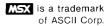
SONY



MSX-BASIC Version 2.0 REFERENCE CHART

HITBIT



CONSTANTS AND VARIABLES

| | String type | Character string of 0 to 255 characters (enclosed in quotation marks) | |
|----------|------------------------|---|--|
| | Integer type | -32768 to +32767 | |
| Constant | Floating-point type | Significant digits: 6 (single precision) or 14 (double precision) Exponent part: -64 to +62 | |
| | Hexadecimal expression | Takes a prefix "&H" | |
| | Octal expression | Takes prefix "&O" or "O" | |
| | Binary expression | Takes a prefix "&B" | |

| | Variable name | First two characters are effective. |
|----------|-----------------|---|
| Variable | Type declarator | Written after variable name %: Integer type !: Single precision #: Double precision \$: String type |

SPECIAL VARIABLES

TIME: Retain a value in the timer. Can be rewritten.

SPRITE\$ (sprite number): Retain the sprite pattern.

[Example] SPRITE\$(1)=CHR\$(&H18)+CHR\$(&H3C)+CHR\$(&H7E)+CHR\$(&HFF)+

CHR\$(&H18)+CHR\$(&H18)+CHR\$(&H18)+CHR\$(&H18)

Special commands and functions for VDP (Video Display Processor)

BASE (expression) : Used to read or write the base address of the VDP table. VDP (numeric value) : Used to read or write the contents of the VDP register.

| | | 0FT 1/10FO 0 |
|---------------------------------|-------------------------------------|--------------|
| SET VIDEO [mode], | Specify superimposing | SET VIDEO 2 |
| [brightness], [color bus], | and other modes. (used | |
| [sync], [sound], [video input], | only with computers that | |
| [AV control] | have the superimpose | |
| | function) | |
| | Mode | |
| | 0: computer signal | |
| | (internal sync | |
| | only) | |
| | computer signal | |
| | 2: superimpose | |
| | 3: TV signal | |
| | Brightness | |
| | 0: normal brightness | |
| | 1: half-brightness | |
| | Color Bus | |
| | Ø: VDP color bus | |
| | input | |
| | 1: VDP color bus | |
| | output | |
| | Simultaneous | |
| | 0: internal sync | i |
| | 1: external sync | |
| | Sound | |
| | 0: external sound | |
| | signal not mixed | |
| | 1: right channel | |
| | external sound | |
| | signal mixed | |
| | 2: left channel | |
| | external sound | |
| | signal mixed | |
| | 3: both channels external sound | |
| | | |
| | signal mixed Video Input | |
| | Ø: select RGB | |
| | multiconnector | |
| | | |
| | input 1: select external | |
| | video input | |
| | connector input | |
| | AV Control | |
| | Ø: TV signal | |
| | 1: External video | |
| | i, External video | |

MEMORY SWITCH FUNCTION COMMANDS

| format | function | example |
|------------------------------|--|-------------------------|
| SET ADJUST (X, Y) | Change the location of the display on the screen | SET ADJUST (-4, 3) |
| SET BEEP [sound], [volume] | Select the BEEP sound. | SET BEEP 2, 3 |
| SET TITLE ["title"], [color] | Set the title to be displayed on the initial screen display. | SET TITLE "SONY" |
| SET PROMPT "prompt" | Set the prompt statement to be displayed when BASIC is in a command- wait status. | SET PROMPT "Ready" |
| SET PASSWORD "password" | Set the system password. | SET PASSWORD "BASIC" |
| SET SCREEN | Set the current SCREEN values as the initial start-up values. | |

COMMANDS FOR DATA INPUT/OUTPUT

| format | function | example | |
|--|--|--|--|
| DATA constant [, constant] [, constant] | Give data to be read with a READ statement. | DATA 3, 4, 5, 6, ABC, "C, D" | |
| INPUT ("prompt statement";] variable [, variable] [, variable] | Give value of variable from the keyboard. | INPUT "A\$=";A\$ | |
| LINE INPUT ["prompt statement";] variable | Give string of up to 254 characters from the keyboard to the string type variable | LINE INPUT "C\$=";C\$ | |
| [LET] variable=x | Assign data to the variable. | LET A=A+5 | |
| MID\$ (X\$, M[, N]) | Replace characters beginning with the Mth character of the string X\$ with characters from the beginning to Nth character of Y\$. | MID\$ (A\$, 2, 5)=B\$ | |
| PRINT [expression] [separator] [expression] [separator] or ? [expression] [separator] [expression] [separator] | Output data onto display screen. A separator is a semi-colon (;), a comma (,) or a space. | PRINT A;B;C | |
| PRINT USING format symbol; expression [, expression] | Output data onto display screen in the specified format. Format symbols: "!" Output the first character. | 10 A\$="ABCDEFG" 20 PRINT USING "\";A\$ 30 PRINT USING "\ \";A\$ | |

