BASIC

*All the animated characters which you can display on-screen in Basic are introduced in the Character Table A on the back cover.

■ About Direct Mode

From now we're going to introduce how to execute BASIC commands easily.

All you need to do is to press the RETURN key after entering the command from the keyboard. If the command is correct, the computer will display OK on the screen. If incorrect, it will display a ?SN ERROR (syntax error) type of error message. In this case use the ▲ ▼ ◀ ▶ keys to move the cursor to the wrong command and after rewriting the correct command, press the RETURN key again. Or, you can also write the right command again and press the RETURN key.

Entering commands directly from the keyboard and having them executed is called Direct Mode.

Let's try to press the keys of the keyboard, make Mario appear on-screen and make him move.

*We're going to use Mario from the animated characters to explain the Program Input Method.

★Preparation to have Mario appear on-screen

Enter

SPRITE ON RETURN

The preparation to have Mario appear on-screen is completed.

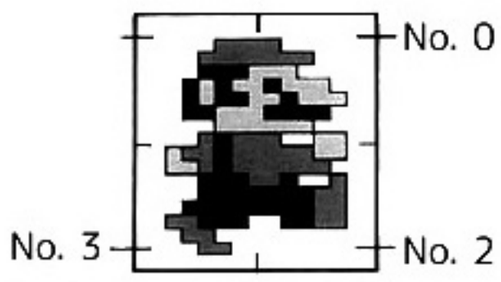
Please enter

DEF SPRITE 0,(0,1,0,1,0)=CHR$(1)+CHR$(0)+CHR$(3)+CHR$(2) RETURN

This line of command is the command which assigns the 4 characters which compose the character (the animated character).

From now on, Mario can be displayed at any time.

★Choosing the animated character...



CHR$(0) : Right Up
CHR$(1) : Left Up
CHR$(2) : Right Down
CHR$(3) : Left Down

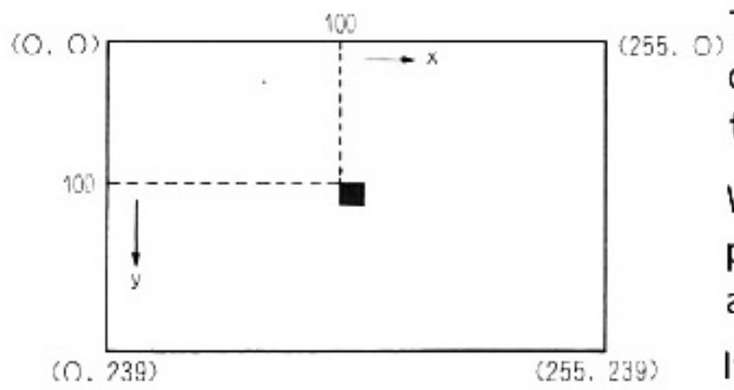
(We're using Mario (Walk 1) from left to right.)

★Let's choose Mario's display position

When you enter

SPRITE 0,100,100 RETURN

Mario will be displayed close to the center of the screen.



This command means that Mario will be displayed at the specified coordinates of the screen (x:100, y:100).

When entering a command, take a look at p. 4's keyboard layout and enter the right alphanumerics and symbols.

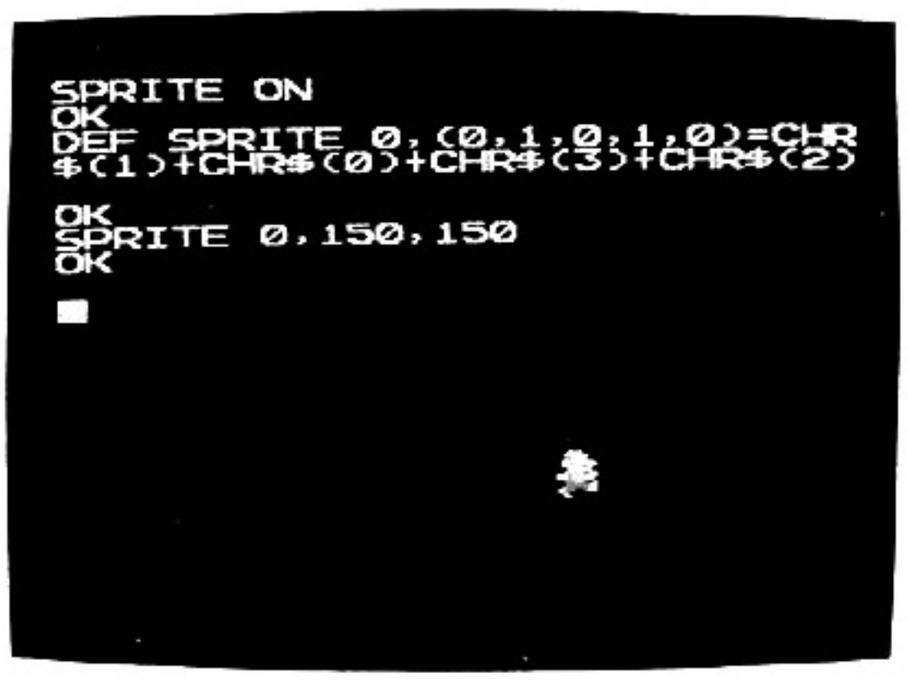
If you made a mistake, use the ▲ ▼ ◀ ▶ keys to move the cursor to the place of mistake and enter the correct command.

★Let's try to change Mario's display position freely

When you enter

SPRITE 0,150,150 RETURN

Mario will be displayed a little further below to the right.



Method to change the SPRITE COMMAND's 100, 100 to 150, 150

▽Press the ▲ key to move the cursor to the SPRITE line.

SPRITE 0, 100, 100

▽Press the ▶ key to move the cursor to the position to correct.

SPRITE 0, 100, 100

▽Press the 5 key, 0 changes into 5.

SPRITE 0, 150, 100

▽Press the ▶ key, move the cursor to another position.

SPRITE 0, 150, 100

▽Press the 5 key, 0 changes into 5.

SPRITE 0, 150, 150

▽When pressing the RETURN key, the command will be rewritten.

When doing the same kind of operation and changing the numbers, you'll be able to display Mario at the position of your choice.

★Change Mario's color

When you enter

CGSET 1,1 RETURN

you will notice that the color in which Mario was displayed has changed and that Mario is now wearing a red hat.

About Program Mode

Until now, we let the computer do the job by entering commands directly into the computer. This is called Direct Mode. In this mode, when entering 1 command, because only 1 job can be executed, in order to have a job executed many times, the command had to be entered as many times as well. If the job's order is being entered into the computer's memory through a command, it won't need and extra effort to execute the same job as many times as desired.
This way of entering the job order into the memory and have it executed is called Program Mode.

★The difference between Program Mode and Direct mode...

```
10 SPRITE ON
20 DEF SPRITE 0,(0,1,0,1,0)=CHR$(1)+CHR$(0)+CHR$(3)+CHR$(2)
30 SPRITE 0,100,100
```

Command Line

Command

Line number

The figure above shows an example of a process to display Mario.
In Program Mode, a number of execution order is added for each command and memorized. In doing so, you can call the memorized programs as many times as you want and execute them easily.
In other words, adding a line number to the command to execute, memorizing it and executing it is called Program Mode. The collection of commands with line numbers is a Program (command line).
The computer executes the program accurately starting with the line with the lowest number.
Like shown in the figure above, when adding the line numbers (integers) while leaving space between the numbers, you can easily add or modify programs. Also, you can not use the same number within one program. Usually, we recommend to increase the line numbers by 10 when creating a program.

★

Before programming, enter

CLS RETURN……(This command erases the alphanumerics, kana's and symbols displayed in the background, or the backdrop, and returns the cursor to the home position (screen up left).)

to erase the commands displayed on-screen.

Or, enter

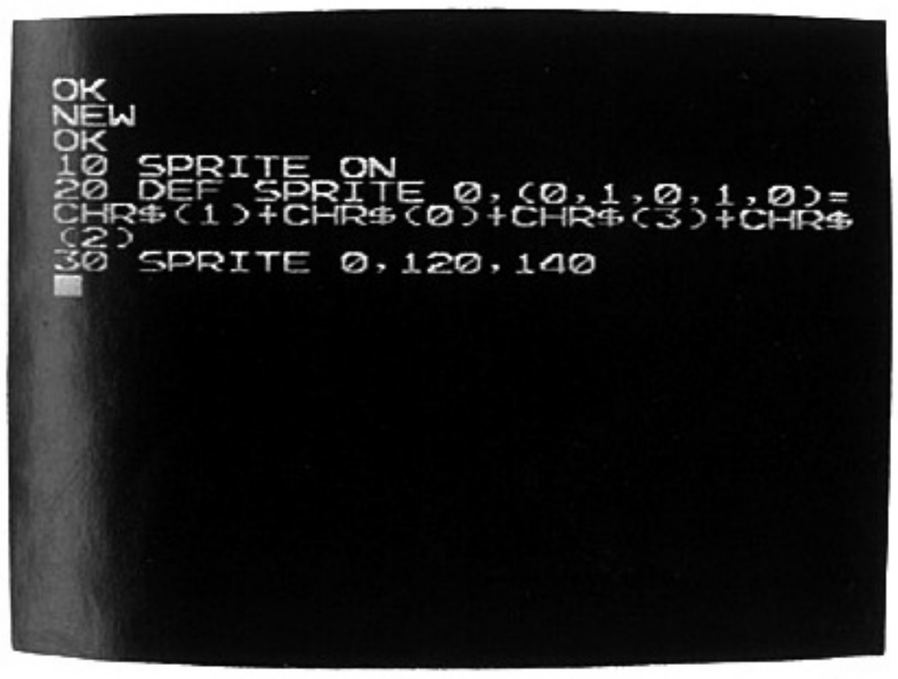
NEW RETURN……(Command to delete all the memorized programs from the memory)

to erase the program entered by the computer from the memory. When erasing previous programs which are unnecessary, please enter this command.

CTR+D……(Press the D key while holding down the CTR key) *Please refer to p. 19.

Erases the sprites, returns the color to the original color sheme. (If Mario was on-screen, he disappears.)

★Let's call Mario in Program Mode



Please enter

```
10 SPRITE ON RETURN
20 DEF SPRITE 0,(0,1,0,1,0)=
   CHR$(1)+CHR$(0)+CHR$(3)
   +CHR$(2) RETURN
30 SPRITE 0,120,140 RETURN
```

Explanation of the program entered above through the line number order.

10 SPRITE ON

Enables the display of animated characters. After enabling it once, it remains effective until you enter the SPRITE OFF command.

20 DEF SPRITE

=Composes and defines an animated character from the expression to the right of it.

DEF SPRITE 0,(0,1,0,1,0)=CHR$( )……

Animated character number
Color
Matching shape
Display priority

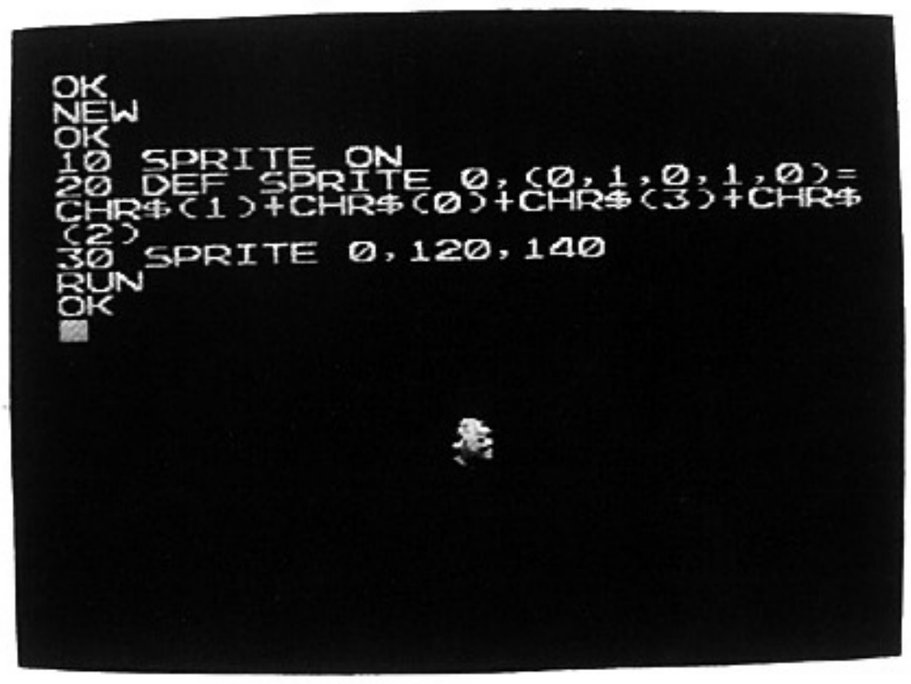
Animated character's inverted instruction's X and Y direction
(In this program, it instructs the inversion of the X direction)

*When you modify the numbers inside CHR$( ), you can switch to a different character.

30 SPRITE 0, 120, 140

Animated character number 0 will be displayed in the position of the X:120 and Y:140 coordinates.

★To run a program...



Please enter

RUN [RETURN]

(You can also execute the command by just pressing the [F8] key)

Now you can see Mario on-screen, right?

●About the Function keys...

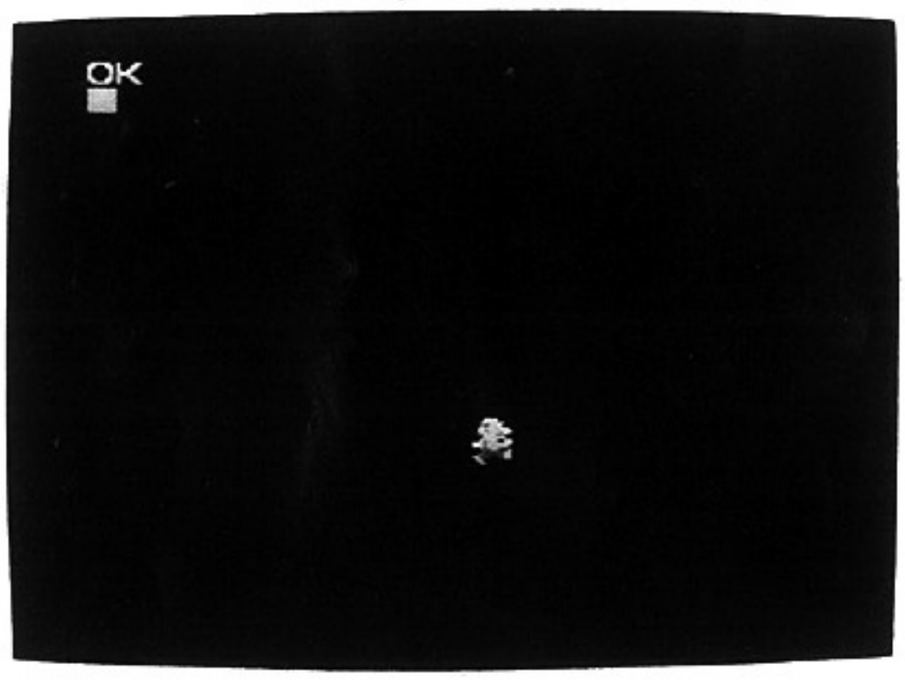
The Function keys from [F1] to [F8] have pre-assigned commands which are useful when creating programs.

[F1]···LOAD [RETURN]	[F5]···SPRITE
[F2]···PRINT	[F6]···CONT [RETURN]
[F3]···GOTO	[F7]···LIST [RETURN]
[F4]···CHR$(	[F8]···RUN [RETURN]

*The content which is assigned to each key gets executed just by pressing one key from [F1] to [F8].
(However, [F1] [F6] [F7] [F8] get displayed and then executed)

*Your program input will progress smoothly by remembering every command.

★To leave only Mario on-screen...



Use the "Let's call Mario in Program Mode" on p. 17 and enter this command:

5 CLS [RETURN]

Please enter:

RUN [RETURN]

The command lines entered previously have disappeared and only Mario remains on-screen, right?

★Check the memorized program

Please enter

LIST [RETURN] ··(Command to display the memorized program on-screen)

A program like the one in the figure on the left gets displayed on-screen.

The computer has added the entered

5 CLS

and memorized it while arranging neatly the program.

Thus, if you enter a commands with line numbers into the program, you will be able to add more programs later on.

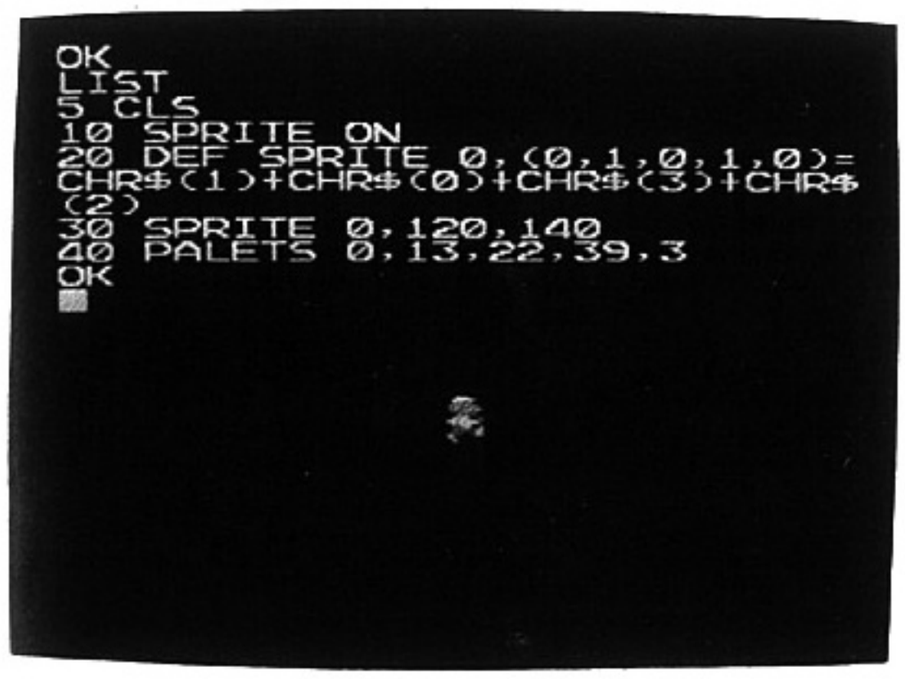
●About LIST...

LIST is the command to display the memorized program on-screen.

LIST m − n
- m: the first line number of the program to display
- n: the last line number of the program to display

LIST [RETURN]	Displays the whole memorized program.
LIST 30 [RETURN]	Displays the program on line number 30.
LIST 20− [RETURN]	Displays the program starting from line number 20.
LIST 20−30 [RETURN]	Displays the program from line number 20 to 30.
LIST −30 [RETURN]	Displays the program from the beginning to line number 30.

★Changing Mario's color



Program 1

Use the "Leave only Mario on-screen..." program on p. 18 and enter

40 CGSET 1,1 RETURN

When you enter

RUN RETURN

the color of Mario from right before will change. Please enter

40 PALETS 0,13,22,39,3 RETURN

When you enter

RUN RETURN

a Mario of a slightly different color will be displayed. Enter

LIST RETURN

and check if the PALETS command which you entered right before has been memorized. When you enter the line number 40's PALETS, the previous program will be rewritten (line number 40's CGSET 1,1).

Press the ▲ ▼ ◀ ▶ keys to move the cursor to the PALETS command line. In order to change the color of Mario and the backdrop, change the PALETS command like below.

40 PALETS 0,17,22,39,3 RETURN

And now enter:

RUN RETURN

The backdrop has a new color, right?

And now we'll revert the original color of Mario. Please enter:

40 PALETS 0,13,54,22,2 RETURN
RUN RETURN

●About CGSET...

CGSET is the command which defines the preferred group of colors among the combination of colors which are pre-provided.

CGSET m, n
- n: Animated character (sprite) palette code (Designs the groups from No. 0 to No. 2)
- m: Background screen palette code (Designs the groups from No. 0 to No. 1)

●About PALET...

PALET is the command which selects among 52 colors and displays the pattern of the animated characters and backdrops. There are 2 types of PALETs.

PALETS·····For animated characters (sprites)

PALETB·····For the backdrop pattern

PALETS 0, 17, 22, 39, 3

- 17: Indicates the color set for the backdrop screen. The color of the backdrop screen becomes active afterwards with an effectively assigned command.
- 22, 39, 3: Indicates the 52 color code. The range of numbers is from 0 to 60.

*Please refer to the color chart on p. 113 or the 52 color code on p.73.

●About CTR+D...

CTR+D means pressing the D key while holding down the CTR key.

When you execute this, you will execute the CGEN 2, SPRITE OFF command.

The color palette becomes the color scheme of the palette code for the background, and for the background + sprite.

Cancels CTR+A (auto insert function).

However, the values of the variables remain unchanged.

If you use this function, in case you change the program, the sprites (animated characters) displayed on-screen will disappear, the color of the font will become white, and therefore, the screen will be easier to see.

*Please refer to CGEN on p. 71.
*Please refer to SPRITE OFF on p. 27.
*Please refer to the Control Code on p. 104.

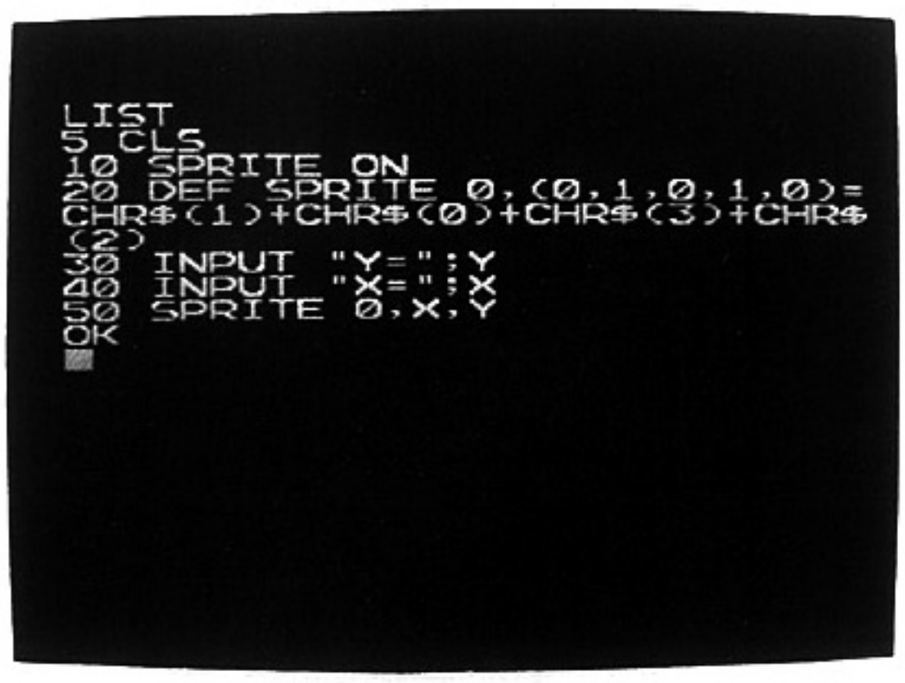
●If the computer's condition becomes incomprehensible...

When executing a program in BASIC, if you can't find the cursor anymore or if you don't know what's going on with the computer, perform the following operations.

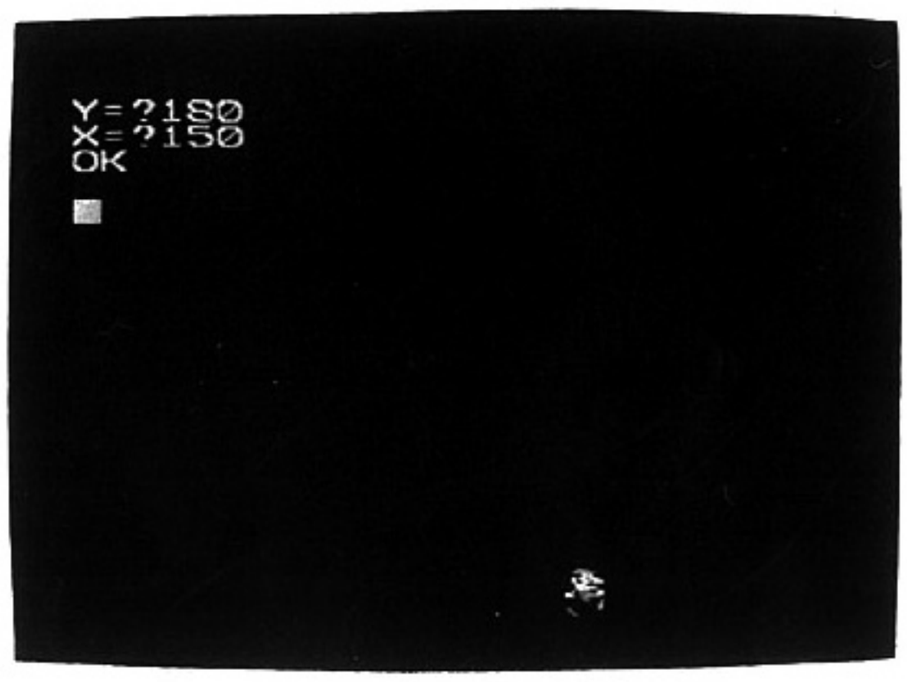
①Press the STOP key
②Press the CTR+D keys
③Press the SHIFT+CLR HOME keys

With these operations, you will go back to the condition where there is nothing on-screen except the cursor blinking in the upper left corner. If, even after performing these operations, the screen doesn't go back to this condition, it has turned into a condition in which the computer can not be controlled (this condition is called "computer on reckless run"). Please press the reset switch of the famicom and restart the operation from the start screen.
Caution: this problem can occur when an error arises in programs which use the POKE or CALL commands.

★Let's use variables to move Mario around



Program 2



Use the program on p. 19 and enter:

30 INPUT"Y=";Y RETURN
40 INPUT"X=";X RETURN
50 SPRITE 0,X,Y RETURN

When you enter

RUN RETURN

Y=? ■

gets displayed and, on the same line, please enter

180 RETURN

Right below,

X=? ■

gets displayed and, on the same line, please enter

150 RETURN

Mario will be displayed on the location of the (150, 180) coordinates.
To execute it as many times as you want, please enter

RUN RETURN

There's a limit to the numbers which you can use in the commands.
X : Ø~255, Y : Ø~255(however, it can only display X:0~240, Y5~220)

*The display range may vary depending on the TV set.

● About INPUT...

INPUT is the command which, when waiting for the data input of a number or a letter from the keyboard, when entering it, substitutes that data in the assigned variable.

A " (double quoatation) mark is used in the INPUT command in the program above.

INPUT "Y=";Y

The Y= symbol enclosed by " is displayed as is on-screen.

*Please refer to p. 60.

● About PRINT...

PRINT is the command which displays letters or the result of a calculation on-screen.

PRINT

Creates 1 blank line.

A$="MARIO SAMPLE V.100.Ø":B$="ノ プログラム"
PRINT A$+B$

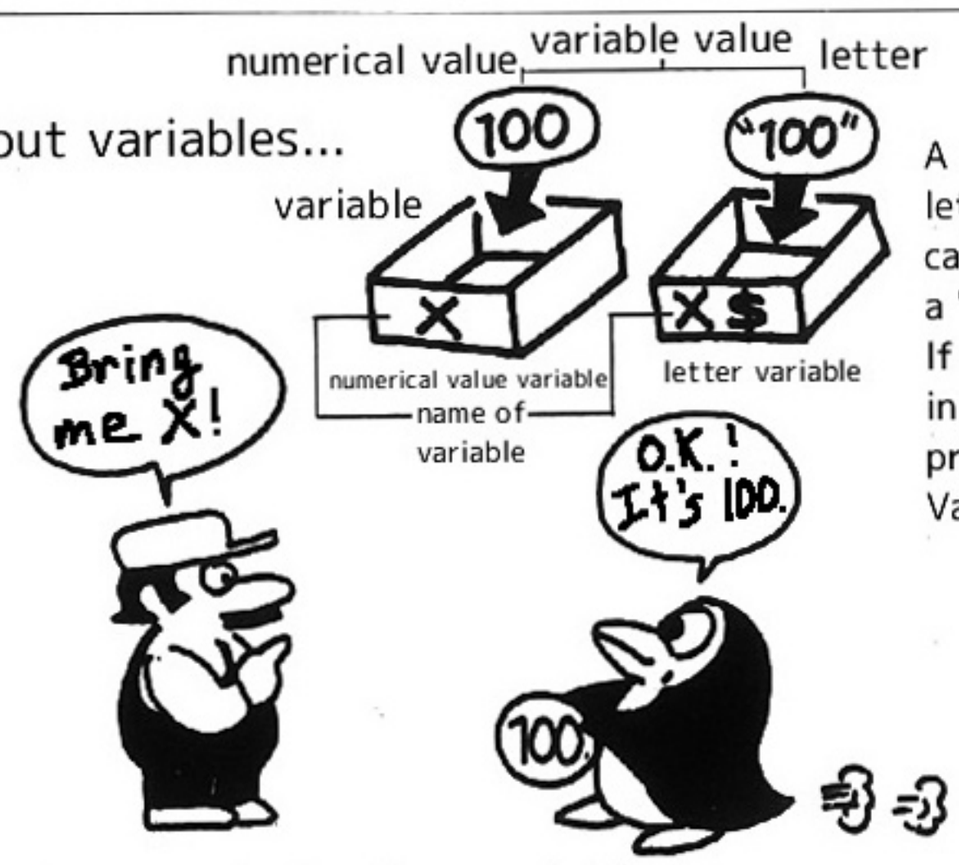
Displays the variable letters A$ and B$ on 1 continuous line. (displays as MARIO SAMPLE V.100.0ノ プログラム)

Enter: PRINT 100+100 RETURN

It calculates 100+100 and displays 200. (Direct Command)

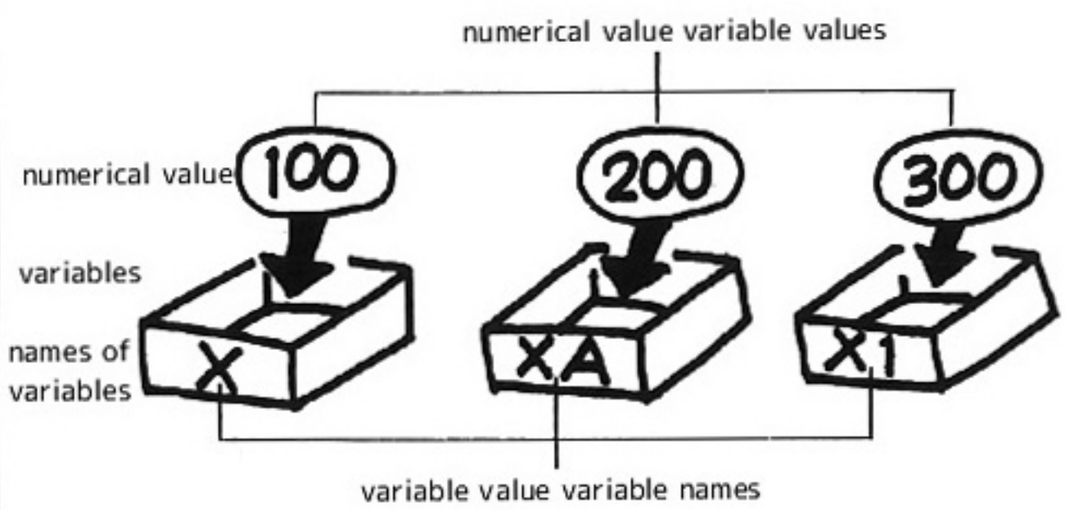
*Please refer to p. 59.

●About variables...



A variable is a box in which you can enter numerical values/ letters. We give this box a name to distinguish it. In this case, we call the box a "name of variable" and the value content of the box a "variable value".
If we give it a name of variable and if we enter a variable value in the box, just by assigning the name of the variable, we can produce the variable value from inside the box.
Variables include numerical value variables and letter variables.

About numerical value variables...



●Numerical value variables' variable names are named with symbols such as X and Y. Variable names can contain up to 255 symbols, however, the computer can only use the first 2 letters to distinguish the variable names. (Even if you use long names, the computer will not discern more than 2 letters)
Also, the first letter symbol has to be an alphabetic letter.
However, even if you use alphabetic letters, you can not use BASIC commands (such as LIST, RUN, PRINT, etc.)

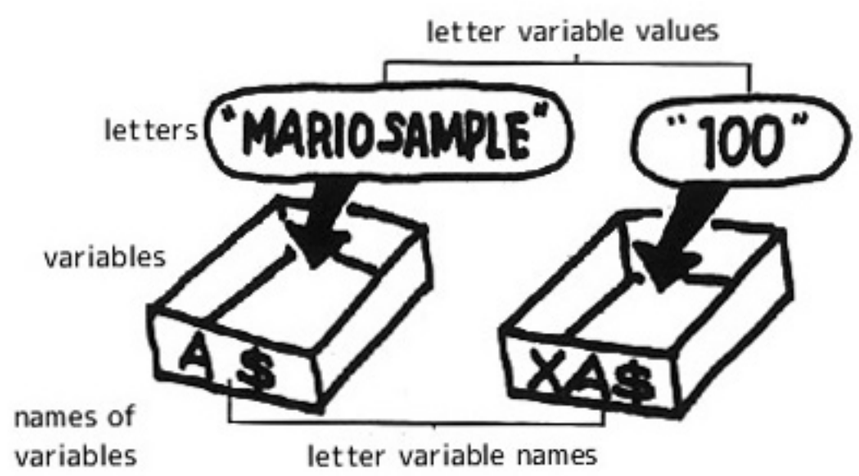
● We use integers which express 100, 200 etc. for the variable values. For example:

X=100 , XA=200 , X1=300

name of numerical value variables
value of numerical value variables

Still, before entering data into the numerical value variables, the variable values contain the numerical value 0.

About letter variables...



●When entering a $ (dollar mark) after symbols such as X or Y in the variable names of the letter variables, we can distinguish them from numerical variables. Just like the names of the numerical value variables, the computer can not distinguish more than the first 2 letters in the variable name although it can accept up to 255 symbols.
Also, the first symbol has to be an alphabetic letter.

●Variable values are entered between " (double quotation marks). These variable values differ from the numerical value variables in that even if they're pure numerics, they're treated as letters. For example:

A$="MARIO SAMPLE" XA$="100"

letter variable names
letter variable values

Still, the boxes remain empty until the variable values receive data.

Try to enter the program which uses letter variables.

Program

```
LIST
5 CLS
10 SPRITE ON
20 DEF SPRITE 0,(0,1,0,1,0)=
CHR$(1)+CHR$(0)+CHR$(3)+CHR$
(2)
30 INPUT "Y=";Y
40 INPUT "X=";X
50 SPRITE 0,X,Y
60 PRINT
70 A$="MARIO SAMPLE V.100.0"

80 B$="ノ プログラム"
90 PRINT A$+B$
OK
```

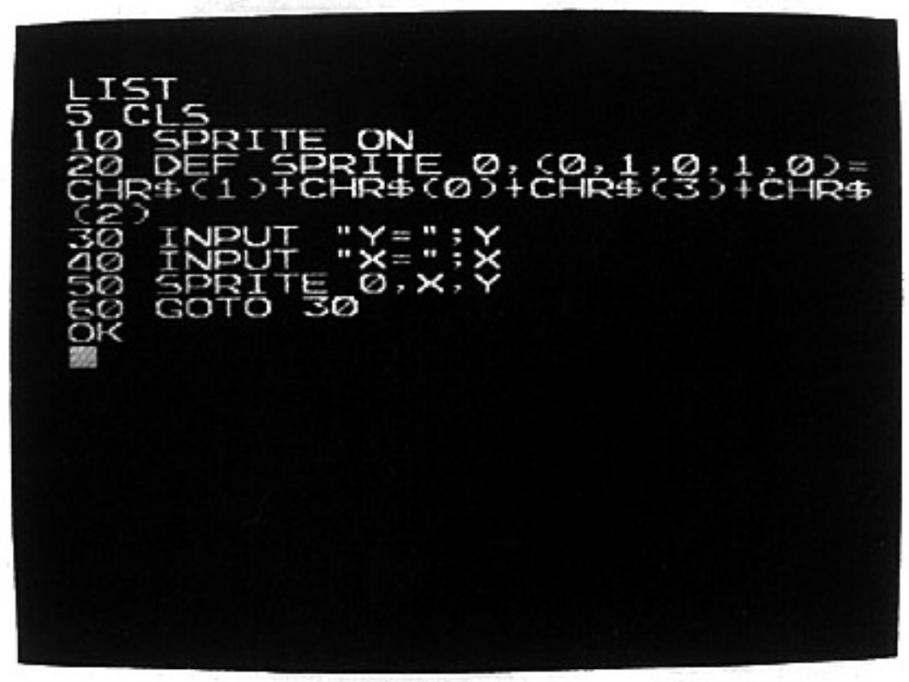
Example of execution screen

```
Y=?150
X=?100

MARIO SAMPLE V.100.0ノ プログラム
OK
```

The letter variable names used in this program are A$ and B$.
The letter variable values are "MARIO SAMPLE V.100.0" and "ノ　プログラム".

◆Learning repetition techniques

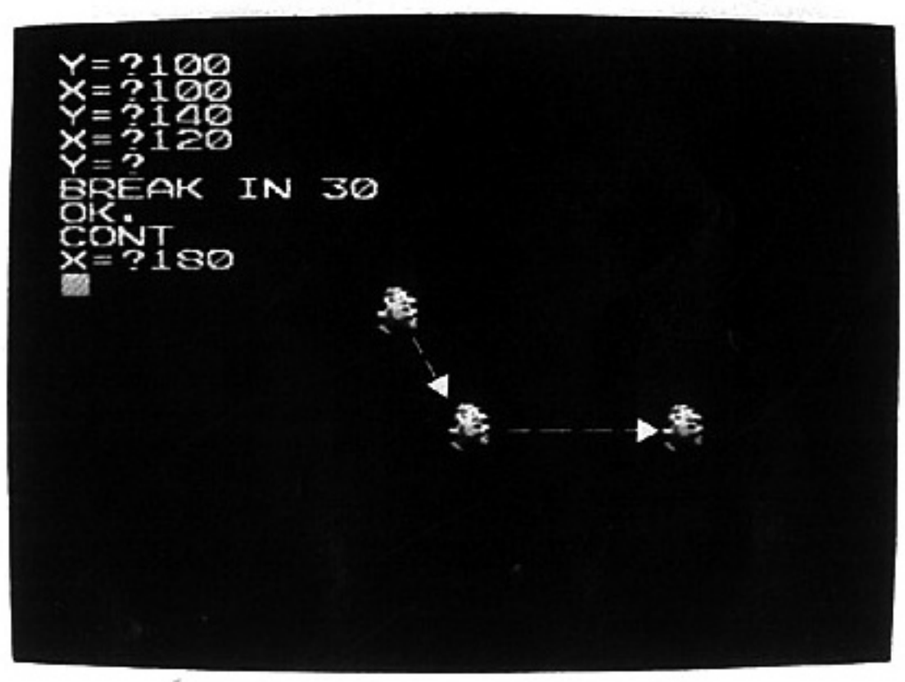
★Let's try to change Mario's positon one after another using the GOTO instruction



Program 3

Use program 2 on p. 20 and enter:

60 GOTO 30 RETURN



Please enter:

RUN RETURN

Y=? appears and enter:

100 RETURN

X=? appears and enter:

100 RETURN

Mario is displayed in the (100, 100) position.
When Y=? appears again, enter:

140 RETURN

X=? appears and enter:

120 RETURN

Mario's display position changes from (100, 100) to (120, 140).

When Y=? appears again, press the STOP key if you want to stop the execution of the program.

BREAK IN 30……(the program was halted at line number 30.) appears.

This shows that the program was halted while executing the command on line number 30. Please enter:

CONT RETURN……(This is the command to continue the execution of a halted program.)

Because X=? is displayed again, please enter:

180 RETURN

Mario's position changes from (120, 140) to (180, 140).

●About GOTO

GOTO is the command which repeats the execution of a program by forcing a jump to the defined line number.

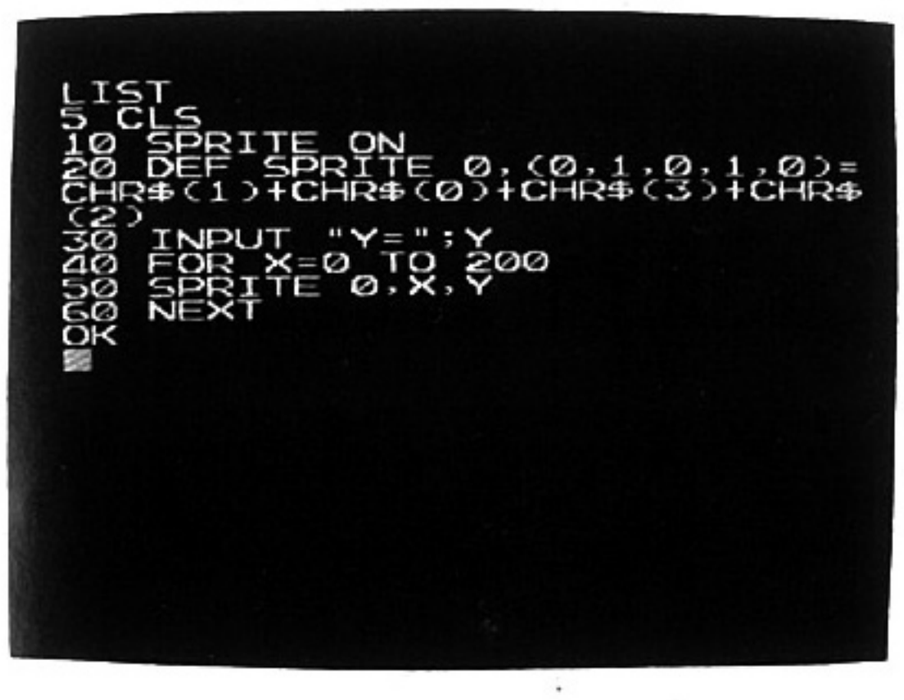
Repeated infinitely

```
30 INPUT "Y=";Y
40 INPUT "X=";X
50 SPRITE 0,X,Y
60 GOTO 30
```

When displaying Mario on line number 50, it will force a jump to line number 30 and execute.

*Please refer to p. 63.

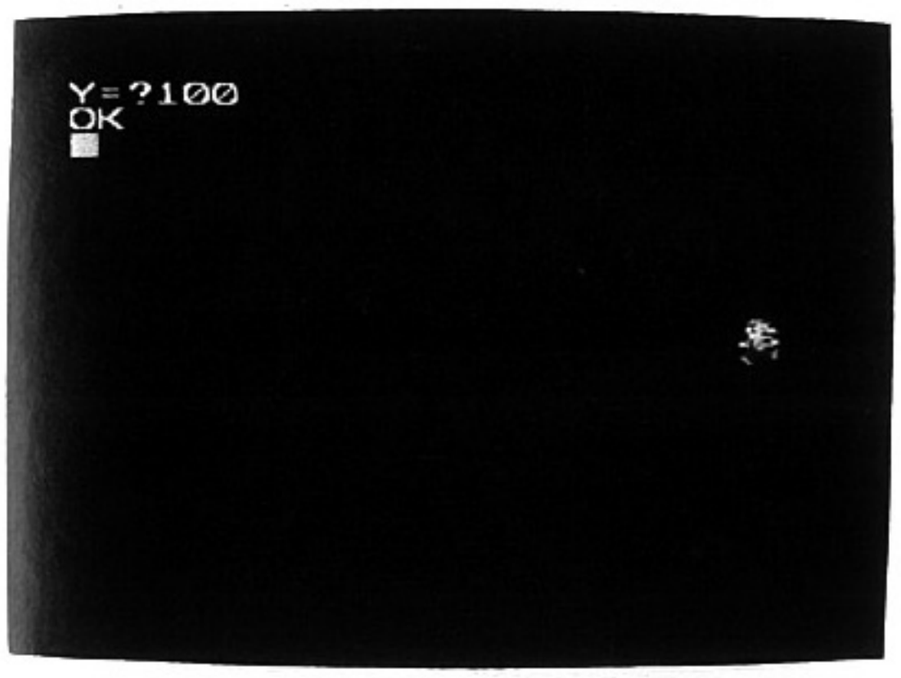
★Let's make Mario move horizontally with the FOR NEXT instruction.



Please use program 3 on p. 22 and enter:

40 FOR X=0 TO 200 RETURN
60 NEXT RETURN

*FOR NEXT is explained on the next page in the "About multiple loops" section.



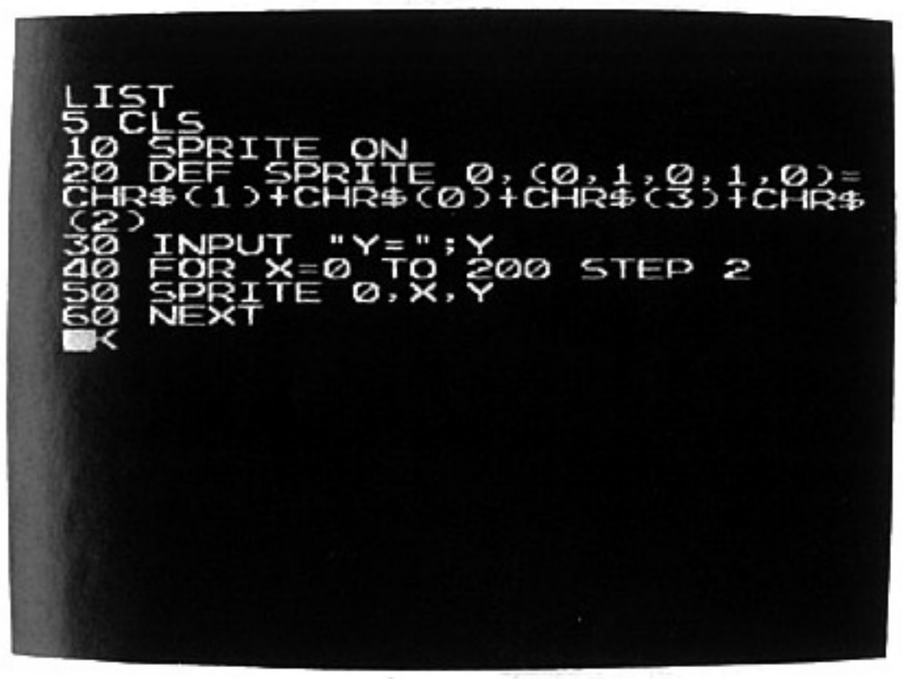
Please enter:

RUN RETURN

When Y=?, please enter:

100 RETURN

Mario will move from left to right.



Please enter:

LIST RETURN

Use the ▲ ▶ keys to move the cursor to line number 40 and rewrite the command.

40 FOR X=0 TO 200 STEP 2 RETURN

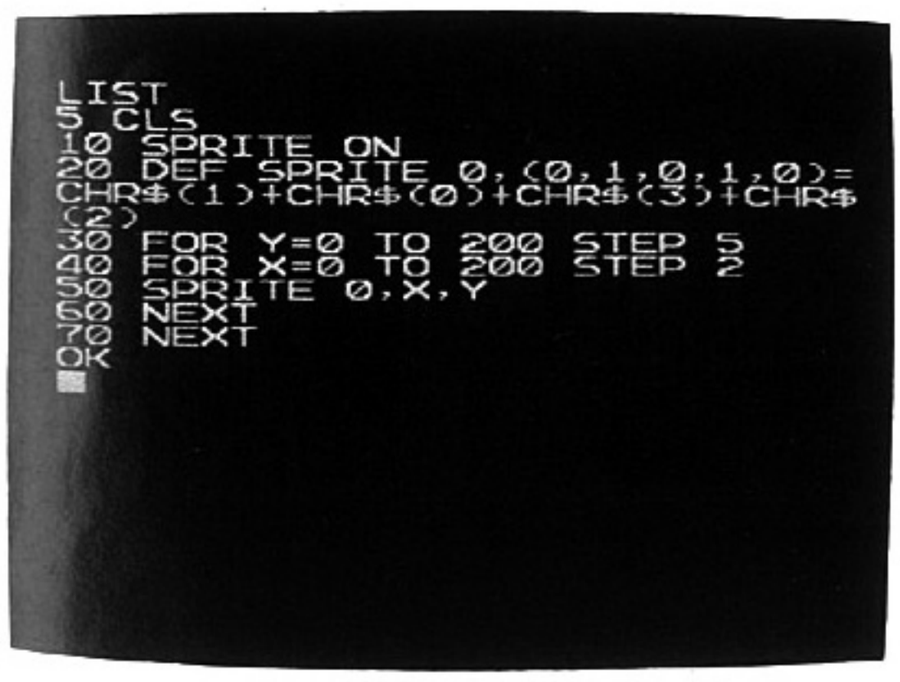
Next, press the ▼ key, move the cursor to the line below line number 60. Enter

RUN RETURN

and because Y=? is displayed, please enter:

100 RETURN

You will notice that Mario moves faster than before from left to right.



Program 4

Please enter:

30 FOR Y=0 TO 200 STEP 5 RETURN
70 NEXT RETURN

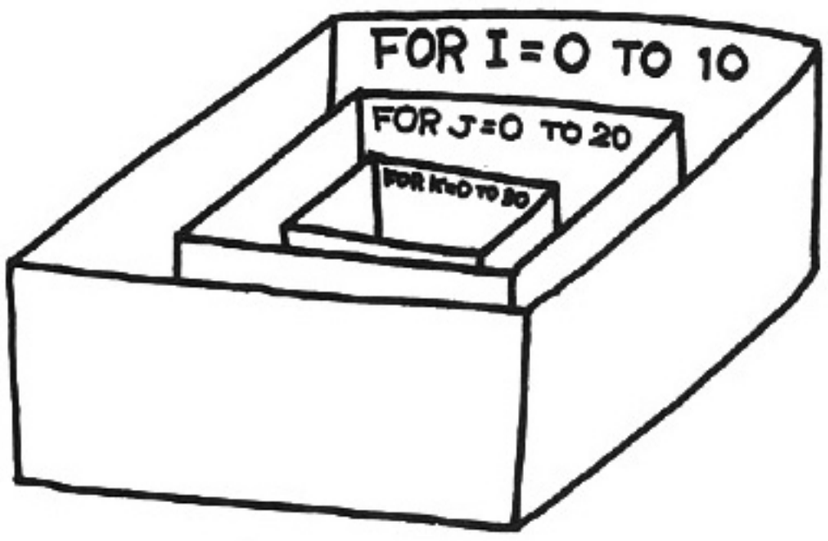
When you enter:

RUN RETURN

Mario moves from the upper left of the screen to the lower right of the screen.

●About multiple loops...

FOR~NEXT repeats and executes a program only for a limited number of times. In other words, the loop of FOR~NEXT only makes up 1 loop. Within this loop, you can use 1 extra FOR~NEXT. Multiple loops is the name given to the fact that you can create any number of FOR~NEXT loops within a FOR~NEXT loop.




```
①FOR I=0 TO 10
②FOR J=0 TO 20
③FOR K=0 TO 30
  :
 NEXT
  :
NEXT
  :
NEXT
```

Loops

Multiple loops are executed in the order which starts with the FOR~NEXT which is located the furthest outside. However, as a program, you can not assign 1 loop to cross another loop.

```
FOR I=0 TO 100
  :
 FOR J=0 TO 30
  :
 NEXT
  :
NEXT
```

Also, you can not use a GOTO command to cut in the middle of a FOR~NEXT instruction.

*Please refer to p. 65. (FOR~TO~STEP NEXT)

●About program content modification/revision/addition...

1. When you want to erase a specific line.

Type the number of the line you would like to erase and press [RETURN]. On-screen it will remain as is, but the command on that line will not be executed.

*When calling the program by pressing LIST [RETURN], you can check if the line number has disappeared.

2. When you want to overwrite a specific line.

Re-type the line number and type the command. If the command of the previous line number remains and the numbers overlap one another, it executes what was written afterwards.

*Use LIST [RETURN] to check.

