HBI-232

SERVICE MANUAL

SPECIFICATIONS

Communication mode
Baud rate
Selectable by software
50, 75, 110, 300, 600, 1200, 1800, 2000, 2400,
38400, 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 x 132 x 26.4 mm (w/h/d)
(4½ x 5½ x 1⅞ inches)
Weight
Approx. 270 g (9.5 oz)

RS-232C INTERFACE CARTRIDGE

SONY®

Scanned and converted to PDF by HansO, 2001
# TABLE OF CONTENTS

1. **EXPLANATION**
   - 1-1. WHAT IS THE RS-232C? ........................................... 3
   - 1-2. SYSTEM CONFIGURATION ........................................ 3
   - 1-3. PREPARATION .................................................... 4
   - 1-3-1. Switching the Signal Flow ................................ 5
   - 1-3-2. Setting the RS-232C Data Format and Communication Mode .................................................... 6
2. **PRACTICING RS-232C SERIAL DATA COMMUNICATION** ........................................... 7
   - 2-1-1. Data and Program Communication ........................................... 7
   - 2-2. TERMINAL OPERATION ............................................. 10
3. **COMMANDS AND FUNCTIONS OF THE MSX-BASIC FOR RS-232C COMMUNICATION** ........................................... 11
   - 3-1-1. Introductory Remarks ........................................ 11
   - 3-1-2. List of Commands and Functions for RS-232C ........................................... 11
   - 3-1-3. MSX-BASIC Commands and Functions ............................. 12

2. **BLOCK DIAGRAM** ............................................. 21

3. **SCHEMATIC DIAGRAM AND PRINTED CIRCUIT BOARD**
   - IF-104 ........................................................................ 23
   - IF-104 BOARD ............................................................. 25
   - SEMICONDUCTOR PIN ASSIGNMENTS .................................. 26

4. **REPAIR PARTS AND FIXTURE**
   - 4-1. EXPLODED VIEW .................................................. 30
   - 4-2. ELECTRICAL PARTS LIST ........................................ 31
   - 4-3. PACKING MATERIAL AND ACCESSORY ............................ 31
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 floppy disk, 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 these jobs 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.
3. Connect to the RS-232C connector on a modem or a terminal.

1. Turn off computer.

2. Signal direction select switch

3. Insert into a cartridge slot.

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 MODEM

TO TERMINAL

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 5MK-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! *

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."

* 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:

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

To connect two terminal equipments (computers)

<table>
<thead>
<tr>
<th>TERMINAL (computer)</th>
<th>TERMINAL (computer)</th>
<th>TERMINAL (computer)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 SD</td>
<td>2 SD</td>
<td>2 SD</td>
</tr>
<tr>
<td>3 RD</td>
<td>3 RD</td>
<td>3 RD</td>
</tr>
<tr>
<td>4 RS</td>
<td>4 RS</td>
<td>4 RS</td>
</tr>
<tr>
<td>5 CS</td>
<td>5 CS</td>
<td>5 CS</td>
</tr>
<tr>
<td>6 DR</td>
<td>6 DR</td>
<td>6 DR</td>
</tr>
<tr>
<td>20 ER</td>
<td>20 ER</td>
<td>20 ER</td>
</tr>
</tbody>
</table>

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

A null modem cable required.

Interface connector pin assignment

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.
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 asynchronous mode which is often employed in the MSX computer.

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

```
[Start bit] [Data (character "S")] [Parity bit (0 or 1)] [Stop bit(s)]
```

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

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: Used for special purposes such as domestic or international telex for example.
- 6 bits: Used only for ASCII code.
- 7 bits: Used to exchange a program in machine language, or codes of 60H to FFH.

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

<table>
<thead>
<tr>
<th>Character data = &quot;A&quot; (41H)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data length = 8 bits</td>
</tr>
<tr>
<td>Even parity</td>
</tr>
</tbody>
</table>

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

```
<Transmitter> 0 01000001 1 Parity bit <Receiver> 00110001
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.

Stop bit length

The stop bits are 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 length 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 bits) 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, 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.
2-1 PRACTICING RS-232C SERIAL DATA COMMUNICATION

There are mainly two ways of using serial data communication for your MSX computer: data exchange with another computer or operating the MSX computer as a terminal of the other host computer. When 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/receiving 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>

