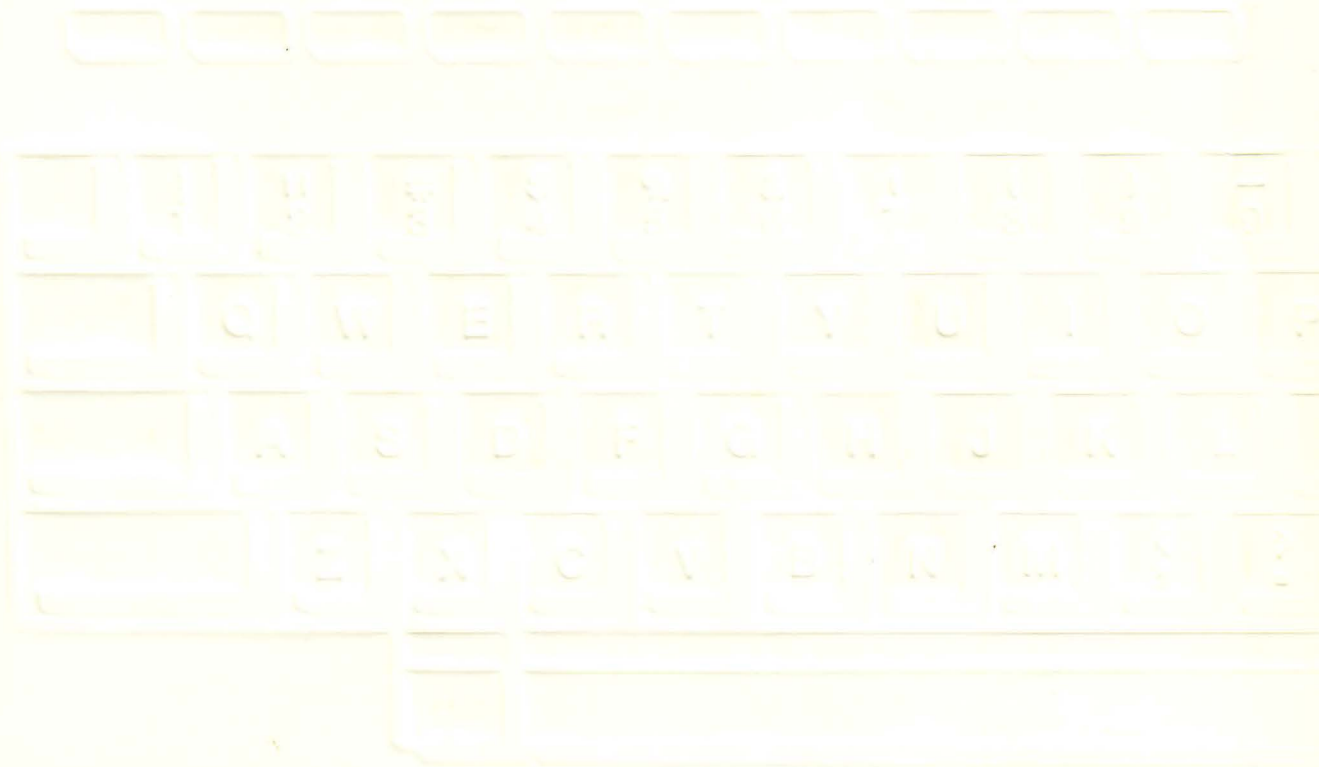


# Personal Computer **MZ-700**

## OWNER'S MANUAL



**SHARP**

### **IMPORTANT**

The wires in this mains lead are coloured in accordance with the following code:

**BLUE:**     **Neutral**  
**BROWN:**   **Live**

As the colours of the wires in the mains lead of this apparatus may not correspond with the coloured markings identifying the terminals in your plug proceed as follows,

The wire which is coloured **BLUE** must be connected to the terminal which is marked with the letter **N** or coloured black.

The wire which is coloured **BROWN** must be connected to the terminal which is marked with the letter **L** or coloured red.



## NOTICE

This manual has been written for the MZ-700 series personal computers and the BASIC interpreter which is provided with the MZ-700.

- (1) All system software for the MZ-700 series computers is supported in software packs (cassette tape, etc.) in file form. The contents of all system software and the material presented in this manual are subject to change without prior notice for the purpose of product improvement and other reasons, and care should be taken to confirm that the file version number of the system software used matches that specified in this manual.
- (2) All system software for the Sharp MZ-700 series personal computer has been developed by the Sharp Corporation, and all rights to such software are reserved. Reproduction of the system software or the contents of this book is prohibited.
- (3) This computer and the contents of this manual have been fully checked for completeness and correctness prior to shipment; however, if you should encounter any problems during operation or have any questions which cannot be resolved by reading this manual, please do not hesitate to contact your Sharp dealer for assistance.

Notwithstanding the foregoing, note that the Sharp Corporation and its representatives will not assume responsibility for any losses or damages incurred as a result of operation or use of this equipment.

Personal Computer  
**MZ-700**

# Owner's Manual

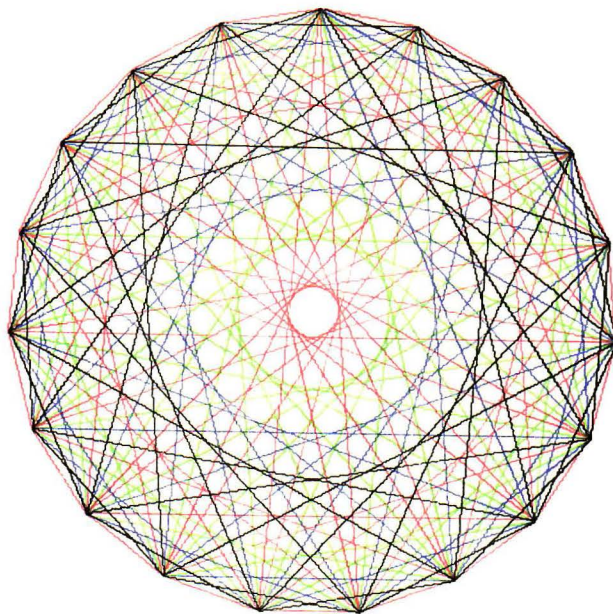
## Preface

---

Congratulations on your purchase of a Sharp MZ-700 series personal computer. Before using your computer, please read and make sure you understand the operating procedures which are described in this manual. The features and general operating procedures are described in Chapters 1 and 3, so please read those chapters first.

All software for the MZ-700 series computers is distributed on cassette tape.

The cassette tape included with the computer contains BASIC 1Z-013B, a high level BASIC interpreter which enables programming in the BASIC language and makes it possible to utilize the full capabilities of the MZ-700. The BASIC 1Z-013B interpreter and procedures for its use are fully described in this manual.



THIS FIGURE DRAWN USING THE COLOR PLOTTER-PRINTER

---

# MZ-700 OWNER'S MANUAL

## CONTENTS

### Chapter 1 The world of MZ-700 Series Personal Computer

- 1. 1 Features of the MZ-700 Series ..... 10
- 1. 2 Using this Manual ..... 12
- 1. 3 An Introduction to the World of Computers ..... 13

### Chapter 2 BASIC

- 2. 1 Introduction to Programming in BASIC ..... 16
- 2. 2 An Outline of BASIC ..... 21
- 2. 3 Frequently Used BASIC Commands and Statements ..... 28
- 2. 4 Built-in Function ..... 71
- 2. 5 String Function ..... 76
- 2. 6 Color display state ment ..... 80
- 2. 7 Color Plotter-Printer Commands ..... 82
- 2. 8 Machine Language Program Control Statements ..... 91
- 2. 9 I/O Statements ..... 95
- 2. 10 Other Statements ..... 96
- 2. 11 Monitor Function ..... 99

### Chapter 3 Operating the MZ-700

- 3. 1 Appearance of the MZ-700 Series Personal Computers ..... 104
- 3. 2 Connection to Display Unit ..... 106
- 3. 3 Data Recorder ..... 108
- 3. 4 Color Plotter-Printer ..... 110
- 3. 5 Key Operation ..... 114

### Chapter 4 Hardware

- 4. 1 MZ-700 System Diagram ..... 122
- 4. 2 Memory configuration ..... 123
- 4. 3 Memory Mapped I/O (\$E000-\$E008) ..... 130
- 4. 4 Signal System of Color V-RAM ..... 133
- 4. 5 MZ-700 Circuit Diagrams ..... 134

### Chapter 5 Monitor Commands and Subroutines

- 5. 1 Monitor Commands ..... 146
  - 5. 2 Functions and Use of Monitor Commands ..... 147
  - 5. 3 Monitor Subroutines ..... 151
-

---

# INDEX

## APPENDICES

A. 1	Code Tables .....	154
A. 2	MZ-700 Series Computer Specifications .....	157
A. 3	BASIC Error Message List .....	159
A. 4	Z80A Instruction Set .....	160
A. 5	Monitor Program Assembly List .....	164
A. 6	Color Plotter-Printer Control Codes .....	198
A. 7	Notes Concerning Operation .....	201

---



# INDEX

[BASIC COMMANDS] ( ) is abbreviated format

## **A**

ABS ..... 71  
 ASC ..... 78  
 ATN ..... 71  
 AUTO ..... (A.) ..... 31  
 AXIS ..... (AX.) ..... 89

## **B**

BYE ..... (B.) ..... 35

## **C**

CHR\$ ..... 78  
 CIRCLE ..... (CI.) ..... 90  
 CLOSE ..... (CLO.) ..... 68  
 CLR ..... 59  
 COLOR ..... (COL.) ..... 80  
 CONSOLE ..... (CONS.) ..... 98  
 CONT ..... (C.) ..... 34  
 COS ..... 71  
 CURSOR ..... (CU.) ..... 61

## **D**

DEF FN ..... 56  
 DEF KEY ..... 57  
 DELETE ..... (D.) ..... 31  
 DIM ..... 56

## **E**

END ..... (E.) ..... 59  
 EXP ..... 71

## **F**

FOR~NEXT ..... (F. ~N.) ..... 47

## **G**

GET ..... 43  
 GOSUB  
   ~RETURN ..... (GOS.~RET.) ..... 49  
 GOTO ..... (G.) ..... 48  
 GPRINT ..... (GP.) ..... 88

## **H**

HSET ..... (H.) ..... 88

## **I**

IF ERL ..... 97  
 IF ERN ..... 96  
 IF~GOSUB ..... (IF~GOS.) ..... 53  
 IF~GOTO ..... (IF~G.) ..... 53  
 IF~THEN ..... (IF~TH.) ..... 50  
 INP ..... 95  
 INPUT ..... (I.) ..... 42  
 INPUT/T ..... (I./T) ..... 68  
 INT ..... 71

## **J**

## **K**

KEY LIST ..... (K. L.) ..... 35

## **L**

LEFT\$ ..... 77  
 LEN ..... 76  
 LET ..... 36  
 LIMIT ..... (LIM.) ..... 92  
 LINE ..... 85  
 LIST ..... (L.) ..... 32  
 LIST/P ..... (L./P) ..... 84  
 LN ..... 71  
 LOAD ..... (LO.) ..... 28  
 LOG ..... 71

## **M**

MERGE ..... (ME.) ..... 32  
 MID\$ ..... 77  
 MODE GR ..... (M. GR) ..... 83  
 MODE TL ..... (M. TL) ..... 83  
 MODE TN ..... (M. TN) ..... 83  
 MODE TS ..... (M. TS) ..... 83  
 MOVE ..... 87  
 MUSIC ..... (MU.) ..... 65

---

**N**

NEW ..... 32

**O**

## ON ERROR

GOTO ..... (ON ERR. G.) ..... 96  
ON~GOSUB ..... (ON~GOS.) ..... 55  
ON~GOTO ..... (ON~G.) ..... 54  
OUT ..... 95

**P**

PAGE ..... 84  
PAI ..... 71  
PCOLOR ..... (PC.) ..... 83  
PEEK ..... 93  
PHOME ..... (PH.) ..... 87  
PLOT OFF ..... (PL. OFF) ..... 98  
PLOT ON ..... (PL. ON) ..... 98  
POKE ..... 92  
PRINT ..... (?) ..... 37  
PRINT USING ..... (?USI.) ..... 38  
PRINT/P ..... (?/P) ..... 84  
PRINT/T ..... (?/T) ..... 68  
PRINT [ $\alpha, \beta$ ] ..... (? [ $\alpha, \beta$ ]) ..... 81

**Q****R**

RAD ..... 71  
READ~DATA ..... (REA. ~DA.) ..... 44  
REM ..... 58  
RENUM ..... (REN.) ..... 33  
RESET ..... 63  
RESTORE ..... (RES.) ..... 46  
RESUME ..... (RESU.) ..... 97  
RIGHT\$ ..... 77  
RLINE ..... (RL.) ..... 86  
RMOVE ..... (RM.) ..... 87  
RND ..... 72  
ROPEN ..... (RO.) ..... 68  
RUN ..... (R.) ..... 34

**S**

SAVE ..... (SA.) ..... 29  
SET ..... 63  
SGN ..... 71  
SIN ..... 71  
SIZE ..... 97  
SKIP ..... 84  
SPC ..... 62  
SQR ..... 71  
STOP ..... (S.) ..... 59  
STR\$ ..... 79

**T**

TAB ..... 62  
TAN ..... 71  
TEMPO ..... (TEM.) ..... 67  
TEST ..... (TE.) ..... 84  
TIS ..... 60

**U**

USR ..... (U.) ..... 93

**V**

VAL ..... 79  
VERIFY ..... (V.) ..... 30

**W**

WOPEN ..... (W.) ..... 68

**X****Y****Z**

---



# THE WORLD OF MZ-700 SERIES PERSONAL COMPUTER

## Chapter 1



---

## 1.1 Features of the MZ-700 Series

In the space of just a few decades, the computer has undergone a dramatic transformation, changing from an intricate, enormously expensive monster weighing several dozen tons into a compact, inexpensive device which can be used by almost anyone. Whereas access to computers used to be limited to a few privileged individuals with special training, the inexpensive, user-friendly machines now available make the world of computing open to people in all different walks of life. The Sharp MZ-700 series computers are representative of such machines.

People use words and expressions to convey meanings.

Computers of the Sharp MZ-700 series, however, convey meaning through an ordinary television set or special printer. Any TV set can be used, either color or black-and-white; or, you may invest in one of the special display screens available if you want greater resolution and sharpness; you will be surprised at the beauty which is provided by such displays.

A tape recorder can be connected to computers of the Sharp MZ-700 series to record programs, the instructions which control the operation of the computer. When *printed* records of such programs or of the results of computer processing are desired, they can be obtained on the MZ-700's compact, elegantly designed 4-color plotter-printer.



MZ-731

**Note:** In the remainder of this manual, the term “MZ-700” will be used to indicate any of the computers of the MZ-700 series (the MZ-710, MZ-711, MZ-721, and MZ-731).





MZ-721



MZ-711

---

## 1.2 Using this Manual

Before starting to study programming, why not try playing with the MZ-700 a bit? We're sure you want to do that anyway, rather than waiting until after you have read this book. **First, read "Operating the MZ-700" in Chapter 3 (you need read only those parts which apply to the model which you are using). Connect the MZ-700 to a television, read the explanation of procedures for using the keyboard, and learn which characters are output when each key is pressed.**

If you are using the MZ-700 for the first time, read Chapters 1 and 2, in that order. At first, you may find it difficult to grasp the meanings of the various commands and statements of the BASIC programming language; however, even if you don't understand the explanations, be sure to key in the examples as they are encountered. As you do so, you will gradually develop a concept of what BASIC is all about.

You may skip over those portions of Chapter 2 which start with 2. 8 "Machine Language Program Control Statements"; however, these sections will prove useful when you have completely mastered programming in BASIC, or wish to become more familiar with the computer's internal operation.

If you have used the MZ-80K, you will find that the commands and statements of BASIC for the MZ-700 are used in the same manner as those of the SP-5025 family, so that the MZ-700 can be used in almost exactly the same manner as the MZ-80K. The major difference between the two is in the color statements (applicable to both the television screen and the color plotter-printer) which have been added; however, you should find it easy to become familiar with these by reading sections 2. 6 "Color display statement" and 2. 7 "Color Plotter-printer Commands." Having done this, you will quickly be captivated by the power of expanded BASIC.

This manual also includes a discussion of "Operating the MZ-700" (Chapter 3), a reference section entitled "Hardware" (Chapter 4), a discussion of the "Monitor Commands and Subroutines" (Chapter 5), and appendices of other information.

Now go ahead and learn everything you can about the MZ-700. We hope that you will find this manual helpful.

---

## 1.3 An Introduction to the World of Computers

### 1.3.1 What is BASIC?

People use language to communicate with each other, and specially designed languages are also used for communication with computers. BASIC is one such language.

#### Beginner's All-purpose Symbolic Instruction Code

Just as human beings use languages such as English, French, German, and Japanese for communication, there are also many different languages which are used for communication with computers. Among these are BASIC, FORTRAN, COBOL, and PASCAL. Of these, BASIC is the computer language whose structure is closest to that of the languages used by humans, and therefore is the easiest for humans to understand.

### 1.3.2 What is a "Clean Computer"?

The MZ-700 is a clean computer. Here, the word "clean" means that the computer's memory is completely blank when its power is turned on i.e., the computer cannot be used immediately, but first must be taught a language. This might seem like a nuisance at first glance; however, it does provide several advantages. For example, suppose that you wanted to use a language other than BASIC. The fact that the computer's memory is empty to start with means that you can teach it just about any language you want. This greatly increases the range of software which can be run on the computer and extends its range of potential applications.

On the other hand, if the computer knows BASIC from the time it is turned on, it is possible to use the language immediately; however, this presents an obstacle to loading any other language into memory.

### 1.3.3 Loading BASIC into the MZ-700

The BASIC language must be loaded into the MZ-700 before it can be used to do any work. A cassette tape containing this language has been included in the case containing the MZ-700. Now let's teach the language to the computer; procedures for doing this are described below. (The explanation assumes that you are using an MZ-731; however, the procedures are basically the same for all computers of the MZ-700 series.)

- (1) Connect the display as described on page 106.
- (2) Turn on the power switch located on the back of the computer.
- (3) The following characters are displayed on the screen and a square, blinking pattern appears. This pattern is referred to as the cursor.

\*\* MONITOR 1Z-Ø13A \*\*

\*

Cursor

- (4) Set the cassette tape containing the BASIC language in the computer's data recorder.
- (5) Type in the word LOAD<sup>※1</sup> and press the CR<sup>※2</sup> key. After doing this, the message **⌞ PLAY** appears on the screen.

#### Notes:

- ※1 LOAD . . . This is the instruction for loading programs or data from cassette tape.
- ※2 CR . . . . . This is referred to as the carriage return key, and is mainly used to indicate completion of entry of an instruction.

- (6) Press the data recorder's **PLAY** button; the cassette tape starts moving and loading of the BASIC language begins.
- (7) After loading has been completed, the message **READY** is displayed and the cursor starts to blink again.

```

** MONITOR 1Z-013A**
* LOAD
  PLAY
LOADING BASIC
  BASIC INTERPRETER 1Z-013B Vx.xx
  COPYRIGHT 1983 BY SHARP CORP
  XXXXX BYTES
  READY
  
```

This completes loading of the BASIC program. You can talk to the computer using BASIC, and the computer will respond.