3. When you want to modify/revise/add a part.

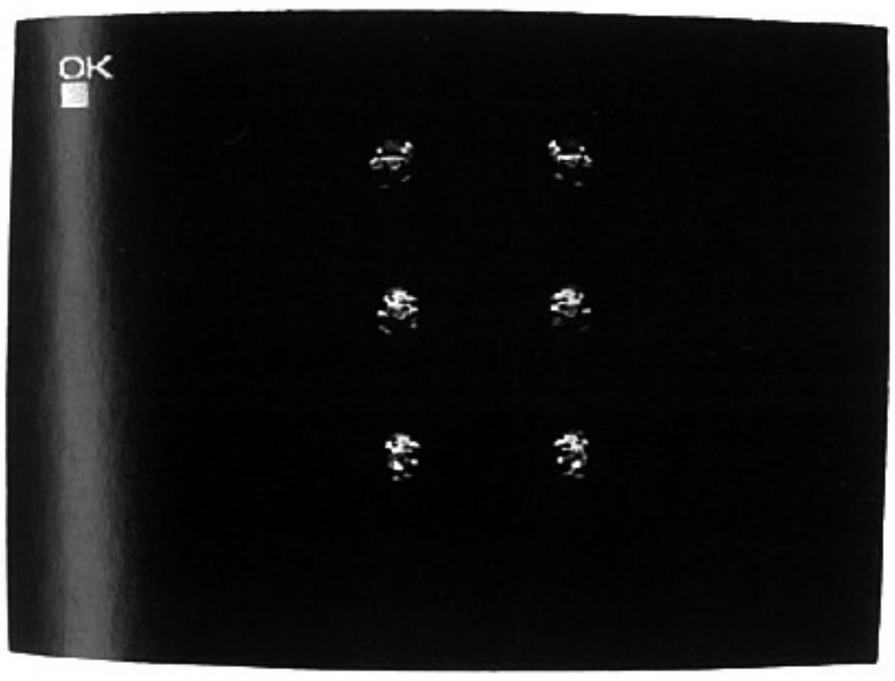
Use the [INS] [DEL] keys.

*Please refer to P. 5 for the detailed usage.

★Let's display a Mario of several poses.

Let's try to display a Mario of several poses on-screen.
Use the program on p. 23 and enter the following program.

```
15 CGSET 1,0 RETURN
21 DEF SPRITE 1,(0,1,0,0,0)=CHR$(0)+CHR$(1)+CHR$(2)+
   CHR$(3) RETURN
22 DEF SPRITE 2,(0,1,0,1,0)=CHR$(5)+CHR$(4)+CHR$(7)+
   CHR$(6) RETURN
23 DEF SPRITE 3,(0,1,0,0,0)=CHR$(4)+CHR$(5)+CHR$(6)+
   CHR$(7) RETURN
24 DEF SPRITE 4,(0,1,0,0,0)=CHR$(20)+CHR$(21)+CHR$(22)
   +CHR$(23) RETURN
25 DEF SPRITE 5,(0,1,0,1,0)=CHR$(21)+CHR$(20)+CHR$(23)
   +CHR$(22) RETURN
30 SPRITE 0,100,100 RETURN
40 SPRITE 1,150,100 RETURN
50 SPRITE 2,100,150 RETURN
60 SPRITE 3,150,150 RETURN
70 SPRITE 4,100,50 RETURN
80 SPRITE 5,150,50 RETURN
```



RUN RETURN

A Mario with 6 types of poses will be displayed on-screen.
*Please refer to "Character Table A" on the back cover to know more about the different poses of Mario.

If you have read up to here, you probably already have an idea of what a BASIC program is and how to make it work. In other words, you have learnt the basic matters of BASIC.
You already are a BASIC programmer. You certainly are able to create simple programs. Please give it a try!
Hereafter, from p. 26 to p. 30, we will develop the basics of BASIC and explain about a program in which you can move an animated character freely using a controller.

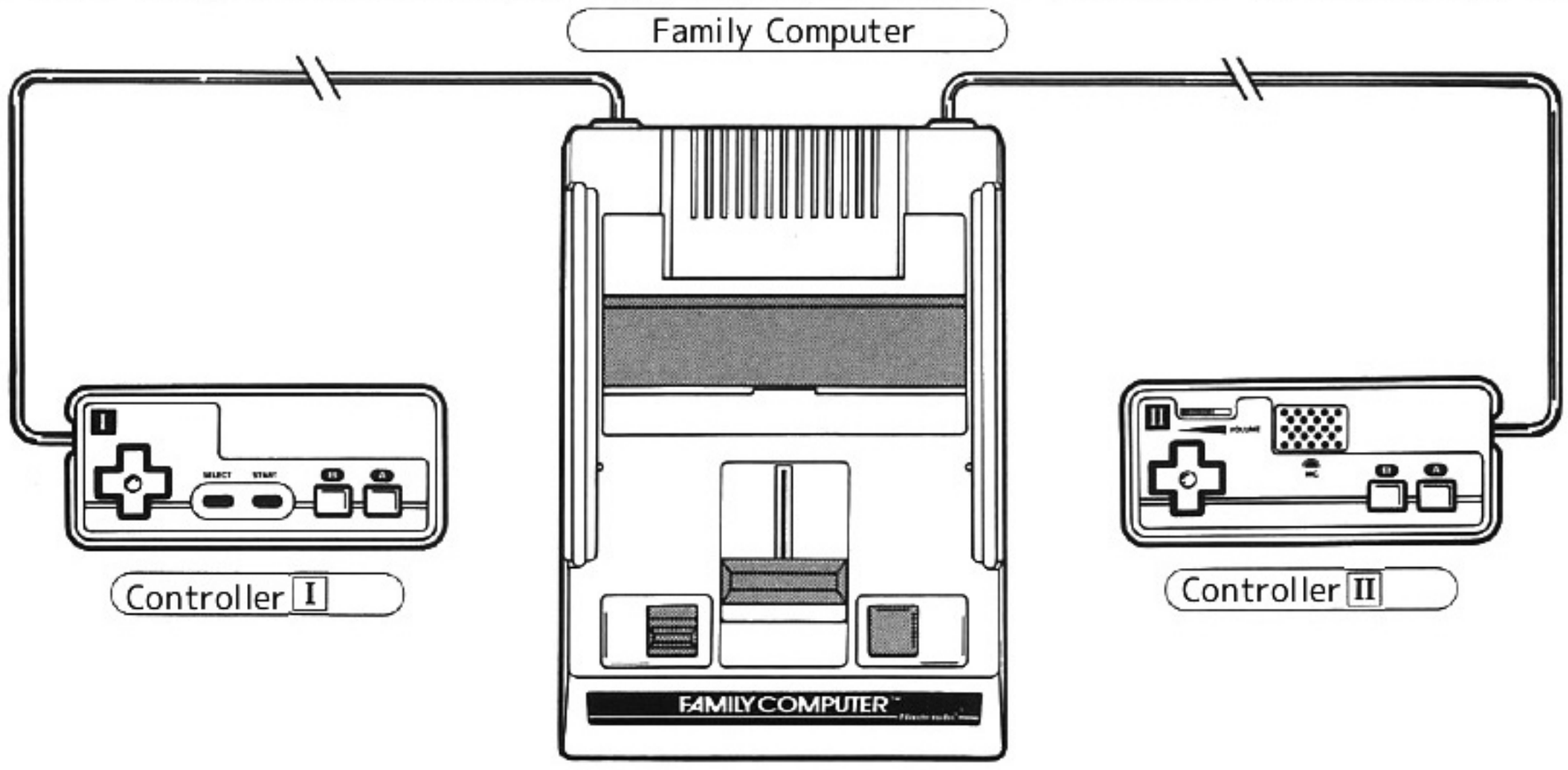
✻About the method to display an animated character

●DEF MOVE

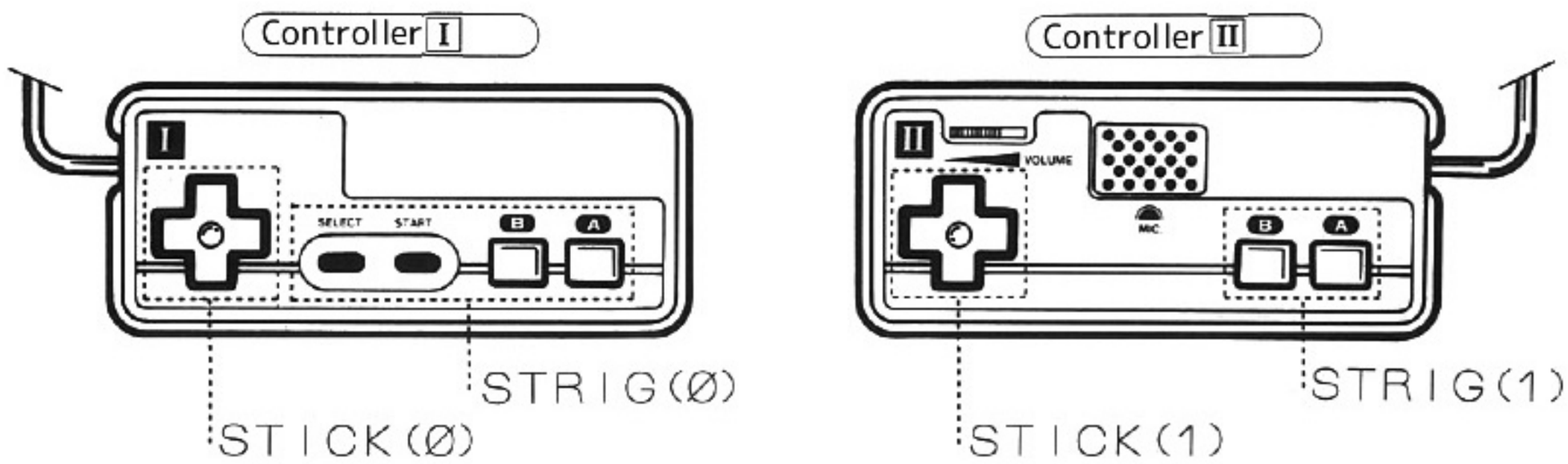
Until now we have explained about the DEF SPRITE (define sprite) command, but we're going to introduce an extra one.
It is called the DEF MOVE (define move) command.
-This is a smart command which moves easily the 16 types of animated characters on Character Table A.
-If you assign the 16 types with the DEF MOVE command, the animated characters will be assembled automatically and you can display their movement.
-You can assign the direction, speed, movement distance, color and display priority.
-The usage of this command is explained in the "About the MOVE command" chapter. *Please refer to p. 32 for more details.
-With the DEF SPRITE command explained in the beginning, you can compose the animated characters freely, in addition, when you use the simple DEF MOVE command, simultaneously, you will also be able to display a total of 16 different characters.

◆Let's move it with the controller

Let's expand the fun of the games even more by making game programs in which you can use the controller, or even games for two players!



There are 2 commands to operate the controller: STICK and STRIG. STICK is the command which gives the computer the numerical value of the part (direction) of the controller's button pressed on the controller. STRIG is the command which gives the computer the numerical value of the 4 types of buttons of the controller (start/select/a/b) pressed.
When you use controller I, you use the STICK(0) and STRIG(0) commands, when you use controller II, you use the STICK(1) and STRIG(1) commands.

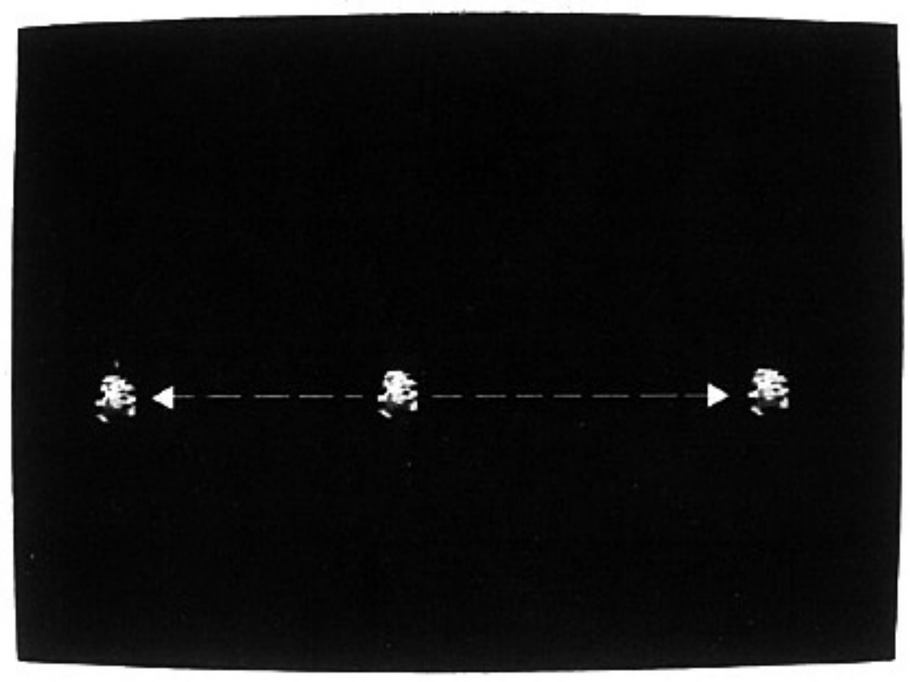


★Let's move Mario left and right with the controller

SPRITE OFF RETURN is entered to erase the displayed Mario.
Use the "Let's display a Mario of several poses" program on p. 25 and enter the following program.

```
26 X=100 RETURN
30 S=STICK(0) RETURN
40 IF S>2 THEN 100 RETURN
50 IF S=2 THEN X=X-1 RETURN
60 IF S=1 THEN X=X+1 RETURN
70 IF X>250 THEN X=X-240 RETURN
80 IF X<5 THEN X=X+245 RETURN
90 SPRITE 0,X,150 RETURN
100 GOTO 30 RETURN
```

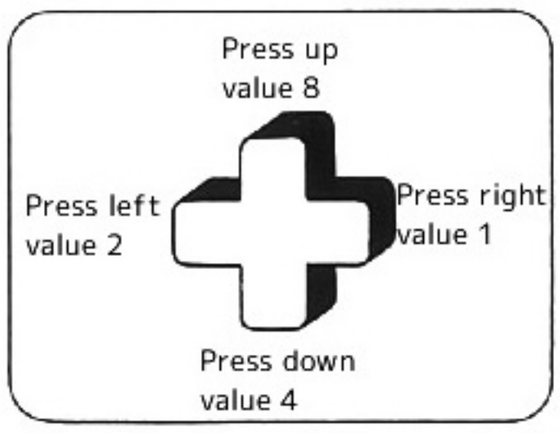
Please enter:
RUN RETURN



Press the button (left side or right side) of controller I, Mario moves left or right.
When you press the STOP key, the cursor will appear.

●About STICK...

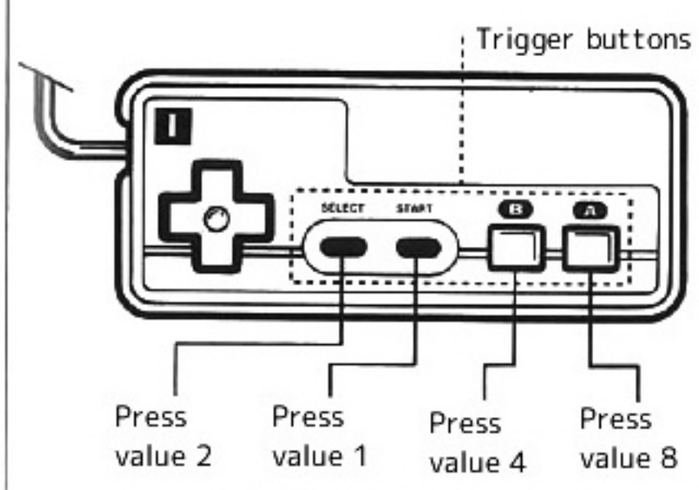
STICK is the command which gives the computer a numerical value when pressing any direction of the button on the controller.



The button of the controller

S=STICK(0) is... controller Ⅰ (STICK(1) is controller Ⅱ)
When the button is pressed to the left, S receives value 1.
When the button is pressed to the right, S receives value 2.
When the button is pressed downwards, S receives value 4.
When the button is pressed upwards, S receives value 8.
By using the STICK command, you are able to move the animated characters freely up-down-left-right while pressing the button on the controller.
*Please refer to p. 86.

●About STRIG...



STRIG is the command which gives the computer a numerical value when pressing any of the 4 types of trigger buttons (start/select/a/b).

STRIG(0)... controller Ⅰ (STRIG(1) is controller Ⅱ)
When the SELECT button is pressed, it receives value 2.
When the START button is pressed, it receives value 1.
When the B button is pressed, it receives value 4.
When the A button is pressed, it receives value 8.
When no button is pressed, it receives value 0.

According to the STRIG command, or in other words, the operation of the trigger buttons, you can stop Mario's movement or even use them to finish the game.
*Please refer to p. 86.

● About IF~THEN...

The IF instruction interprets the condition written between IF and THEN and, according to the result, assigns which command (line) to execute next.

```
40 IF [S>2] THEN [100]
50 IF [S=2] THEN [X=X-1]
```

Condition Command line

In other words, the IF instruction makes the computer interpret it by itself and put out the execution.

Let's explain the IF instruction used in a program which moves Mario left and right with the controller.
On line number 40, if variable S is greater than value 2 (the button is pressed up or down), the command written after THEN is executed, jumps to line 100 and executes the program from there. When it does not meet the conditions (the value of S is lower than or equal to 2), the command after THEN is not executed and it executes line number 50.
On line number 50, if the value of variable S is 2 (the button is pressed to the left), the new value of X is decided by subtracting 1 from the value of X by X=X-1 . (Substracts 1 from the value of the X coordinate to move Mario to the left)

*Please refer to p. 64.

●About SPRITE OFF...

SPRITE OFF is the command to erase all the animated characters displayed on the Sprite screen. In other words, it halts the display of the overlapping of the Sprite screen over the Background screen.
Also, when you only want to erase an animated character specified with a number, displayed on the Sprite screen, enter:

SPRITE n ·····(erases the animated character specified by number n)

Like this you see that there are 2 methods to erase animated characters from the screen.
*Please refer to p. 89.

★Let's make Mario walk with the controller

Enter the following program to show Mario walking to the left or to the right.
Halt the "Let's move Mario left and right with the controller" program of p. 26 and enter the lines from number 30 to 2020 from Program 5 below.
*In case the cursor or the program does not appear on screen, press the STOP key to make the cursor appear and use LIST to call the program.

```
5 CLS                                          ....Erases the display on-screen.
10 SPRITE ON                                   ....Prepares the display of animated characters on-screen.
15 CGSET 1,0                                   ....Fixes the color of the animated characters and the backdrop.
20 DEF SPRITE 0,(0,1,0,1,0)=CHR$(1)
   +CHR$(0)+CHR$(3)+CHR$(2)                    ....Fixes the pattern of WALK1 Mario (facing left) (Mario No. 0)
21 DEF SPRITE 1,(0,1,0,0,0)=CHR$(0)
   +CHR$(1)+CHR$(2)+CHR$(3)                    ....Fixes the pattern of WALK1 Mario (Mario No. 1).
22 DEF SPRITE 2,(0,1,0,1,0)=CHR$(5)
   +CHR$(4)+CHR$(7)+CHR$(6)                    ....Fixes the pattern of WALK2 Mario (facing right) (Mario No. 2).
23 DEF SPRITE 3,(0,1,0,0,0)=CHR$(4)
   +CHR$(5)+CHR$(6)+CHR$(7)                    ....Fixes the pattern of WALK2 Mario (Mario No. 3).
24 DEF SPRITE 4,(0,1,0,0,0)=CHR$(20)
   +CHR$(21)+CHR$(22)+CHR$(23)                 ....Fixes the pattern of ladder Mario (Mario No. 4).
25 DEF SPRITE 5,(0,1,0,1,0)=CHR$(21)
   +CHR$(20)+CHR$(23)+CHR$(22)                 ....Fixes the pattern of ladder Mario (left-right inverted) (Mario No. 5).
26 X=100                                       ....Fixes the X coordinate to 100.
30 READ X,Y,A,B,C,D,E,F                        ....Reads the data from X to F.
40 DATA 120,140,1,3,0,2,4,5                    ....(enters a value in the number (variable) of each type of Mario and the position coordinates which display Mario)
                                               ....Data read by the READ command.
50 SPRITE A,X,Y                                ....Displays Mario No. 1 at the (X:120, Y:140) coordinate position.
60 S=STICK(0):IF S=0 THEN 60                   ....Checks if the button has been pressed. If it hasn't been pressed (S=0), it re-executes line number 60.
70 IF S>2 THEN 60                              ....When the button is pressed up or down (Y direction) it jumps to line no. 60.
80 X=X+2:IF S=2 THEN X=X-4                     ....When facing right, it adds 2 to the value of X. When facing left, it subtracts 4 from the value of X.
90 IF S=1 THEN 1000                            ....Jumps to line number 1000 when facing right.
100 IF S=2 THEN 2000                           ....Jumps to line number 2000 when facing left.
1000 PAUSE 5:SPRITE C,X,Y:SWAP C,D
1010 SPRITE A:SPRITE B:SPRITE
     C:SPRITE E:SPRITE F
1020 GOTO 60
2000 PAUSE 5:SPRITE A,X,Y:SWAP A,B
2010 SPRITE A:SPRITE C:SPRITE
     D:SPRITE E:SPRITE F
2020 GOTO 60
```

Refer to back cover "Character Table A".

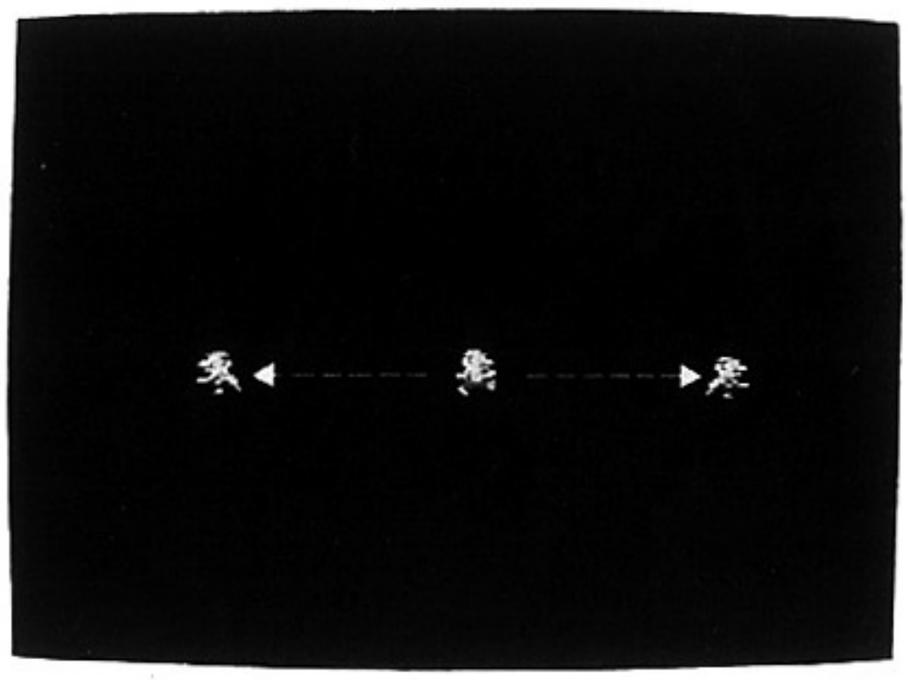
X coordinate handling

Decisive conditions

Displays Mario No. 0. When right is pressed on the button, Mario No. 0 and No. 2 are swapped. The other Mario characters (1, 3, 0, 4 and 5) disappear.

Displays Mario No. 1. When left is pressed on the button, Mario No. 1 and No. 3 are swapped. The other Mario characters (1, 0, 2, 4 and 5) disappear.

Program 5



Please enter:
RUN RETURN

When you press the button on controller I (left side or right side) you can see that Mario is walking smoothly to the left or to the right.

*Try not to let Mario stick out to the left or to the right of the screen. If Mario sticks out, an error araises and Mario stops.
Please enter RUN RETURN again to move Mario once more.

●About SWAP...

SWAP is the command which exchanges the values between variables.
The exchanged variables have to be both numerical value variables or both letter variables or an error arises.
You can not use it as SWAP A, A$ or SWAP A$, B.

SWAP A , B ····· (Exchanges the values between the numerical value variable A and the numerical value variable B. The value that B contained becomes A, the value that A contained becomes B)

SWAP A$, B$ ··· (Exchanges the values between the letter variable A$ and the letter variable B$)

*Please refer to p. 67.

●About multi-statement...

In Program 5, the use of 2 or more commands and a : (colon) within 1 line number is called multi-statement. Within one line number you can write a command of up to 256 letters, colon included.

●About READ~DATA...

READ is the command which reads the data prepared by DATA. Also, DATA is the command which prepares the data read by READ. Like this, READ and DATA are always used together. Moreover, DATA can be placed anywhere withing a program.

Reads the data from X to F (Mario's display position coordinates and each type of Mario's number).

```
READ X,Y,A,B,C,D,E,F
```

```
DATA 120,140,1,3,0,2,4,5
```

Data read by READ

Write a variable (numerical value variable name or letter variable name) after READ and write the corresponding constant data (numerical value variable value or letter variable value) after DATA. For this reason, the variable for READ and the constant data for DATA correspond to each other on a 1 on 1 basis and they both must have the same form.

*Please refer to p. 68.

●About PAUSE...

PAUSE is the command which halts temporarily the execution of the program.

PAUSE n

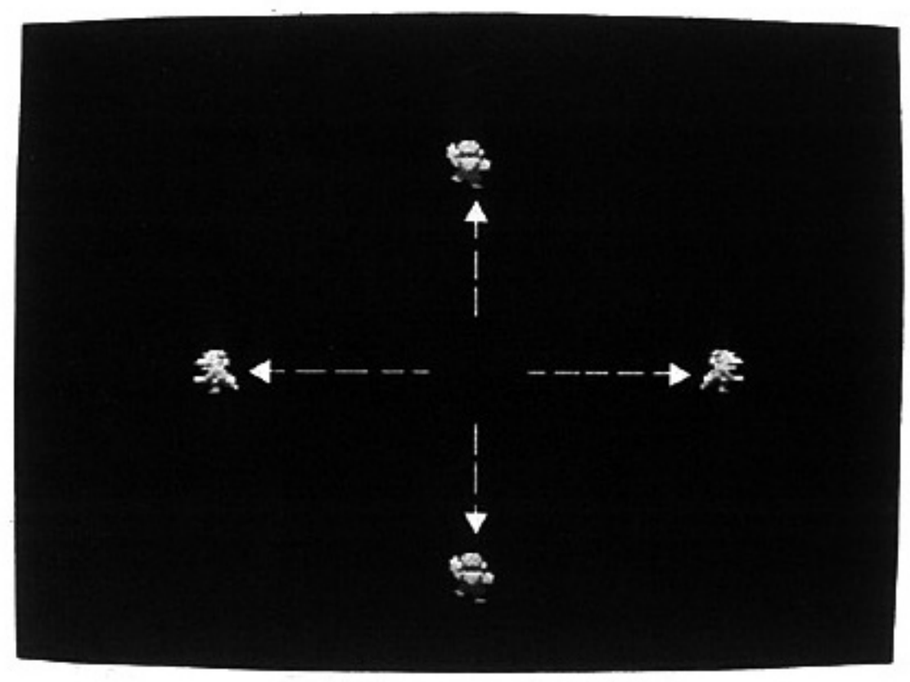
Time to halt the execution of the program (assigned from 0 to 32767)

When n (time to halt the execution of the program) is omitted, there is a function which halts the execution of the program at the place where PAUSE is inserted until any key is pressed. When any key is pressed, the execution of the program continues right after the place where PAUSE is inserted.

*Please refer to p. 78.

When you want to move Mario freely up-down-left-right, use program 5 on p. 28 and enter the lines from no. 60 to 130 and from 3000 to 3020.

```
5 CLS
10 SPRITE ON
15 CGSET 1,0
20 DEF SPRITE 0,(0,1,0,1,0)=CHR$(1)+CHR$(0)
                            +CHR$(3)+CHR$(2)
21 DEF SPRITE 1,(0,1,0,0,0)=CHR$(0)+CHR$(1)
                            +CHR$(2)+CHR$(3)
22 DEF SPRITE 2,(0,1,0,1,0)=CHR$(5)+CHR$(4)
                            +CHR$(7)+CHR$(6)
23 DEF SPRITE 3,(0,1,0,0,0)=CHR$(4)+CHR$(5)
                            +CHR$(6)+CHR$(7)
24 DEF SPRITE 4,(0,1,0,0,0)=CHR$(20)+CHR$(21)
                            +CHR$(22)+CHR$(23)
25 DEF SPRITE 5,(0,1,0,1,0)=CHR$(21)+CHR$(20)
                            +CHR$(23)+CHR$(22)
26 X=100
30 READ X,Y,A,B,C,D,E,F
40 DATA 120,140,1,3,0,2,4,5
50 SPRITE A,X,Y
60 S=STICK(0):IF STRIG(0)<>0 THEN END
65 IF S=0 THEN 60
70 IF S>2 THEN 100
80 X=X+2:IF S=2 THEN X=X-4
90 GOTO 101
100 Y=Y+2:IF S=8 THEN Y=Y-4
101 IF X>255 THEN X=X-252
102 IF X<3 THEN X=X+252
103 IF Y>240 THEN Y=Y-237
104 IF Y<3 THEN Y=Y+237
110 IF S=1 THEN 1000
120 IF S=2 THEN 2000
130 IF S>2 THEN 3000
1000 PAUSE 5:SPRITE C,X,Y:SWAP C,D
1010 SPRITE A:SPRITE B:SPRITE C:SPRITE E:SPRITE F
1020 GOTO 60
2000 PAUSE 5:SPRITE A,X,Y:SWAP A,B
2010 SPRITE A:SPRITE C:SPRITE D:SPRITE E:SPRITE F
2020 GOTO 60
3000 PAUSE 5:SPRITE E,X,Y:SWAP E,F
3010 SPRITE A:SPRITE B:SPRITE C:SPRITE D:SPRITE E
3020 GOTO 60
```

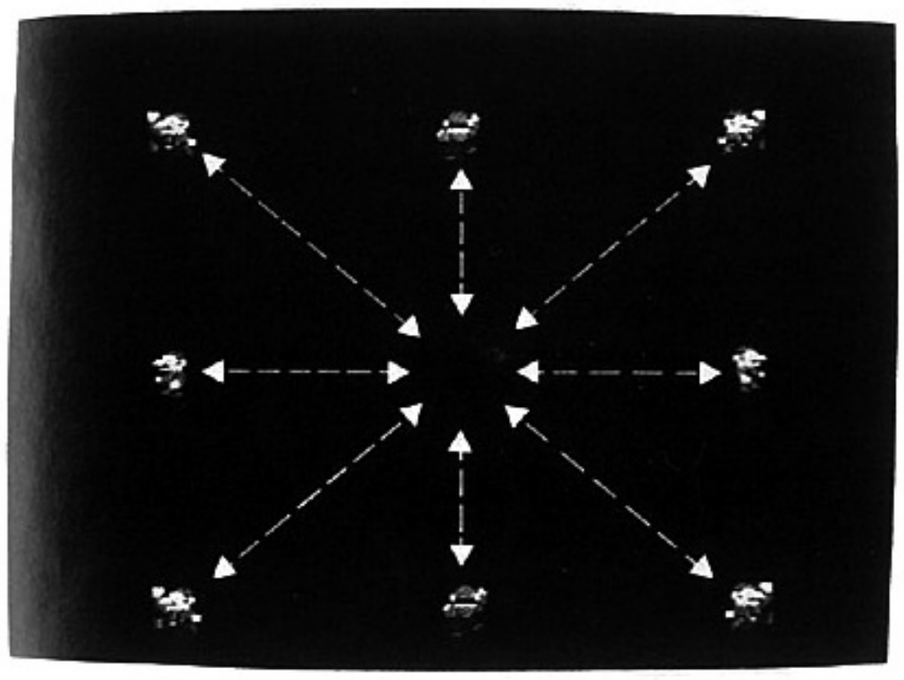


This program is based on program 5 on p. 28 but we have made additions in order to make Mario move up and down, that he does not stick out of the screen and also a way to finish the program.
The up-down movement and display of Mario is done with lines number 100, 130 and from 3000 to 3020.
Mario does not stick out of the screen thanks to the lines from number 101 to 104.
Also, on the lines from number 60 to 65, there is an STRIG command to finish the program using the trigger buttons of the controller.

■ About the MOVE command

The MOVE command is a command specific to Family Basic. Give the MOVE command a try to move dynamically Mario, Lady or any other animated character.

★Let's move Mario in 8 directions simultaneously



NEW RETURN

Enter the above to erase the previous program from the memory. Please enter:

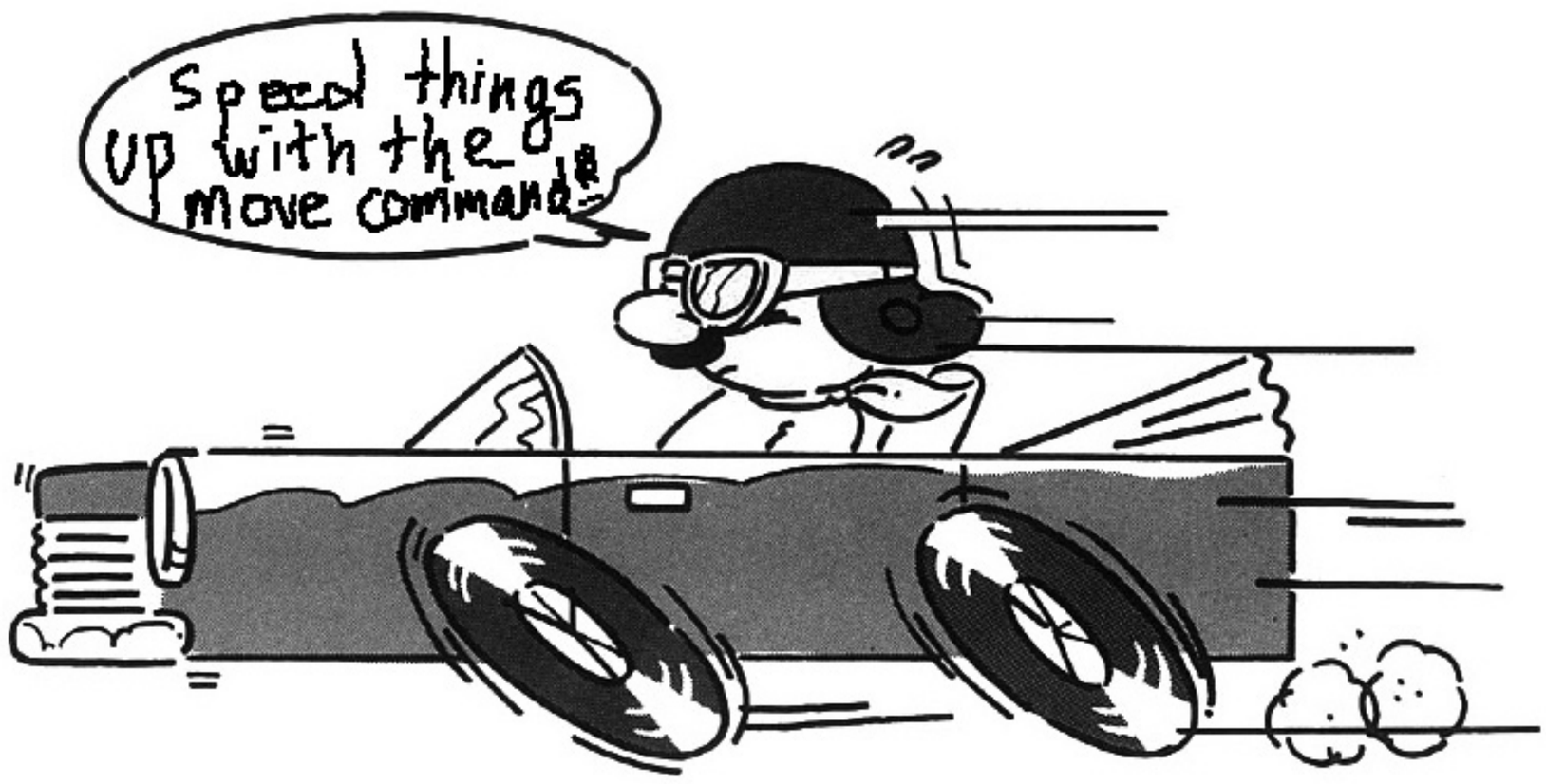
```
5 CLS RETURN
10 SPRITE ON RETURN
20 CGSET 1,0 RETURN
30 FOR N=0 TO 7 RETURN
40 DEF MOVE(N)=SPRITE(0,N+1,3,
   255,0,0) RETURN
50 NEXT RETURN
60 MOVE 0,1,2,3,4,5,6,7 RETURN
70 GOTO 60 RETURN
```

Program 6

When you enter

RUN RETURN

8 Mario's will move into 8 different directions.

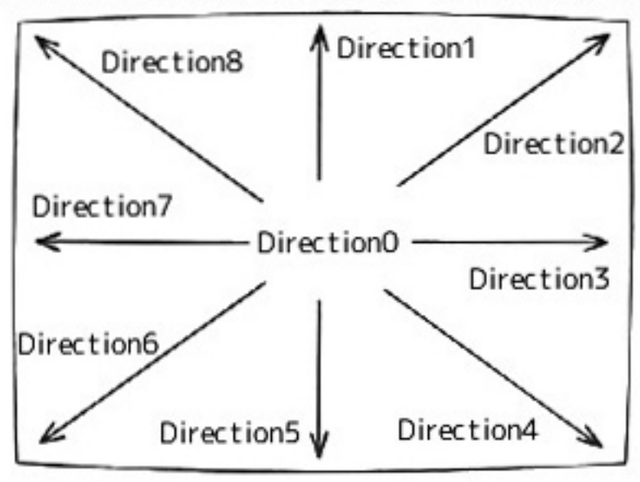


● About DEF MOVE...

DEF MOVE is the command which assigns the type and the movement of animated characters.

```
DEF MOVE(n)=SPRITE(A,B,C,D,E,F)
```

- n: Animated character's action number (0-7)
- A: Animated character's type (0-15) You can assign up to 16 kinds of animated characters.
  - 0 : Mario
  - 1 : Lady
  - 2 : Fighter Fly
  - 3 : Achilles
  - 4 : Penguin
  - 5 : Fireball
  - 6 : Car
  - 7 : Spinner
  - 8 : Star Killer
  - 9 : Starship
  - 10 : Explosion
  - 11 : Smiley
  - 12 : Laser
  - 13 : Shell Creeper
  - 14 : Side Stepper
  - 15 : Nitpicker
- B: Assigns the movement direction of the animated character (0-8) (0 is inactivity)



- C: Movement speed for animated character (1-255)
  - 1: fastest
  - 255: slowest
- D: Total movement volume to move the animated character (1-255) (when 0 is assigned, it is not displayed)
- E: Display priority (0-1)
  - 0 : Displays animated character in front of background.
  - 1 : Displays animated character behind background.
- F: Animated character's color set number (0-3) *Please refer to the color chart on p. 113.

*Please refer to p. 74.

● About MOVE...

MOVE is the command which starts the movement of the animated characters defined by the DEF MOVE command.

MOVE 0, 1, 2, 3, 4, 5, 6, 7

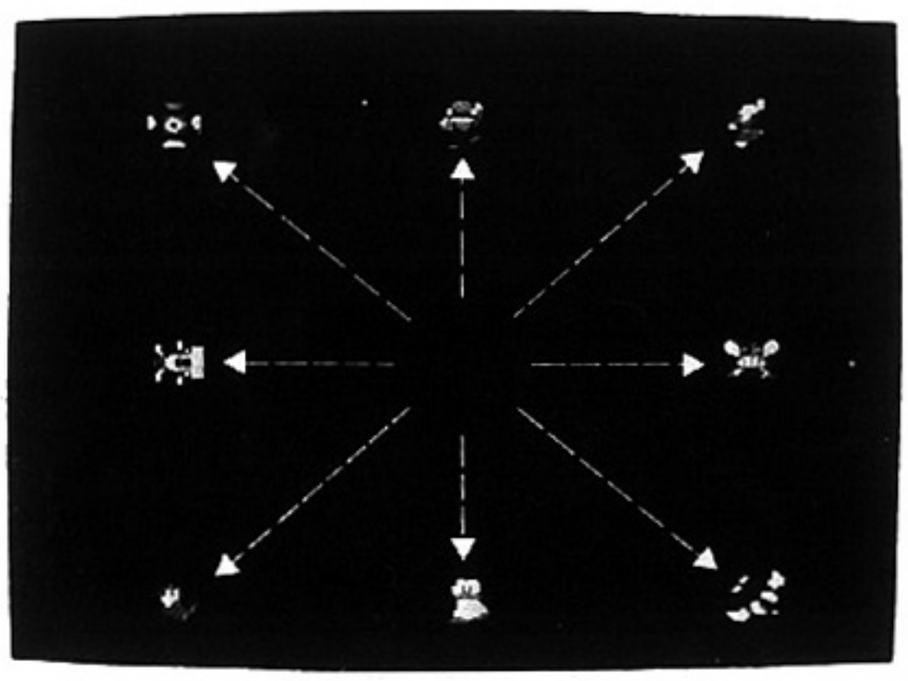
The MOVE command can start the movement of up to 8 animated characters simultaneously. When the MOVE command is executed, the animated characters will move until the movement volume assigned by the value of D (total movement volume) of the DEF MOVE command has reached 0. You can assign the movement of only the animated characters which you desire to move from the numbers you prefer within the animated characters of action numbers from 0 to 7.

MOVE 1.....Only moves the animated character of action number 1.

MOVE 0, 3, 6.....Moves animated characters of action number 0, 3 and 6 simultaneously.

*Please refer to p. 75.

★Let's simultaneously move 8 types of animated characters in 8 directions



Like with Mario, but not just with one character, you can move 8 kinds of characters simultaneously.

Use program 6 on p. 31 and enter:

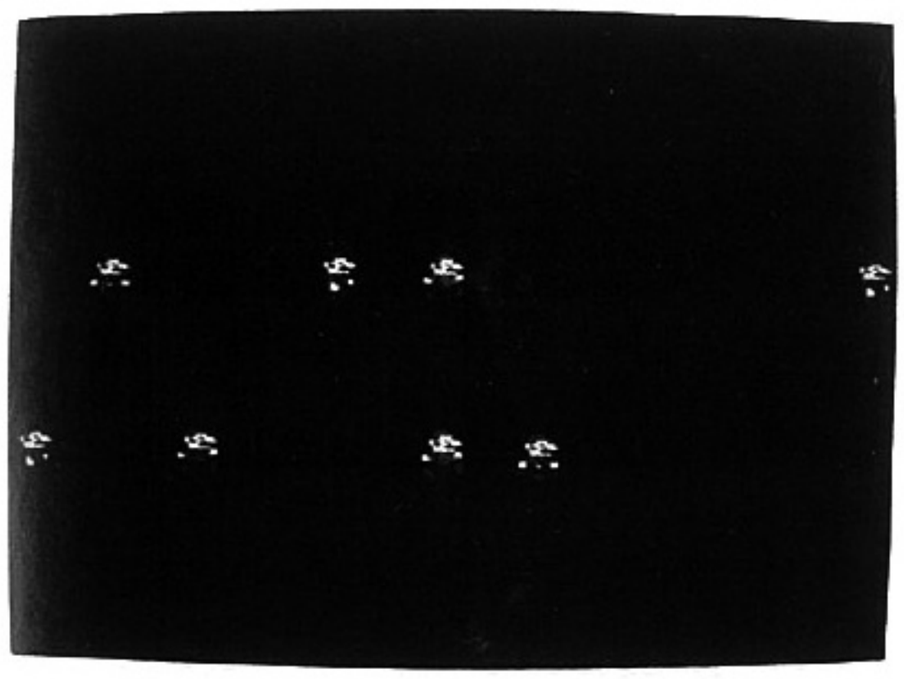
```
40 DEF MOVE(N)=SPRITE(N,N+1,3,
255,0,0) RETURN
```

Upon entering

```
RUN RETURN
```

8 types of characters, such as Mario, Lady, Penguin etc., will move from the center of the screen to the borders of the screen.

★Let's have 8 Mario's march horizontally

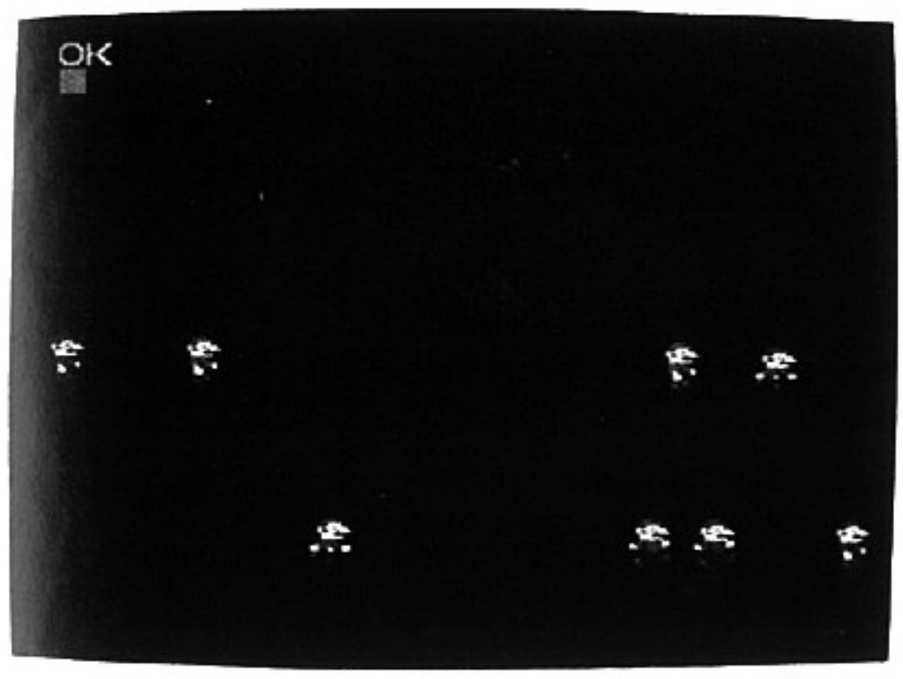


Please use program 6 on p. 31 and enter:

```
40 DEF MOVE(N)=SPRITE(0,3,N+1,255,
   0,0) RETURN
55 FOR N=0 TO 3:POSITION 2*N+1,120,
   180:NEXT RETURN
70 END RETURN
RUN RETURN
```

8 Mario's are walking from left to right.

★Hey Mario, stop!



You can stop all the characters which are moving at the same time, or you can stop one specific character.
Please add the following to the program from above:

```
70 PAUSE RETURN
80 CUT 0,1,2,3,4,5,6,7 RETURN
90 END RETURN
RUN RETURN
```

Please press the SPACE key once when the 8 Mario's are walking. As soon as you press the SPACE key, the command on line number 80 is executed and the 8 Mario's who were walking, stop walking exactly at the same time.
When you enter

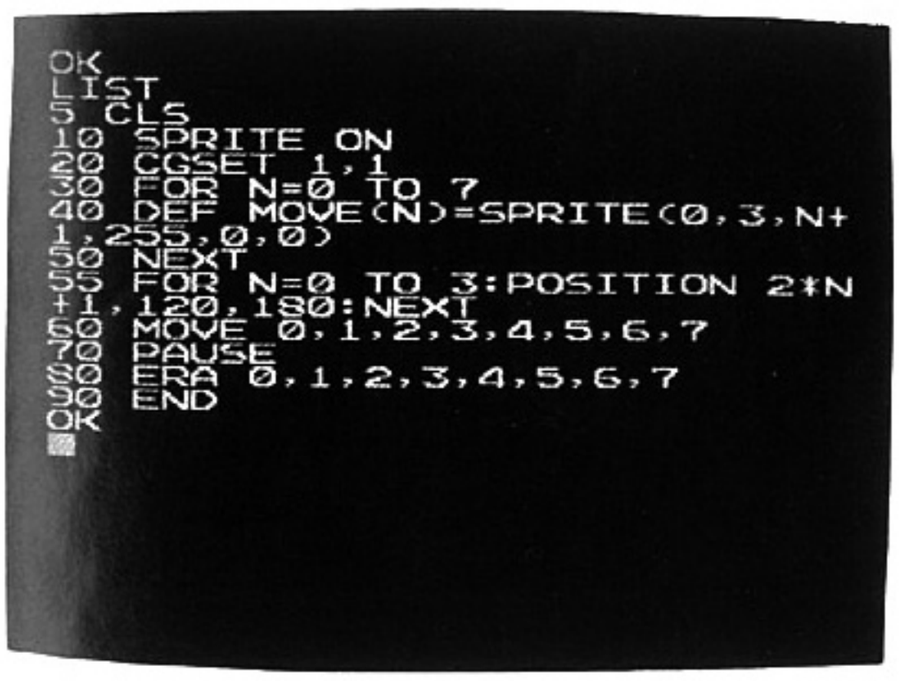
```
MOVE 0,2 RETURN
```

only action number 0 and 2 Mario which had stopped, will start walking.
Please enter also:

```
80 CUT 0,3 RETURN
RUN RETURN
```

Please press the SPACE key once when the 8 Mario's start walking. Only action number 0 and 3 Mario stop walking, all the other Mario's continue walking from left to right as if nothing happened.
You can use the CUT command in direct mode and program mode.

★Ah! Mario has disappeard!



```
OK
LIST
5 CLS
10 SPRITE ON
20 CGSET 1,1
30 FOR N=0 TO 7
40 DEF MOVE(N)=SPRITE(0,3,N+
1,255,0,0)
50 NEXT
55 FOR N=0 TO 3:POSITION 2*N
+1,120,180:NEXT
60 MOVE 0,1,2,3,4,5,6,7
70 PAUSE
80 ERA 0,1,2,3,4,5,6,7
90 END
OK
```

Making animated characters disappear from the screen is a very important technique to make games more interesting. You can erase 8 types of characters simultaneously or only erase specific characters. Please enter:

```
80 ERA 0,1,2,3,4,5,6,7 RETURN
RUN RETURN
```

Please press the SPACE key once when the 8 Mario's start walking. The commanc on line number 80 is executed and the 8 walking Mario's disappear.
When you enter

```
MOVE 0,1 RETURN
```

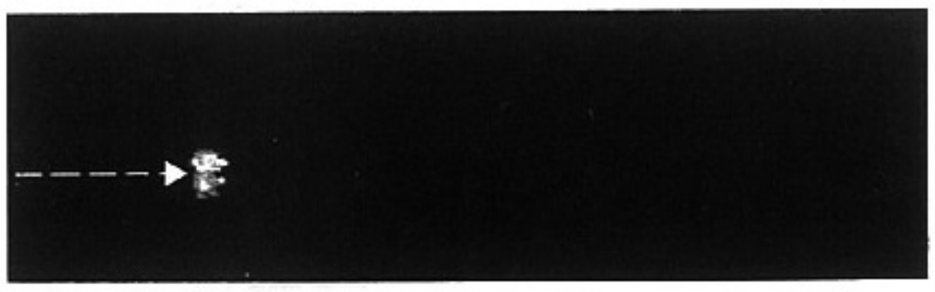
action number 0 and 1 Mario appear and start walking.
Please also enter:

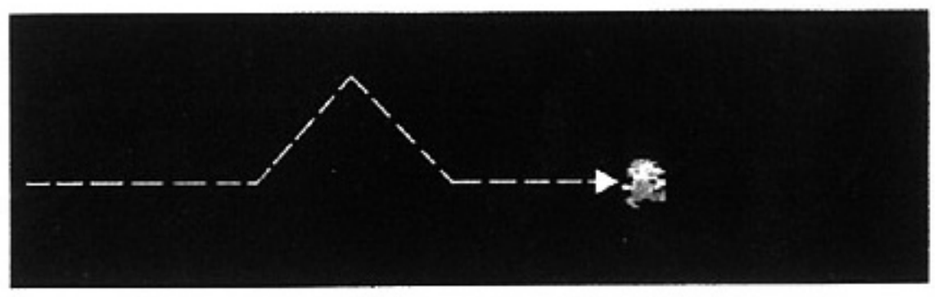
```
80 ERA 3,4 RETURN
RUN RETURN
```

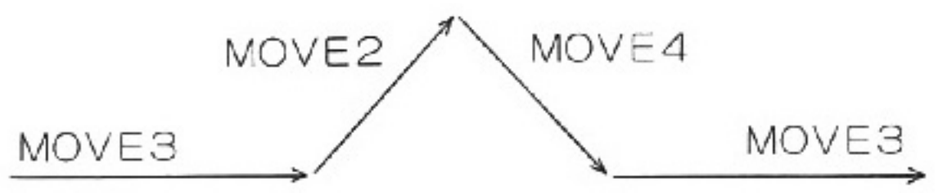
Please press the SPACE key once when the 8 Mario's start walking. Only action number 3 and 4 Mario disappear, the other Mario's continue walking from left to right, as if nothing happened.
The ERA command can be used in Direct Mode and in Program Mode.

