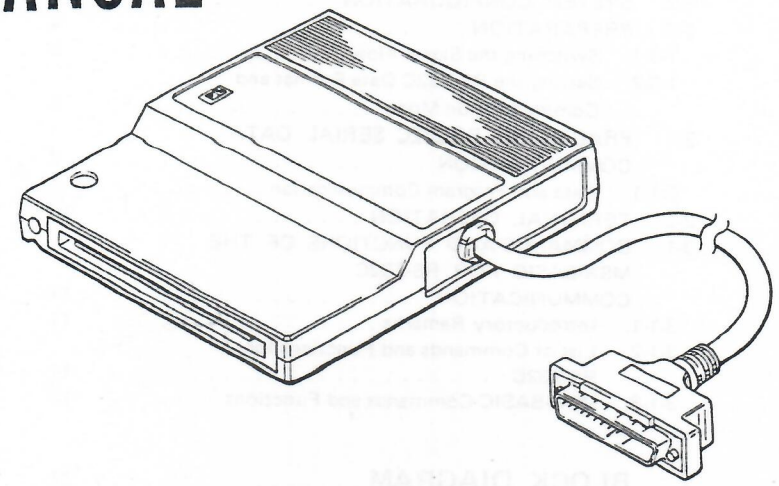
HBI-232

SERVICE MANUAL



SPECIFICATIONS

Communication mode	
Baud rate	Selectable by software 50, 75, 110, 300, 600, 1200, 1800, 2000, 2400, 3600, 4800, 7200, 9600, 19200 bps (bits per second)
Signal description	See next page.
Data format	Selectable by software Data length: 5, 6, 7, or 8 bits Stop bit(s): 1, 1½, or 2 bit(s) Parity: even, odd, ignore, or no parity
Sync mode	Asynchronous mode
Transmit/receive mode	Full duplex mode
General	
Power requirement and power consumption	+12 V, 20 mA −12 V, 20 mA +5 V, 230 mA
Operating temperature and humidity	5 to 35°C (41 to 95°F), 25 to 80%
Dimensions	Approx. 109 × 132 × 26.4 mm (w/h/d) ($4^3/_8$ × $5^1/_4$ × $1^1/_{16}$ inches)
Weight	Approx. 270 g (9.5 oz)

RS-232C INTERFACE CARTRIGE

SONY®

TABLE OF CONTENTS

CHAPTER 1
EXPLANATION

1-1 WHAT IS THE RS-232C?

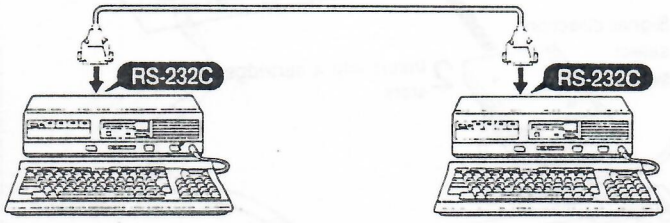
The MSX computer has a variety of functions as a stand-alone personal computer. You can make your own programs using MSX-BASIC, for example, store them on an external memory device such as a floppydisk, and the program can later be loaded and executed, or modified, printed out, and so forth.
The RS-232C can further expand the functions of the stand-alone computer by providing you, the user of the MSX computer, with communication with other devices such as a computer. By setting up communications with other personal computers, you can exchange the programs or data with the other computer lovers who live far away from you. This is because the RS-232C conforms to the industry standard for serial data interface between a modem and a terminal equipment standardized by the EIA (Electronic Industries Association), and two computers with the RS-232C interfaces can be connected via the modems and the telephone line, for example. An increasing number of personal computers have the RS-232C standard interface, and between these computers communication can easily be performed.

Before performing communication through the RS-232C interface, you may have to work on system set-up, communication mode settings such as transmit/receive speed, data length, and signal control. All those ...s which are said to be a little troublesome, can easily be executed using MSX-BASIC commands and functions specially provided for RS-232C communication. The usage of MSX-BASIC commands and functions is thoroughly covered in this manual.

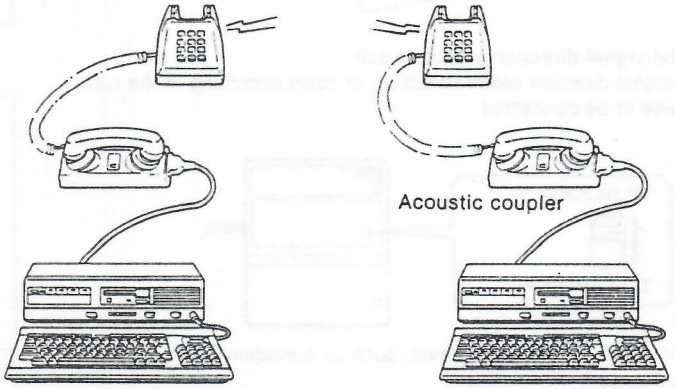
1-2 SYSTEM CONFIGURATION

To perform communication through the RS-232C interfaces, there are mainly three types of system configuration. The following illustrations show the examples when MSX computers have RS-232C interfaces on them.

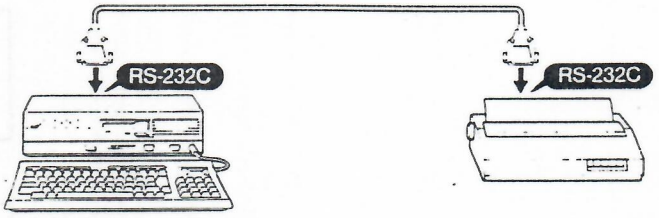
Connecting two computers directly via the RS-232C interface cable
This system is used to exchange data files between the computers via the RS-232C interface, for example.



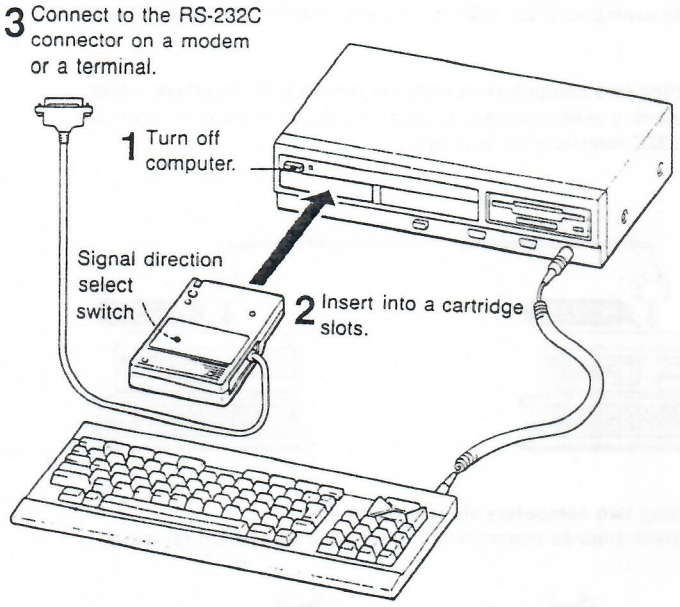
Connecting two computers via telephone line
This system enables communicating with the equipment far away.



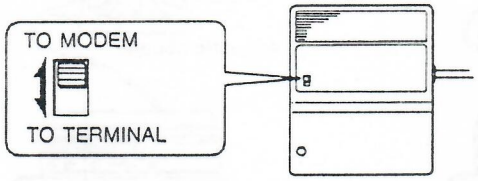
Connecting to peripherals
This system allows the MSX computer to utilize peripherals, such as a printer which is provided with an RS-232C interface.



CONNECTIONS



Setting the signal direction select switch
Slide the signal direction select switch up or down according to the type of the device to be connected.



To connect to modem-type equipment, such as a modem or an acoustic coupler:
Slide the switch to the **TO MODEM** position.

To connect to terminal-type equipment, such as a computer, a printer, or a display monitor:
Slide the switch to the **TO TERMINAL** position.

1-3 PREPARATION

The following flow chart indicates how to start RS-232C communication.
For details of each procedure, refer to the page shown in <**PAGE****>.

Turn off the power to your system.

If you use an RS-232C interface on the HBI-232 RS-232C Interface Cartridge

Insert the Interface Cartridge into the computer's slot, and connect the interface cable connector to the RS-232C interface connector of the other device such as a modem (acoustic coupler) or a computer.

<PAGE 3> and Instruction manual for HBI-232

If you use an RS-232C interface on the MSX computer

Connect the MSX computer to the RS-232C interface connector of the other device such as a modem (acoustic coupler) or a computer using the SMK-0031 RS-232C interface cable.

<PAGE 3> and Instruction manual for Sony MSX computer

Check the signal specifications of the RS-232C connector of the device to be connected, and then set the signal direction select switch on the Sony MSX computer or the HBI-232 RS-232C Interface Cartridge to "TO MODEM" or "TO TERMINAL".

<PAGE 5> and Instruction manual for the HBI-232 RS-232C Interface Cartridge or Sony MSX computer

Turn on the power to your system.

Set the RS-232C data format and communication mode so that communication mode matches to that of the connected device on the other side.
<PAGE 6>

Let's expand your computer world through the RS-232C port [1] !

For details of actual communication, see "2-1 Practicing RS-232C Serial Data Communication" on page 7 and BASIC command and function reference on page 11.

[1] The RS-232C port is the RS-232C interface on the MSX computer, or on the RS-232C Interface Cartridge.

1-3-1 SWITCHING THE SIGNAL FLOW

There are two types of RS-232C interface specifications: **MODEM** type and **TERMINAL** type. Since the RS-232C is a standard for serial data communication between a modem and a terminal equipment, MODEM type interface and TERMINAL type interface can be directly connected with a standard straight cable. However, if two MODEM type equipments or two TERMINAL type equipments are connected, a cross cable called a null modem cable is required. With the Sony MSX computer, switching between MODEM/TERMINAL can easily be performed using the signal direction select switch so that MODEM or TERMINAL equipment can be connected to the MSX computer without the null modem cable.

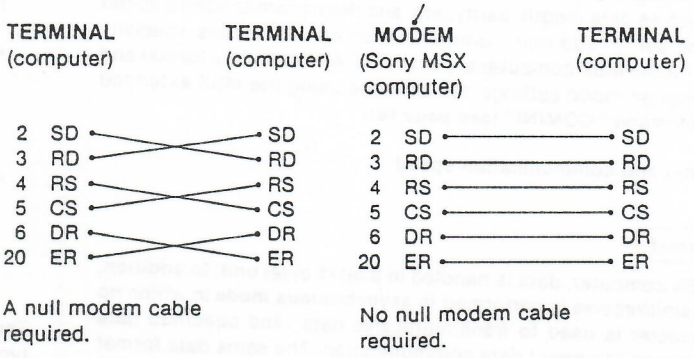
If the equipment to be connected is a TERMINAL type, set the switch to **TO TERMINAL** position, and if it is a MODEM type, set the switch to **TO MODEM** position. The function and flow direction of the signals when the MSX computer's switch is set to TO MODEM is as follows:

Pin No.	Signal name	Function when MSX computer's switch is set to TO MODEM	Singal flow TERM←→MODEM
1	FG	Protective ground	—
2	SD (TXD)	Transmit data	→
3	RD (RXD)	Receive data	←
4	RS (RTS)	Notifies the connected device that your MSX computer is ready to start transmitting data (Request to Send).	→
5	CS (CTS)	The connected device notifies your MSX computer that it is ready to receive data. (Clear to Send)	←
6	DR (DSR)	The connected device notifies your MSX computer that it is ready for both transmitting and receiving data. (Data Set Ready)	←
7	SG	Signal ground	—
8	CD (DCD)	The connected modem notifies your MSX computer that it has detected the carrier signal[1]. (Data Carrier Detect)	←
9-19	NC	No connection	—
20	ER (DTR)	Notifies the connected device that your MSX computer is ready for both transmitting and receiving data. (Data Terminal Ready)	→
22	CI (RI)	The connected modem notifies that it has detected the telephone ringing.	←

[1]Carrier detect signal is used to notify that the communication line is operative when the computers are connected to a telephone line through the modems (acoustic couplers) and a telephone line.

To connect two terminal equipments (computers)

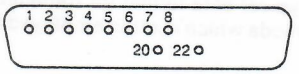
Sony MSX computer is designated to a modem equipment by setting the swith to TO TERMINAL position.



A null modem cable required.

No null modem cable required.

Interface connector pin assignment



1-3-2 SETTING THE RS-232C DATA FORMAT AND COMMUNICATION MODE

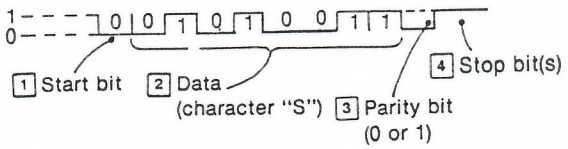
Before starting communication through the RS-232C interface, data format such as data length, parity, etc. and the transmit/receive speed have to be set. In addition, communication control modes specially provided for the MSX computer are available. All those data format and communication mode settings can be made using the MSX extended BASIC command **"COMINI" (see page 18).**

Data format and communication speed

Data format

In the MSX computer, data is handled in 8-bit (1 byte) unit. In addition, data transmit/receive is performed **in asynchronous mode** in which no sync character is used to transmit/receive data and specified data format assures of correct data communication. The same data format needs to be specified in both transmitter and receiver in the RS-232C serial data communication so that data can be exchanged without fail. The chart below shows the typical data format for transmitting ASCII character in asynchrounous mode which is often employed in the MSX computer.

To transmit "S" (53H = 01010011b)



1 Start bit
This bit indicates that the character following this bit is the data character. This bit is defined to be 0.

2 Data length
For the RS-232C communication, data length of one character has to be defined according to the type of data to be transmitted/received. The data length is specified in bit units, and **8-bit data length is usually employed in the MSX computer.**

5 bits, 6 bits: Used for special purposes such as domestic or international telex for example.
7 bits: Used only for ASCII code
8 bits: Used to exchange a program in machine language, or codes of 80H to FFH.

3 Parity bit length
The "parity bit" can be utilized for the RS-232C communication so that incorrect data transfer can be detected. This is the error checking method in which the total number of binary "1"s in a character data is always even or always odd.
The value of the parity bit is automatically set to 1 or 0 so that the total number of binary "1"s in a character data and the parity bit is always even or always odd.