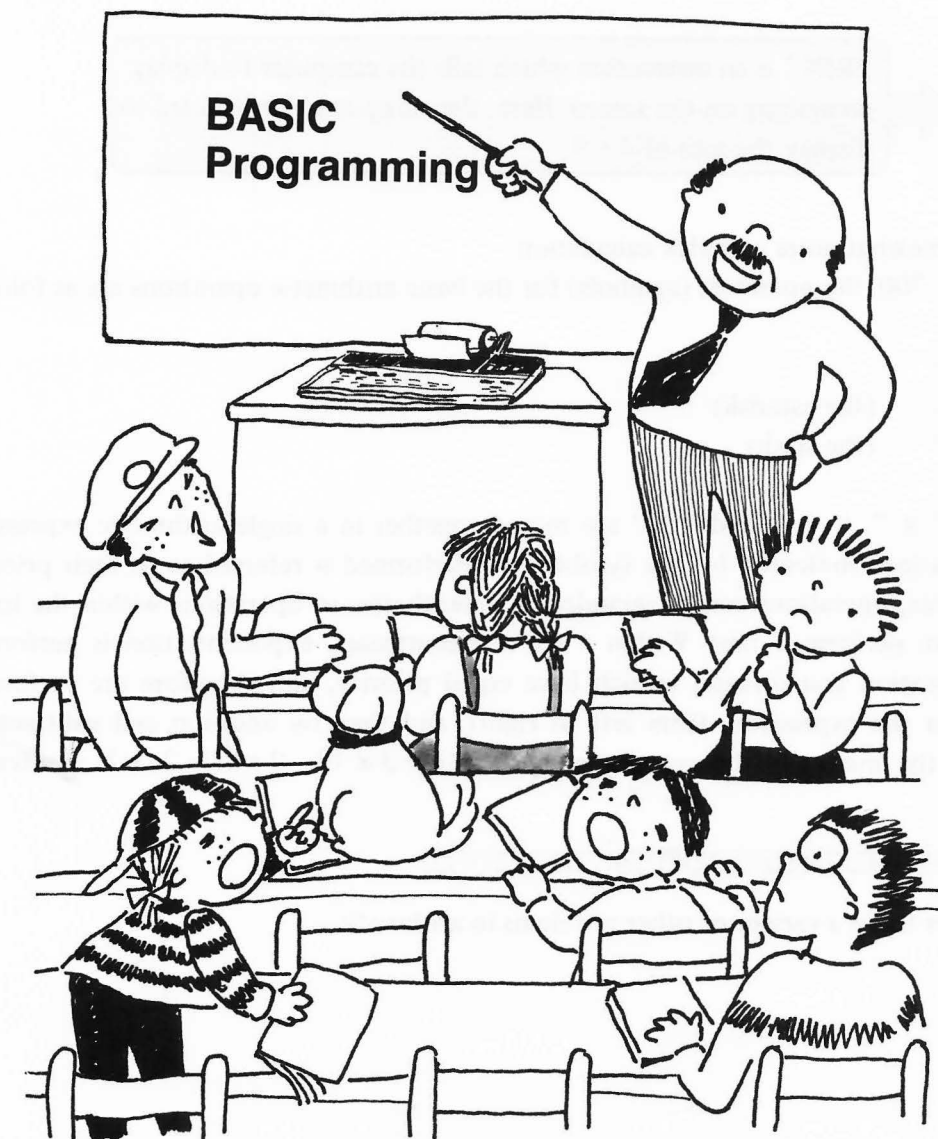
### 1.3.4 Try Executing a Program

Loading BASIC into the computer doesn't cause it to do anything; first, it must be given instructions in BASIC as to what it is to do. Although we will not explain the instructions of BASIC until later, let's go ahead and try executing a BASIC program right now.

Remove the cassette tape from the recorder and turn it over so that the "B" side is up. A sample program is recorded on this side of the cassette tape. Using the following procedures, load this program into the computer and execute it.

- (1) After turning the tape over and reloading it into the recorder, press the **REWIND** button to rewind it. Next, type in **LOAD** and press the **CR** key; when the message **PLAY** is displayed, press the **PLAY** button on the data recorder. This begins loading of the sample program.
- (2) When loading is completed, the cassette tape stops, **READY** is displayed on the screen, and the cursor starts to blink again.
- (3) Now that the program has been loaded into the computer's memory, try executing it. This is done by typing in **RUN** and pressing the **CR** key.
- (4) Now let's take a peek at the program. Hold down the **SHIFT** key and press the **BREAK** key. This stops program execution and displays the words **BREAK** and **READY**, then the cursor starts to blink again.
- (5) Type in **LIST** and press the **CR** key. This lists the lines of the program on the screen one after another. (Output of the list can be temporarily stopped at any time by pressing the space bar.)
- (6) If you wish to resume program execution, type in **RUN** again and hit the **CR** key.
- (7) If you want to run a different program, set the cassette tape containing that program in the recorder, **LOAD** the program, then **RUN** it. The previous program is automatically erased from memory when the new one is loaded, so the computer contains only the BASIC language and the last program loaded.

# BASIC





## 2.1 Introduction to Programming in BASIC

### 2.1.1 Direct Mode

Now that you have made some key entries on the MZ-700, you have reached the point where you are ready to start learning how to program. Before you start, however, try using the MZ-700 as you would an ordinary pocket calculator. (This is called operating the MZ-700 in the “direct mode”.) Key in the following, just as you would on a pocket calculator.

```
4+9=CR
```

As you can see, the computer doesn't do anything when it is presented with a problem in this form; your computer and an ordinary calculator are completely different in this respect, and instructions must be entered in a form which can be understood by the computer (i.e., in the form prescribed by the BASIC language). Now try typing in the following.

```
PRINT 4+9CR
```

If you have done this correctly, the number “13” will be displayed and the screen will appear as shown below.

```
READY
PRINT 4+9
13
READY
```

PRINT is an instruction which tells the computer to display something on the screen. Here, the computer is instructed to display the sum of 4 + 9.

Now let's try doing a somewhat more complex calculation.

With BASIC for the MZ-700, the operators (symbols) for the basic arithmetic operations are as follows.

Addition:	+	
Subtraction:	-	
Multiplication:	*	(the asterisk)
Division:	/	(the slash)
Exponentiation:	↑	

When symbols such as a “\*”, “+”, and “↑” are mixed together in a single arithmetic expression, the order in which calculations indicated by the symbols are performed is referred to as their priority. Just as with ordinary algebra, operations can be included in parentheses, so operations within the innermost set of parentheses are performed first. Within a set of parentheses, exponentiation is performed first, followed by multiplication and division (which have equal priority, and therefore are performed as they are encountered in the expression, from left to right), and then by addition and subtraction.

For example, to obtain the answer to the expression  $3 \times 6 \times (6 + 3 \times (9 - 2 \times (4 - 2)) + 1)$ , enter the following.

```
PRINT 3*6*(6+3*(9-2*(4-2))+1)CR
```

Now try using the computer to do a variety of other problems in arithmetic.

**[EXERCISE]**

1.  $\frac{6+4}{6-4}$

2.  $3 \times 15 + 9 \times (9-2) - \frac{6}{4-2} + 5$

3.  $(3+4) \times (5+6)$

4.  $\frac{10+20}{6} \times (2+3)$

5.  $\frac{10+20}{6 \times (2+3)}$

**[ANSWER]**PRINT (6+4)/(6-4)  
5PRINT 3\*(5+9\*(9-2)-6/(4-2))+5  
200PRINT (3+4)\*(5+6)  
77PRINT (10+20)/6\*(2+3)  
25PRINT (10+20)/(6\*(2+3))  
1

After going through the exercises, try typing in `[?][5][*][8]` and pressing the `[CR]` key; the answer "40" is displayed. The reason for this is that BASIC interprets the question mark in the same manner as the instruction **PRINT**. Remember this as a convenient, abbreviated form of the PRINT instruction.

Now try entering the following. (The quotation marks are entered by holding down `[SHIFT]` and pressing the `[2]` key.)

`PRINT"4+9="CR`

As you can see, the characters within quotation marks are displayed on the screen, but the answer is not. Now try entering the following.

`PRINT"ABCDEFGG"CR`

This causes ABCDEFG to be displayed on the screen.

In other words, using the PRINT instruction together with quotation marks tells the MZ-700 to display characters on the screen exactly as they are specified between quotation marks. The characters within any set of quotation marks are referred to as a "character string" or "string".

Now go on to enter the following.

`PRINT"4+9="";4+9CR`

This causes the following to be displayed on the screen.

4+9=\_ 1 3..... (The "\_" symbol indicates a space. Actually, nothing is displayed on the TV screen in the position indicated by this symbol.)

In other words, the instruction above tells the computer to display both the character string "4 + 9 =" and the result of the arithmetic expression "4 + 9 =". Now try entering the following.

`PRINT"4+9="",4+9CR`

After typing in this entry, the following should be displayed on the screen.

4+9=\_\_\_\_\_13

The reason the screen appears different this time is because the PRINT instruction displays items of information (character strings or the results of arithmetic expressions) differently depending on whether they are separated from each other by semicolons or commas.

Semicolon ( ; ) ..... Instructs the computer to display items immediately adjacent to each other.

Comma ( , ) ..... Instructs the computer to display the item at the position which is 10 spaces (columns) from the beginning of the display line.

If you have the MZ-731 (or a separate plotter-printer), now try appending the characters `[/P]` to the end of the word `PRINT`.

```
PRINT[/P]"4+9=";4+9CR
```

This time nothing appears on the display screen, but the same result is printed out on the plotter-printer. In other words, the `[/P]` symbols switch output from the display to the plotter-printer.

This completes our explanation of procedures for using the MZ-700 as you would a pocket calculator.

**Note:** `PRINT "5 + 8 ="; 5 + 8` displays `5 + 8 = 13`, while `PRINT "5 - 8 ="; 5 - 8` displays `5 - 8 = -3`.

The reason for this is that one space is always reserved for a symbol indicating whether the result is positive or negative, but the symbol is only displayed in that space when the result is negative.

## 2.1.2 Programming

Let's try making a simple program. However, first let's make sure that the area in the computer's memory which is used for storing programs is completely empty. Do this by typing in `NEW` and pressing the `[CR]` key. (This instruction will be explained in more detail later; see page 32.)

Type in the following program exactly as shown.

```
10 A=3CR ..... Assigns the value 3 to A.
20 B=6CR ..... Assigns the value 6 to B.
30 C=A+BCR ..... Assigns the result of A + B to C.
40 ? CCR ..... Displays the value assigned to C.
50 ENDCR ..... Instruction indicating the end of the program.
```

The numbers 10, 20, 30, and so forth at the left end of each line are referred to as program line numbers, or simply line numbers; these numbers indicate the order in which instructions are to be executed by the computer. Instructions on the lowest numbered line are executed first, followed by those on the next lowest numbered line, and so forth. Line numbers must be integers in the range from 1 to 65535.

The line numbers 1, 2, 3, and so forth could have been used in this program instead of 10, 20, 30. However, it is common practice to assign line numbers in increments of 10 to provide room for later insertion of other lines.

Now let's check whether the lines have been correctly entered. Type in `LIST` and press the `[CR]` key; this causes a list of the program lines to be displayed. Notice that the question mark entered at the beginning of line 40 has been converted to `PRINT`, the full form of the command for displaying data on the display screen.



```
LIST
10 A=3
20 B=6
30 C=A+B
40 PRINT C
50 END
READY
```

---

Now let's try executing the program.

`RUNCR`

Enter RUN and press the `CR` key; the result is displayed on line 9 of the screen.

Now we will explain procedures for making changes in programs. First, let's change the instruction on line 20 from  $B = 6$  to  $B = 8$ . Type in LIST 20 and press the `CR` key; this displays just line 20 of the program on the screen. Next, use the cursor control keys (the keys at the right side of the keyboard which are marked with arrows) to move the cursor to the number '6', then press the `8` key and the `CR` key in succession to make the change. **Note that the change is not completed until the `CR` key is pressed.**

Now type in LIST and press the `CR` key again to confirm that the change has been made.

Next, let's change line 30 of the program to  $C = 30 * A + B$ .

Using the cursor control keys, move the cursor so that it is positioned on top of the 'A' in line 30, then press the `INST` key three times in succession. This moves 'A + B' three spaces to the right.

```
C=   A+B
    ↑
    Cursor position
```

Now type in `30X` and press the `CR` key to complete the insertion. LIST the program to confirm that the change has been made correctly.

Now change line 30 again so that it reads " $C = 30 * A$ " instead of " $C = 30 * A + B$ ". Do this by moving the cursor to the position immediately to the right of B and pressing the `DEL` key two times; this deletes "+B". Press the `CR` key to complete the change.

Now LIST the program and confirm that it appears as shown below.

```
10 A=3
20 B=8
30 C=30*A
40 PRINT C
50 END
```

To delete an entire line from a program, simply enter the line number of that line and press the `CR` key; delete line 20 in this manner, then LIST the program to confirm that the line has been deleted.

We could insert the instruction "?A" between lines 30 and 40, by typing in 35?A and pressing the `CR` key. Try this, then LIST the program to confirm that the line has been added. Now delete line 35 by entering 35 and pressing the `CR` key.

The process of changing or inserting lines in a program in this manner is referred to as **editing**, and the program which results from this process is referred to as the **BASIC text**. Each line of the program can include a maximum of 255 characters, including the line number, but the maximum length is reduced by four characters if the question mark is used to represent the PRINT instruction.

At this point, the program contained in the computer's memory should be as follows.

```
10 A=3
30 C=30*A
40 PRINT C
50 END
```

Now we will use this program to explain the procedures for recording programs on cassette tape. Prepare a blank cassette tape (one on which nothing has been recorded) and set it in the data recorder,

---

then type in the following from the keyboard.

SAVE "CALCULATION" ↵

Here, "CALCULATION" is the name which is to be recorded on the cassette tape to identify the program. Any name may be assigned, but the name cannot be longer than 16 characters.

**Note:** The ↵ symbol in the example above represents the CR key.

When the CR key is pressed, "⏏ RECORD. PLAY" is displayed on the screen. Pressing the RECORD button on the data recorder at this time records the program on cassette tape.

The name which is assigned to the program is referred to as its **file name**. Specification of a file name is not absolutely necessary, but from the point of view of file management it is a good idea to assign one. Of course, the file name is recorded on the tape together with the program.

When recording is completed, READY is displayed to indicate that the computer is finished. Now press the STOP button on the data recorder and rewind the tape.

The program is still present in the computer's memory after recording is completed, so type in NEW ↵ to delete it (enter LIST ↵ to confirm that the program has been deleted). Now let's try using the LOAD instruction to load the program back into memory from the cassette tape as described on page 14.

When a cassette tape contains many programs, that which is to be loaded can be identified by specifying the program's file name together with the LOAD instruction as follows.

LOAD "CALCULATION" ↵

Specifying the file name in this manner tells the computer to ignore all programs on the tape other than that with the specified name. If the file name is not specified (if only LOAD ↵ is entered), the computer loads the first program encountered.

**Note:** When using cassette recorder other than the data recorder built into the MZ-731, and MZ-721 read the instructions on page 109 before attempting to record or load programs.

The LIST command shown above can be used in a variety of different ways. For example, during editing LIST 20 ↵ can be used to display just line 20 of a program. The entire program can be listed by entering LIST ↵. Other uses of the instruction are as follows.

<span style="border: 1px solid black; padding: 0 2px;">LIST</span> <span style="border: 1px solid black; padding: 0 2px;">-30</span> <span style="border: 1px solid black; padding: 0 2px;">CR</span>	Lists all lines of the program to line 30.
<span style="border: 1px solid black; padding: 0 2px;">LIST</span> <span style="border: 1px solid black; padding: 0 2px;">30-</span> <span style="border: 1px solid black; padding: 0 2px;">CR</span>	Lists all lines from line 30 to the end of the program.
<span style="border: 1px solid black; padding: 0 2px;">LIST</span> <span style="border: 1px solid black; padding: 0 2px;">30-50</span> <span style="border: 1px solid black; padding: 0 2px;">CR</span>	Lists all lines from line 30 to line 50.
<span style="border: 1px solid black; padding: 0 2px;">LIST</span> <span style="border: 1px solid black; padding: 0 2px;">30</span> <span style="border: 1px solid black; padding: 0 2px;">CR</span>	Lists line 30.

When editing programs by listing individual lines with the LIST instruction, press the CLR key (the INST key) together with the SHIFT key when the screen becomes distractingly crowded. This clears the entire screen and moves the cursor to its upper left corner. (This does not affect the program in memory). Afterwards, enter LIST < line number > ↵ again to list the line which is to be edited.



---

## 2.2 An Outline of BASIC

### 2.2.1 Constants

A constant is a number or string of characters which is written into a program, and which is used by that program as it is executed. Types of constants include numeric constants, string (character) constants, and system constants. These are explained below.

#### Numeric constants

A numeric constant is a number which has a maximum of 8 significant digits. The exponent of such constants must be in the range from  $10^{-38}$  to  $10^{38}$  (the maximum range is 1.548437E-38 to 1.7014118E+38).

(Examples:)

-123. 4

Ø. 789

3748. Ø

3. 7E+12.....3. 7×10<sup>12</sup>

7. 65E-9.....7. 65×10<sup>-9</sup> } E indicates the exponent.

14. 8E9.....14. 8×10<sup>9</sup> }

Hexadecimal numbers: Numbers can be specified in hexadecimal format only for direct memory addressing with the LIMIT, POKE, PEEK, and USR instructions (see pages 92 and 93), and are represented as four digits preceded by a dollar sign (\$).

(Examples:)

LIMIT \$BFFF

USR (\$CØØØ, X\$) ..... X\$ represents a string variable.

#### String constants

String constants are letters and symbols between quotation marks which are included in programs to allow titles or messages to be output to the display screen or printer. The characters "4+9" appearing on page 17 are a character constant, and not a numeric constant. With BASIC, a string constant may consist of a maximum of 255 characters. (Not including quotation marks which cannot be included in a string constant.)

(Examples:)

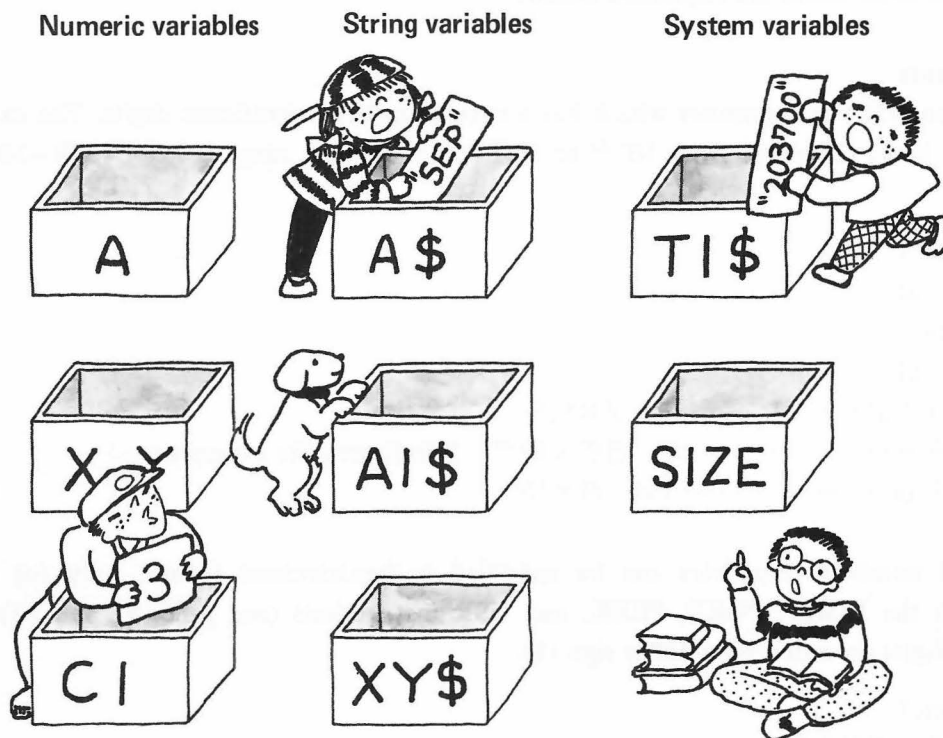
" ABCDEFG "

" 1234567891Ø "

DATA ABCDEFG..... Quotation marks are not needed when string constants are specified in a DATA statement; however, they may be used if desired.

## 2.2.2 Variables

The word “variable” has a different meaning with BASIC than it does when used with regard to algebraic expressions. To put it in very simple terms, the variables of BASIC are “boxes” in memory for the storage of numbers and characters (character strings). The types of variables used in BASIC include numeric variables, string variables, and system variables.



### Numeric variables

Only numeric data can be stored in numeric variables.

Names must be assigned to these variables in accordance with the following rules.

- i) A variable name may consist of any number of characters, but only the first two characters are actually used by the BASIC interpreter to identify the variable. Further, the first character of the variable name must be a letter (A to Z), either letters or numerals may be used for subsequent characters.
- ii) It is not possible to use the names of BASIC commands and statements as variable names.