When you execute a MOVE command after a CUT or ERA command, the animated character will execute the remaining value of the value of D (total movement volume) defined by the DEF MOVE command starting from the position which was CUT or ERAsed, and stop.

★Let's make Mario jump







Enter the following program to make it look as if Mario is jumping.

```
5 CLS
10 SPRITE ON
20 CGSET 1,0
30 FOR N=2 TO 4
40 DEF MOVE(N)=SPRITE(0,N,1,20,1,0)
50 NEXT
60 MOVE 3
70 IF MOVE(3)=-1 THEN 70
80 ERA 3:POSITION 2,XPOS(3),
   YPOS(3):MOVE 2
90 IF MOVE(2)=-1 THEN 90
100 ERA 2:POSITION 4,XPOS(2),
    YPOS(2):MOVE 4
110 IF MOVE(4)=-1THEN 110
120 ERA 4:POSITION 3,XPOS(4),
    YPOS(4):GOTO 60
```

Enter the following as well if you would like to add a sound effect synchronized with Mario's movement.

```
65 PLAY "T104C1B1DEG1CDE1"
85 PLAY "O3CDE1G1A"
```

When you enter

RUN RETURN

Mario comes walking from the left, jumps in the middle of the screen and upon landing, Mario will start walking again.

●About MOVE (n)...

MOVE (n) is the command which requests the execution or the halt of the movement of an animated character moved by the MOVE line.

MOVE (n) ..... { ・If the animated character is in the middle of an action, -1 can be requested. ・If the animated character has halted the action, 0 can be requested. }

Action number of the animated character (0 to 7)

```
70 IF MOVE(3)=-1 THEN 70
80 ERA 3:POSITION 2,XPOS(3),YPOS(3):MOVE 2
```

In this program, if action number 3 of the animated character is being executed, it will execute the command on line 70. If not, it will execute the command on line 80.

※Please refer to page 77.

●About POSITION...

POSITION is the command which gives the initial coordinates which start an action before a MOVE line moves an animated character.

POSITION n , X , Y

Vertical coordinate (0 to 255)
Horizontal coordinate (0 to 255)
Action number of the animated character (0 to 7)

※Please refer to page 76.

●About XPOS, YPOS...

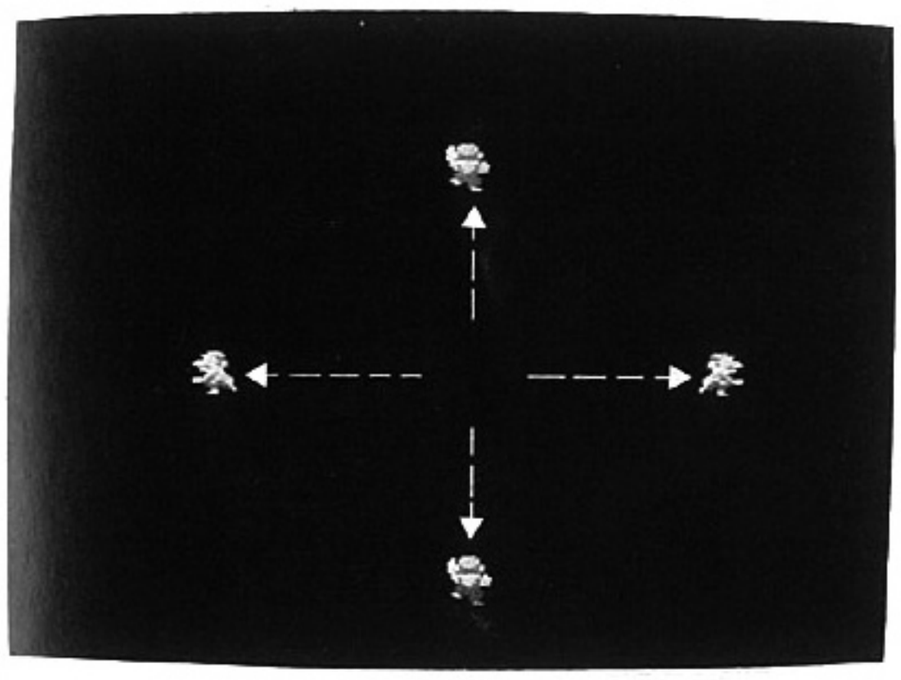
XPOS is the command which requests the horizontal coordinate value of the action number of the animated character. YPOS is the command which requests the vertical coordinate value of the action number of the animated character.

XPOS (n) , YPOS (n)

Action number of the animated character (0 to 7)

※Please refer to page 76.

★Let's use the controller to move Mario



Let's try to move Mario up, down, left, right while using controller [I].
Please enter the following program.

```
5 CLS
10 SPRITE ON
20 CGSET 1,0
30 FOR N=0 TO 7
40 DEF MOVE(N)=SPRITE(0,N,1,3,0,0)
50 NEXT
60 S=STICK(0)
70 IF S=0 THEN N=0:GOTO 120
80 IF S=1 THEN N=3:GOTO 120
90 IF S=2 THEN N=7:GOTO 120
100 IF S=4 THEN N=5:GOTO 120
110 IF S=8 THEN N=1
120 IF MOVE(M)=-1 THEN 120
130 IF M=N THEN 160
140 ERA M:POSITION N,XPOS(M),
    YPOS(M):M=N
150 MOVE N:GOTO 60
160 MOVE M:GOTO 60
```

When you enter

RUN [RETURN]

you can move Mario freely around while using controller [I].
Compare this program to the one on page 30 and you will see that to execute the same actions, using the move command will help you shorten the program.

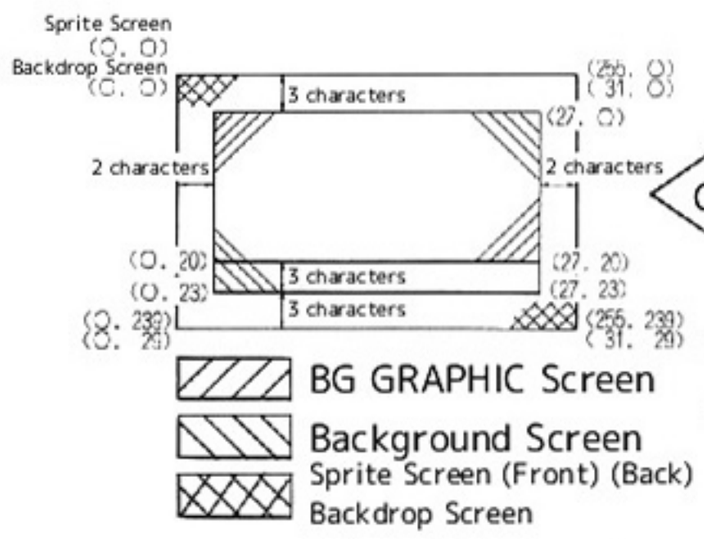
[STOP]:

●The specialties of the MOVE command

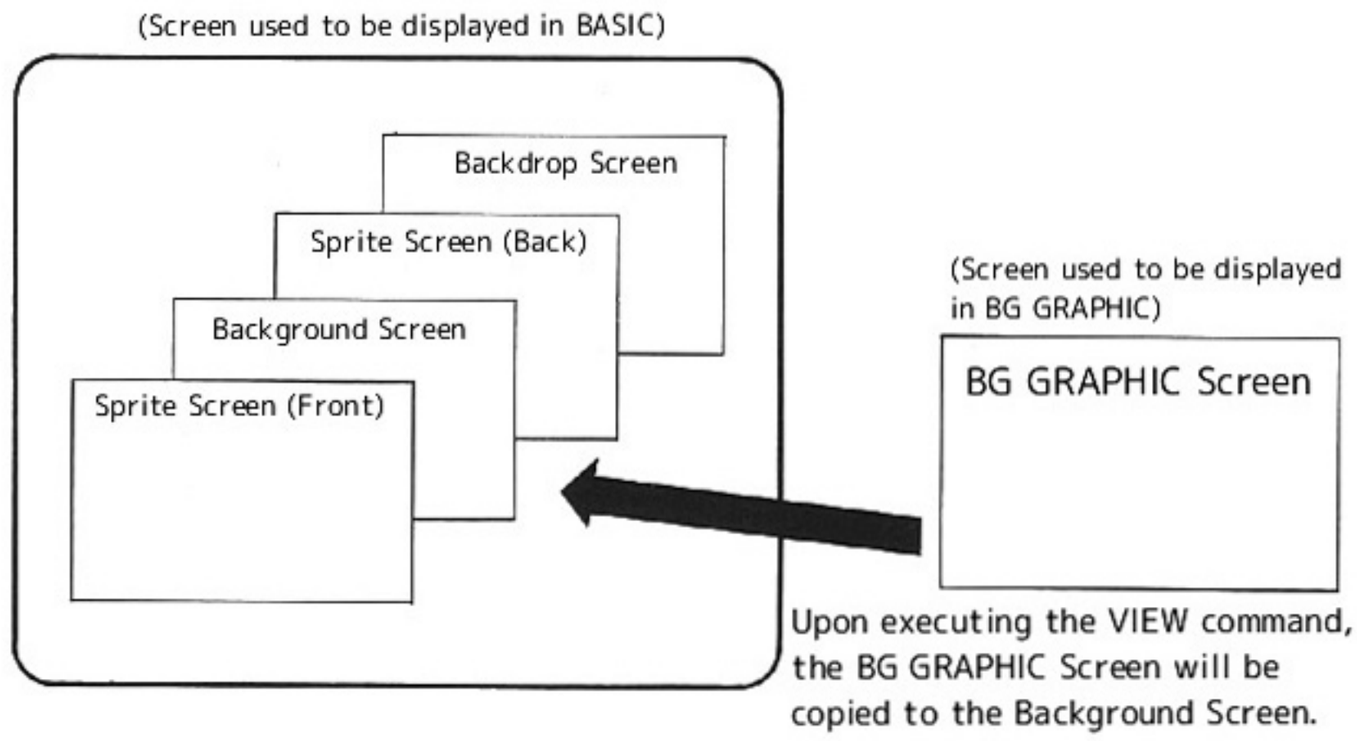
①You can press the key while executing the programe of the MOVE command and display the program list, correct the program, etc.
②While executing the MOVE command, the tempo of the performance becomes slower (1/2).
③You can instruct the movement of up to 8 types of characters simultaneously with the MOVE command. However, for animated characters, you can only instruct up to 4 simultaneously horizontally.
④You can use 16 types of characters with the MOVE command.
⑤You can move 16 types of characters by combining and using the DEF SPRITE and SPRITE commands.

Screen Display Process

The image below represents the front screen of Family Basic.



The image above shows the relative positional relation for each screen.
The numbers above show the coordinate values which can be fixed for each screen.

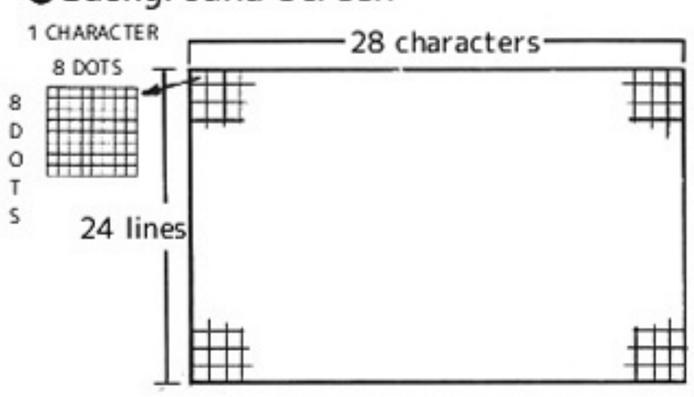


Upon executing the VIEW command, the BG GRAPHIC Screen will be copied to the Background Screen.

The BASIC display screen is composed of 4 screens, which are the Sprite Screen (Front), the Background Screen, the Sprite Screen (Back) and the Backdrop Screen. (You can set Sprite Screens in front of and behind the Background Screen) There is also a BG GRAPHIC Screen which draws BG GRAPHIC.

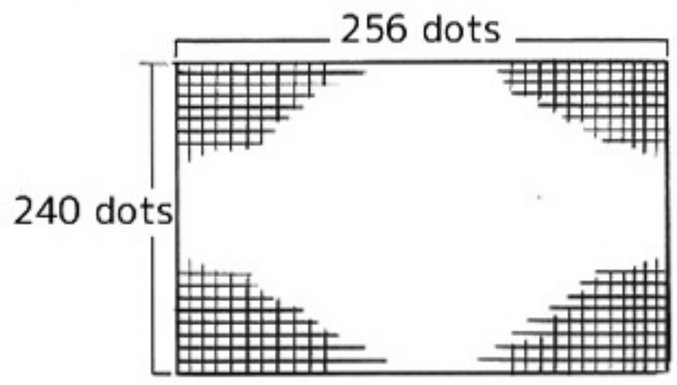
Explanation of each display screen

●Background Screen



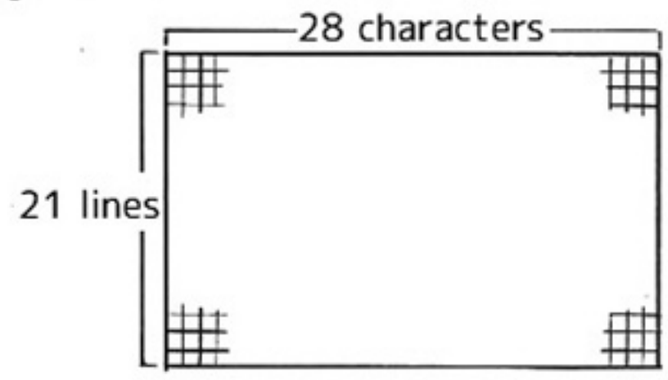
-This is the screen to create BASIC programs, which is always displayed in GAME BASIC mode. (24 vertical lines and 28 horizontal characters)
-Alphanumeric characters, Kana and symbols entered with the keyboard are displayed.
-In GAME BASIC mode, having this screen as the central point, you can display the Sprite Screens in front or behind the Background Screen through BASIC commands.
-The contents of the BG GRAPHIC screen can be copied (duplicated) with the VIEW command.

●Sprite Screen (Front), (Back)



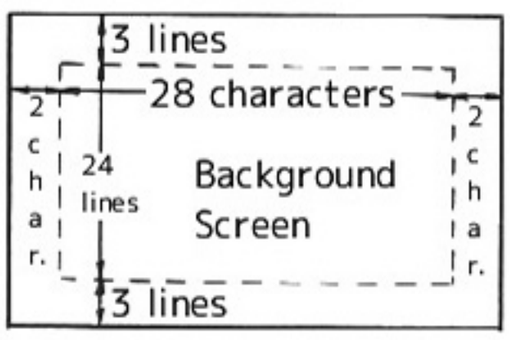
-Screen where Mario and the other animated characters (called sprites on screen) are displayed.
-The position of the sprites on screen can be set within 240 vertical dots and 256 horizontal dots. (The display scope on the TV screen depends on the TV receiver.)
-Use BASIC commands to call and erase this screen. In this case, all the sprites on screen will become visible or disappear.
-You can display perspective by changing the priority values between the alphanumeric characters, kana, symbols and graphical characters of the sprite and background screens (you can make the animated characters appear and disappear on the background screen). This can be done thanks to the display of characters in front and behind the background screen.

●BG GRAPHIC Screen



-The screen which draws BG GRAPHIC. (21 vertical lines and 28 horizontal characters)
-In GAME BASIC mode you can call and copy (duplicate) the content of the screen which is drawn on this screen, onto the background screen through the VIEW command.
-In GAME BASIC mode, you can optimize the game display by displaying the same drawing as in BG GRAPHIC on the background screen and moving the animated character on the sprite screen.
-Through a BASIC command, you can freely modify the screen of the BG GRAPHIC screen which was copied (duplicated) on the background screen, however, the original drawing is saved on the BG GRAPHIC screen.

●Backdrop Screen



-The screen which displays the background color of the background screen which is always displayed.
-Bigger than the background screen, 30 vertical lines and 32 horizontal characters.
-The whole screen can be displayed in 1 color. The default setting is black (transparent).
-This screen is convenient to display skies or oceans.

Computer music and game sound effects

Family Basic includes great music functions such as automatic playback of 3 simultaneous sounds, output of sound effects coordinated with the movements of the animated characters, and so on.
Please try out and enjoy these functions to their full extent.

★Let's create a musical performance!

1 Let's try to compose "Mary had a little lamb" with the computer.

NEW RETURN

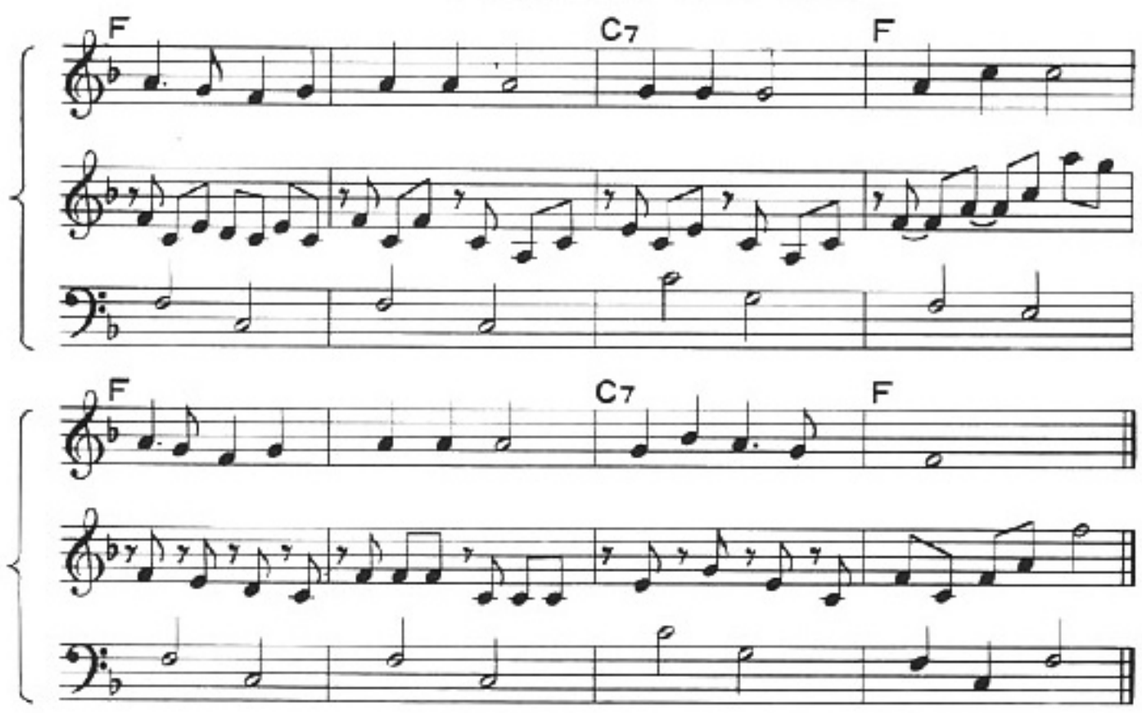
After entering the command from above, the previous program will be erased from the memory.
Please enter the following program.

```
5 CLS
6 LOCATE 8,12:PRINT "メリーサン ノ ヒツジ"
10 PLAY"M1Y2V7T3:M1Y1V5T3:M1T3"
20 PLAY"O2A6G3F5G:O2R3FCEDCEC:O1F7C"
30 PLAY"A5AA7:RFCFRCO1AO2C:FC"
40 PLAY"G5GG7:RECERCO1GO2C:O2CO1G"
50 PLAY"A5O3CC7:RF5AO3C3AG:FC"
60 PLAY"O2A6G3F5G:O2RFRERDRC:FC"
70 PLAY"AAA7:RFFFRCCC:FC"
80 PLAY"G5#AA6G3:RERGRERC:O2CO1G"
90 PLAY"F9:FCFAO3F7:F5CF7"
```

Translation note: the katakana symbol between the quotes read as "Meri-san no hitsuji" which is the Japanese title for "Mary had a little lamb." Feel free to modify the text between those quotes.

Upon entering
RUN RETURN
the computer plays automatically the "Mary had a little lamb" song, right?

"Mary had a little lamb"



2 Let's try to compose Matsuda Seiko's "Rock'n Rouge."

NEW RETURN

After entering the command from above, the previous program will be erased from the memory.
Or you can save it.
Please enter the following program.

```
10 PLAY "M1V10Y2T3:M1V7Y0T3:M1T3
20 PLAY "O4G6C7:O3C3C5C3C5C3C:O2C7C
30 PLAY "R7G1A3GFE1:RCC5CC:O1#A#A
40 PLAY "F4ED3C7:C3C5C3C5C3C:AA
50 PLAY "F4#DD3C5D:O2#G#B#B#G#BG#BF:#GG
60 PLAY "O4G6C7:O2#B3#BG#B#BG#B#B:O2CC
70 PLAY "R7G1A3GFE1:F#B#BF#BF#BF:O1#A#A
80 PLAY "F4ED3C7:#B#BA#B#BA#B#B:AA
90 PLAY "F4#DD3:R3F1G#G#AO3CD:#G6#G3
95 PLAY "#D5F3O3C:C5D3G:#G5#A
97 N=0
100 PLAY "RDECG5C3C:O3:O1#B6#B3#B5G
110 PLAY "RDEC:R7:#A6#A3
115 PLAY "G6C3:F1ED#CC#CD#C:#A5F
```

Continues on the next page.

```
120 PLAY "RDECG5C3G:C7:A6A3A5F
130 PLAY "RGFC:R7:#G6#G3
135 PLAY "CDRC:F1GA#AO4CDEF:#G5#A3#B
140 PLAY "RDECG5C3C:G7O3:O1#B6#B3#B5G
150 PLAY "RDEC:R7:#A6#A3
155 PLAY "G6C3:F1ED#CC#CD#C:#A5F
160 PLAY "RDECG5C3G:C7O2:A6A3A5F
170 PLAY "RGFCC5D:R7#D5F:#B3#B#A#G#G5#A
180 PLAY "D6#D3:O2#A6#B3:#G5#G
185 PLAY "#D5:O3C5C3D:#G#G3#G1#G
190 PLAY "R5D3#D:#D5:#G5#G
195 PLAY "D#DF#D:R5C3C:#G5#G3#G
200 PLAY "D6#D3:O2#A6#B3:G5G
205 PLAY "#D5:#B5G3A:GG3G1G
210 PLAY "R5D3#DD#DF#D:#A8O2#A3#A:O1G5GGG
220 PLAY "F6G3G7:O3D6#D3#D7:F5F3G#G5#G3G
230 PLAY "F5G#DF:D5#DCD:F5F3G#G5F
240 PLAY "G8:O3BO4CDCO3:GGG3EDO0B
245 IF N=1 THEN 260
250 PLAY "R8R3C:B7G1ABO4CDEFG:O1G5GG3AB#B
255 N=1:GOTO 100
260 PLAY "#G5GFD:O1R3DF#GB7:O0B5
270 PLAY "F6#D3#D7:O2G5O3CD#D:O1#B#B#A#A
280 PLAY "F5G#G#A:D#DFG:#G#G#A#A
290 PLAY "#B8:O3C1DEFG#GABO4CDEF:#B#B#B
295 PLAY "R5:G#GAB:#B
300 PLAY "R7#B3B#BB:O4#B5#B:#BO4B
305 M=0
310 PLAY "O3F5:R5O3FA#B:O1D6D5E3F#F
320 PLAY "R:B0#BB#BB#BB#BB5:G6G3
325 PLAY "#B3B#BB:R7:R3GAB
330 PLAY "O3E5:R5GBO4D:#B6#B3RO1EGB
340 PLAY "R:#C0D#CD#CD#CD#C5:A6A3
345 PLAY "A3GAG:O3R:R3GE#CO1
347 IF M=1 THEN 390
350 PLAY "C5C3FD5D3G:R3FARRDF:D6D3#A6#A3
360 PLAY "E5AB#B:RCERRO2A#B:A6A3#F6#F3
370 PLAY "#B7R3BAB:O3RCDFGGFG:G6G3G5
380 PLAY "R:O2G1O3CDFDFGB:G6G3
385 PLAY "#B3B#BB:GBO4DFDFGB:G
387 M=1:GOTO 310
390 PLAY "F5F3GF5F3G:D5D3ED5D3E:D7O1#A5A
400 PLAY "F5A#BB:D5FGF:G6G3G6G3
410 PLAY "#B8:O4C3C5C3C5C3C:#B5#B#B#B
420 PLAY ":RCC5CC:#A#A#A#A
430 PLAY ":C3C5C3C5C3C:AAAA
440 PLAY "R3#DFG#G5:RCC5C:#G#G#G
445 PLAY "#A3#B:C3C:#A3#B
450 PLAY "O1R7C:R7C
```

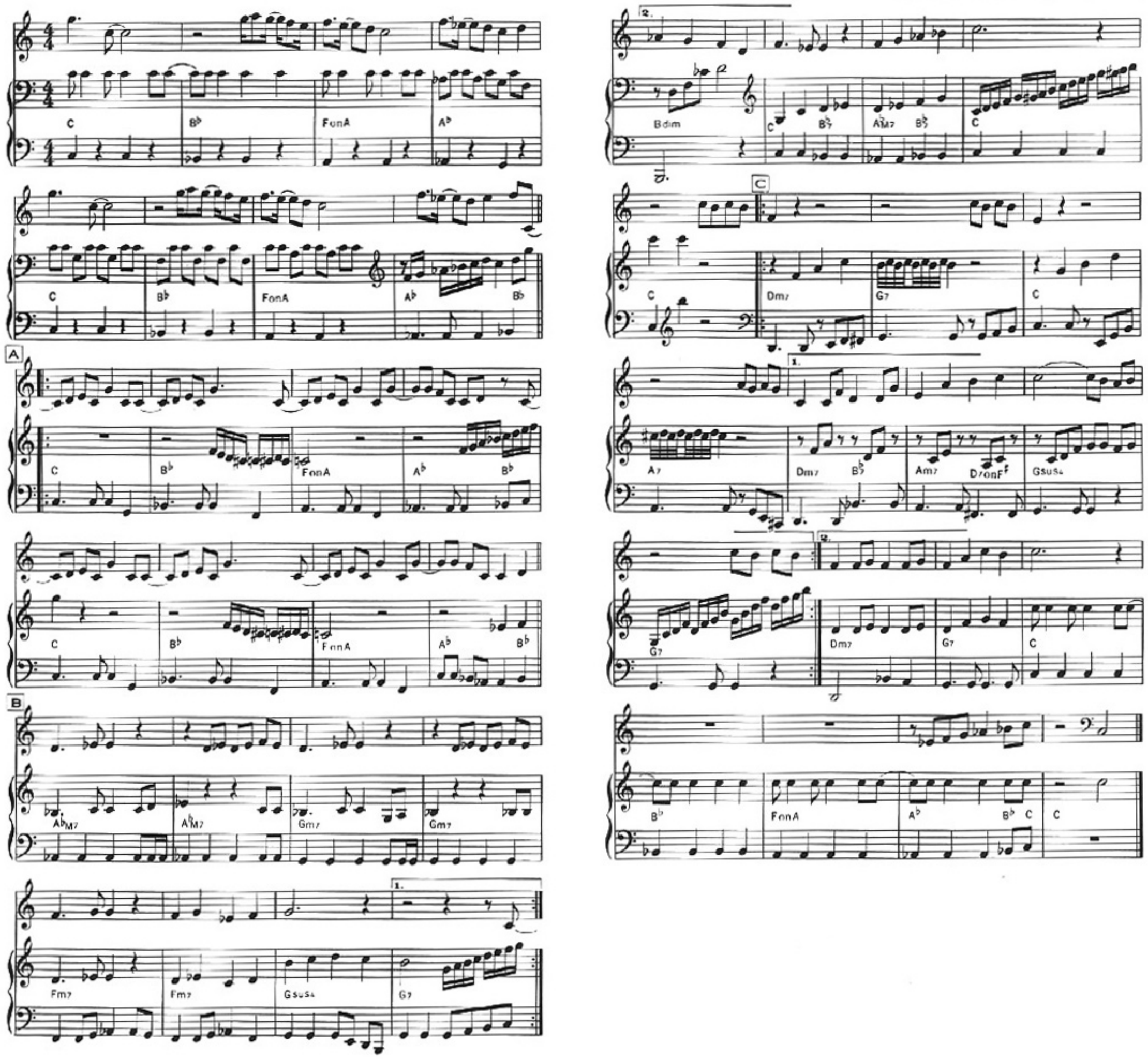
Upon entering

RUN [RETURN]

the computer will play "Rock 'n Rouge."

Rock'n Rouge

Karuho Kureta (composer)



★Let's synchronize sounds with Mario's movements!

Adding sound is very effective to increase the fun of a game!
Please enter the following program.

```
5 CLS:SPRITE ON:CGSET 1,0
10 FOR N=0 TO 2:DEF MOVE(N)=SPRITE(0,N+2,1,20,1,0):NEXT
20 FOR K=3 TO 7:DEF MOVE(K)=SPRITE(8+K,7,1,255,1,0):NEXT
30 POSITION 7,0,30:MOVE 1
40 PLAY"T104C1D1G1B1CGAB"
50 IF MOVE(1)=-1 THEN 50
60 ERA 1:POSITION 0,XPOS(1),YPOS(1):MOVE 0,6
70 PLAY"T105CDEABGD"
80 IF MOVE(0)=-1 THEN 80
90 ERA 0:POSITION 2,XPOS(0),YPOS(0):MOVE 2,5
100 IF MOVE(2)=-1 THEN 100
110 ERA 2:POSITION 1,XPOS(2),YPOS(2):MOVE 7,3:GOTO 30
```

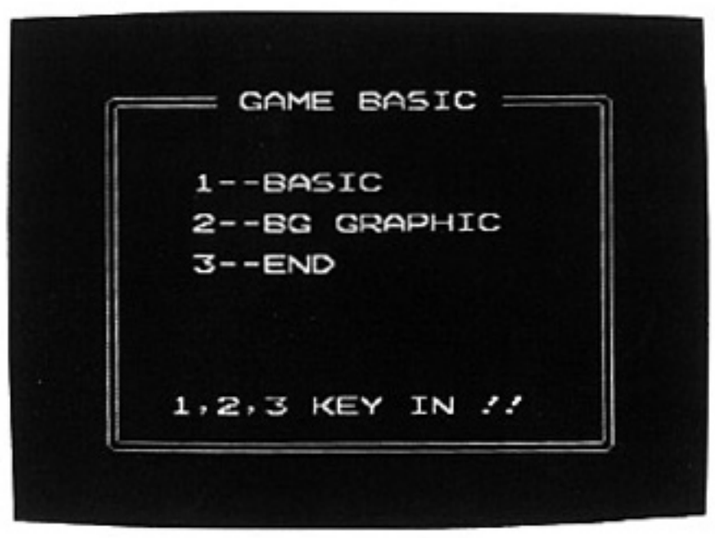
Upon entering
RUN RETURN
sound will be played at the same time as Mario's actions on screen, making it even more fun!

※Please refer to p. 80 for more info about the PLAY command. Also, refer to p. 70 for more info about the LOCATE command.

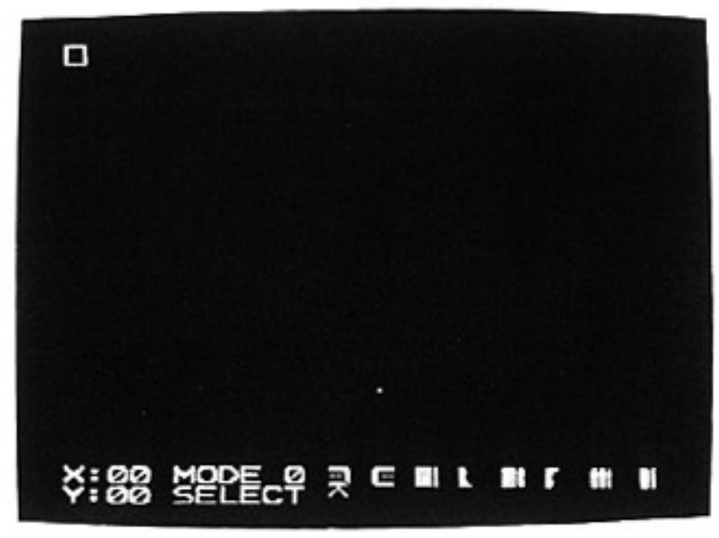
BG GRAPHIC

Here we shall explain how to draw the backgrounds over which the animated characters will interact.
All the characters which you can use in these background drawings are introduced in character table B, p. 113.

※Please enter [S][Y][S][T][E][M][RETURN] to call the GAME BASIC mode screen from the BASIC screen.



Press key [2] to execute BG GRAPHIC. (Do not press key [3] as this will end the GAME BASIC mode.)



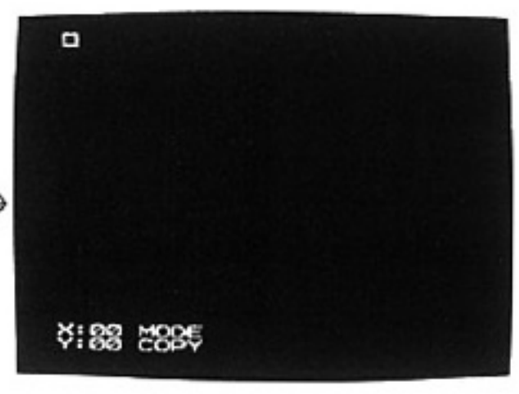
Use BG GRAPHIC to draw a background or to create patterns.

FUNCTION MENU

Press the [ESC] key to display the function menu in this position.

SELECT	Selects the character(s) displayed on screen. Use the [D] key to delete the character(s) within the cursor.
COPY	Copies a character which is displayed on screen to duplicate it in a different position.
MOVE	Moves a character which is displayed on screen to a different position.
CLEAR	Deletes all the characters which are displayed on screen.
FILE	Used to SAVE the drawing displayed on screen onto a cassette tape, or to LOAD a drawing from a cassette tape. Please refer to p. 46.
CHAR	Displays alphanumerics, symbols and kana.

Use the [▼] key to select a function.
Upon pressing the [SPACE] key, the menu of the selected function will be executed.



Coordinate value display

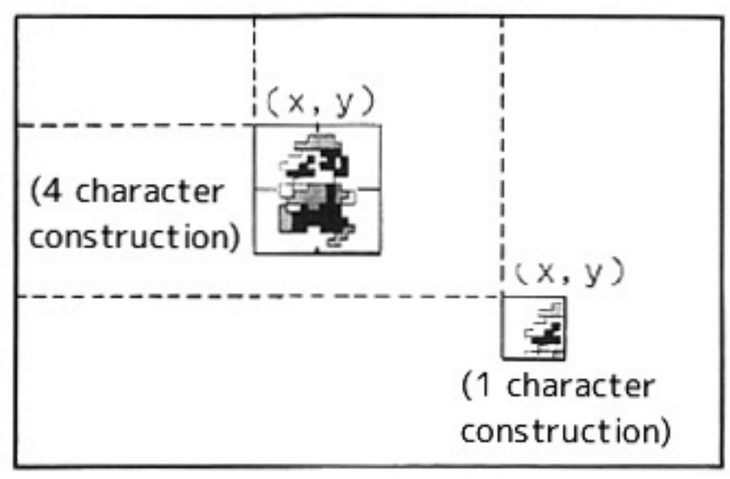
X (horizontal) and Y (vertical) show the position of the cursor which displays the character on screen. (Only during SELECT, COPY and MOVE. During CHAR, the value of the coordinates does not change.)
The characters can be drawn within the following range:
X : 00-27 (28 squares), Y: 00-20 (21 lines)

The relation between the sprite screen and the BG GRAPHIC screen coordinates

x ... x coordinate for sprites
y ... y coordinate for sprites
X ... X coordinate for BG GRAPHIC
Y ... Y coordinate for BG GRAPHIC
When fixing the coordinates of the sprite screen and the BG GRAPHIC screen as above, the coordinates to assemble animated characters use the following type of formula.

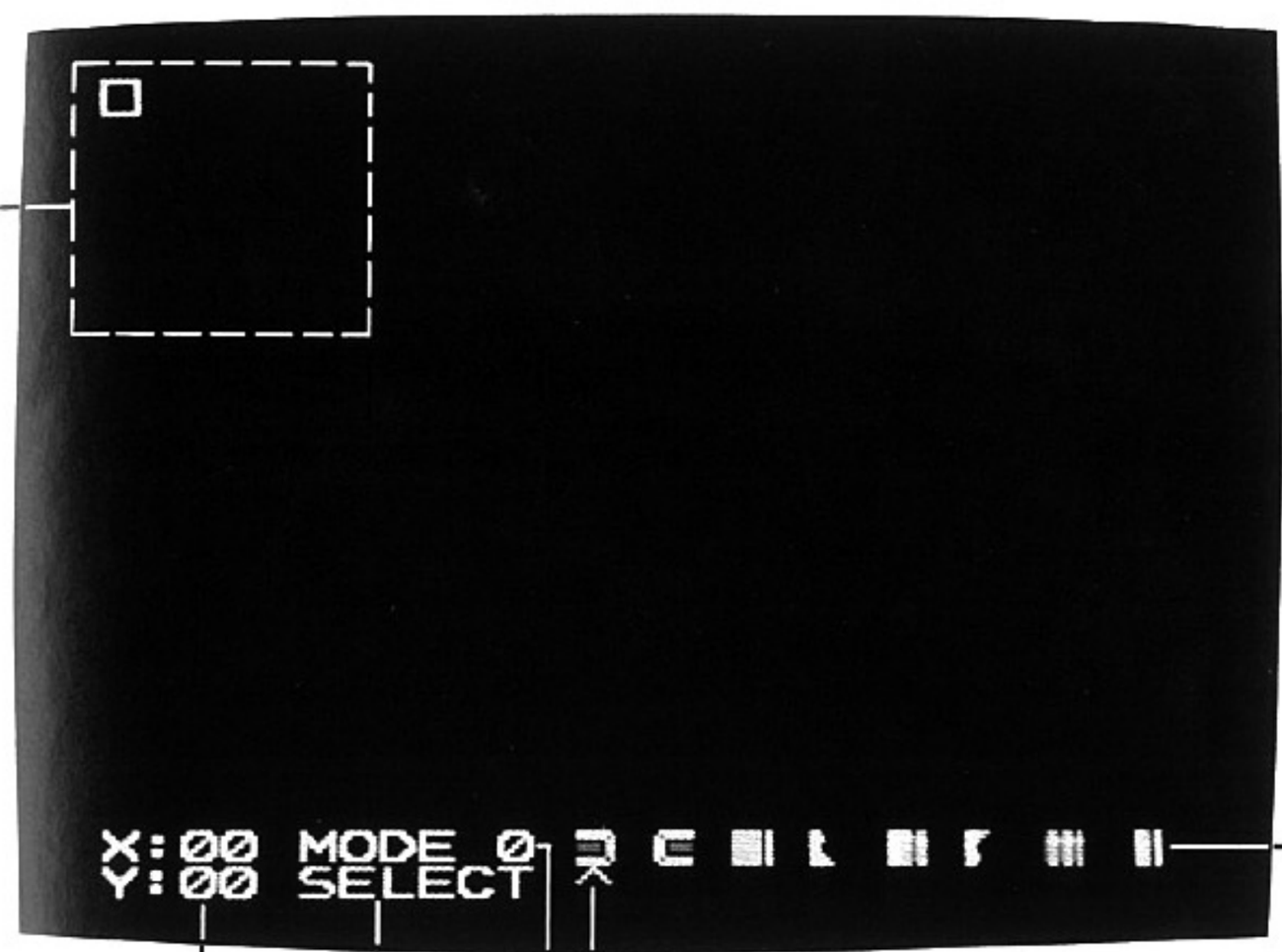
$$x = (X \times 8) + 16 \qquad y = (Y \times 8) + 24$$

Therefore, when you would like to display animated characters over the background pattern, please use the formula from above to create programs.



※Please refer to p. 36 "Screen Display Process".

Use the [▲] [▼] [◀] [▶] keys to move the cursor (□) in the coordinate values which display the character.



Character group

Each group consists of 8 characters. Press the [CLR HOME] key to switch the groups in regular order. There are 13 groups. Press the [CLR HOME] key while holding down the [SHIFT] key to switch the groups in backward order. You are provided with 8 x 13 = 104 characters. (Please refer to character table B on p. 113)

Execution mode display

Displays the mode selected from the function menu.

Coloring number

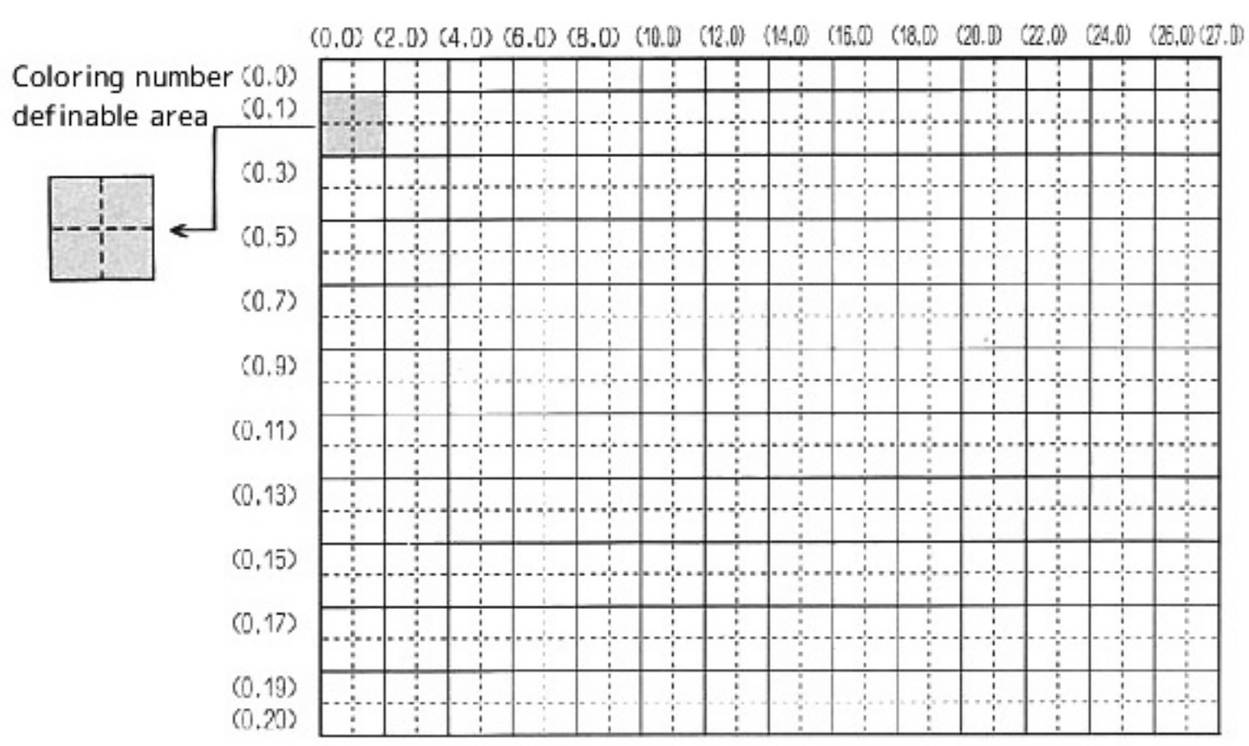
Every time you press the [RETURN] key, the color of the characters changes. You can choose a coloring among 4 types (0-3).

You can use a coloring number within a area. For this reason, if you redraw a picture within an area with a different coloring number, all the colors within this area will change.

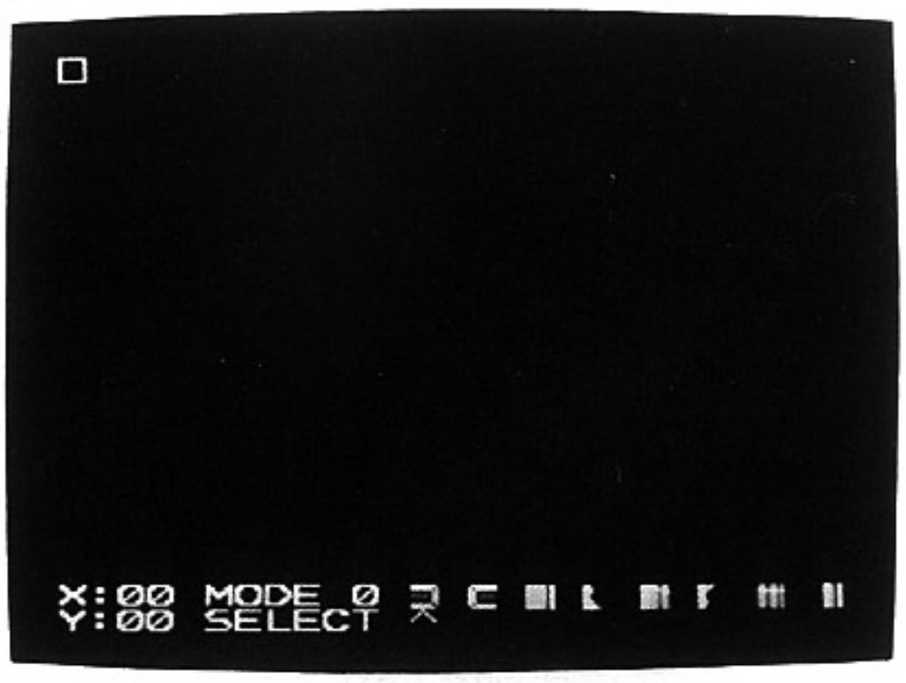
※Please refer to the color chart on p. 113.

Character marker

Marker to inform which character is selected within a character group. When pressing the [DEL] key, the character marker moves to the right. When pressing the [INS] key, the character marker moves to the left.



★Use SELECT to draw backgrounds and pictures



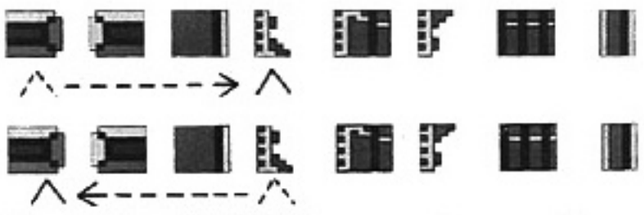
Use SELECT to select the types or the colorings of the characters to display on screen. First, press the [ESC] key to display the function menu.

①Press the [SPACE] key to enter SELECT mode.
Press the [CLR HOME] key or press the [CLR HOME] key while holding down the [SHIFT] key in order to select a character group which appears to the lower right of the screen.

…Press the [CLR HOME] key to display the characters in regular order.

…Press the [CLR HOME] key while holding down the [SHIFT] key to display the characters in backward order.

②Press the [DEL] and the [INS] key to select the character to display.

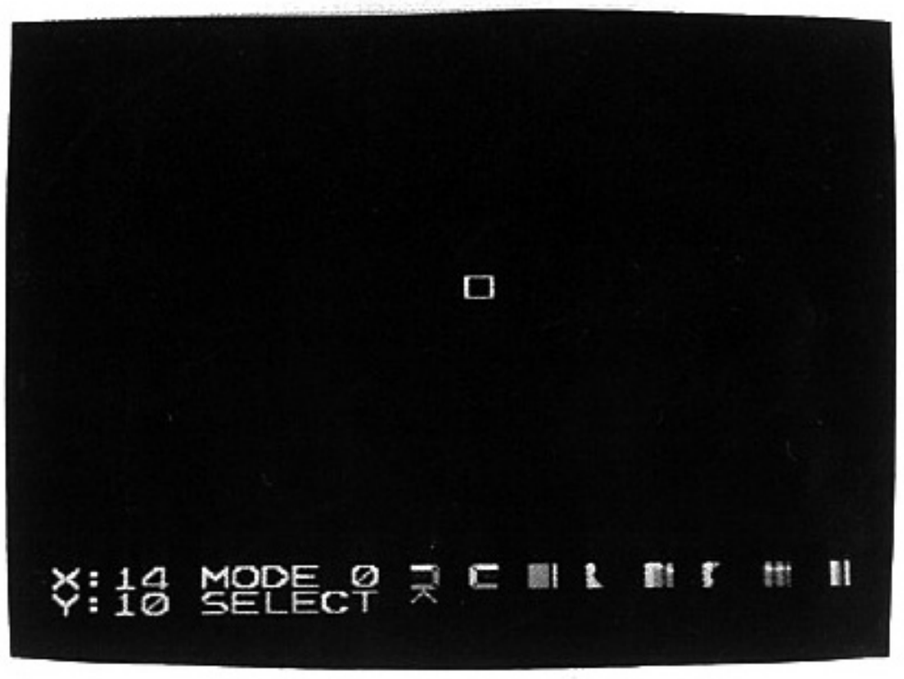


…Press the [DEL] key to move the character marker to the right

…Press the [INS] key to move the character marker to the left

③Press the [RETURN] key to change the coloring of the character to be displayed. There are four types of coloring, from 0 to 3.

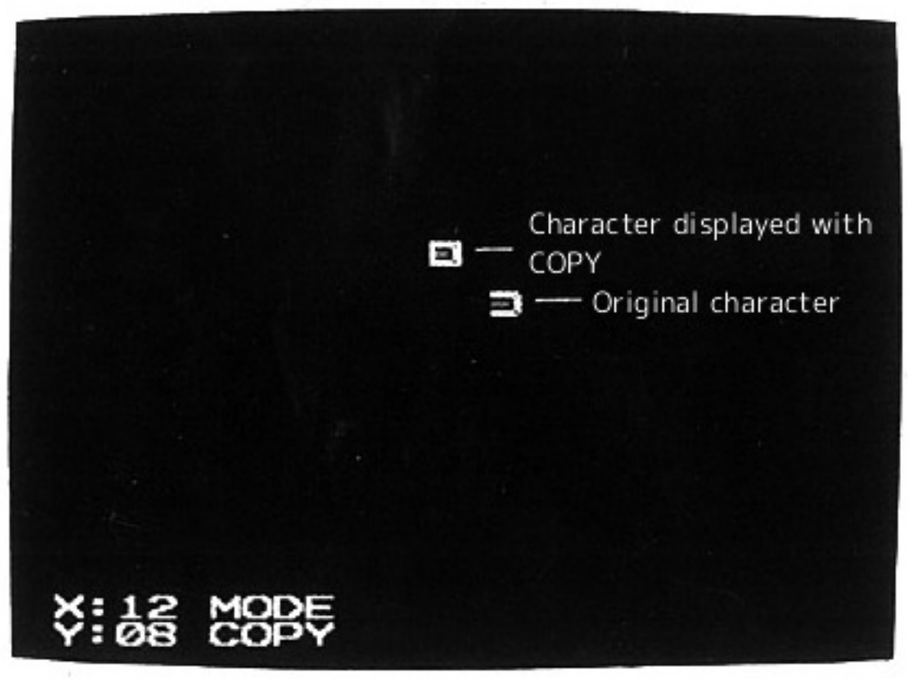
④Use the [▲] [▼] [◀] [▶] keys to move the cursor to the position to display. (For example, move the cursor to X:14, Y:10)



⑤Press the [SPACE] key to display the selected character at the position of the cursor. If you would like to continue displaying more characters, keep pressing the [SPACE] key. The [SPACE] key is used to display the selected character on screen.

⑥Use the [D] key to delete a character from the position of the cursor.

★Use COPY to display characters with the same pattern



COPY is used to copy (duplicate) a displayed character in a different position.
First of all, press the [ESC] key to display the function menu.

①Press the [▼] key and select COPY.