Example

Character data = "A" (41H)
Data length = 8 bits
Even parity

<Transmitter> 0 01000001 (Parity bit) ⇒ <Receiver> 001000001
OK!: The total number of binary 1s is even.

<Transmitter> 0 01000001 ⇒ <Receiver> 001100001
Error!: The total number of binary 1s is odd.

The type of the parity check can be selected out of the following 4 types. **No parity check type is usually employed in the MSX computer.**

Even parity: Total number of binary "1"s is always set to even.
Odd parity: Total number of binary "1"s is always set to odd.
No parity: No parity check which is often employed in the RS-232C communication.
Ignore parity: When transmitting, no parity bit will be sent to the connected device, and when receiving the received parity bit will be ignored. This mode is effective only when the data length is 5 to 7 bits.

4 Stop bit length
The stop bit(s) is added at the end of a data character, and it indicates the end of the data. For the RS-232C communication, stop bit length has to be defined. Either 1 bit, 1.5 bits, or 2 bits is selectable, and **1-bit stop bit lengh is usually employed in the MSX computer.**

Baud rate

Baud rate is the data flow speed to transmit or receive data including start bit and stop bit(s) specified by the number of bits per second. The same speed has to be specified in both the connected devices on a communication line. However, it is possible to set a different speed for transmitting and receiving data in one device. One of the following baud rates can be selected:

50, 75, 110, 300, 600, 1200, 1800, 2000, 2400, 3600, 4800, 7200, 9600, 19200 bps (bits/second)

When communicating via an acoustic coupler and a telephone line, it is recommended to specify 300 or 1200 bps.

Note: To perform communication using BASIC, it is recommended to set a speed at or lower than 1200 bps so that correct data transfer is assured.

To set communication control modes
The following communication control modes are useful to modify the data delimiter according to the connected device, prevent data overflow, and so forth.

Shift-in/shift-out control

When the connected device on the other side uses 7-bit JIS (Japan Industry Standard) character set, the SI code (0FH) and SO code (0EH) control is used. SI code notifies that data following the code is the Japanese Katakana characters, and SO code the alphanumeric characters. **Since the MSX computer usually employs 8-bit character data, it is not necessary to specify SI/SO control unless the connected device uses the 7-bit JIS character set.**

Automatic line feed insert/delete control

The MSX computer uses a set of carriage return code (0DH) and line feed code (0AH) as a data delimiter. However, to communicate with a computer which puts only a carriage return code as a data delimiter, the line feed code after the carriage return code has to be deleted when transmitting data, and has to be added when receiving data by activating this control mode.
To perform RS-232C serial data communication with another MSX computer, normally it is not necessary to specify this mode.

XON/XOFF control

In RS-232C communication, data to transmit or received data is once stored in a specified buffer area which is called a file. The XON/XOFF control mode preventes overflow of the buffer. In this method, the receiver will send an XOFF code (13H) to the transmitter when 113 characters of data is in the receive buffer (128 characters), and will send XON code (11H) when there are 2 characters remaining in the receive buffer.
The transmitter will suspend data transmission when it receives the XOFF code, and will resume data transmission when it receives XON code.
When the computer is connected to a device which is not programmed to use the overflow control such as a printer, do not activate this overflow control mode.

CS-RS handshake

In this method, computer first sends the RS (Request to Send) signal to the modem to notify that it is to start data transmission, and when the modem sends back the CS (Clear to Send) signal to the computer, the computer starts data transmission. Namely, data exchange is started after the connected modem notifies the computer that it is ready to receive data responding to the request of data transmission from the computer.
Normally, CS-RS handshake control is employed. If the CS-RS handshake is activated with the MSX computer, the computer will suspend data transmission by PRINT#, SAVE, until the CS signal is set to ON. **When the CS-RS handshake method is not adopted in the connected device, do not specify the CS-RS handshake method.**

Time out

When CS-RS handshake is designated, data transmission is not resumed until the CS signal changes to ON. If a certain time is specified, time out error can be signaled. The time out error will be declared when the specified time has elapsed before the CS signal is set to ON, which prevents the computer from endlessly waiting for the CS signal set to ON. The time is specified in seconds in the range from 0 to 255.

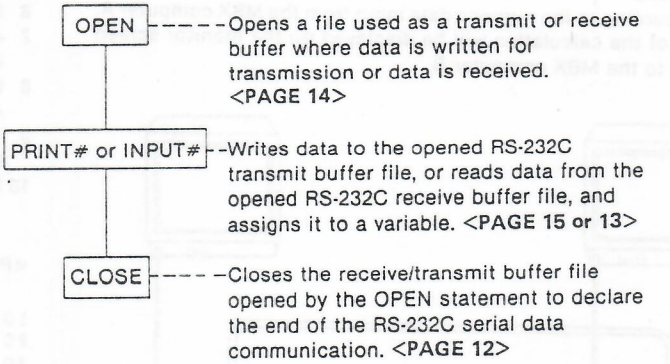
2-1 PRACTICING RS-232C SERIAL DATA COMMUNICATION

There are mainly two ways of using serial data communication for your MSX computer: data/program exchange with another computer or operating the MSX computer as a terminal of the other host computer. In the terminal mode, all your MSX computer can do is sending data from the keyboard, or receiving program from the host computer, for example.

2-1-1 DATA AND PROGRAM COMMUNICATION

Basic procedure for transmitting or receiving data
The following is the basic procedure for transmitting or receiving data after preparation is made. Setting a specified area in the memory is required for data exchange in the RS-232C communication. (Not necessary for transmitting/receiciving program) The specified area is called a file. For details of the MSX-BASIC command names described below, see the explanation in Paragraph 3.

<Transmit/receive data>



Note
The signals status will be as follows:
RS signal: **ON** when the OPEN statement is executed.
OFF when the CLOSE statement is executed.
ER signal: **ON** when the computer is turned on.
CS signal: In case CS-RS handshake control is activated by the MSX-BASIC command COMINI, data transmission by the PRINT# will be suspended until the CS signal is set to ON.

Basic procedure for transmitting or receiving program
If the connected device is another MSX computer, the transmitted or received programs can be utilized in both MSX computers. However, when the connected device is an other type of computer, the exchanged programs cannot be utilized either in the connected computer or in your MSX computer. However, if the MSX-BASIC programs is transmitted as a data file from an other type of computer to your MSX computer, you can use the MSX-BASIC program (see the program example on page 9).

For details of the MSX-BASIC command names described below, see the explanation in Paragraph 3.
The following are the MSX-BASIC commands to transmit or receive program via a specified RS-232C port (interface) after preparation is made.

<Transmit> SAVE ------ Sends a program in ASCII format through the specified RS-232C port.

<Receive> LOAD ------ Loads a program in ASCII format from the specified RS-232C port

Notes

The signals status will be as follows:

RS signal: ON before and while receiving a program.
OFF after receiving a program.

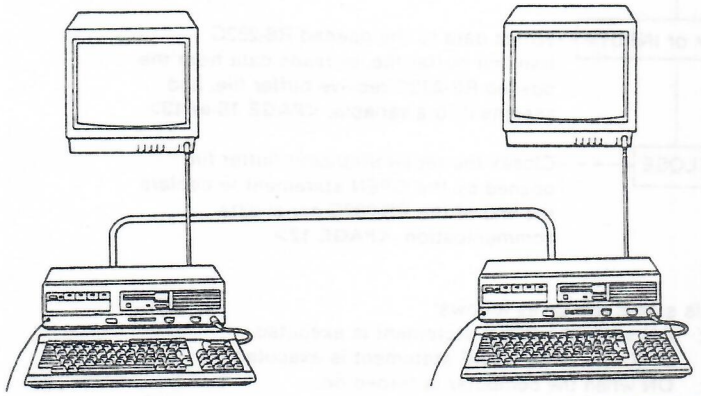
ER signal: ON when the computer is turned on.

CS signal: In case CS-RS handshake control is activated by the MSX-BASIC command COMINI, program transmission by the SAVE will be suspended until the CS signal is set to ON.

Examples for transmitting and receiving data

The following example is the case in which numeric data input from the keyboard of the MSX computer A is transmitted to the MSX computer B where the input numeric data is processed and then displayed on the monitor screen.

The calculation A + B + C = will be performed on the MSX computer B according to the numeric data input from the MSX computer A. The result of the calculation will be displayed on the monitor screen connected to the MSX computer B.



MSX computer A for transmitter MSX computer B for receiver

≪Program for MSX computer A as a transmitter≫

```
10 DIM A(2)                                                    -1
20 CALL COMINI ("0:8N3XNNNN",300,300,5)                        -2
30 OPEN "COM0:" FOR OUTPUT AS #1                               -3
40 PRINT: PRINT "Input data to send"                           -4
50 INPUT "A=";A(0)
60 INPUT "B=";A(1)                                             -5
70 INPUT "C=";A(2)
80 PRINT #1,"Start"                                            -6
90 S$=""                                                       -7
100 FOR I=0 TO 2
110  IF I<>2 THEN S$=S$+STR$(A(I))+"," ELSE S$=S$+STR$(A(I))   -8
120 NEXT I
130 PRINT #1,S$                                                -9
140 PRINT #1,"End"                                             -10
150 GOTO 40
```

The each step above means:

1 Declares the name of numeric type array variables from A (0) to A (2) where the value of A, B, and C will be assigned.
2 Initializes the RS-232C port numbered 0 (see page 11) so that the data format and communication modes settings will be performed as follows:

RS-232C port number: 0 (0)
Data length: 8 bits (8)
Parity check: No parity check (N)
Stop bit: 2 bits (3)
XON/XOFF control: Enables control (X)
CS-RS handshaking: No handshaking (N)
Automatic line feed insert/delete: No insert/delete (NN)
Shift-in/shift-out control: No control (N)
Transmit/receive speed: 300 bps (,300,300)
Time out: 5 seconds (,5)

3 Opens the file for the RS-232C transmit buffer with the file number 1.
4 Displays the message "Input data to send" on the monitor screen.
5 Prompts you to input the value of the variables A (0), A (1), and A (2) from the keyboard, one by one according to the displays "A = ", "B = ", and "C = " on the monitor screen.
6 Sends the message "Start" to the connected MSX computer B.
7 Assigns a null string to the string type variable S$ so that value set in S$ will be cleared.
8 Converts numeric type data assinged to the variables A (0), A (1), A (2) to the string type variable S$.
9 Writes data assigned to the S$ to the transmit buffer file 1 so that the data will be sent to the MSX computer B.
10 Sends the message "End" to the connected MSX computer B.

≪Program for MSX computer B as a receiver≫

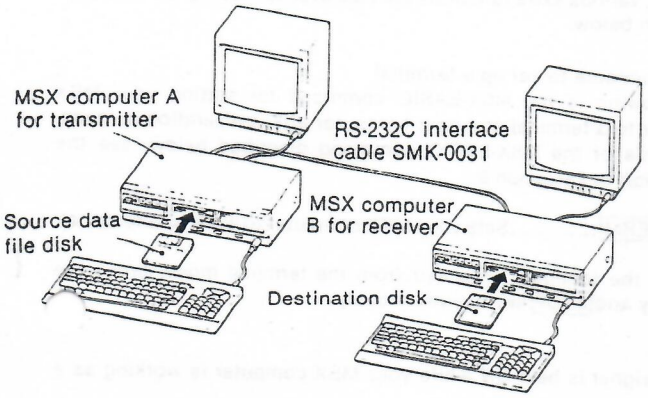
```
10 CALL COMINI("0:8N3XNNNN",300;300,5)                         -1
20 OPEN "COM0:" FOR INPUT AS #1                                -2
30 PRINT: PRINT "#### Now waiting data ####"                   -3
40 LINE INPUT #1,G$
50 IF G$="Start" THEN 60 ELSE 30                               -4
60 INPUT #1,A,B,C
70 PRINT "A=";A
80 PRINT "B=";B                                                -5
90 PRINT "C=";C
100 D=A+B+C                                                    -6
110 PRINT "A+B+C=";D
120 LINE INPUT #1,G$
130 IF G$="End" THEN 30 ELSE 120                               -7
140 END
```

The each step above means:

1 Initializes the RS-232C port numbered 0 so that the data format and communication modes settings will be performed the same as the MSX computer A, the transmitter.
2 Opens the file for the RS-232C receive buffer with the file number 1.
3 Displays the message "#### Now waiting data ####" on the monitor screen.
4 Starts reading data from the receive buffer file, and assigns it to numeric type variables A, B, and C if the message "Start" is sent from the transmitter.
5 Displays the numeric data assigned to the variables A, B and C on the monitor screen.
6 Calculates "A + B + C" and displayes the answer "D" by numeric constants.
7 If the "End" message is sent from the transmitter, waits for another data input from the transmitter's keyboard.

Exa es for transmitting and receiving a data file

The following example is the case in which data is exchanged between the two MSX computers. The data file of a MSX-BASIC program stored in a floppydisk of computer A is transmitted to and received by the MSX computer B, and the data file is stored on a destination floppydisk of the MSX comptuer B. Any desired file on the source floppydisk can be specified for transmission by inputing its file name from the keyboard.