Correct variable names: ABC, XY, ABCD, A12345  
(ABC and ABCD are regarded as the same variable.)

Incorrect variable names: PRINT ..... (PRINT is a BASIC statement)  
C@ ..... (Variable names may not include special characters.)

(Example:)

1Ø A=5..... Stores 5 in variable A.

2Ø PRINT A..... Displays the value stored in variable A.

---

### String variables

String variables are variables which are used for storing character strings. Names assigned to string variables must conform to the same rules as those assigned to numeric variables; however a dollar sign (\$) is appended to the end of string variable names to differentiate them from other types of variables.

String variables may be used to store a maximum of 255 characters. Such variables are blank until string data is assigned to them.

The only operator which can be used in expressions including more than one string variable is the "+" sign.

(Example:)

- 10 A\$ = " ABCD " ..... Substitutes the character string ABCD into string variable A\$.
- 20 B\$ = " XYZ " ..... Substitutes the character string XYZ into string variable B\$.
- 30 C\$ = A\$ + B\$ ..... Substitutes the sum of string variables A\$ and B\$ (ABCDXYZ) into string variable C\$.
- 40 PRINT C\$ ..... Displays the contents of string variable C\$.

### System Variables

System variables contain values which are automatically changed by the BASIC interpreter. The system variables are SIZE (the variable which indicates the amount of BASIC free area) and TI\$ (a 6-digit variable which contains the value of the system's 24-hour clock).

(Examples:)

- 10 TI\$ = " 013500 " ... This statement assigns the value corresponding to 1:35:00 A.M. to system variable TI\$ and sets the system clock to that time.
- 20 PRINT TI\$ ..... Executing this statement displays the current time of the system clock (24-hour time).

Display format:

132819 ..... Indicates that the time is 13:28:19.

- PRINT SIZE ..... This displays the current amount of free space in the computer's memory (in other words, the amount of space which is available for additional program lines). The value indicated by this variable is reduced each time a program line is entered.

## 2.2.3 Arrays

Arrays can be thought of as shelves within the computer's memory which contain rows of boxes, each of which represents a variable. The boxes on these shelves are arranged in an orderly sequence, and are identified by means of numbers; these numbers are referred to as subscripts, because they are subscripted to the name which identifies the entire group of boxes.

Such shelves of boxes are set up simply by executing an instruction which declares that they exist; this is referred to as making an array declaration. The array declaration specifies the number of boxes which are to be included in each set of shelves (i.e., the size of the shelves) and the manner in which they are to be arranged.

The boxes in each unit of shelves may be arranged in sequences which have any number of dimensions. Thus, a one-dimensional array can be thought of as a single shelf which holds, one row of boxes; a two-dimensional array can be thought of as a stack of shelves, each of which holds one row of boxes; and so forth. These boxes, or variables, are referred to as the array's elements.

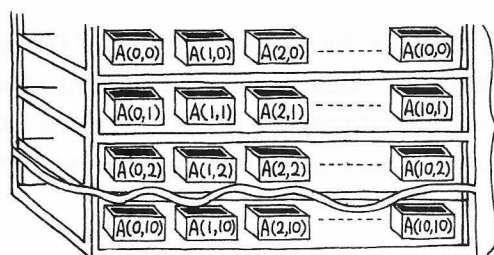
The number of subscripts used to identify each of the array elements of a corresponds to the number of dimensions in that array. For example, each of the elements in a one-dimensional array is identified by a single subscript which indicates the box's position in the row; each of the elements in a two dimensional array is identified by two subscripts, one which identifies the box's row, and one which indicates the box's position within that row; and so forth. The numbers which are used as the subscripts start with zero, and have a maximum value which is determined by the size of each of the array's dimensions (i.e., the number of boxes in each row, etc.).

The maximum size of an array is limited by the amount of free space which is available in the computer's memory (i.e., by the size of the program, the number of items of data which are to be stored in the array, and so forth). The syntax of BASIC places no restrictions on the number of dimensions which can be used for any array, but in practice the number of dimensions is limited by the amount of free memory space which is available for storage of array variables.

An array must be declared before values can be stored in any of its elements.

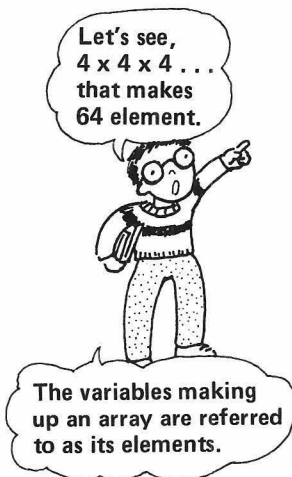
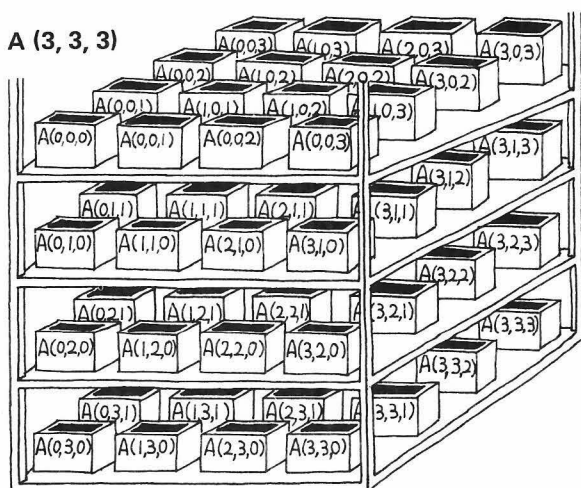


**A one-dimensional array consisting of 101 elements.**



**A two-dimensional array consisting of 11 x 11 elements.**

**A three-dimensional array consisting of 4 x 4 x 4 elements.**



---

(Example 1)

```
10 DIM A(5)..... Declares 1-dimensional numeric array A with 6 elements.
20 DIM X$(8)..... Declares 1-dimensional string array X$ with 9 elements.

10 DIM A(5), X$(8)..... Performs the same function as lines 10 and 20 above.
```

(Example 2)

```
10 DIM B(5, 5)..... Declares 2-dimensional numeric array B with 6 x 6
                      elements.
20 DIM Y$(5, 8)..... Declares 2-dimensional string array Y$ with 6 x 9 elements.

10 DIM B(5, 5), Y$(5, 8), A(5), X$(8)..... Declares two numeric arrays
                                              and two string arrays.
```

(Example 3)

```
10 DIM C(3, 3, 3)..... Declares 3-dimensional array C with 4 x 4 x 4 elements.
```

**Note:** Different names must be used for each array which is declared; for example, the instruction DIM A(5), A(6) is not a legal array declaration.

Try executing the program shown below and check the results which are obtained.

```
10 DIM A(2), B$(2)
20 A(0)=26
30 A(1)=9
40 A(2)=-100
50 B$(0)="ABC "
60 B$(1)="XYZ "
70 B$(2)="MZ-700 "
80 PRINT A(1)
90 PRINT B$(2)
100 PRINT A(2)
110 PRINT B$(0)+B$(1)
120 PRINT A(0)
```

**Note:** Individual variables within an array, such as A(5) and X\$(8), are referred to as an array's elements. Numeric constants, numeric variables, and numeric arrays are collectively referred to as numeric expressions, and string constants, string variables, and string arrays are collectively referred to as string expressions.

## 2.2.4 BASIC Operations

In BASIC, arithmetic operations take a slightly different form than is the case with ordinary arithmetic. The various arithmetic operators used in BASIC are shown in the table below. The priority of these operators when they are used together within a single expression (the sequence in which the different arithmetic operations are performed) is as indicated by the numbers in the left column of the table; however, operators within parentheses always have the highest priority.

### Arithmetic operations

	Operator	Operation	Format
1	$\uparrow$	Exponentiation	$X \uparrow Y$ (Indicates $X^Y$ ; i.e., X to the Yth power.)
2	$-$	Negation	$-X$
3	$*$ , $/$	Multiplication, division	$X * Y$ (X times Y), $X/Y$ ( $\frac{X}{Y}$ ; i.e., X divided by Y)
4	$+$ , $-$	Plus, minus	$X + Y$ (X plus Y), $X - Y$ (X minus Y)



#### (Example 1)

1Ø  $A=3*8/4$ .....When a series of operators with the same priority are used in an arithmetic expression, calculations are carried out from left to right; thus, the result of the expression at left is 6.

#### (Example 2)

1Ø  $A=6Ø-6*8+2$ .....Result is 14.  
2Ø  $B=(6Ø-6)*8+2$ .....Result is 434.

#### (Example 3)

1Ø  $A=2 \uparrow 3$  ..... Assigns 2 to the 3rd power to A; result is 8.

### String operations

String operations are used to create new strings of character data by concatenating (linking) two or more shorter strings. The only operator which can be used in string operations is the "+" sign.

#### (Example)

PRINT "ABC" + "DEF" J  $\longrightarrow$  Displays the character string "ABCDEF".



## 2.2.5 Initial settings

Initial settings made when BASIC 1Z-013B is started are as described below.

### ■ Keyboard

- 1) Operation mode: Normal (alphanumeric)
- 2) Definable function keys

[F1] : .....	"RUN" + CHR\$ (13)	[SHIFT] + [F1] : .....	"CHR\$ ("
[F2] : .....	"LIST"	[SHIFT] + [F2] : .....	"DEF KEY ("
[F3] : .....	"AUTO"	[SHIFT] + [F3] : .....	"CONT "
[F4] : .....	"RENUM"	[SHIFT] + [F4] : .....	"SAVE "
[F5] : .....	"COLOR"	[SHIFT] + [F5] : .....	"LOAD "

**Note** A carriage return code is included in the definition of function key F1 .

### ■ Built-in clock

The initial value set to system variable TIS is "000000" .

### ■ Music function

- 1) Musical performance tempo: 4 (moderato, approximately medium speed)
- 2) Note duration: 5 (quarter note J )

### ■ Control keys and control characters

The control keys are keys which perform special functions when pressed together with the [CTRL] key. Functions of these keys and their corresponding ASCII codes are as shown in the table below.

[Control codes]

CTRL +	ASCII code (decimal)	Function
E	5	Selects the lowercase letter input mode for alphanumeric characters.
F	6	Selects the uppercase letter input mode for alphanumeric characters.
M	13	Carriage return ([CR]).
P	16	Same as the [DEL] key.
Q	17	Moves the cursor down one line ([↓]).
R	18	Moves the cursor up one line ([↑]).
S	19	Moves the cursor one column (character) to the right ([→]).
T	20	Moves the cursor one column (character) to the left ([←]).
U	21	Moves the cursor to the home position ([HOME]).
V	22	Clears the screen to the background color ([CLR]).
W	23	Places the computer in the graphic character input mode ([GRAPH]).
X	24	Inserts one space ([INST]).
Y	25	Places the computer in the alphanumeric input mode.

### ■ Other

The lower limit of the BASIC text area is set to address \$FEFF; this is the same as LIMIT MAX is executed).

For initial printer settings, see the discussion of the printer.

## 2.3 Frequently Used BASIC Commands and Statements

### 2.3.1 Program file input/output instructions

2.3.1.1 LOAD ..... (abbreviated format: LO.)

Format
--------

LOAD or LOAD "filename"

Function
----------

This command loads the specified BASIC text file or a machine language file to be linked with a BASIC program from cassette tape.

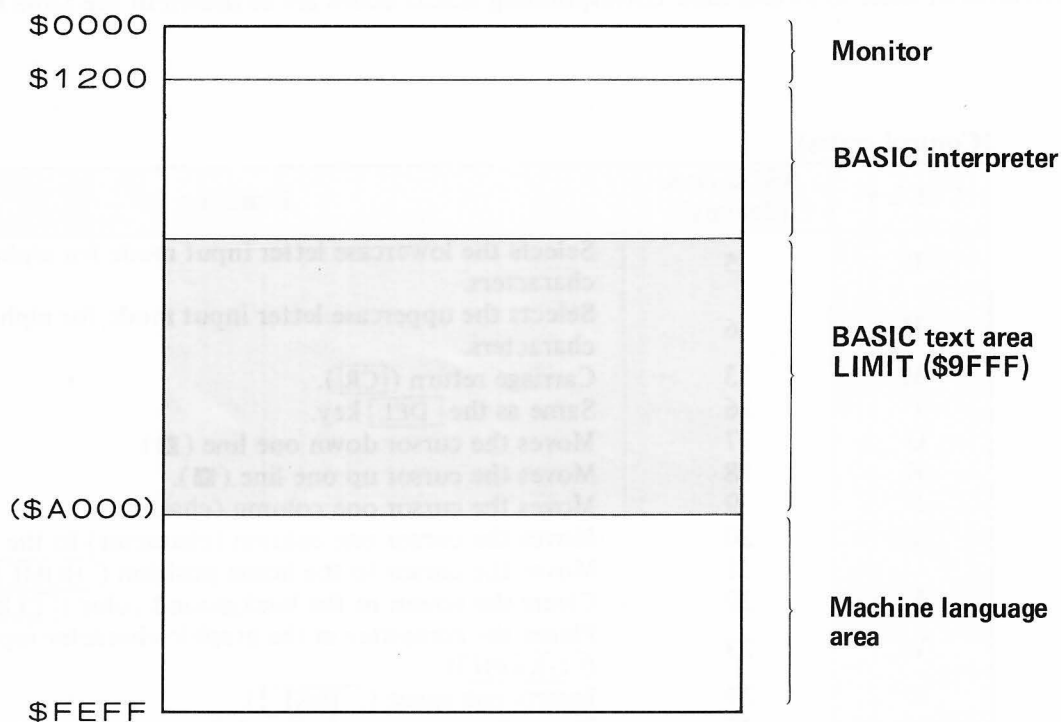
(See pages 14 and 20.)

Only BASIC text files and machine language programs can be loaded with this command. When the file to be loaded is a BASIC text file, the current program is cleared from the BASIC text area when the new program is loaded.

Note
------

When loading a machine language routine to be linked with a BASIC program, the LIMIT statement must be executed to reserve a machine language program area in memory. Further, the applicable machine language program file is executed as soon as loading is completed if the loading address is inside that area. (In this case, the BASIC text is not erased.)

The LOAD command can be used within a program to load a machine language program file.



**Note:** The lower limit of the BASIC text area shifts according to the size the program text loaded.

## 2.3.1.2 SAVE ..... (abbreviated format: SA.)

Format	SAVE or SAVE "filename"
Function	This command assigns a file name to the BASIC program in the computer's memory and saves it on cassette tape.



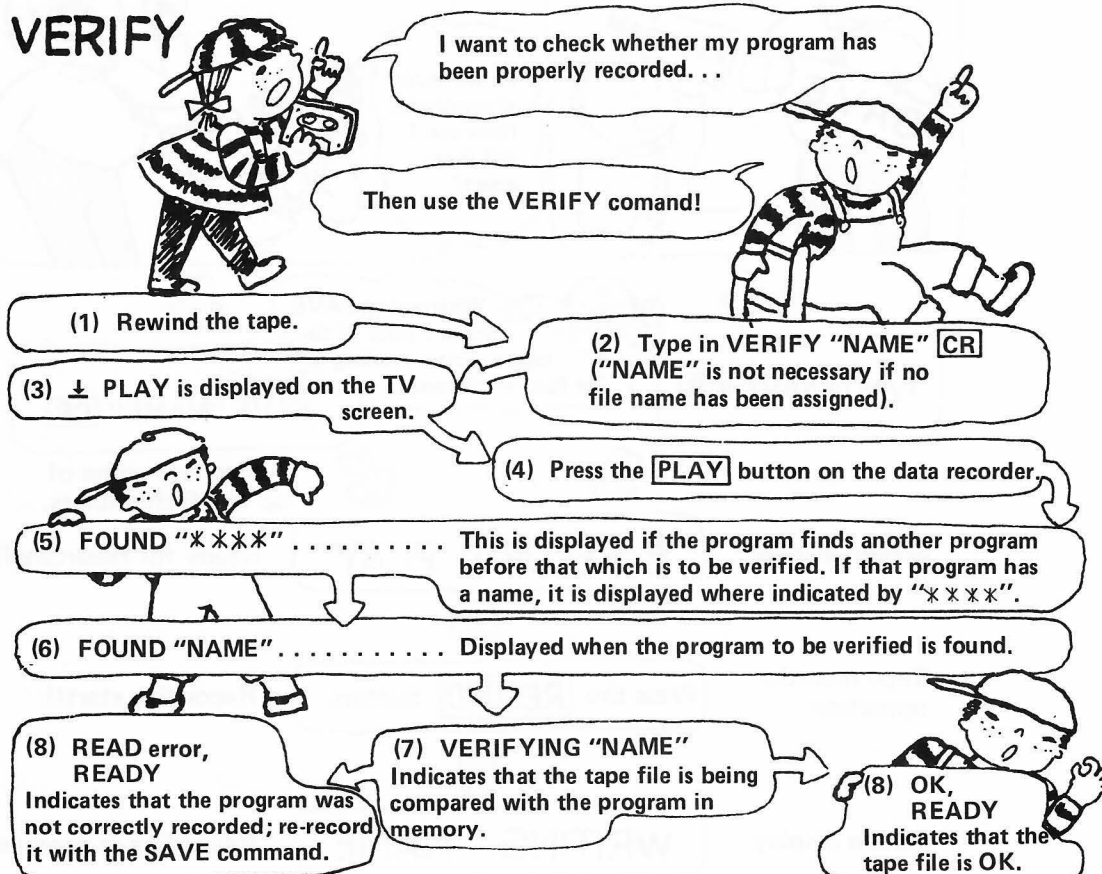
Note	<p>This command saves only the BASIC program text (i.e., the program text displayed by executing the LIST command); it does not save any machine language program in the machine language area.</p> <p>The file name specified is recorded on tape together with the BASIC text file; specify any name desired using up to 16 characters. If no file name is specified, the program is recorded without a file name; however note that this can make file management difficult if more than one program is recorded on a single tape.</p>
------	---

### 2.3.1.3 VERIFY ..... (abbreviated format: V.)

Format
Function

**VERIFY** or **VERIFY "filename"**

This command is used to confirm that programs have been properly recorded on tape by the SAVE command. This is done by playing the tape and comparing the program read with the program contained in memory. If both programs are the same, "OK" is displayed; if they are different, "READ error" is displayed. In the latter case, save the program again.



## 2.3.2 Text editing commands

### 2.3.2.1 AUTO ..... (abbreviated format: A.)

<b>Format</b>	AUTO or AUTO Ls, n Ls ..... Starting line number n ..... Line number increment
<b>Function</b>	This command automatically generates program line numbers during entry of BASIC program statements.
<b>Example</b>	<p>(Example 1)</p> <pre>AUTO ↵ 10..... ↵ 20..... ↵ 30..... ↵</pre> <p>(Example 2)</p> <pre>AUTO 300, 5 ↵ 300..... ↵ 305..... ↵ 310..... ↵</pre> <p>Automatically generates program line numbers with an increment of 5, starting with line 300.</p> <p>(Example 3)</p> <pre>AUTO 100 ↵ 100..... ↵ 110..... ↵ 120..... ↵</pre> <p>Generates program line numbers with an increment of 10, starting with line 100.</p> <p>(Example 4)</p> <pre>AUTO, 20 ↵ 10..... ↵ 30..... ↵ 50..... ↵</pre> <p>Generates program line numbers with an increment of 20, starting with line 10.</p>

**Note:** The AUTO command is terminated by pressing **SHIFT** and **BREAK**.

### 2.3.2.2 DELETE ..... (abbreviated format: D.)

<b>Format</b>	DELETE Ls—Le..... Deletes program lines from Ls to Le. DELETE —Le..... Deletes all program lines from the beginning of the program to line Le. DELETE Ls— ..... Deletes all program lines from line Ls to the end of the program. DELETE Ls ..... Deletes line Ls.
<b>Example</b>	<p>(Example 1)</p> <pre>DELETE 150—350 ↵.....Deletes all program lines from 150 to 350.</pre> <p>(Example 2)</p> <pre>DELETE —100 ↵.....Deletes all program lines up to line 100.</pre> <p>(Example 3)</p> <pre>DELETE 400— ↵.....Deletes all program lines from 400 to the end of the program.</pre>

### 2. 3. 2. 3 LIST ..... (abbreviated format: L.)

Format	LIST	} ..... Ls indicates the starting line number and Le indicates the ending line number.
	LIST Ls-Le	
	LIST Ls-	
	LIST -Le	

Function	This command lists all or part of the program lines contained in the BASIC text area on the display screen.
----------	---

LIST J ..... Lists the entire program.  
 LIST -30 J ..... Lists all lines of the program to line 30.  
 LIST 30- J ..... Lists all lines of the program from line 30 to the end.  
 LIST 30-50 J ..... Lists all lines of the program from line 30 to line 50.  
 LIST 30 J ..... Lists line 30 of the program.

