIT}}$	Floating (Low when it is active)
		1	$\overline{\text{READY}}$	High
1	0 (Transmit)	0	$\overline{\text{WAIT}}$	Low when transmit buffer is full and an SIO data port is selected. Floating when transmit buffer is empty.
		1	$\overline{\text{READY}}$	High when transmit buffer is full. Low when transmit buffer is empty.
	1 (Receive)	0	$\overline{\text{WAIT}}$	Floating when receive buffer is full. Low when receive buffer is empty and an SIO data port is selected.
		1	$\overline{\text{READY}}$	Low when receive buffer is full. High when receive buffer is empty.

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_DHϕ(WR) from the falling edge of ϕ. The $\overline{\text{READY}}$ output High-to-Low transition occurs with the delay t_DLϕ(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/$\overline{\text{A}}$ and C/$\overline{\text{D}}$ inputs to the Z80-SIO to transfer data. The $\overline{\text{READY}}$ output becomes inactive as soon as $\overline{\text{IORQ}}$ and $\overline{\text{CS}}$ becomes 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.

(3) 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

(4) WRITE REGISTER 3

WR3 contains receiver logic control bits and parameters.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Receiver Bits/ Char 1	Receiver Bits/ Char 0	Auto Enables	Enter Hunt Phase	Receiver CRC Enable	Address Search Mode	Sync Char Load Inhibit	Receiver Enable

① 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.

② 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.

③ 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.

④ **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.

⑤ **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.

⑥ **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.

⑦ **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

(5) WRITE REGISTER 4

WR4 contains the control bits that affect both the receiver and transmitter. In the transmit and receive intialization routine, these bits should be set before issueing 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

① **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.

② **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).

③ **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	Stop Bit
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

④ **Sync Mode 0 and 1 (D_4 and D_5)**

These bits select the various options for character synchronization.

D_5 Sync Mode 1	D_4 Sync Mode 0	Sync Mode
0	0	8-bit programmed sync
0	1	16-bit programmed sync
1	0	SDLC mode (01111110 flag pattern)
1	1	External Sync mode

⑤ **Clock Rate 0 and 1 (D_6 and D_7)**

These bits specify the multiplier between the clock ($\overline{TxC}$ and $\overline{RxC}$) and data rates. For synchronous modes, the x1 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 x1 clock rate is selected, bit synchronization must be accomplished externally.

D_7 Clock Rate 1	D_6 Clock Rate 0	Clock Rate
0	0	Data Rate x 1 = Clock Rate
0	1	Data Rate x 16 = Clock Rate
1	0	Data Rate x 32 = Clock Rate
1	1	Data Rate x 64 = Clock Rate

(6) 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

① 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.

② Request To Sent (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.

③ 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 polynomial 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).

④ 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 of flag characters are sent instead of CRC.

⑤ **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.

⑥ **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	Transmit Bits/Character
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

D: Data bit

⑦ **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).

(7) 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

(8) 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

Sync Mode	WR6	WR7
Monosync	Transmit sync character	Receive sync character
Bisync	First 8 bits of a 16-bit sync character	Last 8 bits of a 16-bit sync character
External Sync	Transmit sync character	Not used
SDLC	SDLC check address	SDLC flag (0111 1110)

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).

(1) 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

① 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.

② 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.

③ 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.

④ 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, CTS, Sync/Hunt, Break/Abort of 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.

⑤ 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 $\overline{SYNC}$ input. Any High-to-Low transition on the $\overline{SYNC}$ 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 $\overline{SYNC}$ 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 $\overline{SYNC}$ 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 $\overline{SYNC}$ input must be held High by the external logic until external character synchronization is achieved. A High at the $\overline{SYNC}$ input holds the Sync/Hunt status bit in the reset condition.

When external synchronization is achieved, $\overline{SYNC}$ must be driven Low on the second rising edge of $\overline{RxC}$ after that rising edge of $\overline{RxC}$ 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 $\overline{SYNC}$ input. Once $\overline{SYNC}$ 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 $\overline{SYNC}$ input sets the Sync/Hunt bit, which – in turn – sets the External/Status interrupt. The CPU must clear the interrupt by issueing the Reset External/Status Interrupt command.

When the $\overline{SYNC}$ 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 Hihg-to-Low transition on the $\overline{SYNC}$ 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 $\overline{SYNC}$ to become active.

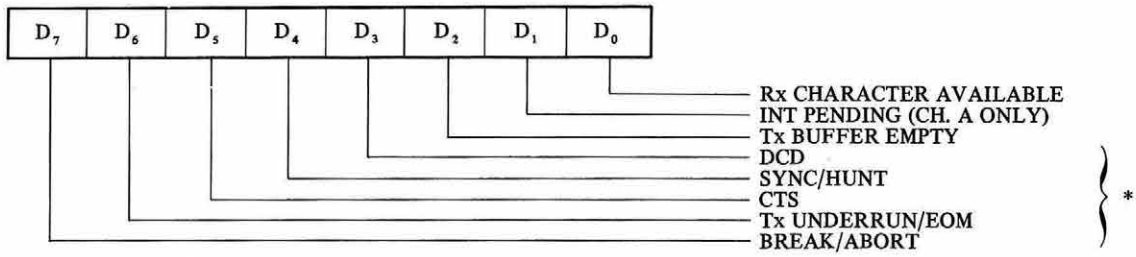
In the Monosync and Bisync Receive modes, the Sync/Hunt status bit is initially set to 1 by the Enter Hunt Mode bit. The Sync/Hunt bit is reset when the Z80-SIO establishes character synchronization. The High-to-Low transition of the Sync/Hunt bit causes an External/Status interrupt that must be cleared by the CPU issueing 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 $\overline{SYNC}$ 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.