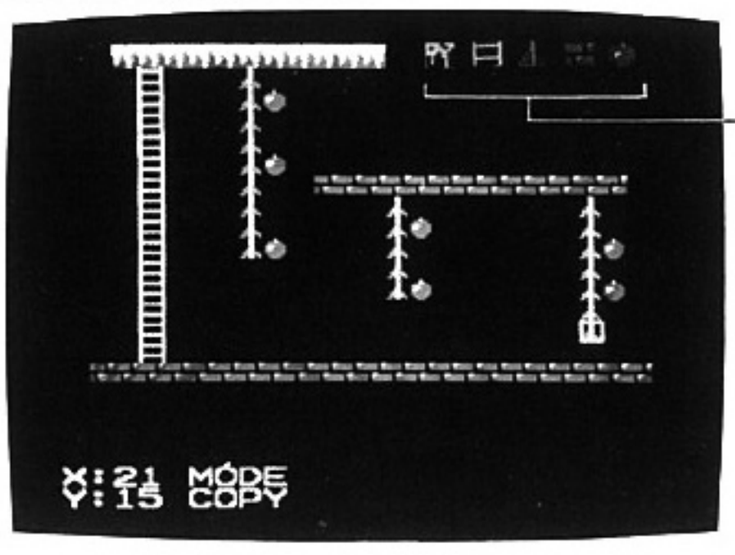
②Press the [SPACE] key to switch to COPY mode. Press the [▲] [▼] [◀] [▶] keys and move to the position of the character which you want to copy (duplicate). (For example, the character in X:14, Y:10)

③Press the [INS] key, now you can move the character to any place you want.

④Press the [▲] [▼] [◀] [▶] keys to move the cursor to the position where you would like to display it (for example, X:12, Y:08). The original character is still displayed.

⑤After selecting the position, press the [DEL] key to delete the character at the cursor. In the same way, use the [▲] [▼] [◀] [▶] keys with the [DEL] key to copy several characters (duplicate).

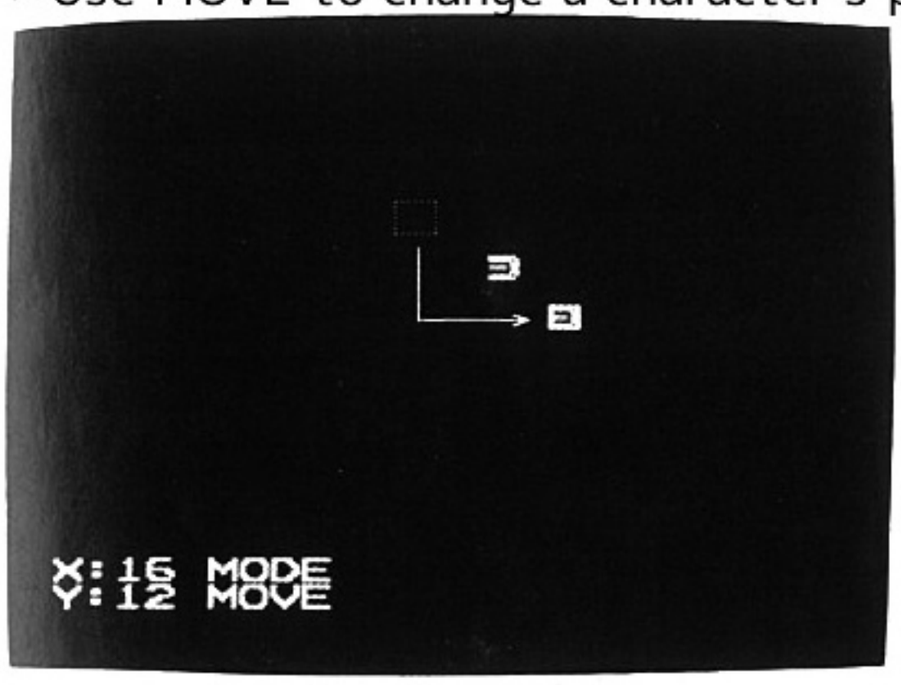
●Convenient use of COPY



Characters set by using SELECT

For example, when drawing a picture with 5 characters
①Use SELECT to display 5 characters 1 by 1 on screen.
②Use COPY to copy (duplicate) each character and draw them on screen.

★Use MOVE to change a character's position



Use MOVE to move a displayed character to a different position. This is very convenient when you made a mistake by placing a character in a wrong position and you would like to correct this.

First of all, press the ESC key to display the function menu.

① Press the ▼ key and select MOVE.

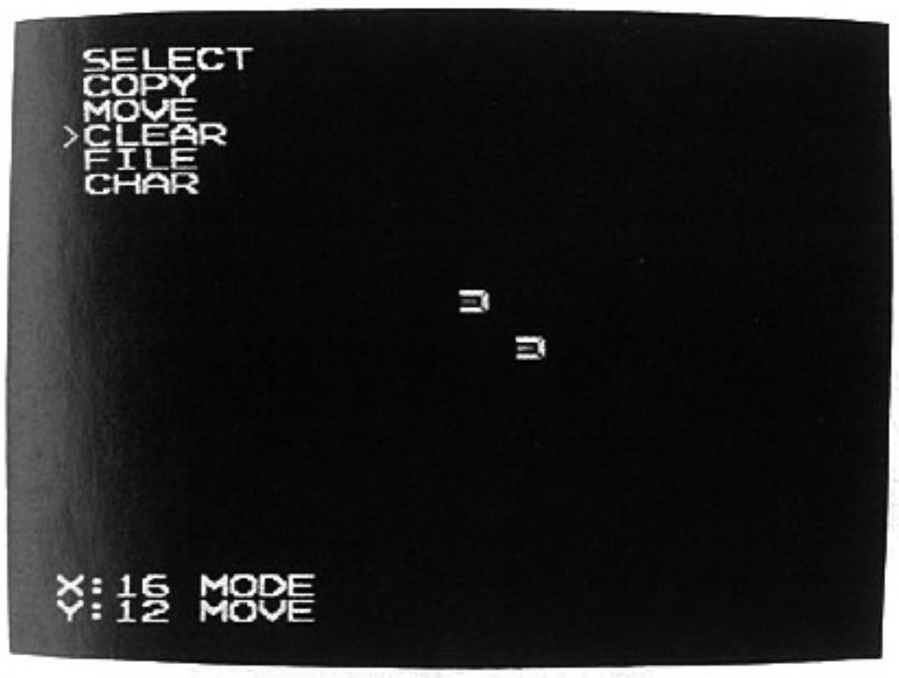
② Press the SPACE key to enter the MOVE mode. Press the ▲ ▼ ◀ ▶ keys to move to the position where you would like to display the character (for example, X:12, Y:8).

③ Upon pressing the INS key once, you will be able to cut the character and move it to wherever you like. (Pressing INS twice will delete it)

④ Press the ▲ ▼ ◀ ▶ keys to move the character to the desired position (for example, X:16, Y:12).

⑤ When you have selected the position, press the DEL key to display the character at the position of the cursor. This finishes the transfer of the character.

★Use CLEAR to delete all the characters displayed on screen



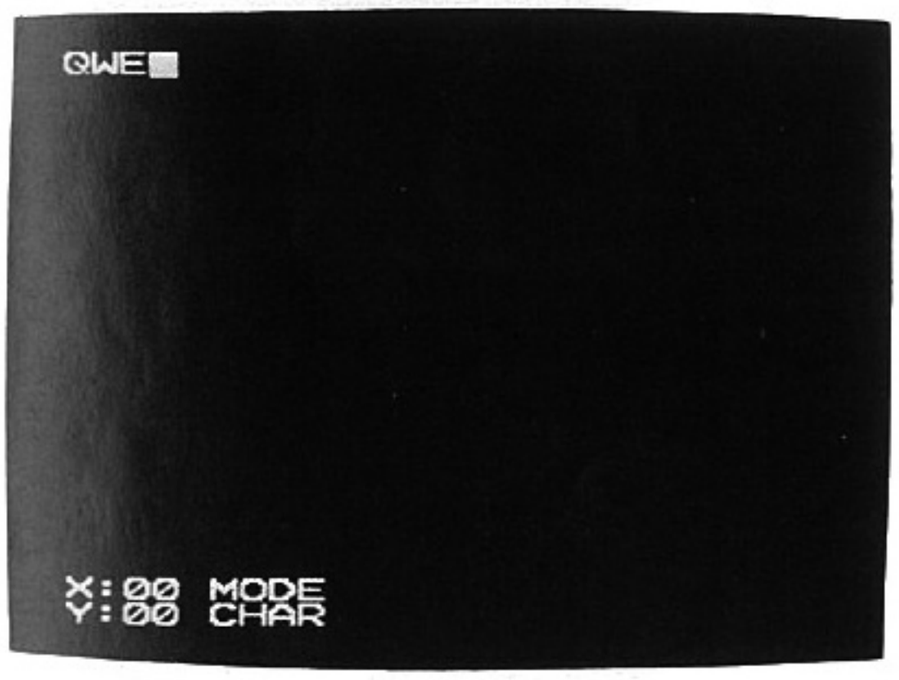
Use CLEAR to delete all the characters displayed on screen.
Use this function when you would like to modify all the characters drawn on screen or to redraw the characters from the beginning.
First of all, press the ESC key to display the function menu.

① Press the ▼ key and select CLEAR.

② Press the SPACE key to execute the CLEAR mode and all the characters displayed on screen will be deleted.

③ Turns automatically into SELECT after deleting everything.

★Use CHAR to display numbers, letters, symbols and kana



You can use CHAR to display the numbers, letters, symbols and kana which appear on the keyboard.
You can use not only character groups, but also numbers, letters, symbols and kana.
First of all, press the ESC key to display the function menu.

① Press the ▼ key and select CHAR.

② Press the SPACE key to execute the CHAR mode. Enter directly numbers, letters, symbols and kana from the keyboard to the position of the cursor.

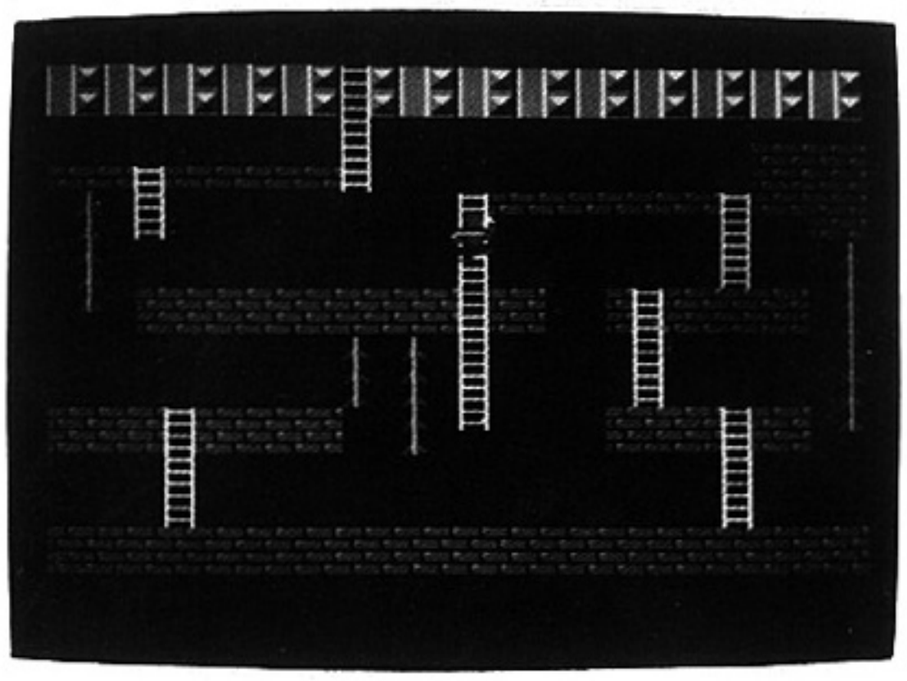
※ The coordinate values of X and Y will not change, even if you move the cursor.

★Use FILE to SAVE or LOAD onto a cassette tape. ※Please refer to p. 46.

The composition of BASIC and BG GRAPHIC

By now, you have probably understood how to create a program to move Mario's animated character or how to build a background. Here we will explain how to compound a program made in "BASIC" and a background from "BG GRAPHIC".

★Display Mario and a background on screen



▽Draw a background picture in BG GRAPHIC.

▽Press the [STOP] key after pressing the [ESC] key and return to the GAME BASIC mode screen.

▽Press the [1] key to turn it into the BASIC screen.

▽Enter the BASIC program from below.

▽Change CLS from 5 CLS and enter VIEW [RETURN].

※When using other programs, be careful when entering VIEW [RETURN].

If CLS is absent·····Add a line number smaller than the first number of the program and add VIEW.

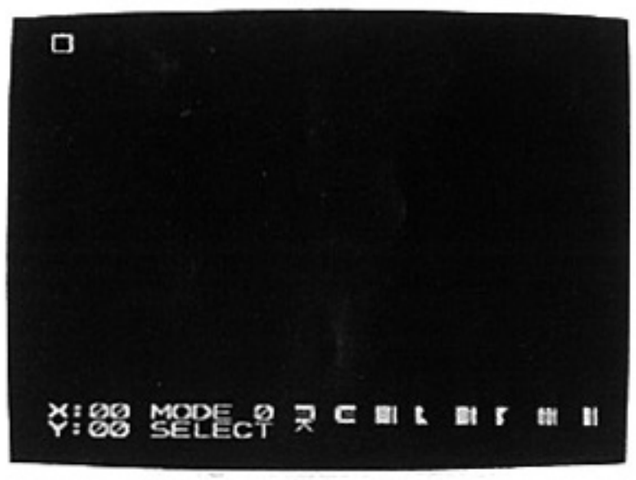
If CLS is present·····Change CLS into VIEW.

▽RUN [RETURN]

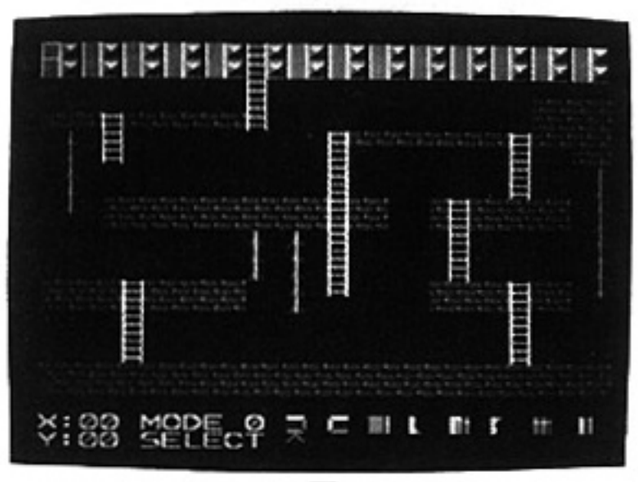
And like this, the animated character and the background are displayed together, right?

● How to compound Mario and the background

Press key [2]



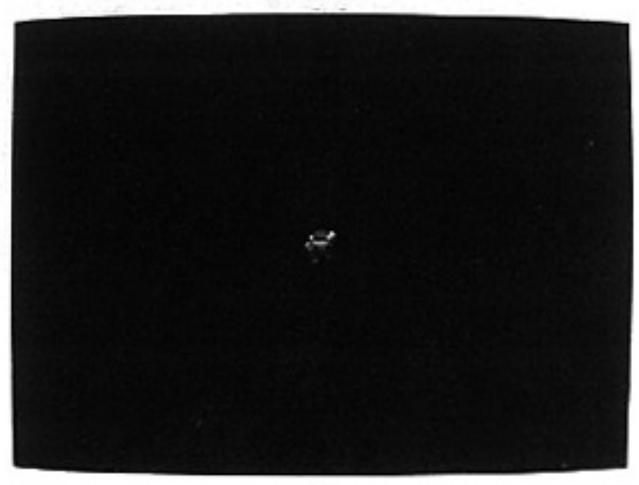
Draw the picture which becomes the background.



The background picture is drawn onto the BG GRAPHIC screen

Press key [1]

Create the program to display MARIO.

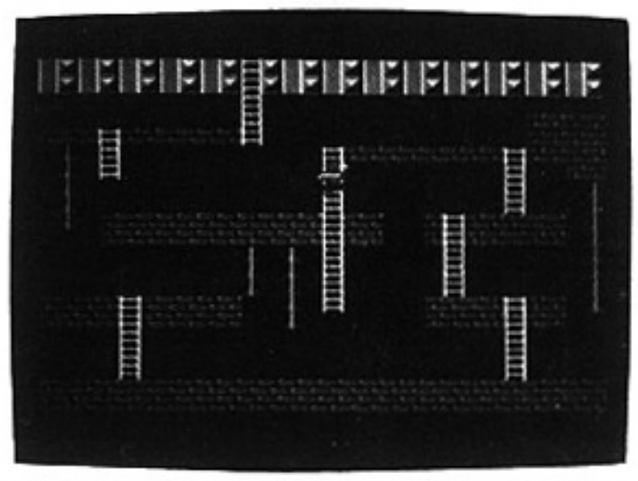


Mario is drawn onto the sprite screen

The VIEW command copies (duplicates) the background screen onto the BG GRAPHIC screen

RUN [RETURN]

Set the pallette code for the background to 1 in order to copy the coloring in the picture of the BG GRAPHIC screen onto the background screen. (CGSET1,1)



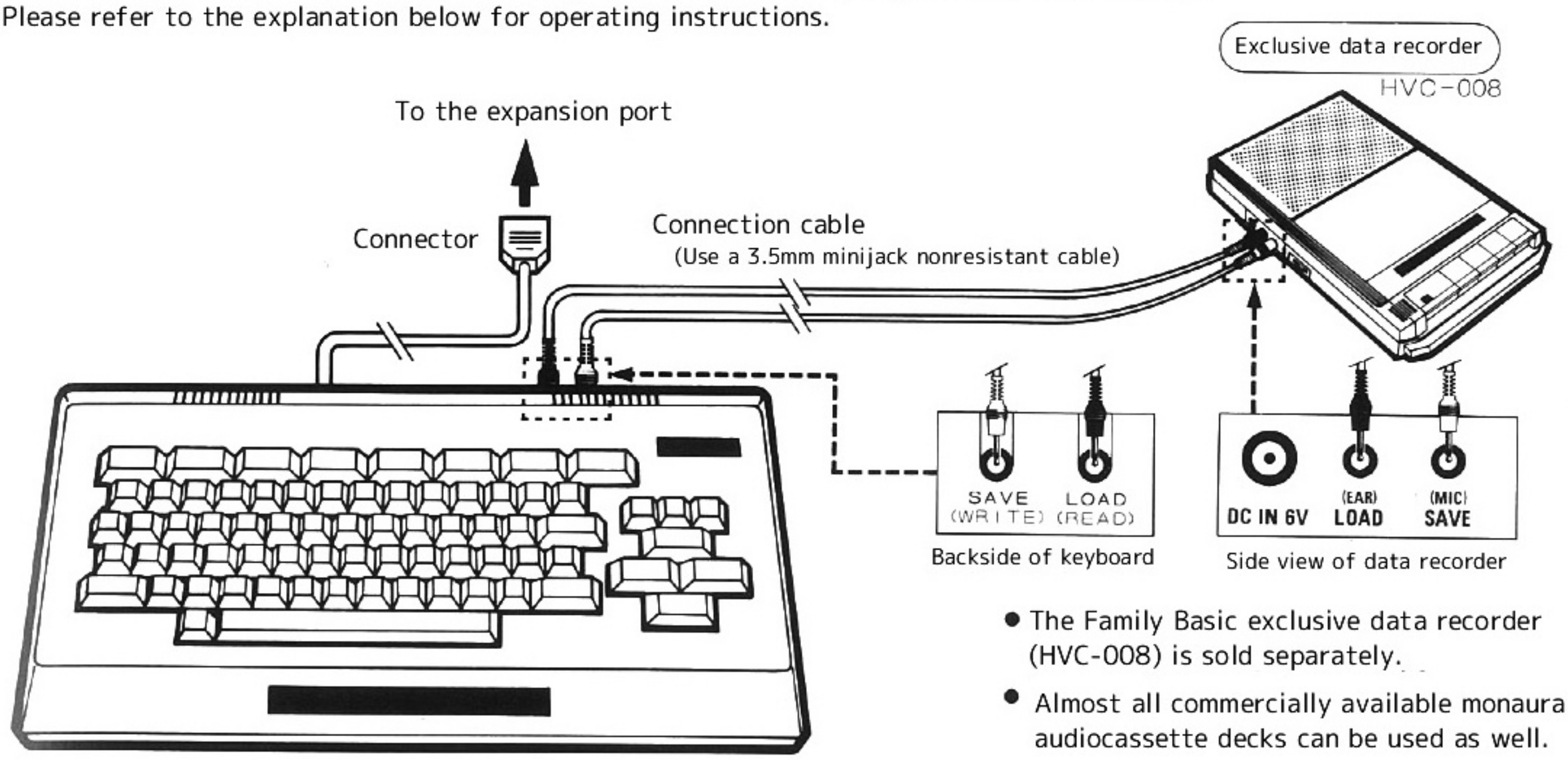
Since the background screen which was drawn in the BG GRAPHIC screen becomes copied (duplicated) onto the background screen, Mario from the sprite screen is displayed over the same picture from BG GRAPHIC.

★Warning: when doing changes or corrections in programs

- Before doing any changes or corrections to programs, delete the BG GRAPHIC screen (background).
  If you change or correct a program without deleting the background, errors can occur.
- Press the [CLR HOME] key while holding down the [SHIFT] key in order to delete the BG GRAPHIC screen. The cursor will return to the home position.
- Use LIST to call the program and execute the changes or corrections.

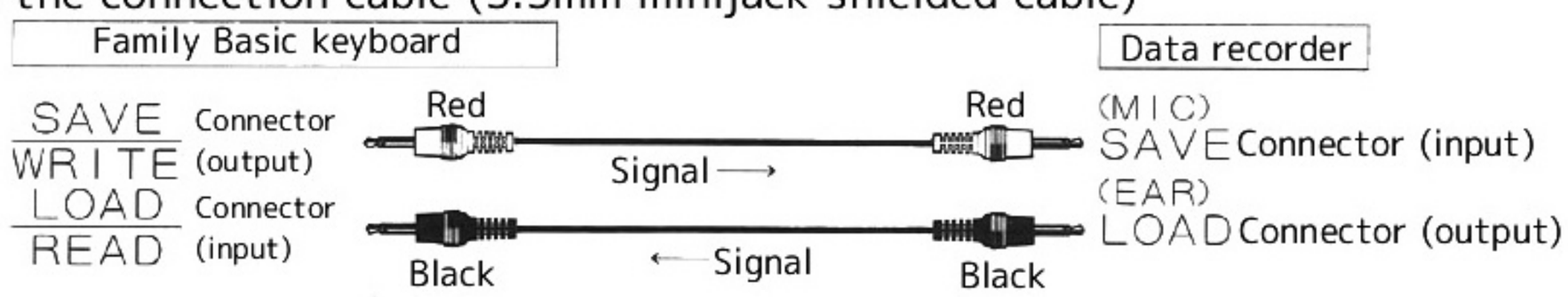
SAVE and LOAD programs

You can SAVE or LOAD the BG GRAPHIC background data or BASIC programs on a cassette tape.
Please refer to the explanation below for operating instructions.



- The Family Basic exclusive data recorder (HVC-008) is sold separately.
- Almost all commercially available monaural audiocassette decks can be used as well.

How to connect the connection cable (3.5mm minijack shielded cable)



※Do not use resistive connection cables.

About the knob setting of the official data recorder (HVC-008)

During LOAD, if you set the audio volume, the monitor volume from the speaker will change. Please set it to the volume of your choice. However, it will not load if the volume is very low. Also, if you are LOADing a tape saved on a different tape recorder / radio cassette deck, set the volume to the most appropriate level.

About the knob setting of a radio cassette deck

During SAVE or LOAD, set the radio cassette knobs as described below.

- Mode ·····Set to tape mode
  (mode in which you can record, play back cassette tapes)
- Audio volume ·····Find the best level for your deck, as this will vary for each type of deck.
  (in general, you should position the level to the middle)
- Audio quality ·····Set to "low".
  (this is the best setting for low sounds)

※When using your own radio cassette deck etc. (use a monaural radio cassette deck for audio)

- It can happen that you are unable to SAVE or LOAD depending on your device.
- Refer to the instructions manual of your monaural radio cassette deck when dealing with the connector cables, knobs etc.

About the tapes

Normal tapes (tapes for normal bias/normal equalizer)

※Save only 1 program per cassette tape.
This is convenient in case you need to LOAD a program, because you can shorten the file name or you don't need to rewind to the part where you saved it.

※We recommend short recording time cassette tapes (for example, 5 minutes per side, 10 minutes for both) in order to retrieve the program quickly or to be convenient to manage the program saving.

◆Use FILE (on the BG GRAPHIC screen) to SAVE or LOAD background pictures

★Instructions on how to save pictures made in BG GRAPHIC onto a cassette tape

① Introduce a cassette tape (normal tape) into the cassette recorder.
② Press the [ESC] key to display the function menu.
③ Press the [▼] key and select FILE.
④ Upon pressing the [SPACE] key you will see the following message.

```
SAVE(S),LOAD(L)?■
```

⑤ Upon pressing the [S] key on the keyboard, the following message will be displayed.

```
SAVE(S),LOAD(L)?S
FILE NAME"■
```

You will go back to select mode if you press any other key besides the [S] or [L] key.

⑥ Press the record button on the data recorder and press the pause button. (Standby-recording condition)
⑦ Enter the file name. Release the pause button of the data recorder to start the recording. Press the [RETURN] key to start the SAVE process.
File names can contain up to 16 characters.

```
SAVE(S),LOAD(L)?S
FILE NAME"TEST
WRITING TEST
```

⑧ When SAVE ends (1 m 30 s) select mode appears on screen.
⑨ Press the stop button of the data recorder to stop the recording.

★Instructions on how to LOAD the picture data created in BG GRAPHIC from a cassette tape

① Introduce the cassette tape which contains the data saved from BG GRAPHIC into the cassette tape recorder.
② Rewind the cassette tape and press the play button, listen to the speaker to get to the beginning of the data. Once you get there, adjust the audio volume to the adequate level.
③ Press the pause button to stop the tape.
④ Press the [ESC] key to display the function menu.
⑤ Press the [▼] key to select FILE.
⑥ Upon pressing the [SPACE] key, you will see the following message.

```
SAVE(S),LOAD(L)?■
```

⑦ Upon pressing the [L] key on the keyboard, the following message will be displayed.

```
SAVE(S),LOAD(L)?L
FILE NAME"■
```

You will go back to select mode if you press any other key besides the [L] or [S] key.

⑧ Enter the file name of the file you would like to load and press the [RETURN] key to start the loading process. Release the pause button to start the playback.

The BG GRAPHIC data will be skipped until the specified file name is found.
If you do not specify a file name, upon pressing the [RETURN] key, the file at the beginning will be read.

```
SAVE(S),LOAD(L)?L
FILE NAME"TEST
SKIP
LOADING TEST
```

If you press the [STOP] key during load or if a reading error occurs, you will return to select mode.

⑨ When LOAD has finished, the picture will appear on the BG GRAPHIC screen and you will return to select mode.

⑩ Press the stop button on the data recorder to stop the playback.

※Please refer to p. 48 if an error occurs or if the data cannot be read.
※In case of a read error or if you pushed the [STOP] key during the reading or if the LOAD is stopped, an incomplete picture might appear on the BG GRAPHIC screen. Execute CLEAR or go back to the GAME BASIC mode, select BG GRAPHIC once more and clean up the screen.

◆How to SAVE and LOAD a BASIC program

★Instructions on how to SAVE a program on a cassette tape

① Introduce a cassette tape (normal tape) into the data recorder.
② Press the record button on the data recorder and press the pause button. (Standby-recording condition)
③ Use the keyboard to enter the SAVE command.

```
SAVE"TEST"■
```

Keep the program name under 16 characters.

④ Release the pause button on the data recorder to start the recording.
⑤ Upon pressing the [RETURN] key, the program data goes from the keyboard to the data recorder and the following screen appears.

```
SAVE"TEST"
WRITING TEST
```

⑥ When the program SAVE ends (program data export), the following appears on screen.

```
SAVE"TEST"
WRITING TEST
OK
■
```

⑦ Press the stop button on the data recorder.
⑧ Check if it has been SAVEd correctly. (Please refer to LOAD? (check))

※If the leader tape part of the cassette tape lasts more than 3 seconds, the program might not be saved correctly. When using a cassette tape with a long leader tape, start saving after fast forwarding the beginning of the tape without recording anything.

★Instructions on how to check if the program has been saved correctly onto the cassette tape by using LOAD? (check)

① Rewind the tape which you used to SAVE until the part where you started to SAVE.
② Enter the [L][O][A][D][?] command from the keyboard.

```
LOAD?■
```

③ After pressing the [RETURN] key, upon pressing the play button on the data recorder, the cassette tape will be played back. The screen displays the following.

```
LOAD?
LOADING TEST
■
```

name of the program

④ LOAD? ends and if the program has been saved correctly, the screen displays the following.

```
LOAD?
LOADING TEST
OK
■
```

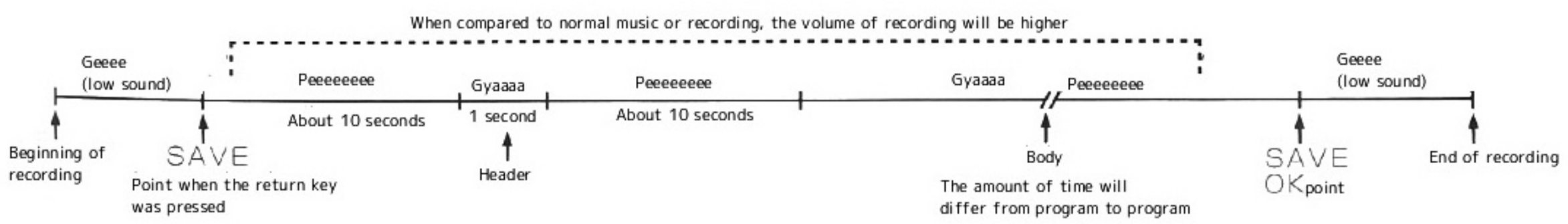
If the program has not been saved correctly, the following will appear on your screen. Please try to save again.

```
LOAD?
LOADING TEST
?TP ERROR
OK
■
```

If after saving again you still receive an error message, please check the recording status of the cassette tape or try with a different tape and execute SAVE and LOAD? once more.

※Please refer to the next page in case an error occurs.

When program data has been saved onto a cassette tape and you play it back from the speakers of the data recorder, you will hear the following type of sounds.



★How to load a program from a cassette tape

① Insert the cassette tape which contains the saved program into the data recorder. Forward the tape until the beginning of the program which you want to load.

② Use the keyboard to enter the LOAD command.

```
LOAD"TEST"■
```

Name of the saved program which you would like to read

Keeps skipping until it finds the requested program.

③ When pressing the pause button on the data recorder after pressing the [RETURN] key, the keyboard will read the program from the data recorder. You will see the following displayed on screen.

```
LOAD"TEST"
LOADING TEST
```

※ When pressing only LOAD [RETURN] key, it will only read the first program it finds.

④ When the program finishes LOADing, you will see the following on screen.

```
LOAD"TEST"
LOADING TEST
OK
■
```

When the program has not been loaded correctly, you will see the message below displayed on screen. LOAD again.

```
LOAD"TEST"
LOADING TEST
?TP ERROR
OK
■
```

※Please refer to the paragraph about errors here below.

★About the errors which occur during LOAD, LOAD (check) (Refer to this item as well in case of errors which occur during SAVE, LOAD for BG GRAPHIC)

When an error (?TP ERROR) is displayed during the LOAD or LOAD? of a BASIC program and the program can't be read from the data recorder, keep the following in mind and retry SAVE, LOAD and LOAD?.

When SAVE hasn't been executed correctly

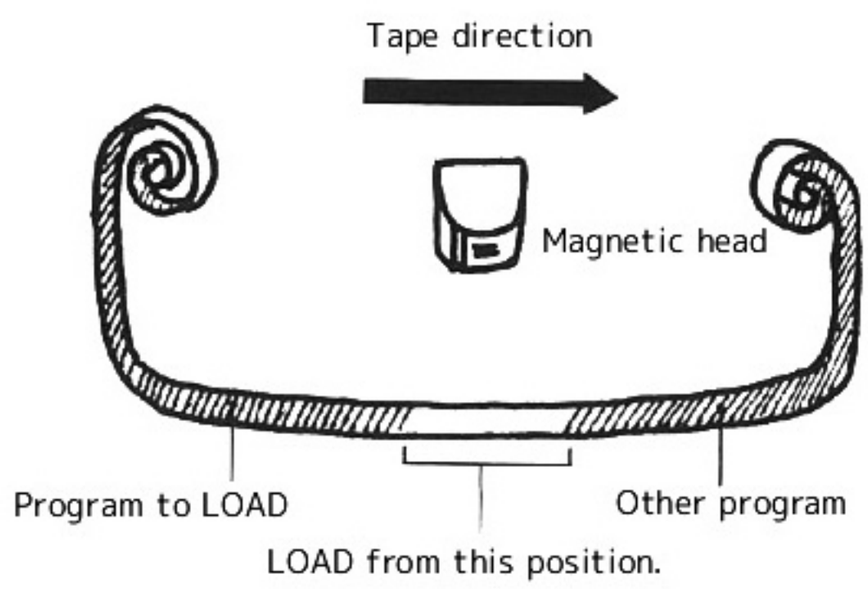
(Cause)
- The connection between the keyboard and the data recorder or the operation of the radio cassette recorder is incorrect.
- The cassette tape type is incompatible with the tape recorder.
- The cassette tape is damaged or the audio on the tape is uneven.
- There's an intrusion of noise while saving.
- When using a radio cassette recorder other than the exclusive data recorder for Family Basic, depending on the model, you can monitor the recording with headphones. When monitoring, howling can occur while saving.

(Treatment)
- Change the save position on the cassette tape or try with a different cassette tape.
- For models which allow monitoring, when saving, remove the cable from the headphone connector on the recorder.

SAVE is ok, LOAD isn't

(Cause)
- The connector between the keyboard and the data recorder or the playback volume of the recorder is incorrect.
- When using a radio cassette recorder other than the exclusive data recorder for Family Basic, depending on the model, microphone mixing is available during playback. This can create noise during LOAD.

(Treatment)
- Try setting the knobs of the cassette recorder (decreasing the volume little by little or changing the sound quality) and check the connection cable.
- Check the SAVE position and LOAD the cassette tape from the point where the sound starts.
- For models which allow microphone mixing, remove the cable from the microphone connector on the recorder when executing LOAD.



The program will not LOAD correctly even if you LOAD from a position which is not the beginning of the saved program (from a different program or from the middle of the program you want to LOAD). When loading a program, always start from the beginning of the program.

Warning) When placing the data recorder or the cassette recorder close to a TV set, the noise from the TV can be the cause of (?TP ERROR) and you will not be able to LOAD at all.
In such a case you should place your data recorder or cassette recorder further away from your TV set.

BASIC GRAMMAR

BASIC Specification Standards

Type of characters	numbers/alphabet/kana/symbols
Display range for numbers (integers)	decimal (-32768 to + 32767) hexadecimal (&H0000 to &HFFFF) line of characters (0 to 31 characters)
Variable name types	Besides the 2 letters at the beginning, up to 255 characters
Range for line numbers	0 to 65535
Digits per line of text	255 digits
Size of array	Up to 2 dimensions, no element limit (within scope of memory)
Multi statement	Possible : separate by colon
Sub-routine, nest	No limit (within scope of memory)
FOR NEXT loop	No limit (within scope of memory)
Editing function	Screen editor

Screen modes	BG GRAPHIC screen, sprite screen, background screen, backdrop screen
Graphical resolution	Background screen (28 characters x 24 lines) Sprite screen (256 x 240 dots) 1 character (8 x 8 dots)
Colors	Color generator for 52 colors
Sound function	Scale, tempo, 3 note polyphony, tone
Controller input	Left-right controller (directional/trigger input possible)
File function	Cassette tape (1200 bauds)
Amount of commands	Basic 74
Animated character setup	Selection among 16 types of preset characters

※The display of results from calculations which handle numbers beyond the display range of -32768 to +32767 is not guaranteed.

Warning
This manual was created based on NS-HUBASIC.
(1) Please check the version number carefully, as the system software version or this manual might have been modified without notice (NS-HUBASIC V2.0A).

(2)It is forbidden to duplicate the system software and also this book.
(3)A lot of effort was put into the development of this extremely complicated product, including its manual.
If you were to find a defect within these, please contact Nintendo or the place where you bought this product.
Moreover, please understand that we can not be held responsible for any effects of what you create with this.

Command index by function

( )contain the abbreviation of the command.

Commands

General statements

Screen control statements

MOVE Commands

Particular statements

Sound control statements

Functions

● Integer functions

● Character functions

● Particular functions

● Input output character functions

Sprite control statements

Operators

+, - type of symbols which are used in calculations are called operators. There are 3 types of operators: arithmetical operators, relational operators and logical operators.

Arithmetical operators

There are 5 arithmetical operators.

Operator	Operator content	Example	Mathematical display
+	Addition	A+B	A+B
−	Subtraction	A−B	A−B
*	Multiplication	A*B	AB
/	Division	A/B	A÷B or $\frac{A}{B}$(yet 、B≠0)
MOD	Residue	A MOD B	Remainder

MOD calculates the remainder of a division.
For example 10:3=3 remainder 1, when you enter
10MOD3 RETURN
the result will be 1.
Operations are executed in the following order. This order is considered as a priority order, but in case of equal priority, the operations are executed from the operator on the left.

(priority order) (arithmetical operators)

1. * /
2. MOD
3. + − (addition, subtraction)

In case you would like to change the order of operation, use ( ) to enclose the operation which you would like to execute first.

(Display in BASIC)	(Mathematical display)
A*X+Y	AX+Y
(X+Y)/2	$\frac{X+Y}{2}$
X+Y/2	$X+\frac{Y}{2}$

Warning: NS-HUBASIC is an integer type. It can not calculate with decimals.
In case of the result for a division, it will display integer results, getting rid of the decimals.

Relational operators

Relational operators are used when comparing fixed values or variables. The result becomes -1 if true, 0 if false.
The relational operators are listed below.

(relational operator) (meaning)

=	Both sides are equal
<>	Both sides are not equal (≠) You can not use ><.
>	The left side is greater than the right one
<	The left side is smaller than the right one
>=	The left side is greater than or equal to the right side (≧) You can not use =>.
<=	The left side is smaller than or equal to the right side (≦) You can not use =<.

Relational operators are used within programs using IF statements like below.

IF X>0 THEN 1000 (meaning) if the value of X is greater than 0, jump to no. 1000.

Please refer to IF-THEN statements for more details.
Also,

A=X>0 or A=(X>0)

when used like this, if X is greater than 0, A is replaced by -1 and if X is smaller than or equal to 0, A is replaced by 0.
Thus, you can know from the value of A if X is greater than 0 or not.

Logical operators

Logical operators are the symbols which execute bit unit 0 and 1 type of boolean calculations or other bit operations (these are called logical calculations) and there are 4 of them.

Calculation order	Logical operator	Abbreviation	Meaning
1.	NOT	NO.	not (negation)
2.	AND	A.	and (logical product)
3.	OR		inclusive or (logical sum)
4.	XOR	XO.	exclusive or (exclusive logical sum)

Logical calculations are executed between 2 values, true is 1, false is 0, and the result (value of truth) is also displayed as one of two values, 1 (true) or 0 (false). We're going to summarize each of the 6 logical operators from above

NOT (negation)

X	NOT X
1	0
0	1

Warning) This shows that when X equalss 1, NOT X equals 0, and when X equals 0, NOT X equals 1.

AND (logical product)

XY	X AND Y
11	1
10	0
01	0
00	0

Warning) This shows that when both X and Y equal 1, only then 'X AND Y' equal 1, other times it will equal 0. This applies exactly to X and Y multiplication and its result.

OR (logical sum)

XY	X OR Y
11	1
10	1
01	1
00	0

Warning) This applies to the sum of X and Y and their sum.

XOR (exclusive logical sum)

XY	X XOR Y
11	0
10	1
01	1
00	0

Warning) Exclusive means that when X and Y have the same value, it becomes 0 (=false).

Warning) When writing chains of variables and logical operators, leave spaces between those.

(Examples) NOT 13 13= (0000 0000 0000 1101)₂
∴NOT13=(1111 1111 1111 0010)₂
=-14

●15AND5 15= (0000 0000 0000 1111)₂
5= (0000 0000 0000 0101)₂
∴15AND5= (0000 0000 0000 0101)₂
=5

●50 OR 44 50= (0000 0000 0011 0010)₂
44= (0000 0000 0010 1100)₂
∴50 OR 44= (0000 0000 0011 1110)₂
=62

●42XOR36 42= (0000 0000 0010 1010)₂
36= (0000 0000 0010 0100)₂
∴42XOR36= (0000 0000 0000 1110)₂
=14

※ $(000\cdots)_2$→Used to display binary numbers.

Logical operators used in IF sentences can change the flow of a program.

(Examples) IF X>0 AND X<10 THEN 1000

(Meaning) if the value of X is between 0 and 10, jump to line 1000.

IF X<0 OR X>10 THEN 1000

(Meaning) if the value of X is less than 0 or greater than 10, jump to line 1000.

The 3 types of operators have the following order of execution.

1. Arithmetical operators
2. Relational operators
3. Logical operators

We can summarize the operator order of execution as follows:

1. Part between ( )
2. Functions
3. ＊、／ (Multiplications and divisions)
4. MOD (Remainders)
5. ＋、－ (Additions and subtractions)
6. =、<>、>、<、>=、<=、
7. NOT (Negations)
8. AND (Logical products)
9. OR (Logical sums)
10. XOR (Exclusive logical sums)

Special symbols

In BASIC there are also other operators which also have a specific function and are represented by the symbols here below.

(1) — (Hyphen)

Used when defining the scope of LIST sentences or lines, specifying from which line to which line.

(Example) LIST 100-300

(2) , (Comma)

Used as a separating symbol (separator) when operands are lined up in PRINT, INPUT or DATA sentences.

(Example)
```
PRINT A,B,C
INPUT A,B,C
DATA 10,13,9,14,13
```

(3) : (Colon)

Used as a separating symbol in a multi-statement.

(Example)
```
A=B-C:PRINT A
FOR I=0 TO 9:PRINT X$(I):NEXT
X1=X:Y1=Y:GOSUB10000:GOSUB20000
```

(4) ; (Semi-colon)

Used as a separating symbol in a PRINT sentence.

(Example)
```
PRINT"answer=";A
INPUT"A=";A
```

(5) ? (Question mark)

Can be used as a substitute for a PRINT sentence.

(Example) ?A,B,C

(6) &HOO shows a hexadecimal

All others are decimals.

(Example) &H2C is 44 in decimals.

How to read the Grammar

In this chapter we will explain the NS-HUBASIC commands using the structure from below.

Working — Explains the working of the command. This will help you to understand the meaning of the command.

Grammar — The grammar of the command is written in its general form for you to understand how to use it.
Moreover, the following symbols and characters are used in the same way throughout to define the following:
1. Capital alphabet characters are to be used as is.
2. Anything between 〔 〕 can be omitted at will by the user.
3. Any of the elements listed vertically between { } can be selected at will by the user.
4. ( ) , 「,」 (comma), 「:」 (colon), 「;」 (semi-colon), 「—」 (hyphen), 「=」 (equal) is to be entered correctly in the specified position.
5. ..... means that you can repeat it at will within the scope of one line (255 characters).
6. 「 」 (empty space) means that you must leave a space open. However, there are times where you can ignore this.
7. " " shows an example of characters which you can use.
8. Other expressions used in grammar will be explained right below the grammar part or in the Explanation part.

Abbreviation — Shows the abbreviated form of the command.

Explanation — Explains in detail how to use a command, its working and points to pay attention to.
☞ (Please refer to this) shows related commands or elements.

Sample Program — Shows an appropriate example or result of execution.
It might differ from what appears on-screen because it is written as a program list. On-screen, even if the line changes, enter the programs with the same line number successively.

Default Value — When the parameters of the command are not specified, it means the value of the parameter is set automatically.
When entering the BASIC screen from the GAME BASIC mode screen, the following are set as below:
CGEN2
SPRITE OFF
POSITION n,120,120
PLAY"03V15T4M0" or musical pitch length of 5
The color palette will use, for both the background and the sprites, the background palette code 1 color scheme.

Command

CLEAR

Working Assigns the top address of the memory region which you can use in BASIC.

Grammar address

Address -> Top address of the memory region which you can use in BASIC
&H77FF or less
(refer to the memory map on p. 104)

Abbreviation CLE.

Explanation Assigns the top address of the region which BASIC uses within the memory.
When using a machine language program concurrently with a BASIC program, it is necessary to assign the top address of the memory region used by BASIC to avoid destruction by BASIC programs or variables.
At the same time CLEAR assigns the top address, nesting with FOR-NEXT, GOSUB, etc. or variables and arrays will be erased.
This command can not be written in the header of a regular program or in a subroutine. Moreover, an assigned address remains valid until it is reassigned by CLEAR.
When skipping an address with CLEAR, variables and array variables which have number variables assigned to them are cleared into 0, those with character variables are cleared into " " (null string).
☞ Refer to p. 61 for CLEAR

Sample Program

```
10 REM * CLEAR *
20 CLEAR &H7600
```

The top address of the memory region which you can use in BASIC is assigned to &H7600. Check it with PRINT FRE.

NEW

Working Deletes the whole program.

Grammar NEW

Abbreviation N/A.

Explanation When entering a new program, deleting an old program from the memory entered until then will avoid trouble. In order to do this, you should use the NEW command to delete the old program.
NEW doesn't just delete the program, it also deletes all of the variable content from the memory. However, even if you execute NEW, you will not change the status of the user area or the screen within the memory.
Also, there is no abbreviated form for NEW, in order to avoid deleting a program by mistake. Please be careful when using this command.
(Do not use it within a program)

Sample Program

```
10 REM * NEW *
20 X=999
30 PRINT X
LIST
10 REM * NEW *
20 X=999
30 PRINT X
OK
RUN
 999
OK
NEW
OK
LIST
OK
■
```

Entering the program.
Checking the program list.
The program was entered without error.
Execute the entered program.
The 999 value of X is displayed on-screen.
Deleting the program.
Checking the program list.
The program has been deleted

LIST

Working Displays the program from the memory on the screen.

Grammar LIST [(m)] [– (n)]

m→Number of the first line to be displayed.

n→Number of the last line to be displayed.

Abbreviation L.

Explanation LIST is the command to display on screen the program which has been entered in the memory in order to look up or edit that program. By specifying the line numbers, you can freely decide which part to display.

LISTm.....display only number m...(1)
LISTm, or LISTm -.....display number m and above...(2)
LISTm, n or LISTm - n...display from number m to n...(3)
LIST, n or LIST - n.....display up to number n...(4)
LIST.....display everything...(5)

If the specified line number does not exist within the program, BASIC will display the following.
In case (1), it won't display anything and will wait for your next command.
In case (2) it will start displaying from the first number bigger than m. If there's nothing after m, it won't do anything and just await your next command.
In case (3) it will do the same process as (2) for the first m, the same process as (4) for the last n, and finish displaying.
In case (4), it will display up to the first number smaller than n. If there's nothing smaller than n, it won't do anything and just await your next command.

Moreover, pressing the [ESC] button will halt the display temporarily. Press any key to resume the display.

Sample Program

```
10 REM * LIST *
20 INPUT X
30 Y=X*X
40 PRINT X;Y
50 END
```

Entering the program.

```
LIST 30
30 Y=X*X
OK
```

Example of (1) lists only program line number 30.

```
LIST 40-
40 PRINT X;Y
50 END
OK
```

Example of (2) lists program line number 40 and above.

```
LIST 20-40
20 INPUT X
30 Y=X*X
40 PRINT X;Y
OK
```

Example of (3) lists program lines from number 20 to 40.

```
LIST -20
10 REM * LIST *
20 INPUT X
OK
```

Example of (4) lists program lines up to number 20.

```
LIST
10 REM * LIST *
20 INPUT X
30 Y=X*X
40 PRINT X;Y
50 END
OK
■
```

Example of (5) lists all of the program lines.

RUN

Working Executes the program.

Grammar RUN n

n→Number of the line to start the execution.

Abbreviation R.

Explanation RUN is the command to execute a program.
When entering RUN, the program will be executed in order from the beginning, but at this time, all the variables will be cleared.
Also, in order to start executing the program from the middle, you can add the n number of the line from which you'd like to start.
(Example: RUN 1000 [RETURN])
After stopping the program by pressing [STOP], upon resuming the execution, when using RUN or RUN n, since the content of the variables entered until then has been cleared, if you want to execute without clearing the variables, please use GOTO (Refer to GOTO) or CONT.
(Example: GOTO 1000 [RETURN])
☞ Refer to GOTO, CONT

Sample Program

```
10 REM * RUN *
20 PRINT "ハイ!"
30 PRINT "コンニチハ"
40 END
```

Entering the program. Translator's note: line 20 means "YES!", 30 "Hello".

```
RUN
ハイ!
コンニチハ
OK
■
```

Executing the program. The program is being executed.

CONT

Working Resume the program.

Grammar CONT

Abbreviation C.

Explanation In case the program stopped when executing the STOP command or when pressing the STOP key, when pressing CONT, the program will start executing from the command from the next line number. In this case, the variables from before stopping will remain unchanged.
However, when the program has stopped because of END or because an error occurred, right after executing CLEAR (refer to CLEAR), or else when rewriting the program after stopping the program, CONT will not execute correctly and an error will occur (CC ERROR). However, if it can resume, OK. (with a full stop) will appear and when it cannot resume, OK (without a full stop) will appear.

☞ Refer to STOP, END

Sample Program

☞ Refer to STOP

LOAD

Working Read a program from a cassette tape.
(⇔ SAVE)

Grammar LOAD ("File name)

File name->name added to a program, 16 characters or less

Abbreviation LO.

Explanation Use the LOAD command to save in the read memory a program which is saved on a cassette tape.
When specifying the file name, it will skip the programs until it finds the program with that name and will read it.
The file name is omitted only when reading the first program on the cassette tape.
Please refer to p. 45 to learn how to use this command.

Sample Program

```
LOAD
LOADING TEST
OK
■
```

.....LOAD the program from the cassette tape without specifying the file name.

```
LOAD"TEST-2"
SKIP TEST
SKIP TEST-1
LOADING TEST-2
OK
■
```

.....Specify the file name and LOAD.
.....If the file name of the program differs, it will SKIP it.

SAVE

Working Record a program on a cassette tape. (⇔ LOAD)

Grammar SAVE ("File name")

File name -> Name added to the saved program, 16 characters or less

Abbreviation SA.

Explanation The save command is used to save the memorized programs on a cassette tape.
The file name is the name added to the saved program and delimited by " (double quotation marks).
You may omit the file name but when managing several programs, adding file names helps keeping them apart.
Please refer to p. 45 to learn how to use this command.

Sample Program

```
SAVE
WRITING
OK
■
```

.....SAVE without adding a file name.

```
SAVE"TEST"
WRITING TEST
OK
■
```

.....Adds a file name (ex.: TEST) and SAVEs the program on the cassette tape.

LOAD?

Working Checks whether a SAVEd program has been correctly saved as a file.

Grammar LOAD? ("File name")

Abbreviation LO.? or LO.P.

Explanation Use the LOAD? command to check whether the program saved on the cassette tape is the same as the program saved in the memory.
When specifying the file name (name added to a program, 16 characters or less), all the other programs will be skipped until it finds the program with that name. When it finds the specified program, the checking starts.
When omitting the file name, the program of the first found file name will be checked against the program in the memory.
When a program is SAVEd on a cassette tape, it is important to check whether it has been saved correclt by using LOAD?.
Please refer to p. 47 to learn how to use this command.

Sample Program

```
LOAD?                 ·····In case there is no file name.
LOADING TEST          ·····Checking.
OK                    ·····Check result OK check finished.
■

LOAD?"TEST"           ·····In case a file name was added.
LOADING TEST
OK
■

?TP ERROR             In case the check result is
OK                    unavailable.
■                     Please SAVE again.
```

General Statements

Substitutes

Working Substitutes values of variables in functions.

Grammar Variables = Functions

Explanation A substitution substitutes the value of the function to the right of the "=" with the variable value to the left.
While the function to the right may contain just invariables or variables, there can only be variables to the left. The variables to the right must be defined before (this is not limited to the previous line). If not defined, a variable, in case of a numerical variable, will be seen as 0, in case of a character variable, as " " (null string).
Also, when connected by "=", both left and right sides have to be numerical value functions or character variable functions. (You cannot mix character variables and numerical variables.)