Output of the program list to the display screen can be temporarily interrupted by pressing the space bar; listing is then resumed when the space bar is released. To terminate list output, press the **BREAK** key together with the **SHIFT** key.

### 2. 3. 2. 4 LIST/P ..... (abbreviated format: L./P)

Format	LIST/P <Ls-Le>
	Ls ..... Starting line number
	Le ..... Ending line number

Function	This command lists all or part of the program in the BASIC text area on the printer. The range of program lines to be listed is specified in the same manner as with the LIST command described above.
----------	--

**Note:** The angle brackets <...> in the above indicate that the enclosed item is optional.

### 2. 3. 2. 5 MERGE ..... (abbreviated format: ME.)

Format	MERGE or MERGE "filename"
--------	---------------------------

Function	The MERGE command is used to read a program from cassette tape. When a program is read using this command, it is appended to the program in memory. If "filename" is omitted, the computer reads the first file encountered on the cassette tape.
----------	---

If any line numbers in the program read are the same as those of the program in memory, corresponding lines of the program in memory are replaced with lines of the program read.

### 2. 3. 2. 6 NEW

Format	NEW
--------	-----

Function	The NEW command erases the BASIC text area and clears all variables. Execute this command when you wish to clear the program in memory prior to entering another program. This command does not erase the machine language area reserved by the LIMIT statement.
----------	--

Since the BASIC text area is automatically cleared by the LOAD command, it is not necessary to execute this command before loading a BASIC program from cassette tape.



## 2.3.2.7 RENUM ..... (abbreviated format: REN.)

### Format

RENUM		}	Ln ....	New line number
RENUM	Ln		.....Lo	Old line number
RENUM	Ln, Lo, n		n .....	Increment

### Function

This command renumbers the lines of a BASIC program. When this command is executed, line numbers referenced in branch statements such as GOTO, GOSUB, ON ~ GOTO, and ON ~ GOSUB are also reassigned.

RENUM ..... Renumbers the lines of the current program in memory so that they start with 10 and are incremented in units of 10.

RENUM 100 ..... Renumbers the lines of the current program in memory so that they start with 100 and incremented in units of 10.

RENUM 100, 50, 20 ..... Renumbers lines of the current program in memory starting with line number 50; line number 50 is renumbered to 100, and subsequent line numbers are incremented in units of 20.

### Example

The example below shows the result of executing RENUM 100, 50, 20 for a sample program.

(Before renumbering)		(After renumbering)
50 A=1	}	100 A=1
60 A=A+1		120 A=A+1
70 PRINT A		140 PRINT A
100 GOTO 60		160 GOTO 120

### Note

When specifying the new and old line numbers, the new line number specified must be larger than the old line number. Note that an error will result if execution of this command results in generation of a line number which is greater than 65535.

## 2.3.3 Control commands

### 2.3.3.1 RUN ..... (abbreviated format: R.)

Format
--------

**RUN or RUN Ls**

Ls .... Starting line number

Function
----------

This command executes the current program in the BASIC text area.

If the program is to be executed starting with the first program line, just enter RUN and press the 

CR
----

 key. If execution is to begin with a line other than that the lowest line number, type in RUN Ls (where Ls is the line number at which execution is to start) and press the 

CR
----

 key.

When this command is executed, the BASIC interpreter clears all variables and arrays before passing control to the BASIC program.

### 2.3.3.2 CONT ..... (abbreviated format: C.)

Format
--------

**CONT**

Function
----------

The CONT command is used to resume execution of a program which has been interrupted by pressing 

SHIFT
-------

 + 

BREAK
-------

 or by a STOP statement in the program. This command can also be used to continue execution of a program which has been interrupted by an END statement; however, in this case care must be taken to ensure that lines following the END statement are not the lines of a subroutine. Examples of situations in which the CONT command can and cannot be used are shown in the table below.

Program continuation possible	Program continuation not possible						
<ul style="list-style-type: none"><li>● Program execution stopped by pressing <table><tr><td>SHIFT</td></tr></table> + <table><tr><td>BREAK</td></tr></table>.</li><li>● Program execution stopped by a STOP command.</li><li>● Program execution stopped by pressing <table><tr><td>SHIFT</td></tr></table> + <table><tr><td>BREAK</td></tr></table> while the program was a waiting input for an INPUT statement.</li></ul>	SHIFT	BREAK	SHIFT	BREAK	<ul style="list-style-type: none"><li>● Before a RUN command has been executed.</li><li>● "READY" displayed due to an error occurring during program execution.</li><li>● Cassette tape operation interrupted by pressing <table><tr><td>SHIFT</td></tr></table> + <table><tr><td>BREAK</td></tr></table>.</li><li>● Program execution stopped during execution of a MUSIC statement.</li><li>● Program execution stopped and "READY" displayed after execution of an END statement.</li></ul>	SHIFT	BREAK
SHIFT							
BREAK							
SHIFT							
BREAK							
SHIFT							
BREAK							

---

#### 2.3.3.3 BYE ..... (abbreviated format: B.)

Format
Function

##### **BYE**

This command returns control of the computer from BASIC interpreter 1Z-013B to the monitor program in RAM. (The monitor commands are explained starting on page 99.)

#### 2.3.3.4 KEY LIST ..... (abbreviated format: K. L.)

Format
Function

##### **KEY LIST**

This command displays a list of the character strings assigned to the definable functions keys.

```
KEY LIST
DEF KEY (1) = " RUN " + CHR$ (13)
DEF KEY (2) = " LIST "
DEF KEY (3) = " AUTO "
DEF KEY (4) = " RENUM "
DEF KEY (5) = " COLOR "
DEF KEY (6) = " CHR$ ( "
DEF KEY (7) = " DEF KEY ( "
DEF KEY (8) = " CONT "
DEF KEY (9) = " SAVE "
DEF KEY (10) = " LOAD "
READY
```



## 2.3.4 Assignment statement

### LET

#### Format

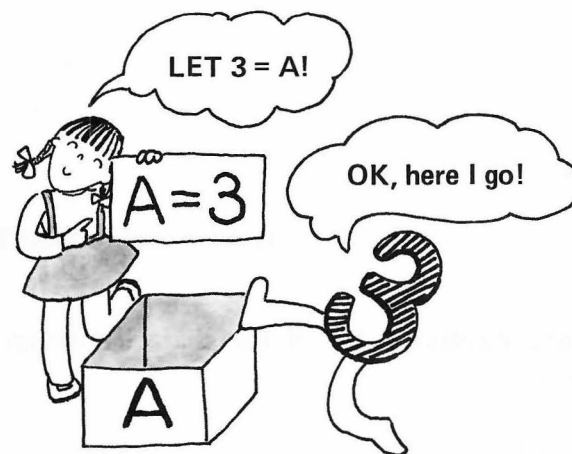
LET  $v = e$  or  $v = e$

$v$  . . . Numeric variable or array element, or string variable or array element.

$e$  . . . Numeric expression (consisting of one or more constants, variables, or array elements) or string expression (consisting of one or more constants, variables, or array elements).

#### Function

This statement assigns the value (numeric or string) specified by  $e$  to the variable or array element specified by  $v$ . As shown in the examples below, LET may be omitted.



#### Example

```
1Ø A=1Ø
2Ø B=2Ø
3Ø A=A+B
4Ø PRINT A
5Ø END
```

```
1Ø LET A=1Ø
2Ø LET B=2Ø
3Ø LET A=A+B
4Ø PRINT A
5Ø END
```

RUN ↵

3Ø

.....The two programs above produce exactly the same result.

The following are examples of incorrect use of the LET statement.

2Ø A\$=A+B.....Invalid because different types of variables (string and numeric) are specified on either sides of the "=" sign.

2Ø LOG(LK)=LK+1.....Invalid because the left side of the statement is not an numeric variable or array element.

## 2.3.5 Input/output statements

Input/output statements are the means by which data is submitted to the computer for processing, and by which the results of processing are output to the TV screen or printer.

### 2.3.5.1 PRINT

<b>Format</b>	$\left\{ \begin{array}{c} \text{PRINT} \\ ? \end{array} \right\} \left\{ \begin{array}{c} \text{variable} \\ \text{constant} \\ \text{expression} \end{array} \right\} < \left\{ \begin{array}{c} ; \\ , \end{array} \right\} \left\{ \begin{array}{c} \text{variable} \\ \text{constant} \\ \text{expression} \end{array} \right\} \dots \dots \dots$
---------------	--

<b>Function</b>	<p>This statement outputs the values of variables, constants, character strings, or expressions to the display screen. Values are displayed starting at the cursor's current location on the screen. (To move the cursor down one line on the screen, execute the PRINT statement without specifying any variables, constants, or expressions.)</p>
-----------------	---

To simplify key input when entering this statement, a question mark (?) may be typed instead of the word PRINT.

Numeric data is displayed by this statement in one of two formats: real number format or exponential format.

#### Read number format

Numeric values in the range from  $1 \times 10^{-8}$  to  $1 \times 10^8$  are displayed in real number format.

```
-1. 9999
63598757
0. 00000001 .....1 x 10-8
99999999
```







#### Exponential format

Numbers which cannot be displayed in real number format are displayed in exponential format.

```
- .31415E+9 ..... -0.31415 x 109
.513606E-20 ..... 0.513606 x 10-20
```

A plus (+) or minus (-) sign is always displayed ahead of the exponent (the number following "E") of a number displayed in exponential format.

Some special methods of using the PRINT statement are shown below.

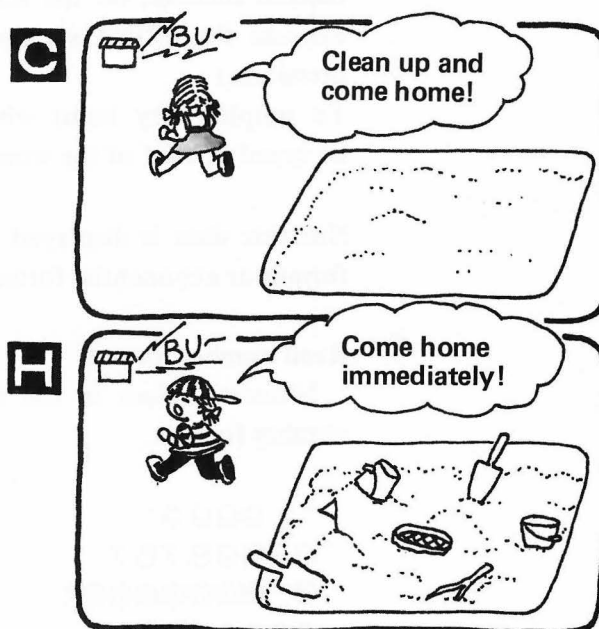
PRINT "  "	Clears the entire screen and moves the cursor to the home position (the upper left corner of the screen).
PRINT "  "	Moves the cursor to the home position without clearing the screen.
PRINT "  "	Moves the cursor one column to the right.
PRINT "  "	Moves the cursor one column to the left.
PRINT "  "	Moves the cursor up one line.
PRINT "  "	Moves the cursor down one line.

PRINT "C↓↓↓↓↓A" ..... Clears the screen, then displays the character "A" at the beginning of the sixth line from the top.

**Note:** The vertical bars {...} in the format description indicate that any one of the enclosed items may be selected.

To enter the special characters for cursor control, press the **GRAPH** key; this places BASIC in the graphic character input mode and changes the form of the cursor to "█". Next, enter the characters as follows.

- █..... Press the **CLR** key.
- █..... Press the **HOME** key.
- █..... Press the **→** key.
- █..... Press the **←** key.
- █..... Press the **↑** key.
- █..... Press the **↓** key.



After entering a special character, press the **ALPHA** key to return from the graphic character input mode to the alphanumeric input mode.

## 2. 3. 5. 2 PRINT USING ..... (abbreviated format: ?USI.)

**Format** PRINT USING "format string"; variable { ; } variable ...

**Function** This statement displays data on the screen in a specific format. The format specification consists of a character or string of characters in quotation marks, and is specified immediately after the word USING as follows.

(1) Format specification strings for numeric values

(a) #

The number sign is used to specify the maximum number of digits to be displayed. If the number of digits in the number displayed is smaller than the number of # signs specified in "format string", numbers are right-justified in the field defined by that string.

(Example:)

10 A = 123

20 PRINT USING "####" ; A

RUN ↵

123



(b) .

A period may be included in a format string consisting of # signs to specify the position in which the decimal point is to be displayed. The number of # signs to the right of the decimal point specifies the number of decimal places to be displayed.

(Example:)

```
10 A = 12.345 : B = 6.789
```

```
20 PRINT USING "###.##" ; A
```

```
30 PRINT USING "###.##" ; B
```

```
RUN ↵
```

```
  12.34
```

```
  6.79
```

(c) ,

Commas may also be included in "format string" to indicate positions in which commas are to be displayed. Numbers are right-justified in the same manner as when # signs are used alone.

(Example:)

```
10 A = 6345123 : B = 987324
```

```
20 PRINT USING "#,###,###" ; A
```

```
30 PRINT USING "#,###,###" ; B
```

```
RUN ↵
```

```
6,345,123
```

```
  987,324
```

(d) + and -

A plus (+) or minus (-) sign may be included at the end of "format string" to specify that the sign of the number is to be displayed in that position instead of a space. For instance, PRINT USING "#####+" will cause the sign to be displayed immediately after the number. (PRINT USING "#####-" causes a minus sign to be displayed following the number if the number is negative; if the number is positive, only a space is displayed in that position.) Further, a plus sign may be specified at the beginning of a format string to indicate that the number's sign is to be displayed in that position regardless of whether it is positive or negative.

(Examples)

```
PRINT USING "#####+" ; -13
```

```
  13-
```

```
PRINT USING "+#####" ; 25
```

```
 +25
```

(Note:)

Although a minus sign will be displayed if one is specified at the beginning of the format string, it will have no relationship to the sign of the number.

(e) \*\*

Specifying a pair of asterisks at the beginning of the format string indicates that asterisks are to be displayed in the positions of leading zeros.

(Example:)

```
10 A = 1234
20 PRINT USING " **####" ; A
RUN J
**1234
```

(f) ££

Specifying a pair of pound signs at the beginning of the format string indicates that a pound sign is to be displayed in the position immediately to the left of the number.

(Example:)

```
10 A = 123
20 PRINT USING " ££#### " ; A
RUN J
_ _ £123
```

(g) ↑↑↑↑

Four exponential operators may be included at the end of a format string to control display of numbers in exponential format.

(Example:)

```
10 A = 51 123
20 PRINT USING " ##.### ↑↑↑↑ " ; A
RUN J
_ 5.112E+04
```

In this case, the first number sign is reserved for display of the sign of the number.

(h) Extended list of operands

A list of variables may be specified following a single PRINT USING statement by separating them from each others with commas or semicolons. When this is done, the format specified in "format string" is used for display of all resulting values.

(Example:)

```
10 A = 5.3 : B = 6.9 : C = 7.123
20 PRINT USING " ##.### " ; A, B, C
RUN J
_ 5.300 _ 6.900 _ 7.123
```

---

(2) Format specification for string values

(a) !

When the values being displayed are character strings, specifying an exclamation mark in "format string" causes just the first character of the string specified to be displayed.

(Example:)

```
10 A$ = "CDE"  
20 PRINT USING "!" ; A$  
RUN  
C
```

(b) & \_ \_ \_ \_ &

Specifying "& \_ \_ \_ \_ &" in the format string causes the first 2 + n characters of specified string expressions to be displayed (where n is the number of spaces between the two ampersands). If fewer than 2 + n characters are specified in a string expression, characters displayed are left-justified in the field defined by "& \_ \_ \_ \_ &".

(Examples:)

```
10 A$ = "ABCDEFGH"  
20 PRINT USING "& _ _ _ _ &" ; A$  
RUN  
ABCDEF  
10 A$ = "XY"  
20 PRINT USING "& _ _ _ _ &" ; A$  
RUN  
XY
```

(3) String constant output function

When any character other than those described above is included in the format string of a PRINT USING statement, that character is displayed together with the value specified following the semicolon.

(Example:)

```
10 A = 123  
20 PRINT USING "DATA####" ; A  
RUN  
DATA_123
```

(4) Separation of USING

Usually, PRINT and USING are specified adjacent to each other; however, it is possible to use them separately within the same statement.

(Example:)

```
10 A = -12 : B = 14 : C = 12  
20 PRINT A; B; USING "####" ; C  
      Normal PRINT function    USING function  
RUN  
-12_14__12
```

### 2. 3. 5. 3 INPUT ..... (abbreviated format: I.)

#### Format

**INPUT** { numeric variable  
string variable  
array element } ... or **INPUT** "character string"; { numeric variable  
string variable  
array element } ...

```
INPUT A
INPUT B$
INPUT X(5)
```

```
INPUT "DATA A=" ; A
INPUT "YES OR NO" ; B$
INPUT "KEY IN" ; X (5)
```

#### Function

**INPUT** is one of the statements which is used for entering values for assignment to variables during program execution. Program execution pauses when an **INPUT** statement is encountered to allow values to be typed in from the keyboard. After input has been completed, the values are substituted into specified variables by pressing the **CR** key, then program execution resumes.

(Example:)

```
10 INPUT A, B
20 C=A+B
30 PRINT C
40 END
```

When the program above is executed, a question mark is displayed and the cursor blinks to indicate that the computer is waiting for data input; enter any arbitrary number, then press the **CR** key. This assigns the value entered to variable A.

After doing this, the question mark will be displayed again. The reason for this is that two variables (A and B) are specified in the **INPUT** statement on line 10, but only one value has been entered (that which is substituted into variable A). Enter another arbitrary number and press the **CR** key again; this substitutes the second value entered into variable B and causes execution to go on to the next line of the program. In the example above, subsequent lines add the values of A and B, substitute the result into C, then display the contents of C.

Since the variables used in this example are numeric variables, the computer will display the message **ILLEGAL DATA ERROR** if an attempt is made to enter any characters other than numerics. The question mark is then redisplayed to prompt the user to reenter a legal value (a value whose type is the same as that of the variable or array element into which it is to be substituted). Be sure to enter data whose type matches that of the variable(s) specified in the **INPUT** statement.

During program execution, it may be difficult to remember what data is to be entered when the question mark is displayed; therefore, prompt strings are usually included in **INPUT** statements for display on the screen as a reminder. This is done as shown in the program example below.

```
10 INPUT "A=" ; A
20 INPUT "B=" ; B
30 PRINT "A+B=" ; A+B
40 PRINT "A-B=" ; A-B
50 PRINT "A*B=" ; A*B
60 PRINT "A/B=" ; A/B
70 END
```

Try running the program shown above. Inclusion of character strings in the PRINT and INPUT statements provides a clear indication of the program's operation. Practical computer programs consist of combinations of sequences similar to the one shown here. By combining commands, statements, and sequences in different manners, you will soon find that there are many different methods of achieving a desired result.

#### 2. 3. 5. 4 GET

##### Format

GET v

v . . . . . Numeric variable or array element, or string variable or array element.

##### Function

When this statement is encountered during program execution, the BASIC interpreter checks whether any key on the keyboard is being pressed and, if so, assigns the corresponding value to the variable specified in v. Whereas the INPUT statement prompts for entry of data and waits until that data has been entered before resuming execution, the GET statement continues execution regardless of whether any key is being pressed.

Although data is substituted into variable v by the GET statement if any keys are pressed when the statement is executed, the variable will be left empty (0 for a numeric variable or null for a string variable) if no keys are pressed.

With numeric variables, this statement allows a single digit (from 0 to 9) to be entered; with string variables, it allows a single character to be entered.

This statement can be extremely useful when you want to enter data without pressing the CR key, as is often the case with game programs.