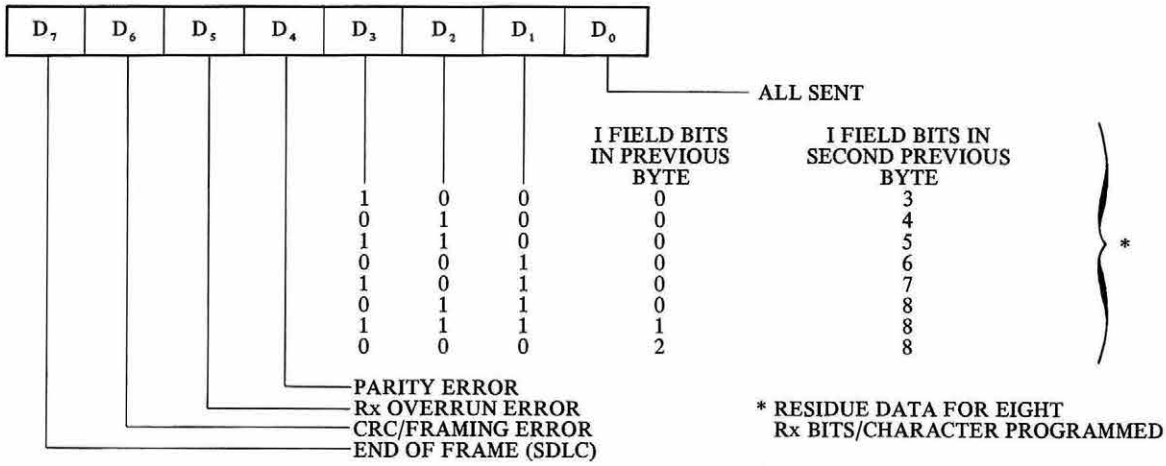
Unlike the Monosync and Bisync modes, once the Sync/Hunt bit is reset in the SDLC mode, it does not need to be set when the end of message is detected. The Z80-SIO automatically maintains synchronization. The only way the Sync/Hunt bit can be set again is by the Enter Hunt Mode bit, or by disabling the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

READ REGISTER 0



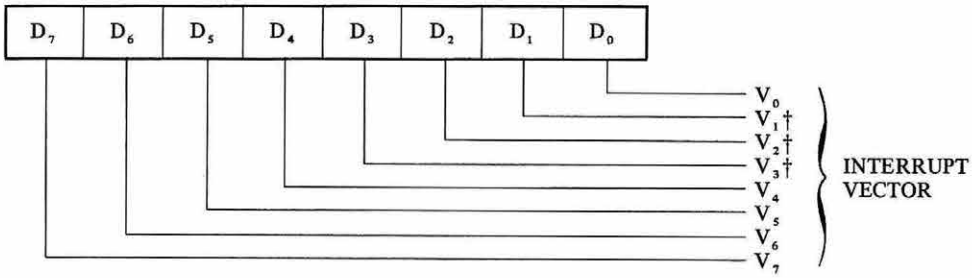
* USED WITH "EXTERNAL/STATUS INTERRUPT" MODE

READ REGISTER 1†



† USED WITH SPECIAL RECEIVE CONDITION MODE

READ REGISTER 2



† VARIABLE IF "STATUS AFFECT VECTOR" IS PROGRAMMED

Figure 7-3. Read Register Bit Functions

⑥ **Clear To Send (D_5)**

This bit is similar to the DCD bit, except that it shows the inverted state of the CTS pin.

⑦ **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 issueing 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.

⑧ **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.

(2) 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

① 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.

② 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	D_3 Residue Code 2	D_2 Residue Code 1	D_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

③ 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.

④ 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.

⑤ 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 issueing 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.

⑥ 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 issueing the Error Reset command. It is also updated by the first character of the following frame.

(3) 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

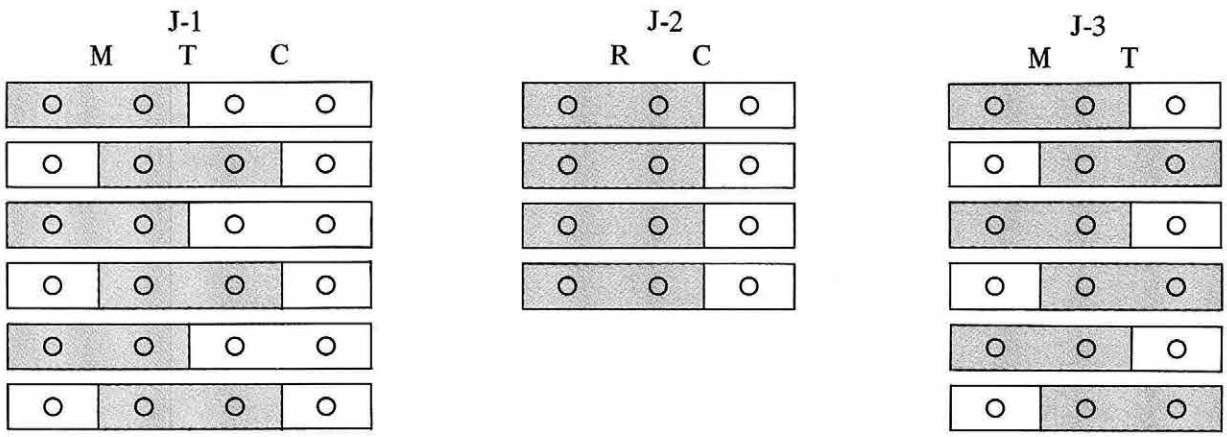
8. Sample Programs

Self-diagnosis program

Let us consider a program for self-diagnosis of this interface card and assume that data are transmitted from Channel A are received by Channel A. For channel B, similar communication system shall be adopted. Self-diagnosis of the interface card can be made by examining whether the receive data and transmit data are the same at the time of transmission/receiving.