(Correct) X=A Y=A*X+10 X$="&H"+A$
(Incorrect) X$=A Y$=A*X+"10"
X="&H"+A$

Sample Program

```
10 REM ---Substitutes---
20 A=10            ·····Substitutes numerical value 10 with numerical variable A
30 A$="ABC"        ·····Substitutes characters ABC with character variable A$.
40 PRINT A$;A      ·····Prints variables A$ and A.
RUN
ABC 10             ·····Substituted by A$ and A.
```

PRINT

Working Displays information such as results of computation etc. on screen.

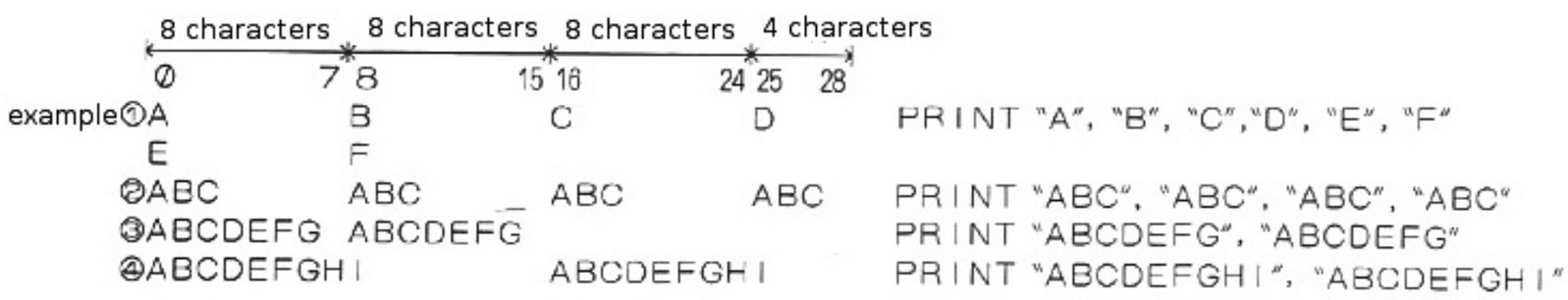
Grammar PRINT [Constant / Variable / Function] [; / , [Constant / Variable / Function] ·····]

Constants, variables, functions -> information displayed on screen

Abbreviation ? or P.

Explanation PRINT outputs constant and numerical variable values, character variable values (character strings) as well as function values (calculation results) on screen.
PRINT means "to print out" but this all originates from the first output devices called the teletypewriter.
Because PRINT is a statement which is used a lot, we decided to abbreviate it as "?".
When outputting values with PRINT, these can be separated by ";" or "," to write several multiples. In this case ";" and "," are called "separators".
When using ";" as a separator, you can write immediately after a character. However, a " " (blank) is needed right before and right after the symbol of a value (for a positive value).
When using "," as a separator, it is output consecutively in units of 8 characters on screen. The display field on screen is divided in 4 blocks like below and the output information is always displayed at the beginning of each block.

Example of display on screen



Moreover, when writing character constants, only when there's no separating symbol at the end, you may omit the double quotation mark at the end. Also, when adding ";" at the end of a PRINT sentence, the content of the following PRINT sentence will be displayed in connection.
When using PRINT by itself, there will be a one line break.

INPUT

Working Inputs numerical values or characters from the keyboard.

Grammar INPUT ["char. string"] { ; / , } variable[,variable, ···]

Character string -> character string to be displayed on screen.
Variable -> variable which takes the numerical value or character entered by keyboard

Abbreviation I .

Explanation Enters the numerical value and character data input with the keyboard into variables. For certain types of data, it is necessary to prepare variables to enter the data in advance. For that reason, you must add a variable after INPUT. You can use either a numerical variable or a character variable. Also, you can use a "," (comma) as a separator in order to add several of them. In this case, when inputting data, separate the corresponding data with "," and input once. The entered amount of data has to match the amount of variables after INPUT.
You can enter almost any kind of character variables. However, you should use a " (double quotation mark) to surround a "," (comma). This is because "," is used as a separator.
When executing an INPUT sentence, a "?" appears on screen, awaiting your input, but with "?" only you would not know what to enter. This is why there is a character string PRINT function included in INPUT. You can use " to surround the character string which you would like to display on screen and use ";" (semi colon) to interleave and write variables.
Also, if you do not wish to display "?", you may use " to surround a character string and "," to interleave between variables.
When specifying the input of character variables and entering numerical values, numbers are considered as characters, but when specifying the input of numerical variables and entering characters, these are considered as 0 value variables. For both numerical variables and character variables, when pressing the RETURN key without entering anything, the current value or characters which were attributed to that variable will disappear.

Sample Program

The following sample is a program which adds, subtracts and multiplies the data read on line 20 and 30.

```
10 REM * INPUT *
20 INPUT"A=";A
30 INPUT"B=";B
40 C=A+B
50 D=A-B
60 E=A*B
70 PRINT"A+B=";C
80 PRINT"A-B=";D
90 PRINT"A*B=";E
```

(The computer can handle numerical values from -32768 to +32767.)

LINPUT

Working Inputs characters from the keyboard.

Grammar LINPUT ["char string"] { ; / , } character variable

Character variable->variable which takes the character string entered by keyboard

Abbreviation LIN.

Explanation LINPUT is like INPUT a statement to input data from the keyboard, but what is different is that "," can be input as data as well.
Also, you can only input character strings of up to 31 characters and you can only specify one variable to input. LINPUT also includes the character string PRINT function, like INPUT, which allows you to display a message on screen about the characters to enter. However, because in LINPUT this message is also entered as data within a variable, when you want to turn only the entered characters into data, it is necessary to enter a processing in the program which will remove the part of the message.
Yet, the amount of characters you can input in this case must be 31 or less, inlcuding the characters of the message.
Moreover, like in INPUT sentences, "?" will not appear, and you can interleave character strings and character constants surrounded by " with either ";" or "," and get the same effect.

Sample Program

```
LINPUT"STRING=";A$
STRING=23.5,"",,,;
OK
PRINT A$
STRING=23.5,"",,,;
OK
■
```

CLEAR

Working Clears variables or arrays.

Grammar CLEAR

Abbreviation CLE.

Explanation Clears all the variables or arrays completely from the memory. Namely, it turns numerical types into 0 and character types into null strings (empty status).

Sample Program

```
X$="CLEAR"        ·····Substitute X$ with the characters CLEAR.
OK
Y=125             ·····Substitute Y with numerical value 125.
OK
PRINT X$;Y        ·····Print and check the values of X$ and Y.
CLEAR 125         ·····These are substituted.
OK
CLEAR             ·····Let's clear them.
OK
PRINT X$
OK
PRINT Y           ···They have been cleared.
 0
OK
```

DIM

Working Declares an array.

Grammar DIM name of array(m1 [, m2])[, name of array(n1 [,n2])
, ...]

Name of array->name of array, numerical variable, character variable O.K.

Abbreviation DI.

Explanation Sets the name of the array and the amount of dimensions, as well as the size of additional characters.
You can declare multiple arrays within one DIM sentence, and you can specify additional characters of up to two dimensions within the scope of the memory for each array.
After declaring the array, all of the content of the array become cleared, numerical type arrays become 0,

Sample Program

(1)Example of a numerical type array

```
10 REM * DIM --- (1) *
20 DIM A(3),B(3,3)
30 FOR I=0 TO 3
40 A(I)=I
50 PRINT A(I);
60 NEXT
70 PRINT:PRINT
80 FOR I=0 TO 3
90 FOR J=0 TO 3
100 B(I,J)=I*10+J
110 PRINT B(I,J);
120 NEXT
130 NEXT
```

---array declaration

Substitute and display the numerical values within a one dimensional array A(1).

Substitute and display the numerical values within a two dimensional array B(I,J).

(2)Declaration of a character type array

```
10 REM * DIM --- (2) *
20 DIM A$(3),B$(3,3)
30 FOR I=0 TO 3
40 A$(I)="TEST"+STR$(I)
50 PRINT A$(I)
60 NEXT
70 PRINT
80 FOR I=0 TO 3
90 FOR J=0 TO 3
100 B$(I,J)="TOKYO"+STR$(I)+STR$(J)
110 PRINT B$(I,J)
120 NEXT
130 NEXT
```

---Array declaration

Substitute and display the numerical values within one dimensional array A$(1).

Substitute and display the numerical values within one dimensional array B$(I,J).

GOTO

Working Jumps unconditionally to the specified line number.

Grammar GOTO {line number}

Line number->jump target line number

Abbreviation G.

Explanation GOTO is a statement to jump unconditionally to a specified line number and used to change the flow of a program. When used as a direct command, you can execute a program from a specific line number without clearing the values of constants.

(Ex. GOTO 2000 RETURN)

☞ Please refer to RUN

Sample Program

```
10 REM * GOTO *
20 A$=INKEY$(0)
30 PRINT A$,ASC(A$)
40 IF A$="Z" THEN END
50 GOTO 20
```

Awaits the input of any key.

Displays the ascii code of the pressed character or symbol

Pressing the Z key will end the program.

Jumps to line 20 and executes line 20.

GOSUB

Working Calls a subroutine within a program.

Grammar GOSUB {line number}

Line number->Subroutine start number

Abbreviation GOS.

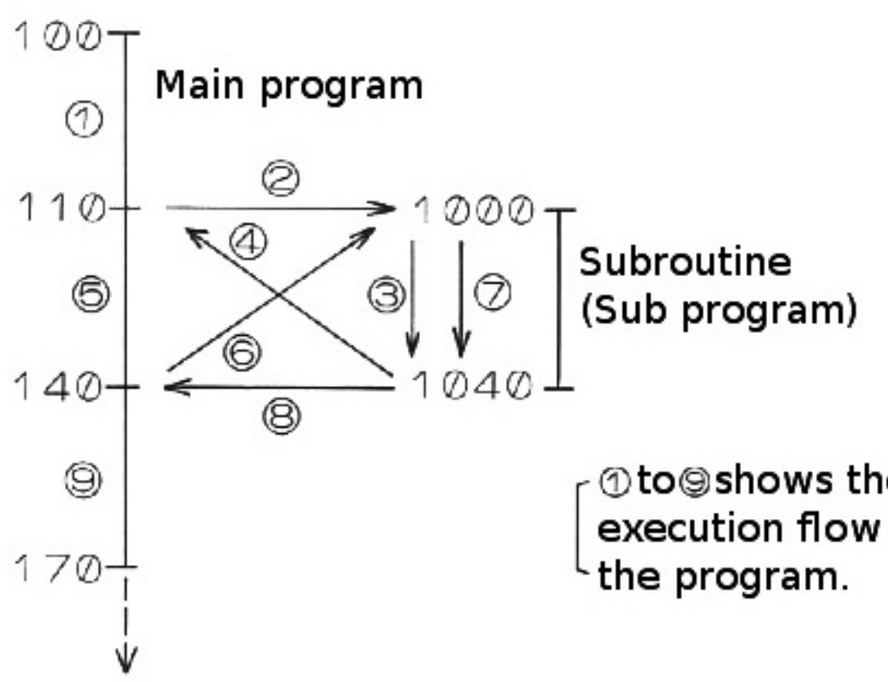
Explanation Loops made inside IF sentences (refer to IF-THEN) or FOR-NEXT sentences (refer to FOR-NEXT) execute a repetition of the same process for an infinite number of times, but when you need to repeat the same process in several places within a program, you can include the part which you would like to repeat within a sub program and when necessary call it from the main program. Calling a sub program from a main program is called a subroutine and you can use the GOSUB statement to call the subroutine.
The line number which comes after GOSUB is the first line of the sub routine and you must add RETURN on the last line of the subroutine. You can also write other GOSUB sentences within a subroutine. (However, these are limited by the memory's capacity.)

Sample Program

```
10 REM * GOSUB *
100 FOR I=1 TO 25
110 GOSUB 1000
120 NEXT
130 FOR I=25 TO 1 STEP -1
140 GOSUB 1000
150 NEXT
160 PRINT "END"
170 END
1000 FOR J=0 TO I
1010 PRINT"*";
1020 NEXT
1030 PRINT
1040 RETURN
```

Main program

Sub program



①to⑨shows the execution flow of the program.

The main program goes from line 100 to 170. The sub program goes from line 1000 to 1040. Lines 110 and 140 of the main program call the subroutine. The program's flow is described in the diagram on the left.

When adding annotations in the subroutine of a GOSUB sentence, you can not use "REM". Please use " ' ".

RETURN

Working Returns from the subroutine.

Grammar RETURN [{line number}]

Line number->line number of the subroutine to return to

Abbreviation RE.

Explanation RETURN is placed at the end of a subroutine called by GOSUB to return.
When defining a line number, it returns to that line, but when omitting it, it will go to the end of the GOSUB sentence.

Sample Program

☞ Please refer to GOSUB.

IF—THEN

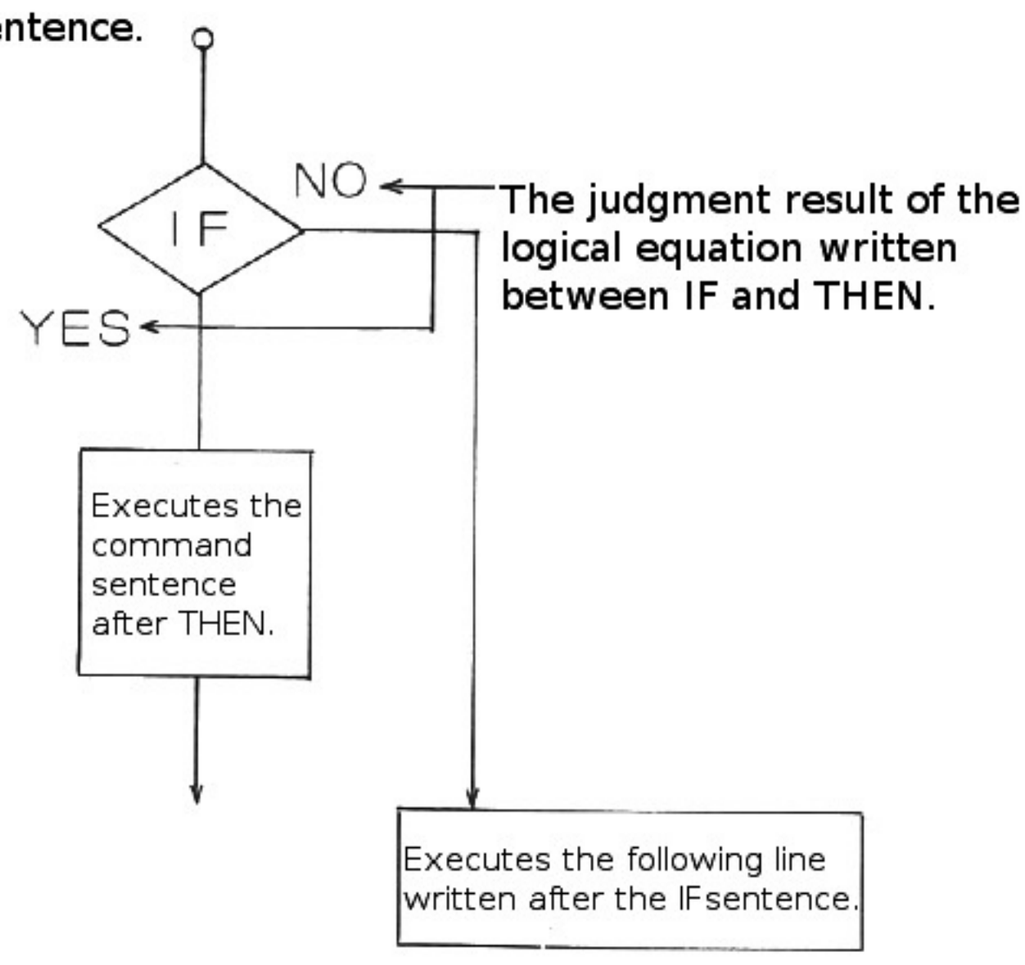
Working Creates a ramification in a logical equation.

Grammar IF equation THEN {line number / sentence}

Equation->logical equation, refer to p. 52.
Line number->line number to jump to
sentence->optional statement

Abbreviation IF—T.

Explanation IF sentences can be used with THEN as a condition ramification executing statement. Write the logical equation between IF and THEN, and in case the logical equation is established, it executes the part after THEN, if it is not established, it executes the next line. For example:
IF X=10 THEN 500 (If X equals 10, jumps to line 500 and executes it.)
or,
IF X=10 GOTO 500
when there's THEN after the GOTO sentence, you can omit THEN.
You can enter other IF sentences within an IF sentence.



Sample Program

```
10 REM * IF - THEN *
20 PRINT "PUSH Y!";
30 A$=INKEY$(0)
40 IF A$<>"Y" THEN BEEP:GOTO 30
50 PRINT:PRINT"Y was pressed."
60 PRINT
70 GOTO 20
```

On line 30, the computer awaits the input of one character. Upon pressing one character, it will check whether it was "Y" or not. If it wasn't "Y", it will emit a "BEEP" and go back to line 30 to wait for the input of one character.

If "Y" was pressed, because the logical equation on line 40 (A$<>"Y") will not be established, the command after THEN will not be executed and line 50 will be executed instead.

FOR~TO~STEP NEXT

Working Repeats and executes the process between the FOR-NEXT loop.

Grammar

```
FOR i=ℓ TO m (STEPs)
NEXT
```

i → Loop variable
ℓ → Start value
m → End value
s → Incremental, negative numbers also O.K. Default value is 1. If you omit STEPs, executed as STEP1.

} Integers from -32768 to +32767

Abbreviation F.-TO-ST.
N.

Explanation In a FOR-NEXT loop, FOR sentences show the beginning of the loop, NEXT sentences show the end of the loop.

For example:

Program flow

```
FOR I=1 TO 10 STEP2
  [Process]
NEXT
```

FOR-NEXT loops have the following behavior. If the flow of the program comes in the FOR sentence, the loop variable I value becomes 1 and enters the first process. When the first process finishes and finds NEXT, the value of I increases by 2 (see STEP2), I becomes the next value 3, goes back to FOR and enters the second process. Like this, when the value of I exceeds 3, 5, 7, 9, 11 and the end value 10, the loop ends and moves to the next line. (Will end repeating the process when the value of I is equal to the end value or the value right before.) However, when omitting STEPs, the incremental value becomes 1.
When there's no matching FOR sentence for the NEXT sentence, "NF ERROR" message will appear.
When the conditions of the loop variable i start value ℓ and end value m are met, it will go to the NEXT sentence.

```
FOR I=1 TO 10
  FOR J=1 TO 20
    FOR K=1 TO 5
    NEXT
  NEXT
NEXT
```

You can not add a loop variable name after NEXT. (An error will occur)

Wrong
```
NEXT K
NEXT J
NEXT I
```

Sample Program

```
10 REM * FOR-NEXT (1) *
20 FOR I=0 TO 10 STEP 2
30 PRINT I;
40 NEXT
RUN
 0 2 4 6 8 10
OK
■
```

```
10 REM * FOR - NEXT (2) *
20 CLS
30 FOR I=1 TO 20
40 FOR J=1 TO I
50 FOR K=1 TO J
60 PRINT "X";
70 NEXT
80 PRINT
90 NEXT
100 PRINT
110 NEXT
```

ON

Working Jumps to the specified line according to the value of the equation.

Grammar

```
ON equation { GOTO   } line number[ , line number, line number,
            { GOSUB  }                          ......]
            { RETURN }
            { RESTORE }
```

Equation->numerical value variable where the value of the integer starts with 1

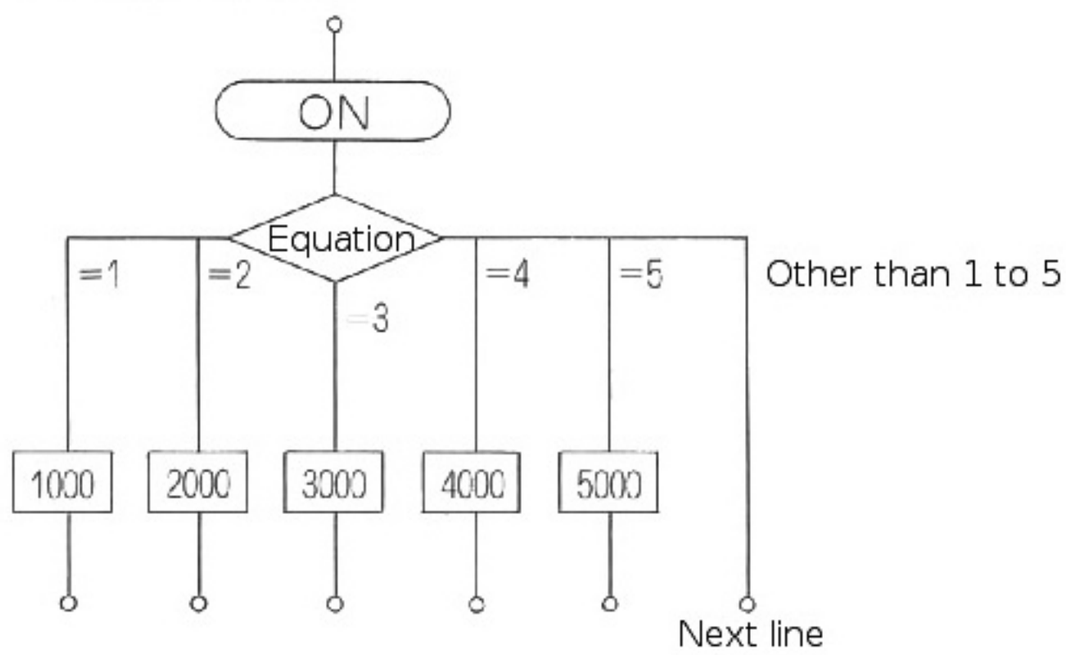
Line->0 to 65534

Abbreviation O.

Explanation When the value of the equation is 1, it jumps to the first line of the range of lines which is written after { }.
If the equation value is 1, first line, if 2, second line, if 3, third line, ... etc. When the value of the equation is 0 or when it exceeds the specified number of lines, it moves to the next sentence after the ON sentence.

```
IF X=1 THEN 1000
IF X=2 THEN 2000
IF X=3 THEN 3000
IF X=4 THEN 4000
IF X=5 THEN 5000
        ↓
ON X GOTO 1000,2000,3000,
          4000,5000
```

〈Flowchart〉



☞ Please refer to GOTO, GOSUB, RETURN and RESTORE

Sample Program

```
10 REM * ON-GOSUB *
20 INPUT " CHOOSE A NUMBER FROM 1 TO 6. ";N
30 ON N GOSUB 100,200,300,400,500,600
40 IF N<1 OR N>6 THEN 20
50 PRINT N;" IS THE SYMBOL OF ";X$;"."
60 GOTO 20
100 X$="ETERNITY":RETURN
200 X$="HOPE":RETURN
300 X$="WOMAN":RETURN
400 X$="MAN":RETURN
500 X$="PERFECTION":RETURN
600 X$="WEDDING":RETURN
```

The numerical value selected on line 20 will be interpreted on line 30. Then it jumps to the matching line numbers (100 to 600), each character variable will be given and will be displayed on line 50. If a number other than 1 to 6 was entered, it will return to line 20 and ask for a number once more.

STOP

Working Stops the program execution.

Grammar STOP

Abbreviation STO.

Explanation When entering a STOP sentence inside a program, "BREAK IN line number" OK. appears on screen and stops. When this happens, the line number shows the line which contains STOP.
Programs stopped by a STOP sentence, because the content of the variables is not cleared and you can resume from the next sentence using a CONT sentence (refer to CONT), it is considered as the best statement to debug programs. When pressing the [STOP] key while executing a program, this will stop a program in the same way and show the stopped line number.

☞ Please refer to CONT, RUN

Sample Program

```
10 REM * STOP *
20 FOR I=1 TO 100
30 PRINT I
40 STOP
50 NEXT
RUN
 1
BREAK IN 40          ....Executes STOP on line 40.
OK.
CONT                 ....Refer to CONT command.
 2
BREAK IN 40          ....Executes STOP on line 40.
OK.
■
```

Program

END

Working Declares the end of the execution of a program.

Grammar END

Abbreviation E.

Explanation END ends the execution of a program.
When executing this statement, it ends the program and goes into a command awaiting status.
When you want to end a program with END, you can put it anywhere. You can also omit END at the end of a program.

Sample Program

```
10 REM * END *
20 FOR I=1 TO 1000
30 PRINT I;
40 A$=INKEY$
50 IF A$="Z" THEN END
60 NEXT
RUN
 1 2 3 4 5 6 7 8
OK
■
```

When you press the Z key, the program execution ends. (this is an example where the Z key was pressed when the numbers up to 8 had been printed.)

Receives the key input on line 40, then interprets the key inpu on line 50. Upon pressing the Z key, the program execution ends, but in other cases, it reads and displays the numerical values up to 1000. (However, it still receives the STOP key press as usual.)

SWAP

Working Swaps the contents of 2 variables.

Grammar SWAP variable(1), variable(2)

Variable->content swapping variable (the types have to be the same)

Abbreviation SW.

Explanation A swap sentence swaps the contents between 2 variables.
Example:
SWAP A,B
in this case, the contents of variables A and B are swapped.
However, the types of the contents must match each other.
(You can not use SWAP A, B$)

Sample Program

```
10 REM * SWAP *
20 DIM A(10)
30 FOR I=1 TO 10
40 READ A(I)
50 PRINT A(I);
60 NEXT
70 FOR I=1 TO 10
80 FOR J=1 TO 10
90 IF A(I)<A(J) THEN SWAP A(I),A(J)
100 NEXT
110 NEXT
120 PRINT
130 FOR I=1 TO 10
140 PRINT A(I);
150 NEXT
160 DATA 2,3,5,1,7,4,8,9,6,0
RUN
 2  3  5  1  7  4  8  9  6  0
 0  1  2  3  4  5  6  7  8  9
OK
■
```

Reads the data inside the array variables and rearranges the order from smallest to biggest value, becoming a base for sorting.

REM

Working Inserts comments within a program sentence.

Grammar REM [Comment]

Comment->optional character string (message) up to 255 characters
Please use " ' " (apostrophe) when using it in the subroutine of a GOSUB sentence
You can not use "REM"

Abbreviation ' (apostrophe)

Explanation Statement to insert a comment in a program and which has no effect on the execution of a program.
You can write it anywhere within a program.

You can use ' (apostrophe) instead of REM.

Program lines written inside REM sentences will not be executed at all. (They do take up program memory though.)

Sample Program

```
10 ' * REM *
20 REM
30 REM NS-HUBASIC V1.0
40 '   SAMPLE PROGRAM
50 '
60 REM   「REM」=「'」
70 REM
RUN
OK
■
```

READ

Working Inputs the data prepared with DATA into a variable in a READ sentence.

Grammar READ variable [,variable, variable , ·····]

Variable->variable which takes data from a DATA sentence

Abbreviation REA.

Explanation Besides INPUT sentences (refer to INPUT) you can also use READ sentences to input data. INPUT sentences enter data during the execution of the program but when using certain data during the execution it can be unhandy, therefore it is better to use READ sentences in this kind of case.
READ sentences are statements which always go together with DATA sentences (refer to DATA) and you should write a variable after READ as well as the matching constant data after DATA.
Because the READ sentence variable and DATA sentence constant data have to be both 1 on 1, they both have to be of the same type.
The DATA sentence can be anywhere inside the program and because with one READ sentence it can read data from 2 or more DATA sentences, you can share and read one DATA sentence from several READ sentences. In any case, DATA sentence line numbers are read from the smallest and data is read from the beginning.
In case the amount of data from the DATA sentences is greater than the amount of variables from the READ sentences, it continues reading from the next READ sentence, if there is no READ sentence, it ignores the rest of the data. In contrary, if the amount of data from the DATA sentences is lacking, the "OD ERROR" message appears.
When using RESTORE (refer to RESTORE), it can read the same DATA sentence as many times as necessary and you can change the line number of the read DATA sentence.

☞Please refer to DATA and RESTORE

Sample Program

(1)Numerical value data reading

```
10 REM * READ --- (1) *
20 FOR I=1 TO 10
30 READ X
40 PRINT X;
50 NEXT
60 DATA 3,4,1,6,2,7,8,3,4,9
RUN
 3  4  1  6  2  7  8  3  4  9
Ok
■
```

(2)Character data reading

```
10 REM * READ --- (2) *
20 READ A$,B$,C$
30 PRINT A$;" ";B$;"."
40 PRINT A$;" ";C$;"."
50 DATA GOOD,MORNING,EVENING
RUN
GOOD MORNING.
GOOD EVENING.
Ok
■
```

(3)Array reading

```
10 REM * READ --- (3) *
20 DIM A(5)
30 FOR I=0 TO 5
40 READ A(I)
50 PRINT A(I);
60 NEXT
70 DATA 9,1,8,3,4,8
RUN
 9  1  8  3  4  8
Ok
■
```

DATA

Working Prepares the data to be read by READ.

Grammar DATA constant[,constant,constant, ·····]

Constant->numerical constant (-32768 to +32767), character constant (character string) type of data

Abbreviation D.

Explanation A DATA sentence is a statement which prepares the data to be read by READ sentences (refer to READ). DATA sentences accept numerical constant or character constant data of up to 255 charaters per line.
Because this statement does not execute anything, you can put as many as you want anywhere within the program, but in general it is best to put it right below a READ sentence or all together at the end of a program in order to make it easy to read a program.
There is no need to surround character constant data (character strings) with double quotes. However, because "," (comma) and ":" (colon) are used as separators, when using "," and ":" as data, you must surround them with " " ".

☞ Please refer to READ and RESTORE.

Sample Program

☞Refer to READ

```
DATA ABC, DE, ",", F
     ↓    ↓   ↓   ↓
     ABC  DE  ,   F
```

RESTORE

Working Specifies the DATA sentences read with READ sentences.

Grammar RESTORE [line number]

Line number-> DATA sentence from where it starts reading

Abbreviation RES.

Explanation The BASIC interpreter has pointers which point to DATA and when executing READ sentences, it starts looking for DATA sentences at the beginning of that program and sets a pointer at the DATA that appears first. When executing RESTORE, the pointers for DATA get set on the line of the specified line number. Only when using RESTORE the pointer is set at the beginning of the program.
When using RESTORE sentences, you can use the same DATA sentence as many times as you wish and you can specify the read DATA sentence as optional.

☞Please refer to READ, DATA

Sample Program

```
10 REM * RESTORE *
20 RESTORE 1010
30 FOR I=0 TO 5
40 READ A
50 PRINT A;
60 NEXT
70 PRINT
80 RESTORE 1000
90 FOR I=0 TO 5
100 READ A
110 PRINT A;
120 NEXT
130 REM
1000 DATA 23,43,55,65,42,9
1010 DATA 12,56,34,68,53,2
RUN
 12  56  34  68  53  2
 23  43  55  65  42  9
OK
■
```

CALL

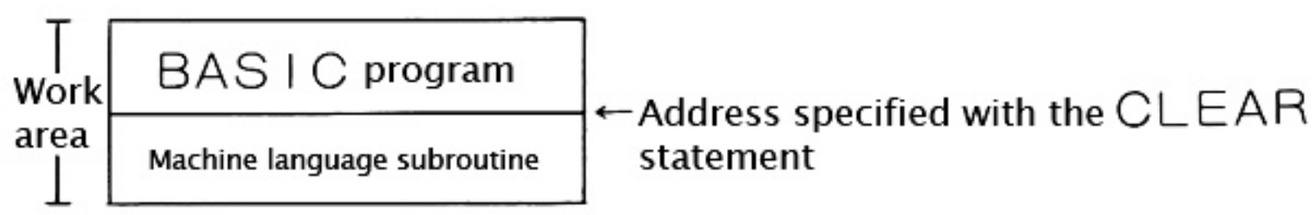
Working Calls machine language subroutines directly.

Grammar CALL Address

Address → execution start address of the subroutine.

Abbreviation CA.

Explanation Calls machine language subroutines using the specified address as the execution start address. The address is specified with a hexadecimal literal (in this form &HOOOO where O are numbers), a decimal integer literal (between -32768~+32767) or an integer variable expression. Please place the machine language subroutine after the address specified with the CLEAR statement.



POKE

Working Writes 1 byte of data in the memory. (⇔PEEK)

Grammar POKE address, data [, data, data, ...]

Address->address where to write data to (please refer to the memory map on p. 104)

Data->integer from 0 to 255

Abbreviation PO.

Explanation Writes 1 byte (8 bits) of data directly into the specified address of the memory. The data must be a value from 0 to 255 (&H0 to HFF). When writing data continuously separated by "," (comma), you can write all the addresses continuously after the specified address. However, since POKE rewrites the content of the current memory, because a careless usage destroys the system region of family basic, you must be careful when using it.
The memory scope which you can use with POKE goes from &H7040 to &H77FF. Because &H7000 to &H703F is used by the system, you should not use it.

Sample Program

```
10 REM * POKE *
20 CLEAR &H7600
30 D=0
40 FOR A=&H7600 TO &H761F
50 POKE A,D
60 D=D+1
70 NEXT
80 FOR A=&H7600 TO &H761F
90 RD=PEEK(A)
100 PRINT "  ";HEX$(RD);
110 NEXT
```

The loop from line 40 to 70 writes the numerical values from 0 to 31 in the memory addresses from 7600 to 761F.
The loop from 80 to 110, in order to check whether data has been actually entered or not in each address, uses the PEEK function (p. 85), reads the data and shows it in hexadecimal.

Screen Control Statements

LOCATE

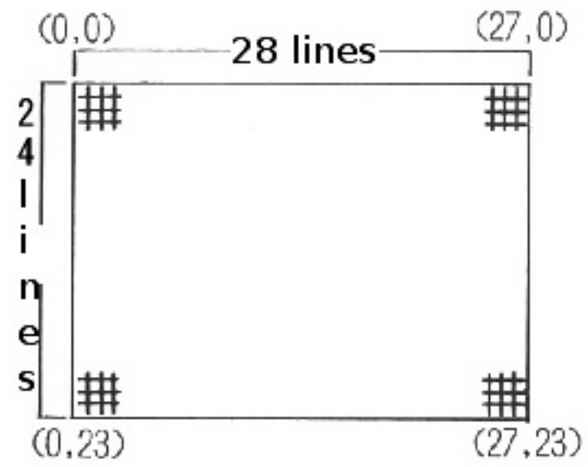
Working Moves the cursor to the specified position.

Grammar LOCATE X, Y

X……Horizontal display column 0 to 27
Y……Vertical display line 0 to 23

Abbreviation LOC.

Explanation LOCATE specifies the cursor position of the background screen.



Sample Program

```
10 REM * LOCATE *
20 CLS
30 FOR I=0 TO 20
40 LOCATE I,I:PRINT"*";
50 NEXT
60 LOCATE 0,10
```

Line number 40 specifies the position of the cursor and prints *. Line number 60 moves the display position and ends the execution of the program. (It can not receive more commands after the end position. Please move the cursor to a line which has no display of *.)

COLOR

Working Specifies the color pattern number of the characters to display on the background screen per area of the screen.

Grammar COLOR X, Y, n

X…Horizontal display column 0 to 27
Y…Vertical display line 0 to 23
n…Color pattern number 0 to 3

Abbreviation COL.

Explanation Selects display color of the background or character from within the color pattern number inside the color pallet code specified by CGSET for each area on screen which includes the position specified by X, Y.
Please refer to the color chart on p. 113.
BG GRAPHIC is displayed in the color pattern of color pattern number (0 to 3) from color pallet 1. Even if you change the color pallet code with CGSET, the color pattern number specified within the area by the COLOR sentence remains active.

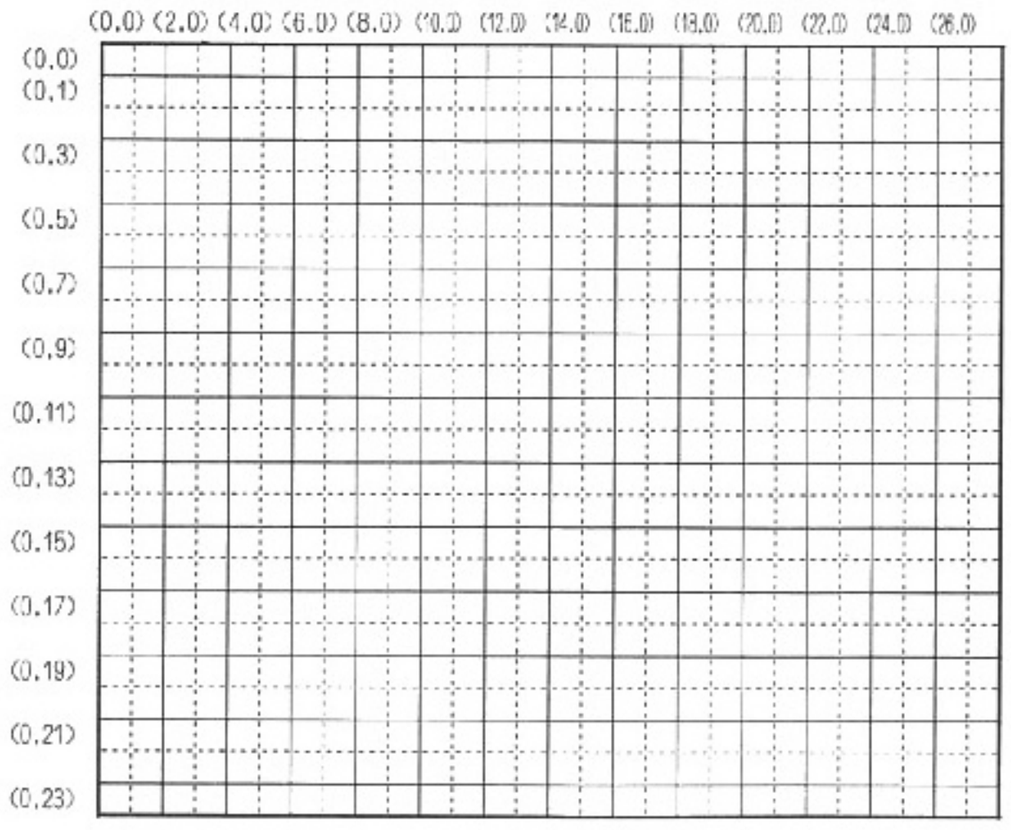
Sample Program

☞ Please refer to CGSET

```
10 REM * COLOR *
20 CLS
30 FOR I=0 TO 447
40 PRINT CHR$(195);
50 NEXT
60 FOR C=0 TO 3
70 COLOR 5+C*3,5+C*2,C
80 NEXT
90 LOCATE 0,20
```

Displays the area which includes the position specified on line number 70 in color pattern according to the specified color pattern number. On line number 90 it moves the cursor to the specified position and ends the execution of the program.

The area for which you can specify colors.



COLOR can specify the color pattern number of each area of the scope shown by ⊞.

	Column 10	Column 11
9th line	A	B
10th line	C	D

For example, if you pick
COLOR 10, 10, 3
the 4 characters (letters, symbols, graphical characters) within the area (A, B, C and D) on screen which included the (10, 10) specified by X, Y will be displayed in the colors of color pattern number 3.

CGEN

Working Decides the allocation of the sprites on the background screen and the sprite screen.

Grammar CGEN n

n···Allocation combination 0 to 3

Abbreviation CGE .

Explanation Selects which characters from character table A (back cover) and B (p. 113) such as Mario's animated characters (sprites), numbers, letters, symbols, kana, background pattern characters (BG GRAPHIC) to use on either the background screen or the sprite screen. This allows you to display Mario and other animated characters on the background screen, as well as display character symbols as sprites on the sprite screen.

Character table A	Character table B
Mario and other animated characters	Background patterns including alphanumerics, symbols and kana

Please refer to the character code list from p. 106 to 109,

n	Background screen	Sprite screen	Meaning
0	A	A	Uses the characters from character table A on both background and sprite screens.
1	A	B	Uses char. from table A in BG screen, char. from table B in sprite screen.
2	B	A	Uses char. from table B in BG screen, char. from table A in sprite screen.
3	B	B	Uses the characters from character table B on both background and sprite screens.

Default value···CGEN2

CTR+D key resets CGEN to value 2.

Sample Program

```
10 REM * CGEN *
20 CLS:SPRITE ON:CGSET 0,1
30 FOR I=32 TO 255
40 PRINT CHR$(I);
50 NEXT
60 DEF SPRITE 0,(0,1,0,0,0)=CHR$(64)
   +CHR$(65)+CHR$(66)+CHR$(67)
70 SPRITE 0,100,150
80 PAUSE 100:BEEP
90 CGEN 0
100 PAUSE 100:BEEP
110 CGEN 1
120 PAUSE 100:BEEP
130 CGEN 3
140 PAUSE 100:BEEP
150 CGEN 2
```

Displays a preview of character symbols on screen.

Defines Achilles (left 1).

line number 10—80 ··· Displayed by CGEN2.
line number 90—100 ··· Displayed by CGEN0.
line number 110—120 ··· Displayed by CGEN1.
line number 130—140 ··· Displayed by CGEN3.
line number 150 ··· Returns to display by CGEN2.

CLS

Working Clears the screen.

Grammar CLS

Abbreviation CL

Explanation Clears the background screen.
BG GRAPHIC copied to the background screen will disappear at the same time. Use the VIEW command instead of the CLS command when copying BG GRAPHIC to the background screen in a program.

Sample Program

☞Please refer to LOCATE

CLS RETURN·····Screen gets cleared, OK appears on upper left of the screen.

```
OK
■
```

10 CLS·····After clearing the screen, it executes the program of the next line number.

CGSET

Working Selects the allocation of the palet used for BG and sprites. (specifies the color combination to display)

Grammar CGSET [m] [, n]

m…Palet code 0 or 1 for the background screen

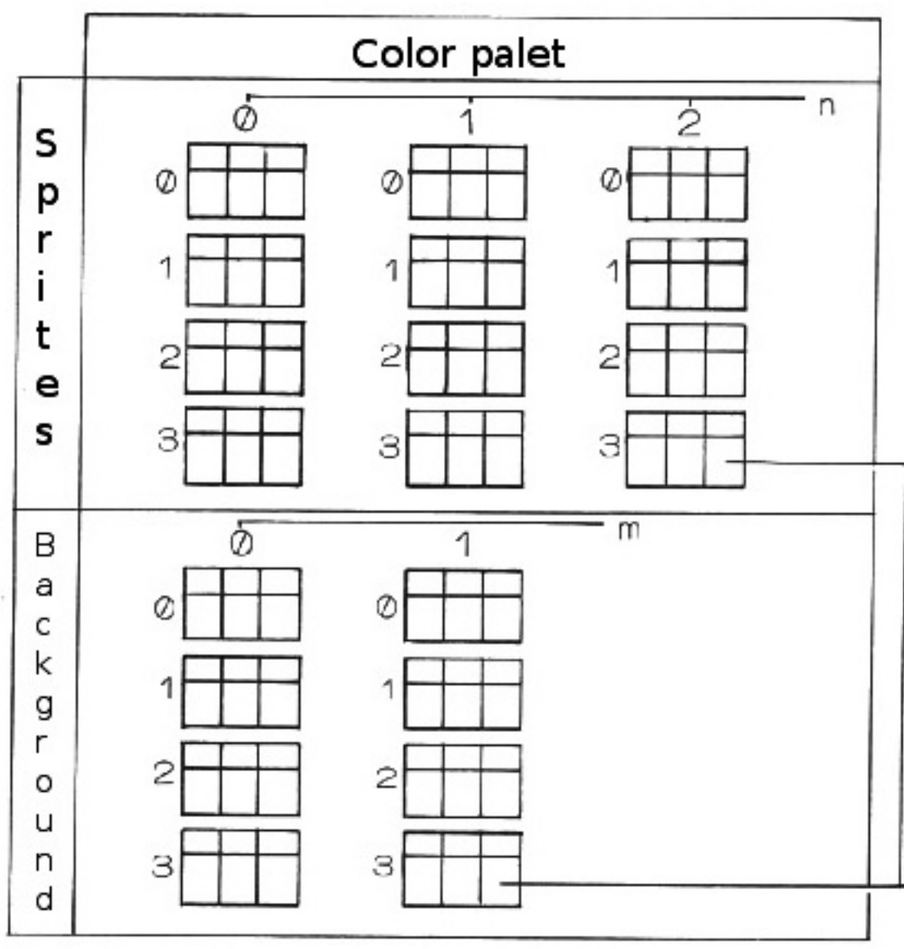
n…Palet code 0 to 2 for sprites (animated characters)

Abbreviation CG .

Explanation Selects amond the group of combined colors prepared in advance and chooses a display color for the animated characters and backdrop. All of the color palet colors (color codes) have been set in order to display color charts (p. 113) for background and sprites among all of the 52 displayable colors. There are 2 types of background color palets (palet code 0 and 1) and 3 types of sprite color palets (palet code 0 and 2). You can maintain a total of 12 color codes, 3 types per color group of color combination number (0 to 3) for each color palet.
Once defined, it remains valid until you redefine it or until you use the PALET command to change the color code of the color combination.
(The backdrop screen color is black (see-through).)

Sample Program

```
CGSET 0,0
OK
■
10 CGSET 0,1
```



All of the color codes for the 52 colors, allocated to each palet.

The color combination number, specified by the DEF SPRITE and DEF MOVE sentences, specifies the color of the sprite to be displayed by specifying the combination of the color of the color combination number (0 to 3) within the palet selected by CGSET.
For example:

```
CGSET 1, 1   specification of palet code for sprite
DEF SPRITE 0, (0, 1, 0, 0, 0) =…
DEF MOVE (N) =SPRITE (0, 3, 1, 10, 0,
3)
```

Specifies the color combination number within palet number 1 for a sprite specified by CGSET above. (0 and 3)

Sprites are displayed in this color combination.
The background screen color combination specification is done through the COLR sentence.
Please refer to p. 70.

The color codes matching every color within all of the displayable 52 colors. (Please refer to the PALET command.)

Please use the background screen palet code 1 (CGSET 1,1) to copy the drawing from the BG GRAPHIC screen with the VIEW command on the background screen with the same colors .

Default value…m=1、n=1

Please refer to the color chart (CGSET) on p. 113.
When using CTR+D or selecting 1--BASIC from the GAME BASIC mode screen, when you have reached the BASIC screen, the color palet becomes the background palet code 1 for background and sprites.

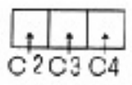
PALET

Working Resets the color code within a color combination number to an arbitrary color code.

Grammar PALET {B/S} n , C1 , C2 , C3 , C4

B···BG (Background)
S···Sprite (animated character)
n···Color combination number 0 to 3
C1···Defines the color of the background screen.
Becomes valid when the value of n is 0.

C1, C2, C3, C4 ···Defines the color codes of the 52 colors.
C2, C3 and C4 match every color the left edge, center and right edge of the color combination number.
Example:
Color combination number 0



Each code uses a value (0 to 60) from the list below.

Abbreviation PAL.B
PAL.S

Explanation Picks a color code within the 52 colors and colorizes the background and the sprites. The backdrop color colorizes the screen even if you use that color also for the background and for the sprites. (Disregarding PALETB and PALETS) within a program, the color defined in the greatest execution line number program will eventually be used to display.
It is displayed according to the palet chosen with a CGSET sentence. On top of this, it chooses the n value matching the color group within that palet and sets the 4 color (C1, C2, C3 and C4) code. The color code can select among the 52 colors produced by the color generator.

Sample Program

```
10 REM * PALET *
20 SPRITE ON
30 DEF SPRITE 0,(3,1,0,0,0)=CHR$(88)
   +CHR$(89)+CHR$(90)+CHR$(91)
40 SPRITE 0,100,150
RUN
OK
CGSET 1,0
OK
PALETS 3,&H0F,&H30,&H26,&H12
OK
■
```

OK ···Displays with the colors of color combination number 3 through sprite palet code 1.
OK ···Displays with the colors of color combination number 3 through sprite palet code 0.
PALETS ···Dislays with colors &H30, &H26 and &H12.

《52 color codes》

	Hex	DEC	Hex	DEC	Hex	DEC	Hex	DEC	
	00	0	10	16	20	32	30	48	Gray ~ White
BLUE	01	1	11	17	21	33	31	49	
	02	2	12	18	22	34	32	50	
	03	3	13	19	23	35	33	51	
	04	4	14	20	24	36	34	52	
RED	05	5	15	21	25	37	35	53	Colored
	06	6	16	22	26	38	36	54	
	07	7	17	23	27	39	37	55	
	08	8	18	24	28	40	38	56	
GREEN	09	9	19	25	29	41	39	57	
	0A	10	1A	26	2A	42	3A	58	
	0B	11	1B	27	2B	43	3B	59	
	0C	12	1C	28	2C	44	3C	60	
	0D	13	1D	29	2D	45			Black
	0E	14	1E	30	2E	46			
	0F	15	1F	31	2F	47			

HUE ↕ Dark ⟷ Bright

```
CGSET 1,0
DEF SPRITE 0,(0,1,0,0,0)="@ABC"
```

CGSET 1,0 — Defines sprite palet code 0.
DEF SPRITE 0,(0,1,0,0,0)="@ABC" — Defines color combination number 0 among palet code 0.

The sprite will be shown this time with a color combination of color codes 54, 22 and 2.

```
PALETS 0,1,48,25,18
```

PALETS 0,1,48,25,18
- 1 — Change backdrop color to 1.
- 48 — Change color 54 to 48.
- 25 — Change color 22 to 25.
- 18 — Change color 2 to 18.

MOVE command

Cluster of commands to display and move animated characters over the sprite screen. They are similar to DEF SPRITE and SPRITE n, x, y, but allow for a much a easier execution of game expressions.

DEF MOVE

Working Defines a specific movement for an animated character.
(There is a total of 16 types of animated characters which you can use.)

Grammar DEF MOVE(n)=SPRITE(A,B,C,D,E,F)

n…Animated character movement number 0 to 7

A…Animated character type 0 to 15

B…Appointment of the movement direction 0 to 8

C…Speed of movement 1 to 255 (in case of 0, it will move every 256 frames) 1…fastest 255… slowest

D…Complete movement distance 1 to 255 (0 doesn't display anything)

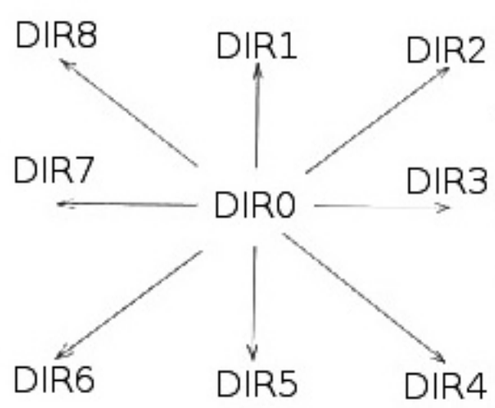
E…Display priority 0 to 1 (0…displays on the sprite screen in front of the background)
(1…displays on the sprite screen behind the background)

F…Color combination number 0 to 3

Abbreviation DE.M.

Explanation Select up to 8 animated characters among the 16 types and define their movement.
-The animated character movement number is the number after = defined for the movement of a specific character.
-The type of animated character is allocated among the following animated characters. (Please refer to character table A on the back cover.)

0 : Mario	9 : Starship
1 : Lady	10 : Explosion
2 : Fighter Fly	11 : Smiley
3 : Achilles	12 : Laser
4 : Penguin	13 : Shell Creeper
5 : Fireball	14 : Side Stepper
6 : Car	15 : Nitpicker
7 : Spinner	
8 : Star Killer	



The movement directions match the eight directions clockwise. When specifying 0, the character does not move.

-The speed of movement is displayed as a movement of 2 dots (in case of diagonal movement, 2 dots for both X and Y) per amount of frames (about 1/30 second) which is the double of the specified speed of movement (C), and the speed (1 dot/second)=2dots/(CX1/30 second) = 60/C (1 dot/second).
For example, when you specify 1, the speed moves 60 dots per second, in case of 255, the speed moves 60 dots per 255 seconds.
-The complete movement distance specifies the distance of movement of animated characters. The animated characters move only the double of the specified amount of dots. The complete movement amount of dots = 2 x D (dots) (in case of diagonal movement, 2 dots for both X and Y).
-The display priority specifies whether an animated character is displayed on the sprite screen in front of the background screen (0) or on the sprite screen behind the background screen (1).
-The color combination code specifies the color combination to display with the color combination number (0 to 3) specified within the palet code by CGSET.

MOVE

Working Starts the movement of the animated character.

Grammar MOVE n_0 [, n_1 , n_2 , n_3 , n_4 , n_5 , n_6 , n_7]

n_0～n_7 …Action number defined by DEF MOVE 0 to 7

Abbreviation M .