```
OPEN

PRINT# or INPUT#

CLOSE
```

Opens a file used as a transmit or receive buffer where data is written for transmission or data is received. <PAGE 14>

Writes data to the opened RS-232C transmit buffer file, or reads data from the opened RS-232C receive buffer file, and assigns it to a variable. <PAGE 15 or 13>

Closes the receive/transmit buffer file opened by the OPEN statement to declare the end of the RS-232C serial data communication. <PAGE 12>

Note

The signals status will be as follows:

- RS signal: OFF when the CLOSE statement is executed.
- ER signal: ON when the computer is turned on.
- In case CS-RS handshake control is activated by the MSX-BASIC command COMIN, 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 another 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 another 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.

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 (2)
   - 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)
   - Transmission/receive speed: 300 bps, 1300, 3000
   - 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 SS so that value set in SS will be cleared.
8. Converts numeric type data assigned to the variables A (0), A (1), A (2) to the string type variable SS.
9. Writes data assigned to the SS 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("COM1",48000,51)   1
20 OPEN "COM1" FOR OUTPUT AS #1  2
30 PRINT: PRINT "**** Now waiting data ****" 3
40 LINE INPUT #1,$  4
50 IF $="Start" THEN 60 ELSE 10  5
60 INPUT #1,A,B,C  6
70 PRINT "A=":A  7
80 PRINT "B=":B  8
90 PRINT "C=":C  9
100 D=A+B+C 10
110 PRINT "A+B+C";D 11
120 LINE INPUT #1,$  12
130 IF $="End" THEN 70 ELSE 130 13
140 END  14
```

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 displays 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.
Examples 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 computer B. Any desired file on the source floppydisk can be specified for transmission by inputting its file name from the keyboard.

<Program for MSX computer A as a transmitter>

```plaintext
10 REM FILES=2
20 INPUT "Input name of file to send:";F$  
30 PRINT "###Now sending data ###"  
40 CALL COMINT(0,"sinx13xxxx",300,5)  
50 OPEN "COM1:" FOR OUTPUT AS #1  
60 OPEN #1 FOR INPUT AS #2  
70 PRINT "Start":PRINT #1,0$  
80 IF EOF(2) THEN GOTO 120  
90 LINE INPUT #1,S$  
100 PRINT #1,S$  
110 GOTO 80  
120 IF S$="End" THEN BUFS=15 ELSE 160  
130 PRNT BUFS$  
140 BUFS  
150 GOTO 90  
160 PRINT "###Now saving data ###"  
170 W$=1  
180 FOR I=1 TO W$  
190 PRINT #2, BUFS$  
200 NEXT  
210 CLOSE  
220 END
```

The step above the program means:
1. Declares the number of files that can be simultaneously opened in this program to 2.
2. Prompts the user to input the name of the file to be transmitted from the keyboard. The input file name will be assigned to the variable F$.
3. Displays the message "### Now sending data ###" on the monitor screen.
4. Initializes the RS-232C port numbered 0 so that the data format and communication modes settings will be performed as follows:
   - RS-232C port number: 0 (D)
   - Data length: 8 bits (B)
   - Parity check: No parity check (N)
   - Stop bit: 2 bits (D)
   - XON/XOFF control: Enables control (X)
   - CS-RS handshaking: No handshaking (N)
   - Automatic line feed insert/delete: No insert/delete (NN)
   - Shift-in/shift-out control: No control (N)
   - Transmits/receive speed: 300 bps (300,300)
   - Time out: 5 seconds (5)
5. Opens the RS-232C file for the transmit buffer as the file number 1.
6. Opens the data file "F$" 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 EDF 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 BUFS. Writes data assigned to the BUFS to the transmit buffer file 1 so that the data will be sent to the MSX computer B.
9. If the EDF send-of-files 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>

```plaintext
10 REM FILES=2
20 CLEAR 2000  
30 DIM BUF$(500)  
40 INPUT "Input file name to save:";F$  
50 CALL COMINT("sinx13xxx",300,300,5)  
60 OPEN "COM1:" FOR INPUT AS #1  
70 OPEN #1 FOR OUTPUT AS #2  
80 IF S$="End" THEN BUFS=15 ELSE 160  
90 IF S$="Start" THEN GOTO 100 ELSE BO  
100 N$=1  
110 LINE INPUT #1,S$  
120 IF S$="End" THEN BUFS=15 ELSE 160  
130 PRINT BUFS$  
140 N$=1  
150 GOTO 190  
160 PRINT "###Now saving data ###"  
170 N$=1  
180 FOR I=1 TO N$  
190 PRINT #2, BUFS$  
200 NEXT  
210 CLOSE  
220 END
```

The step above the program means:
1. Declares the number of files simultaneously opened in this program to 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 BUFS (0) to BUFS (500).
4. Prompts the user to input the name of the file to save the received data from the keyboard.
5. 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.
7. Waits for the message "Start" to be sent from the transmitter.
8. Upon receipt of the message "Start", starts receiving character strings one by one until a 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("0:893XNNNN",1200,1200,5)
SAVE "COM1:"

MSX computer B (receiver)
CALL COMINI("0:893XNNNN",1200,1200,51)
LOAD "COM1:"

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 described 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 sequence1 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] + [F5]
Displays the received control codes (00H to 1FH) by "\^\n" and the character assigned to the control code plus 40H. (ex.) The return code (0DH) will be displayed as follows:

\nM 4D

To exit from this function mode, press the [SHIFT] and [F5] keys simultaneously again.