≪Program for MSX computer A as a transmitter≫

```
10 MAXFILES=2                                              ---1
20 INPUT "Input data file name to send:";F$                ---2
30 PRINT "####Now sending data ####"                       ---3
40 CALL COMINI("0:8N3XNNNN",300,300,5)                     ---4
50 OPEN "COM0:" FOR OUTPUT AS #1                           ---5
60 OPEN F$ FOR INPUT AS #2                                 ---6
70 BUF$="Start":PRINT #1,BUF$                              ---7
80 IF EOF(2) THEN GOTO 120                                 ---8
90 LINE INPUT #2,BUF$                                      ---8
100 PRINT #1,BUF$                                          ---8
110 GOTO 80                                                ---8
120 BUF$="End":PRINT #1,BUF$                               ---9
130 CLOSE                                                  ---9
140 END                                                    ---9
```

The each step above means:

1 [illegible]ares the number of files that can be simultaneously opened in this program is 2.
2 Prompts the user to input the name of the file to be transmitted from the keyboard. The input file name will be assinged to the variable F$.
3 Displays the message "#### Now sending data ####" on the monitor screen.
4 Initializes the RS-232C port numbered 0 so that the data format and communication modes settings will be performed as follows:

RS-232C port number: 0 (0)
Data length: 8 bits (8)
Parity check: No parity check (N)
Stop bit: 2 bits (3)
XON/XOFF control: Enables control (X)
CS-RS handshaking: No handshaking (N)
Automatic line feed insert/delete: No insert/delete (NN)
Shift-in/shift-out control: No control (N)
Transmit/receive speed: 300 bps (,300,300)
Time out: 5 seconds (,5)

5 Opens the RS-232C file for the transmit buffer as the file number 1.
6 Opens the data file "F$" stored on the floppydisk of the MSX computer A as the file number 2.
  • The mode "INPUT" is specified so that the contents of the file number 2 can be read and assigned to a string type variable.
7 Sends the message "Start" to the connected MSX computer B.
8 Checks if the EOF (end-of-file) code has been read from the file number 2. If EOF code has not been received, reads a character string one by one from the file number 2 on the floppydisk, and then assigns it to a string type variable BUF$. Writes data assigned to the BUF$ to the transmit buffer file 1 so that the data will be sent to the MSX computer B.
9 If the EOF (end-of-file) code has been received in the file numbered 2, the message "End" will be sent to the MSX computer B, and all opened files will be closed, and this program will end.

≪Program for MSX computer B as a receiver≫

```
10 MAXFILES=2                                              ---1
20 CLEAR 2000                                              ---2
30 DIM BUF$(500)                                           ---3
40 INPUT "Input file name to save:";F$                     ---4
50 CALL COMINI("0:8N3XNNNN",300,300,5)                     ---5
60 OPEN "COM0:" FOR INPUT AS #1                            ---6
70 OPEN F$ FOR OUTPUT AS #2                                ---7
80 LINE INPUT #1,S$                                        ---8
90 IF S$="Start" THEN GOTO 100 ELSE 80                     ---8
100 N=1                                                    ---9
110 LINE INPUT #1,S$                                       ---9
120 IF S$<>"End" THEN BUF$(N)=S$ ELSE 160                  ---10
130 PRINT BUF$(N)                                          ---10
140 N=N+1                                                  ---10
150 GOTO 110                                               ---10
160 PRINT"####Now saving data ####"                        ---10
170 N=N-1                                                  ---10
180 FOR I=1 TO N                                           ---10
190 PRINT #2, BUF$(I)                                      ---10
200 NEXT                                                   ---10
210 CLOSE                                                  ---11
220 END                                                    ---11
```

The each step of the above program means:

1 Declares the number of files simultaneously opened in this program is 2.
2 Sets the size of the character string area to 2000 bytes in memory.
3 Declares an area of 501 string type variables from BUF$ (0) to BUF$ (500).
4 Prompts the user to input the name of the file to save the received data from the keyboard.
5 Initializes the RS-232C port numbered 0 so that the same data format and communication modes settings as the MSX computer A are performed on the MSX computer B.
6 Opens the receive buffer file with the file number 1.
7 Opens the data file "F$" on a floppydisk of the MSX computer B with the file number 2 so that received data file will be written into the file.
  • The mode "OUTPUT" is specified so that the received data will be written into the file numbered 2.
8 Waits for the massage "Start" to be sent from the transmitter.
9 Upon receipt of the message "Start", starts receiving character strings one by one until the message "End" will be sent from the transmitter. While receiving character strings one by one, the character string last received is displayed on the screen.

10 Upon receipt of the message "End", the message "#### Now saving data ####" will be displayed on the screen, and starts writing the received data in the receive buffer to the file numbered 2 on the floppydisk.
11 Closes all opened files when all data in the receive buffer is stored on the floppydisk, and the program will end.

Examples for transmitting and receiving a program
The following example is the case in which MSX computer A will transmit a program in an ASCII format, and the MSX computer B will receive the program also in an ASCII format until EOF (end-of-file) code is received. The RS-232C communication port number 0 is utilized in both transmitter and receiver.

MSX computer A (transmitter)

```
CALL COMINI("0:8N3XNNNN",1200,1200,5)
SAVE "COM0:"
```

MSX computer B (receiver)

```
CALL COMINI("0:8N3XNNNN,1200,1200,5)
LOAD "COM0:"
```

2-2 TERMINAL OPERATION

In this mode, your MSX computer is often connected to the host computer through a modem equipment (acoustic coupler), for example, and is used as a terminal of the host computer. If set to the terminal mode, the program of your MSX computer is no longer operative, and all your MSX computer can do is to just display the data transmitted from the host computer, and to input the data from the keyboard to transmit to the host computer.
However, various extra functions are also available using the keyboard as shown below.

Basic procedure to set up a terminal
The following is the MSX-BASIC command for setting your MSX computer to a terminal of a host computer after preparation is made. For details of the MSX-BASIC command discribed below, see the explanation in Paragraph 3.

COMTERM.....Sets your MSX computer to work as a terminal.

To reset the terminal, or to exit from the terminal mode, press the CTRL key and STOP key simultaneously.

Note
The RS signal is held ON while your MSX computer is working as a terminal.

Extra terminal functions using the keyboard
The data received from the host computer, or input from the keyboard and transmitted to the host computer, is also displayed on the screen, printed out, and so forth. In addition, the break sequence[1] can be transmitted using the keys on the keyboard of your MSX computer.
Press the following set of keys to activate the extra terminal function mode.

SHIFT + F1
Displays the received control codes (00H to 1FH) by "^" and the character assigned to the control code plus 40H.
ex.) The return code (0DH) will be displayed as follows:

To exit from this function mode, press the SHIFT and F1 keys simultaneously again.

[1]Break sequence: The break sequence are used to set t... SD signal to spacing state 0.

SHIFT + F2
Displays the data input from the keyboard on the screen. To exit from this function mode, press the SHIFT and F2 keys simultanously again.

SHIFT + F3
Displays and prints out the data input from the keyboard at the same time. To exit from this function mode, press the SHIFT and F3 keys simultaneously again.

STOP
Press and hold this key to transmit break sequence to the host computer.

Note
The SHIFT key is identical with the ⇧ key on the MSX computers such as HB-10P.

3-1 COMMANDS AND FUNCTIONS OF THE MSX-BASIC FOR RS-232C COMMUNICATION

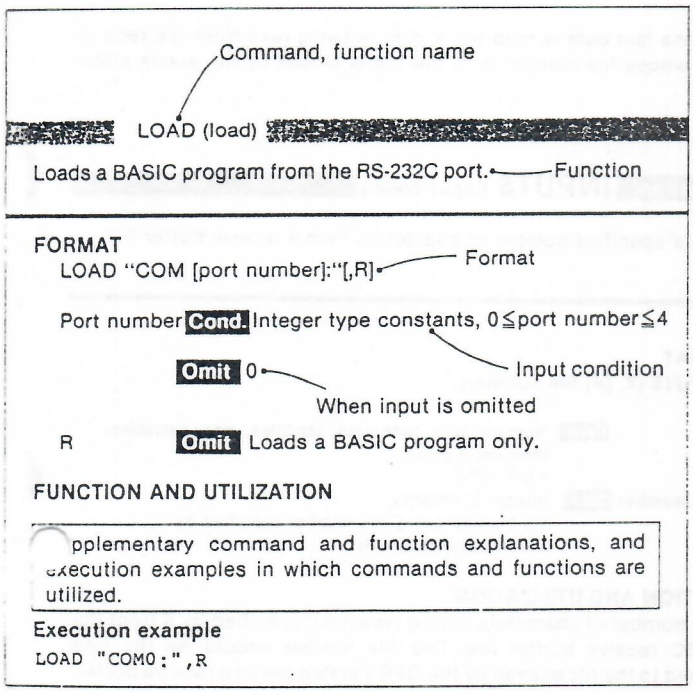
The MSX-BASIC commands and functions are specially provided for the RS-232C communication. Using those commands and functions, various communication modes, interrupt control, receive/trasmit control, all of which are indispensable for the RS-232C communication, can easily be set.
The follwoing MSX-BASIC commands and functions are divided into two categories:

MSX-BASIC commands and functions used for RS-232C communication

Extended MSX-BASIC commands for RS-232C communication
.....Executed with the "CALL" statement

The commands and functions in each category are explained in alphabetical sequence.

3-1 INTRODUCTORY REMARKS



- In regard to a function, **FUNCTION** is written in front of the function name as follows:

EOF (end of file)

- An input item inside [ ] in the Format section can be omitted.
- The "..." indicates that the input item can be repeatedly specified within the input range per line as follows:

INPUT #file number, variable [,variable].....

3-1-2 LIST OF COMMANDS AND FUNCTIONS FOR RS-232C

<MSX-BASIC commands and functions for RS-232C>

	Name of command or function	Function	Page
Command	OPEN	Opens an RS-232C file[1].	14
	CLOSE	Closes the file opened by an OPEN statement.	12
	PRINT#	Writes data to a transmit buffer file.	15
	PRINT#USING	Writes data to a transmit buffer file in a specified format.	15
	INPUT#	Reads data from a receive buffer file, and assigns it to a variable.	13
	LINE INPUT#	Reads a string from a receive buffer file, and assigned it to a variable.	13
	SAVE	Sends a BASIC program to the RS-232C port[2].	17
	LOAD	Loads a BASIC program from the RS-232C port.	13

[1]The file is the specified buffer area which is used as a **receive/transmit buffer** for the RS-232C communication.

[2]The RS-232C port number is specified to the RS-232C interface as follows.

0:
- RS-232C interface on the MSX computer
- RS-232C interface on the RS-232C interface cartridge which is first inserted into the computer's slot when the MSX computer has no resident RS-232C interface.

1-4: Port number increases one by one as an RS-232C interface is added to the MSX computer using the RS-232C interface cartridge, for example.

	Name of command or function	Function	Page
Command	RUN	Loads a program from the RS-232C port and executes the program.	17
	MERGE	Loads a program in ASCII format from the RS-232C port, and merges it into the program currently in memory.	14
Function	EOF	When the last data of a file has been read, −1 is given, otherwise 0 is given.	12
	INPUT$	Inputs a specified number of characters from the receive buffer file.	12
	LOC	Returns the number of characters in the receive buffer file.	14
	LOF	Returns the free space remaining in the receive buffer file.	14

<MSX-BASIC extended commands for RS-232C>

Name of the commands	Function	Page
COMINI	Initializes the communication mode.	18
COMTERM	Sets the MSX computer in the terminal mode.	20
COMDTR	Sets the ER (DTR) signal to ON/OFF.	18
COMBREAK	Sends the break sequence.	17
COMSTAT	Reads the RS-232C port status.	19
COM GOSUB	Declares a subroutine to which program branches when an interrupt occurs from the RS-232C port.	17
COMON	Enables the interrupt from the RS-232C port.	19
COMOFF	Disables the interrupt from the RS-232C port.	19
COMSTOP	Suspends the interrupt from the RS-232C port.	19

The above extended commands are executed with the CALL (–) statement.

3-1-3 MSX-BASIC COMMANDS AND FUNCTIONS

MSX-BASIC commands and functions used for RS-232C communication.

CLOSE (close)

Closes a file opened by an OPEN statement.

FORMAT

CLOSE [#] [file number] [,file number]. . . . . .

File number Cond. Integer constants,
1 ≦ file number ≦ the number specified by MAXFILES = statement
Omit Closes all the files.

FUNCTION AND UTILIZATION

The file number has to be the one assigned to the file opened with the OPEN statement. The file number of the file closed can be used again when opening a new file. If the file closed is a transmit buffer file, an EOF code (1AH) will be sent to the connected device.

- The opened files will also be closed by the RUN, END, CLEAR, or NEW commands.

Execution example

```
CLOSE #1,2,3
```

The files numbered 1, 2, and 3 are all closed.

Function EOF (end of file)

When the last data of a file has been read, –1 is given, otherwise 0 is given.

FORMAT

EOF (file number)

File number Cond. Integer constants, variables, array variables, their expressions,
1 ≦ file number ≦ the number specified by MAXFILES = statement

Given value: Integer type (–1 or 0)

FUNCTION AND UTILIZATION

The file is the one opened as a receive buffer by the OPEN statement. This function checks if the EOF code (1AH) which indicates the end of data is received in the receive buffer file or not. If –1 is given, EOF code is received, otherwise 0 is given.

Execution example

```
IF EOF(1) THEN CLOSE #1
```

When the last data is read while data is being read from the receive buffer whose file number is 1, the file is closed by the above statement.

Function INPUT$ (input dollar)

Inputs a specified number of characters from a receive buffer file.

FORMAT

INPUT$ (X, [#] file number)

X Cond. Numeric type constants, variables, array variables, their expression,
1 ≦ X <256

File number Cond. Integer constants,
1 ≦ file number ≦ the number specified by MAXFILE = statement

FUNCTION AND UTILIZATION

Reads number of characters (string type data) specified by X from the RS-232C receive burffer file. The file number should be the one assigned to the file opened by the OPEN statement as a receive buffer.

Execution example

```
10 OPEN "COM0:" FOR INPUT AS #1
20 X$=INPUT$(50,#1)
30 CLOSE
```

Opens an RS-232C receive buffer file with the file number 1, inputs 50 characters from the file, and then closes the file.

Range of "X"

During initial status, if X is outside the range from 1 to 200, an error occurs. When the size of the character area is set to more than 255 by a CLEAR statement, a value from 1 to 255 can be selected.

INPUT# (input number)

Reads data from a receive buffer file, and assigns it to a variable.

FORMAT

INPUT# file number, variable [,variable]. . . . .

File number **Cond.** Integer constants,
1 ≦ file number ≦ the number specified by MAXFILES = statement

Variable **Cond.** Numeric type or string type, their array variables

FUNCTION AND UTILIZATION

Reads data from the receive buffer file. The file number has to be the one assigned to the file opened by the OPEN statement as a receive buffer. If the data is numeric type, spaces, return codes, and line feed codes before the data are ignored. If the data is string type, the data from the first character to the character before a space, comma, return code, or line feed code is read as one data. If the characters are inside " ", only these characters are read as data.