Explanation Starts the movement of the animated characters defined previously by the DEF MOVE command. (Please use SPRITE ON to enable the display of sprites.)
You can start the movement of up to 8 animated characters simultaneously.
Upon executing the MOVE sentence, until the movement specified by the DEF MOVE command ends, the animated characters move asynchronuously of the BASIC controls. In other words, even if the program ends, the animated characters move until the movement's
distance, specified direction and according speed have ended, as defined in the DEF MOVE sentence.
Up to 4 can be displayed on the same horizontal direction on the sprite screen.

CUT

Working Stops the movement of animated characters started in a MOVE sentence.

Grammar CUT n_0 [, n_1 , n_2 , n_3 , n_4 , n_5 , n_6 , n_7]

n_0～n_7 …Action number defined by DEF MOVE 0 to 7

Abbreviation CU .

Explanation Stops the movement of a character number chosen at will for animated characters for which the movement was started by a MOVE sentence. You can specify up to 8 characters simultaneously.
After executing CUT, when restarting the movement of animated characters of the same action numbers, they start
moving from the position where they stopped until they complete the rest of the movement distance of the total movement distance (D) defined beforehand by DEF MOVE.

Sample Program

```
CUT 0,1,2
OK
■
10 CUT 0,1,2,3,4,5,6,7
```

ERA

Working Stops the movement of animated characters started with a MOVE sentence and erases them from the sprite screen.

Grammar ERA n_0 [, n_1 , n_2 , n_3 , n_4 , n_5 , n_6 , n_7]

n_0～n_7 …Action number defined by DEF MOVE 0 to 7

Abbreviation ER .

Explanation Erases at will the dispay of animated characters of which the movement was started by a MOVE sentence or stopped by a CUT sentence (or completed their movement).
After executing ERA, when restarting the movement of animated characters of the same action numbers, they appear at the position where they disappeared and start moving until they complete the rest of the movement distance of the total movement distance (D) defined beforehand by DEF MOVE.

Sample Program

```
ERA 0,1,2
OK
■
10 ERA 0,1,2,3,4,5,6,7
```

POSITION

Working Assigns the initial coordinates before starting the movement with the MOVE sentence. (Sprite screen coordinates.)

Grammar POSITION n, X, Y

n…Action number defined by DEF MOVE 0 to 7

X…Horizontal coordinate 0 to 255

Y…Vertical coordinate 0 to 255

(However, the available range on the sprite screen is X: 0 to 240 and Y: 5 to 220.)

(The available display range on the screen might differ depending on the TV set.)

Abbreviation POS.

Explanation Assigns the initial coordinates at which to start the action before starting the movement of the animated characters with the MOVE sentence. The values of X and Y are the coordinates on the sprite screen which are located on the upper left corner of the animated characters.
When not specified, the movement starts at the default values X = 120 and Y = 120.
When you start the movement of animated characters of the same action number within a program repeatedly with MOVE, you can start the position of the finished movement defined by DEF MOVE as the initial coordinates of the next movement.
With RUN or with another POSITION command you can respecify the initial coordinates of the movement, or, until you redefine the movement with DEF MOVE, the animated characters with the same action number keep the coordination position of the finished movement from before.

Sample Program

```
10 REM * POSITION *
20 CLS:SPRITE ON
30 DEF MOVE(0)=SPRITE(11,3,2,10,1,2)
40 X=RND(256):Y=RND(240)
50 PRINT"X,Y=";X;",";Y
60 POSITION 0,X,Y
70 MOVE 0
80 PAUSE 80
90 GOTO 20
```

Line number 30…Defines that Smiley moves 20 dots to the right.
Line number 40…Generates X and Y with random numbers.
Line number 50…Displays the value of X and Y.
Line number 60…Specifies the start position of Smiley's movement.
Line number 70…Starts Smiley's movement.

XPOS

Working Invokes the horizontal coordinate value of the position of the animated character of the action number defined by DEF MOVE.

Grammar XPOS(n)

n…Action number of the animated character defined by DEF MOVE 0 to 7

Abbreviation XP.

Explanation When an XPOS function is executed, it invokes the value of the horizontal coordinate of the character of the action number specified by n.
During a sequential transition of the action, use this together with the POSITION sentence.

Sample Program

```
10 REM * XPOS  YPOS *
20 CLS:SPRITE ON
30 DEF MOVE(0)=SPRITE(0,2,2,10,0,0)
40 MOVE 0
50 LOCATE 8,20:PRINT"      "
60 LOCATE 8,21:PRINT"      "
70 LOCATE 0,20:PRINT"XPOS(0)=";XPOS(0)
80 LOCATE 0,21:PRINT"YPOS(0)=";YPOS(0)
90 PAUSE 50
100 GOTO 40
```

Line number 30…Defines that Mario moves X and Y 20 dots each to the upper right.
Line number 70-80…Displays the value of Mario's XPOS and YPOS.

YPOS

Working Invokes the vertical coordinate value of the position of the animated character of the action number defined by DEF MOVE.

Grammar YPOS(n)

n…Action number of the animated character defined by DEF MOVE 0 to 7

Abbreviation YP.

Explanation When a YPOS function is executed, it invokes the value of the vertical coordinate of the character of the action number specified by n.
During a sequential transition of the action, use this together with the POSITION sentence.

Sample Program

☞ Refer to XPOS

MOVE (n)

Working Provides whether or not the animated characters, of the action numbers which started their movement through the MOVE sentence, have finished their movement defined by DEF MOVE via the value of the function.

Grammar MOVE (n)

n … Action number of the animated characters defined by DEF MOVE 0 to 7

Abbreviation M. (n)

Explanation The value granted by this function may differ depending on the movement of the animated character of which the movement was started by the MOVE sentence.
The MOVE(n) function grants value -1 when the movement defined by DEF MOVE is still being executed, and 0 when the movement is finished, to the animated characters for which the movement has been specified by DEF MOVE.

Sample Program

```
10 REM * MOVE(N) *
20 SPRITE ON :CGSET 1,0
30 DEF MOVE(0)=SPRITE(0,3,1,150,0,0)
40 MOVE 0
50 IF MOVE(0)=-1 THEN PRINT"MOVE(0)=
   ";MOVE(0):GOTO 50
60 PRINT"MOVE(0)=";MOVE(0)
70 END
```

Line number 30…Specifies Mario to move 300 dots to the right.
Line number 50…Distinguishes the value of MOVE(0) and decides whether Mario has finished moving or not.
Line number 60…Displays MOVE(0)=0 when Mario has finished moving.

KEY

Working Defines character strings according to function keys.

Grammar KEY Function key number, character string

Function key number ····· 1 to 8 (matches the 1, 2, 3, 4, 5, 6, 7 and 8 function keys from the left corner)

Character string ····· You can define up to 15 characters and all the characters above 15 to the right are ignored.

Abbreviation K.

Explanation Makes a character string responsive to the key specified by a function number. When defining a character string with this statement, from then on, every time you will press that key it will be the same as if you would enter the defined character string. The function keys are defined as follows when selecting the BASIC screen from the GAME BASIC mode screen.

F1. LOAD(M)
F2. PRINT
F3. GOTO
F4. CHR$(
F5. SPRITE
F6. CONT(M)
F7. LIST(M)
F8. RUN(M)

(M) stands for RETURN. Please enter +CHR$(13) behind a character string in order to define (M).

☞ Please refer to KEY LIST

Sample Program

```
KEYLIST                    ···Displays the contents of the function keys.
F1 LOAD(M)                 ···(M) stands for RETURN.
F2 PRINT
F3 GOTO
F4 CHR$(
F5 SPRITE
F6 CONT(M)
F7 LIST(M)
F8 RUN(M)
OK
KEY1,"SAVE"+CHR$(13)       ···Defines SAVE RETURN on F1.
                               CHR$(13) means RETURN
OK
KEY5,"DEF MOVE("           ···Defines DEF MOVE ( on F5.
OK
KEYLIST
F1 SAVE(M)
F2 PRINT
F3 GOTO                    ···Defined in F1 and F5
F4 CHR$(
F5 DEF MOVE(
F6 CONT(M)
F7 LIST(M)
F8 RUN(M)
OK
■
```

Default value

KEYLIST

Working Displays the definition status of the function keys.

Grammar KEY LIST

Abbreviation K.L.

Explanation When executing this statement, a list of the function key numbers and character strings will be displayed on screen.

☞ Please refer to KEY

PAUSE

Working Halts temporarily the execution of the program.

Grammar PAUSE [n]

n···0~32767

Abbreviation PA.

Explanation When executing this statement, the program execution will resume after halting during the set unit of time. When omitting n, the program halts until a key is pressed. Upon pressing a key, it moves on to the next program.

Sample Program

```
10 REM * PAUSE *
20 FOR I=0 TO 10
30 BEEP:PAUSE 10
40 NEXT
50 PAUSE                   ···Halts the progression of the program
60 FOR I=0 TO 10               here until a key is pressed.
70 PLAY"C":PAUSE 20
80 NEXT
```

SYSTEM

Working Returns from the BASIC execution mode to the GAME BASIC mode screen.

Grammar SYSTEM

Abbreviation S .

Explanation Switches to the GAME BASIC screen when entering a SYSTEM command on the BASIC screen. Do not use it within a program. The program remains protected.
1--Select BASIC and upon entering the BASIC screen again, you can continue using BASIC.

Sample Program

```
SYSTEM        ···Direct command
```

VIEW

Working Copies (duplicates) the BG GRAPHIC screen to the background screen.

Grammar VIEW

Abbreviation V .

Explanation Upon entering the VIEW command while executing BASIC, the picture drawn on the BG GRAPHIC screen gets copied (duplicated) onto the background screen. You can use BASIC to redraw the BG GRAPHIC copied onto the background screen, however that picture remains on the BG GRAPHIC screen.
Please use 1 (CGSET 1, 1) from the palet code of the background screen to copy and display the picture drawn on the BG GRAPHIC screen with its colors onto the background screen.

Sample Program

```
VIEW          ···Direct command
                 (The cursor returns to the home position)

10 VIEW       ···Within the program
```

Sound Control Statements

BEEP

Working Outputs a "beep" type of sound.

Grammar BEEP

Abbreviation B.

Sample Program

☞ Please refer to PAUSE, IF ~ THEN.

PLAY

Working Plays back music.

Grammar PLAY String data

String data···The string variables explained below are called string data

Abbreviation PL.

Explanation Plays back according to the sound specified by the string data. The string data specifies the interval, octave, length of sound, volume, chord, envelope and duty, and looks like the string of characters that we will explain below.
You can use the channel separator (colon) to play back up to 3 sounds,

1) Tempo...Specifies the tempo with T1~T8.
   T1 (fast) <----------->T8 (slow)
2) Duty effect (tone)...Specifies tone with Y0~Y3.
   Y0: 12.5% -Duty effect becomes effective after
   Y1: 25.0% executing the specified envelope (M0 or M1).
   Y2: 50.0% -After specifying M0 or M1, please specify the
   Y3: 75.0% tone (Y0~Y3).
3) Envelope...Specifies with M0 and M1 whether to use or not the envelope. When using the envelope, the sound becomes resonant, flat or sloppy.
   -When using M0 (no envelope), you can specify the volume with Vn from V0 to V15.

small ⊘ ←volume→ 15 big

M0V15 — Volume 15

-When using M1 (envelop), you can specify the length of the envelope with Vn from V0 to V15.

short ⊘ ← envelope → 15 long

M1V3 — Envelope 3, volume 15.

4) Octave...Specifies the octave with O0 ~ O5.
O0 (low tone)←→O5 (high tone)
5) Interval...The interval is displayed as below by using the characters C~B.

Interval	Designation
Do	C
Do♯ (Re♭)	♯C
Re	D
Re♯ (Mi♭)	♯D
Mi	E
Fa	F
Fa♯ (So♭)	♯F
So	G
So♯ (La♭)	♯G
La	A
La♯ (Si♭)	♯A
Si	B

Sample Program

☞ Please refer to RND.

```
PLAY"CRDRE"
OK
PLAY"T4Y2M0V1503C5R5D5R5E5"
OK
PLAY"T2Y0M1V901C3R5D6R1E4"
OK
PLAY"C:E:G"
OK
PLAY"05C5:04E5:01G5"
OK
■
```

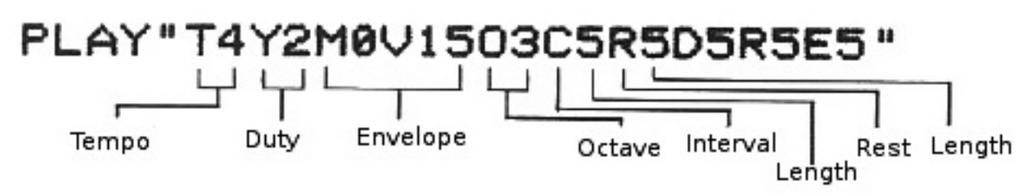
···multiphonic(3 simultaneous sound output)separated by colons

half/high tone low tone

Specifies each parameter for each channel in case of 3 simultaneous sound output.

Once specified, T, Y, M, V and O will carry on their values.

T (Tempo), Y (Duty effect), M (Envelope),
V (Envelope's length or volume), O (Octave)



6) Rest...Specifies the length of the rest from R0 to R9.
7) Length...Specifies the length of sounds or rests by adding integers 0 ~9 after intervals, rests or symbols.

Length of sound		Matching integers
$\frac{1}{8}$ (32nd note 𝅘𝅥𝅯)	𝄿	0
$\frac{1}{4}$ (16th note 𝅘𝅥𝅮)	𝄾	1
$\frac{3}{8}$ (dotted 16th note 𝅘𝅥𝅮.)	𝄾.	2
$\frac{1}{2}$ (8th note ♪)	𝄽	3
$\frac{3}{4}$ (dotted 8th note ♪.)	𝄽.	4
1 (4th note ♩)	𝄼	5
$1\frac{1}{2}$ (dotted 4th note ♩.)	𝄼.	6
2 (2nd note 𝅗𝅥)	𝄻	7
3 (dotted 2nd note 𝅗𝅥.)	𝄻.	8
4 (whole note 𝅝)	𝄺	9

Warning) The length of the note is equal to the relative value of a quarter note (integer 5) considered as 1.
(Written as C5R1)
When no integer is specified, it takes the same length of note as the one written before.

8) Multiphonic...Insert a channel separator : (colon) to play back 2 or 3 sounds at the same time.
"Channel A:Channel B:Channel C"
You have to specify the interval, length of note, tempo, envelope and duty for each channel.
However, envelope and duty do not change for channel C.
Default value...T4, M0, V15, O3, length of sound or length of rest 5.
Once specified, the values for each parameter remain unchanged until they are respecified.

Numeric Functions

ABS

Working Gives the absolute value of a mathematical expression.

Grammar ABS (x)

x ···Mathematical expression which requests an absolute value (integer -32768 to +32767)

Abbreviation AB .

Explanation The absolute value 1 x 1 of the mathematical expression x becomes the value of this function.

Sample Program

```
PRINT ABS(41)
 41
OK
PRINT ABS(-41)
 41
OK
PRINT ABS(10-34)
 24
OK
■
```

SGN

Working Gives the sign of the mathematical expression.

Grammar SGN (x)

x ···Mathematical expression (integer -32768 to +32767)

Abbreviation SG .

Explanation According to the mathematical expression, the following value becomes the value of this function.

x>0 -> 1
x=0 -> 0
x<0 -> -1

Sample Program

```
PRINT SGN(41)
 1                  ·····Positive.
OK
PRINT SGN(0)
 0                  ·····0 .
OK
PRINT SGN(-41)
-1                  ·····Negative.
OK
■
```

RND

Working Generates random numbers (integers) without arguments.

Grammar RND (x)

x ···Mathematical expression (integer) 1 to 32767

Abbreviation RN .

Explanation The generated random integer becomes the value of this function. It becomes a random number of "argument (x) -1".
RND(1) is always equal to 0.

Sample Program

Let's play back music with random numbers through RND

```
10 REM * RND *
20 PLAY"M1V6T201"
30 A$="CDEFGABR"
40 N=RND(8)
50 B$=MID$(A$,N+1,1)
60 PRINT B$;
70 PLAY B$
80 GOTO 40
```

·····Extract 1 character from the A$ character string and call it B$.
The extracted character uses the random value generated on line 40, which is lower than 8 (0 to 7) .
Plays back the sound of the extracted character while displaying that character.

Character Functions

ASC

Working Converts character codes into numerical values. (<=>CHR$)

ASC (character string)

Grammar Character string...Character strings located to the left where characters are converted into numerical values (Only the characters to the left are converted)

Abbreviation AS .

Explanation The character code of the first character of the character string becomes the value of this function. The character code is an integer value from 0 to 255. The character string can also be a formula or a variable. Also, when the character string is a null string, 0 becomes the value of this function.

Sample Program

```
10 REM * ASC *
20 INPUT"CHARACTER";A$ ............ Only converts the
30 A=ASC(A$)                         first character.
40 PRINT A$;"CHARACTER CODE IS";A
RUN
CHARACTER?H                          ..... When entering H
H CHARACTER CODE IS 72
Ok
```

Please refer to the character code table on p. 102 to 105 to check the ASC code comparison table.

CHR$

Working Considers a numerical value as a character code and converts it into the matching character. (<=>ASC)

Grammar CHR$ (x)

x ·····Mathematical expression to convert into characters
All from 0 to 255

Abbreviation CH .

Explanation Yields a character as a character code from a numerical value. You can obtain one numerical value per character. The characters and symbols to be printed are for the X value from 32 to 255.

Sample Program

```
10 REM * CHR$ *
20 INPUT"CHARACTER CODE IS (0-255)";A
30 A$=CHR$(A)
40 PRINT A;"THE MATCHING CHARACTER IS";A$
RUN
CHARACTER CODE IS (0-255)?65        .....When entering 65
 65 THE MATCHING CHARACTER IS A
Ok
```

■

VAL

Working Converts the number character strings into numerical values.

Grammar VAL (character strings)

Character strings...number character strings -32768 to +32767
&H0 to &H7FFF

Abbreviation VA .

Explanation Converts the numbers considered as characters within a character string into numerical values. In case the first character of the character string is not +, -, & or a number, the value of this function becomes 0. Also, if characters which are not numbers appear in the character string (except hexadecimal A to F), the characters appearing after it will be ignored.

Sample Program

```
10 REM * VAL *
20 INPUT"PLEASE ENTER A HEXADECIMAL NUMBER";A$
30 V=VAL("&H"+A$)
40 PRINT A$,"&H";A$;"=";V
RUN
PLEASE ENTER A HEXADECIMAL NUMBER?AD.....When entering AD
AD          &HAD= 173
Ok
```

■

STR$

Working Converts the values of mathematical expressions to number characters.

Grammar STR$ (x)

x ···Mathematical expression

Abbreviation STR .

Explanation Character strings displaying mathematical expressions or numerical values become the value of this function. In case of positive numbers, 1 character space is entered at the beginning.

Sample Program

```
10 REM * STR$ *
20 INPUT"NUMERICAL VALUE OF A IS";A
30 A$=STR$(A)                  ..... Converts the numerical value of
                                     A into a character numerical value
40 INPUT"NUMERICAL VALUE OF B IS";B
50 B$=STR$(B)                  ..... Converts the numerical value of
                                     B into a character numerical value
60 PRINT A;B;A+B               ..... Displays as a numerical value
70 PRINT A$;B$;A$+B$           ..... Displays as a character of a
                                     numerical value
```

HEX$

Working Converts a mathematical expression into a hexadecimal character string.

Grammar HEX$ (x)

x ···Mathematical expression -32768 to +32767

Abbreviation H.

Explanation The result of the mathematical expression turned into a hexadecimal character string becomes the value of this function.

Sample Program

Displays a conversion table of decimals from 0 to 20 rewritten as hexadecimals.

```
10 REM * HEX$ *
20 FOR I=0 TO 20
30 PRINT"&H";HEX$(I);"=";I
40 NEXT
```

LEFT$

Working Takes out only the specified amount of characters from a character string starting from the

(⇔RIGHT$)

Grammar LEFT$ (Character string, n)

Character string.....up to 31 characters
n...amount of characters to take out of the character string 0 to 255

Abbreviation LEF.

Explanation The n amount of character strings taken out from the left of the character string becomes the value of this function. When the value of n is bigger than the amount of characters in the character string, all of the characters become each a null string as the value of those functions when n = 0.

Sample Program

```
10 REM * LEFT$ *
20 A$="HELLO"
30 FOR I=1 TO 5
40 PRINT LEFT$(A$, I)
50 NEXT
RUN
H
HE
HEL
HELL
HELLO
OK
■
```

RIGHT$

Working Takes out only the specified amount of characters from a character string starting from the right.

(⇔LEFT$)

Grammar RIGHT$ (Character string, n)

Character string.....up to 31 characters
n...amount of characters to take out of the character string 0 to 255

Abbreviation RI.

Explanation The n amount of character strings taken out from the right of the character string becomes the value of this function. When the value of n is bigger than the amount of characters in the character string, all of the characters become each a null string as the value of those functions when n = 0.

Sample Program

```
10 REM *RIGHT$ *
20 A$="HELLO"
30 FOR I=1 TO 5
40 PRINT RIGHT$(A$, I)
50 NEXT
RUN
O
LO
LLO
ELLO
HELLO
OK
■
```

MID$

Working Takes out only a specified amount sequence of characters from within a character string.
(⇒LEFT$、RIGHT$)

Grammar MID$ (Character string, initial position, n)

Character string...Up to 31 characters
Initial position...Initial character position which considers the first character position of the character string as number 1
n.....Amount of characters to take out, starting from the specified initial position
0 to 255

Abbreviation MI .

Explanation The n amount of character strings taken out from within the character string becomes the value of this function. When the initial position is bigger than the amount of the character string, null string becomes the value this function.

Sample Program

```
10 REM * MID$ *
20 A$="HIYA"
30 FOR I=1 TO 4
40 PRINT MID$(A$, I, 1)
50 NEXT
RUN
H
I
Y
A
Ok
■
```

LEN

Working Yields the amount of characters of a character string.

Grammar LEN (Character string)

Abbreviation LE .

Explanation All of the characters included within a character string become the value of this function. The amount of characters can go from 0 to 31 and when the character string is a null string, it becomes 0. However, spaces and control codes which do not appear on screen are also counted as one character.

Sample Program

```
10 REM * LEN *
20 INPUT"PLEASE ENTER CHARACTERS";A$
30 L=LEN(A$)
40 PRINT"THE LENGTH IS";L;"CHARACTERS"
```

Particular Functions

PEEK

Working Takes out data from the specified memory address.
(⇔POKE)

Grammar PEEK (address)

Address...address inside the memory
Please refer to p. 104 for the memory map.

Abbreviation PE .

Explanation The 1 byte data taken out of the specified memory address becomes the value of this function.

Sample Program

☞ Please refer to POKE

POS

Working Yields the horizontal position of the cursor on screen.

Grammar POS (Mathematical expression)

Mathematical expression...dummy value 0

Abbreviation None

Explanation The value of the current horizontal position of the cursor on screen becomes the value of this function. The range of the value is 0 to 27.

Sample Program

```
10 REM * POS *
20 CLS
30 FOR X=0 TO 25
40 LOCATE X,0
50 PRINT POS(0)
60 PAUSE 10
70 NEXT
```

On lines 30 to 70, moves the cursor horizontally from 0 to 25, and displays then the horizontal position with line 50.

FRE

Working Yields the size of the unused area of the user memory.

Grammar FRE

Abbreviation FR.

Explanation The amount of unused bytes of the user memory for BASIC programs becomes the value of this function.
Warning) The value may differ depending on the version of BASIC, the status of variables, the presence of programs and the execution of programs, before and after.

Sample Program

```
PRINT FRE
 1854
OK
■
```

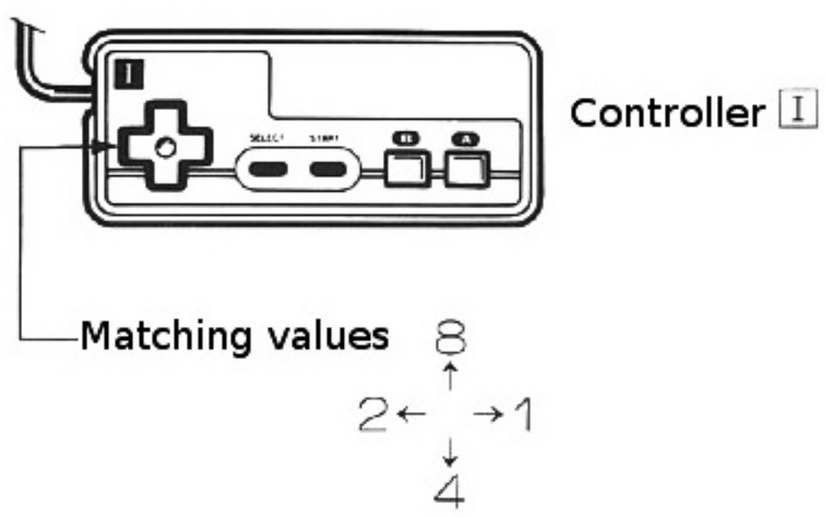
STICK

Working Yields the input value from the ✥ button of the controller.

Grammar STICK(x)
x…0, 1 (0 is for controller I, 1 is for controller II)

Abbreviation STI.

Explanation Yields the value of the ✥ button of the controller.



The value is 0 when nothing is pressed.

Sample Program

```
10 REM * STICK *
20 S=STICK(0)
30 IF S=0 THEN PRINT"NOT PRESSED"
40 IF S=1 THEN PRINT"RIGHT"
50 IF S=2 THEN PRINT"LEFT"
60 IF S=4 THEN PRINT"DOWN"
70 IF S=8 THEN PRINT"UP"
80 GOTO 20
```

When doing RUN, NOT PRESSED will appear immediately on the screen. Please press the ✥ button of controller I to check whether the input value is correct.
Press the STOP key to quit.

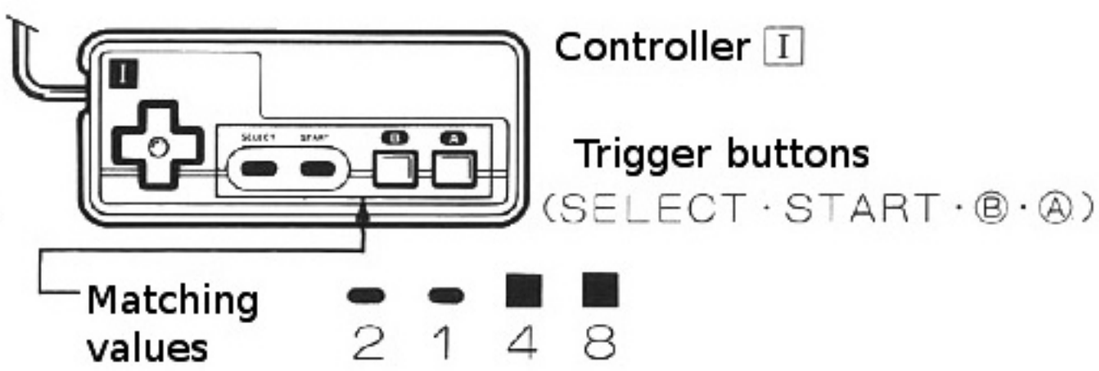
STRIG

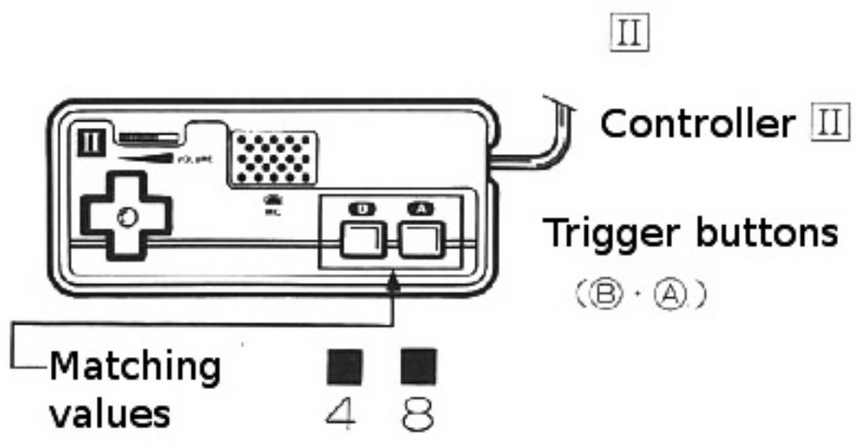
Working Yields the input status value of the trigger buttons of the controller.

Grammar STRIG(x)
x…0, 1 (0 is for controller I, 1 is for controller II)

Abbreviation STR

Explanation Yields a value when a trigger button of the controller is pressed.





The value is 0 when nothing is pressed.

Sample Program

```
10 REM * STRIG *
20 T=STRIG(0)
30 IF T=1 THEN PRINT"START"
40 IF T=2 THEN PRINT"SELECT"
50 IF T=4 THEN PRINT"B"
60 IF T=8 THEN PRINT"A"
70 GOTO 20
```

Please press the trigger buttons of controller I to check whether the input value is correct.
Press the STOP key to quit.

CSRLIN

Working Yields the vertical position of the cursor.

Grammar CSRLIN

There are no arguments.

Abbreviation CSR.

Explanation The current vertical position of the cursor becomes the value of this variable. The range of the value is 0 to 23.

Sample Program

```
10 REM * CSRLIN *
20 CLS
30 FOR I=0 TO 20
40 LOCATE I,I
50 PRINT POS(0);",";CSRLIN
60 PAUSE 20
70 NEXT
```

Through lines 30 to 70, moves the cursor horizontally and vertically 1 character and 1 line, and displays the horizontal and vertical position of the cursor at that time from line 50.

SCR$

Working Function which requests the characters or pictures displayed on the BG GRAPHIC screen.

Grammar SCR$ (X, Y, Sw)

X……Horizontal display column 0 to 27

Y……Vertical display row 0 to 23

Sw…Request a color combination 0 or 1
(You may omit 0.)

Please refer to the color chart on p. 113.

Abbreviation SC.

Explanation Specifies the columns and rows on the BG GRAPHIC screen and lets you know about the characters or pictures displayed there. When selecting 1 for Sw, you can request the color combination number (o to 3) for that character or that picture. Please refer to the sample program to learn how to request.

Sample Program

```
10 CLS
20 LOCATE 0,10
30 PRINT"FAMILY - COMPUTER"
40 PRINT"-----------------------"
50 LOCATE 10,15
60 PRINT SCR$(0,10);
70 A$=SCR$(1,10)
80 PRINT A$
90 C$=SCR$(1,10,1)
100 PRINT"COLOR=";ASC(C$)
110 END
```

The X and Y of SCR$(X,Y) specify columns and rows and this program takes out the characters of the specified position from the "FAMILY-COMPUTER" character string entered on the screen.

Input output character functions

INKEY$

Working Yields one character from the keyboard

Grammar INKEY$ { (n) / omit }

n=∅……Argument

Abbreviation INK.

Explanation The character input from the keyboard becomes the value of this function.
(1) When omitting the argument
When a key is pressed, that character becomes the value. When no key is pressed, null string becomes the value of the function.
(2) When n=0
The cursor blinks and waits until the input of a character.
*You cannot rely on group A to D from character table B on p. 113.

Sample Program

☞ Please refer to END and GOTO

Sprite Control Statements

DEF SPRITE

Working Defines the sprites (animated characters) to be displayed on the sprite screen.

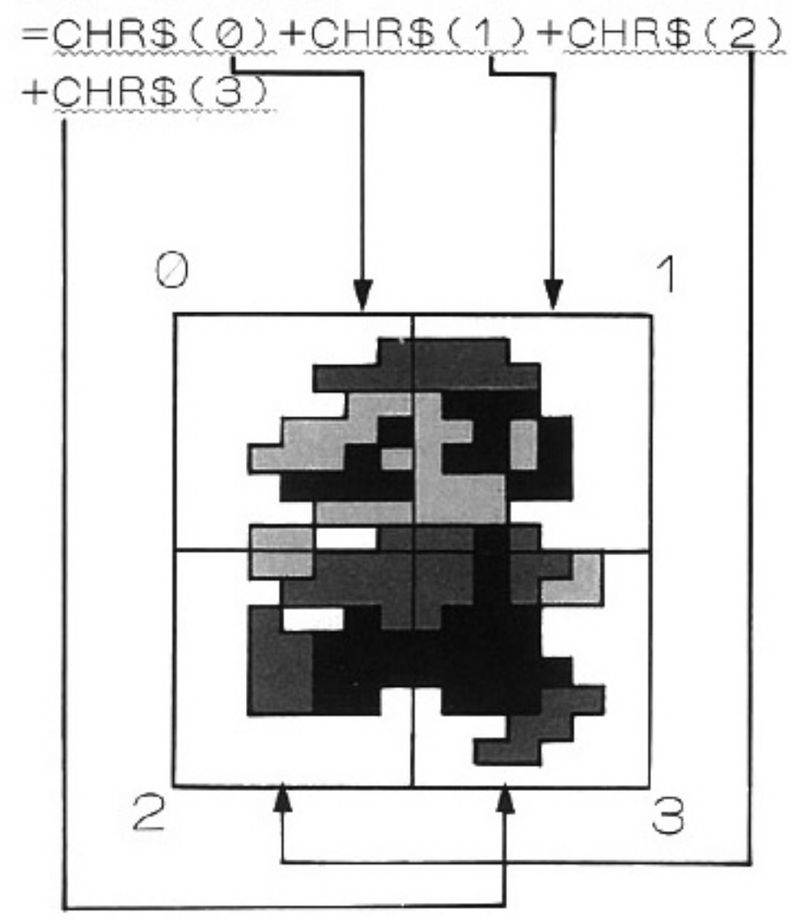
Grammar DEF SPRITE n,(A,B,C,D,E)=char. set

Character set...CHR$(N) or character string "@ABC" or character variable
n...Sprite number 0 to 7
A...Color combination number 0 to 3 (please refer to the color chart or CGSET)
B...Character construction pattern 0 or 1 (0...8x8 dots (size of 1 character), 1...16x16 dots (size of 4 characters))
C...Display priority level 0 or 1 (0...on sprite screen in front of background screen, 1...on sprite screen behind background screen)
D...X axis inversion instruction 0 or 1 (0...same as character table, 1...left right inverted)
E...X axis inversion instruction 0 or 1 (0...same as character table, 1...up down inverted)

Sample Program

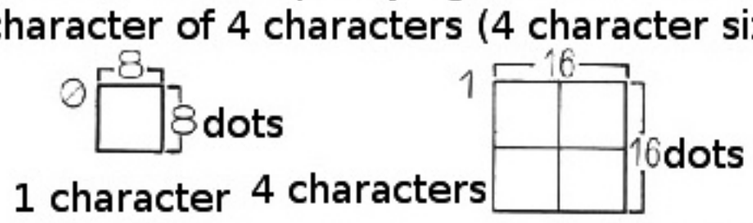
```
10 REM * DEF SPRITE *
20 SPRITE ON
30 DEF SPRITE 0,(0,1,0,0,0)=CHR$(0)
   +CHR$(1)+CHR$(2)+CHR$(3)
40 DEF SPRITE 1,(0,1,0,0,0)="@ABC"
50 SPRITE 0,100,150
60 SPRITE 1,100,100
```

〈Character set and character response〉



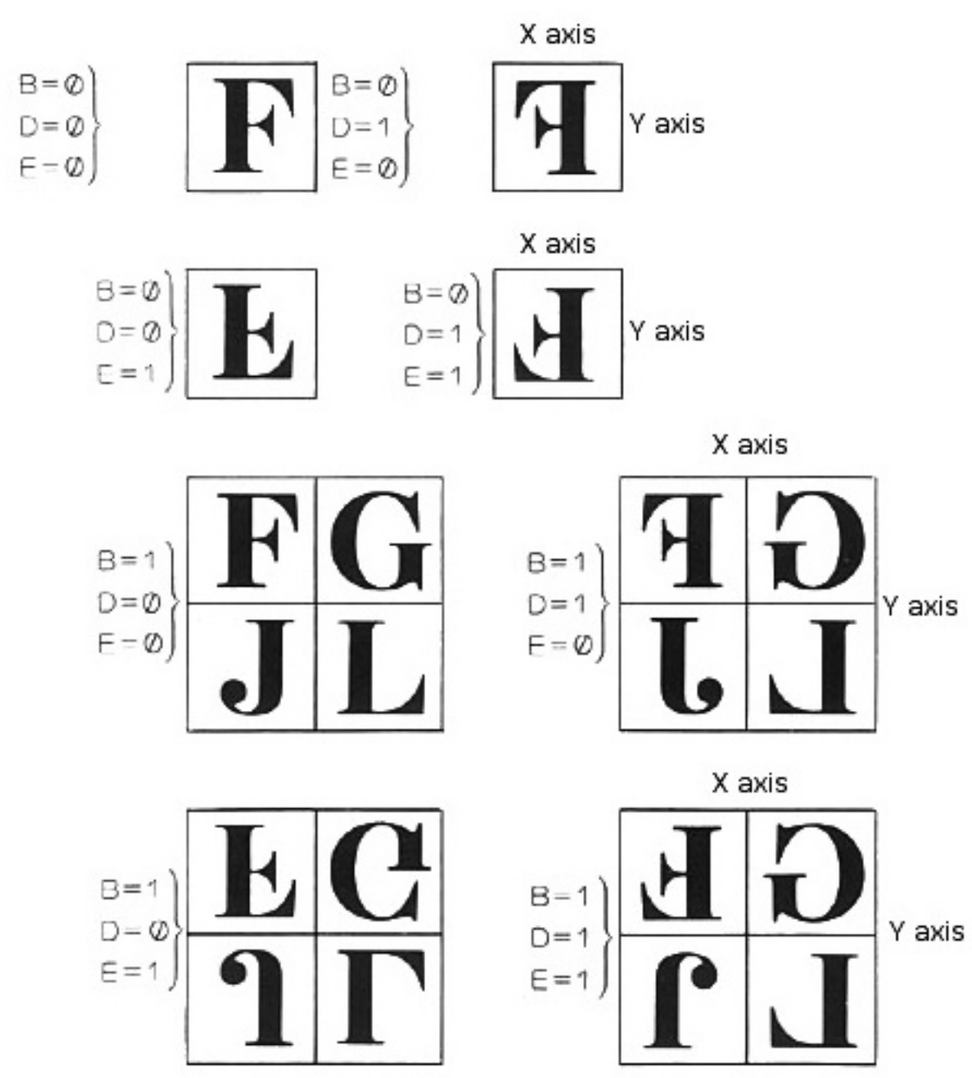
Abbreviation DE.SP.

Explanation Defines the sprites to be displayed on the sprite screen. You can specify and define up to 8 sprite numbers from 0 to 7. When specifying 0 for the character construction pattern, it creates 1 animated character with 1 character (1 character size). When specifying 4, it creates an animated character of 4 characters (4 character sizes).



When specifying 0 for the display priority level, the animated character appears on the sprite screen in front of the background screen. When specifying 1, the animated character appears on the sprite screen behind the background screen.
When specifying 1 for the X, Y inversion instruction, you define characters inverted along the specified axis. When specifying 0, the character remains as is (character table A and B).
-You can use the MOVE command independently from DEF MOVE.
-When used together with CGEN commands, you can use characters or symbols as sprites.
-The N from character string CHR$(N) can be specified with the numbers of the 4 corners of the character table on the back cover or with the codes from character table A and B. You can also use the &HOO hexadeximal value as a numerical value.
-In case of CGEN2,
you can also define
DEF SPRITE 0,(0,1,0,0,0)=
CHR$(68)+CHR$(69)+CHR$(70)+CHR$(71)
as
DEF SPRITE 0,(0,1,0,0,0)=
"DEFG".

〈Definition of inverted characters〉



SPRITE

Working Erases or displays defined sprites on any position.

Grammar SPRITE n[, x , y]

n ···Sprite number 0 to 7

x ···Sprite screen horizontal coordinate 0 to 255

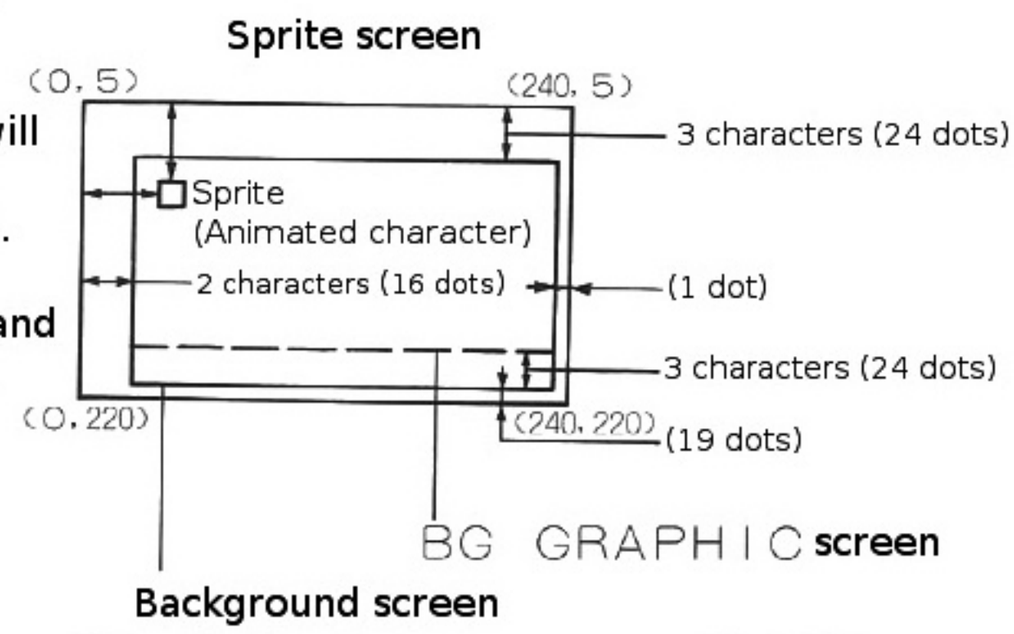
y ···Sprite screen vertical coordinate 0 to 255
(However, the availabe coordinates are x: 0 to 240 and y: 5 to 220)

Abbreviation SP .

Explanation Displays an animated character which has already been defined by DEF SPRITE on the position of your choice on the sprite screen.
When you omit the horizontal and vertical coordinates, the animated character of the specified sprite number will disappear from the displayed animated characters. (SPRITE n)
When multiple animated characters are displayed on the same coordinates, the one with the lowest sprite number will appear in the front. (The display priority ranking among sprites is greater.)
You can display simultaneously up to 4 (8 character parts) animated characters horizontally. (More than 4 characters will not be visible on screen.)
The display of animated characters on the sprite screen will appear on the specified coordinates of the position of the upper left border of the character defined by DEF SPRITE.
When superimposing with BG GRAPHIC, please keep in mind the relationship of the position of the sprite screen and the BG GRAPHIC screen or the sprite display coordinates.

Relational expression
$$\begin{cases} x = (X \times 8) + 16 \\ y = (Y \times 8) + 24 \end{cases}$$

x ···x coordinate of the sprite
y ···y coordinate of the sprite
X ···X coordinate of BG GRAPHIC
Y ···Y coordinate of BG GRAPHIC



*The displayable screen range might differ depending on your TV set.

SPRITE ON

Working Enables the display of the sprite screen. Sprite display mode.

Grammar SPRITE ON

Abbreviation SP.O.

Explanation Enables the stacking of the sprite screen over the background screen and displaying it.
Sprites (animated characters) on the sprite screen become visible on the screen.
You have to enable SPRITE ON before executing DEF SPRITE or DEF MOVE.
It remains active continuously until executing SPRITE OFF.

Sample Program

```
SPRITE ON
OK
■
10 SPRITE ON
SPRITE OFF
OK
■
10 SPRITE OFF
```

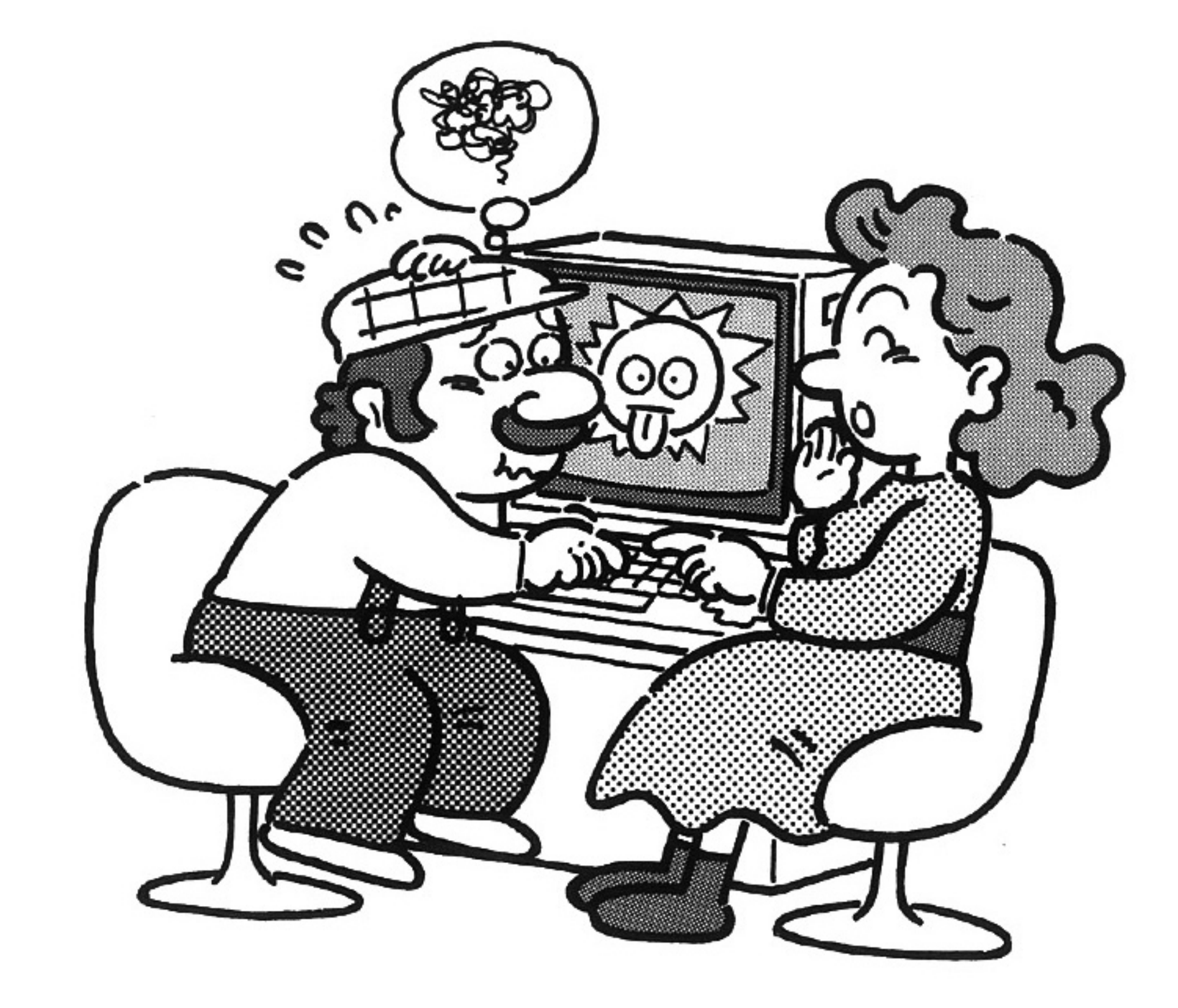
SPRITE OFF

Working Disable the display of the sprite screen. Turns off the sprite mode.

Grammar SPRITE OFF

Abbreviation SP.OF.

Explanation All of the animated characters displayed on the sprite screen stop being visible.
The sprite screen is not stacked on the background screen and displayed anymore.



GAME PROGRAM

Let's enjoy original games with Family Basic!

Sample programs

1、KNIGHT
2、SUPER MEMORY
3、UFO
4、ROUTE 66
5、TYPE MASTER
6、TURTLE
7、CARD
8、PENGUIN

You will find exciting programs which use the specifications of family basic from the next page.
Please enter the programs and enjoy them!
By modifying or following the hints of the sample programs, you will be able to create original games and the world of Family Basic will expand.
*Please start creating these sample programs after mastering the beginner's part of entering programs.

How to read the sample programs

Program list

Please enter the line numbers and command names correctly when entering a program.

```
5 CLS
10 SPRITE ON
20 CGSET 1,1
30 FOR N=0 TO 7
40 DEF MOVE(N)=SPRITE(0,N+1,3,255,0,0)
50 NEXT
60 MOVE 0,1,2,3,4,5,6,7
70 GOTO 60
80 END
```

For long commands, the program list and the actual screen which displays what has been entered differ. Even if the lines change, please continue entering.

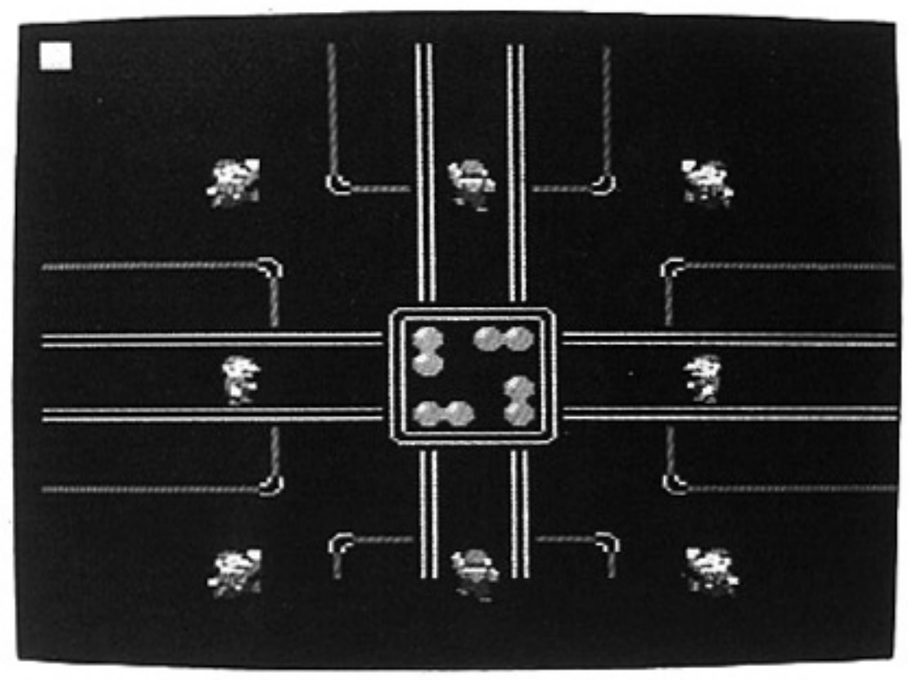
(Example)

```
30 FOR N=0 TO 7
40 DEF MOVE(N)=SPRITE(0,N+1,
3,255,0,0)
50 NEXT
```

Please enter the data by following the list. (Do not leave a space between GOTO/GOSUB/IF~THEN and the line number. This could result in a memory overload (OM ERROR).
(Example: GOTO120)

⟨Sample screen⟩

Let's draw a background screen with BG GRAPHIC! Where should the animated character be moving around? The ocean, the mountains or even in space? This time we'll program an original game in BASIC. You can create very exciting games with the MOVE command! And with the VIEW command, when you superimpose the background screen, you'll put together a fun game!



★Warning: When changing or modifying a program

- When creating, changing or modifying a BASIC program, always erase the BG GRAPHIC (background) screen beforehand. Not doing this might result in an error.
- Press the [CLR HOME] key while holding down the [SHIFT] key to erase the BG GRAPHIC screen. The cursor will return to its home position.
- Call the program with LIST and execute the changes and modifications.