(Example:)

```
10 PRINT "NEXT GO? (Y OR N) "
20 GET A$
30 IF A$="Y" THEN 50
```

In the example above, execution jumps from line 30 to line 50 if the value of variable A\$ is "Y".

```
40 GOTO 20
50 PRINT "PROGRAM END "
60 END
```

Line 40 unconditionally transfers execution to line 20.

This program displays the prompt "NEXT GO? (Y OR N)" and waits for input. When the Y key is pressed, execution moves to line 50 and the program ends. Until that time, however, execution loops repeatedly between lines 20 and 40. Now delete lines 30 and 40 and try executing the program again. As you can see, execution is completed immediately regardless of whether any keys have been pressed.

**Note:** When GET statements are executed in succession, a routine should be included between them to ensure that each is completed before going on to the next. The reason for this is that key chatter (vibration of the contacts of the key switches) may result in two GET statements being executed simultaneously.

**2.3.5.5 READ ~ DATA** ..... (abbreviated format: REA. ~ DA.)

### Format

## READ

$$\left\{ \begin{array}{l} \text{numeric variable} \\ \text{string variable} \\ \text{array element} \end{array} \right\}, \left\{ \begin{array}{l} \text{numeric variable} \\ \text{string variable} \\ \text{array element} \end{array} \right\}, \dots$$

## DATA

$$\left\{ \begin{array}{l} \text{numeric constant} \\ \text{string constant} \end{array} \right\}, \left\{ \begin{array}{l} \text{numeric constant} \\ \text{string constant} \end{array} \right\}, \dots$$

Function

Like the INPUT and GET statements, the READ statement is used to submit data to the computer for processing. However, unlike the INPUT and GET statements, data is not entered from the keyboard, but is stored in the program itself in DATA statements. More specifically, the function of the READ statement is to read successive items of data into variables from a list of values which follows a DATA statement. When doing this, there must be a one-to-one correspondence between the variables of the READ statements and the data items specified in the DATA statements.

### Example

(Example 1)

```
10 READ A, B, C, D
20 PRINT A;B;C;D
30 END
40 DATA 10, 100, 50, 60
RUN J
```

10 100 50 60 ..... In this example, values specified in the DATA statement are read into variables A, B, C, and D by the READ statement, then the values of those variable are displayed.

(Example 2)

```
10 READ X$, A1, Z$
20 PRINT X$;A1;Z$
30 END
40 DATA A, 1, C .....
```

As shown by the example below, string data included in DATA statements does not need to be enclosed in quotation marks.

RUN ↵

A\_1C

.....The READ statement in this example picks successive data items from the list specified in the DATA statement, then substitutes each item into the corresponding variable in the list following the READ statement.



---

(Example 3)

```
10 DIM A (2)
20 READ A (0) , A (1) , A (2)
30 PRINT A (0) ; A (1) ; A (2)
40 END
50 DATA 3, 4, 5
RUN
3 4 5
```

.....The READ statement in this program substitutes the numeric values following the DATA statement into array elements A(0), A(1), and A(2), then the PRINT statement on line 30 displays the values of those array elements.

(Example 4)

```
10 READ A
20 READ B
30 DATA X
```

..... The example above is incorrect because (1) a numeric variable is specified by the READ statement on line 10, but the value specified following the DATA statement is a string value, and (2) there is no data which can be read by the READ statement on line 20.

## 2.3.5.6 RESTORE ..... (abbreviated format: ... RES.)

**Format**

**RESTORE or RESTORE Ln**

**Function**

When READ statements are executed, a pointer managed by the BASIC interpreter is incremented to keep track of the next item of data to be read from DATA statements. The RESTORE statement resets this pointer to (1) the beginning of the first DATA statement in the program or (2) the beginning of the DATA statement on a specified line.

**Example**

```
10 DATA 1, 2, 3
20 DATA "AA", "BB"
30 READ X, Y
40 READ Z, V$
.....
100 RESTORE
110 READ A, B, C, D$, E$
.....
200 READ I, J
210 RESTORE
220 READ M, N
230 RESTORE 260
240 READ O, P
250 DATA 1, 2, 3, 4
260 DATA -1, -2, -3, -4
```

An error will result if the number specified in Ln is the number of non-existent line.

```
10 X=33*RND(1)
20 FOR A=1 TO 5
30 READ M$
40 PRINT TAB(0); "◆"; TAB(X); M$;
50 PRINT TAB(37); "◆"
60 NEXT A
70 Y=10*RND(1)
80 FOR A=1 TO Y
90 PRINT TAB(0); "◆";
100 PRINT TAB(37); "◆":NEXT
110 RESTORE:GOTO 10
120 DATA "▲●▲", "●■●■●"
130 DATA "■●■", "●■●■●"
140 DATA "▲▲▲"
```

This function creates random numbers (see page 72 ).

**Note:** See page 62 for the TAB function and page 47 for the FOR ... NEXT statement.

## 2.3.6 Loop and branch instructions

### 2.3.6.1 FOR ~ NEXT ..... (abbreviated format: F. ~ N.)

#### Format

**FOR** cv = iv **TO** fv < **STEP** sv >

.....  
**NEXT** < cv >

cv .... Control variable; a numeric variable or array element.

iv .... Initial value; a numeric expression.

fv .... Final value; a numeric expression.

sv .... Increment, or step value; a numeric expression (if omitted, 1 is assumed).

#### Function

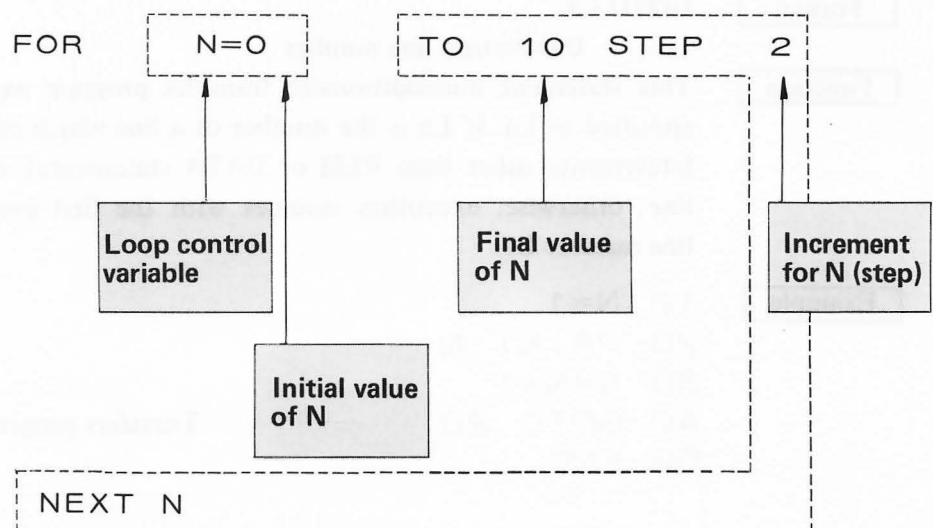
This statement repeats the instructions between **FOR** and **NEXT** a certain number of times.

```
10 A=0
20 FOR N=0 TO 10 STEP 2
30 A=A+1
40 PRINT "N=" ; N,
50 PRINT "A=" ; A
60 NEXT N
```

- (1) In the program above, 0 is assigned to N as the initial value.
- (2) Next, lines 20 through 50 are executed and the values of variables A and N displayed.
- (3) In line 60, the value of N is increased by 2, after which the BASIC interpreter checks to see whether N is greater than 10, the final value. If not, lines following line 20 are repeated.

When the value of N exceeds 10, execution leaves the loop and subsequent instructions (on lines following line 60) are executed. The program above repeats the loop 6 times.

If < **STEP** sv > is omitted from the statement specification, the value of N is increased by 1 each time the loop is repeated. In the case of the program above, omitting < **STEP** sv > in this manner would result in 11 repetitions of the loop.



FOR . . . NEXT loops may be nested within other FOR . . . NEXT loops. When doing this, inner loops must be completely included within outer ones. Further, separate control variables must be used for each loop.

**Example**

```

10 FOR X=1 TO 9
20 FOR Y=1 TO 9
30 PRINT X*Y;
40 NEXT Y
50 PRINT
60 NEXT X
70 END

```

FOR A=1 TO 3  
 FOR B=1 TO 5  
 FOR C=1 TO 7  
 .....  
 NEXT C  
 NEXT B  
 NEXT A

} NEXT C, B, A

When loops C, B, and A all end at the same point as in the example above, one NEXT statement may be used to indicate the end of all the loops.

Incorrect example:

```

FOR J=1 TO 10
FOR J=K TO K+5
NEXT J

```

```

FOR I=1 TO 10
FOR J=K TO K+5
NEXT I
NEXT J

```

× Different control variables must be used in each loop.

× Loops may not cross one another.

**Note**

The syntax of BASIC does not limit the number of levels to which loops may be nested; however, space is required to store return addresses for each level, so the number of levels is limited by the amount of available free space.

The CLR statement (see page 59) cannot be used within a FOR . . . NEXT loop.

## 2. 3. 6. 2 GOTO . . . . . (abbreviated format: . . . G.)

**Format**

**GOTO Ln**

Ln . . . . Destination line number

**Function**

This statement unconditionally transfers program execution to the line number specified in Ln. If Ln is the number of a line which contains executable statements (statements other than REM or DATA statements), execution resumes with that line; otherwise, execution resumes with the first executable statement following line number Ln.

**Example**

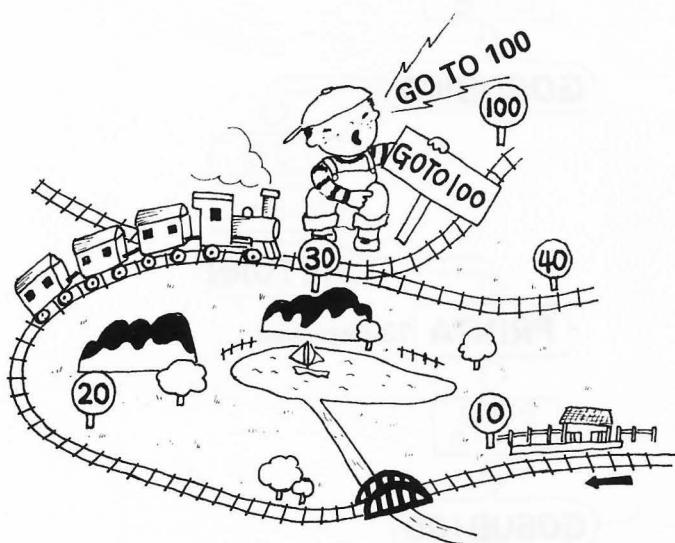
```

10 N=1
20 PRINT N
30 N=N+1
40 GOTO 20
50 END

```

.....Transfers program execution to line 20.

Since execution of the program shown above will continue indefinitely, stop it by pressing the **SHIFT** and **BREAK** keys together (this may be done at any time to stop execution of a BASIC program). To resume execution, execute the **CONT** command.



**Note** The line number specified in a GOTO statement may not be that of a line included within a FOR ... NEXT loop.

### 2. 3. 6. 3 GOSUB ~ RETURN ..... (abbreviated format: GOS. ~ RET.)

**Format** GOSUB Ln  
.....  
RETURN

Ln ... Destination line number

**Function** The GOSUB statement unconditionally transfers program execution to a BASIC subroutine beginning at the line number specified in Ln; after execution of the subroutine has been completed, execution is returned to the statement following GOSUB when a RETURN statement is executed.

GOSUB ~ RETURN statements are frequently used when the same processing is required at several different points in a program. In such cases, a subroutine which performs this processing is included at some point in the program, and execution is branched to this subroutine at appropriate points by means of the GOSUB statement. After the required processing has been completed, execution is returned to the main routine by the RETURN statement.

**Example**

```

100 X=10
110 GOSUB 200
120 PRINT X
130 END
200 X=X*2
210 RETURN
  
```

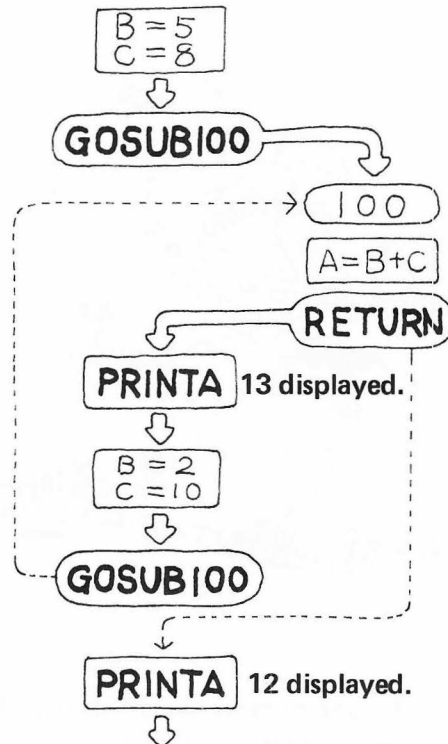
**Note**

The syntax of BASIC imposes no limit on the extent to which subroutines can be nested (that is, on the number of levels of subroutine calls which can be made from other subroutines); however, in practice a limitation is imposed by the amount of free space in memory which is available for storing return addresses.

```

10 B=5
20 C=8
30 GOSUB 100
40 PRINT A
50 B=2
60 C=10
70 GOSUB 100
80 PRINT A
90 END
100 A=B+C
110 RETURN

```



#### 2. 3. 6. 4 IF ~ THEN ..... (abbreviated format: ... IF ~ TH.)

**Format**

**IF e THEN Ln**

**IF e THEN statement**

e: A relational expression or logical expression

Ln: Destination line number

**Function**

IF ... THEN statements are used to control branching of program execution according to the result of a logical or relational expression. When the result of such an expression is true, statements following THEN are executed. If a line number is specified following THEN, program execution jumps to that line of the program if the result of the expression is true.

If the result of the logical or relational expression is false, execution continues with the program line following that containing the IF ... THEN statement.

IF	Condition	THEN	Statement or line number
----	-----------	------	--------------------------

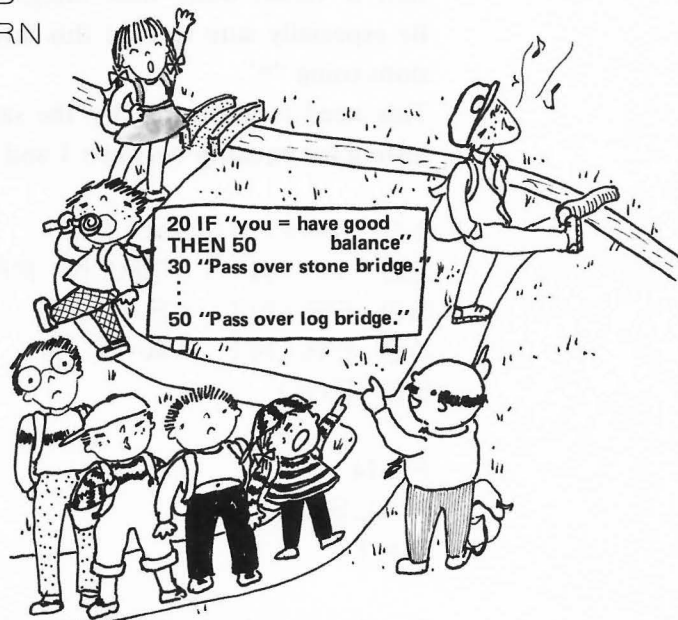


**Example**

```

IF.....THEN 100
IF.....THEN GOTO or IF.....GOTO
IF.....THEN PRINT or IF.....THEN ?
IF.....THEN A=5*7 assignment
IF.....THEN I=10:J=50
IF.....THEN INPUT
IF.....THEN READ
IF.....THEN GOSUB
IF.....THEN RETURN
IF.....THEN STOP
IF.....THEN END

```

**Examples of logical and relational expressions**

	Operator	Sample application	Explanation
Relational expressions	=	IF A=X THEN...	If the value of numeric variable A equals the value of X, execute the statements following THEN.
		IF A\$="XYZ" THEN...	If the contents of string variable A\$ equal "XYZ", execute the statements following THEN.
	>	IF A>X THEN...	If the value of variable A is greater than X, execute the statements following THEN.
	<	IF A<X THEN...	If the value of variable A is less than X, execute the statements following THEN.
	<> or >>	IF A<>X THEN...	If the value of variable A is not equal to X, execute the statements following THEN.
	>= or =>	IF A>=X THEN...	If the value of variable A is greater than or equal to X, execute the statements following THEN.
Logical expressions	<= or =<	IF A<=X THEN...	If the value of variable A is less than or equal to X, execute the statements following THEN.
	*	IF (A>X)*(B>Y) THEN...	If the value of variable A is greater than X and the value of variable B is greater than Y, execute the statements following THEN.
Logical expressions	+	IF (A>X)+(B>Y) THEN...	If the value of variable A is greater than X or the value of variable B is greater than Y, execute the statements following THEN.

---

**Note**

Precautions on comparison of numeric values with BASIC 1Z-013B, numeric values are internally represented in binary floating point representation; since such values must be converted to other forms for processing or external display (such as in decimal format with the PRINT statement), a certain amount of conversion error can occur.

For example, when an arithmetic expression is evaluated whose mathematical result is an integer, an integer value may not be returned upon completion of the operation if values other than integers are handled while calculations are being made. Be especially sure to take this into consideration when evaluating relational expressions using “=”.

This need is illustrated by the sample program below, which returns FALSE after testing for equality between 1 and  $1/100 \times 100$ .

```
10 A=1/100*100
20 IF A=1 THEN PRINT "TRUE":GOTO 40
30 PRINT "FALSE"
40 PRINT "A=";A
50 END
```

```
RUN
FALSE
A=1
```

The fact that both “FALSE” and “A = 1” are displayed as the result of this program shows that external representation of numbers may differ from the number’s internal representation.

Therefore, a better method of checking for equality in the program example above is as follows.

```
20 IF ABS(A-1) < .1E-8 THEN PRINT "TRUE":
   GOTO 40
```

### 2.3.6.5 IF ~ GOTO ..... (abbreviated format: IF ~ G.)

**Format** IF e GOTO Lr

e: Relational expression or logical expression

Lr: Destination line number

**Function**

This statement sequence evaluates the condition defined by relational or logical expression e, then branches to the line number specified in Lr if the condition is satisfied. As with the IF . . . THEN sequence, IF ~ GOTO is used for conditional branching; when the specified condition is satisfied, program execution jumps to the line number specified in Lr. If the condition is not satisfied, execution continues with the next line of the program. (Any statements following IF ~ GOTO on the same program line will be ignored.)

**Example**

```
10 G=0:N=0
20 INPUT "GRADE=" ; X
30 IF X=999 GOTO 100
40 T=T+X:N=N+1
50 GOTO 20
100 PRINT "-----"
110 PRINT "TOTAL:" ; T
120 PRINT "NO. PEOPLE:" ; N
130 PRINT "AVERAGE:" ; T/N
140 END
```

### 2.3.6.6 IF ~ GOSUB ..... (abbreviated format: IF ~ GOS.)

**Format** IF e GOSUB Lr

e: Relational expression or logical expression

Lr: Destination line number

**Function**

This statement evaluates the condition defined by relational or logical expression e, then, if the condition is satisfied, branches to the subroutine beginning on the line number specified in Lr. Upon completion of the subroutine, execution returns to the first executable statement following the calling IF ~ GOSUB statement; therefore, if multiple statements are included on the line with the IF ~ GOSUB statement, execution returns to the first statement following IF ~ GOSUB.

**Example**

```
10 INPUT " X=" ; X
20 IF X<0 GOSUB 100:PRINT "X<0"
30 IF X=0 GOSUB 200:PRINT "X=0"
40 IF X>0 GOSUB 300:PRINT "X>0"
50 PRINT "-----"
60 GOTO 10
100 PRINT " * PROGRAM LINE 100 " : RETURN
200 PRINT " * PROGRAM LINE 200 " : RETURN
300 PRINT " * PROGRAM LINE 300 " : RETURN
```

### 2.3.6.7 ON~GOTO ..... (abbreviated format: ON~G.)

#### Format

**ON e GOTO**  $Lr_1 <, Lr_2, Lr_3, \dots, Lr_i >$

e ... Numeric variable, array element, or expression

$Lr_i$  . List of destination line numbers

#### Function

This statement branches execution to one of the line numbers following GOTO, depending on the value of e.