To specify the variables, be sure to assign the type of variables appropriate for the data to be read as follows:

ex.)
"ABCDEFG"_ _ _ _A$ (string type variable)
1,2,3,4,5_ _ _ _ _ _A% (numeric type variable)

Execution example

```
10 OPEN "COM0:" FOR INPUT AS #1
20 IF EOF(1) THEN GOTO 50
30 INPUT #1,A$:PRINT A$
40 GOTO 20
50 CLOSE #1
```

Opens a receive buffer file numbered 1, reads string type data from the file, and assigns the data to the variable A$ while displaying it on the screen.
If the EOF code is received (the last data has been read), the file is closed.

LINE INPUT# (line input number)

Reads a string from a receive buffer file, and assign it to a variable.

FORMAT

LINE INPUT # file number, variable

File number **Cond.** Integer constants,
1 ≦ file number ≦ the number specified by MAXFILES = statement

Variable **Cond.** String type variables, array variables

FUNCTION AND UTILIZATION

Reads string type data from the RS-232C receive buffer file. However, a space, comma, and line feed codes are not considered as punctuation for the data string, which differs from the INPUT# statement. The character string including those items is assinged to a variable as character string data. Only the return code is considered to be punctuation for data. Up to 254 characters can be read from the file.

Execution example

```
10 OPEN "COM0:" FOR INPUT AS #1
20 IF EOF(1) THEN GOTO 60
30 LINE INPUT #1,A$
40 PRINT A$
50 GOTO 20
60 CLOSE #1:END
```

Opens an RS-232C receive buffer file with the file number 1, reads string data from the file, and assings the data to the string type variable A$. The contents of the data is displayed on the screen. If end of data character is received, the file numbered 1 is closed.

LOAD (load)

Loads a BASIC program from the RS-232C port.

FORMAT

LOAD "COM [port number] :"[,R]

Port number **Cond.** Integer type constants,
0 ≦ port number ≦ 4
Omit 0

R **Omit** Loading the program only

FUNCTION AND UTILIZATION

A LOAD statement closes all opened files and deletes the current program from memory, then loads a BASIC program in the ASCII format into memory from the specified port. If the "R" option is specified, however, all data files remain open and the program that is loaded is automatically executed. Upon receipt of the EOF code (1AH), the program loading will end.

Execution example

```
LOAD "COM0:",R
```

Function LOC (location)

Returns the number of characters in the receive buffer file.

FORMAT

LOC (file number)

File number Cond. Numeric constants, variables, array variables, their expressions,
1 ≦ file number ≦ the number specified in MAXFILES = statement

FUNCTION AND UTILIZATION

The file number should be the one assinged to the file opened by the OPEN statement as a receive buffer.
The size of the RS-232C receive buffer is 128 charcters max.

Function LOF (length of file)

Returns the free space remaining in the receive buffer file.

FORMAT

LOF (file number)

File number Cond. Numeric constants, variables, array variables, their expressions,
1 ≦ file number ≦ the number specified by the MAXFILES = statement

Given value: Integer type

FUNCTION AND UTILIZATION

Returns the size of the free space remaining in the receive buffer by the number of characters. The file number should be the one assigned to the file opened by the OPEN statement as a receive buffer.

MERGE (merge)

Loads a program in ASCII format from the RS-232C port, and merges it into the program currently in memory.

FORMAT

MERGE "COM [port number]:"

Port number Cond. Integer type constants,
0 ≦ port number ≦ 4
Omit 0

Given value: Integer type

FUNCTION AND UTILIZATION

If some of the line numbers of the program in memory match line numbers of the program incoming from the RS-232C port, the lines of the program from the RS-232C port replaces the matching lines of the program currently in memory.
After the MERGE command executed, the merged program will reside in memory, and control will return to BASIC at the command level.

Execution example

```
MERGE "COM0:"
```

Loads lines of the program from the RS-232C port numbered 0, and merges them with the program in memory.

OPEN (open)

Opens an RS-232C file.

FORMAT

OPEN "COM [port number]:" [FOR mode] AS [#] file number

Port number Cond. Integer type constants,
0 ≦ port number ≦ 4
Omit 0

Mode Cond. OUTPUT, INPUT
Omit OUTPUT/INPUT

File number Cond. Integer constants,
1 ≦ file number ≦ the number specified by MAXFILES = statement

FUNCTION AND UTILIZATION

Allocates an I/O buffer which will be used as a transmit or receive buffer for RS-232C communication. The buffer allocated is called a file. The transmit buffer file will be opened if OUTPUT is specified as the mode, and the receive buffer file will be opened if INPUT as the mode. If "mode" is not specified, and no EOF (end-of-file) code handling is done, the RS-232C port can be accessed for both transmitting and receiving data.
An OPEN statement must be executed before the following statements using the RS-232C files:

PRINT#, PRINT# USING, INPUT#, LINE INPUT#, INPUT$

Execution example

```
OPEN "COM0:" FOR OUTPUT AS #1
```

Opens RS-232C transmit buffer with the file number 1.

PRINT # (print number)

Writes data to an RS-232C transmit buffer file.

FORMAT

PRINT # file number, expression [separator] [expression].....

File number	Cond.	Integer constants, 1 ≦ file number ≦ the number specified by MAXFILES = statement
Expression	Cond.	String type and numeric type constansts, variables, array variables, their expressions
Separator	Cond.	Comma (,) or semicolon (;)

FUNCTION AND UTILIZATION

The file is the one opened by the OPEN statement as a transmit buffer. Numeric type constants, numeric type and string type variables are written as they are, and string type constants are written inside quotation marks (" ").

Separator function

When data is punctuated with a comma (,), spaces are inserted between the data by a 14-digit tab function, and when it is punctuated a semicolon (;), it is followed by the next data. If a separator is not written at the end, return code and line feed code will be output.

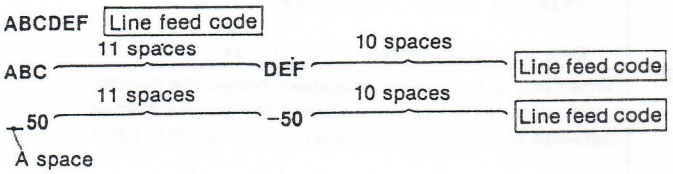
Numeric data and signs

In regard to signs that indicate positive or negative, " + " is omitted while "–" sign is transmitted.

Execution example

```
10 OPEN "COM0:" FOR OUTPUT AS #1
20 A$-"ABC":B$-"DEF"
30 PRINT #1,A$;B$
40 PRINT #1,A$,B$
50 PRINT +50,-50
60 CLOSE #1
```

Using the above program, data will be transmitted in the following format:



PRINT # USING (print number using)

Writes data to a transmit buffer file in a specified format.

FORMAT

PRINT # file number USING format symbol; expression [,expression]. . . . .

File number	Cond.	Integer constants, 1 ≦ file number ≦ the number specified by MAXFILES = statement
Expression	Cond.	String type and numeric type constants, variables, array variables, their expressions

FUNCTION AND UTILIZATION

Writes data specified by the expression in a specified format to a transmit buffer file, and then the data will be transmitted from the port. The file should be the one opened by the OPEN statement as a transmit buffer file. The value of an expression is displayed in a format specified by a format symbol as follows:

Symbol	Expression format and Execution example
" ! "	Outputs the first 1 character. `PRINT #1 USING "!";"United","Nation"` Data to be transmitted → UN
"\␣\" (n spaces)	Outputs n + 2 characters. When data is smaller than n + 2 characters, inserts spaces for the residual characters. `PRINT #1 USING "\  \";"ABCDEF","GHI", "JKLM"` Data to be transmitted → ABCDGHI JKLM

" & "	Outputs all character string.
	`10 OPEN "COM0:" FOR OUTPUT AS #1`
	`20 A$-"North":B$-"South"`
	`30 PRINT #1 USING "&Pole";A$,B$`
	`40 CLOSE #1`
	Data to be transmitted → North Pole South Pole

" # "	Writes # by the number of numeral digits to be transmitted.
	Decimal point is " . ".
	
	`PRINT #1 USING "POINT:###.#";123.4`
	
	Data to be transmitted → `POINT:123.4`
	
	● When the number of integer digits is less than the specified # number, transmitted data is preceded by spaces, and if it is more, "%" is added before the data.
	
	`10 OPEN "COM0:" FOR OUTPUT AS #1`
	`20 PRINT #1 USING "####";12`
	`30 PRINT #1 USING "####";12345`
	`40 CLOSE #1`
	
	Data to be transmitted → ␣12 [LINE FEED CODE]
	%12345
	space
	● When the number of digits in a fraction of numeric data is smaller than the specified # number, "0" is added, and when it is larger, it is rounded to the nearest whole number.
	
	`10 OPEN "COM0:" FOR OUTPUT AS #1`
	`20 PRINT #1 USING "##.##";25.3`
	`30 PRINT #1 USING "##.##";25.345`
	`40 CLOSE #1`
	
	Data to be transmitted → `25.30` [Line Feed Code]
	25.35
	
	The " + " sign of numeric data is ignored and the "␣" sign is counted as one digit.

	`10 OPEN "COM0:" FOR OUTPUT AS #1`
	`20 PRINT #1 USING "###";+123`
	`30 PRINT #1 USING "###";-123`
	`40 CLOSE #1`
	
	Data to be transmitted → `123` [Line Feed Code]
	%-123

" + "	"+" is added if it is a positive numeral, and "−" is added if it is a negative numeral before or after the numeric data.
	
	`10 OPEN "COM0:" FOR OUTPUT AS #1`
	`20 PRINT #1 USING "+####";123,-123`
	`30 PRINT #1 USING "####+";123,-123`
	
	Data to be transmitted → `123 -123` [Line Feed Code]
	123+ 123-

"−"	"−" is added after negative numeric data. `PRINT #1 USING "###-";123,-123` Data to be transmitted → `123 123-`

" * * "	The space before numeric data is filled with "*". One "*" in the format expresses one digit.
	
	`10 OPEN "COM0:" FOR OUTPUT AS #1`
	`20 PRINT #1 USING "**######";123`
	`30 PRINT #1 USING "**######";-234`
	`40 CLOSE #1`
	
	Data to be transmitted → `*****123` [Line Feed Code] `****-234`

"££"	Adds "£" before numeric data. One "£" in the format is counted as one digit.
	
	`10 OPEN "COM0:" FOR OUTPUT AS #1`
	`20 PRINT #1 USING "££###";1234`
	`30 PRINT #1 USING "+££###";-1234`
	`40 CLOSE #1`
	
	Data to be transmitted → `£1234` [Line Feed Code]
	-£1234

"**£"	Adds "£" just before the numeric data, and the space before that is filled with "*" `PRINT #1 USING "**£###.##";12.34` Data to be transmitted → `***£12.34`
" , "	When this is specified somewhere before the decimal point, data is transmitted by the insertion of commas between each 3 digits to the left of the decimal point. `PRINT #1 USING "#,######.##";12345.67` Data to be transmitted → ␣␣12,345.67
"^^^^"	Transmit numeric data by floating point type format. "^^^^" corresponds to the digits for the exponent part. `PRINT #1 USING "##.##^^^^";234.56` Data to be transmitted → ␣2.35E+02

RUN (run)

Loads a program from the RS-232C port, and executes the program.

FORMAT

RUN "COM [port number]:" [,R]

Port number	Cond.	Integer type constants, 0 ≦ port number ≦ 4
	Omit	0
R	Omit	All data files are closed.

FUNCTION AND UTILIZATION

Loads a program in ASCII format from the RS-232C port, and upon receipt of the EOF code (1AH), stops loading the program and executes it.
The RUN command closes all opened files and deletes the current contents of memory before loading the designated program. When the "R" option is specified, however, all data files remain opened.

Execution example

```
RUN "COM0:",R
```

Loads a program from the RS-232C port numbered 0, and executes the loaded program. The all data files remain opened, and no memory contents will be erased by this command.

SAVE (save)

Sends a BASIC program to the RS-232C port.

FORMAT

SAVE "COM [port number]:"

Port number	Cond.	Intergers type constants, 0 ≦ port number ≦ 4
	Omit	0

FUNCTION AND UTILIZATION

Sends an MSX-BASIC program to the specified RS-232C port. and the program will be transmitted in ASCII format from the port.
When the transmission of data is completed, the EOF code (1AH) will be sent at the end of the data.

Execution example

```
SAVE "COM0:"
```

COM GOSUB

Declares a subroutine to which program branches when an interrupt occurs from the RS-232C port.

FORMAT

CALL COM ([Port number:], GOSUB start line number)

Port number	Cond.	Integer type constants, 0 ≦ port number ≦ 4
	Omit	0
Start line number	Cond.	Integer constants, 0 ≦ number ≦ 65529

FUNCTION AND UTILIZATION

Sets the starting line number of a subroutine to trap when the first character is received after CALL COMON (see page 58) is executed. If another interrupt occurs while the subroutine, the interrupt will be suspended because CALL COMSTOP is automatically executed.
Append the RETURN statement at the end of the interrupt service routine so that program execution will return to a location next to the CALL COM GOSUB after completing the subroutine. The RETURN statement automatically executes CALL COMON to enable interrupt from the RS-232C port unless CALL COMOFF has been explicitly executed inside the subroutine.

Note: Interrupt does not take place when MSX-BASIC is not executing a program. When an error trap (resulting from an ON ERROR statement) takes place, it automatically disables all event trappings (including ERROR, STRIG, STOP, SPRITE, INTERVAL and KEY).

Execution example

```
CALL COM(,GOSUB 1000)
```

Specifies the line 1000 as the start line of the subroutine which is executed when a character is input from the RS-232C port number 0.

COMBREAK (communication break)

Sends break sequence.

FORMAT

CALL COMBREAK (["port number:"], expression)

Port number	Cond.	Integer type constants, 0 ≦ port number ≦ 4
Expression	Cond.	Numeric type constants, variables, array variables, their expression, 3 ≦ expression ≦ 32767

FUNCTION AND UTILIZATION

Sends break sequence to the specified RS-232C port by the number of characters specified by the "expression".
All transmit data will be 0 by sending the break sequence, which indicate that transmission is suspended.