Background screen data

In BG GRAPHIC you can write backgroud lines within the range of 28 horizontal cells and 21 vertical cells. This data shows the design data of each character which you draw in BG GRAPHIC.
An example of "background screen data":

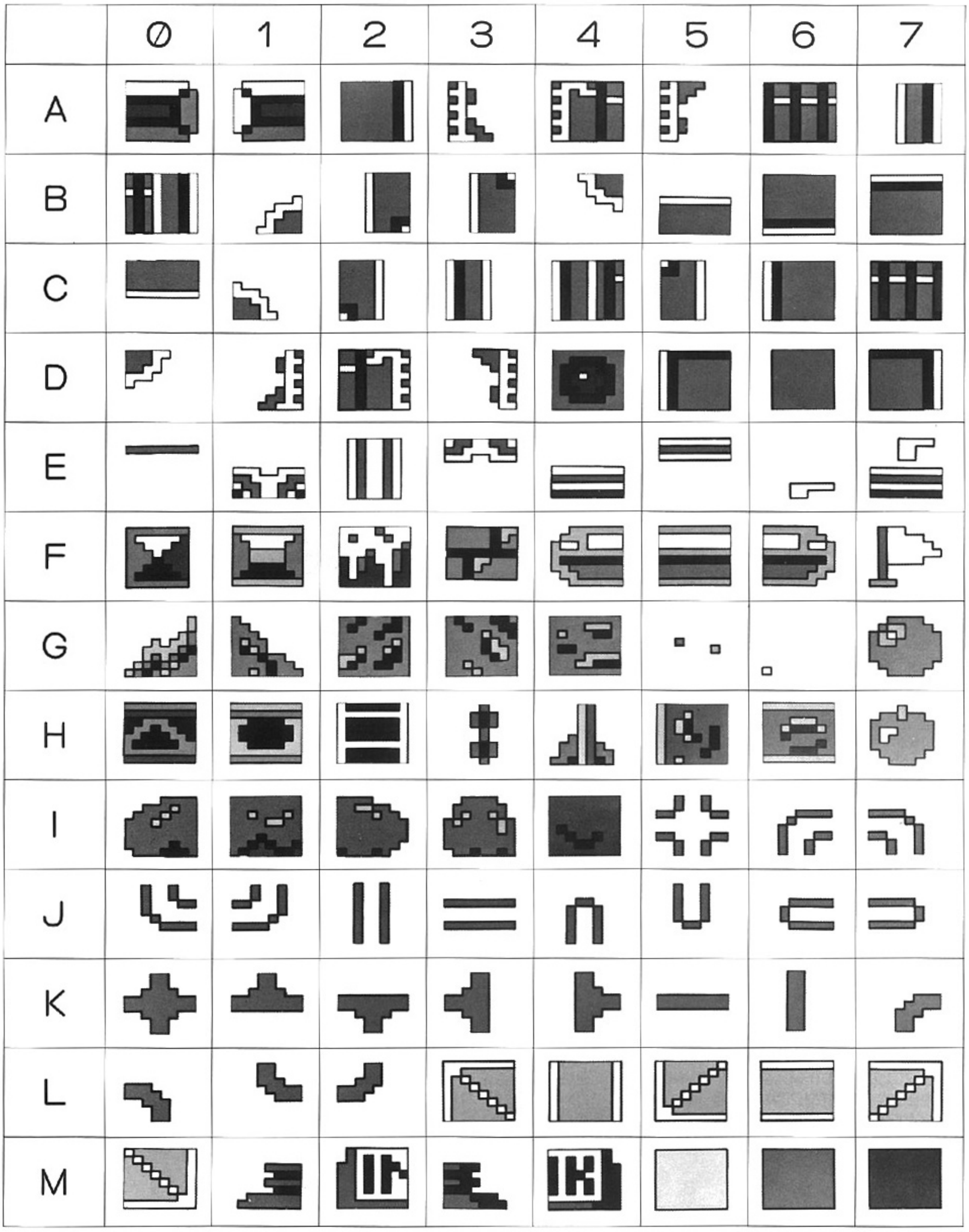
■ K50 means that it uses character 5 within group K together with color combination 0 (the color combination of number 0 which you select in SELECT mode).

■ B73 means that it uses character 7 within group B and color combination 3.

※ Character data of "background screen data" shown in numbers or symbols (example: number 1 or letter A) are to be entered directly with the keyboard in CHAR mode.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
0										K60			J20			J20			K60									
1										K60			J20			J20			K60									
2										K60			J20			J20			K60									
3										K60			J20			J20			K60									
4										K60			J20			J20			K60									
5										J00	K50	K50	J20			J20	K50	K50	J10									
6													J20			J20												
7													J20			J20												
8	K50	K50	K50	K50	K50	K50	K50	I70					J20			J20					I60	K50	K50	K50	K50	K50	K50	K50
9								K60					J20			J20					K60							
10								K60				I60	J30	J30	J30	J30	I70				K60							
11	J30	J30	J30	J30	J30	J30	J30	J30	J30	J30	J30	J20	G72		G72	G72	J20	J30	J30	J30	J30	J30	J30	J30	J30	J30	J30	J30
12												J20	G72				J20											
13												J20				G72	J20											
14	J30	J30	J30	J30	J30	J30	J30	J30	J30	J30	J30	J20	G72	G72		G72	J20	J30	J30	J30	J30	J30	J30	J30	J30	J30	J30	J30
15								K60				J00	J30	J30	J30	J30	J10				K60							
16								K60					J20			J20					K60							
17	K50	K50	K50	K50	K50	K50	K50	J10					J20			J20					J00	K50	K50	K50	K50	K50	K50	K50
18													J20			J20												
19										I60	K50	K50	J20			J20	K50	K50	I70									
20										K60			J20			J20			K60									

You can select among the 13 types of groups (A to M) from below,
and also a color combination among 4 from 0 to 3 to use as a character in BG GRAPHIC.



Sample program 1

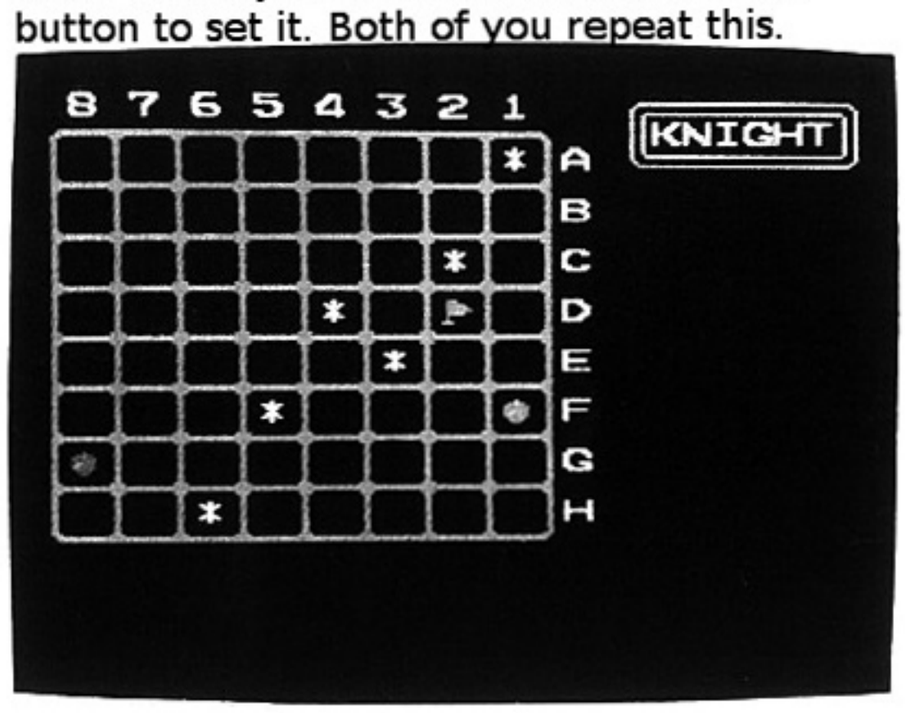
```
** EXERCISE 1 **

10 VIEW:CGEN 3:CGSET 1,1
20 DEF SPRITE 0,(0,0,0,0,0)=CHR$(&HC7)
30 PALETS 0,13,&H16,&H27,2
40 DEF SPRITE 2,(0,0,0,0,0)=CHR$(&HD7)
50 DEF SPRITE 1,(0,0,0,0,0)=CHR$(&HC7)
60 PALETS 1,13,&H16,&H17,4
70 DEF SPRITE 3,(1,0,0,0,0)=CHR$(&HD7)
80 SPRITE ON
90 DIM HX(1),HY(1)
100 DIM X(7),Y(7),B(7,7)
110 X(0)=-1:Y(0)=-2
120 X(1)=-2:Y(1)=-1
130 X(2)=-2:Y(2)= 1
140 X(3)=-1:Y(3)= 2
150 X(4)= 1:Y(4)= 2
160 X(5)= 2:Y(5)= 1
170 X(6)= 2:Y(6)=-1
180 X(7)= 1:Y(7)=-2
190 C=0:GOSUB 250
200 C=1:GOSUB 250
210 C=1+(C=1)
220 GOSUB 390
230 IF F=-1 THEN 560
240 GOTO 210
250 X=0:Y=0:F=0
260 GOSUB 440
270 IF F=1 THEN PLAY"T103C2":F=0:GOTO 260
280 IF T=8 THEN RETURN
290 IF S=4 THEN Y=Y+1:IF Y>7 THEN Y=7
300 IF S=8 THEN Y=Y-1:IF Y<0 THEN Y=0
310 X=X+(S=1)-(S=2)
320 X=-X*(X>0)+(X>7)
330 GOTO 260
340 GOSUB 440:F=0
350 IF T=8 THEN RETURN
360 IF S=0 THEN S=4
370 IF S=8 THEN N=N-1:IF N<0 THEN N=7
380 IF S=4 THEN N=N+1:IF N>7 THEN N=0
390 X=HX(C)+X(N):Y=HY(C)+Y(N)
400 F=F+1:IF F>8 THEN F=-1:RETURN
410 IF X<0 OR X>7 OR Y<0 OR Y>7 THEN 360
420 IF B(X,Y)=1 THEN 360
430 GOTO 340
440 SPRITE C,136-16*X,16*Y+47
450 T=STRIG(C):S=STICK(C)
460 IF (S+T)=0 THEN 450
470 IF T<>8 THEN 540
480 IF B(X,Y)=1 THEN F=1:RETURN
490 B(X,Y)=1
500 HX(C)=X:HY(C)=Y
510 SPRITE C+2,136-HX(C)*16,16*HY(C)+47
520 LOCATE 15-2*HX(C),3+2*HY(C)
530 PRINT"*":PLAY"T103CDEG"
540 SPRITE C
550 RETURN
560 ' END ROUTINE
570 LOCATE 3,20:PLAY"T103CDET204EGAC"
580 IF C=1 THEN PRINT "BLUE ";
590 IF C=0 THEN PRINT "RED ";
600 PRINT "WIN !!":END
```

〈KNIGHT〉

Use the knight's move to move around while placing your pieces on the chessboard. Take time to anticipate the other side's next move. The one who cannot move any more loses.

How to play:

2 players face off each other. Use the up and down directions of the ✥ button of the controller to know where you can move to next. When you have decided, use the A button to set it. Both of you repeat this.



★Warning: When changing or modifying the program

- When creating, changing or modifying a BASIC program, always erase the BG GRAPHIC (background) screen beforehand. Not doing this might result in an error.
- Press the [CLR HOME] key while holding down the [SHIFT] key to erase the BG GRAPHIC screen.
- The cursor will return to its home position.

Call the program with LIST and execute the changes and modifications.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
0																												
1		8		7		6		5		4		3		2		1				I60	J30	J30	J30	J30	J30	J30	J70	
2	K72	K52	K22	K52	K22	K52	K22	K52	K22	K52	K22	K52	K22	K52	K22	K52	L02			J20	K	N	I	G	H	T	J20	
3	K62		K62		K62		K62		K62		K62		K62		K62		K62	A		J00	J30	J30	J30	J30	J30	J30	J10	
4	K42	K52	K02	K52	K02	K52	K02	K52	K02	K52	K02	K52	K02	K52	K02	K52	K32											
5	K62		K62		K62		K62		K62		K62		K62		K62		K62	B										
6	K42	K52	K02	K52	K02	K52	K02	K52	K02	K52	K02	K52	K02	K52	K02	K52	K32											
7	K62		K62		K62		K62		K62		K62		K62		K62		K62	C										
8	K42	K52	K02	K52	K02	K52	K02	K52	K02	K52	K02	K52	K02	K52	K02	K52	K32											
9	K62		K62		K62		K62		K62		K62		K62		K62		K62	D										
10	K42	K52	K02	K52	K02	K52	K02	K52	K02	K52	K02	K52	K02	K52	K02	K52	K32											
11	K62		K62		K62		K62		K62		K62		K62		K62		K62	E										
12	K42	K52	K02	K52	K02	K52	K02	K52	K02	K52	K02	K52	K02	K52	K02	K52	K32											
13	K62		K62		K62		K62		K62		K62		K62		K62		K62	F										
14	K42	K52	K02	K52	K02	K52	K02	K52	K02	K52	K02	K52	K02	K52	K02	K52	K32											
15	K62		K62		K62		K62		K62		K62		K62		K62		K62	G										
16	K42	K52	K02	K52	K02	K52	K02	K52	K02	K52	K02	K52	K02	K52	K02	K52	K32											
17	K62		K62		K62		K62		K62		K62		K62		K62		K62	H										
18	L12	K52	K12	K52	K12	K52	K12	K52	K12	K52	K12	K52	K12	K52	K12	K52	L22											
19																												
20																												

Sample program 2

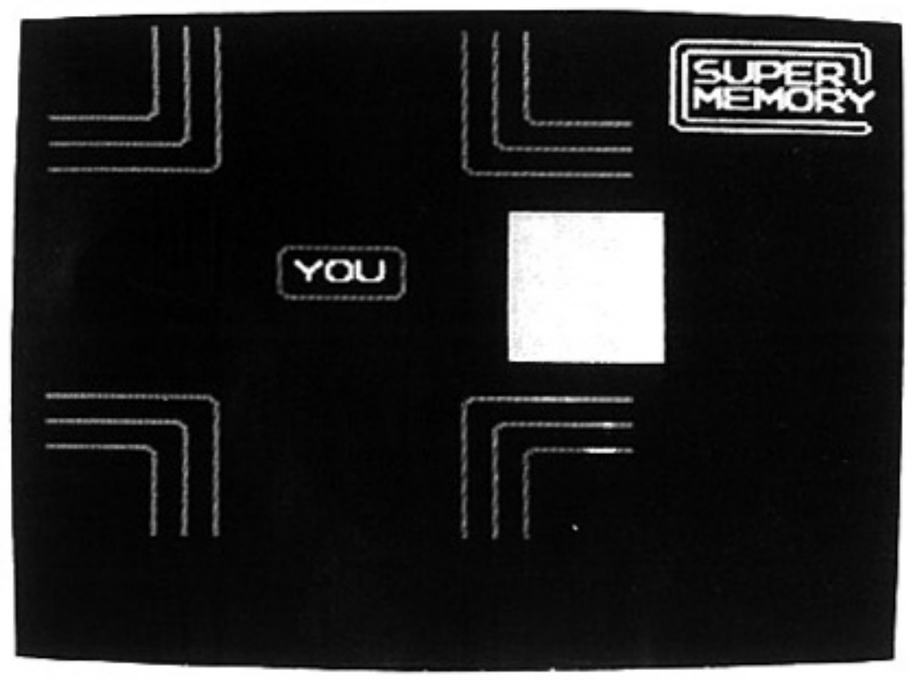
```
** EXERCISE 2 **

10 VIEW:CGEN 2
20 MAX=5:I=Z:X=Y:C=0
30 PP$="CFGE"
40 Z$=CHR$(254):Z$=Z$+Z$+Z$+Z$+Z$
50 N$="      "
60 DIM PL(MAX),PX(3),PY(3),C(3)
70 PX(0)=16:PY(0)=8:C(0)=2
80 PX(1)=0:PY(1)=8:C(1)=4
90 PX(2)=8:PY(2)=0:C(2)=6
100 PX(3)=8:PY(3)=16:C(3)=8
110 '
120 PL(Z)=RND(4)
130 FOR I=0 TO Z
140 X=PX(PL(I)):Y=PY(PL(I))
150 C=C(PL(I))
160 PALETB 0,13,13,13,13
170 GOSUB 440
180 PLAY"T204"+MID$(PP$,PL(I)+1,1)+"3"
190 GOSUB 500
200 PALETB 0,13,&H16,&H27,2
210 PAUSE 10
220 NEXT
230 '
240 I=0
250 LOCATE 9,10:PRINT"YOU"
260 A$=INKEY$:IF A$="" THEN 260
270 IF A$<CHR$(28) OR A$>CHR$(31) THEN PLAY"T101C1C1C1":GOTO 260
280 IF (ASC(A$)-28)<>PL(I) THEN PLAY"T205C2R2F2R2E2":LOCATE 9,10:
    PRINT"   ":GOTO 130
290 PALETB 0,13,13,13,13
300 X=PX(PL(I)):Y=PY(PL(I))
310 C=C(PL(I))
320 GOSUB 440
330 PLAY"T204"+MID$(PP$,PL(I)+1,1)+"3"
340 GOSUB 500
350 PALETB 0,13,&H16,&H27,2
360 I=I+1:IF I<=Z THEN 250
370 Z=Z+1:IF Z>MAX THEN 410
380 LOCATE 9,10:PRINT"   "
390 PAUSE 50
400 GOTO 110
410 ' END
420 CLS:CGSET 1,1:PRINT"GOOD!!"
430 END
440 FOR J=0 TO 5
450 LOCATE X,Y+J
460 PRINT Z$
470 NEXT
480 PALETB 0,13,C,C,C
490 RETURN
500 PALETB 0,13,13,13,13
510 FOR J=0 TO 5
520 LOCATE X,Y+J
530 PRINT N$
540 NEXT
550 RETURN
```

⟨SUPER MEMORY⟩

The computer tests your faculty to memorize. Can you correctly memorize the sequences of up, down, left, right and color panel blinking?

How to play:

1 player. Remember the four-directional color panel sequences which blink at random. When "YOU" appears in the center, press the cursor key in the same order. The up, down, left, right of the screen matches [▲] [▼] [◀] [▶]. If you make a mistake, the computer will boo at you. It will blink the same sequence. On line 20, you can set the maximum amount of blinking. If you increase the number, the amount of blinking will increase as well.



★Warning: When changing or modifying the program

- When creating, changing or modifying a BASIC program, always erase the BG GRAPHIC (background) screen beforehand. Not doing this might result in an error.
- Press the [CLR HOME] key while holding down the [SHIFT] key to erase the BG GRAPHIC screen. The cursor will return to its home position.
- Call the program with LIST and execute the changes and modifications.

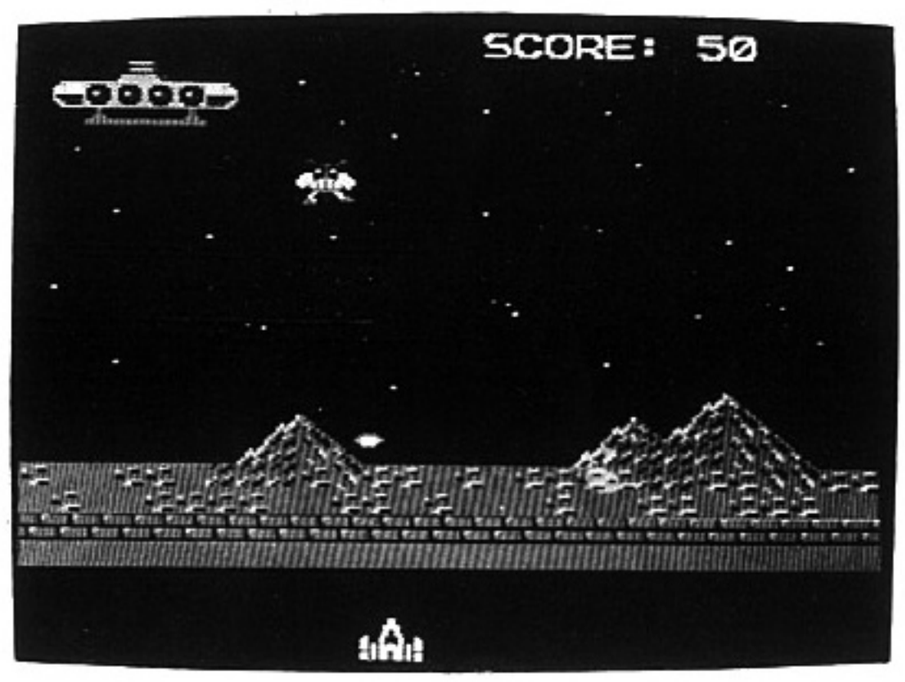
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
0																												
1					K62	K62	K62								K62	K62	K62					I60	J30	J30	J30	J30	J30	I70
2					K62	K62	K62								K62	K62	K62					J20	S	U	P	E	R	J50
3					K62	K62	K62								K62	K62	K62					J20	M	E	M	O	R	Y
4		K52	K52	K52	L22	K62	K62								K62	K62	L12	K52	K52	K52		J00	J30	J30	J30	J30	J30	J70
5		K52	K52	K52	K52	L22	K62								K62	L12	K52	K52	K52	K52								
6		K52	K52	K52	K52	K52	L22								L12	K52	K52	K52	K52	K52								
7																												
8																												
9								K72	K52	K52	K52	L02																
10								K62				K62																
11								L12	K52	K52	K52	L22																
12																												
13																												
14																												
15		K52	K52	K52	K52	K52	L02								K72	K52	K52	K52	K52	K52								
16		K52	K52	K52	K52	L02	K62								K62	K72	K52	K52	K52	K52								
17		K52	K52	K52	L02	K62	K62								K62	K62	K72	K52	K52	K52								
18					K62	K62	K62								K62	K62	K62											
19					K62	K62	K62								K62	K62	K62											
20					K62	K62	K62								K62	K62	K62											

Sample program 3

```
** EXERCISE 3 **

10 VIEW:CGEN 2:PLAY"T1C1"
20 U0$="89:;":U1$="<=>?":B$="ﾋﾟﾌﾟﾍﾟﾎﾟ":FOR I=0 TO 3:
   X$=X$+CHR$(180+I):NEXT:V=1
30 LG=16:RG=220:TG=50:BG=100:GX=0:GY=0:GV=16:BX=100:
   BY=220:BV=6:DX=0:DY=0:DV=30:R=12:X=0:Y=0:D=0
40 G0$=CHR$(212):G1$=CHR$(213):D0$=CHR$(209):D1$=G1$
50 PALETS 1,13,48,22,7:PALETS 2,13,48,22,1:PALETS 3,13,48,22,1:
   DEF SPRITE 1,(2,1,0,0,0)=X$:DEF SPRITE 6,(3,1,0,0,0)=B$:
   GOSUB310:SPRITE ON
60 VX=V+RND(R):VY=V+RND(R)
70 C=RND(5):Y=Y+VY:X=X+VX*SGN(C):GOSUB260:T=STRIG(0):S=STICK(0)
80 SWAP G0$,G1$:SWAP D0$,D1$:SWAP U0$,U1$:GOSUB310
90 IF G=-1 THEN120
100 IF T<8 THEN160
110 GX=BX+6:GY=BY:G=-1
120 GY=GY-GV
130 IF GY<TG THEN G=0:GX=0:GY=0:GOTO160
140 SPRITE 7,GX,GY
150 IF ABS(GX-X-5)<6 THEN IF ABS(GY-Y+8)<8 THEN330
160 IF S=1 THEN BX=BX+BV:IF BX>RG THEN BX=RG
170 IF S=2 THEN BX=BX-BV:IF BX<LG THEN BX=LG
180 IF D=-1 THEN210
190 IF ABS(BX-X)>5 THEN240
200 DX=X+6:DY=Y:D=-1
210 DY=DY+DV:IF DY>255 THEN SPRITE 5:D=0:DX=0:DY=0:GOTO240
220 SPRITE 5,DX,DY
230 IF (BY-DY)<6 THEN IF ABS(BX-DX+6)<8 THEN350
240 IF BG<210 THEN IF RND(4)=0 THEN TG=TG+V:BG=BG+V
250 GOTO70
260 IF X<LG THEN X=LG:VX=-VX
270 IF X>RG THEN X=RG:VX=-VX
280 IF Y<TG THEN Y=TG:VY=-VY
290 IF Y>BG THEN Y=BG:VY=-VY:IF BG>=210 THEN360
300 RETURN
310 DEF SPRITE 0,(1,1,0,0,0)=U0$:DEF SPRITE 5,(0,0,0,0,0)=D0$:
    DEF SPRITE 7,(3,0,0,0,0)=G0$
320 SPRITE 0,X,Y:SPRITE 7,GX,GY:SPRITE 5,DX,DY:SPRITE 6,BX,BY:RETURN
330 FOR I=0 TO 9:SWAP X$,U0$:DEF SPRITE 0,(3,1,0,0,0)=U0$:
    SPRITE 0,X,Y:PAUSE 10:PLAY"O5F1":NEXT
340 PR=PR+10:LOCATE 15,0:PRINT"SCORE:";PR:GX=0:GY=0:V=1+PR/20:GOTO30
350 FOR I=0 TO 5:SPRITE 6:PLAY"O1B1":PAUSE 8:SPRITE 6,BX,BY:
    PAUSE 10:NEXT:SPRITE 5:DX=0:DY=0:D=0:B=B+1:IF B<4 THEN30
360 PLAY"O1C4EC":INPUT"RETURN";I:RUN
```

〈UFO〉

Appearance of a fighter fly from an unidentified flying object! Your starship is being targeted. Move quickly to protect yourself from the attacks of the fighter fly and start firing back!
How to play:
1 player. Press the left and right directions of the ✚button on controller [I] to move the starship. Press button A to shoot missiles. Hit that fighter fly!



★Warning: When changing or modifying the program

- When creating, changing or modifying a BASIC program, always erase the BG GRAPHIC (background) screen beforehand. Not doing this might result in an error.

- Press the [CLR HOME] key while holding down the [SHIFT] key to erase the BG GRAPHIC screen. The cursor will return to its home position.
- Call the program with LIST and execute the changes and modifications.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
0																												
1				M11	M31					G62			G52															
2		F41	D41	D41	D41	D41	F61									G62				G62					G52			
3			K12	K52	K52	K12							G62															
4		G52																			G62							G62
5																												
6				G62												G62												
7							G62				G62								G52					G62				
8																										G62		
9		G62																										
10																G52	G62						G62		G52			
11								G52																			G62	
12				G62																G62								
13													G62															
14																							G00	G10				
15									G00	G10										G00	G10	G00	G20	G30	G10			
16								G00	G20	G30	G10								G00	G20	G20	G20	G20	G30	G30	G10		
17	G40	M70	M70	G40	G40	M70	G20	G20	G20	G40	G30	G40	G40	M70	G40	M70	G40	G40	G40	G40	G20	G20	G40	G40	G30	G30	G40	G40
18	M70	G40	M70	M70	G40	G40	G40	G40	M70	M70	G40	G40	M70	G40	M70	M70	M70	G40	M70	M70	G40	G40	M70	G40	G40	G40	M70	M70
19	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32
20	M72	M72	M72	M72	M72	M72	M72	M72	M72	M72	M72	M72	M72	M72	M72	M72	M72	M72	M72	M72	M72	M72	M72	M72	M72	M72	M72	M72

Sample program 4

```
** EXERCISE 4 **

10 VIEW:CGEN 2:CGSET 1,2:PALETS 0,13,22,39,2:SPRITE ON:PLAY"T103"
20 X1=150:X2=170:X3=190:X6=136:X7=208:
   C0$="ﾙﾚﾛﾌ":C1$="ﾝﾗｯｲ":B$=CHR$(214)
30 FOR I=4 TO 7:DEF SPRITE I,(0,0,0,0,0)=B$:NEXT
40 F=2:SC=0:CA=3:L=0:X=208:Y0=150:V=0
50 L=L+1:M=L*3+15:MV=M+33:D=(M+27)*400
60 C=0
70 S=STICK(0):T=STRIG(0):V=V-1
80 IF S=1 THEN X=X+5
90 IF S=2 THEN X=X-5
100 IF T=8 THEN V=V+4:IF V>MV THEN V=MV
110 IF T=4 THEN V=V-4
120 IF V<1 THEN V=0
130 IF X<143 OR X>193 THEN310
140 F=1
150 Y=Y0-V/2:DEF SPRITE 0,(0,1,0,0,0)=C0$:SPRITE 0,X,Y:SWAP C0$,C1$
160 IF F1=-1 AND F2=-1 AND F3=-1 THEN SC=SC+1:F1=0:F2=0:F3=0
170 FOR I=1 TO 3:DEF SPRITE I,(1,1,0,0,0)=C0$:NEXT
180 YY=(40-Y5)*(Y5>40):SPRITE1,X1,Y1:SPRITE2,X2,Y2:SPRITE3,X3,Y3:
 SPRITE4,X6,Y5:SPRITE5,X7,Y5:SPRITE6,X6,YY:SPRITE7,X7,YY
190 IF ABS(Y1-Y)<12 THEN F1=V>V1:IF ABS(X1-X)<10 THEN340
200 IF ABS(Y2-Y)<12 THEN F2=V>V2:IF ABS(X2-X)<10 THEN340
210 IF ABS(Y3-Y)<15 THEN F3=V>V3:IF ABS(X3-X)<10 THEN340
220 Y1=Y1+V-V1:Y2=Y2+V-V2:Y3=Y3+V-V3:Y5=Y5+V:XX=XX+2:X1=ABS(XX)+143:
    IF XX>48 THEN XX=-51
230 IF Y1>250 THEN Y1=0:V1=RND(M)+30
240 IF Y2>250 THEN Y2=0:X2=RND(58)+138:V2=RND(M)+30
250 IF Y3>250 THEN Y3=0:X3=RND(58)+138:V3=RND(M)+30
260 Y5=-Y5*(Y5<250):Y1=Y1-(Y1<0)*255:Y2=Y2-(Y2<0)*255:Y3=Y3-(Y3<0)*255
270 D=D-V:IF C-C/100*100=0 THEN GOSUB380
280 IF D<0 THEN GOSUB380:PRINT"GOOD!  ":PLAY"CGDAB":SC=SC+(500-C)/10:
    GOTO50
290 C=C+1:IF C<401 THEN70
300 PRINT"TIME UP":PLAY"BAGFEDC":F=3:L=L-1:GOTO340
310 V=V-4:IF F=2 THEN V=V+4:IF V>MV-15 THEN V=MV-15
320 IF V<0 THEN V=0:F=2
330 GOTO150
340 PLAY"BAEDC":CA=CA-1:IF CA>-1 THEN X=208:V=0:GOSUB380:
    ON F GOTO 70,70,50
350 PRINT"GAME OVER!":PRINT"TRY AGAIN"
360 IF STRIG(0)<>1 THEN360
370 GOSUB380:GOTO10
380 IF SC>HI THEN HI=SC
390 LOCATE 0,0:PRINT"HIGH SCORE ";HI:PRINT"LEVEL ";L:PRINT"SCORE ";SC:
    PRINT"CARS ";CA:PRINT:PRINT:PRINT"LEFT";D/4;" M ":PRINT:
    PRINT"LEFT";400-C;" SEC ":PRINT:RETURN
```

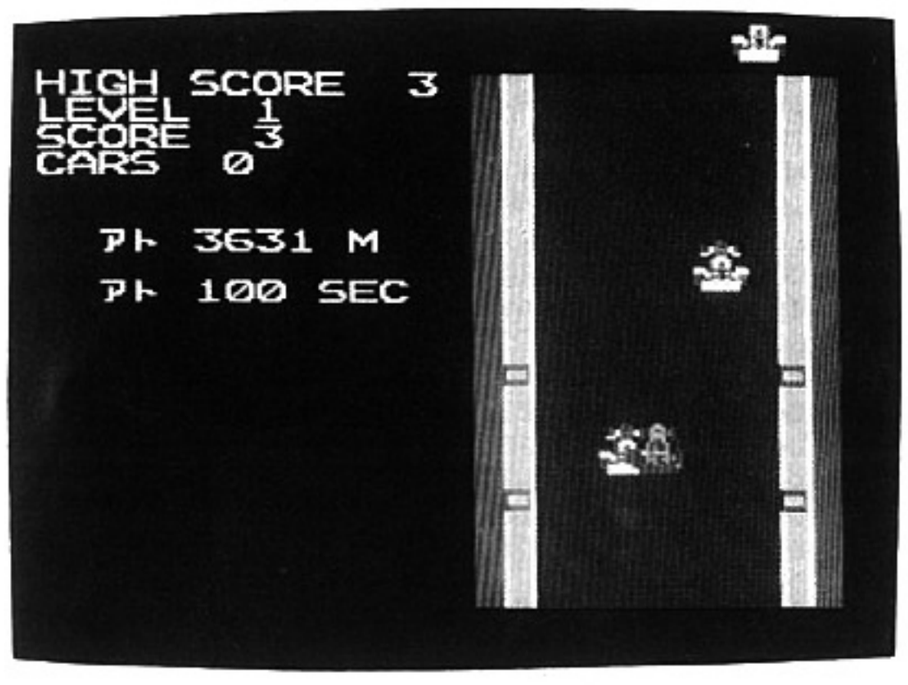
※If an OM ERROR occurs, it means that an unnecessary space has been entered into the program.
Remove the unnecessary spaces and execute the program once more.

★Warning: When changing or modifying the program

- When creating, changing or modifying a BASIC program, always erase the BG GRAPHIC (background) screen beforehand. Not doing this might result in an error.
- Press the [CLR HOME] key while holding down the [SHIFT] key to erase the BG GRAPHIC screen. The cursor will return to its home position.
- Call the program with LIST and execute the changes and modifications.

〈ROUTE 66〉

Scorch the endless road in your racing car! Accelerate, steer sharply and get ahead of those who are in your way.
How to play:
1 player. Cover the set distance within the time limit. Press the left and right directions of the ✥ button on controller [I] to steer and A to accelerate, B to brake. You will explode if you bump into other cars. Start the run while accelerating from the start position.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
0														M70	L40	M71	M71	M71	M71	M71	M71	M71	M71	M71	L40	M70		
1														M70	L40	M71	M71	M71	M71	M71	M71	M71	M71	M71	L40	M70		
2														M70	L40	M71	M71	M71	M71	M71	M71	M71	M71	M71	L40	M70		
3														M70	L40	M71	M71	M71	M71	M71	M71	M71	M71	M71	L40	M70		
4														M70	L40	M71	M71	M71	M71	M71	M71	M71	M71	M71	L40	M70		
5														M70	L40	M71	M71	M71	M71	M71	M71	M71	M71	M71	L40	M70		
6														M70	L40	M71	M71	M71	M71	M71	M71	M71	M71	M71	L40	M70		
7														M70	L40	M71	M71	M71	M71	M71	M71	M71	M71	M71	L40	M70		
8														M70	L40	M71	M71	M71	M71	M71	M71	M71	M71	M71	L40	M70		
9														M70	L40	M71	M71	M71	M71	M71	M71	M71	M71	M71	L40	M70		
10														M70	L40	M71	M71	M71	M71	M71	M71	M71	M71	M71	L40	M70		
11														M70	L40	M71	M71	M71	M71	M71	M71	M71	M71	M71	L40	M70		
12														M70	L40	M71	M71	M71	M71	M71	M71	M71	M71	M71	L40	M70		
13														M70	L40	M71	M71	M71	M71	M71	M71	M71	M71	M71	L40	M70		
14														M70	L40	M71	M71	M71	M71	M71	M71	M71	M71	M71	L40	M70		
15														M70	L40	M71	M71	M71	M71	M71	M71	M71	M71	M71	L40	M70		
16														M70	L40	M71	M71	M71	M71	M71	M71	M71	M71	M71	L40	M70		
17														M70	L40	M71	M71	M71	M71	M71	M71	M71	M71	M71	L40	M70		
18														M70	L40	M71	M71	M71	M71	M71	M71	M71	M71	M71	L40	M70		
19														M70	L40	M71	M71	M71	M71	M71	M71	M71	M71	M71	L40	M70		
20														M70	L40	M71	M71	M71	M71	M71	M71	M71	M71	M71	L40	M70		

Sample program 5

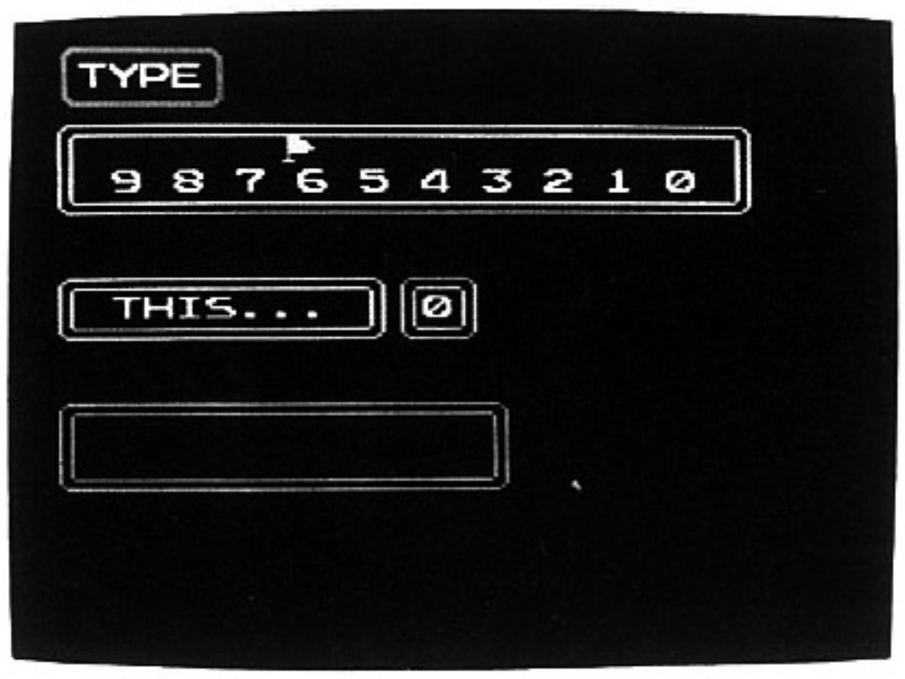
```
** EXERCISE 5 **

10 CGEN 3:CGSET 1,1
20 VIEW:PLAY"T104C1"
30 PO=PP:I=J:K=L
40 PO$="0102030405"
50 PP$="CDEFGAB"
60 A$=B$
70 M3$="THIS IS KANA!"
80 N$=CHR$(227):NN$=N$+N$+N$+N$+N$+N$+N$
90 DIM DP(64),PP(64)
100 FOR I=0 TO 63:READ DP(I):NEXT
110 DEF SPRITE 0,(0,0,0,0,0)=CHR$(199)
120 SPRITE ON
130 VIEW:K=0:LOCATE 2,5:FOR I=9 TO 0 STEP -1:PRINT I;:NEXT
140 PAUSE 100:A$=CHR$(48+RND(43)):SPRITE 0
150 LOCATE 3,10:PRINT "THIS...";
160 LOCATE 13,10:PRINT A$
170 GOSUB 460
180 FOR I=0 TO 75
190 FOR J=0 TO 5:B$=INKEY$
200 SPRITE 0,38+2*I,53
210 IF B$<>"" THEN SWAP A$,B$:GOSUB 370:SWAP A$,B$:
    IF A$=B$ THEN I=500:J=I
220 NEXT:NEXT:LOCATE 5,3:PRINT NN$
230 IF I>100 THEN PLAY"04B1AG2FE3D4C":K=K+1:GOTO 140
240 LOCATE 4,15:PRINT K;"TIME(S) CORRECT"
250 LOCATE 3,16:PRINT"TRY AGAIN ?";:A$=INKEY$(0):
    IF A$="N" OR A$="↑" THEN LOCATE 0,18:END
260 GOTO 130
270 '
280 DATA 0,21,22,23,24,24,25,25,26,27
290 DATA 14,14,5,28,6,7
300 DATA 29,21,22,23,24,24,25,25,26,27
310 DATA 14,14,5,28,6,7,21
320 DATA 7,3,2,9,16,10,10
330 DATA 11,19,11,12,13,4,4
340 DATA 20,21,14,17,8,17,18
350 DATA 3,15,1,18,0
360 DATA 0,28,0,28,7
370 IF A$ <" " THEN PLAY"01C1EC":A$=""
380 IF A$ >"_" THEN PLAY"05B1AB":LOCATE 5,3:PRINT M3$:A$=" "
390 IF A$="" THEN RETURN
400 IF A$="F" OR A$="J" THEN PLAY"00A1:00B1":GOTO 450
410 PO=DP(ASC(A$)-32)/7
420 PP=DP(ASC(A$)-32)-PO*7
430 PLAY MID$(PO$,PO*2+1,2)+MID$(PP$,PP+1,1)
440 PLAY MID$(PP$,PP+1,1)
450 RETURN
460 FOR J=0 TO 5:GOSUB 400:NEXT:RETURN
```

⟨TYPE MASTER⟩

Remembering the placement of keys is hard, right? But, it's all right. Let's memorize their placement while having fun by playing TYPE MASTER! Can you become a master typist?

How to play:

1 player. Search the keys and enter the same letters, numbers or symbols which appear to the right in the middle. If the flag moves completely to the right, time is up.



★Warning: When changing or modifying the program

- When creating, changing or modifying a BASIC program, always erase the BG GRAPHIC (background) screen beforehand. Not doing this might result in an error.
- Press the [CLR HOME] key while holding down the [SHIFT] key to erase the BG GRAPHIC screen. The cursor will return to its home position.
- Call the program with LIST and execute the changes and modifications.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
0		K72	K52	K52	K52	K52	L02																					
1		K62	T	Y	P	E	K62																					
2		L12	K52	K52	K52	K52	L22																					
3		I60	J30	J30	J30	J30	J30	J30	J30	J30	J30	J30	J30	J30	J30	J30	J30	J30	J30	J30	J30	J30	J30	I70				
4		J20																						J20				
5		J20																						J20				
6		J00	J30	J30	J30	J30	J30	J30	J30	J30	J30	J30	J30	J30	J30	J30	J30	J30	J30	J30	J30	J30	J30	J10				
7																												
8																												
9		I60	J30	J30	J30	J30	J30	J30	J30	J30	J30	I70	I62	J32	I72													
10		J20										J20	J22		J22													
11		J00	J30	J30	J30	J30	J30	J30	J30	J30	J30	J10	J02	J32	J12													
12																												
13																												
14		I63	J33	J33	J33	J33	J33	J33	J33	J33	J33	J33	J33	J33	J33	J33	I73											
15		J23															J23											
16		J23															J23											
17		J03	J33	J33	J33	J33	J33	J33	J33	J33	J33	J33	J33	J33	J33	J33	J13											
18																												
19																												
20																												

Sample program 6

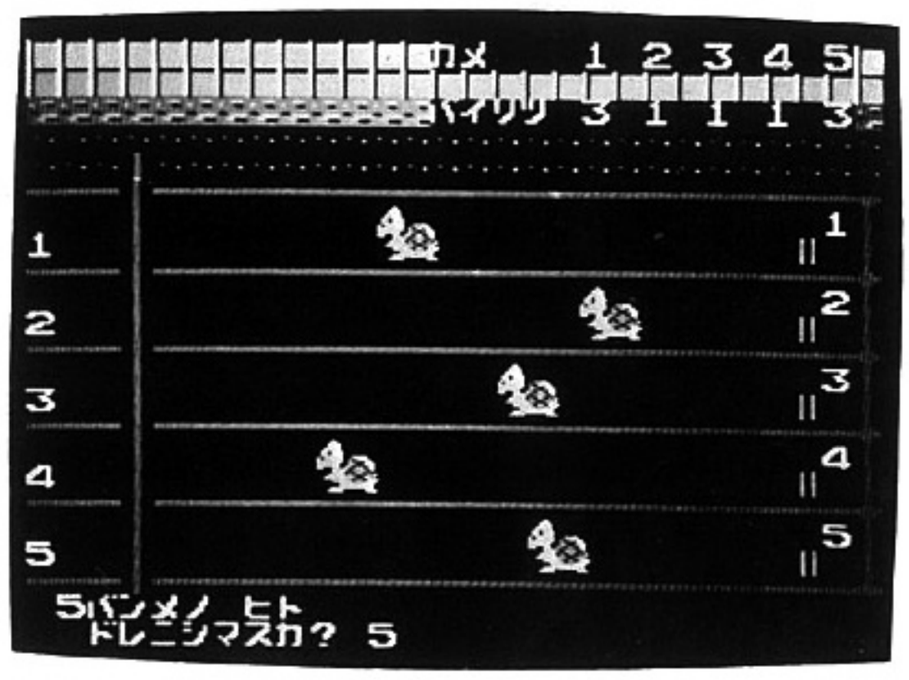
```
** EXERCISE 6 **

10 CLS:CGEN 2:CGSET 1,2
20 I=K:A=J:BT=5:MAX=3
30 PR=PP:W=-1
40 PLAY"O4T1"
50 P$="C2D2E2F2G2A2B2C2"
60 DIM AX(4),AY(4),Q(4)
70 DIM P(5,1),C(5)
80 INPUT "HOW MANY PLAYERS?",PL
90 IF PL=0 OR PL>5 THEN PLAY"C1CC":CLS:GOTO 80
100 VIEW
110 LOCATE 13,0:PRINT"TURTLE "
120 LOCATE 13,2:PRINT"FACTOR"
130 FOR I=0 TO 4
140 Q(I)=RND(MAX)+BT
150 C(I)=MAX+BT-Q(I)
160 DEF SPRITE I,(0,1,0,0,0)=CHR$(184)+CHR$(185)
    +CHR$(186)+CHR$(187)
170 LOCATE 17+2*I,0:PRINT I+1
180 LOCATE 17+2*I,2:PRINT C(I)
190 AX(I)=220
200 P(I,0)=-1
210 NEXT
220 '
230 FOR I=0 TO PL-1
240 LOCATE 0,21:PRINT I+1;"  PLAYER   "
250 INPUT "   TURTLE?   ",M
260 IF M=0 OR M>5 THEN PLAY"C1CC":GOTO 240
270 P(I,0)=M-1
280 NEXT
290 FOR M=4 TO 20:LOCATE 3,M:PRINT CHR$(238):NEXT
300 FOR M=0 TO 4:LOCATE 25,6+3*M:PRINT M+1:NEXT
310 '
320 SPRITE ON
330 FOR I=0 TO 4
340 A=RND(RND(Q(I)))
350 AY(I)=71+24*I
360 AX(I)=AX(I)-A*2
370 IF AX(I)<0 THEN AX(I)=0
380 PLAY MID$(P$,A*2+1,2)
390 SPRITE I,AX(I),AY(I)
400 IF AX(I)<50 THEN GOSUB 560
410 NEXT
420 K=K+1
430 IF K=20 OR K=50 THEN GOSUB 450
440 GOTO 310
450 '
460 FOR I=0 TO 4
470 LOCATE 10,10:PRINT"CHANGE!"
480 A=RND(8)
490 IF A=7 THEN Q(I)=9:GOTO 530
500 IF A=6 THEN IF RND(5)=4 THEN Q(I)=0:GOTO 530
510 IF A<5 THEN 530
520 Q(I)=11-Q(I)
530 NEXT
540 LOCATE 10,10:PRINT"
550 RETURN
560 '
570 IF I=W THEN RETURN
580 IF W<>-1 THEN 600
590 W=I:RETURN
600 '
610 PLAY"T6CEDFEGC2T1"
620 LOCATE 10,6:PRINT"WIN";W+1;"-";I+1
630 SPRITE OFF
640 FOR J=0 TO PL-1
650 PR=0
660 IF P(J,0)=W THEN PR=2
670 IF P(J,0)=I THEN PR=1
680 PP=PR*C(P(J,0))
690 LOCATE 9,9+J*2:PRINT J+1;"PLAYER";PP;"POINTS"
700 P(J,1)=P(J,1)+PP
710 NEXT
720 LOCATE 16,22:PRINT"TRY AGAIN ?";:A$=
    INKEY$(0)
730 IF A$="N" OR A$="ｎ" THEN 750
740 RUN
750 CLS:LOCATE 5,10:PRINT"----- END -----":
    PLAY"CDGAC":END
```

〈TURTLE〉

5 turtles are racing to reach the goal. The ones with a low factor are the favorites, but there could be unexpected changes. Until reaching the goal you can't know who's gonna win, making this game very exciting!

How to play:

Up to 5 players can play together. First choose the amount of players and while looking at the factor, imagine which turtle will win. If you're right, you'll receive points according to that factor. When "CHANGE" appears in the middle of the game, it makes it more exciting by changing an internal random number.



★Warning: When changing or modifying the program

- When creating, changing or modifying a BASIC program, always erase the BG GRAPHIC (background) screen beforehand. Not doing this might result in an error.
- Press the [CLR HOME] key while holding down the [SHIFT] key to erase the BG GRAPHIC screen. The cursor will return to its home position.
- Call the program with LIST and execute the changes and modifications.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
0	D50	D50	D50	D50	D50	D50	D50	D50	D50	D50	D50	D50	D50	D50	D50	D50	D50	D50	D50	D50	D50	D50	D50	D50	D50	D50	D50	D50
1	D52	D52	D52	D52	D52	D52	D52	D52	D52	D52	D52	D52	D52	D52	D52	D52	D52	D52	D52	D52	D52	D52	D52	D52	D52	D52	D52	D52
2	G42	G42	G42	G42	G42	G42	G42	G42	G42	G42	G42	G42	G42	G42	G42	G42	G42	G42	G42	G42	G42	G42	G42	G42	G42	G42	G42	G42
3	G52	G52	G52	G52	G52	G52	G52	G52	G52	G52	G52	G52	G52	G52	G52	G52	G52	G52	G52	G52	G52	G52	G52	G52	G52	G52	G52	G52
4	G52	G52	G52	G52	G52	G52	G52	G52	G52	G52	G52	G52	G52	G52	G52	G52	G52	G52	G52	G52	G52	G52	G52	G52	G52	G52	G52	G52
5	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K52	K52	K52	K22
6				J20																						J22		K62
7	1			J20																						J22		K62
8	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K52	K52	K52	K02
9				J20																						J22		K62
10	2			J20																						J22		K62
11	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K52	K52	K52	K02
12				J20																						J22		K62
13	3			J20																						J22		K62
14	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K52	K52	K52	K02
15				J20																						J22		K62
16	4			J20																						J22		K62
17	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K52	K52	K52	K02
18				J20																						J22		K62
19	5			J20																						J22		K62
20	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K50	K52	K52	K52	K12

Sample program 7

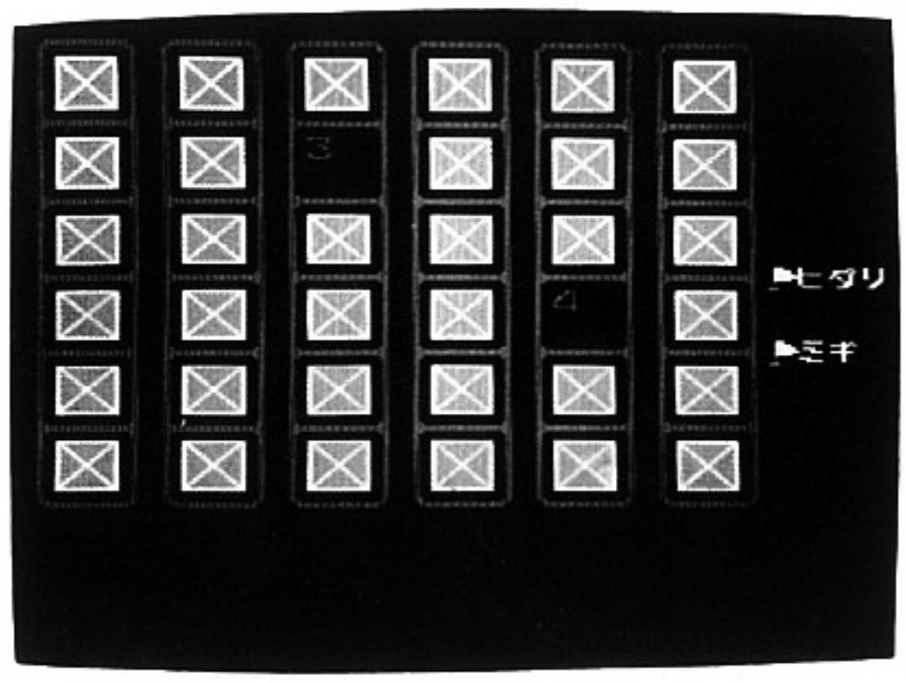
```
** EXERCISE 7 **

10 VIEW:CGEN 3:CGSET 1,1:SPRITE ON
20 I=J:A=C:S=T:X=-1:Y=-1:ED=P:CX=XY:T0=T1:TT=0
30 N$=CHR$(254):N$=N$+N$+N$+N$:M0$=CHR$(243)+CHR$(247):
   M1$=CHR$(245)+CHR$(248)
40 DIM D(5,5),PT(17),PR(1)
50 FOR I=0 TO 5:FOR J=0 TO 5:
   LOCATE I*4+1,J*3+1:PRINT M0$:
   LOCATE I*4+1,J*3+2:PRINT M1$
60 A=RND(18):IF PT(A)=2 THEN 60
70 D(I,J)=A:PT(A)=PT(A)+1:NEXT:NEXT
80 PALETS 0,13,&H12,&H22,2:PALETS 1,13,&H14,&H24,4:
   PALETS 2,13,&H16,&H26,6:PALETS 3,13,2,25,&H36
90 DEF SPRITE 0,(3,1,0,0,0)=N$:DEF SPRITE 5,(3,1,0,0,0)=N$
100 LOCATE 25,9:PRINT"LEFT ":LOCATE 25,12:PRINT"RIGHT"
110 SPRITE 0,CX*32+24,CY*24+31:T0=-1:T1=-1:TT=7:GOSUB 170:
    SWAP T0,T1:X=CX:Y=CY:TT=6:GOSUB 170
120 IF T0<>T1 THEN 150
130 PLAY"T204C1E1G1O5C6":ED=ED+1:IF ED=18 THEN 300
140 PR(P)=PR(P)+10:LOCATE 23,10+P*3:PRINT PR(P):SPRITE 6:
    SPRITE 7:GOTO 110
150 PLAY"O1T2D5E5C5"
160 D(X,Y)=T1:D(CX,CY)=T0:LOCATE CX*4+1,CY*3+1:PRINT M0$:
    LOCATE CX*4+1,CY*3+2:PRINT M1$:LOCATE X*4+1,Y*3+1:PRINT M0$:
    LOCATE X*4+1,Y*3+2:PRINT M1$:P=1-P:GOTO 110
170 SPRITE 5,216,95+24*P:PAUSE 10:SPRITE 5:S=STICK(P):T=STRIG(P)
180 IF S=1 THEN CX=CX+1:IF CX>5 THEN CX=0
190 IF S=2 THEN CX=CX-1:IF CX<0 THEN CX=5
200 IF S=4 THEN CY=CY+1:IF CY>5 THEN CY=0
210 IF S=8 THEN CY=CY-1:IF CY<0 THEN CY=5
220 IF S=0 THEN 240
230 SPRITE 0,CX*32+24,CY*24+31
240 IF T<8 THEN 170
250 SWAP D(CX,CY),T0
260 IF T0=-1 THEN SWAP D(CX,CY),T0:GOTO 170
270 SPRITE 0:PLAY"O5T1C1C1":PAUSE 10:LOCATE CX*4+1,CY*3+1:
    PRINT"  ":LOCATE CX*4+1,CY*3+2:PRINT"  "
280 DEF SPRITE TT,(T0/6,1,1,0,0)=CHR$(48+T0-T0/6*6):
    SPRITE TT,CX*32+24,CY*24+31
290 RETURN
300 PRINT"TRY AGAIN !!"
310 T=STRIG(0):IF T=0 THEN 310
320 IF T=1 THEN RUN
330 IF T=2 THEN END
340 GOTO 300
350 END
```

〈CARD〉

It's a card memorizing game. You can either memorize the numbers and colors of all the cards which you have flipped or challenge your sixth sense. Which one will you do?