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

[SHIFT] + [F3]
Displays the data input from the keyboard on the screen. To exit from this function mode, press the [SHIFT] and [F3] keys simultaneously again.

[SHIFT] + [F5]
Displays and prints out the data input from the keyboard at the same time. To exit from this function mode, press the [SHIFT] and [F5] 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 [X] key on the MSX computers such as HB-10P.
3-1-2 LIST OF COMMANDS AND FUNCTIONS FOR RS-232C

<table>
<thead>
<tr>
<th>Command</th>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPEN</td>
<td>Opens an RS-232C file.</td>
<td>14</td>
</tr>
<tr>
<td>CLOSE</td>
<td>Closes the file opened by an OPEN statement.</td>
<td>12</td>
</tr>
<tr>
<td>PRINT</td>
<td>Writes data to a transmit buffer file</td>
<td>15</td>
</tr>
<tr>
<td>PRINT#</td>
<td>Writes data to a transmit buffer file in a specified format.</td>
<td>15</td>
</tr>
<tr>
<td>INPUT</td>
<td>Reads data from a receive buffer file, and assigns it to a variable</td>
<td>13</td>
</tr>
<tr>
<td>INPUT#</td>
<td>Reads a string from a receive buffer file, and assigns it to a variable</td>
<td>13</td>
</tr>
<tr>
<td>SAVE</td>
<td>Sends a BASIC program to the RS-232C port.</td>
<td>17</td>
</tr>
<tr>
<td>LOAD</td>
<td>Loads a BASIC program from the RS-232C port.</td>
<td>13</td>
</tr>
</tbody>
</table>

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

The RS-232C port number is specified to the RS-232C interface as follows;
0: RS-232C interface on the MSX computer
1: 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.

<table>
<thead>
<tr>
<th>Command</th>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>RUN</td>
<td>Loads a program from the RS-232C port and executes the program.</td>
<td>17</td>
</tr>
<tr>
<td>MERGE</td>
<td>Loads a program in ASCII format from the RS-232C port, and merge it into the program currently in memory.</td>
<td>14</td>
</tr>
<tr>
<td>EOF</td>
<td>When the last data of a file has been read, 1 is given, otherwise 0 is given.</td>
<td>12</td>
</tr>
<tr>
<td>INPUTS</td>
<td>Inputs a specified number of characters from the receive buffer file.</td>
<td>12</td>
</tr>
<tr>
<td>LOC</td>
<td>Returns the number of characters in the receive buffer file.</td>
<td>14</td>
</tr>
<tr>
<td>LOF</td>
<td>Returns the free space remaining in the receive buffer file.</td>
<td>14</td>
</tr>
</tbody>
</table>
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: Integer constants, variables, array variables, their expressions, their maximum file number specified by MAXFILE = statement

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 IEOF** (end of file)

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

**FORMAT**

IEOF (file number)

File number $\text{FILE}$ integer constants, variables, array variables, their expressions, their maximum file number specified by MAXFILE = 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 IEOF(1) THEN CLEAR #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 INPUTS** (input dollar)

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

**FORMAT**

INPUTS [X, [FILE]] file number

X $\text{FILE}$ numeric type constants, variables, array variables, their expression, their maximum file number specified by MAXFILE = statement

**FUNCTION AND UTILIZATION**

Reads number of characters (string type data) specified by X from the RS-232C receive buffer 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 "COM1" FOR INPUT AS #1
20 INPUTS(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.
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 assigned 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 "COM1:*" FOR INPUT AS #1
20 IF EOF(1) THEN GOTO 50
30 LINE INPUT #1, A$.
40 PRINT A$
50 GOTO 30
60 CLOSE #1:END

Opens an RS-232C receive buffer file with the file number 1, reads string data from the file, and assigns the data to the string type variable AS. 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.

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 (IAH), the program loading will end.