| | "\n spaces \" Outputs n+2 charac- ters. "&" Output the entire string. "#" Specify the number of display digits of the numeric data. "+" Add + or - before (after) numeric data. "-" Add - after negative numeric data. "**" Fill space befor numeric data with *. "££" Put £ in front of numeric data. "**\$" Put £ in front of numeric data and fill space in front of it with *. "," Put, after every third digit to the left of the decimal point. "^^^^^\" | 40 PRINT USING "SSS&TTT";A\$ PRINT USING "####";123.45,10.5 PRINT USING "+ # # #";100, 200 PRINT USING "## - ";100, 200 PRINT USING "** # # #";100, -200 PRINT USING "\$\sum \text{*} # # ";100, -200 PRINT USING "** \sum \text{*} # ";10, -200 PRINT USING "** \sum \text{*} # ";10, -20 PRINT USING "* # # # # #, # #";1234.56 PRINT USING "# # # # # #, # #";1234.56 PRINT USING "# # # # # # # # # # # # # # # # # # # |
|---|---|---|
| READ variable [. variable] [, variable] | Read data in DATA statement. | READ A% |
| RESTORE [line number] | Specify the DATA statement to be read with a READ statement executed next. | RESTORE100 |
| SWAP variable, variable | Exchange values of two variables. | SWAP A,B |

COMMANDS FOR CONTROLLING PROGRAM EXECUTION AND FLOW

| format | function | example |
|--|---|---|
| RUN [line number] | Start program execution. | RUN 100 |
| RUN "[drive name] file name [. type name]" [, R] | Load program and execute it | RUN "PROG.BAS" |
| STOP | Interrupt program execution. | |
| CONT | Restart program execution. | |
| END | Terminate program execution. | |
| TRON | Display line number that was executed. | |
| TROFF | Cancel TRON. | |
| FOR variable=initial value TO end value [STEP increment] NEXT [variable] | Repeat the program execution between FOR and NEXT. | FOR I=1 TO 10 STEP 2 NEXT I |
| GOSUB line number RETURN [line number] | Transfer control to the specified subroutine. Return to the main routine with RETURN. | 100 GOSUB 100 |
| GOTO line number | Transfer control to the specified line. | GOTO 100 |
| IF expression Statement Ine number GOTO line number | Branch control according to the expression value. | IF X=0 THEN 100 ELSE 200 |
| ON expression GOTO line number [, line number] | Branch control according to the expression value. | ON A GOTO 100, 200, 300 |
| ON expression GOSUB line number [, line number] | Branch control according to the expression value. | ON SGN (A)+2 GOSUB 1000, 2000, 3000 |

COMMANDS FOR DISPLAY SCREEN

| format | function | example |
|---|--|-----------------------|
| format SCREEN [mode], [sprite size], [key click switch], [baud rate], [printer type], [interlace mode] | function Specify the screen display mode. Mode 0: 80 × 24 character text mode 1: 32 × 24 text mode 2: 256 × 192 dot, 16-color graphic mode 3: 64 × 48 dot, 16-color multicolor mode 4: 256 × 192 dot, 16-color graphic mode, sprite enhanced 5: 256 × 212 dot, 16-color graphic mode, sprite enhanced 6: 512 × 212 dot, 4-color graphic mode, sprite enhanced 7: 512 × 212 dot, 16-color graphic mode, sprite enhanced 8: 256 × 212 dot, 16-color graphic mode, sprite enhanced 8: 256 × 212 dot, 16-color graphic mode, sprite enhanced 8: 256 × 212 dot, 256-color graphic mode, sprite enhanced 8: 256 × 216 dot, 256-color graphic mode, sprite enhanced 8: 256 × 216 dot, 256-color graphic mode, sprite enhanced 8: 16 × 16 dot unmagnified 1: 8 × 8 dot magnified 2: 16 × 16 dot unmagnified 3: 16 × 16 dot magnified Key click switch 0: Supress key click sounds. 1: Produce key click sounds. Baud rate 0: 1200 baud Printer type 0: MSX printer 1: Non-MSX printer Interlace mode 0: non-interlace 1: interlace 2: interlace, even/odd page change display 3: interlace, even/odd page change display 3: interlace, even/odd page change display | example SCREEN 2, 0,0 |
| SET PAGE [display page], | Specify the display page | SET PAGE 0, 1 |
| [active page] | and the active page. | |