Execution example

```
CALL COMBREAK(,20)
```

The 20 break characters will be sent to the RS-232C port number 0.

COMDTR

Sets the ER (DTR) signal to ON/OFF.

FORMAT

CALL COMDTR (["port number:"], expression)

Port number	Cond.	Integer type constants, 0 ≦ port number ≦ 4
	Omit	0
Expression	Cond.	Numeric type constants, variables, array variables, their expression

FUNCTION AND UTILIZATION

Turns off the ER (DTR) signal when the value of "expression" is 0, otherwise turns on the ER signal. At the computer's power-on, the ER signal is ON.

Execution example

```
CALL COMDTR(,0)
```

The ER (DTR) signal from the RS-232C port 0 will be turned off.

COMINI (communication initialize)

Initializes the communication mode.

FORMAT

CALL COMINI (["data string expression"] [,receive baud rate] [,transmit baud rate] [,time out]

Data string	Cond.	String type constants, variables, array variables, and their expression
	Omit	"0:8N1XHNNN" .
Receive baud rate	Cond.	Numeric type constants, variables, array variables, and their expression, 50 ≦ Receive baud rate ≦ 1200
	Omit	1200
Transmit baud rate	Cond.	Numeric type constants, variables, array variables, and their expression, 50 ≦ Transmit baud rate ≦ 1200
	Omit	the same baud rate as the receive baud rate
Time out	Cond.	Numeric type constants, variables, array variables, and their expression 0 ≦ time out ≦ 255
	Omit	0

FUNCTION AND UTILIZATION

For details of the data format and the communication modes, read the "1-3-2 SETTING THE RS-232C DATA FORMAT AND COMMUNICATION MODE."

At the computer's power-on, CALL COMINI will automatically be executed with the initial settings as shown below. Only when you need to change those initial settings, execute CALL COMINI.

The contents of the "data string expression" consists of a value for data length, parity, stop bit, and so forth. Define the value of the data string expression according the following format:

[0:] [8 [N [1 [X [H [N [N [N]]]]]]]

RS-232C port number
0* to 4

Data length
5: 5 bits
6: 6 bits
7: 7 bits
8: 8 bits*

Parity check
E: Even parity
O: Odd parity
I: Ignore parity
N: No parity*

Length of stop bits
1: 1 bit*
2: 1.5 bits
3: 2 bits

Shift-in/Shift-out control
S: Enables control
N: Disables control*

Automatic LF code delete (when transmitting)
A: Deletes LF code
N: Not delete LF code*

Automatic LF code insert (when receiving)
A: Inserts LF code
N: Not insert LF code*

CS-RS handshake
H: Handshaking*
N: No handshaking

XON/XOFF control
X: Enables control*
N: Disables control

***Initial settings**

As for the "transmit/receive speed", it is possible to set a different baud rate (speed) for data transmission and data reception.

Execution example

```
CALL COMINI("0:8E")
```

Even parity is newly set instead of the initial setting of "no parity. Items within the (" ") can be omitted from the right.

```
CALL COMINI("0:8 3  N",1200)
```

Length of stop bit is set to 2 bits, automatic LF code insert is not set and 1200 baud rate for receiving and transmission with other modes remain at their initial settings. The items within the " " can be omitted by spaces in place.

- To omit all items in ( ), omit the ( ) together with the items in it.

COMON

Enables the interrupt from the RS-232C port.

FORMAT

CALL COMON (["port number:"])

Port number **Cond.** Integer type constants, 0 ≦ port number ≦ 4
Omit 0

FUNCTION AND UTILIZATION

Enables interrupt caused by incoming characters from the specified RS-232C port. If the starting line number of the subroutine is specified with the CALL COM GOSUB statement, the subroutine will be executed.

COMOFF

Disables the interrupt from the RS-232C port.

FORMAT

CALL COMOFF (["port number:"])

Port number **Cond.** Integer type constants, 0 ≦ port number ≦ 4
Omit

FUNCTION AND UTILIZATION

Disables interrupt caused by incoming character from the specified RS-232C port. After this statement is executed, the interrupt will not take place even if there is an interrupt request from the RS-232C port.

COMSTOP

Suspends the interrupt from the RS-232C port.

FORMAT

CALL COMSTOP (["port number:"])

pc ımber **Cond.** Integer type constatns, 0 ≦ port number ≦ 4
Omit 0

FUNCTION AND UTILIZATION

Suspends the interrupt request by incoming characters from the RS-232C port until the CALL COMON statement is executed.

COMSTAT (communication status)

Reads the RS-232C port status.

FORMAT

CALL COMSTAT (["port number:"], variable)

Port number **Cond.** Integer type constants, 0 ≦ port number ≦ 4
Omit 0

Variable **Cond.** Numeric type variables, array variables

FUNCTION AND UTILIZATION

Reads the status of the specified RS-232C port. The status is returned in numeric data, and it is assigned to the variable. The bit assignments of the numeric data, if its binary expression is given, are as follows:

MSB bit 15 **Receive buffer overflow error** (Data is transmitted when the buffer is full.)
0: No error
1: Error occurred

bit 14 **Time out error** (The specified time has elasped since the CS signal had been OFF.)
0: No error
1: Error occurred

bit 13 **Framing error** (The binary "0" bit has been received instead of the stop bit.)
0: No error
1: Error occurred

bit 12 **Overrun error** (Next data is received before reading the last data from the receive buffer file.)
0: No error
1: Error occurred

bit 11 **Parity error** (see page 6)
0: No error
1: Error occurred

bit 10 **Control break key (CTRL + STOP keys) was pressed**
0: Not pressed
1: Pressed

bit 9 Reserved: 0
bit 8 Reserved: 0

bit 7 **CS (CTS) signal status**
0: OFF
1: ON

bit 6 **Timer/counter set for the time out error detection**
0: Not set
1: Set

bit 5 Reserved: 0
bit 4 Reserved: 0

bit 3 **DR (DSR) signal status**
0: OFF
1: ON

bit 2 **Break sequence detected since COMSTAT is executed.**
0: Not detected
1: Detected

bit 1 Reserved: 0

bit 0 **CD signal status**
0: OFF
1: ON

Execution example

```
CALL COMSTAT("0:",A):PRINT BIN$(A)
```

The numeric data of the RS-232C port 0 status is assigned to the numeric type variable "A", and a binary expression of A is given as string type data.

COMTERM

Sets the MSX computer in the terminal mode.

FORMAT

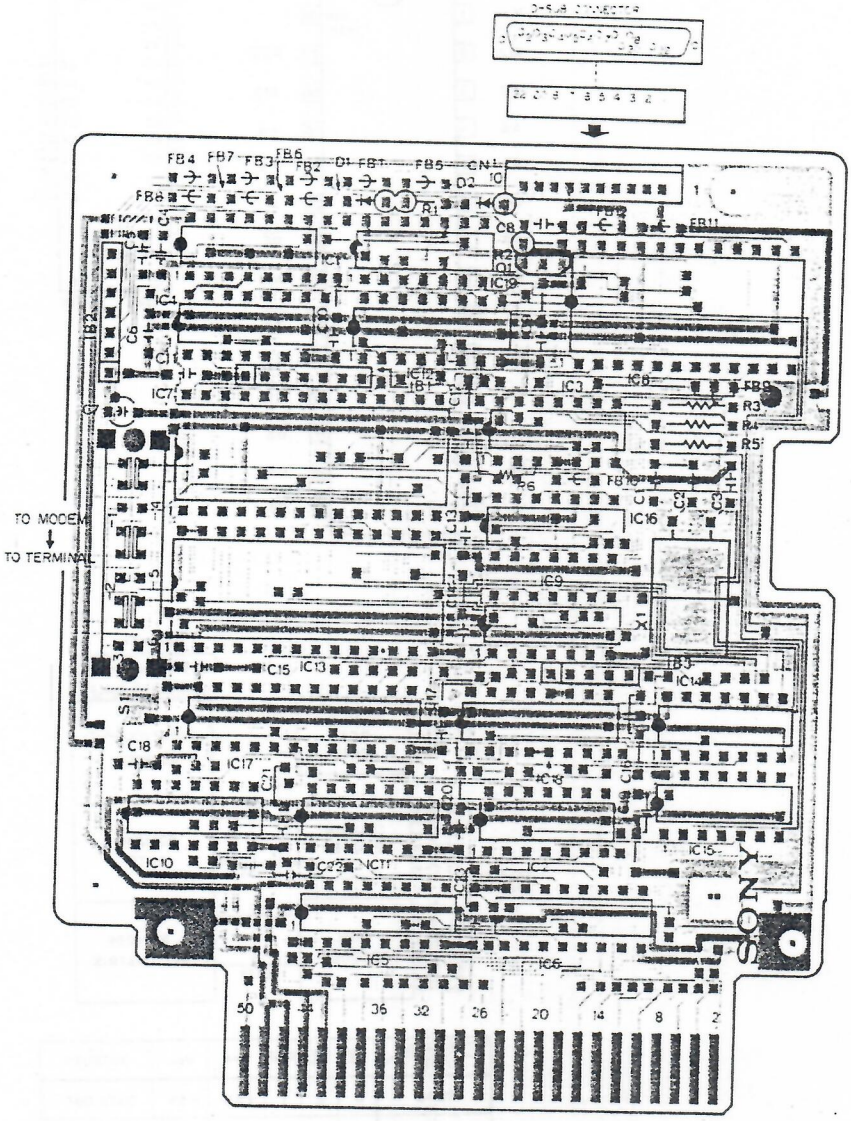
CALL COMTERM [("port number:")]

Port number Cond. Integer type constants,
0 ≦ port number ≦ 4
Omit 0

FUNCTION AND UTILIZATION

Enters a terminal emulator mode. Before entering the terminal mode, all the RS-232C files should be closed. The function keys have special use in the terminal mode. For details of the terminal mode and the usage of the function keys, read "Terminal Mode" on page 10.

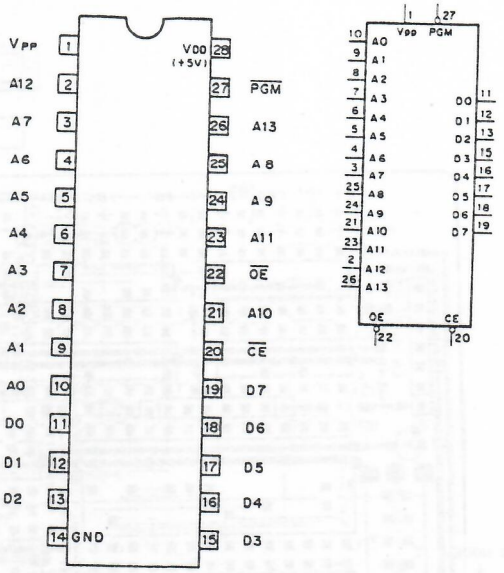
F-104 BOARD

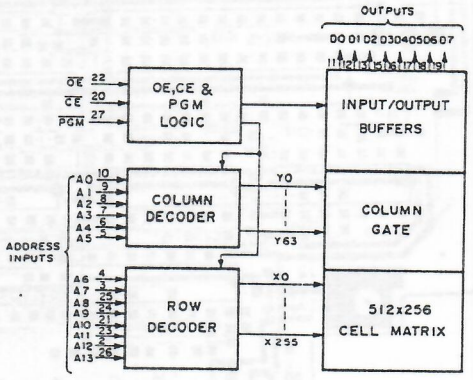


IF-104
COMPONENT SIDE
HBI-232 (EK)
HBI-232 (J)

SEMICONDUCTOR PIN ASSIGNMENTS

27128-RS232CHBI232
N-MOS UV EPROM 128K-BIT (16384x8)
— TOP VIEW —

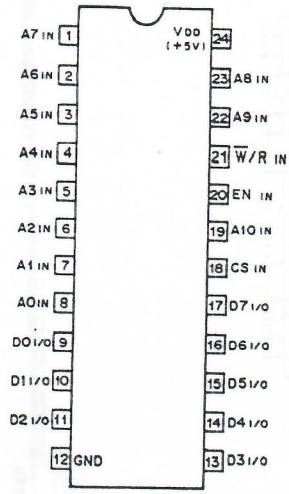


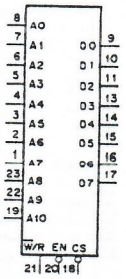


TERMINAL MODE	$\overline{CE}$	$\overline{OE}$	$\overline{PGM}$	V_{PP}	V_{CC}	OUTPUTS
READ	0	0	1	+5V	+5V	DATA OUT
STANDBY	1	X	X	+5V	+5V	HIGH IMPEDANCE
PROGRAMMING	0	1	0	+21V	+21V	DATA IN
PROGRAM VERIFY	0	0	1	+21V	+5V	DATA OUT
PROGRAM INHIBIT	1	X	X	+21V	+5V	HIGH IMPEDANCE
HIGH SPEED PROGRAMMING	0	1	0	+21V	+6V	DATA IN

1 ; TTL LEVEL HIGH VOLTAGE IN
0 ; TTL LEVEL LOW VOLTAGE IN
X ; DON'T CARE (1 OR 0)

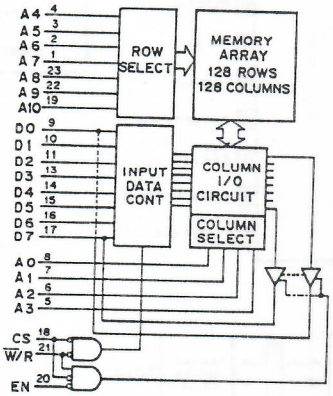
MB8416A-12P-SK (FUJITSU) (ACCESS TIME = 120 nS)
C-MOS 16384(2048x8)-BIT HIGH SPEED STATIC RAM
— TOP VIEW —





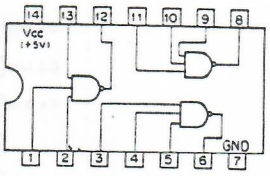
INPUTS			FUNCTION
CS	W/R	EN	
0	0	X	WRITE
0	1	0	READ
0	1	1	DISABLE (OUTPUT=HI-Z)
1	X	X	DISABLE (OUTPUT=HI-Z)