Setting jumper blocks on the card

For the self-diagnosis as mentioned above, set the jumper blocks on the card as follows.



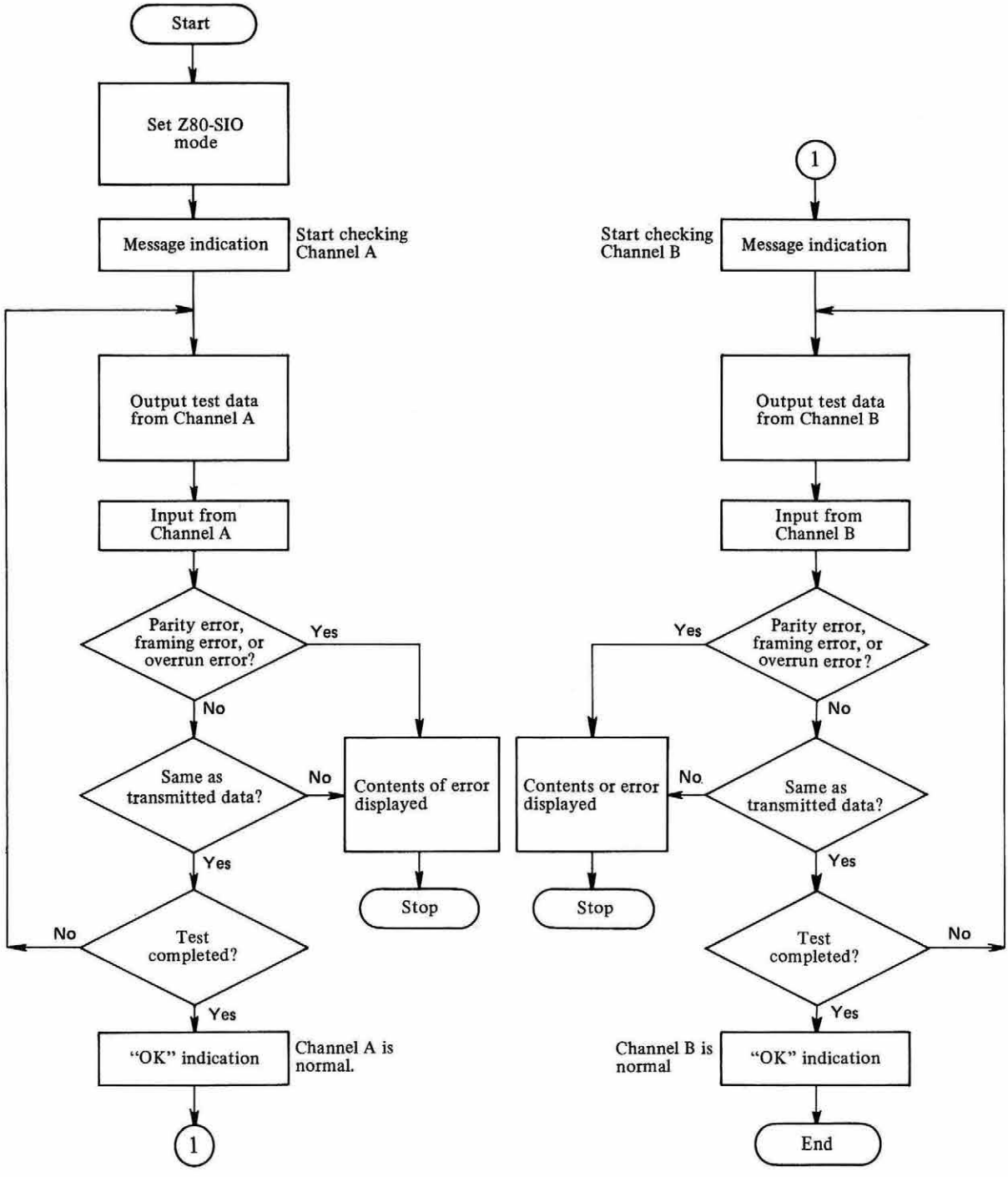
Since port addresses B0H, B1H, B2H, and B3H are used in the program, set the switch SW-1 in the following (factory setting).

Switch segment	6	5	4	3	2	1
Switch position	OFF	ON	OFF	OFF	ON	ON

Set the switch for baud rate setting as desired. After setting each mode, install the card in MZ-80B. On this occasion, signal cable needs not to be connected.

Flow Chart of Self-diagnosis Program

Start testing Channel A. Stop the program when an error occurs. If Channel A is found normal, test Channel B then.



Program by BASIC/PASCAL language

Now, let us draw up the program described in the above-mentioned flow chart, using †BASIC and PASCAL languages. The program list is shown later.

In the BASIC program, the routine to control the interface card is composed of machine language. The machine language data are written into the memory by POKE statement and the routine is called by USR statement. The routine is made by programming statement Nos. 1000 through 1990.

Statement Nos. 1080~ 1160. . . . . .Parameters

Clock rate	: x 16
Stop bit	: 2 bits
Parity	: Present
Odd/even of parity	: Even
Transmit/receive character	: 8 bits
Auto enable	: Set

Statement Nos. 1180~ 1370. . . . . .Mode setting routine

Statement Nos. 1550~ 1610. . . . . .Channel A input routine
Input data are stored in Address CHAR.
Error flags are stored in Address INER@.

Statement Nos. 1630~ 1740. . . . . .Channel B input routine
Input data are stored in Address CAHR.
Error flags are stored in Address INER@.

Statement Nos. 1780~ 1840. . . . . .Channel A output routine.
Transmit data are stored in Address CHAR.

Statement Nos. 1860~ 1930. . . . . .Channel B output routine
Transmit data are stored in Address CHAR.