| WIDTH number of characters | Specify the number of characters per line in the text mode. | WIDTH 28 | |
|--|---|--|--|
| CLS | Erase all displays on the screen. | | |
| KEY (ON OFF) | Display or erase the contents of function keys. | KEY OFF | |
| LOCATE [x-coordinate], [y-coordinate], [cursor switch] | Move the cursor. Cursor switch Ø: Not display the cursor. 1: Display the cursor. | LOCATE 10, 12, 1 | |
| COLOR [foreground color], [background color], [border color] | Specify colors of the foreground, background and the border. | COLOR 8, 15, 2 | |
| COLOR=(palette number, red brightness, green brightness, blue brightness) | Assign colors to the color palette | COLOR=(2, 0, 3, 7) | |
| COLOR=RESTORE | Assign the content of the color lookup table in the video RAM to the VDP color palette register. | | |
| COLOR[=NEW] | Return color palette to initial default settings | | |
| PUT SPRITE sprite plane number, [[STEP] (x-coordinate, y-coordinate)], [color], [sprite number] | Display the specified sprite pattern at the specified position on the specified sprite plane. | PUT SPRITE 0, (100, 50), 7, 2 | |
| COLOR SPRITE\$ (sprite plane no.) = "character expression" | Specify the color of each line of a sprite. Significance of each character bit: B7 B6 B4 B3 B2 B1 B9 B7: For 1, moves sprite 32 dots to the left. B6: For 1, ignores sprite priority position and overlap, and displays the color whose code is the result of OR of the overlapping colors. B5: For 1, ignores sprite overlap. B4: Not used. B3—B0: color code | COLOR SPRITE\$(Ø) = CHR\$(1) + CHR\$(7) | |
| COLOR SPRITE (sprite plane no.)=palette no. | Change the color of the sprite on the specified sprite plane. | COLOR SPRITE (1)=4 | |
| Logical Operations | PSET, PRESET, AND, OR, XOR, TPSET, TPRESET, TAND, TOR, TXOR | | |

COMMANDS FOR GRAPHIC DISPLAY

| format | function | example |
|---|--|----------------------------|
| CIRCLE [STEP] (x-coordinate, y-coordinate), radius, [color code], [start angle], [end angle], [aspect ratio] | Draw a circle. | CIRCLE (80, 60), 15, 8 |
| DRAW "graphic subcommands" | Draw an arbitrary graphic. | DRAW "S4ØU5R5D5L5" |
| LINE [[STEP] (x-coordinate, y-coordinate)]-[STEP] (x-coordinate, y-coordinate), [color code] {[, BF]} [, logical operation] | Draw a line or a square. | LINE -STEP (20, 50),, B |
| PAINT [STEP] (x-coordinate, y-coordinate), [display color], [border line color code] | Color the area inside the border line. | PAINT (120, 100) |
| PSET [STEP] (x-coordinate, y-coordinate), [color code], [logical operation] | Mark a dot. | PSET STEP (10, 10), 14 |
| PRESET [STEP] (x-coordinate, y-coordinate), [color code], [logical operation] | Mark or erase a dot. | PRESET (100, 100) |

COMMANDS FOR SCREEN DATA PROCESSING

| format | function | example |
|--|---|--|
| COPY (X1, Y1)-(X2, Y2) [, source page] TO (X3, Y3), [destination page], [logical operation] | Transfer image data in the VRAM to other sectors in the VRAM | COPY (20, 30)— (70, 50), 1 TO (90, 60), 0, AND |
| COPY (X1, Y1)—(X2, Y2) [, source page] TO array variable name | Transfer image data in the VRAM to an array variable | COPY (20-,30)- (70,50), 0 TO S |
| COPY array variable name [, direction] TO (X3, Y3), [destination page], [logical operation] | Transfer image data in an array variable to the VRAM | COPY S,1 TO (100, 100), 1, XOR |
| COPY (X1, Y1)-(X2, Y2) [, source page] TO "[drive name] file name [. type name]" | Save the image data in the VRAM to the disk file. | COPY (10, 10) (120, 90) TO "PORTRAIT.PIC" |
| COPY "[drive name] file name [. type name]" [, direction] TO (X3, Y3), [destination page], [logical operation] | Load image data in the disk file to the VRAM | COPY "PORTRAIT.PIC" TO (10, 10) |
| COPY "[drive name] file name [. type name]" TO array variable name | Load image data in the disk file to the array variable | COPY "PORTRAIT.PIC" TO S |
| COPY array variable name TO "[drive name] file name [. type name]" | Save the image data in an array variable to the disk file. | COPY S TO "PORTRAIT.PIC" |
| [. type name] COPY SCREEN [mode], [mask] | Digitize an external video signal and write it in the VDP. (used only with computers that have the digitize function) Mode ①: the signal of 1 field is digitized and written on the display page 1: signals of 2 fields (1 frame) are digitized: one is written on the display page, and one is written on the page whose page number is smaller than that of the display page by one. | |

(When B is added, a subcommand changes the starting point only without drawing lines.