Execution example
LOAD "COM1:".R

FORMAT
LOAD "COM [port number]:" ["LR"]
Port number Consta Integer type constants,
0 ≤ port number ≤ 4
Corta Loading the program only

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

FILE INPUT# (input number)
Reads data from a receive buffer file, and assigns it to a variable.

FORMAT
FILE INPUT# file number, variable [variable]....
File number [integer] integer constants,
1 ≤ file number ≤ the number specified by MAXFILES = statement
Variable [string] String type variables, 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 those 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.) "ABCDEF". . . . . . . A$ (string type variable)
1,2,3,4,5. . . . . . . . . A% (numeric type variable)

Execution example
10 OPEN "COM1:*" 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 AS 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 [integer] integer constants,
1 ≤ file number ≤ the number specified by MAXFILES = statement
Variable [string] 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 assigned 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 "COM1:*" FOR INPUT AS #1
20 IF EOF(1) THEN GOTO 50
30 LINE INPUT #1, A$
40 PRINT A$
50 GOTO 30
60 CLOSE #1:END

Opens an RS-232C receive buffer file with the file number 1, reads string data from the file, and assigns the data to the string type variable AS. The contents of the data is displayed on the screen. If end of data character is received, the file numbered 1 is closed.
**Function: LOC (location)**

Returns the number of characters in the receive buffer file.

**FORMAT**

LOC (file number)

- **File number**: Integer constant, variables, array variables, their expressions, 1 ≤ file number ≤ the number specified in MAXFILES = statement

**FUNCTION AND UTILIZATION**

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

**Function: LOF (length of file)**

Returns the free space remaining in the receive buffer file.

**FORMAT**

LOF (file number)

- **File number**: Integer constant, 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)**

Loading 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**: Integer constant, 0 ≤ port number ≤ 4
- **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**: Integer type constants, 0 ≤ file number ≤ 4
- **Mode**: OUTPUT, INPUT
- **File number**: 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-322C transmit buffer file.

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

File number: integer constants. 1 ≤ file number ≤ the number specified by MAXFILES statement
Expression: string type and numeric type constants, variables, array variables, their expressions
Separator: 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 with 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 "CON:1" FOR OUTPUT AS #1
20 A$="ABC"; B$="DEF"
30 PRINT #1; A$; B$
40 PRINT #1; A$; B$
50 PRINT #1; A$; B$
60 CLOSE #1

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

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Expression format and Execution example</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;!&quot;</td>
<td>Outputs the first 1 character.</td>
</tr>
<tr>
<td></td>
<td>PRINT # USING &quot;!&quot; &quot;United&quot;; &quot;Nation&quot;</td>
</tr>
<tr>
<td></td>
<td>Data to be transmitted = &quot;!&quot;</td>
</tr>
</tbody>
</table>

| "!"    | Outputs n + 2 characters. When data is smaller than n + 2 characters, inserts spaces for the residual characters. |
|        | PRINT # USING "!" "ABCDEF"; "GHJ"; "KLM" |
|        | Data to be transmitted = ABCDEF GHJ KLM |

| "&"    | Outputs all character string.          |
|        | 10 OPEN "CON:1" 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 ".".

\[
\text{PRINT} \#1 \text{ USING "POINT:##.##";123.4}
\]
\[
\text{Data to be transmitted} \rightarrow \text{POINT:123.4}
\]

● 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.

\[
10 \text{ OPEN } \text{"CON:1" FOR OUTPUT AS } \#1
20 \text{ PRINT } \#1 \text{ USING "##";123}
30 \text{ PRINT } \#1 \text{ USING "##";12345}
40 \text{ CLOSE } \#1
\]
\[
\text{Data to be transmitted} \rightarrow \text{## \hspace{2.5cm} #12345}
\]

● 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.

\[
10 \text{ OPEN } \text{"CON:1" FOR OUTPUT AS } \#1
20 \text{ PRINT } \#1 \text{ USING "###.###";25.3}
30 \text{ PRINT } \#1 \text{ USING "###.###";25.345}
40 \text{ CLOSE } \#1
\]
\[
\text{Data to be transmitted} \rightarrow \text{25.30 \hspace{2.5cm} Line Feed Code}
\]

The "+" sign of numeric data is ignored and the "-" sign is counted as one digit.

\[
10 \text{ OPEN } \text{"CON:1" FOR OUTPUT AS } \#1
20 \text{ PRINT } \#1 \text{ USING "###";123}
30 \text{ PRINT } \#1 \text{ USING "###";12345}
40 \text{ CLOSE } \#1
\]
\[
\text{Data to be transmitted} \rightarrow \text{123 \hspace{2.5cm} Line Feed Code}
\]