The value of e indicates which of the line numbers following GOTO is to be used for making the branch; in other words, if e is 1, execution branches to the first line number in the list; if e is 2, execution branches to the second line number in the list; and so forth. For example:

```
100 ON A GOTO 200, 300, 400, 500
```

Destination when

A is 1  
A is 2  
A is 3  
A is 4

#### Example

```
10 INPUT "NUMBER " : A
20 ON A GOTO 50, 60, 70
50 PRINT "XXX" : GOTO 10
60 PRINT "YYY" : GOTO 10
70 PRINT "ZZZ" : GOTO 10
RUN
NUMBER ? 1
XXX
NUMBER ? 2
YYY
NUMBER ?
```

If a decimal number such as 1.2 is specified, the decimal portion is truncated before evaluating the statement.

#### Note

When the value of e in an ON~GOTO statement is greater than the number of line numbers specified following GOTO, execution continues with the next line of the program.

This also applies if the value of e is less than 1 or negative.

Further, if the value of e is a non-integer, the decimal portion is truncated to obtain an integer value before the statement is evaluated.

---

### 2.3.6.8 ON~GOSUB ..... (abbreviated format: ON~GOS.)

**Format**

**ON e GOSUB**  $Lr_1 <, Lr_2, Lr_3, \dots, Lr_i >$

e ... Numeric variable, array element, or expression

$Lr_i$  . Destination line numbers

**Function**

This statement branches execution to the subroutine beginning on one of the line numbers following GOSUB, depending on the value of e. Operation of this statement is basically the same as with the ON~GOTO statement, but all branches are made to subroutines. Upon return from the subroutine, execution resumes with the first executable statement following the ON~GOSUB statement which made the call.

**Example**

Let's try using the ON~GOSUB statement in a scheduling program. The most important point to note in the following program is that, a subroutine call is made at line 180, even though line 180 itself is part of a subroutine (from line 170 to 190) which is called by line 90. Subroutines can be nested to many levels in this manner.

```
10 A$=" ENGL " : B$=" MATH " : C$=" FREN "
20 D$=" SCI " : E$=" MUS " : F$=" GYM "
30 G$=" HIST " : H$=" ART " : I$=" GEOG "
40 J$=" BUS " : K$=" H RM "
50 INPUT "WHAT DAY? "; X$
60 FOR Z=1 TO 7: Y$=MID$("SUNMONTUEWEDTHU
FRISAT", 1+3*(Z-1), 3) : IF Y$=X$ THEN X=Z
70 NEXT Z
80 FOR Y=0 TO 4: PRINT TAB(5+6*Y); Y+1;
90 NEXT Y: PRINT
100 ON X GOSUB 180, 120, 130, 140, 150, 160, 170
110 PRINT: GOTO 50
120 PRINT "MON " ; A$; B$; D$; G$; K$: RETURN
130 PRINT "TUE " ; B$; E$; H$; H$; D$: RETURN
140 PRINT "WED " ; C$; C$; I$; A$; F$: RETURN
150 PRINT "THU " ; B$; D$; F$; G$; E$: RETURN
160 PRINT "FRI " ; A$; D$; I$; C$; C$: RETURN
170 PRINT "SAT " ; B$; G$; D$; K$: RETURN
180 FOR Y=1 TO 6
190 ON Y GOSUB 120, 130, 140, 150, 160, 170
200 PRINT: NEXT Y
210 RETURN
```

---

## 2.3.7 Definition statements

### 2.3.7.1 DIM

Format
--------

**DIM**  $a_1 (i_1) <, a_2 (i_2), \dots, a_i (i_m) >$   
**DIM**  $b_1 (i_1, j_1) <, b_2 (i_2, j_2), \dots, b_i (i_n, j_n) >$   
 $a_i \dots$  1-dimensional array name (list)  
 $b_i \dots$  2-dimensional array name (table)  
 $i_m, i_n, j_n \dots$  Dimensions

Function
----------

This statement is used to declare (define) arrays with from one to four dimensions and to reserve space in memory for the number of dimensions declared (DIM: dimension). Up to two characters can be specified as the array name, and subscripts of any value may be specified to define the size of dimensions; however, the number of dimensions which can be used is limited in practice by the amount of free memory available.

Example
---------

(Examples:)  
10 DIM A (100)  
20 FOR J=0 TO 100  
30 READ A (J)  
40 NEXT J  
50 DATA 5, 30, 12, .....

(Examples:)  
10 DIM A\$ (1), B\$ (1), C\$ (1)  
20 FOR J=0 TO 1 : READ A\$ (J), B\$ (J)  
30 C\$ (J) = A\$ (J) + " " + B\$ (J)  
40 PRINT A\$ (J), B\$ (J), C\$ (J)  
50 NEXT J  
60 END  
70 DATA YOUNG, GIRL, WHITE, ROSE

Note
------

Execution of the DIM statement sets the values of all elements of declared arrays to 0 (for numeric arrays) or null (for string arrays). Therefore, this statement should be executed before values are assigned to arrays.

Different names must be used for each array which is declared; for example, the instruction DIM A(5), A(6) is not a legal array declaration.

All array declarations are nullified by execution of a CLR statement (see page 59) and a NEW statement (see page 32).

### 2.3.7.2 DEF FN

Format
--------

**DEF FN**  $f (x) = e$   
 $f \dots$  Name assigned to the function being defined (one uppercase letter from A to Z)  
 $x \dots$  Argument (variable name)  
 $e \dots$  Numeric expression (constant, variable, array element, or function) or previously defined user function

Function
----------

The DEF FN statement is used to define user function FN  $f (x)$ . Such functions consist of combinations of functions which are intrinsic to BASIC.



**Example**

DEF FNA (X) = 2 \* X ↑ 2 + 3 \* X + 1 ..... Defines  $2X^2 + 3X + 1$  as FNA (X).

DEF FNE (V) = 1/2 \* M \* V ↑ 2 ..... Defines  $1/2MV^2$  as FNE (V).

10 DEF FNB (X) = TAN (X - PAI (1) / 6)

20 DEF FND (X) = FNB (X) / C + X ..... Defines function FNB using the function defined on line 10.

(Incorrect definitions)

10 DEF FNK (X) = SIN (X/3 + PAI(1)/4), FNL (X) = EXP(-X ↑ 2/K)

.... Only one user function can be defined by a single DEF FN statement.

Find the kinetic energy of a mass of 5.5 when it is imparted with initial accelerations of 3.5,  $3.5 \times 2$ , and  $3.5 \times 3$ .

10 DEF FNE (V) = 1/2 \* M \* V ↑ 2

20 M = 5.5 : V = 3.5

30 PRINT FNE (V), FNE (V \* 2), FNE (V \* 3)

40 END

**Note**

All user function definitions are cleared when the CLR statement and the NEW statement is executed.

**2.3.7.3 DEF KEY****Format**

DEF KEY (k) = S\$

k ..... Definable function key number (1 to 10)

S\$ ..... Character string (up to 15 characters).

**Function**

Character strings can be assigned to any of the ten function keys to allow strings to be entered at any time just by pressing a single key. This statement is used to define such strings and assign them to the definable function keys. Function key numbers 1 to 5 are entered just by pressing the corresponding key at the top left corner of the keyboard; keys 6 to 10 are entered by pressing the **SHIFT** key together with the corresponding key. The function key number (1 to 10) is specified in k, and the string or command which is to be assigned to the key is specified exactly as it is to be entered in S\$. Execution of the DEF KEY statement cancels the previous definition of the definable function key.

No other statement can be specified after a DEF KEY statement on the same line.

(Example:)

10 DEF KEY (1) = " INPUT " ..... Defines key **F1** as INPUT

20 DEF KEY (2) = " RUN " + CHR\$(13) ... Defines **F2** as RUN ↵

**Note:** CHR\$(13) indicates the ASCII code for **CR**, and specifying it together with the string assigned to a definable function key has the same effect as pressing the **CR** key. (See the description of the CHR\$ function on page 78 and the ASCII code table on page 154.)

## 2.3.8 Remark statement and control commands

### 2.3.8.1 REM

Format
--------

REM r

r . . . . Programmer's remark

Function
----------

REM is a non-executable statement which is specified in a program line to cause the BASIC interpreter to ignore the remainder of that line. Since REM statements are non-executable, they may be included at any point in the program without affecting the results of execution. REM statements are generally used to make a program easier to read, or to add explanatory notes to a program.

#### Multiple statement program lines

When more than one statement is included on a single program line, each statement must be separated from the one preceding it by a colon (:). Operation of the BASIC interpreter is generally the same in such cases as when the same statements are specified on different lines. For example, the two programs below produce exactly the same result.

10 A=5	}	→	10 A=5:B=8:C=A*B:PRINT C
20 B=8			
30 C=A*B			
40 PRINT C			

**Note:** Also note that program operation may differ when multiple statement lines are used as shown below.

10 INPUT A	}	This program displays 1 if the value entered at line 10 is greater than or equal to 100, and 0 if the value entered is less than 100.
20 B=0		
30 IF 99<A THEN B=1		
40 PRINT B		
50 END		

10 INPUT A:B=0:IF 99<A THEN B=1:PRINT B  
20 END

This program displays 1 if the value entered is greater than or equal to 100, but nothing at all if the value entered is less than 100. The reason for this is that statements following THEN on line 10 are not executed if the IF condition is not satisfied.

## 2. 3. 8. 2 STOP ..... (abbreviated format: S.)

Format
--------

### STOP

Function
----------

Temporarily stops program execution, displays BREAK and READY, then waits for entry of executable commands in the direct mode.

The STOP statement is used to temporarily interrupt program execution, and may be inserted at as many points and locations in the program as required. Since execution of the program is only interrupted temporarily, the PRINT statement can be used in the direct mode to check the values stored in variables, after which execution can be resumed by entering CONT J .

Example
---------

```
10 READ A, B
20 X=A*B
30 STOP
40 Y=A/B
50 PRINT X, Y
60 DATA 15, 5
70 END
RUN
BREAK IN 30
```

Note
------

Unlike the END statement, no files are closed by the STOP statement. (See page 68 concerning procedures for opening and closing of files.)

## 2. 3. 8. 3 END ..... (abbreviated format: E.)

Format
--------

### END

Function
----------

The END statement terminates program execution and returns the BASIC interpreter to the command mode for input of direct mode commands. When this statement is executed, READY is displayed to indicate that the BASIC interpreter is ready. After the END statement has been executed, execution cannot be resumed by executing the CONT command even if there are executable statements on program lines following the END statement.

Note
------

All open files are closed when the END statement is executed. (See page 68 concerning procedures for opening and closing files.)

### Differences between the STOP and END statements

	Screen display	Files	Resumption of execution
STOP	BREAK IN xxxx READY	Open files are not closed.	Can be resumed by executing CONT.
END	READY	Open files are closed	Cannot be resumed.

## 2. 3. 8. 4 CLR

Format
--------

### CLR

Function
----------

The CLR command clears all variables and cancels all array definitions. All numeric variables are cleared to 0, and null strings ( " ") are placed in all string variables; arrays are eliminated entirely by nullifying all previously executed DIM statements. Therefore, DIM statements must be executed to redefine the dimensions of required arrays before they can be used again.

The CLR command also cancels all function definitions made with the DEF FN statement; therefore, it is also necessary to reexecute DEF FN statements to redefine such functions before they can be used again.

**Note**

CLR statements cannot be included in a FOR~NEXT loop or BASIC subroutine.

## 2.3.8.5 TI\$

**Format**

TI\$ "hh mm ss"

**Function**

TI\$ is the name of the system string variable which contains the time of the computer's built-in clock.

This built-in variable is automatically incremented once each second, and the six character string contained in this variable indicates the hour, minute, and second, with two characters used for each. For example, if the string contained in TI\$ is "092035", the time is 9:20:35 A. M.

Variable TI\$ is automatically set to 00:00:00 when BASIC is loaded into the computer. To set the current time of day, use the string assignment statement. For example, the clock can be set to 7:00:00 P. M. by executing the following.

TI\$ = "190000"

The clock is set to 7:00:00 and then restarted automatically when the CR key is pressed.

The digits specified for the hour must be in the range from 00 to 23, and those specified for the minute and second must each be in the range from 00 to 59.

**Example**

The following program displays the current local time in various cities of the world.

```

10 PRINT "C"
20 DIM C$(10), D(10), E(10), T$(10)
30 FOR I=1 TO 10:READ C$(I), D(I):NEXT I
40 PRINT "ENTER NEW YORK TIME (HOUR, MINUT
E, SECOND) "
50 INPUT B$:TI$=B$:PRINT "C"
60 PRINT "H":T$(1)=TI$
70 FOR I=1 TO 10
80 E(I)=VAL(LEFT$(T$(1), 2))+D(I)
90 IF E(I)=24 THEN E(I)=0
100 IF E(I)<0 THEN E(I)=24+E(I)
110 T$(I)=STR$(E(I))+RIGHT$(T$(1), 4)
120 IF LEN(T$(I))=5 THEN T$(I)="0"+T$(I)
130 PRINT C$(I);TAB(15);LEFT$(T$(I), 2);
140 PRINT " ";MID$(T$(I), 3, 2);" ";RIGHT$(
T$(I), 2);
150 NEXT I:GOTO 60
160 DATA NEW YORK, 0, MOSCOW, 8, RIO DE JANE
IRO, 2
170 DATA SYDNEY, 15, HONOLULU, -5, LONDON, 5,
CAIRO, 7
180 DATA TOKYO, 14, SAN FRANCISCO, -3, PARIS
, 6

```

**Note**

The TI\$ variable cannot be specified in an INPUT statement. Further, after the time changes from 23:59:59 to 00:00:00, the time "00:00:01" is not displayed.

## 2.3.8.6 CURSOR ..... (abbreviated format: CU.)

**Format**

**CURSOR** x, y

x ... X coordinate (0 to 39)

y ... Y coordinate (0 to 24)

**Function**

This command is used to move the cursor to a specified position on the TV (display) screen, and can be used together with the PRINT and INPUT statements to display characters in any desired location.

In the system of screen coordinates used, the columns of the screen are numbered from left to right, starting with 0 on the left side and ending with 39 on the right side; lines of the screen are numbered from top to bottom, with 0 indicating the top line of the screen and 24 indicating the bottom line. Thus, the cursor can be moved to any desired position in the range from (0, 0), which indicates the top left corner of the screen, to (39, 24) indicates the bottom right corner.

**Example**

The following program moves an asterisk (\*) about on the screen as the cursor keys are pressed.

```

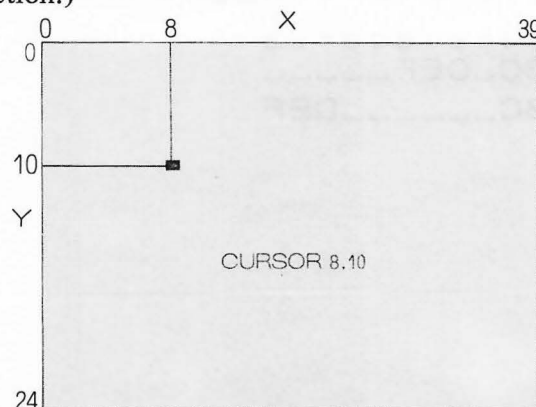
10 X=0:Y=0
15 PRINT "C"
20 CURSOR X,Y:PRINT "*";
30 GET A$:IF A$=" " THEN 30
40 CURSOR X,Y:PRINT " ";
50 IF A$="↑" THEN Y=Y-1 :REM "UP"
60 IF A$="↓" THEN Y=Y+1 :REM "DOWN"
70 IF A$="←" THEN X=X-1 :REM "LEFT"
80 IF A$="→" THEN X=X+1 :REM "RIGHT"
90 IF X<0 THEN X=0
100 IF Y<0 THEN Y=0
110 IF X>38 THEN X=38
120 IF Y>24 THEN Y=24
150 GOTO 20

```

**Note**

If the value specified for either X or Y is other than an integer, it is converted to an integer by truncating the decimal portion before the cursor is moved.

Other methods of moving the cursor which are used together with the PRINT statement include the TAB and SPC functions. (See page 62 for a description of the SPC function.)



### 2.3.8.7 TAB

#### Format

**TAB (x)**

x ... A numeric expression

#### Function

The TAB function is used together with the PRINT statement to move the cursor to the character position which is x + 1 positions from the left side of the screen. (This is referred to as space tabulation.)

#### Example

```
PRINT TAB (5) ; " XYZ " ; TAB (10) ; " ABC "
```

```
0 1 2 3 4 5 6 7 8 9 0 1 2 ← Not actually displayed.
   _ _ _ _ _ XYZ _ _ _ ABC
```

#### Note

Tabulation can only be used to move the cursor to the right; therefore, **nothing happens** if this function is used together with the PRINT statement when the cursor is already to the right of the character position specified in (x).

(Example:)

```
PRINT TAB (5) ; " XYZ " ; TAB (5) ; " ABC "
```

```
0 1 2 3 4 5 6 7 8 9 0
   _ _ _ _ _ XYZ ABC
```

### 2.3.8.8 SPC

#### Format

**SPC (n)**

n ... A numeric expression

#### Function

Use together with the PRINT statement, this function outputs a string of n spaces and thus moves the cursor n character positions to the right of its current position.

#### Example

(Example 1)

```
PRINT SPC (5) ; " ABC "
```

```
0 1 2 3 4 5 6 7
   _ _ _ _ _ ABC
```

(Example 2)

The following example illustrates the difference between the TAB and SPC functions.

```
10 ? TAB (2) ; " ABC " ; TAB (6) ; " DEF "
```

```
20 ? SPC (2) ; " ABC " ; SPC (6) ; " DEF "
```

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3
  _ _ ABC _ DEF _ _ _ _ _
  _ _ ABC _ _ _ _ _ DEF
```



### 2.3.8.9 SET, RESET

These statements are used to turn dots on or off at a specified position on the screen.

Format	Function	Range of X, Y coordinates
<b>SET X, Y, C &gt;</b> X ... Numeric expression specifying the X coordinate. Y ... Numeric expression specifying the Y coordinate. C ... Color code (0 to 7).	Turns on the dots at the screen coordinates specified by X and Y. (SET)	$0 \leq X \leq 79$
<b>RESET X, Y</b> X ... Numeric expression specifying the X coordinate. Y ... Numeric expression specifying the Y coordinate.	Turns off the dots at the screen coordinates specified by X and Y. (RESET)	$0 \leq Y \leq 49$

When a color code is specified, the color of the dots displayed by the SET statement is as follows.

- (0) ..... Black
- (1) ..... Blue
- (2) ..... Red
- (3) ..... Purple
- (4) ..... Green
- (5) ..... Light blue
- (6) ..... Yellow
- (7) ..... White

Since four dots are turned on simultaneously by the SET statement, changing the color of any one dot in that four dot group also causes the color of the other dots to change.

The SET and RESET statements can be used to produce a wide variety of interesting effects; some examples are introduced below.

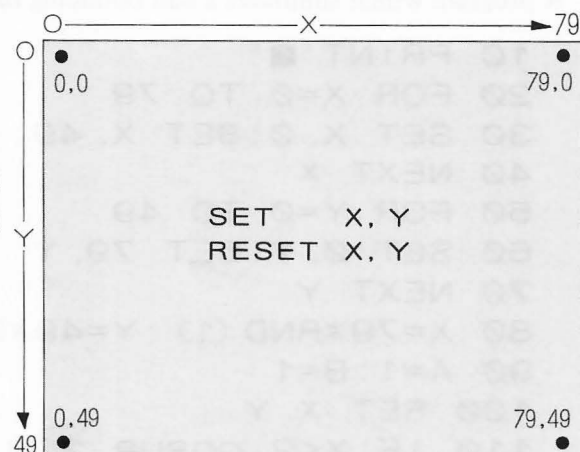
#### 1. Turning on one dot on the screen.

```

10 PRINT "■"
20 X=79:Y=49
30 SET X, Y, 2
40 RESET X, Y
50 GOTO 30

```

← Turns dots on.  
← Turns dots off.



#### 2. Coloring the entire screen white.

```

10 PRINT "■"
20 FOR X=0 TO 79
30 FOR Y=0 TO 49
40 SET X, Y, 7
50 NEXT Y, X
60 GOTO 10

```

3. Drawing a rectangle around the edge of the screen.