If N is added, it draws lines but does not move starting

point.)

| subcommand | function | initial value | subcommand | function | initial value |
|------------|---------------------------------------|------------------|----------------------------|--|------------------|
| Mx, y | To an absolute position (x, y) | | Fn | Move down to the right. | n=1 |
| M ± x, ± y | Move by ±x, ±y from current position. | | Gn | Move down to the left. | n=1 |
| Un | Move up. | n=1 | Hn | Move up to the left. | n=1 |
| Dn | Move down. | n=1 | An | Rotate the coordinate system. | |
| Rn | Move to the right. | n=1 | Cn | Specify a color. | n=15 |
| Ln | Move to the left. | n=1 | Sn | Specify the unit number of dots. | n = 4 |
| En | Move up to the right. | n=1 | X string type variable; | Execute the subcommand assigned to the string type variable. | |

COMMANDS FOR MUSIC PERFORMANCE

| format | function | example |
|--|-------------------------------|-----------------------|
| BEEP | Generate a beep sound. | BEEP: BEEP: BEEP |
| SOUND PSG register number, expression | Write data into PSG register. | SOUND 7, 7 |
| PLAY "music subcommands" [, "music subcommands"] [, "music subcommands"] | Play music. | PLAY "O4L4CEGEL1C" |

Music subcommands

| subcommand | function and range | initial value | subcommand | function and range | initial value |
|-----------------------------|--|------------------|------------|-------------------------------------|------------------|
| A # - - G + - | Music notes | | Tn | Tempo 32 ≦ n ≦ 255 | n=120 |
| On | Octave 1≤n≤8 | n = 4 | Vn | Volume Ø≦n≦15 | n=8 |
| Nn | Pitch 0≤n≤96 | | Mn | Envelope frequency 1≦n≦ 65535 | n=255 |
| Ln | Length 1≤n≤64 | n=4 | Sn | Envelope pattern 1≤n≤15 | n=1 |
| Rn | Rest 1 ≦ n ≦ 64 | n = 4 | | Dot | |
| X string type variable; | Execute the subcommand assigned to the string type variable. | | | | |

COMMANDS FOR PROGRAM AND DATA FILES

| format | function | example |
|---|---|--|
| MAXFILES=expression | Set the number of files that can be opened in a program. | MAXFILES=3 |
| OPEN "[device name] [file name [. type name]]" [FOR mode] AS [#] file number [LEN=record length] | Open a file and specify a mode. Modes: OUTPUT Write INPUT Read When the mode is specified a sequential file is opened. When the mode is not specified, a random access file is opened. | OPEN "CRT : TEST" FOR OUTPUT AS#1 |
| PRINT # file number, [expression] [separater] [expression] | Write data into sequential file in sequence. | PRINT #1, "ABC" |
| PRINT # file number, USING format symbol; expression [, expression] | Write data into sequential file in sequence in the specified format. (See PRINT USING.) | PRINT #1, USING "\ \";A\$ |
| INPUT # file number, variable [, variable] | Read data from sequential file in sequence and assign them to variables. | INPUT #1, A, B, C |
| LINE INPUT # file number, string type variable | Read string up to 254 characters from sequential file and assign them to variable. | LINE INPUT #1, A\$ |
| CLOSE [#] [file number] [, file number] | Close files. | CLOSE #1, 2 |
| SAVE "[device name] [file name]" | Save an ASCII format program (other than disk). | SAVE "CAS:PROGRAM" |
| SAVE "[drive name] file name [. type name]" [,A] | Save a program on the disk. The program is saved in the ASCII format when the A option is specified, and in intermediate language when the A option is omitted. | SAVE "GAME1.BAS" SAVE "GAME2.ASC",A |
| LOAD "[device name] [file name]" | Load an ASCII format program (other than disk). | LOAD "CAS:PROGRAM" |
| LOAD "[drive name] file name [. type name]" [,R] | Load a program from the disk. | LOAD "GAME1.BAS",R |
| MERGE "[device name] [file name]" | Load an ASCII format program and merge it with the program in memory. | MERGE "CAS:PROG2" |