"+" "+" is added if it is a positive numeral, and "-" is added if it is a negative numeral before or after the numeric data.

\[
10 \text{ OPEN } \text{"CON:1" FOR OUTPUT AS } \#1
20 \text{ PRINT } \#1 \text{ USING "###";123}
30 \text{ PRINT } \#1 \text{ USING "###";12345}
40 \text{ CLOSE } \#1
\]
\[
\text{Data to be transmitted} \rightarrow \text{123 \hspace{2.5cm} Line Feed Code}
\]

"-" "-" is added after negative numeric data.

\[
10 \text{ OPEN } \text{"CON:1" FOR OUTPUT AS } \#1
20 \text{ PRINT } \#1 \text{ USING "###";123}
30 \text{ PRINT } \#1 \text{ USING "###";12345}
40 \text{ CLOSE } \#1
\]
\[
\text{Data to be transmitted} \rightarrow \text{123 \hspace{2.5cm} Line Feed Code}
\]

"*" The space before numeric data is filled with "*": One "*" in the format expresses one digit.

\[
10 \text{ OPEN } \text{"CON:1" FOR OUTPUT AS } \#1
20 \text{ PRINT } \#1 \text{ USING "*";123}
30 \text{ PRINT } \#1 \text{ USING "*";12345}
40 \text{ CLOSE } \#1
\]
\[
\text{Data to be transmitted} \rightarrow \text{*123 \hspace{2.5cm} Line Feed Code}
\]

"**" Adds "E" before numeric data: One "E" in the format is counted as one digit.

\[
10 \text{ OPEN } \text{"CON:1" FOR OUTPUT AS } \#1
20 \text{ PRINT } \#1 \text{ USING "**";123}
30 \text{ PRINT } \#1 \text{ USING "**";12345}
40 \text{ CLOSE } \#1
\]
\[
\text{Data to be transmitted} \rightarrow \text{123 \hspace{2.5cm} Line Feed Code}
\]

"***" Adds "E" before the numeric data, and the space before that is filled with "*":

\[
\text{PRINT } \#1 \text{ USING "***";123}
\]
\[
\text{Data to be transmitted} \rightarrow \text{**123 \hspace{2.5cm} Line Feed Code}
\]

"""" 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.

\[
\text{PRINT } \#1 \text{ USING "###,###,###;12345,67}
\]
\[
\text{Data to be transmitted} \rightarrow \text{123,45,67 \hspace{2.5cm} Line Feed Code}
\]

"^^^^" Transmit numeric data by floating point type format. "^^^^" corresponds to the digits for the exponent part.

\[
\text{PRINT } \#1 \text{ USING "##.##";123}
\]
\[
\text{Data to be transmitted} \rightarrow \text{123 \hspace{2.5cm} Line Feed Code}
\]
RUN (run)

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

**FORMAT**

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

- **Port number**: integer type constants.
  - 0 ≤ port number ≤ 4
- **R**: 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 (IAH), 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 open.

**Execution example**

```
300 "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**: integer type constants.
  - 0 ≤ port number ≤ 4
  - 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 (IAH) 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**: integer type constants.
  - 0 ≤ port number ≤ 4
- **GOSUB start line number**: integer type 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 56) 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 trapings (including ERROR, STRING, STOP, SPRITE, INTERVAL and KEY).

**Execution example**

```
CALL COM1, 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 1.

**COMBREAK (communication break)**

Sends break sequence.

**FORMAT**

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

- **Port number**: integer type constants.
  - 0 ≤ port number ≤ 4
- **Expression**: 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 indicates 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: Integer type constants, 0 ≤ port number ≤ 4

Expression: 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: String type constants, variables, array variables, and their expression

Receive baud rate: Numeric type constants, variables, array variables, and their expression, 50 ≤ Receive baud rate ≤ 1200

Transmit baud rate: Numeric type constants, variables, array variables, and their expression, 50 ≤ Transmit baud rate ≤ 1200

Time out: Numeric type constants, variables, array variables, and their expression, 0 ≤ time out ≤ 255

**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 to the following format:

<table>
<thead>
<tr>
<th>Data length</th>
<th>Parity check</th>
<th>Stop bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>5: 5 bits</td>
<td>E: Even parity</td>
<td>1: 1 bit</td>
</tr>
<tr>
<td>6: 6 bits</td>
<td>O: Odd parity</td>
<td>2: 2 bits</td>
</tr>
<tr>
<td>7: 7 bits</td>
<td>I: Ignore parity</td>
<td>1.5 bits</td>
</tr>
<tr>
<td>8: 8 bits</td>
<td>N: No parity</td>
<td></td>
</tr>
</tbody>
</table>