Statement Nos. 1950~ 1990. . . . . .Writes machine language data in the memory

Since the concept of programming by PASCAL language is the same as that of BASIC programming, refer to the program list for actual programming.

†BASIC Version: SB-6510, SB-5510 PASCAL Version: SB-4515

by BASIC language

```
1000 REM ************************************************
1010 REM *                                              *
1020 REM *  Serial I/F Subroutine for MZ-8BI03 on MZ-80B  *
1025 REM *           ( by BASIC interpreter )           *
1030 REM *                                              *
1040 REM ************************************************
1050 REM *
1060 REM ***  Parameter  ***
1070 REM *
1080 REM WR      DEFB    18H         :DATA 18
1090 REM         DEFB    10H         :DATA 10
1100 REM         DEFB    10H         :DATA 10
1110 REM WR4     DEFB    4           :DATA 04
1120 REM         DEFB    4FH         :DATA 4F
1130 REM WR5     DEFB    5           :DATA 05
1140 REM         DEFB    EAH         :DATA EA
1150 REM WR3     DEFB    3           :DATA 03
1160 REM         DEFB    E0H         :DATA E0
1170 REM
1180 REM MODE    ENT                 ; [ adr.=$F009 ]
1190 REM         LD      C,CHACT     :DATA 0E,B1
1200 REM         LD      B,9         :DATA 06,09
1210 REM         LD      HL,WR       :DATA 21,00,F0
1220 REM         OTIR                :DATA ED,B3
1230 REM         LD      C,CHBCT     :DATA 0E,B3
1240 REM         LD      B,9         :DATA 06,09
1250 REM         LD      HL,WR       :DATA 21,00,F0
1260 REM         OTIR                :DATA ED,B3
1270 REM         LD      A,3         :DATA 3E,03
1280 REM         OUT     (CHACT),A   :DATA D3,B1
1290 REM         LD      A,(WR3+1)   :DATA 3A,08,F0
1300 REM         OR      1           :DATA F6,01
1310 REM         OUT     (CHACT),A   :DATA D3,B1
1320 REM         LD      A,3         :DATA 3E,03
1330 REM         OUT     (CHBCT),A   :DATA D3,B3
1340 REM         LD      A,(WR3+1)   :DATA 3A,08,F0
1350 REM         OR      1           :DATA F6,01
1360 REM         OUT     (CHBCT),A   :DATA D3,B3
1370 REM         RET                 :DATA C9
1380 REM
1390 REM ***  INPUT ROUTINE  ***
1400 REM     (INER@)=0    NO ERROR
1410 REM        bit4=1    PARITY ERROR
1420 REM        bit5=1    OVERRUN ERROR
1430 REM        bit6=1    FRAMING ERROR
1440 REM
1450 REM INER@   ENT                 ; [ adr.=$F032 ]
1460 REM         DEFS    1           :DATA 00
1470 REM CHAR    ENT                 ; [ adr.=$F033 ]
1480 REM         DEFS    1           :DATA 00
1490 REM
1500 REM CHAIN   ENT                 ;' [ adr.=$F034 ]
1510 REM         IN A,(CHACT)        :DATA DB,B1
1520 REM         RRCA                :DATA 0F
1530 REM         JR      NC,CHAIN    :DATA 30,FB
1540 REM         LD      A,1         :DATA 3E,01
1550 REM         OUT     (CHACT),A   :DATA D3,B1
1560 REM         IN      A,(CHACT)   :DATA DB,B1
1570 REM         AND     70H         :DATA E6,70
1580 REM         LD      (INER@),A   :DATA 32,32,F0
1590 REM         IN      A,(CHADT)   :DATA DB,B0
1600 REM         LD      (CHAR),A    :DATA 32,33,F0
1610 REM         RET                 :DATA C9
```

```
1630 REM CHBIN  ENT                ; [ adr.=$F04A ]
1640 REM        IN A,(CHBCT)       :DATA DB,B3
1650 REM        RRCA               :DATA OF
1660 REM        JR      NC,CHBIN   :DATA 30,FB
1670 REM        LD      A,1        :DATA 3E,01
1680 REM        OUT     (CHBCT),A  :DATA D3,B3
1690 REM        IN      A,(CHBCT)  :DATA DB,B3
1700 REM        AND     70H        :DATA E6,70
1710 REM        LD      (INER@),A  :DATA 32,32,F0
1720 REM        IN      A,(CHBDT)  :DATA DB,B2
1730 REM        LD      (CHAR),A   :DATA 32,33,F0
1740 REM        RET                :DATA C9
1750 REM
1760 REM ***  OUTPUT ROUTINE  ***
1770 REM
1780 REM CHAOUT ENT                ; [ adr.=$F060 ]
1790 REM        IN      A,(CHACT)  :DATA DB,B1
1800 REM        BIT     2,A        :DATA CB,57
1810 REM        JR      Z,CHAOUT   :DATA 28,FA
1820 REM        LD      A,(CHAR)   :DATA 3A,33,F0
1830 REM        OUT     (CHADT),A  :DATA D3,B0
1840 REM        RET                :DATA C9
1850 REM
1860 REM CHBOUT ENT                ; [ adr.=$F06C ]
1870 REM        IN      A,(CHBCT)  :DATA DB,B3
1880 REM        BIT     2,A        :DATA CB,57
1890 REM        JR      Z,CHBOUT   :DATA 28,FA
1900 REM        LD      A,(CHAR)   :DATA 3A,33,F0
1910 REM        OUT     (CHBDT),A  :DATA D3,B2
1920 REM        RET                :DATA C9
1930 REM        END                :DATA END
1940 REM
1950 DIM X(30):LIMIT $F000 :P=15*4096
1960 FOR J=0 TO 9:X(J)=J:NEXT:FOR J=0 TO 5:X(17+J)=J+10:NEXT
1965 PRINT"SIF SUBROUTIN LOADING"
1970 READ X$:IF X$="END" THEN 3000
1980 J=16*X(ASC(MID$(X$,1,1))-48)+X(ASC(MID$(X$,2,1))-48)
1990 POKE P,J:P=P+1:GOTO 1970
3000 REM  ************************
3010 REM *                      *
3020 REM *      MAIN PROGRAM    *
3030 REM *                      *
3040 REM ************************
3050 REM
3060 USR($F009):REM   mode set
3070 PRINT:PRINT:PRINT "*****  TEST PROGRAM  (Serial I/F MZ-8BI03)  ***** "
3080 PRINT
3090 PRINT "Channel A TEST "
3100 FOR I=0 TO 255
3110 POKE  $F033,I :USR($F060):POKE $F033,0  : REM channel-A output
3120 USR($F034):A=PEEK($F033):ER=PEEK($F032) : REM channel-A input
3130 IF ER<>0 THEN PRINT"COMMUNICATION ER = ";ER:STOP
3140 IF I<>A THEN PRINT"COMPARA ER":STOP
3150 PRINT".";:NEXT
3160 PRINT"***  OK  ***"
3170 PRINT
3180 PRINT"Channel B TEST "
3190 FOR I=0 TO 255
3200 POKE $F033,I:USR($F06C):POKE $F033,0    : REM channel-B output
3210 USR($F04A):A=PEEK($F033):ER=PEEK($F032) : REM channel-B input
3220 IF ER<>0 THEN PRINT "COMMUNICATION ER = ";ER:STOP
3230 IF I<>A THEN PRINT "COMPARA ER":STOP
3240 PRINT"#";:NEXT
3250 PRINT"***  OK  ***"
3260 END
```