```
10 PRINT "■"  
20 FOR X=0 TO 79  
30 SET X, 0  
40 SET X, 49  
50 NEXT X  
60 FOR Y=0 TO 49  
70 SET 0, Y  
80 SET 79, Y  
90 NEXT Y  
100 GOTO 100
```

4. A program which simulates the ripples produced by throwing a pebble into a pond.

```
10 X=40:Y=25  
20 DEF FNY(Z)=SQR(R*R-Z*Z)  
30 PRINT "■":SET X, Y  
40 R=R+5  
50 FOR Z=0 TO R  
60 T=FNY(Z)  
70 SET X+Z, Y+T  
80 SET X+Z, Y-T  
90 SET X-Z, Y+T  
100 SET X-Z, Y-T  
110 NEXT Z  
120 IF R<>25 THEN 40  
130 GOTO 130
```

5. A program which simulates a ball bouncing off four walls.

```
10 PRINT "■"  
20 FOR X=0 TO 79  
30 SET X, 0:SET X, 49  
40 NEXT X  
50 FOR Y=0 TO 49  
60 SET 0, Y:SET 79, Y  
70 NEXT Y  
80 X=79*RND(1):Y=49*RND(1)  
90 A=1:B=1  
100 SET X, Y  
110 IF X<2 GOSUB 200  
120 IF X>78 GOSUB 200  
130 IF Y<2 GOSUB 250  
140 IF Y>48 GOSUB 250  
150 RESET X, Y  
160 X=X+A:Y=Y+B:GOTO 100  
200 A=-A:MUSIC"A0":RETURN  
250 B=-B:MUSIC"A0":RETURN
```

## 2.3.9 Music control statements

This section discusses the MUSIC and TEMPO statements which are used to control performance of music by the computer. As its name implies, the TEMPO statement specifies the speed with which music is performed. The notes (including half notes and upper and lower octaves) and duration of notes produced are controlled by the MUSIC statement.

Tempo:	Specified with TEMPO as a numeric variable or constant with a value from 1 (slow) to 7 (fast).			
Melody:	Specified with MUSIC as a string variable consisting of a collection of notes.			
Note specification:	octave	# (sharp)	note name	duration

### 2.3.9.1 MUSIC ..... (abbreviated format: MU.)

#### Format

**MUSIC X\$**

X\$ ... String data

Automatically performs music.

#### Discussion

This statement outputs the melody or sound effects specified by the character string or string variable of its argument to the speaker. The speed with which this melody is played is that which is specified with the TEMPO statement (see page 67).

The format for specification of each note is as follows:

< octave specification > < # (sharp) > note name < duration >

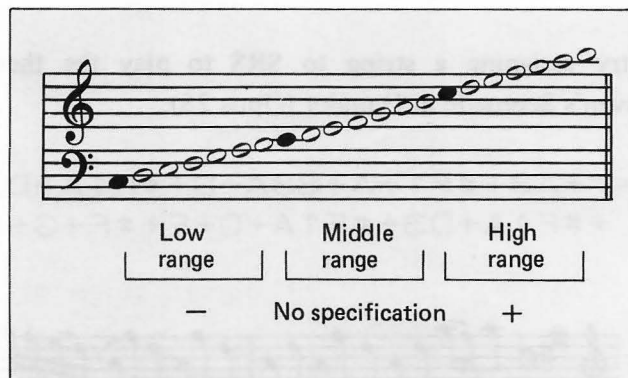
The plus or minus signs are used to specify the octave. If neither is specified, the middle range is assumed.

The three ranges of sounds which can be output by the computer are as shown in the figure below. For example, the C notes ("do" on the 8-note C scale) indicated by the black dots below are differentiated from each other by the octave specification.

Low C ..... -C

Middle C ..... C

High C ..... +C

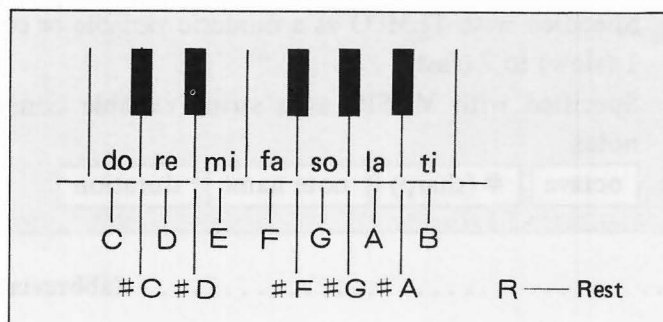


### Note specification

The symbols used to specify notes within each range are as follows:

CDEFGAB # R

The relationship between the 8-note scale (do, re, mi, fa, so, la, ti, do) and these symbols are as shown below. The sharp symbol (#) is used to specify half notes. Silent intervals are specified with "R".



### Duration specification

The duration specification determines the length of the specified note. The durations from 1/32 to whole are specified as numbers from 0 to 9. (When R is specified, this determines the length of the silent interval.)

1/32 rest	1/16 rest	Dotted 1/16 rest	1/4 rest	Dotted 1/8 rest	1/4 rest	Dotted 1/4 rest	1/2 rest	Dotted 1/2 rest	Whole rest
1/32 note	1/16 note	Dotted 1/16 note	1/8 note	Dotted 1/8 note	1/4 note	Dotted 1/4 note	1/2 note	Dotted 1/2 note	Whole note
0	1	2	3	4	5	6	7	8	9

When successive notes have the same duration, the duration specification can be omitted for the second and following notes. If no duration is specified for the first note, 1/4 notes are assumed.

### Sound volume

The volume of sound produced cannot be controlled by the program, but can be adjusted with the computer's external volume control.

#### Example

Let's try assigning a string to SR\$ to play the theme from the beginning of Beethoven's Serenade in D major (Opus 25).

```
SR$= "+A3+#F1+A+B3A+D+#F1A+D3A+D
      + #F1A+D3+ #F1A+D+E+ #F+G+A3R "
```



## 2.3.9.2 TEMPO ..... (abbreviated format: TEM.)

### Format

**TEMPO x**

x ... Numeric expression (1 to 7)

### Function

This statement sets the tempo with which music is played by the music statement. If this statement is not executed, TEMPO 4 is assumed for execution of MUSIC statements.

30 TEMPO 1 Slowest tempo .... (Lento, adagio)

30 TEMPO 4 Medium tempo .... (Moderato);

four times as fast as TEMPO 1.

30 TEMPO 7 Fastest tempo ..... (Motto allegro, presto);

seven times as fast as TEMPO 1.

### Example

10 REM Chopin's mazourka

20 MM\$= " A3 " : M1\$= " A5+#C3+D+E+#F+G+#FO+G+#  
F4+E3+D+#CB "

30 M2\$= " A3+D2RO+D1+E2+D+#C3B+#C7+#C3 "

40 M3\$= " A3+#C2RO+#C1+D2+#CB3A+D7+D3 "

50 TEMPO 3

60 MUSIC MM\$, M1\$, M2\$, M1\$, M3\$, M1\$, M2\$, M1  
\$, M3\$

70 END



## 2.3.10 Data file input/output commands

Although the SAVE and LOAD commands can be used to write or read program text, other commands are used to record or read the various types of data which is handled by programs. These commands are described below.

	Format	Function
<b>WOPEN</b> (abbreviated <b>W.</b> )	WOPEN < file name >	Opens a data file on cassette tape prior to writing data to it. This command also assigns a name to the data file.
<b>PRINT/T</b> (abbreviated <b>?/T</b> )	PRINT/T $d_1 <, d_2, d_3, \dots, d_n >$ $d_n \dots$ Numeric data or string data	Writes data to cassette tape in the same format as it would be displayed by the PRINT statement.
<b>ROPEN</b> (abbreviated <b>RO.</b> )	ROPEN < file name >	Searches for the data file on cassette tape with the specified name and opens that file to prepare for reading data from it.
<b>INPUT/T</b> (abbreviated <b>I./T</b> )	INPUT/T $v_1 <, v_2, v_3, \dots, v_n >$ $v_n \dots$ Numeric data or string data	Used to input data from a cassette file and pass it to the program (in a manner similar to that in which the INPUT statement is used to input data from the keyboard).
<b>CLOSE</b> (abbreviated <b>CLO.</b> )	CLOSE	Statement which closes cassette data files after writing or reading has been completed.

Unlike the LOAD and SAVE commands, no messages are displayed by execution of the WOPEN and ROPEN statements.

If display of a message is desired, use the PRINT statement to define one in the program.

**Note:** When an ordinary cassette recorder is used, it may not be possible to record data files even if no problems are encountered in storing or reading programs with the SAVE and LOAD commands.

### (Example 1)

The following program writes the numbers from 1 to 99 on cassette tape.

```
10 WOPEN "DATA "  
20 FOR X=1 TO 99  
30 PRINT/T X  
40 NEXT X  
50 CLOSE  
60 END
```

### (Example 2)

The following program reads data from the data file prepared in Example 1 above. Before executing this program, be sure to rewind the cassette tape.

```
10 ROPEN "DATA "  
20 FOR X=1 TO 99  
30 INPUT/T A  
40 PRINT A  
50 NEXT X  
60 CLOSE  
70 END
```



---

(Example 3)

The following program creates a data file consisting of string data.

```
10 DIM N$ (5)
20 N$ (1) = " BACH "
30 N$ (2) = " MOZART "
40 N$ (3) = " BEETHOVEN "
50 N$ (4) = " CHOPIN "
60 N$ (5) = " BRAHMS "
70 WOPEN " GREAT MUSICIAN "
80 FOR J=1 TO 5
90 PRINT/T N$ (J)
100 NEXT J
110 CLOSE
120 END
```

(Example 4)

The following program reads string data from the file created in Example 3. Before executing this program, be sure to rewind the cassette tape.

```
200 DIM M$ (5)
210 ROPEN " GREAT MUSICIAN "
220 FOR K=1 TO 5
230 INPUT/T M$ (K)
240 PRINT M$ (K)
250 NEXT K
260 CLOSE
270 END
```

It is also possible to create data files which include both numeric and string data. However, since an error will occur if the type of data read does not match the type of variable specified in the INPUT/T statement, it is generally best to limit files to one type of data or the other.

**Note:** It is possible to omit the file name when opening a sequential file with the WOPEN statement. However, this is likely to result in errors if many files are included on the same tape; therefore, it is recommended that you make a habit of assigning file names to sequential data files.

The following program records student grades in English, French, science, and mathematics to a sequential data cassette file.

```

10 INPUT "ENTER NO. OF STUDENTS "; N
20 DIM N$(N), K(N), E(N)
30 DIM R(N), S(N)
40 A$ = "GRADE IS "
50 FOR X=1 TO N
60 PRINT:PRINT "STUDENT NO. "; X
70 INPUT "ENTER STUDENT NAME: "; N$(X)
80 PRINT "ENG "; A$;: INPUT K(X)
90 PRINT "FREN "; A$;: INPUT E(X)
100 PRINT "SCI "; A$;: INPUT R(X)
110 PRINT "MATH "; A$;: INPUT S(X)
120 NEXT X
130 WOPEN "GRADES" ← Opens data file "GRADES" for output on cassette tape.
140 PRINT/T N ← Writes the number of students in the class to the file.
150 FOR X=1 TO N
160 PRINT/T N$(X), K(X), E(X), R(X), S(X) ← Writes grades to the file.
170 NEXT X
180 CLOSE ← Closes the cassette file.
190 END

```

The following program reads the grade data written to the cassette file by the program shown above, then calculates displays the grade average for each student and class averages for each of the various subjects.

```

10 ROPEN "GRADES" ← Opens cassette file "GRADES" for input.
20 INPUT/T N ← Reads the number of people in the class.
30 DIM N$(N), K(N), E(N)
40 DIM R(N), S(N)
50 FOR X=1 TO N
60 INPUT/T N$(X), K(X) ← Reads student names and the grades for English.
70 INPUT/T E(X), R(X), S(X) ← Reads the grades for French, science and mathematics.
80 NEXT X
90 CLOSE ← Closes the file.
100 PRINT TAB(10); "ENG ";
110 PRINT TAB(15); "FREN ";
120 PRINT TAB(20); "SCI ";
130 PRINT TAB(25); "MATH "
140 FOR X=1 TO N
150 PRINT N$(X); TAB(10); K(X);
160 PRINT TAB(15); E(X);
170 PRINT TAB(20); R(X);
180 PRINT TAB(25); S(X);
190 PRINT TAB(30); (K(X) + E(X) + R(X) + S(X)) / 4
200 K(0) = K(0) + K(X) : E(0) = E(0) + E(X)
210 R(0) = R(0) + R(X) : S(0) = S(0) + S(X)
220 NEXT X
230 PRINT TAB(10); K(0) / N; TAB(15); E(0) / N;
240 PRINT TAB(20); R(0) / N; TAB(25); S(0) / N
250 END

```

## 2.4 Built-in Function

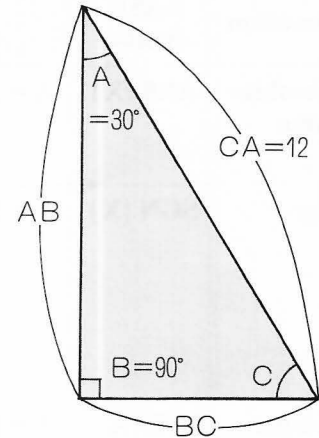
Function	BASIC symbol	Example	Description
Absolute value	ABS (X)	A = ABS (X)	Assigns the absolute value of variable   X   to variable A. Example: A = ABS (2. 9) → A = 2. 9 A = ABS (-5. 5) → A = 5. 5
Sign	SGN (X)	A = SGN (X)	Assigns the numeric sign of variable X to variable A. If the value of X is negative, -1 is assigned to A; if X is 0, 0 is assigned to A; and if X is positive, 1 is assigned to A. Example: 1 is assigned to variable A when A = SGN (0.4) is executed. $A = \begin{cases} 1 & (X > 0) \\ 0 & (X = 0) \\ -1 & (X < 0) \end{cases}$
Integer conversion	INT (X)	A = INT (X)	Assigns the greatest integer value to A which is less than or equal to the value of variable X. Examples: A = INT (3. 87) → A = 3 A = INT (0. 6) → A = 0 A = INT (-3. 87) → A = -4
Trigonometric functions	SIN (X)	A = SIN (X)  A=SIN(30* PA(1)/180)	Assigns the sine of X (where X is in radians) to variable A. If the value of X is in degrees, it must be converted to radians before this function is used to obtain the sine. Since 1 degree equals PI/180 radians, the value in radians is obtained by multiplying the number of degrees by PAI(1)/180. For example, 30° = 30 * PAI(1)/180 radians. The same applies to the COS, TAN, and ATN functions.
	COS (X)	A = COS (X) A=COS (200* PA(1)/180)	Assigns the cosine of X (where X is in radians) to variable A.
	TAN (X)	A = TAN (X) A=TAN(Y* PA(1)/180)	Assigns the tangent of X (where X is in radians) to variable A.
	ATN (X)	A = ATN (X) A=180/PA(1)* ATN(X)	Assigns the arctangent in radians of X ( $\tan^{-1} X$ ) to variable A. The value returned will be in the range from -PI/2 to PI/2.
Square root	SQR (X)	A = SQR (X)	Calculates the square root of X and assigns the result to variable A. X must be a positive number or 0.
Exponentiation	EXP (X)	A = EXP (X)	Calculates the value of $e^x$ and assigns the result to variable A.
Common logarithm	LOG (X)	A = LOG (X)	Calculates the common logarithm of X ( $\log_{10} X$ ) and assigns the result to variable A.
Natural logarithm	LN (X)	A = LN (X)	Calculates the natural logarithm of X ( $\log_e X$ ) and assigns the result to variable A.
Ratio of circumference to diameter	PAI (X)	A = PAI (X)	Assigns the value to variable A which is X times the value of PI.
Radians	RAD (X)	A = RAD (X)	Converts the value of X (where X is in degrees) to radians and assigns the result to variable A.

## Examples of use of the built-in functions

### (Example 1)

Let's try solving the various elements of a triangle with a BASIC program.

Angle A of the triangle shown in the figure at right is  $30^\circ$ , angle B is a right angle, and side CA has a length of 12. The following program finds all angles of the triangle, the length of its sides, and its total area.



```
10 A=30: B=90: CA=12
20 AB=CA*COS (A*PI/180)
30 BC=CA*SIN (A*PI/180)
40 S=AB*BC/2
50 C=180-A-B
60 PRINT "AB=" ; AB, "BC=" ; BC, "CA=" ; CA
70 PRINT "AREAS=" ; S
80 PRINT "A=" ; A, "B=" ; B, "C=" ; C
90 END
```

### (Example 2)

Now let's change line 50 of the program to use ATN, the function for finding the arctangent of a number, to find angle C from sides AB and BC.

```
10 A=30: B=90: CA=12
20 AB=CA*COS (A*PI/180)
30 BC=CA*SIN (A*PI/180)
40 S=AB*BC/2
50 C=ATN (AB/BC) *180/PI
60 PRINT "AB=" ; AB, "BC=" ; BC, "CA=" ; CA
70 PRINT "AREAS=" ; S
80 PRINT "A=" ; A, "B=" ; B, "C=" ; C
90 END
```

## RND function

Format

RND (X)

X .. Numeric expression

Function

The RND function returns a pseudo-random number in the range from 0.00000001 to 0.99999999.

When X is greater than 0, the random number returned is the one which follows that previously generated by the BASIC interpreter in a given pseudo-random number series.

When  $X \leq 0$ , the BASIC Interpreter's pseudo-random number generator is reinitialized to start a new series, and the pseudo-random number returned is the first one in that series. Reinitialization of the pseudo-random number series in this manner can be used to allow simulations based on random numbers to be reproduced.

The RND function is often used in game programs to produce unpredictable numbers, as in games of chance. Let's try using the RND function to investigate the percentage of times each of the six sides of a die comes up by simulating the action of throwing it a given number of times.

Since the sides of each die are numbered from 1 to 6, we must multiply the value returned by the RND function by 6.

$$0 < \text{RND}(1) < 1 \xrightarrow{\times 6} 0 < 6 * \text{RND}(1) < 6$$

Then we must use the INT function to convert the value obtained to an integer.

$$\text{INT}(6 * \text{RND}(1)) \rightarrow 0, 1, 2, 3, 4, 5$$

The result will be an integer between 0 and 5; now 1 is added to obtain the numbers which correspond to the number of dots on each of the 6 sides of a die.

$$\text{INT}(6 * \text{RND}(1)) + 1 \rightarrow 1, 2, 3, 4, 5, 6$$

This sequence is performed a specified number of times for each die thrown. Now let's incorporate the sequence into a program and check the results.

**Example**

```

10 PRINT "ENTER NO. OF
    TIMES DIE THROWN";
20 INPUT N
30 FOR J=1 TO N
40 R=INT(6*RND(1))+1
50 IF R=1 THEN N1=N1+1
60 IF R=2 THEN N2=N2+1
70 IF R=3 THEN N3=N3+1
80 IF R=4 THEN N4=N4+1
90 IF R=5 THEN N5=N5+1
100 IF R=6 THEN N6=N6+1
110 NEXT J
120 P1=N1/N:P2=N2/N:P3=N3/N
130 P4=N4/N:P5=N5/N:P6=N6/N
140 PRINT P1,P2,P3,P4,P5,P6
150 END

```

The RND function generates numbers in the range from 0.0000001 to 0.99999999.

Here's how to obtain random integers in the range from 6 to 8!

$$\text{INT}(3 * \text{RND}(1)) + 6 = 6 \text{ or } 7 \text{ or } 8$$

How about it? If the die is thrown enough times, the percentage of the time each number appears should be about the same. Mathematically speaking, each number should occur an average of once in six throws, or about 16.7% of the time. This mathematical ideal is approached more closely as the number of throws is increased.

**Example**

Now let's try using the RND function in a program which tests your ability to solve for the area of a triangle of random size. Here, the RND function is used to determine the length of each of the three sides of the triangle, then you compute the area of the triangle yourself and submit your answer to the computer for checking.