**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

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("0:8E",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 O到4, integer type constants,
0 ≤ port number ≤ 4

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 O到4, integer type constants,
0 ≤ port number ≤ 4

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")

Port number O到4, integer type constants,
0 ≤ port number ≤ 4

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 O到4, integer type constants, 0 ≤ port number ≤ 4

Variable O到2, 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 elapsed
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 (CONTROL + 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
      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.
HBI-232
IF-104 BOARD

IF-104 COMPONENT SIDE
H81-322 (EK)
H81-322 (LJ)
<table>
<thead>
<tr>
<th>Type</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1S155S</td>
<td>29</td>
</tr>
<tr>
<td>2SC641K</td>
<td>29</td>
</tr>
<tr>
<td>27128-RS232CHB1232</td>
<td>26</td>
</tr>
<tr>
<td>MB8416A-2P-5K</td>
<td>27</td>
</tr>
<tr>
<td>SN74LS133N</td>
<td>27</td>
</tr>
<tr>
<td>SN74LS04N</td>
<td>27</td>
</tr>
<tr>
<td>SN74LS08N</td>
<td>27</td>
</tr>
<tr>
<td>SN74LS10N</td>
<td>27</td>
</tr>
<tr>
<td>SN74LS138N</td>
<td>27</td>
</tr>
<tr>
<td>SN74LS158N</td>
<td>27</td>
</tr>
<tr>
<td>SN74LS322N</td>
<td>27</td>
</tr>
<tr>
<td>SN74LS337AN</td>
<td>28</td>
</tr>
<tr>
<td>SN74LS744AN</td>
<td>28</td>
</tr>
<tr>
<td>SN75188N</td>
<td>28</td>
</tr>
<tr>
<td>SN75189AN</td>
<td>28</td>
</tr>
<tr>
<td>µP08251AFC</td>
<td>28</td>
</tr>
<tr>
<td>µP08253C-5</td>
<td>29</td>
</tr>
</tbody>
</table>

![Diagram of semiconductor pin assignments](image_url)

### Table: Terminal Mode

<table>
<thead>
<tr>
<th>Terminal Mode</th>
<th>CE</th>
<th>OE</th>
<th>PDN</th>
<th>VPP</th>
<th>VCC</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>+5V</td>
<td>+5V</td>
<td>DATA OUT</td>
</tr>
<tr>
<td>STANDBY</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>+5V</td>
<td>+5V</td>
<td>HIGH IMPEDANCE</td>
</tr>
<tr>
<td>PROGRAMMING</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>+21V</td>
<td>+5V</td>
<td>DATA IN</td>
</tr>
<tr>
<td>PROGRAM VERIFY</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>+21V</td>
<td>+5V</td>
<td>DATA OUT</td>
</tr>
<tr>
<td>PROGRAM INIT</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>+21V</td>
<td>+5V</td>
<td>HIGH IMPEDANCE</td>
</tr>
<tr>
<td>HIGH SPEED PROGRAMMING</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>+21V</td>
<td>+5V</td>
<td>DATA IN</td>
</tr>
</tbody>
</table>

1: TTL LEVEL HIGH VOLTAGE IN
0: TTL LEVEL LOW VOLTAGE IN
X: DON'T CARE 1 OR 0
FUNCTION TABLE

<table>
<thead>
<tr>
<th>INPUTS</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1:0</td>
<td>COUNTER No. 0</td>
</tr>
<tr>
<td>D2:1</td>
<td>COUNTER No. 1</td>
</tr>
<tr>
<td>D3</td>
<td>COUNTER No. 2</td>
</tr>
<tr>
<td>D4</td>
<td>LOW LEVEL</td>
</tr>
<tr>
<td></td>
<td>HIGH LEVEL</td>
</tr>
<tr>
<td></td>
<td>HIGH IMPEDANCE</td>
</tr>
</tbody>
</table>

CONTROL WORD FORMAT

<table>
<thead>
<tr>
<th>CODE</th>
<th>OPERATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0-BIT BINARY</td>
</tr>
<tr>
<td>1</td>
<td>1- Bit BINARY</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CODE</th>
<th>SELECTED</th>
<th>WDL</th>
<th>WUX</th>
<th>WLO</th>
<th>X</th>
<th>B</th>
<th>A</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>COUNTER No. 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>COUNTER No. 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>COUNTER No. 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CODE</th>
<th>SELECTED</th>
<th>COUNTER No. 0</th>
<th>COUNTER No. 1</th>
<th>COUNTER No. 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>COUNTER No. 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>COUNTER No. 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>COUNTER No. 2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