| MERGE "[drive name] [file | Load a program fra th | MEDOE |
|--|---|---|
| name [. type name]]" | Load a program from the disk saved in the ASCII format and merge it with a program in memory. | MERGE "GAME2.ASC" |
| BSAVE "[device name] [file name]", start address, end address [, execution starting address] | Save the contents of memory within the specified range (other than disk). | BSAVE "CAS:GAME", &H3000, &H3FFF |
| BSAVE "[drive name] [file name [. type name]]", start address, end address [, execution start address] [, S] | Save the content of the main memory (without S option) or the video RAM on the disk (with S option). | BSAVE "PROG.BIN", &HE000, &HE8000 BSAVE "CHART", 0, &H3FFF,S |
| BLOAD "[device name] [file name]" [, R] [, offset] | Load machine language program (other than disk). Load and execute program when, R is added. The offset is one for the memory address at the time of loading. | BLOAD "CAS:GAME", R |
| BLOAD "[drive name] [file name [. type name]]" [.R] [,S] [, offset] | Load a machine language program from the disk. When the R option is specified, loads the program and executes it. When the S option is specified, loads the file data to the video RAM. | BLOAD "PROG.BIN",R BLOAD "CHART",S |
| CSAVE "file name" [, baud rate] | Save a program onto cassette tape in intermediate language. Baud rate: 1. 1200 baud 2. 2400 baud | CSAVE "STAR" |
| CLOAD ["file name"] | Load program from cassette tape. | CLOAD "STAR" |
| CLOAD? ["file name"] | Compare program saved on cassette tape and program in memory. | CLOAD? "STAR" |
| FIELD [#] file number, character length AS string variable [,character length AS string variable] | Define 1 random access file record. | FIELD #1, 12, AS NAM\$, 14 AS TEL\$ |
| LSET string variable= string expression RSET string variable= string expression | Write the content of a string expression to the string variable defined in the record. (LSET provides left justification; RSET provides right justification) | LSET TEL\$=B\$ RSET NAM\$="TOM" |
| PUT [#] file number [, record number] | Write the content of a record to a random access file on the disk. | PUT #1,1 |
| GET [#] file number [, record number] | Read 1 record from a random access file on the disk. | GET #1, 10 |

Device name

CAS: Cassette tape
CRT: Text mode screen
GRP: Graphic mode screen
LT: Printer
MEM: Memory disk
A: Floppy disk drive names
B: Floppy disk drive names
D: Floppy disk drive names
E: Floppy disk drive names
H: Floppy disk drive names

COMMANDS FOR FLOPPY DISK AND MEMORY DISK MANAGEMENT

| format | function | example |
|---|--|--|
| CALL FORMAT | Format a disk. | |
| FILES ["[drive name] [file name [. type name]]"] | Display file names saved on the disk. | FILES " * .BAS" |
| KILL "[drive name] file name [. type name]" | Erase a file on the disk. | KILL "TEST.BAS" |
| NAME "[drive name] old file name [. old type name]" AS "new file name [. new type name]" | Change the name of a file on the disk. | NAME "OLD.DAT" AS "NEW.DAT" |
| COPY "[drive name 1] file name [.type name]" [TO "[drive name 2] file name [.type name]"] | Copy a file on the disk to the same disk or to another disk. | COPY "ABC.BAS" TO "XYZ.BAS" COPY "A:ABC.BAS" TO "B:" |
| CALL MEMINI [(size)] | Allocate a section of memory to be used as a memory disk, and initialize it. | CALL MEMINI (20000) |
| CALL MFILES | Display file names on the memory disk. | |
| CALL MKILL ("file name [.type name]") | Erase a file on the memory disk. | CALL MKILL ("ADRS.DAT") |
| CALL MNAME ("old file name [. old type name]" AS "new file name [. new type name]") | Change a file name on the memory disk. | CALL MFILES ("OLD.DAT" AS "NEW.DAT") |

COMMANDS FOR INTERRUPT

| format | function | example |
|--|--|----------------------------------|
| ON KEY GOSUB line number [, line number] | Interrupt with a function key. | ON KEY GOSUB 1000, 2000, 3000 |
| KEY (function key number) ON | Enable an interrupt with a function key. | KEY (1) ON |
| KEY (function key number) OFF | Disable an interrupt with a function key. | KEY (2) OFF |
| KEY (function key number) STOP | Hold an interrupt with a function key. | KEY (3) STOP |
| ON STRIG GOSUB line number [, line number] | Interrupt with a trigger button of the joystick. | ON STRIG GOSUB 1000,, 2000 |
| STRIG (pointing device number) ON | Enable an interrupt with a joystick. Joystick number: 0 space bar 1 joystick 1 2 joystick 2 | STRIG (1) ON |
| STRIG (pointing device number) OFF | Disable an interrupt with a joystick. | STRIG (2) OFF |
| STRIG (pointing device number) STOP | Hold an interrupt with a joystick. | STIRG (Ø) STOP |
| ON STOP GOSUB line number | Interrupt with the CTRL and STOP keys. | ON STOP GOSUB 1000 |
| STOP ON | Enable an interrupt with the CTRL and STOP keys. | |
| STOP OFF | Disable an interrupt with the CTRL and STOP keys. | |
| STOP STOP | Hold an interrupt with the CTRL and STOP keys. | |
| ON SPRITE GOSUB line number | Interrupt with an overlap of sprite patterns. | ON SPRITE GOSUB 1000 |
| SPRITE ON | Enable an interrupt with an overlap of sprite patterns. | |
| SPRITE OFF | Disable an interrupt with an overlap of sprite patterns. | |
| SPRITE STOP | Hold an interrupt with an overlap of sprite patterns. | |
| ON INTERVAL=interval time GOSUB line number | Interrupt after an interval. Time between interrupts is the interval × 1/50 second. | ON INTERVAL=120 GOSUB 1000 |
| INTERVAL ON | Enable intervalled interrupts. | |
| INTERVAL OFF | Disable intervalled interrupts. | |
| INTERVAL STOP | Hold intervalled interrupts. | |