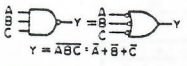
0 ; LOW LEVEL
1 ; HIGH LEVEL
X ; DON'T CARE
HI-Z ; HIGH IMPEDANCE



A10 ; ADDRESS INPUTS
; WRITE/READ ENABLE
EN ; OUTPUT ENABLE
CS ; CHIP SELECT
DO -D7 ; DATA INPUTS/OUTPUTS

SN74LS10N (TI)
TTL 3-INPUT POSITIVE NAND GATE
— TOP VIEW —

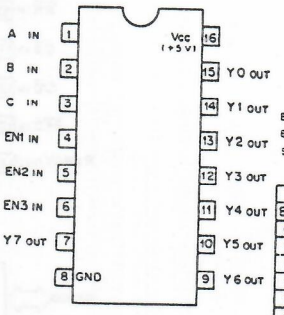


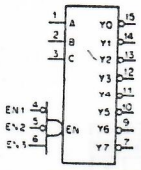


A	B	C	Y
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

0 : LOW LEVEL
1 : HIGH LEVEL

SN74LS138N (TI)
TTL 3-TO-8-LINE DECODER/DEMULTIPLEXER
— TOP VIEW —



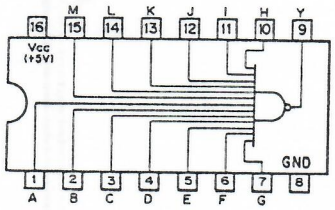


INPUTS				OUTPUTS							
EN	C	B	A	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
0	X	X	X	1	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1	1	1	1	0
1	0	0	1	1	1	1	1	1	1	0	1
1	0	1	0	1	1	1	1	1	0	1	1
1	0	1	1	1	1	1	1	0	1	1	1
1	1	0	0	1	1	1	0	1	1	1	1
1	1	0	1	1	1	0	1	1	1	1	1
1	1	1	0	1	0	1	1	1	1	1	1
1	1	1	1	0	1	1	1	1	1	1	1

EN = $\overline{EN1} \cdot \overline{EN2} \cdot EN3$

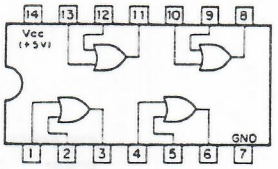
0 ; LOW LEVEL
1 ; HIGH LEVEL
X ; DON'T CARE

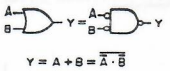
SN74ALS133N (TI)
TTL 13-INPUT NAND GATE
— TOP VIEW —



$Y = \overline{A \cdot B \cdot C \cdot D \cdot E \cdot F \cdot G \cdot H \cdot I \cdot J \cdot K \cdot L \cdot M}$

SN74LS32N (TI)
TTL 2-INPUT POSITIVE-OR GATE
— TOP VIEW —



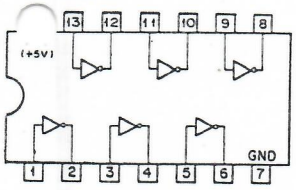


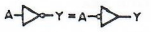
$Y = A + B = \overline{\overline{A} \cdot \overline{B}}$

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

0 : LOW LEVEL
1 : HIGH LEVEL

SN74LS04N (TI)
TTL INVERTER
— TOP VIEW —



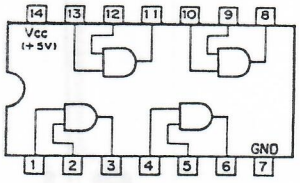


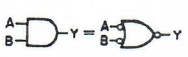
$Y = \overline{A}$

A	Y
0	1
1	0

0 : LOW LEVEL
1 : HIGH LEVEL

SN74LS08N (TI)
TTL 2-INPUT POSITIVE-AND GATE
— TOP VIEW —



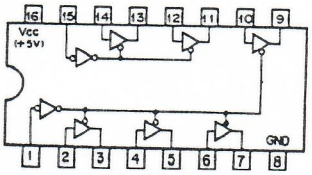


$Y = A \cdot B = \overline{\overline{A} + \overline{B}}$

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

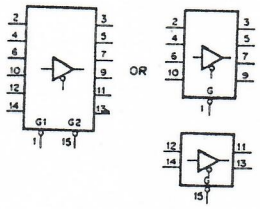
0 ; LOW LEVEL
1 ; HIGH LEVEL

SN74LS367AN (TI)
TTL BUS DRIVER WITH 3-STATE OUTPUTS
— TOP VIEW —

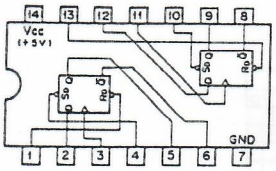


G	A	Y
0	0	0
0	1	1
1	X	HI-Z

0;LOW LEVEL
1;HIGH LEVEL
X;DON'T CARE
HI-Z;HIGH IMPEDANCE



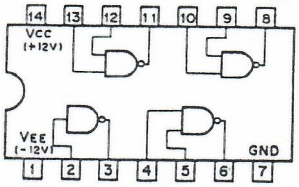
SN74LS74AN (TI)
TTL D-TYPE FLIP FLOP WITH DIRECT SET/RESET
— TOP VIEW —

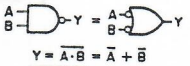


INPUTS				OUTPUTS	
S̄d	R̄d	CK	D	Qn+1	Q̄n+1
0	1	X	X	1	0
1	0	X	X	0	1
0	0	X	X	1*	1*
1	1	↑	1	1	0
1	1	↑	0	0	1
1	1	0	X	Qn	Q̄n

0;LOW LEVEL
1;HIGH LEVEL
X;DON'T CARE
1*;NONSTABLE

SN75188N (TI)
2-INPUT (1-INPUT) POSITIVE-NAND LINE DRIVER
— TOP VIEW —

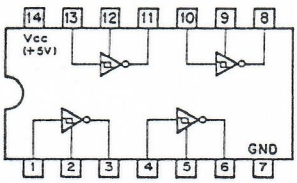




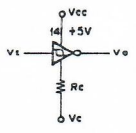
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

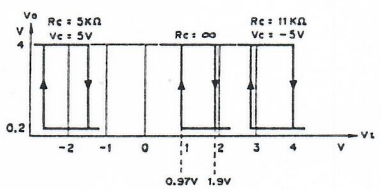
0;LOW LEVEL
1;HIGH LEVEL

SN75189AN (TI)
QUADRUPLE LINE RECEIVER
— TOP VIEW —

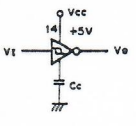


INPUT THRESHOLD SHIFTING

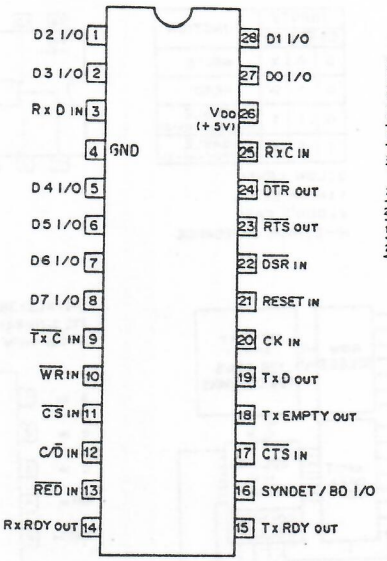


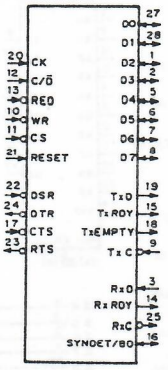


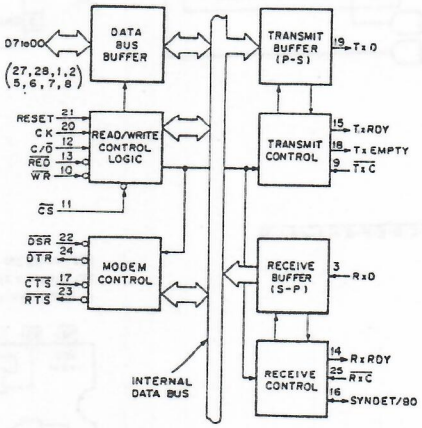
INPUT NOISE FILTERING



uPD8251AFC (NEC)
N-MOS PROGRAMMABLE COMMUNICATION INTERFACE
— TOP VIEW —



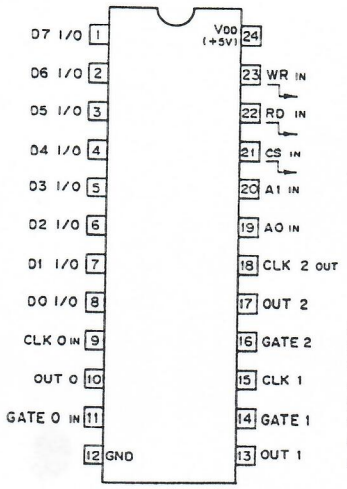


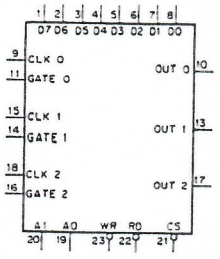


PIN NAMES

D0toD7	; DATA BUS
C/D̄	; CONTROL or DATA IS TO BE WRITTEN or READ
RED̄	; READ DATA COMMAND
WR̄	; WRITE DATA or CONTROL COMMAND
CS̄	; CHIP ENABLE
CK	; CLOCK PULSE
RESET	; RESET
TxC̄	; TRANSMITTER CLOCK
TxD	; TRANSMITTER DATA
RxC̄	; RECEIVER CLOCK
RxD	; RECEIVER DATA
RxRDY	; RECEIVER READY
TxRDY	; TRANSMITTER READY
DSR̄	; DATA SET READY
DTR̄	; DATA TERMINAL READY
SYNDET/BD	; SYNC DETECT/BREAK DETECT
RTS̄	; REQUEST TO SEND DATA
CTS̄	; CLEAR TO SEND DATA
TxEMPTY	; TRANSMITTER EMPTY

μPD8253C-5 (NEC)
N-MOS PROGRAMABLE INTERVAL TIMER
— TOP VIEW —



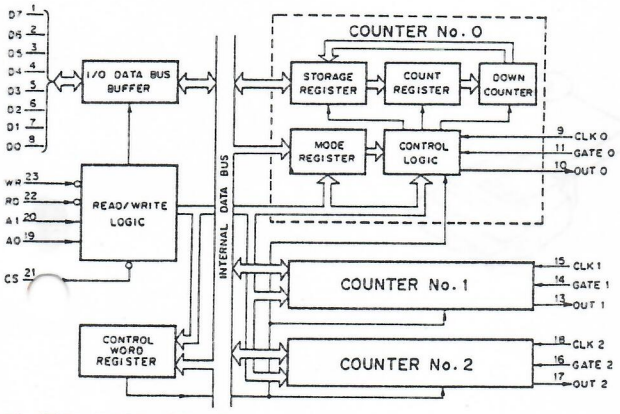


D7-D0 ; 8-BIT DATA I/O
CLK N ; COUNTER CLOCK
GATE N ; COUNTER GATE
OUT N ; COUNTER OUTPUT
RD ; READ COUNTER
WR ; WRITE CMD OR DATA
CS ; CHIP SELECT
A1, A0 ; COUNTER SELECT

FUNCTION TABLE

INPUTS					FUNCTION
CS	RD	WR	A1	A0	
0	1	0	0	0	Load Counter No. 0
0	1	0	0	1	Load Counter No. 1
0	1	0	1	0	Load Counter No. 2
0	1	0	1	1	Control Word
0	0	1	0	0	Read Counter 0
0	0	1	0	1	Read Counter 1
0	0	1	1	0	Read Counter 2
0	0	1	1	1	No-Operation (HI-Z)
1	X	X	X	X	Disable (HI-Z)
0	1	1	X	X	No-Operation (HI-Z)

0;LOW LEVEL
1;HIGH LEVEL
X;DON'T CARE
HI-Z;HIGH IMPEDANCE



CONTROL WORD FORMAT

D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	RL1	RL0	M2	M1	M0	BCD

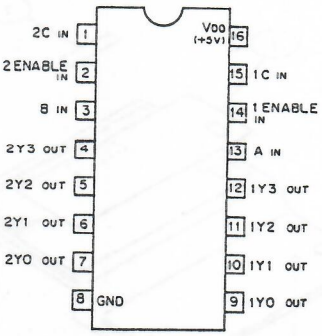
BCD	OPERATION
0	16-BIT BINARY
1	BCD(4-DECADE)

M2	M1	M0	MODE
0	0	0	0
0	0	1	1
X	1	0	2
X	1	1	3
1	0	0	4
1	0	1	5

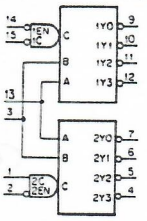
RL1	RL0	OPERATION
0	0	COUNTER LATCHING
0	1	READ/LOAD LSB ONLY
1	0	READ/LOAD MSB ONLY
1	1	LSB FIRST THEN MSB

SC1	SC0	SELECTED COUNTER
0	0	COUNTER No. 0
0	1	COUNTER No. 1
1	0	COUNTER No. 2
1	1	ILLEGAL

SN74LS156N (TI)
TTL DUAL 2-ILNE-TO-4-LINE DECODER/DEMULTIPLEXER (OPEN COLLECTOR OUTPUT)
— TOP VIEW —



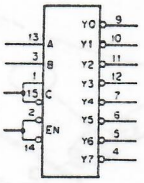
DUAL 2-LINE-TO-4-LINE DECODER
OR DUAL 1-LINE-TO-4-LINE DEMULTIPLEXER



INPUTS				OUTPUTS			
ENABLE	DATA	SELECT		1	1	1	1
1EN	1C	B	A	Y3	Y2	Y1	YO
0	0	0	0	1	1	1	0
0	0	0	1	1	1	0	1
0	0	1	0	1	0	1	1
0	0	1	1	0	1	1	1
X	1	X	X	1	1	1	1
1	X	X	X	1	1	1	1