D1-D4: 8-BIT DATA IN
CLK: COUNTER CLOCK
GATE: COUNTER GATE
OUT: COUNTER OUTPUT
X: X-BIT DATA ON DATA CS: CHIP SELECT
A1, A0: COUNTER SELECT

3-LINE-TO-8-LINE DECODER

D1-D4: ENCODED SELECT
D5: SELECT 0-3
D6: SELECT 4-7
D7: SELECT 8-15
D8: SELECT 16-31

INPUTS |
-------|
D1-D4  | 0 |
D5     | 0 |
D6     | 0 |
D7     | 0 |
D8     | 0 |

OUTPUTS |
--------|
D1-D4  | 0 |
D5     | 0 |
D6     | 0 |
D7     | 0 |
D8     | 0 |

32,000V
## CHAPTER 4
REPAIR PARTS AND FIXTURE

### 4-1. EXPLODED VIEW

<table>
<thead>
<tr>
<th>No.</th>
<th>Parts No.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2-224-904-00</td>
<td>BUSH, CORD</td>
</tr>
<tr>
<td>2</td>
<td>3-701-690-00</td>
<td>LABEL (MADE IN JAPAN)</td>
</tr>
<tr>
<td>3</td>
<td>4-606-567-02</td>
<td>PROTECTOR</td>
</tr>
<tr>
<td>4</td>
<td>4-606-568-01</td>
<td>SPRING, TORSION</td>
</tr>
<tr>
<td>5</td>
<td>4-606-569-02</td>
<td>CASE (REAR), CARTRIDGE</td>
</tr>
<tr>
<td>6</td>
<td>4-606-570-02</td>
<td>CASE (FRONT), CARTRIDGE</td>
</tr>
<tr>
<td>7</td>
<td>4-606-310-01</td>
<td>LABEL, CARTRIDGE</td>
</tr>
<tr>
<td>8</td>
<td>4-606-667-01</td>
<td>COVER, SWITCH</td>
</tr>
<tr>
<td>9</td>
<td>1-558-396-11</td>
<td>CORD, CONNECTION</td>
</tr>
<tr>
<td>10</td>
<td>4-609-304-01</td>
<td>STOPPER, CABLE</td>
</tr>
</tbody>
</table>

**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. Items with no part number and/or no description are not stocked because they are seldom required for routine service.
## 4.2. ELECTRICAL PARTS LIST