COMMANDS FOR CONNECTED DEVICES

| format | function | example |
|---|--|--------------------------|
| LPRINT [expression] [separator] [expression] [separator] [expression] | Output data on the printer. | LPRINT A, B, C |
| LPRINT USING format symbol; expression [separater] [expression] [separater] | Output data on the printer in the specified format. (See PRINT USING.) | LPRINT USING "###"; A, B |
| LLIST (starting line number) [-] [end line number] | Print program list on a connected printer. | LLIST 100-200 |
| MOTOR [{ON OFF}] | Turn the tape recorder motor on or off. | MOTOR OFF |

COMMANDS FOR INTERNAL CLOCK

| format | function | example |
|--------------------------|--|---------------------|
| SET DATE "DD/MM/YY" [,A] | Set the date on the internal clock. | SET DATE "05/10/85" |
| GET DATE D\$ [,A] | Assign the current date to a variable. | GET DATE D\$ |
| SET TIME "HH:MM:SS" [,A] | Set the time on the internal clock. | SET TIME "14:05:00 |
| GET TIME T\$ [,A] | Assign the current time to a variable. | GET TIME T\$ |

COMMANDS FOR ERROR PROCESSING

| format | function | example |
|---------------------------------|---|---------------------------------------|
| ERROR error number | Generate an error of the specified error code. Define error codes. | ERROR 3 IF A>100 THEN ERROR 250 |
| ON ERROR GOTO line number | Transfer control to the specified line when an error occurs. | ON ERROR GOTO 1000 |
| RESUME [{ line number NEXT }] | Return control to the main program after executing an error processing routine. | RESUME 10 |

COMMANDS FOR MACHINE LANGUAGE SUBROUTINES

| format | function | example |
|------------------------------------|---|-------------------|
| DEFUSR [integers]=starting address | Define the starting address of user subroutine. | DEFUSRØ=53248 |
| POKE address, expression | Write data into memory. | POKE &HA400, &HFF |

COMMANDS FOR I/O PORT AND MEMORY

| format | function | example |
|--|--|----------------|
| OUT port number, expression | Output data to the I/O port. | OUT &H9Ø, 3 |
| WAIT port number, expression 1 [, expression 2] | Hold program execution until the input data form the I/O port reaches a certain value. | WAIT &H90, 255 |
| VPOKE address, expression | Write one bit of data to the video RAM. | VPOKE 263, Ø1 |

COMMANDS FOR EXTENDED COMMANDS

| format | function | example |
|---|---|----------|
| CALL subroutine name orsubroutine name CALL extended command [argument, argument] orextended command [argument, argument] | Transfer control to the machine language subroutine, or transfer control to an extended command of the ROM cartridge. | CALL SUB |

COMMAND FOR SHIFTING CONTROL TO MSX-DOS

| format | function | example |
|-------------|---------------------------|---------|
| CALL SYSTEM | Shift control to MSX-DOS. | |

FUNCTIONS

NUMERICAL FUNCTIONS

ABS (X) : Give an absolute value. ATN (X) : Give arc tangent.

CDBL (X) : Convert to the double precision type.

CINT (X) Convert to the integer type. $(-32768 \le X \le 32767)$

COS (X) : Give cosine of X radians.

CSNG (X) : Convert to the single precision type.
ERL : Give the number of the line with an error.

ERR : Give the error code.

EXP (X) : Give ex.

FIX (X) : Give the integer part of X

INT (X) : Give the maximum integer less than or equal to X

LOG (X) : Give natural logarithm.
RND (X) : Give random number.

RND (X) : Give random number. SGN (X) : Give 1 if X>0, 0 if X=0 and -1 if X<0

SIN (X) : Give sine of X radians. SQR (X) : Give square root.

TAN(X): Give tangent of X radians.

STRING FUNCTIONS

LEFT\$ (X\$, N) : Give N characters from the left of X\$.

MID\$ (X\$, M [, N]) : Give N characters beginning with the Mth character

from the left of X\$.

RIGHT\$ (X\$, N) : Give N characters from the right of X\$.

SPACES\$ (N) : Give N spaces.

STRING\$ (N, J) : Give N characters whose character code is J.

STRING\$ (N, X\$) : Give N times the first character of X\$. TAB (N) : Move the cursor to the Nth position.

SPC (N) : Give N spaces.

FUNCTION FOR CONVERSION BETWEEN NUMERICAL AND STRING TYPES

ASC (X\$) : Give the character code of the first character of X\$.

BIN\$ (X) : Give a binary expression of X as a string type data.

 $(-32768 \le X \le 65535)$

CHR\$ (X) : Give a character whose character code is X.

