HBI-232MKII

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

CLOSE (close)

Closes a file opened by an OPEN statement.

FORMAT

CLOSE [#] [file number] Pile number] …

File number: Integer constants,
1 <= file number <= the number specified by MAXFILES = statement
If 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

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: 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.

INPUT$ (input dollar)

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

FORMAT

INPUT$ (X, [#] file number)

X : Numeric type constants, variables, array variables, their expression,
1 < X <256

File number : 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 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 "COMO:" FOR INPUT AS #1
20 X$=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.

INPUT# (input number)

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

FORMAT

INPUT# File number, variable [,variable]

File number: Integer constants,1<= file number <= the number specified by MAXFILES = statement

Variable : 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 "COMO:" 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: Integer constants, 1<= file number <= the number specified by MAXFILES = statement

Variable : String type variable, 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 "COMO:" 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 assigns 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: Integer type constants, 0 <= port number <= 4
If omit = 0
If '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 "COM:",R

LOC (location)
Returns the number of characters in the receive buffer file.

FORMAT

LOC (file number)

File number: 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 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.

LOF (length of file)
Returns the free space remaining in the receive buffer file.

FORMAT
LOF (file number)

File number: 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: Integer type constants, 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 merge 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 : Interger type constants, 0<= port number <=4

If Omit = 0

Mode : Output , Input

if omit = input/output

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#, INPUTS

Execution example

OPEN "COMO:" 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: 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 "COMO:" FOR OUTPUT AS #1
20 A$=”ABC":B$="DEF"
30 PRINT #1,A$;B$
40PRINT#1,A$,B$
50PRINT+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: Integer constants, 1 <= file number <= the number specified by MAXFILES= statement

Expression: 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 "COMO:" 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 "COMO:" 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

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 "COMO:" 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 "COMO:" FOR OUTPUT AS #1
20 PRINT #1 USING "###";+123
30 PRINT #1 USING "###";-123
40 CLOSE #1
```

Data to be transmitted —> 123
%-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 "COMO:" 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 "COMO:" FOR OUTPUT AS #1
 20 PRINT #1            USING "**######";123
          30 PRINT #1             USING
          "**######";-234
          40 CLOSE #1
```

Data to be transmitted -> *****123 (Line Feed)
****-234

" ££"

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

```
 10 OPEN "COMO:" FOR OUTPUT AS #1
 20 PRINT #1            USING "EE###";1234
30 PRINT #1             USING "+EE###";-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: Integer type constants, 0 < port number < 4
if omit = 0
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 (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 "COMO:",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:** Integers type constants, 0 <= port number <= 4
if 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 "COMO:"

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 indicate that transmission is suspended.

Execution example
CALL COMBREAK(,20)

The 20 break characters will be sent to the RS-232C

COMGOSUB

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
If omit = 0

Start line **number :** 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.
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
if omit = 0

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
If omit = "0:8N1XHNNN"

Receive baud rate : Numeric type constants, variables, array variables, and their expression, 50 <= Receive baud rate <= 1200
If omit = 1200

Transmit baud rate: Numeric type constants, variables, array variables, and their expression, 50 <= Transmit baud rate <= 1200
If omit = the same baud rate as the receive baud rate

Time out : Numeric type constants, variables, array variables, and their expression. 0 < time out < 255
If 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” consist 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-232 Port Number control

Data Length
5: 5 bits
6: 6 bits
7: 7 bits
8: 8 bits
del

Parity Check
E: Even Parity
O: Odd Parity
I: Ignore Parity
N: No Parity
Ins

Length of stop bits
1: 1 bit*
2: 1,5 bits
3: 2 bits

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

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

Shift-in/Shift-out

S: Enable Control
N: Disable Control

Automatic LF Code

A : Delete LF Code
N : Not delete LF Code

Automatic LF Code

A : Insert LF Code
N : Not insert LF code

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: Integer type constants, 0 <= port number <= 4
if 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:1)

Port number: 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 interrupt from the RS-232C port

FORMAT
CALL COMSTOP ([“port number:”])

Port numbed: Integer type constants, 0 <= port number <= **4**
if 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: Integer type constants, 0 <= port number <= **4**
If omit = 0

Variable: 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:

bit 15 : **Receive buffer overflow error** (Data is transmitted when 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** [CTRL] + [STOP] was pressed

0: Not pressed
1: Pressed

bit 9 : Reserved

bit 8 : Reserved

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

bit 4 : Reserved

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

bit 0 : **CD Signal Status**
0 : OFF
1 : ON

COMTERM

Sets the MSX computer in the terminal mode.

FORMAT

CALL COMTERM [("port number:”)]

Port number: Integer type constants, 0 <= port number<=4
if 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 the following chapter.

USING TERMINAL

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.

COMTERM

Sets your MSX computer to work as a terminal.
To reset the terminal, or to exit from the terminal mode, press the [XTRL] + [STOP] keys 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. 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 (OOH to 1FH) by **""** and the character assigned to the control code plus 40H.
ex.) The return code (ODH) will be displayed as follows:

```
        ^ M
              4DH
```

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

"Break sequence": The break sequence are used to set the 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 simultaneously again.

[SHIFT]+[F3]

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

[STOP]

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

OTHER TECHNICALS DETAILS

[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 over-flow, and so forth.

[SHIF-IN / SHIFT-OUT control]

When the connected device on the other side uses 7-bit JIS (Japan Industry Standard) character set, the SI code (OFH) and SO code (OEH) 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 / DERLETE CONTROL]

The MSX computer uses a set of carriage return code (ODH) and line feed code (OAH) 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 prevents 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 hand-shake 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.