by PASCAL language

```
 0.{*****************************************************}
 1.{                                                     }
 2.{    Serial I/F subroutine for MZ-8BI03 on MZ-80      }
 3.{              ( by PASCAL interpreter )              }
 4.{                                                     }
 5.{*****************************************************}
 6.var BIT4,BIT5,BIT6,BIT7:integer;
 7.    CHADT,CHACT,CHBDT,CHBCT,ERFLAG:integer;
 8.    MODEPARAMETER:array[9]of char;
 9.    OUTDATA,INPDATA:char;
10.procedure BITPAT(BYTE:integer);
11.{  bit pattern exchange  }
12.  begin
13.     BIT4:=0;BIT5:=0;BIT6:=0;BIT7:=0;
14.     if(BYTE div 128)=1 then begin BIT7:=1;BYTE:=BYTE-128 end;
15.     if(BYTE div 64)=1 then begin BIT6:=1;BYTE:=BYTE-64 end;
16.     if(BYTE div 32)=1 then begin BIT5:=1;BYTE:=BYTE-32 end;
17.     if(BYTE div 16)=1 then begin BIT4:=1;BYTE:=BYTE-16 end;
18.  end;
19.procedure PARAMETERSET;
20.{  constant & parameter setting  }
21.  begin
22.     CHADT:=11*16;{B0H,Channel A data port};
23.     CHACT:=CHADT+1;{B1H,channel A control port};
24.     CHBDT:=CHACT+1;{B2H,channel B data port};
25.     CHBCT:=CHBDT+1;{B3H,channel B control port};
26.     MODEPARAMETER[1]:=chr(1*16+8);{18H,Channel reset};
27.     MODEPARAMETER[2]:=chr(16);{10H,ext+status reset};
28.     MODEPARAMETER[3]:=chr(16);
29.     MODEPARAMETER[4]:=chr(4);{Register NO.};
30.     MODEPARAMETER[5]:=chr(4*16+15);{4FH,x16,2stop,EV,PE};
31.     MODEPARAMETER[6]:=chr(5);{Register NO.};
32.     MODEPARAMETER[7]:=chr(14*16+10);{EAH,DTR,TxDT=8,TxEN,RTS};
33.     MODEPARAMETER[8]:=chr(6);{Register NO.};
34.     MODEPARAMETER[9]:=chr(14*16);{E0H,RxDT=8}
35.  end;
36.procedure MODESET(CHXCT:integer);
37.{  mode set routine  }
38.  var N:integer;
39.      RXEN:char;
40.  begin
41.     for N:=1 to 9 do output(MODEPARAMETER[N],CHXCT);
42.     RXEN:=succ(MODEPARAMETER[9]);
43.     output(chr(3),CHXCT);
44.     output(RXEN,CHXCT)
45.  end;
46.procedure SINP(CHXCT,CHXDT:integer);
47.{   serial I/F input routine  }
48.  var STATUS:char;
49.  begin
50.     ERFLAG:=0;
51.     repeat
52.       STATUS:=input(CHXCT);
53.     until odd(ord(STATUS));
54.         output(chr(1),CHXCT);
55.         STATUS:=input(CHXCT);
56.         INPDATA:=input(CHXDT);
57.         BITPAT(ord(STATUS));
58.         if(BIT6=1)or(BIT5=1)or(BIT4=1)then ERFLAG:=1
59.  end;
```

```
60.procedure SOUT(CHXCT,CHXDT:integer;DATA:char);
61.{   serial I/F output routine   }
62.  var STATUS:char;N:integer;
63.  begin
64.     ERFLAG:=0;
65.     repeat
66.       STATUS:=input(CHXCT);
67.       N:=ord(STATUS);N:=N div 4
68.     until odd(N);
69.       output(DATA,CHXDT)
70.  end;
71.{*************************************}
72.{                                     }
73.{           test procedure            }
74.{                                     }
75.{       channel-X  test routine       }
76.{                                     }
77.{*************************************}
78.procedure TESTX(CHXCT,CHXDT:integer);
79.  var I:integer;
80.      A:char;
81.  begin
82.     for I:=0 to 255 do
83.       begin
84.          OUTDATA:=chr(I);
85.          SOUT(CHXCT,CHXDT,OUTDATA);
86.          SINP(CHXCT,CHXDT);
87.          if ERFLAG<>0 then
88.             begin
89.               if BIT6=1 then writeln("FRAMING ERROR");
90.               if BIT5=1 then writeln("OVERRUN ERROR");
91.               if BIT4=1 then writeln("PARITY ERROR")
92.             end;
93.          if OUTDATA<>INPDATA then
94.               begin
95.               write("COMPARA ER : OUTDATA=",ord(OUTDATA):4);
96.               writeln("  INPDATA=",ord(INPDATA):4);
97.               ERFLAG:=2
98.               end;
99.          if ERFLAG=0 then write(".")
100.          else begin
101.              write("C  :  CONTINUE  ");
102.              repeat readln(A)until A='C';
103.              output(chr(3*16),CHACT);{SIO ER RESET};
104.              output(chr(3*16),CHBCT);{           };
105.           end;
106.      end;
107.    if ERFLAG=0 then writeln("***  OK  ***");
108. end;
109.{*****************************}
110.{                             }
111.{        MAIN PROGRAM         }
112.{                             }
113.{*****************************}
114.begin
115.    PARAMETERSET;
116.    MODESET(CHACT);
117.    MODESET(CHBCT);
118.    writeln("Channel A TEST ");
119.    TESTX(CHACT,CHADT);
120.    writeln();
121.    writeln("Channel B TEST ");
122.    TESTX(CHBCT,CHBDT);
123.end.
124.
```