HEX\$ (X) : Give a hexadecimal expression of X as a string type

data. (-32768≤X≤65535)

INSTR ([N,] X\$, Y\$) : Give the position of Y\$ after the Nth character of X\$.

LEN (X\$) : Give a number of characters of X\$.

OCT\$ (X) : Give an octal expression of X as a string type data.

 $(-32768 \le X \le 65535)$

STR\$ (X) : Convert to the string type.
VAL (X\$) : Convert to the numeric type.

P CVI, CVS, CVD : Change character string data in a random access file to

numeric data.

MKI\$, MKS\$, MKD\$: Change numeric data into string data to write in a

random access file.

OTHER FUNCTION

PLAY (N) : Check if music is playing.

When N=1, 2 or 3 it gives -1 when music is playing; otherwise it gives 0. When N=0, the status (-1 or 0) of each music subcommand are ORed and the result is

given.

FUNCTIONS FOR DATA INPUT

From the screen

CSRLIN : Give y-coordinate of the cursor. POS (X) : Give x-coordinate of the cursor. POINT (X, Y) : Give color code at point (X, Y).

From data file

EOF (file number) : Give -1 when last data in file is read; otherwise give 0.

INPUT\$ (N, [#] file number): Input and give N characters from the file.

LOF (file number) : Give the file length (bytes).

LOC (file number) : Give the current location in the file.

From the printer

LPOS (X) : Give the position of the print head in the printer buffer.

From memory

FRE (0) : Give unused area in memory. FRE (" ") : Give unused part or string area.

PEEK (address) : Give the memory contents of the address.

VARPTR (variable) : Give the starting address of the memory area storing

the variable.

□ VARPTR : Give the first address of the file control block to which (# file number)

the specified file is assigned.

VPEEK (address) : Give the video RAM contents of the address.

From the keyboard

INKEY\$: Give the character corresponding to the pressed key. INPUT\$ (X)

: Input X characters from the keyboard.

From the disk : Give the space remaining in the disk in cluster units. DSKF (drive number)

From I/O port

INP (port number) : Input data from the I/O port.

From machine language subroutine

(0) USR ₹ to ⅓ (X) : Give the value from the user subroutine. 9

From joystick, paddle or touch pad

STICK (N) : Give the direction of the joystick. $(N=\emptyset)$ for cursor move

(Center=0, Up=1, Right up=2, Right=3, Right down=4,

Down=5, Left down=6, Left=7, Left up=8)

: Give -1 when the joystick trigger button is pressed; STRIG (N)

otherwise, give \emptyset . (N = \emptyset for the space bar)

PDL (N) : Input data from the paddle.

PAD (N) : Give status of the touch pad, light pen, mouse, or track

When $N = \emptyset$ or 4: Give -1 if the touch pad is touched;

otherwise, give 0.

When N=1 or 5: Give the x-coordinate of the position

where the touch pad is touched.

When N=2 or 6: Give the y-coordinate of the position

where the touch pad is touched.

When N=3 or 7: Give -1 if the touch pad switch is

touched; otherwise, give 0.

N=8: -1 if light pen data is valid; Ø if invalid

N=9: light pen X-coordinate

N=10: light pen Y-coordinate

N=11: -1 if light pen switch is pressed; 0 if not pressed

N=12 or 16: request mouse or track ball input (-1 is always returned)

N=13 or 17: mouse or track ball X-coordinate N=14 or 18: mouse or track ball Y-coordinate

N=15 or 19: 0 is always returned

COMMANDS AND STATEMENTS

COMMANDS FOR PROGRAMMING

| format | function | example | |
|---|--------------------------------------|-------------------|--|
| AUTO [starting line number] [, increment] | Generate line numbers automatically. | AUTO 100, 10 | |
| DELETE [line number] [– line number] | Delete lines in a program. | DELETE 30-60 | |
| LIST [starting line number] [-] [end line number] | Display program list. | LIST | |
| NEW | Erase program. | | |
| RENUM [new starting line number], [old starting line number], [increment] | Renumber lines. | RENUM 100, 10, 10 | |
| REM or ' | Insert a comment. | REMPROGRAM 1 | |
| KEY LIST | Display the function key contents. | | |

COMMANDS FOR DEFINITION AND SETTING

| format | function | example |
|--|--|---------------------------------|
| CLEAR (size of character area) [, highest address] | Initialize all variables and set the size of the character string area and the highest memory to be used by BASIC. | CLEAR 400, 55296 |
| DIM variable name (maximum value of a subscript [, maximum value of a subscript]) [, variable name (),] | Declare the name, type, size and dimension of array. | DIM A\$ (100) |
| DEF SNG DBL character STR [- character] [, character [- character]] | Define matching between the first character of a variable name and the type of variable. (INT: integer, SNG: single precision, DBL: double precision, STR: string) | DEFINT I-N |
| DEF FN function name [(parameter [, parameter])] = expression | Define user functions. | DEF FNA (X)=A * X^ 2+B * X+C |
| ERASE [array variable name] [, array variable name] | Erase arrays | ERASE A, B, C |
| KEY function key number, character string | Define strings for function keys. | KEY 1, "LLIST"+CHR\$ (13) |