INPUTS				OUTPUTS			
ENABLE	DATA	SELECT		2	2	2	2
2EN	2C	B	A	Y3	Y2	Y1	YO
0	1	0	0	1	1	1	0
0	1	0	1	1	1	0	1
0	1	1	0	1	0	1	1
0	1	1	1	0	1	1	1
X	0	X	X	1	1	1	1
1	X	X	X	1	1	1	1

3-LINE-TO-8-LINE DECODER
OR 1-LINE-TO-8-LINE DEMULTIPLEXER



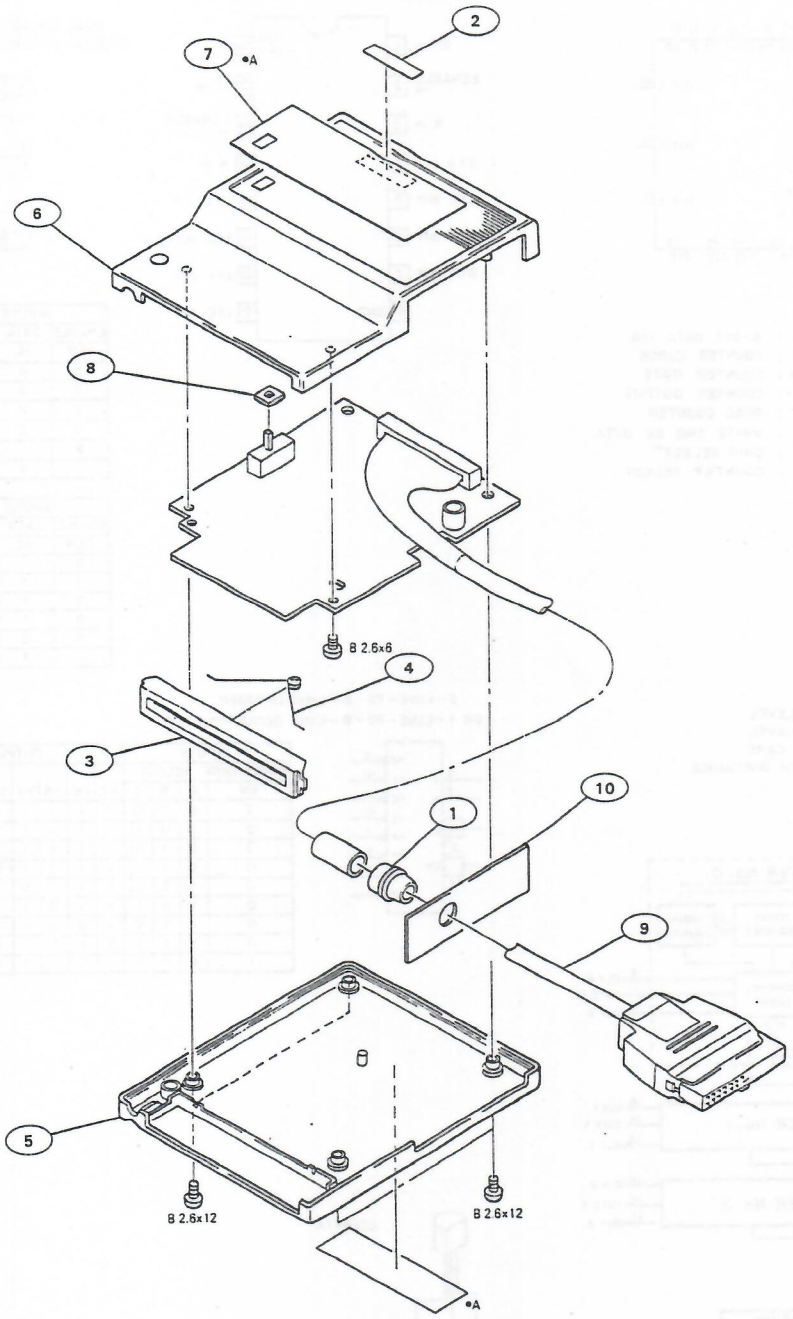
INPUTS				OUTPUTS							
ENABLE/DATA	SELECT										
EN	C	B	A	Y7	Y6	Y5	Y4	Y3	Y2	Y1	YO
0	0	0	0	1	1	1	1	1	1	1	0
0	0	0	1	1	1	1	1	1	1	0	1
0	0	1	0	1	1	1	1	1	0	1	1
0	0	1	1	1	1	1	1	0	1	1	1
0	1	0	0	1	1	1	0	1	1	1	1
0	1	0	1	1	1	0	1	1	1	1	1
0	1	1	0	1	0	1	1	1	1	1	1
0	1	1	1	0	1	1	1	1	1	1	1
1	X	X	X	1	1	1	1	1	1	1	1

2SC641K

1S1555

CHAPTER 4
REPAIR PARTS AND FIXTURE

4-1. EXPLODED VIEW



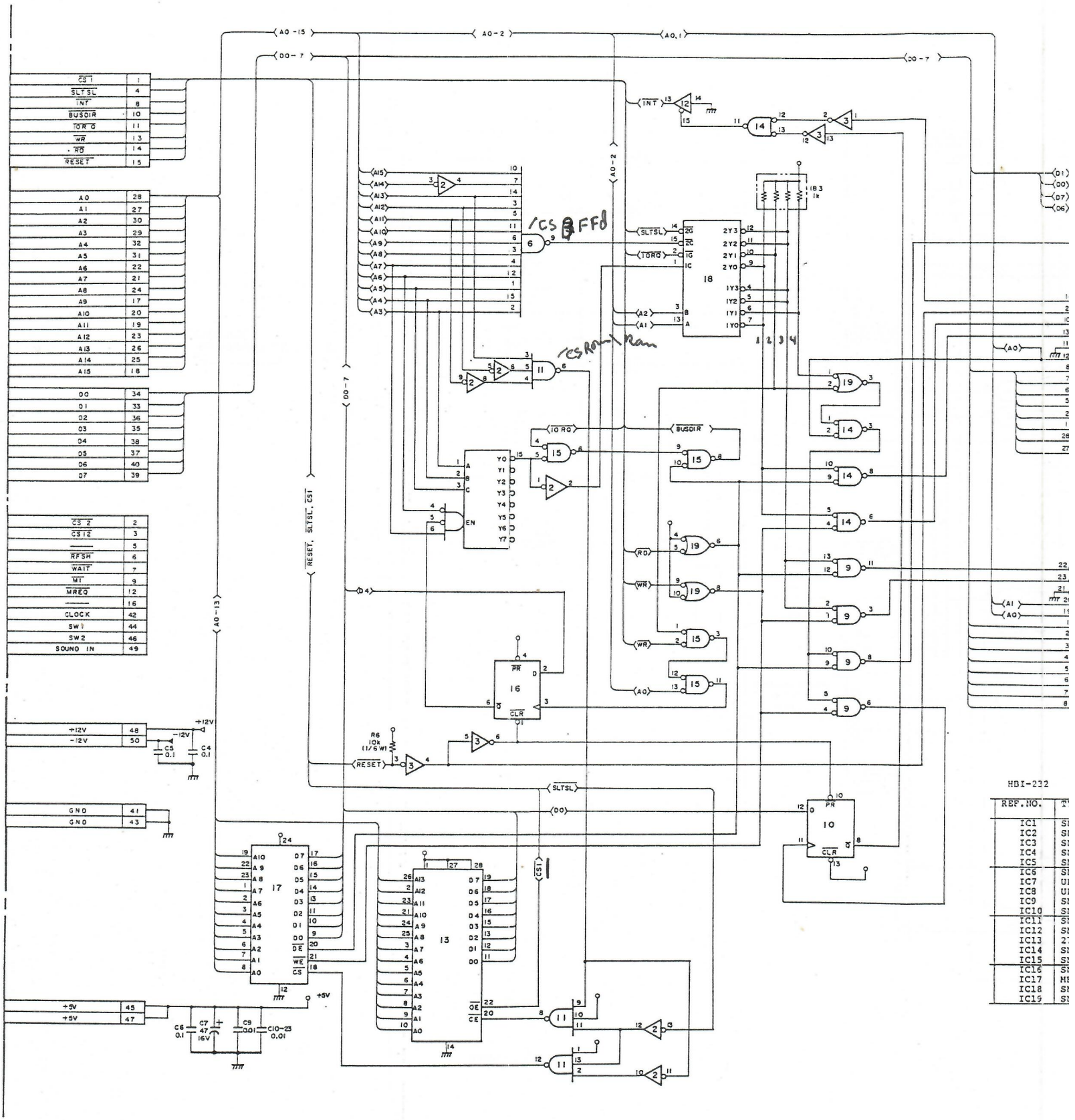
No.	Parts No.	Description
1	2-234-904-00	BUSH, CORD
2	3-701-690-00	LABEL (MADE IN JAPAN)
3	4-606-567-02	PROTECTOR
4	4-606-568-01	SPRING, TORSION
5	4-606-569-02	CASE (REAR), CARTRIDGE
6	4-606-570-02	CASE (FRONT), CARTRIDGE
7	4-609-310-01	LABEL, CARTRIDGE
8	4-608-657-01	COVER, SWITCH
9	1-558-396-11	CORD, CONNECTION
10	4-609-304-01	STOPPER, CABLE

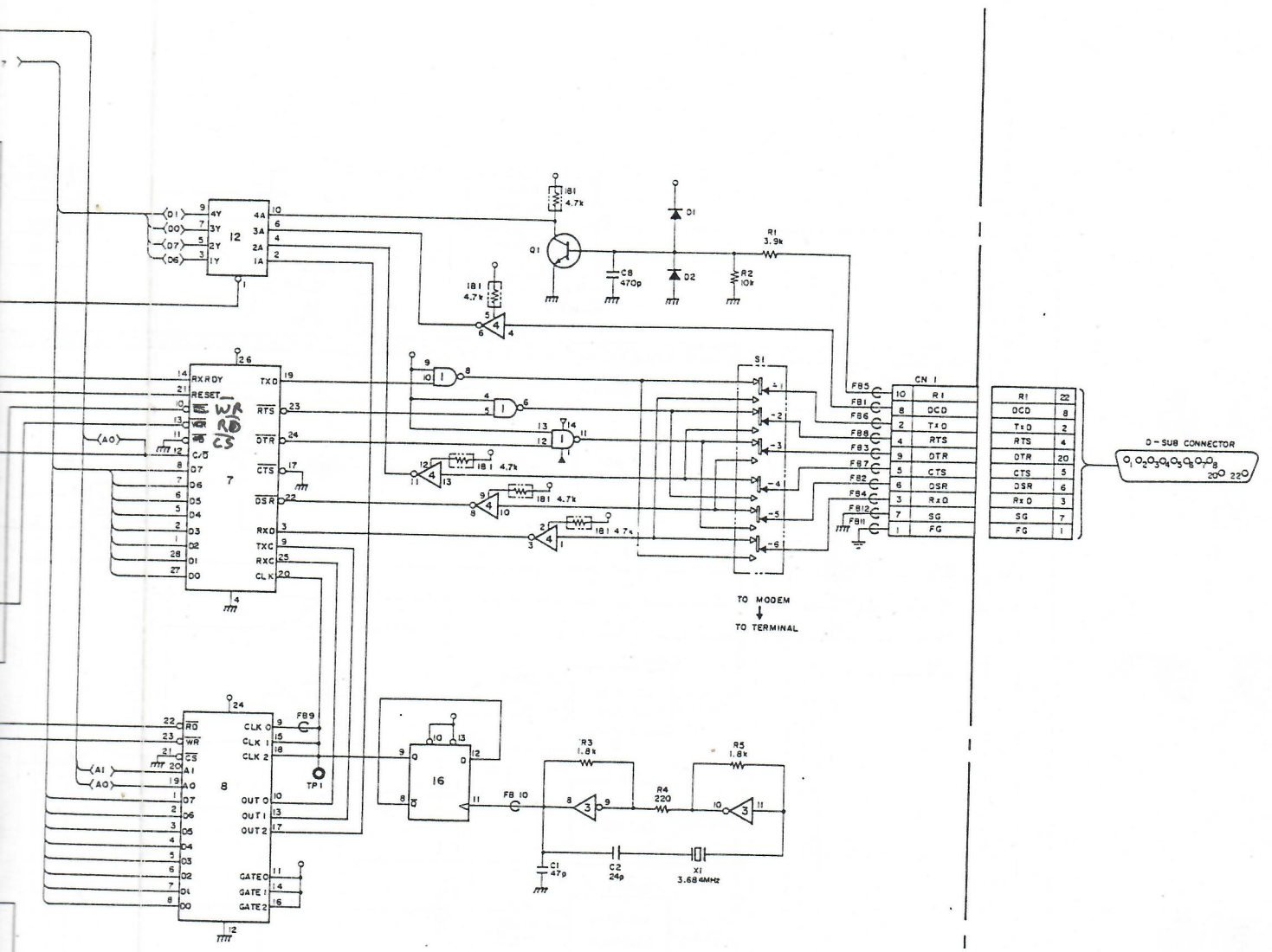
NOTE:

1. **The shaded and ⚠-marked components are critical to safety. Replace only with same components as specified.**
2. Parts printed in Bold-Face type are normally stocked for replacement purposes. The remaining parts shown in this manual are not normally required for routine service work. Orders for parts not shown in Bold-Face type will be processed, but allow for additional delivery time.
3. Item with no part number and/or no description are not stocked because they are seldom required for routine service.