<table>
<thead>
<tr>
<th>Ref. No.</th>
<th>Parts No.</th>
<th>Description</th>
<th>Ref. No.</th>
<th>Parts No.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IF 104 Board</td>
<td></td>
<td></td>
<td>IC1</td>
<td>8-759-951-88</td>
<td>SN75188N</td>
</tr>
<tr>
<td>C1 1-102-852-00</td>
<td>CERAMIC 47PF 5% 50V</td>
<td></td>
<td>IC2</td>
<td>8-759-900-04</td>
<td>SN74LS04N</td>
</tr>
<tr>
<td>C2 1-102-515-00</td>
<td>CERAMIC 24PF 5% 50V</td>
<td></td>
<td>IC3</td>
<td>8-759-900-04</td>
<td>SN74LS04N</td>
</tr>
<tr>
<td>C4 1-162-561-11</td>
<td>CERAMIC 0.1 16V</td>
<td></td>
<td>IC4</td>
<td>8-759-951-89</td>
<td>SN75188AN</td>
</tr>
<tr>
<td>C5 1-162-561-11</td>
<td>CERAMIC 0.1 16V</td>
<td></td>
<td>IC5</td>
<td>8-759-901-38</td>
<td>SN74LS138N</td>
</tr>
<tr>
<td>C6 1-162-561-11</td>
<td>CERAMIC 0.1 16V</td>
<td></td>
<td>IC6</td>
<td>8-759-904-53</td>
<td>SN74ALS133N</td>
</tr>
<tr>
<td>C7 1-124-236-00</td>
<td>ELECT 47 20% 16V</td>
<td></td>
<td>IC7</td>
<td>8-759-103-27</td>
<td>P008251AF</td>
</tr>
<tr>
<td>C8 1-102-114-00</td>
<td>CERAMIC 470PF 10% 50V</td>
<td></td>
<td>IC8</td>
<td>8-759-182-53</td>
<td>μP08253C-5</td>
</tr>
<tr>
<td>C9 1-161-330-00</td>
<td>CERAMIC 0.01 30% 25V</td>
<td></td>
<td>IC9</td>
<td>8-759-900-32</td>
<td>SN74LS32N</td>
</tr>
<tr>
<td>C10 1-161-330-00</td>
<td>CERAMIC 0.01 30% 25V</td>
<td></td>
<td>IC10</td>
<td>8-759-900-74</td>
<td>SN74LS74AN</td>
</tr>
<tr>
<td>C11 1-161-330-00</td>
<td>CERAMIC 0.01 30% 25V</td>
<td></td>
<td>IC11</td>
<td>8-759-901-10</td>
<td>SN74LS10N</td>
</tr>
<tr>
<td>C12 1-161-330-00</td>
<td>CERAMIC 0.01 30% 25V</td>
<td></td>
<td>IC12</td>
<td>8-759-903-67</td>
<td>SN74LS9371AN</td>
</tr>
<tr>
<td>C13 1-161-330-00</td>
<td>CERAMIC 0.01 30% 25V</td>
<td></td>
<td>IC13</td>
<td>8-759-767-64</td>
<td>27128-25222CHB1232</td>
</tr>
<tr>
<td>C14 1-161-330-00</td>
<td>CERAMIC 0.01 30% 25V</td>
<td></td>
<td>IC14</td>
<td>8-759-900-32</td>
<td>SN74LS32N</td>
</tr>
<tr>
<td>C15 1-161-330-00</td>
<td>CERAMIC 0.01 30% 25V</td>
<td></td>
<td>IC15</td>
<td>8-759-900-32</td>
<td>SN74LS32N</td>
</tr>
<tr>
<td>C16 1-161-330-00</td>
<td>CERAMIC 0.01 30% 25V</td>
<td></td>
<td>IC16</td>
<td>8-759-900-74</td>
<td>SN74LS74AN</td>
</tr>
<tr>
<td>C17 1-161-330-00</td>
<td>CERAMIC 0.01 30% 25V</td>
<td></td>
<td>IC17</td>
<td>8-759-911-92</td>
<td>MB8410A-12F-3K</td>
</tr>
<tr>
<td>C18 1-161-330-00</td>
<td>CERAMIC 0.01 30% 25V</td>
<td></td>
<td>IC18</td>
<td>8-759-901-56</td>
<td>SN74LS136N</td>
</tr>
<tr>
<td>C19 1-161-330-00</td>
<td>CERAMIC 0.01 30% 25V</td>
<td></td>
<td>IC19</td>
<td>8-759-900-08</td>
<td>SN74LS08N</td>
</tr>
<tr>
<td>C20 1-161-330-00</td>
<td>CERAMIC 0.01 30% 25V</td>
<td></td>
<td>G1</td>
<td>8-729-364-12</td>
<td>25S641K</td>
</tr>
<tr>
<td>C21 1-161-330-00</td>
<td>CERAMIC 0.01 30% 25V</td>
<td></td>
<td>R1</td>
<td>1-247-145-00</td>
<td>CARBON 3.9K 5% 1/4W</td>
</tr>
<tr>
<td>C22 1-161-330-00</td>
<td>CERAMIC 0.01 30% 25V</td>
<td></td>
<td>R2</td>
<td>1-247-725-11</td>
<td>CARBON 10K 5% 1/4W</td>
</tr>
<tr>
<td>C23 1-161-330-00</td>
<td>CERAMIC 0.01 30% 25V</td>
<td></td>
<td>R3</td>
<td>1-247-137-00</td>
<td>CARBON 1.8K 5% 1/4W</td>
</tr>
<tr>
<td>CN1 1-064-009-00</td>
<td>PIN, CONNECTOR 10P</td>
<td></td>
<td>R4</td>
<td>1-247-704-11</td>
<td>CARBON 220 5% 1/4W</td>
</tr>
<tr>
<td>D1 8-719-815-55</td>
<td>1S1555</td>
<td></td>
<td>R5</td>
<td>1-247-137-00</td>
<td>CARBON 1.8K 5% 1/4W</td>
</tr>
<tr>
<td>D2 8-719-815-55</td>
<td>1S1555</td>
<td></td>
<td>R6</td>
<td>1-247-855-00</td>
<td>CARBON 10K 5% 1/8W</td>
</tr>
<tr>
<td>S1 1-554-949-11</td>
<td>SWITCH, SLIDE</td>
<td></td>
<td>X1 1-567-483-11</td>
<td>VIBRATOR, CRYSTAL 3.684MHz</td>
<td></td>
</tr>
</tbody>
</table>

### 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.