ERROR MESSAGES

| 1 NEXT without FOR | : No FOR statement corresponding to NEXT statement. |
|---|--|
| 2 Syntax error 3 RETURN without GOSUB | : Syntax error in the statement. : No GOSUB statement corresponding to RETURN |
| 4 Out of DATA | statement. : No more data to be read. |
| 4 Out of DATA 5 Illegal function call | : Illegal specification in function or command. |
| 6 Overflow | : Too big or too small data. |
| 7 Out of memory | : No more memory. |
| 8 Undefined line number | : Undefined line number was specified. |
| 9 Subscript out of range | : Array subscript outside defined range. |
| 10 Redimensioned array | : Array in DIM statement was already specified. |
| 11 Division by zero | : Divided by zero. : The command can not be used in direct command |
| 12 Illegal direct | mode. |
| 13 Type mismatch | : Data type mismatch. |
| 14 Out of string space | : No more string variable area. |
| 15 String too long | : String is too long. |
| 16 String formula too complex | : String is too complex. |
| 17 Can't CONTINUE | : Impossible to continue program execution. |
| 18 Undefined user function | : A function which is not defined by DEF FN |
| 10 Davisa I/O orror | statement was used. : Error in connected equipment. |
| 19 Device I/O error 20 Verify error | : Program in cassette tape and program in memory |
| 20 Verify error | differ. |
| 21 No RESUME | : No RESUME statement that corresponds to ON |
| | ERROR statement. |
| 22 RESUME without error | : No ON ERROR statement that corresponds to |
| 00 11 | RESUME statement. : An error without an error message has occurred. |
| 23 Unprintable error | : Operand is missing. |
| 24 Missing operand 25 Line buffer overflow | : The entered program exceeds the buffer size. |
| 50 P FIELD overflow | : The specified area of a FIELD statement has |
| 56 [] | exceeded the length of the record. |
| 51 Internal error | : Memory content or text is not normal. |
| 52 Bad file number | : Incorrect file number. |
| 53 File not found | The specified file does not exist. |
| 54 File already open | : The file is already open. |
| 55 Input past end | : Last data has been already read. : Incorrect file specification. |
| 56 Bad file name 57 Direct statement in file | : Command in direct command mode was entered |
| 57 Direct statement in the | during file loading. |
| 58 Sequential I/O only | : When a GET statement or PUT statement is |
| oo ooquomaa no om, | attempted for a sequential file. |
| 59 Fine not OPEN | : The file needs to be opened. |
| 60 💾 Bad FAT | : The disk has not been formatted. |
| 61 Bad file mode | : Sequential file, random access file command or |
| OO D Dad drive seems | function mistake. |
| 62 Bad drive name | : Disk drive not in use was specified. : Record specified in PUT or GET statement is 0 or |
| 63 💾 Bad sector number | larger than 32767. |
| 64 File still open | : File has not been closed. |
| 65 File already exists | : New file name specified in a NAME, CALL MNAME |
| • | statement already exists. |
| 66 💾 Disk full | : No more space on the disk. |
| [RAM] disk full | : No space on the memory disk. |
| 67 Too many files | : The number of files has exceeded 255. |
| 68 Disk write protected | : Writing was performed on a write-protected diek. |
| 69 💾 Disk I/O error | : An error occurred which makes recovery impossible at the time of disk input or output. |
| | at the time of disk input of output. |

| 70 | Disk offline |
|----|--------------------|
| | [RAM] disk offline |

: Disk drive is not connected.

: Memory disk use was attempted without executing

CALL MEMINI. 71 Paname across disk : NAME statement was attempted between different disk drives.

COLOR CODE

| code | color | code | color |
|------|--------------|------|--------------|
| 0 | Transparent | 8 | Medium red |
| 1 | Black | 9 | Light red |
| 2 | Medium green | 1Ø | Dark yellow |
| 3 | Light green | 11 | Light yellow |
| 4 | Dark blue | 12 | Dark green |
| 5 | Light blue | 13 | Magenta |
| 6 | Dark red | 14 | Gray |
| 7 | Sky blue | 15 | White |

OPERATORS

| Arithmetic operators | ∧ power — change signs *,/ multiplication, division \ integral division MOD integral residue +,- addition, subtraction (Priority increases from bottom to up) |
|----------------------|---|
| Relational operators | <>= comparison |
| Logical operators | NOT negation AND logical product OR logical sum XOR exclusive logical sum EQV negation of exclusive logical sum IMP implication |