CHAPTER 3
SCHEMATIC DIAGRAM AND PRINTED CIRCUIT BOARD

IF-104

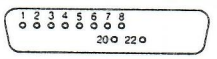




HDI-232

REF.NO.	TYPE NO.	PIN NO.			
		+12V	+5V	GND	-12V
IC1	SN75188N	14		7	1
IC2	SN74LS04N		14	7	
IC3	SN74LS04N		14	7	
IC4	SN75189AN		14	7	
IC5	SN74LS138N		16	8	
IC6	SN74ALS133N		16	8	
IC7	UPD8251AFC		26	4	
IC8	UPD8253C-5		24	12	
IC9	SN74LS32N		14	7	
IC10	SN74LS74AN		14	7	
IC11	SN74LS10N		14	7	
IC12	SN74LS367AN		16	8	
IC13	27128-RS232CHBI232		28	14	
IC14	SN74LS32N		14	7	
IC15	SN74LS32N		14	7	
IC16	SN74LS74AN		14	7	
IC17	MB8416A-12P-SK		24	12	
IC18	SN74LS156N		16	8	
IC19	SN74LS08N		14	7	

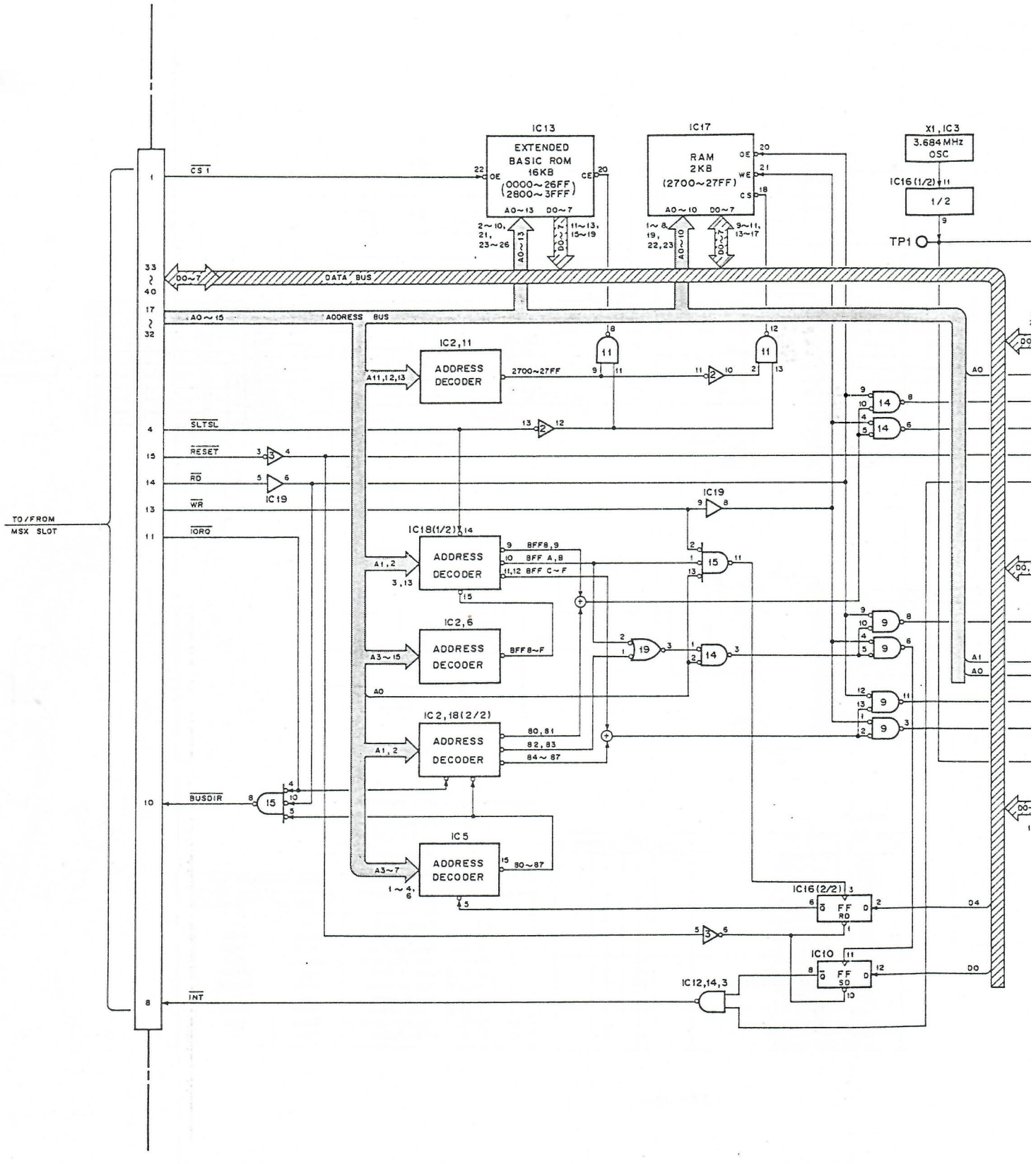
Interface connector pin assignment

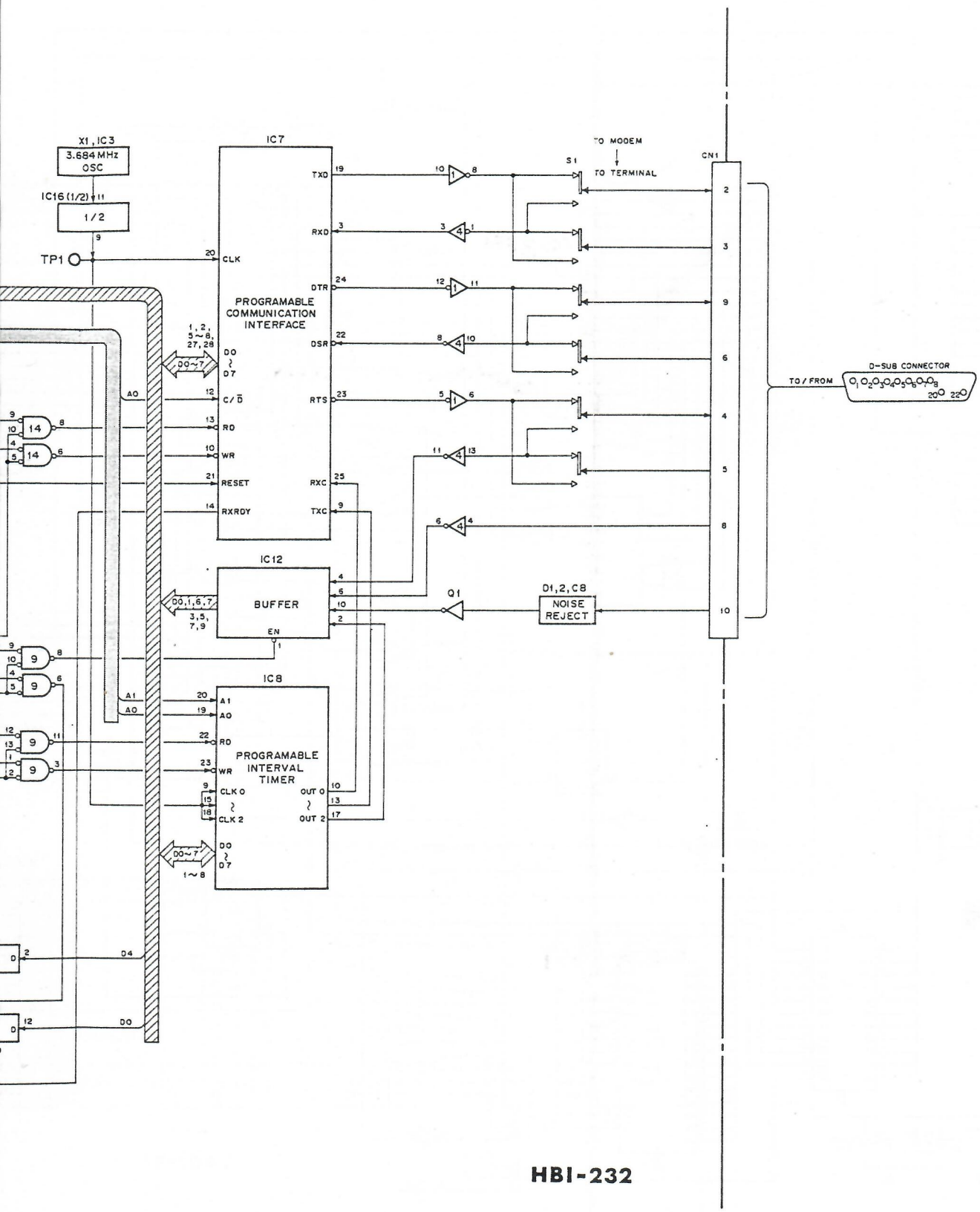


Pin No.	Symbol and description	Signal flow direction with signal direction select switch	
		TO MODEM	TO TERMINAL
1	FG Frame ground	—	—
2	SD (T × D) Transmit data	Output	Input
3	RD (R × D) Received data	Input	Output
4	RS (RTS) Request to send	Output	Input
5	CS (CTS) Clear to send	Input	Output
6	DR (DSR) Data set ready	Input	Output
7	SG Signal ground	—	—
8	CD (DCD) Carrier detection	Input	Output
20	ER (DTR) Data terminal ready	Output	Input
22	CI (RI) Ring indicator	Input	—

IF-104

CHAPTER 2
BLOCK DIAGRAM




X1, IC3
3.684 MHz
OSC
IC16 (1/2)
1/2
TP1
IC7
PROGRAMABLE
COMMUNICATION
INTERFACE
CLK
TXD
RXD
DTR
DSR
RTS
RXC
TXC
RESET
RXRDY
C/D
RD
WR
D0
~
D7
A0
TO MODEM
TO TERMINAL
S1
CN1
D-SUB CONNECTOR
TO / FROM
IC12
BUFFER
EN
Q1
D1,2,C8
NOISE
REJECT
IC8
PROGRAMABLE
INTERVAL
TIMER
A1
CLK 0
CLK 2
OUT 0
OUT 2
D4
D0

HBI-232

4-2. ELECTRICAL PARTS LIST

f. No.	Parts No.	Description
IF-104 Board		
C1	1-102-852-00	CERAMIC 47PF 5% 50V
C2	1-102-515-00	CERAMIC 24PF 5% 50V
C4	1-162-561-11	CERAMIC 0.1 16V
C5	1-162-561-11	CERAMIC 0.1 16V
C6	1-162-561-11	CERAMIC 0.1 16V
C7	1-124-236-00	ELECT 47 20% 16V
C8	1-102-114-00	CERAMIC 470PF 10% 50V
C9	1-161-330-00	CERAMIC 0.01 30% 25V
C10	1-161-330-00	CERAMIC 0.01 30% 25V
C11	1-161-330-00	CERAMIC 0.01 30% 25V
C12	1-161-330-00	CERAMIC 0.01 30% 25V
C13	1-161-330-00	CERAMIC 0.01 30% 25V
C14	1-161-330-00	CERAMIC 0.01 30% 25V
C15	1-161-330-00	CERAMIC 0.01 30% 25V
C16	1-161-330-00	CERAMIC 0.01 30% 25V
C17	1-161-330-00	CERAMIC 0.01 30% 25V
C18	1-161-330-00	CERAMIC 0.01 30% 25V
C19	1-161-330-00	CERAMIC 0.01 30% 25V
C20	1-161-330-00	CERAMIC 0.01 30% 25V
C21	1-161-330-00	CERAMIC 0.01 30% 25V
C22	1-161-330-00	CERAMIC 0.01 30% 25V
C23	1-161-330-00	CERAMIC 0.01 30% 25V
CN1	1-564-009-00	PIN, CONNECTOR 10P
D1	8-719-815-55	1S1555
D2	8-719-815-55	1S1555
FB1	1-543-255-11	BEAD, FERRITE
FB2	1-543-255-11	BEAD, FERRITE
FB3	1-543-255-11	BEAD, FERRITE
FB4	1-543-255-11	BEAD, FERRITE
FB5	1-543-255-11	BEAD, FERRITE
FB6	1-543-255-11	BEAD, FERRITE
FB7	1-543-255-11	BEAD, FERRITE
FB8	1-543-255-11	BEAD, FERRITE
FB9	1-543-255-11	BEAD, FERRITE
FB10	1-543-255-11	BEAD, FERRITE
FB11	1-543-255-11	BEAD, FERRITE
FB12	1-543-255-11	BEAD, FERRITE

Ref. No.	Parts No.	Description
IC1	8-759-951-88	SN75188N
IC2	8-759-900-04	SN74LS04N
IC3	8-759-900-04	SN74LS04N
IC4	8-759-951-89	SN75189AN
IC5	8-759-901-38	SN74LS138N
IC6	8-759-904-53	SN74ALS133N
IC7	8-759-103-27	μPD8251AFC
IC8	8-759-182-53	μPD8253C-5
IC9	8-759-900-32	SN74LS32N
IC10	8-759-900-74	SN74LS74AN
IC11	8-759-900-10	SN74LS10N
IC12	8-759-903-67	SN74LS367AN
IC13	8-759-767-64	27128-RS232CHB1232
IC14	8-759-900-32	SN74LS32N
IC15	8-759-900-32	SN74LS32N
IC16	8-759-900-74	SN74LS74AN
IC17	8-759-911-92	MB8416A-12P-SK
IC18	8-759-901-56	SN74LS156N
IC19	8-759-900-08	SN74LS08N
Q1	8-729-364-12	2SC641K
R1	1-247-145-00	CARBON 3.9K 5% 1/4W
R2	1-247-725-11	CARBON 10K 5% 1/4W
R3	1-247-137-00	CARBON 1.8K 5% 1/4W
R4	1-247-704-11	CARBON 220 5% 1/4W
R5	1-247-137-00	CARBON 1.8K 5% 1/4W
R6	1-247-855-00	CARBON 10K 5% 1/6W
S1	1-554-949-11	SWITCH, SLIDE
X1	1-567-483-11	VIBRATOR, CRYSTAL 3.684MHz

4-3. PACKING MATERIAL AND ACCESSORY

No.	Parts No.	Description
	X-4604-452-1	SCREW ASSY, CONNECTOR
	3-701-625-00	BAG, POLYETHYLENE
	3-760-996-11	**MANUAL, INSTRUCTION**
	3-764-312-12	**MANUAL, RS-232C**
	4-609-359-11	INDIVIDUAL CARTON
	4-608-630-01	CUSHION
	4-608-631-01	SPACER

NOTE:

1. **The shaded and ⚠-marked components are critical to safety.**
   Replace only with same components as specified.

2. Parts printed in **Bold-Face type** are normally stocked for replacement purposes. The remaining parts shown in this manual are not normally required for routine service work. Orders for parts not shown in Bold-Face type will be processed, but allow for additional delivery time.

HBI-232 (EK)
9-975-618-01

Sony Corporation

Printed in Japan
1986.2 05