How to play:

2 players. Select the cards with the button of your controller and use the A button to flip the cards. Flip 2 cards and if they have the same color and number, you win. Then you can continue flipping cards. If the cards are not the same, it's the other one's turn.



★ Warning: When changing or modifying the program

- When creating, changing or modifying a BASIC program, always erase the BG GRAPHIC (background) screen beforehand. Not doing this might result in an error.
- Press the [CLR HOME] key while holding down the [SHIFT] key to erase the BG GRAPHIC screen. The cursor will return to its home position.
- Call the program with LIST and execute the changes and modifications.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
0	K72	K52	K52	L02	K72	K52	K52	L02	K72	K52	K52	L02	K72	K52	K52	L02	K72	K52	K52	L02	K72	K52	K52	L02				
1	K62			K62	K62			K62	K62			K62	K62			K62	K62			K62	K62			K62				
2	K62			K62	K62			K62	K62			K62	K62			K62	K62			K62	K62			K62				
3	K42	K52	K52	K32	K42	K52	K52	K32	K42	K52	K52	K32	K42	K52	K52	K32	K42	K52	K52	K32	K42	K52	K52	K32				
4	K62			K62	K62			K62	K62			K62	K62			K62	K62			K62	K62			K62				
5	K62			K62	K62			K62	K62			K62	K62			K62	K62			K62	K62			K62				
6	K42	K52	K52	K32	K42	K52	K52	K32	K42	K52	K52	K32	K42	K52	K52	K32	K42	K52	K52	K32	K42	K52	K52	K32				
7	K62			K62	K62			K62	K62			K62	K62			K62	K62			K62	K62			K62				
8	K62			K62	K62			K62	K62			K62	K62			K62	K62			K62	K62			K62				
9	K42	K52	K52	K32	K42	K52	K52	K32	K42	K52	K52	K32	K42	K52	K52	K32	K42	K52	K52	K32	K42	K52	K52	K32	F72			
10	K62			K62	K62			K62	K62			K62	K62			K62	K62			K62	K62			K62				
11	K62			K62	K62			K62	K62			K62	K62			K62	K62			K62	K62			K62				
12	K42	K52	K52	K32	K42	K52	K52	K32	K42	K52	K52	K32	K42	K52	K52	K32	K42	K52	K52	K32	K42	K52	K52	K32	F72			
13	K62			K62	K62			K62	K62			K62	K62			K62	K62			K62	K62			K62				
14	K62			K62	K62			K62	K62			K62	K62			K62	K62			K62	K62			K62				
15	K42	K52	K52	K32	K42	K52	K52	K32	K42	K52	K52	K32	K42	K52	K52	K32	K42	K52	K52	K32	K42	K52	K52	K32				
16	K62			K62	K62			K62	K62			K62	K62			K62	K62			K62	K62			K62				
17	K62			K62	K62			K62	K62			K62	K62			K62	K62			K62	K62			K62				
18	L12	K52	K52	L22	L12	K52	K52	L22	L12	K52	K52	L22	L12	K52	K52	L22	L12	K52	K52	L22	L12	K52	K52	L22				
19																												
20																												

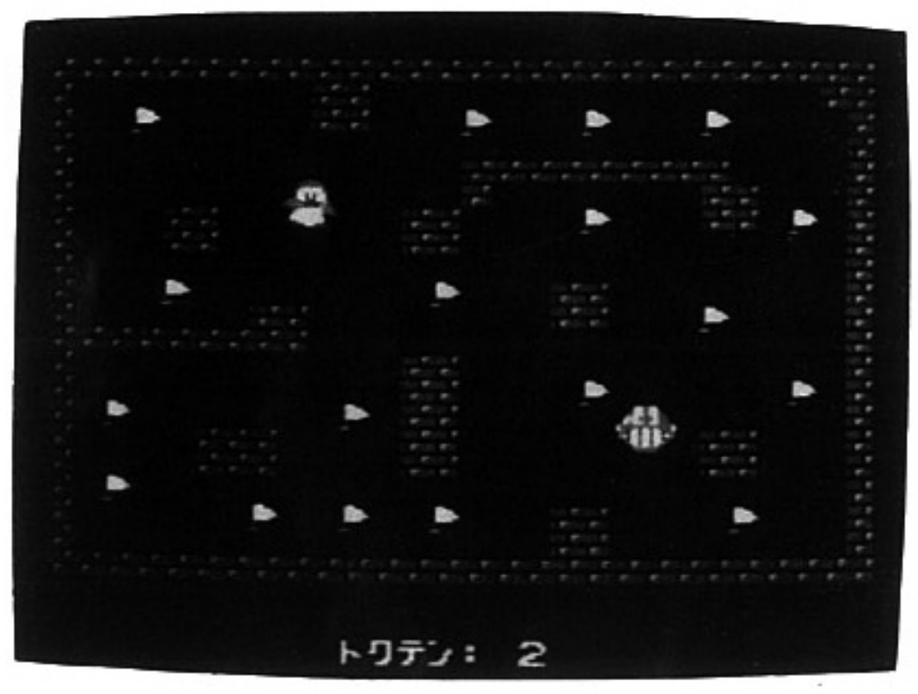
Sample program 8

```
** EXERCISE 8 **

10 VIEW
20 PLAY"O4C1D1A1G1E1B"
30 SPRITE ON
40 CGSET 1,0
50 PX=50:PY=56:MX=190:MY=150:DEF MOVE(0)=SPRITE(4,D,1,1 ,
0,0):POSITION 0,PX,PY
60 DX=PX-MX:DY=PY-MY
70 IF ABS(DX)<8 AND ABS(DY)<8 THEN 360
80 S1=-1*(DX>0)-2*(DX<0):S2=-4*(DY>0)-8*(DY<0)
90 IF ABS(DX) < ABS(DY) THEN SWAP S1,S2
100 S=S1:GOSUB 270:GOSUB 290
110 IF D <> 0 THEN 140
120 SWAP S1,S2:S=S1:GOSUB 270:GOSUB 290
130 IF D=0 THEN 160
140 DEF MOVE(1)=SPRITE(11,D,1,3,0,0):POSITION 1,MX,MY
150 MOVE 1:PLAY"O1C1C1C1"
160 S0=STICK(0)
170 S=S0:GOSUB 280:GOSUB 290
180 IF D=0 THEN 250
190 DEF MOVE(0)=SPRITE(4,D,1,3,0,0)
200 POSITION 0,PX,PY
210 MOVE0:PLAY"O3B1D1"
220 XX=(PX+7)/8-2:YY=(PY+7)/8-3
230 IF SCR$(XX,YY)=CHR$(199) THEN LOCATE XX,YY:PRINT
" ";:CN=CN+1:LOCATE 10,23:PRINT"SCORE:";CN;:PLAY"O4
C1A1G1"
240 IF MOVE(0)=-1 OR MOVE(1)=-1 THEN 240
250 PX=XPOS(0):PY=YPOS(0):MX=XPOS(1):MY=YPOS(1)
260 GOTO 60
270 X=MX-(S=1)*4+(S=2)*4:Y=MY-(S=4)*4+(S=8)*4:RETURN
280 X=PX-(S=1)*4+(S=2)*4:Y=PY-(S=4)*4+(S=8)*4:RETURN
290 C1=(X-1)/8-2:L1=(Y-1)/8-3
300 C2=X+16:C2=(C2-1)/8-2:L2=Y+16:L2=(L2-1)/8-3
310 D= -3*(S=1)*(SCR$(C2,L1)=" ")*(SCR$(C2,L2)=" ")
320 D=D-7*(S=2)*(SCR$(C1,L1)=" ")*(SCR$(C1,L2)=" ")
330 D=D-1*(S=8)*(SCR$(C1,L1)=" ")*(SCR$(C2,L1)=" ")
340 D=D-5*(S=4)*(SCR$(C1,L2)=" ")*(SCR$(C2,L2)=" ")
350 RETURN
360 PLAY"O4G1C1G1":FOR Q=0 TO 3:CGSET0,0:CGSET1,1:CGSET,0:NEXT
370 PLAY"O1C1G1A1C1D1":CLS:SPRITE OFF
380 LOCATE 5,10:PRINT"----- END -----":END
```

This is a sample which uses SCR$.
(It is not a game)

- Please execute it after drawing the background first.
- Control Penguin with the ◀ ▶ ▲ ▼ keys.
- Because Smiley will try to get close to Penguin, you should try to pass through the place with the flag while running away.
- When passing through a flag, control Penguin as if its center superimposes with the upper part of the flag.
- Penguin and Smiley will be unable to move on if they bump into the bricks.

※ Please change the position of the flags or the bricks in BG GRAPHIC and change the characters to try out several patterns.



★Warning: When changing or modifying the program

- When creating, changing or modifying a BASIC program, always erase the BG GRAPHIC (background) screen beforehand. Not doing this might result in an error.
- Press the CLR HOME key while holding down the SHIFT key to erase the BG GRAPHIC screen. The cursor will return to its home position.
- Call the program with LIST and execute the changes and modifications.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
0	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32
1	F32									F32	F32																F32	F32
2	F32			F72						F32	F32				F72				F72				F72					F32
3	F32																											F32
4	F32														F32	F32	F32	F32	F32	F32	F32	F32	F32					F32
5	F32								F72						F32								F32	F32				F32
6	F32				F32	F32							F32	F32					F72				F32	F32		F72		F32
7	F32				F32	F32							F32	F32														F32
8	F32																											F32
9	F32				F72									F72				F32	F32									F32
10	F32							F32	F32									F32	F32				F72					F32
11	F32	F32	F32	F32	F32	F32	F32	F32	F32																			F32
12	F32												F32	F32														F32
13	F32												F32	F32					F72							F72		F32
14	F32		F72								F72		F32	F32														F32
15	F32					F32	F32	F32					F32	F32									F32	F32				F32
16	F32					F32	F32	F32					F32	F32									F32	F32				F32
17	F32		F72																									F32
18	F32							F72			F72			F72				F32	F32					F72				F32
19	F32																	F32	F32									F32
20	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32	F32

MEMO

APPENDIX

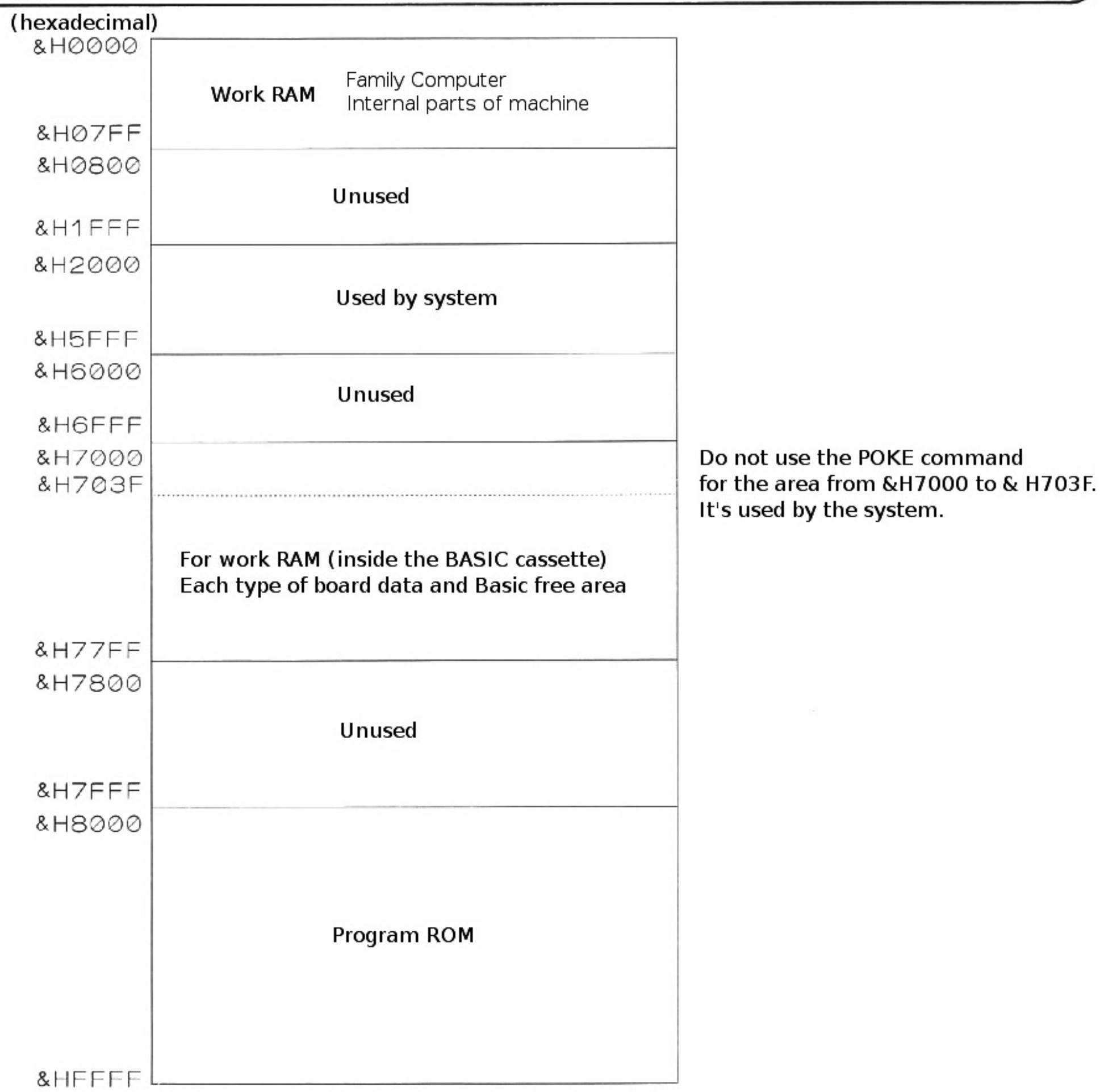
CONTROL CODES

In NS-HUBASIC there are many key functions which are ready to be used besides the regular operation keys.

CTR+	Processing contents	Reference
A	INS mode's ON/OFF switch.	
C	BREAK. Can't be used while executing a program.	
D	Init. settings※CGEN2, SPRITE OFF and releasing CTR+A. Turns the color palet of Sprite and Background into palet code 1 for background.	
E	Erases one line after the cursor.	
G	Emits a BEEP sound.	
H	Same function as DEL.	DEL
J	Line feed. Goes to the next line. Same as the ▼ key.	
K	Returns the cursor to the home position.	SHIFT+CLR HOME
L	Screen clear.	
M	Enters one line and goes to the next line. (carriage return)	RETURN
R	Same function as INS.	INS
V	Turns to Kana mode.	
W	Turns to alphanumeric status.	
Z	Clears from the cursor to the end of the screen.	

※CTR+A means pressing the A key while holding down the CTR key.

MEMORY MAP



LIST OF ERROR MESSAGES

When NS-HUBASIC finds an error while executing a program, it displays an error message on screen, halts the execution of the program and goes into a command waiting status.

When executing in direct mode
(example) ?SN ERROR (syntax)
When executing in program mode
(example) ?SN ERROR IN 100
is displayed.

SN···error message
100 ···line number which contains the error

Please refer to the list below when you encounter an error message while executing a program for which you do not know the cause.

Error Code	Error Message	Explanation
NF	NEXT without FOR	There's a NEXT but no FOR.
SN	Syntax error	The grammar is wrong.
RG	RETURN without GOSUB	There's a RETURN but there's no GOSUB.
OD	Out of DATA	The data that is supposed to be read by READ can't be found within the DATA sentence.
I L	Illegal function call	The statement or function call is wrong.
OV	Overflow	The result of the calculus has transgressed the limit of the allowed scope.
OM	Out of memory	There's not enough memory.
UL	Undefined line Number	The line number specified by GOTO, GOSUB, IF etc. does not exist.
SO	Subscript out of range	The subscript of the array variable is out of range.
DD	Duplicate Definition	The array has been defined twice.
DZ	Division by zero	Division by zero.
TM	Type mismatch	Mismatch of type of variable.
ST	String too long	The characters contain more than 31 characters.
FT	Formula Too complex	The formula is too complex. For example, there are too many brackets.
CC	Can't continue	Can't continue the execution of the program with CONT.
MO	Missing operand	There's no designation towards the necessary command of the parameter.
TP	Tape read ERROR	Can't read correctly the data from the cassette tape.

CHARACTER CODE LIST

This page shows the Family Basic character codes in a list. The BASIC character and code conversion according to CHR$ (n) and ASC ("A$") matches this list. You can use character code list A and B's characters, symbols in the background and sprite screens. (Use the CGEN command to specify which list you will use in which screen.)

●Character code list A (mainly for sprites)

The (decimal) code matches the numbers in the four corners of the animated characters in character table A.

Code (decimal)	Code (hexa-decimal)	Explanation	Code (decimal)	Code (hexa-decimal)	Explanation	Code (decimal)	Code (hexa-decimal)	Explanation	Code (decimal)	Code (hexa-decimal)	Explanation
0	00		32	20		64	40		96	60	
1	01	Mario (WALK1)	33	21	Lady (WALK2)	65	41	Achiles (Left1)	97	61	Penguin (Walk left1)
2	02		34	22		66	42		98	62	
3	03		35	23		67	43		99	63	
4	04		36	24		68	44		100	64	
5	05	Mario (WALK2)	37	25	Lady (WALK3)	69	45	Achiles (Left2)	101	65	Penguin (Walk left2)
6	06		38	26		70	46		102	66	
7	07		39	27		71	47		103	67	
8	08		40	28		72	48		104	68	
9	09	Mario (WALK3)	41	29	Lady (JUMP)	73	49	Achiles (Left up1)	105	69	Penguin (Front)
10	0A		42	2A		74	4A		106	6A	
11	0B		43	2B		75	4B		107	6B	
12	0C		44	2C		76	4C		108	6C	
13	0D	Mario (JUMP)	45	2D	Lady (Slip Land)	77	4D	Achiles (Left up2)	109	6D	Penguin (Back)
14	0E		46	2E		78	4E		110	6E	
15	0F		47	2F		79	4F		111	6F	
16	10		48	30		80	40		112	70	
17	11	Mario (SLIP LAND)	49	31	Lady (Ladder)	81	51	Achiles (Up1)	113	71	Fire Ball (1)
18	12		50	32		82	52		114	72	
19	13		51	33		83	53		115	73	
20	14		52	34		84	54		116	74	
21	15	Mario (Ladder)	53	35	Lady (DOWN)	85	55	Achiles (Up2)	117	75	Fire Ball (2)
22	16		54	36		86	56		118	76	
23	17		55	37		87	57		119	77	
24	18		56	38		88	58		120	78	
25	19	Mario (DOWN)	57	39	Fighter Fly (1)	89	59	Smiley (1)	121	79	Car (Left1)
26	1A		58	3A		90	5A		122	7A	
27	1B		59	3B		91	5B		123	7B	
28	1C		60	3C		92	5C		124	7C	
29	1D	Lady (WALK1)	61	3D	Fighter Fly (2)	93	5D	Smiley (2)	125	7D	Car (Left2)
30	1E		62	3E		94	5E		126	7E	
31	1F		63	3F		95	5F		127	7F	

- Use the codes (decimal or hexadecimal) in the code list to specify the characters which you want to define.
- You can also use this chart as a conversion table for decimals from 0 to 255 into hexadecimals.

Code (decimal)	Code (hexa-decimal)	Explanation
128	80	
129	81	Car (up left1)
130	82	
131	83	
132	84	
133	85	Car (up left2)
134	86	
135	87	
136	88	
137	89	Car (up1)
138	8A	
139	8B	
140	8C	
141	8D	Car (up2)
142	8E	
143	8F	
144	90	
145	91	Spinner (1)
146	92	
147	93	
148	94	
149	95	Spinner (2)
150	96	
151	97	
152	98	
153	99	Star Killer (left)
154	9A	
155	9B	
156	9C	
157	9D	Star Killer (left up)
158	9E	
159	9F	

Code (decimal)	Code (hexa-decimal)	Explanation
160	A0	
161	A1	Star Killer (up)
162	A2	
163	A3	
164	A4	
165	A5	Star Ship (left)
166	A6	
167	A7	
168	A8	
169	A9	Starship (left up)
170	AA	
171	AB	
172	AC	
173	AD	Star Ship (up)
174	AE	
175	AF	
176	B0	
177	B1	Explosion (1)
178	B2	
179	B3	
180	B4	
181	B5	Explosion (2)
182	B6	
183	B7	
184	B8	
185	B9	Shell Creeper (1)
186	BA	
187	BB	
188	BC	
189	BD	Shell Creeper (2)
190	BE	
191	BF	

Code (decimal)	Code (hexa-decimal)	Explanation
192	C0	
193	C1	Side Stepper (1)
194	C2	
195	C3	
196	C4	
197	C5	Side Stepper (2)
198	C6	
199	C7	
200	C8	
201	C9	Knitpicker (1)
202	CA	
203	CB	
204	CC	
205	CD	Knitpicker (2)
206	CE	
207	CF	
208	D0	Laser (horizontal)
209	D1	
210	D2	Laser (diagonal)
211	D3	
212	D4	Laser (vertical)
213	D5	
214	D6	Sprite paint colors (1, 2, 3)
215	D7	
216	D8	
217	D9	
218	DA	Music board cursor (1, 2, 3)
219	DB	
220	DC	1st cursor
221	DD	2nd cursor
222	DE	Lamp (1)
223	DF	Lamp (2)

Code (decimal)	Code (hexa-decimal)	Explanation
224	E0	
225	E1	
226	E2	
227	E3	Quill (1)
228	E4	
229	E5	
230	E6	
231	E7	
232	E8	
233	E9	
234	EA	
235	EB	Quill (2)
236	EC	
237	ED	
238	EE	
239	EF	
240	F0	1
241	F1	2
242	F2	3
243	F3	4
244	F4	
245	F5	COMPUTER characters
246	F6	
247	F7	
248	F8	
249	F9	
250	FA	
251	FB	OPERATOR characters
252	FC	
253	FD	
254	FE	
255	FF	

● Character code list B (mainly characters which you can use on the background screen, as well as characters which you can use as keyboard characters, symbols and for BG GRAPHIC)
The number written in the column between ( ) matches the character in character table B. (p. 113)

Code (decimal)	Code (hexa-decimal)	Matching character	Code (decimal)	Code (hexa-decimal)	Matching character	Code (decimal)	Code (hexa-decimal)	Matching character	Code (decimal)	Code (hexa-decimal)	Matching character
0	00	(A0)	32	20		64	40	@	96	60	ア
1	01	(A1)	33	21	!	65	41	A	97	61	イ
2	02	(A2)	34	22	"	66	42	B	98	62	ウ
3	03	(A3)	35	23	#	67	43	C	99	63	エ
4	04	(A4)	36	24	$	68	44	D	100	64	オ
5	05	(A5)	37	25	%	69	45	E	101	65	カ
6	06	(A6)	38	26	&	70	46	F	102	66	キ
7	07	(A7)	39	27	'	71	47	G	103	67	ク
8	08	(B0)	40	28	(	72	48	H	104	68	ケ
9	09	(B1)	41	29	)	73	49	I	105	69	コ
10	0A	(B2)	42	2A	*	74	4A	J	106	6A	サ
11	0B	(B3)	43	2B	+	75	4B	K	107	6B	シ
12	0C	(B4)	44	2C	,	76	4C	L	108	6C	ス
13	0D	(B5)	45	2D	-	77	4D	M	109	6D	セ
14	0E	(B6)	46	2E	.	78	4E	N	110	6E	ソ
15	0F	(B7)	47	2F	/	79	4F	O	111	6F	タ
16	10	(C0)	48	30	0	80	50	P	112	70	チ
17	11	(C1)	49	31	1	81	51	Q	113	71	ツ
18	12	(C2)	50	32	2	82	52	R	114	72	テ
19	13	(C3)	51	33	3	83	53	S	115	73	ト
20	14	(C4)	52	34	4	84	54	T	116	74	ナ
21	15	(C5)	53	35	5	85	55	U	117	75	ニ
22	16	(C6)	54	36	6	86	56	V	118	76	ヌ
23	17	(C7)	55	37	7	87	57	W	119	77	ネ
24	18	(D0)	56	38	8	88	58	X	120	78	ノ
25	19	(D1)	57	39	9	89	59	Y	121	79	ハ
26	1A	(D2)	58	3A	:	90	5A	Z	122	7A	ヒ
27	1B	(D3)	59	3B	;	91	5B	「	123	7B	フ
28	1C	(D4)	60	3C	<	92	5C	¥	124	7C	ヘ
29	1D	(D5)	61	3D	=	93	5D	」	125	7D	ホ
30	1E	(D6)	62	3E	>	94	5E	^	126	7E	マ
31	1F	(D7)	63	3F	?	95	5F	-	127	7F	ミ

● Please use the codes in this list (decimal, hexadecimal) when defining characters directly for the background.
However, the hexadecimal codes from 00 to 1F may not be specified directly because they are used by the system.
●You can also use this table as a conversion table between decimals from 0 to 255 and hexadecimals.

Code (decimal)	Code (hexa-decimal)	Matching character	Code (decimal)	Code (hexa-decimal)	Matching character	Code (decimal)	Code (hexa-decimal)	Matching character		Code (decimal)	Code (hexa-decimal)	Matching character	
128	80	ム	160	A0	ゾ	192	C0	(F0)	MARIO	224	E0	(J0)	MAZE
129	81	メ	161	A1	ダ	193	C1	(F1)		225	E1	(J1)	
130	82	モ	162	A2	ヂ	194	C2	(F2)		226	E2	(J2)	
131	83	ヤ	163	A3	ヅ	195	C3	(F3)		227	E3	(J3)	
132	84	ユ	164	A4	デ	196	C4	(F4)		228	E4	(J4)	
133	85	ヨ	165	A5	ド	197	C5	(F5)		229	E5	(J5)	
134	86	ラ	166	A6	バ	198	C6	(F6)		230	E6	(J6)	
135	87	リ	167	A7	ビ	199	C7	(F7)		231	E7	(J7)	
136	88	ル	168	A8	ブ	200	C8	(G0)	LANDSCAPE	232	E8	(K0)	RULER FOR GRAPH
137	89	レ	169	A9	ベ	201	C9	(G1)		233	E9	(K1)	
138	8A	ロ	170	AA	ボ	202	CA	(G2)		234	EA	(K2)	
139	8B	ワ	171	AB	パ	203	CB	(G3)		235	EB	(K3)	
140	8C	ン	172	AC	ピ	204	CC	(G4)		236	EC	(K4)	
141	8D	ヲ	173	AD	プ	205	CD	(G5)Star (1)		237	ED	(K5)	
142	8E	ァ	174	AE	ペ	206	CE	(G6)Star (2)		238	EE	(K6)	
143	8F	ィ	175	AF	ポ	207	CF	(G7)Ball		239	EF	(K7)	
144	90	ゥ	176	B0	◻	208	D0	(H0)	DONKEYKONG Jr.	240	F0	(L0)	
145	91	ェ	177	B1	◦	209	D1	(H1)		241	F1	(L1)	
146	92	ォ	178	B2	[	210	D2	(H2)		242	F2	(L2)	
147	93	ャ	179	B3	]	211	D3	(H3)		243	F3	(L3)	BOARD FRAME
148	94	ュ	180	B4	©	212	D4	(H4)		244	F4	(L4)	
149	95	ョ	181	B5	×	213	D5	(H5)		245	F5	(L5)	
150	96	ッ	182	B6	÷	214	D6	(H6)		246	F6	(L6)	
151	97	ガ	183	B7	ファ	215	D7	(H7)		247	F7	(L7)	
152	98	ギ	184	B8	(E0)	216	D8	(I0)	ISLAND	248	F8	(M0)	INKPOT
153	99	グ	185	B9	(E1)	217	D9	(I1)		249	F9	(M1)	
154	9A	ゲ	186	BA	(E2)	218	DA	(I2)		250	FA	(M2)	
155	9B	ゴ	187	BB	(E3)	219	DB	(I3)		251	FB	(M3)	
156	9C	ザ	188	BC	(E4)	220	DC	(I4)		252	FC	(M4)	
157	9D	ジ	189	BD	(E5)	221	DD	(I5)	MAZE	253	FD	(M5)(1)	PAINT
158	9E	ズ	190	BE	(E6)	222	DE	(I6)		254	FE	(M6)(2)	
159	9F	ゼ	191	BF	(E7)	223	DF	(I7)		255	FF	(M7)(3)	

Alphabetically ordered commands

〔 〕shows the abbreviations of commands.

Warning about handling

Connection cables	Be careful when leaving the connection cable of the keyboard under a table or a chair and avoid damaging it by leaving it squeezed between objects. Using a damaged cable is dangerous. Hold the connector when removing the cable.
Power- supply voltage	Use a power-supply of AC100V for the Family Computer. If the voltage is too high or too low, it might cause a malfunction or might not enable all of the functions. In such a case you should contact the store were you purchased it or one of Nintendo's stores.
Moisture and dust	Do not leave this equipment in places with lots of moisture or dust. This might lead to malfunction. Please avoid using it in places where it might absorb a lot of dust.
High temperature	Do not place this equipment in direct sunlight or close to a heater. This might cause damage to the external or internal parts.
Water and foreign substances	Using this equipment while water or liquids or even metallic objects such as needles or pins are inside is dangerous. Do not let foreign objects get inside. If water, liquids or foreign objects do get inside, disconnect the keyboard immediately and bring it to the store where it was purchased.
Shocks	This equipment is made of precise electronic parts. Do not drop it or let it bump into other objects. This might lead to malfunction.
Malfunction	In case of malfunction or defect, stop using it and bring it to the store where it was purchased or one of Nintendo's stores.
When not using it for an extended period of time	When not using it for an extended period of time, please remove the BASIC cassette and the keyboard's connector from the Family Computer.
Addition of accessories	Using something besides a mono radio cassette deck might cause malfunction.
Dirt	This equipment can be cleaned with a soft cloth and water or by adding some cleaning product and wiping lightly. Please avoid using volatiles such as benzin or thinner which might change the color of the exterior.
Power ON - OFF	Please leave at least 10 seconds between turning the switch of the Family Computer ON and OFF. This is in order to ensure the working of the computer. Also avoid removing the cassette or the keyboard while the switch is ON as it may lead to malfunction.

Warning about battery-handling

2 AA batteries are necessary for memory backup. A wrong handling of the batteries might lead to leak or explosion. Please keep the following points in mind.

1. Insert the batteries according to the + and - signs as specified on the BASIC cassette.
2. Do not mix fresh and already used batteries.
3. Do not mix batteries of different types. Even if they have the same size, the voltage might differ.
4. Please remove the batteries when you stop using them or when you do not use the BASIC cassette for an extended period of time.

(cut here)

CUT

FAMILY BASIC Warranty

CUSTOMER	Address TEL.	PURCHASE	Address
	Name		Name of store seal
Date of purchase	Year Month Day	Model HVC-007	Serial number

- Under normal conditions of use, covering 6 months from the date of purchase and based on the rules written on the back of the warranty, repair costs will be free of charge.
- Please contact the store of purchase in case of malfunction. Also, please check if the store of purchase and date are written above. This document shall not be reissued.

※ This warranty does not include the warranty of the "cassette".

Nintendo, co. ltd.

About warranty and servicing

★About warranty

The warranty card guarantees the quality of this product manufactured by Nintendo and that in case of malfunction, all repairs shall be free of charge within the time limit of the warranty. In case your warranty card has not been filled, please write down your name and address and go to the place of purchase to request the filling of the card. (The items related to the warranty are written down on the warranty card.) This warranty card does not include the cassette.

★When requesting servicing

Please contact the place of purchase in case a problem arises.
Furthermore, you should check if the place and date of purchase have been written down on the warranty card.

Family Basic Specifications

- Soft : 1. BASIC (BG GRAPHIC included)
  2. Calculator board
  3. Message board
  4. Computer fortune reading
  5. Music board

Items		Specifications
ROM		Program ROM ····· 8 Bit ×16K×2 Character generator ROM ····· 8 Bit ×8K
RAM		Working or Memory backup S−RAM ····· 8 Bit ×2K
Disp. cap.	Character display	28 characters x 24 lines (during BASIC)
	Character construction	8 x 8 dot matrix characters, alphanumerics, numbers, katakana, English symbols, special symbols and characters
	Color display	52 colors (including black and white)
	Animation (sprites)	8 sprites (however, 1 sprite is made of 16x16 dots or 8x8 dots) 256x240 dots (during BASIC)
Sound		2 square sound wave generators 1 triangle sound wave generator 1 sound effect generators
Key b.	Amount of keys	Total of 72 keys
	Disposition	ASCII disposition compliant (however, includes the 50 katakana syllables)
	Function key	8 keys
	Cursor key	4 keys
Interface		1200 bauds cassette interface integrated 3.5φ READ、WRITE connectors included
Memory backup		Capacity to save temporarily BASIC programs and execution data for each board by using AA batteries inside the BASIC cassette.
DIM	Keyboard (HVC-007)	368(W)×183(D)×54(H)
	Cassette (HVC-FB)	110(W)×110(D)×17(H)
WGT	Keyboard (HVC-007)	956g
	Cassette (HVC-FB)	95g (Batteries excluded)
Usage conditions		Usage temperature from 0 to 40°C, usage level of humidity from 35 to 75%
Accompanying items		Character map

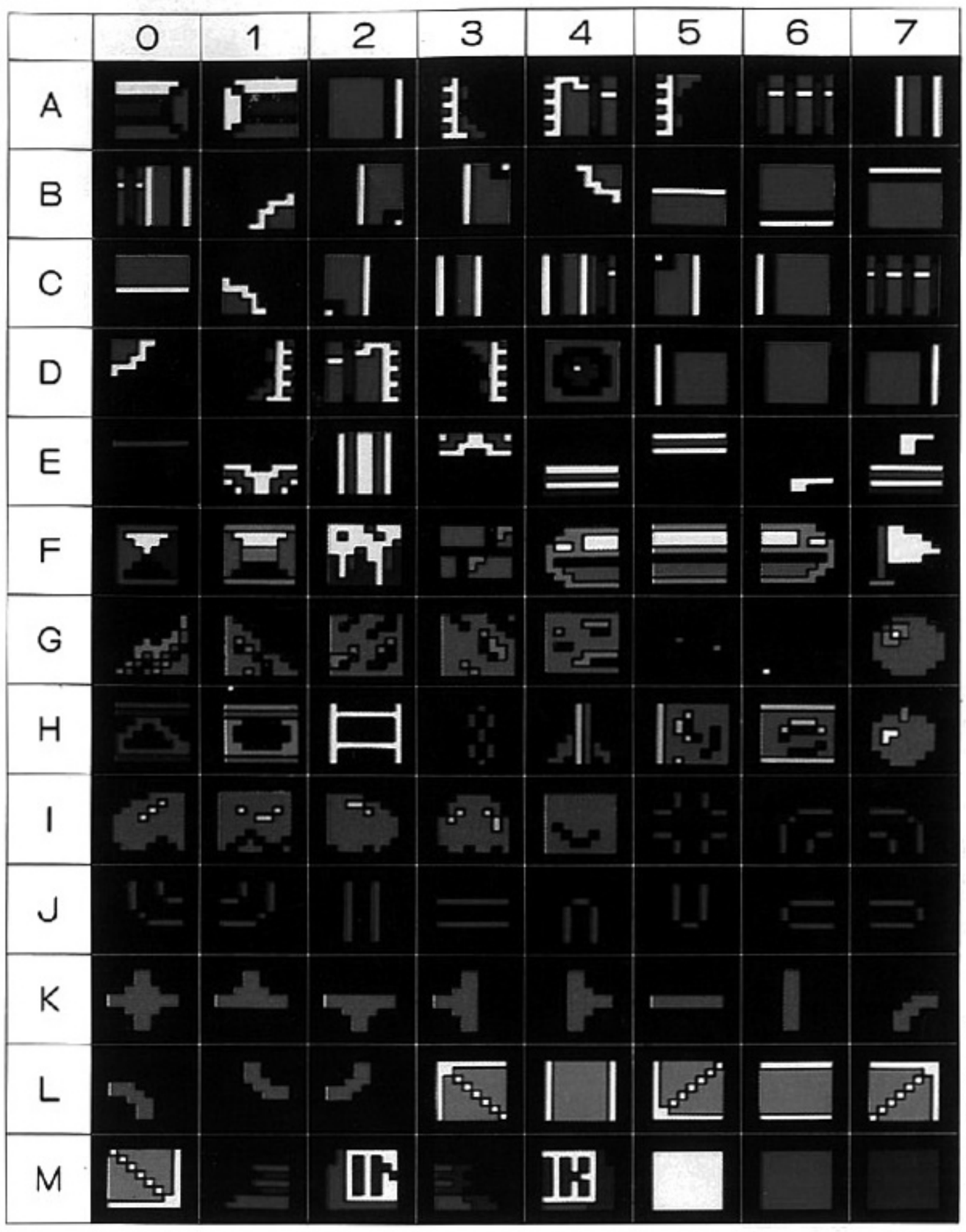
※Please understand that the specifications as well as the appearance might change without prior notice.

Warranty rules

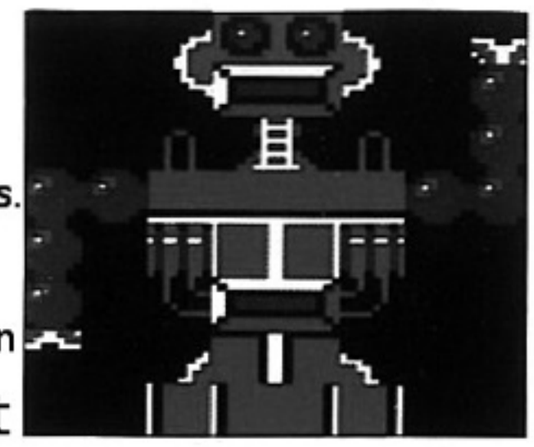
1. A repair fee for a malfunction which arises within the warranty period on the front of this card will be free of charge.
2. The customer shall be charged in the following cases, even if it is within the warranty period.
   ① Malfunction arising from a wrongful usage or unreasonable usage.
   ② Malfunctions due to fire / natural disasters or transportation after purchase.
   ③ Malfunction caused by other products connected to this product.
   ④ If this warranty card has been lost or does not contain the date, place or seal of purchase.
   ⑤ In case this warranty card is not presented to our company.
3. この保証書は国内で使用される場合にだけ有効です。
   This warranty shall be valid only within Japan.

Head quarters: 605 Kyoto, Higashiyama-ku, Fukuine Takamatsu-cho 60 TEL (075)541−6111(Representative)
Tokyo Branch: 101 Tokyo, Chiyoda-ku, Kanda Suda-cho 1-22 TEL (03)254−1781(Representative)
Osaka Branch: 542 Osaka, Minami-ku, Nagatsuka Bashisuji 1-32 TEL (06)271−5514(Representative)
Nagoya Sales Office: 451 Nagoya, Nishi-ku, Habashita 2-18-9 TEL (052)571−2506(Representative)
Sapporo Sales Office: 060 Sapporo, Chuo-ku, Kita 9 Jonishi 18-2 TEL (011)621−0513(Representative)
Okayama Sales Office: 700 Okayama, Hokan-cho 4-4-11 TEL (0862)52−1821(Representative)

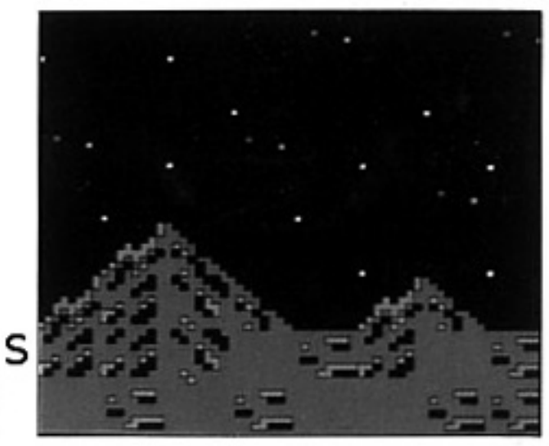
Character Table B



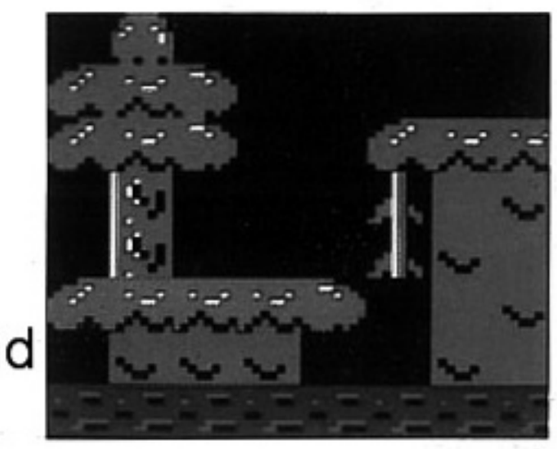
Use the characters from BG GRAPHIC to create many drawings. Look to the right for examples. These are shown as 64x64 dot drawings and are not full-screen images.



Robot



Mountains and stars

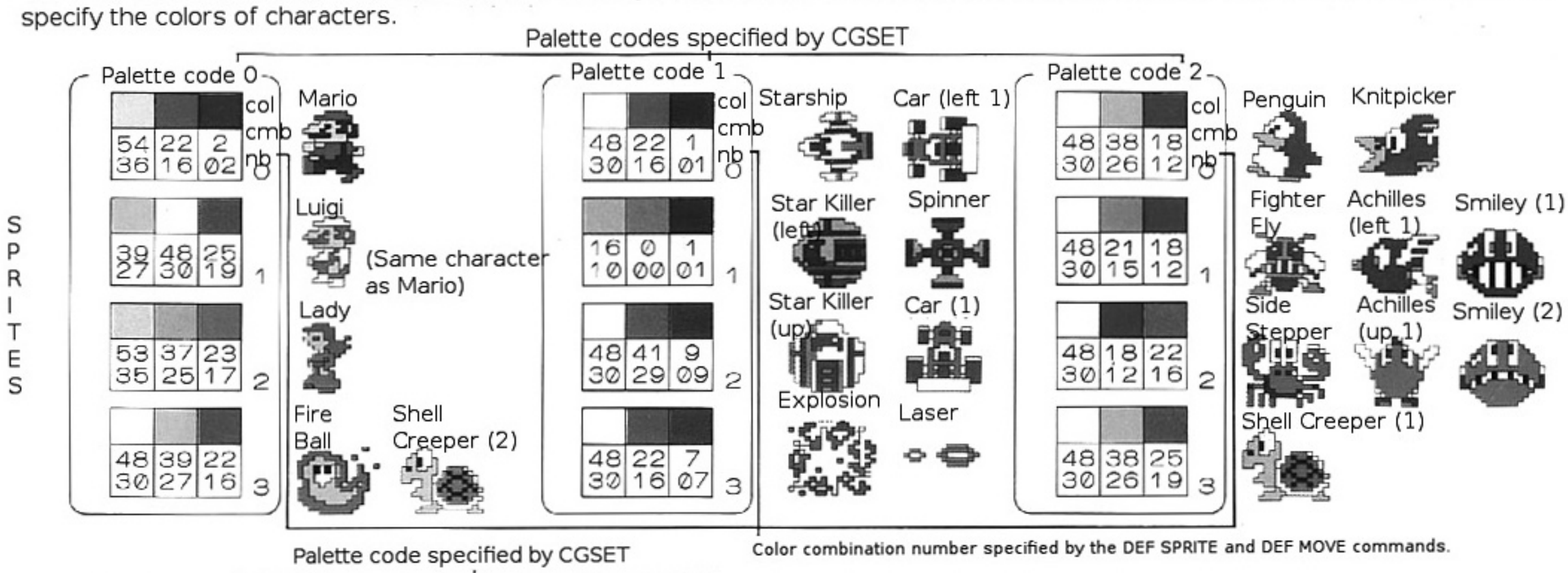


Trees and islands

The characters drawn on character table B show the character group displayed by the SELECT mode of BG GRAPHIC. To change 8 character groups, press the [CLR HOME] key or press the [CLR HOME] key while holding down the [SHIFT] key.

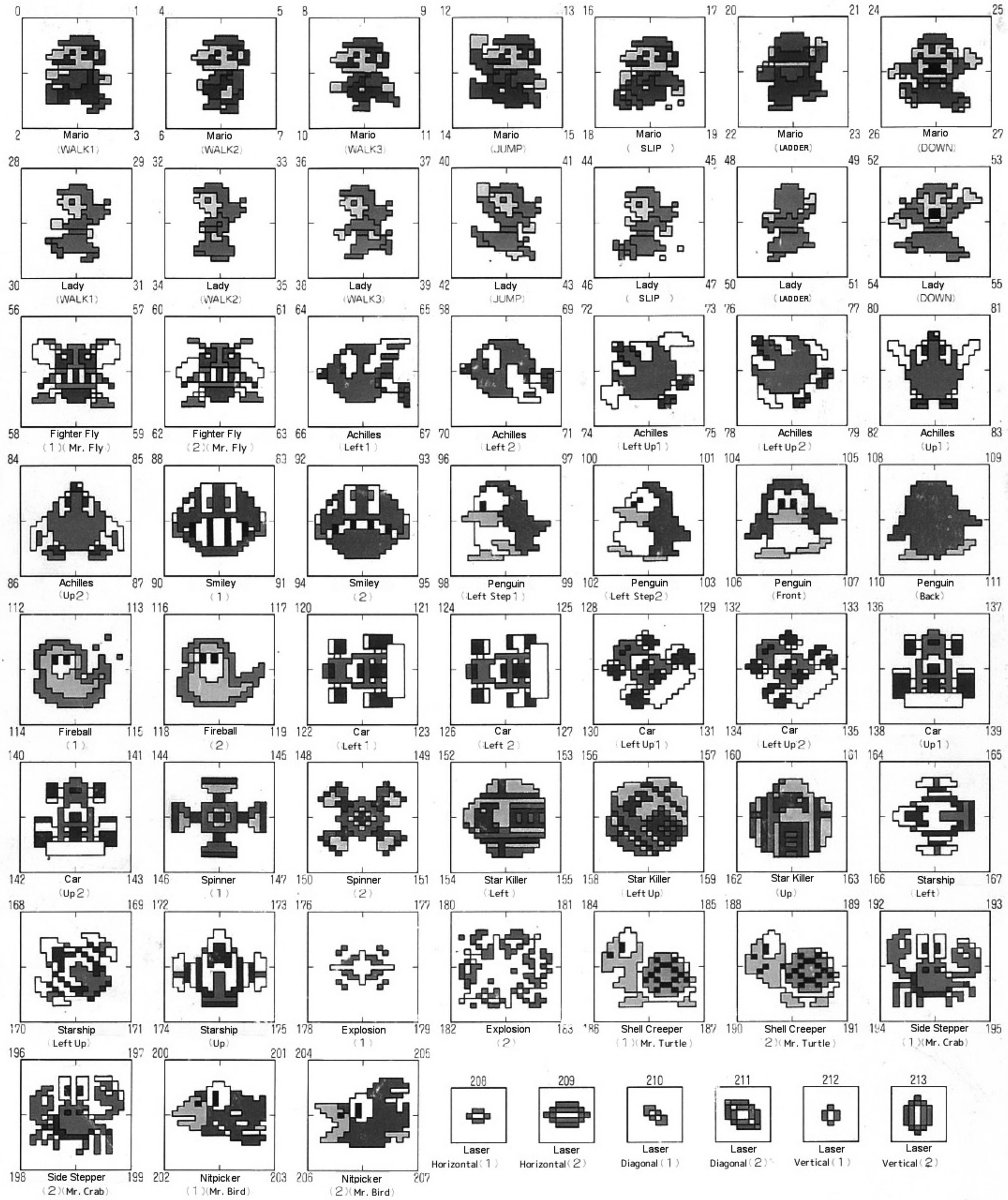
Color Chart

The color chart shows the color combinations which you can specify with the CGSET sentence. Refer to these color combinations to specify the colors of characters.



The numbers (color code) which appear below each color combination are decimals (upper) and hexadecimals (lower). When using the PALET sentence to change the color of an animated character or the color of a background design, you should use one of the colors matching these color codes.
*The color might differ slightly from this print depending on your TV set.

Character Table A



These characters are mainly used as animated characters.
The figures displayed in the four corners of each character are the decimal numbers used for CHR$ (n) of the DEF SPRITE line.

NINTENDO

Head Quarters : 605 Kyoto Higashiyama-ku, Fukuine Takamatsu-cho 60 TEL (075)541-6111 (Representative)

Tokyo Branch : 101 Tokyo Chiyoda-ku, Kanda Suda-cho 1 - 22 TEL (03) 254-1781 (Representative)

Osaka Branch : 542 Osaka, Minami-ku, Nagatsuka Bashisuji 1-32 TEL (06) 271-5514 (Representative)

Nagoya Sales Office : 451 Nagoya, Nishi-ku, Habashita 2-18-9 TEL (052)571-2506 (Representative)

Sapporo Sales Office : 060 Sapporo, Chuo-ku, Kita 9 Jonishi 18-2 TEL (011)621-0513 (Representative)

Okayama Sales Office : 700 Okayama, Hokan-cho 4-4-11 TEL (0862)52-1821 (Representative)