9. BASIC SB-6511

BASIC SB-6511 is an improved version of Disk BASIC which is based on Disk BASIC SB-6510 and includes control statements for the serial interface (MZ-8BI03) and GP-IB interface (MZ-8BI04).

The following is information concerning the serial interface.

Using this version of BASIC allows them to be controlled simply.

1. RSMODE

Format RSMODE *a*, R*b*, T*c*, M*d*, RX*e*

a: Channel Specification

a	Channel
A	A Channel
B	B Channel

b: Specification of the number of bits for the received character.

b	bits/character
5	5
6	6
7	7
8	8

c: Specification of the number of bits for the transmitted character.

c	bits/character
5	5
6	6
7	7
8	8

d: Parity bit specification and number of stop bits specification

d	parity	stop bits
69	odd	1
70	none	
71	even	
73	odd	1½
74	none	
75	even	
77	odd	2
78	none	
79	even	

e: Receive active/inactive specification

e	Receive
0	Inactive
1	Active

Function Setting of the various modes is performed using the above parameters.

Description

- Although it is not necessary to specify all of the parameters, specification of the receive active/inactive status must be made after specification of the various parameters.
- Specification of mode setting parameters other than those given in *a* through *e* are not allowed.
- When BASIC is initialized, all channels are set for the following modes.

b, c	8	8 bits per character
d	79	Even parity/Stop bits 2
e	0	Receive inactive

Example 10 RSMODE A, RX1 .....Enables reception in channel A.

2. RSO

Format RSO *x A$*

x: Channel specification (A or B).

A$: Specify transmission data using a string variable

Function Transmit the data specified using *A$* to channel *x*

Example

```
10 X$ = "Demonstration"
20 RSO B X$ .....Transmit the data in X$ to channel B.
```

3. RSI

Format RSI *x A$*

x: Channel specification (A or B)

A$: A string variable which contains the received data

Function Data is received from channel *x* and stored in the string variable *A$*.

Example

```
10 RSMODE A, RX1 .....Activates channel A for reception.
20 RSI A B$ .....Receives data from channel A.
30 PRINT B$
```

Error Number

Error 29:	Framing error
Error 30:	Overrun error
Error 31:	Parity error
Error 32:	Data transmission is impossible. (The transmit buffer is not empty.)
Error 33:	Buffer overflow

10. Circuit Diagram

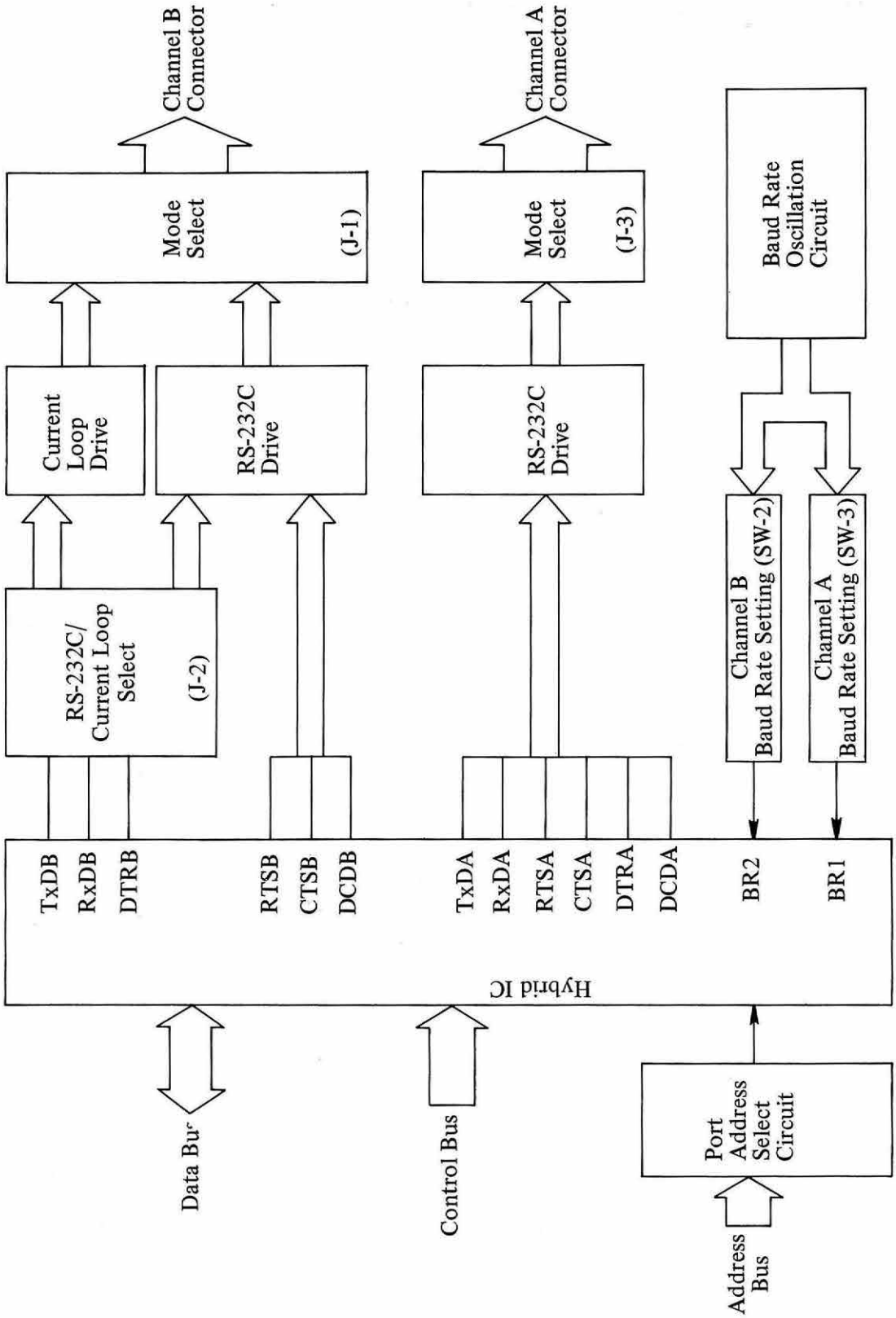


Fig. 10-1 Block Diagram of the Interface Card

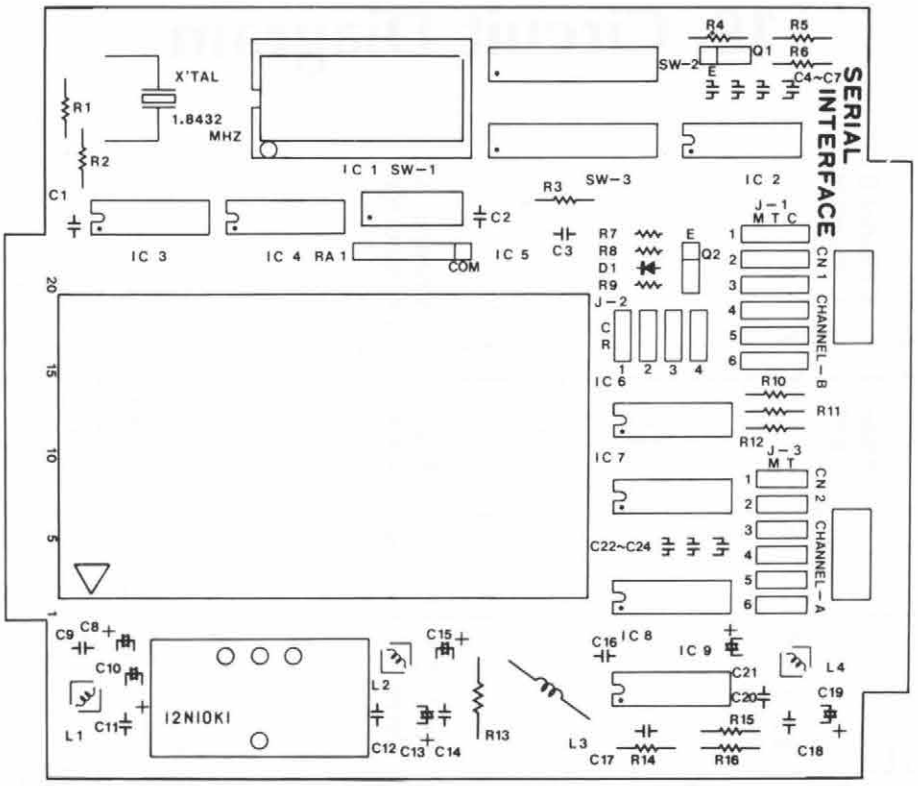
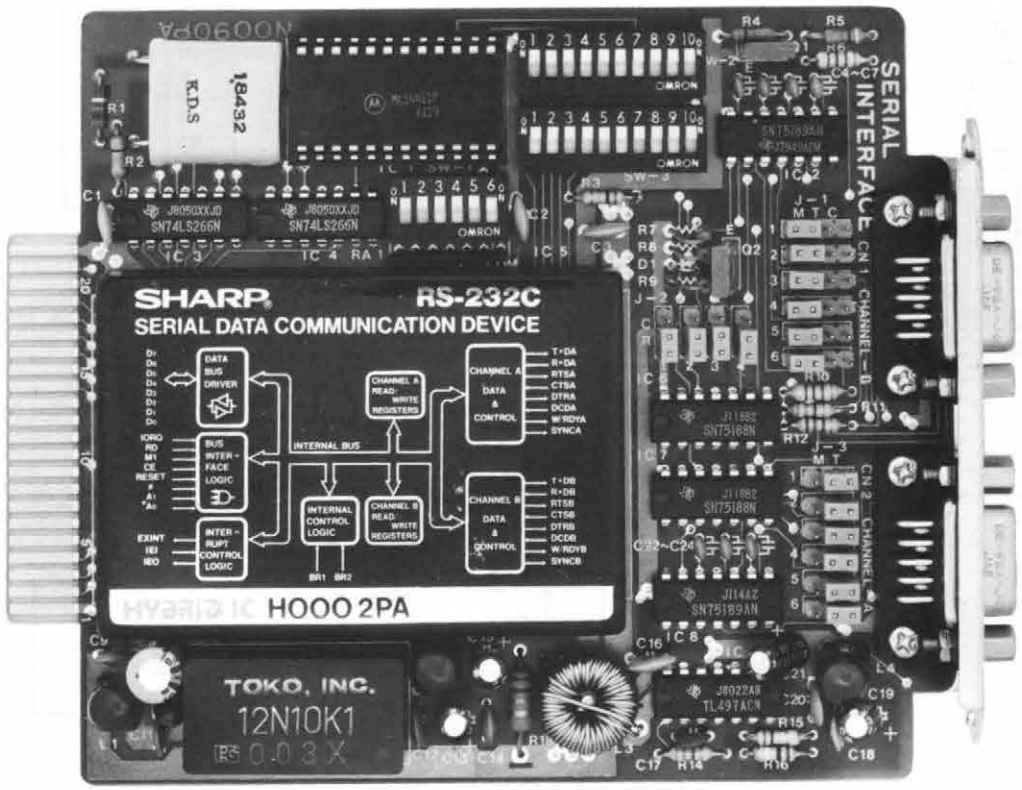
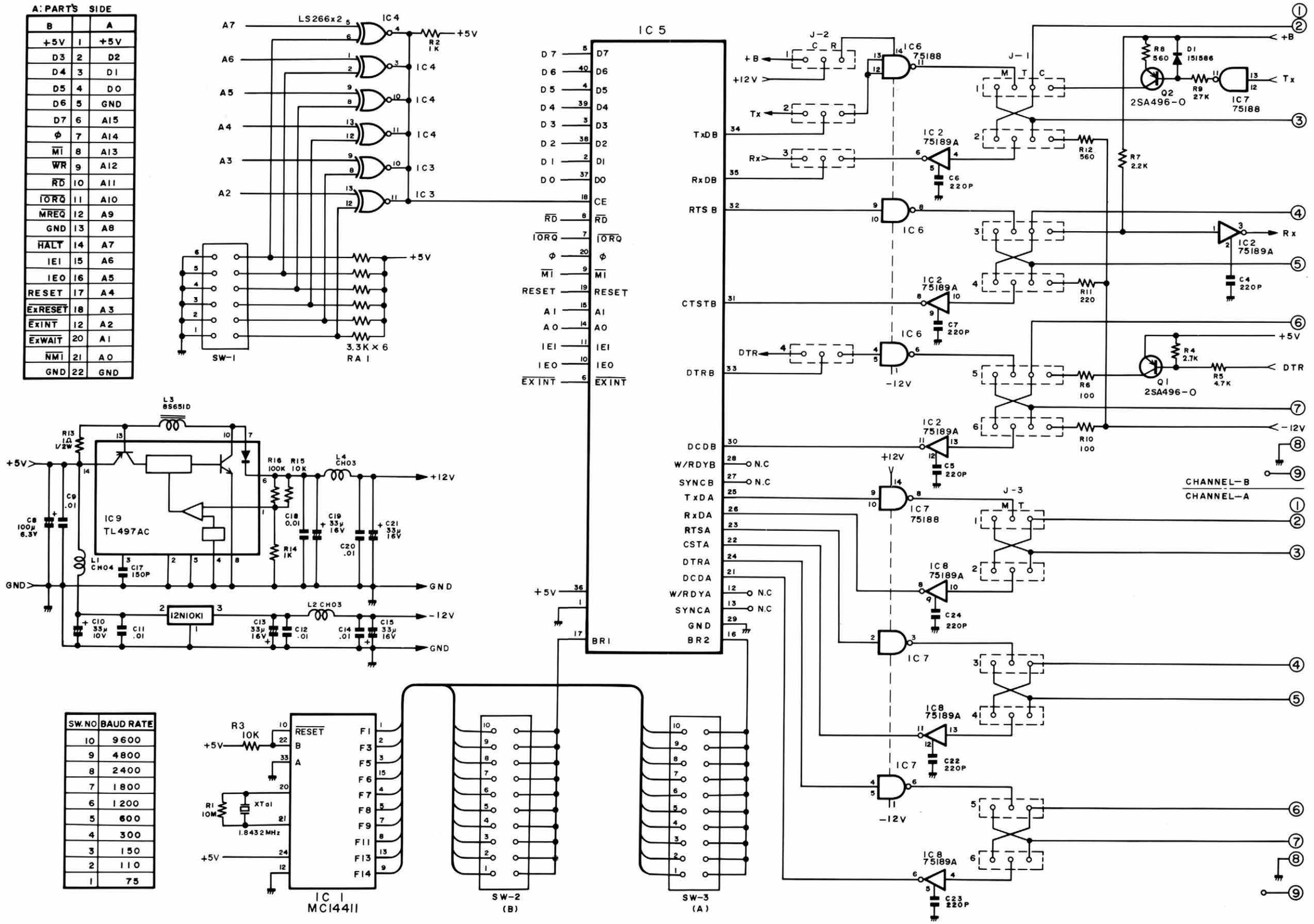


Fig. 10-2. Components Arrangement





This wiring diagram may be changed in the future for improvement of the product without prior notice.

Fig. 10-3. Wiring Diagram

SHARP CORPORATION

TINSE0036PAZZ
2I 301-K
Printed in Japan