```

10 DIM A (3) , L$ (4)
20 FOR J=1 TO 4
30 READ L$ (J) : NEXT J
40 FOR J=1 TO 3
50 A (J) = INT (20*RND (1)) + 1
60 NEXT J
70 IF A (1) >= A (2) + A (3) GOTO 40
80 IF A (2) >= A (1) + A (3) GOTO 40
90 IF A (3) >= A (1) + A (2) GOTO 40
100 W = (A (1) + A (2) + A (3)) / 2
110 T = W : FOR J=1 TO 3
120 T = T * (W - A (J)) : NEXT J
130 SS = SQR (T) : S = INT (SS)
140 IF SS - S > 0.5 THEN S = S + 1
150 PRINT "C I I I I"
160 PRINT "      SOLVE FOR THE AREA OF THE
      FOLLOWING TRIANGLE"
170 PRINT "      ROUND YOUR ANSWER TO THE
      NEAREST WHOLE NUMBER"
180 PRINT
190 PRINT TAB (8) ; "A"
200 PRINT TAB (8) ; "  " ; TAB (15) ; L$ (1)
      ; A (1)
210 PRINT TAB (7) ; "  " ; TAB (15) ; L$ (2)
      ; A (2)
220 PRINT TAB (6) ; "  " ; TAB (15) ; L$ (3)
      ; A (3)
230 PRINT TAB (5) ; "  "
240 PRINT TAB (3) ; "B  "
250 PRINT TAB (4) ; " "
260 PRINT " I I I I"
270 PRINT TAB (3) ; L$ (4) ;
280 INPUT Y
290 IF Y = S THEN PRINT "      OK!!" : GOTO
      40
300 IF Y < S THEN PRINT "      TOO SMALL!"
      : GOTO 320
310 PRINT "      TOO LARGE!"
320 PRINT " I I " ;
330 PRINT TAB (24) ; SPC (25) : PRINT " I " ;
340 GOTO 270
350 DATA LENGTH SIDE AB=, LENGTH SIDE BC=
360 DATA LENGTH SIDE CA=, AREAS OF TRIAN-
      GLE ABC IS AREAS OF TRIANGLE ABC IS

```



---

Note

Note that specifying a value for X which is less than or equal to 0 will always result in the same number for a given value of X. The reason for this is that specifying 0 or a negative number reinitializes the pseudo-random number generator to the beginning of the random number series.

---

## 2.5 String Function

### 2.5.1 LEN

Format
--------

LEN (X\$)

X\$ ... String expression

Function
----------

This function returns the number of characters included in the string expression represented by X\$. This value includes spaces which are not displayed on the screen and any control characters in the string, as well as letters, numerals, and symbols.

Example
---------

(Example 1)

```
10 A$= "ABCDEFGH"
20 PRINT LEN (A$)
```

```

RUN
7
```

(Example 2) The following program uses the LEN function to draw squares on the screen.

```
10 ? "C" : ? "ENTER 30R MORE ASTERISKS"
20 INPUT A$
30 FOR I=1 TO LEN (A$) -2
40 PRINT TAB (2) ; "*" ; SPC (LEN (A$) -2) ; "*"
50 NEXT I
60 PRINT TAB (2) ; A$ : GOTO 20
```

(Example 3) The LEN function can also be used to produce a "parade" of characters as shown below.

```
10 S$= "SHARP BASIC"
20 FOR I=1 TO LEN (S$)
30 ? RIGHT$ (S$, I)
40 NEXT I
50 END
RUN
C
IC
SIC
...
SHARP BASIC
```

(Example 4)

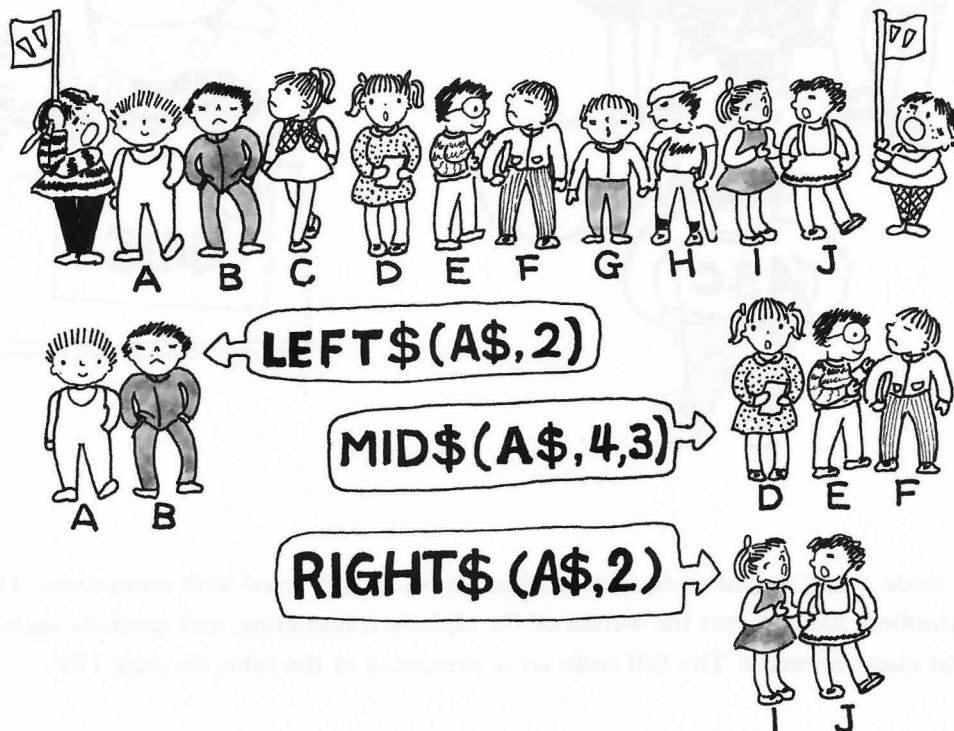
```
PRINT LEN (STR$ (PAI (1)))
9
```

PAI (1), the function which returns the value of the ratio of the circumference of a circle to its diameter, contains the 8-digit constant 3.1415927 (approximately the value of PI). When the length of the character string produced by converting this constant with the STR\$ function is evaluated with the LEN function, a total string length of 9 is returned.

## 2.5.2 LEFT\$, MID\$, and RIGHT\$

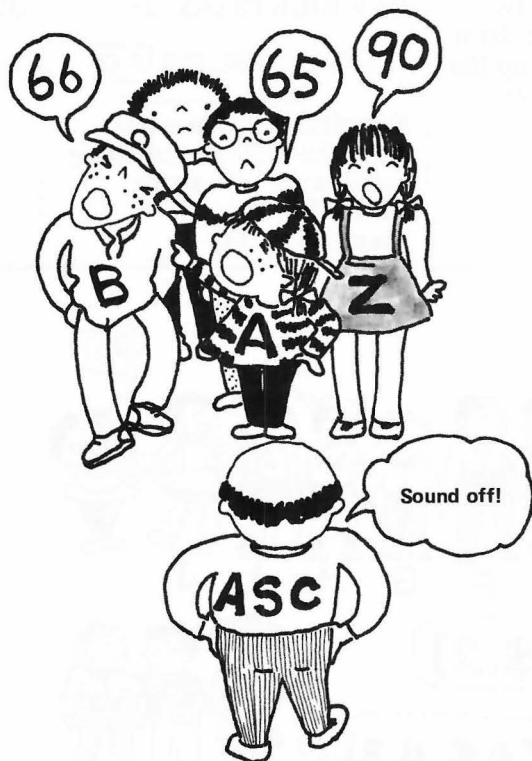
The LEFT\$, MID\$, and RIGHT\$ functions are used to extract character strings from the left end, right end, or middle of a character expression.

Format X\$: String expression m and n: Numeric expressions	Function	Example (when A\$ = "ABCDEFGG")	Remarks
<b>LEFT\$ (X\$, n)</b>	Returns the character string consisting of the n characters making up the left of string expression X\$.	B\$= LEFT\$ (A\$, 2)  B\$ ← AB CDEFG  Substitutes 2 characters from the left end of string variable A\$ into string variable B\$. Thus, B\$ = "AB".	$0 \leq n \leq 255$
<b>MID\$ (X\$, m, n)</b>	Returns the character string consisting of the n characters making up the n characters starting with the mth character in string expression X\$.	B\$=MID\$ (A\$, 3, 3)  B\$ ← ABCDEFG  Substitutes the 3 characters starting at the 3rd character in string variable A\$ into string variable B\$. Thus, B\$ = "CDE".	$1 \leq m \leq 255$ $0 \leq n \leq 255$
<b>RIGHT\$ (X\$, n)</b>	Returns the character string consisting of the n characters making up the right end of string expression X\$.	B\$ = RIGHT\$ (A\$, 2)  B\$ ← ABCDEFG  Substitutes 2 characters from the right end of string variable A\$ into string variable B\$. Thus, B\$ = "GG".	$0 \leq n \leq 255$



## 2.5.3 ASC and CHR\$

Format	Function	Example
<b>ASC (x\$)</b> x\$: String expression	Returns the ASCII code for the first character in string expression x\$.	X=ASC (" A ") Substitutes 65 (the ASCII code for the letter A) into variable X.  Y=ASC (" S HARP ")  Substitutes 83 (the ASCII code for S, the first letter in the string "SHARP") into variable X.
<b>CHR\$ (x)</b> x: Numeric expression	Returns the letter whose ASCII code corresponds to the value of numeric expression X. (No character is returned if the value specified for x is less than 33; therefore, PRINT " " or PRINT SPC (1) should be used to obtain spaces, rather than CHR\$ (32)).	A\$=CHR\$ (65) Assigns A, the letter corresponding to ASCII code 65, to string variable A\$. This function can be used to display characters which cannot be entered from the keyboard as follows. PRINT CHR\$ (107) J This displays the graphic character ☒.



**Note:** ASCII code is a standard code system which is frequently used with computers. This code uses 8 bit numbers to represent the letters of the alphabet, numerals, and symbols such as the dollar sign and question mark. The full code set is presented in the table on page 154.

## 2.5.4 VAL and STR\$

Format	Function	Example
<b>STR\$ (x)</b> x: Numeric expression	Returns a string of ASCII characters representing the value of numeric expression X.	<p>A\$=STR\$ (-12) Substitutes the character string " -12 " into string variable A\$.</p> <p>B\$=STR\$ (70 * 33) Substitutes the character string " 2310 " into string variable B\$.</p> <p>C\$=STR\$ (1200000 * 5000) Substitutes the character string " 6E + 09 " into string variable C\$.</p> <p>Note: Positive numeric values are displayed with a leading space to indicate that the plus sign (+) has been omitted. However, this space is not included in the character string returned by the STR\$ function.</p>
<b>VAL (x\$)</b> x\$: String expression	Converts an ASCII character representation of a numeric value into a numeric value. This is the complement of the STR\$ function.	<p>A=VAL ("123") Converts the character string " 123 " into the number 123 and assigns it to numeric variable A.</p>

The following sample program illustrates use of some of the functions discussed above to display numeric values in tabular format (with the decimal points aligned).

```

1. 23456
12. 3456
10
1
1234

```

If the values read from DATA statements were displayed using only the PRINT statement, the result would appear as shown below.

```

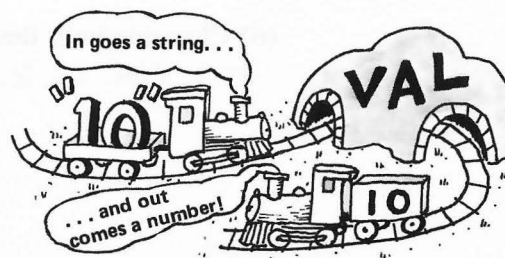
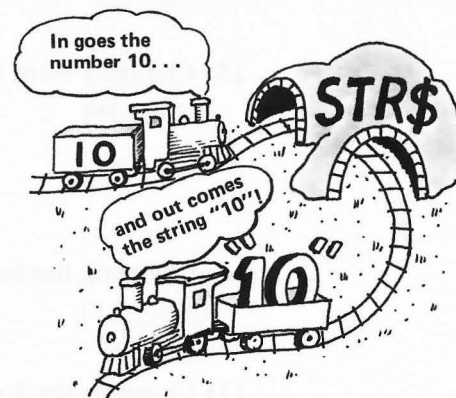
10 FOR X=1 TO 5
20 READ A
30 L=5-LEN (STR$ (INT(A)))
40 PRINT TAB (L) ; A
50 NEXT:END
60 DATA 1. 23456, 12. 3456
70 DATA 123. 456, 1234. 56
80 DATA 12345. 6

```

```

1. 23456
12. 3456
123. 456
1234. 56
12345. 6

```



## 2.6 Color display statement

One of the greatest features of the MZ-700 is that it allows characters and graphics to be displayed using any of up to 8 colors.

### 2.6.1 COLOR ..... (Abbreviated format: COL.)

#### Format

**COLOR** x, y, c <, b >

x . . . . X coordinate (0 to 39)

y . . . . Y coordinate (0 to 24)

c . . . . Character color specification (0 to 7).

b . . . . Background color specification (0 to 7).

#### Function

This statement is used to set the foreground and background colors for the character at a specific position on the screen. Any of up to 8 different colors can be specified for the character foreground (c) or background (b) as shown in the table below.

Color No.	Color
0	Black
1	Blue
2	Red
3	Purple
4	Green
5	Light blue
6	Yellow
7	White

#### Example

- (1) Changing the background color of the entire screen

COLOR , , , 2 . . . . (Changes the background color used for display of characters to red.)

- (2) Changing the foreground color of the entire screen (the color used for display of all characters)

COLOR , , 3 . . . . (Changes the color used for display of all characters to purple.)

- (3) Changing both the background and foreground colors for the entire screen

COLOR , , 1, 0 . . . . (Changes the color used for display of all characters to blue and changes the background used for display of characters to black.)

- (4) Changing the background color at a specific screen location

COLOR 2, 2, , 4 . . . . (Changes the background color at coordinates 2, 2 to green.)

- (5) Changing the foreground color at a specific screen location

COLOR 3, 2, 7 . . . . (Changes the foreground color at coordinates 3, 2 to white.)

- (6) Changing both the foreground and background color at a specific screen location

COLOR 4, 2, 4, 2 . . . . (Changes the foreground color at coordinates 4, 2 to green and changes the background color at that location to red.)







## 2.7.3 Mode Specification Commands

These commands are used to place the printer in the text mode for printout of letters and numerics. This is the mode which is effective when the power is turned on; the initial character size is 40 characters/line.

(1) **MODE TN** ..... (abbreviated format: **M. TN**)

This command returns the printer to the text mode from the graphic mode and sets the character size to 40 characters/line.

(2) **MODE TL** ..... (abbreviated format: **M. TL**)

This command returns the printer to the text mode from the graphic mode and sets the character size to 26 characters/line.

(3) **MODE TS** ..... (abbreviated format: **M. TS**)

This command returns the printer to the **text mode** from the graphic mode and sets the character size to 80 characters/line.

\*\*\* CHARACTER MODE \*\*\*

80 character mode  
ABCDEFGHIJKLMNOPQRSTUVWXYZ

40 character mode  
ABCDEFGHIJKLMNOPQRSTUVWXYZ

26 character mode  
ABCDEFGHIJKLMNOPQRSTUVWXYZ

(4) **MODE GR** ..... (abbreviated format: **M. GR**)

The **MODE GR** command is used to switch the printer from the text mode to the **graphics mode** for printout of charts and graphs. When switching to this mode, it is necessary for the BASIC program being executed to make a note of the character size being used immediately before the mode change is made. The reason for this is in order to return to the text mode when the **BREAK** key is pressed or a **STOP** command is encountered.

**Note:** Executing **MODE** command, every state returns to initial state excluding pen color and print size.

## 2.7.4 Pen color selection commands

<b>PCOLOR</b> n	{	n : 0	black	..... (abbreviated format: <b>PC.</b> )
		n : 1	blue	
		n : 2	green	
		n : 3	red	

This command specifies the color to be used for printout of characters or graphics. n is a number from 0 to 3, with 0 corresponding to black, 1 to blue, 2 to green, and 3 to red.

In text mode, executing **PCOLOR** in text mode every state is on initial state excluding pen color.

To keep current state execute **PRINT/P CHR\$(29)** ..... next color.

**This command can be entered in either the text mode or graphics mode.**





## 2.7.5 Text mode commands

### 2.7.5.1 TEST ..... (abbreviated format: TE.)

Format
Format

#### TEST

This command causes the printer to print squares in each of the four different colors to check the color specification, quantity of pen ink, and so forth. (Only usable in the text mode.)





 ..... Value of n in PCOLOR    n  
 (Black) (Blue) (Green) (Red)

### 2.7.5.2 SKIP

Format
Function

#### SKIP n

n... A number in the range from -20 to 20

This command is used to feed the paper. Paper is fed n lines in the forward direction when the value for n is positive; if the value specified for n is negative, the paper is fed n lines in the reverse direction. Note that PRINTER MODE ERROR will occur if this command is executed while the printer is in the graphics mode.

### 2.7.5.3 PAGE

Format
Function

#### PAGE n

n... An integer in the range  $1 \leq n \leq 72$

This command specifies the number of lines per page. (Executable only in the text mode.)

### 2.7.5.4 LIST/P ..... (abbreviated format: L./P)

Format
Function

#### LIST/P or LIST/P <LS-Le>

Ls ..... Starting line number

Le ..... Ending line number

This command lists all or part of the program lines in memory on the printer. See the explanation of the LIST command on page 32 for an explanation of procedures for specifying the range of lines to be printed. Note that, when graphic characters are included in the program list, most of them will be printed in a different color as hexadecimal ASCII codes. See page 154 for the printer ASCII codes.

This command can only be executed in the text mode.

### 2.7.5.5 PRINT/P ..... (abbreviated format: ?/P)

Format
Function

#### PRINT/P <I<sub>1</sub>, d<sub>1</sub>, I<sub>2</sub>, d<sub>2</sub> ..... I<sub>n</sub>, d<sub>n</sub>>

I<sub>n</sub> ..... Output list (numeric or string expressions)

d<sub>n</sub> ..... Delimiter

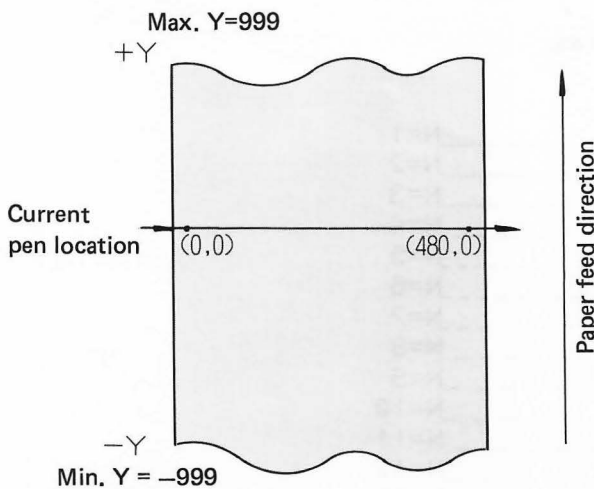
This command outputs the data in the output list to the printer. For details on using this command, see the description of the PRINT command on page 37. See pages 85 for printout of graphic characters.

### 2.7.5.6. PRINT/P USING ..... (abbreviated format: ?/P USI.)

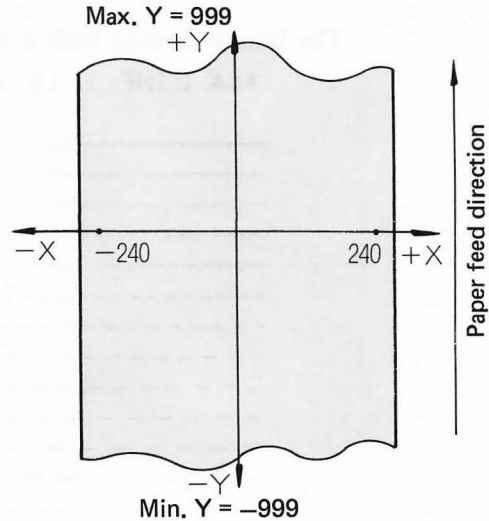
Except that output is directed to the printer, this is the same as the PRINT USING statement described on page 38.

## 2.7.6 Graphic mode statements

The graphic mode statements become effective after the **MODE GR** statement has been executed. When this statement is executed, the current pen location is set to the origin ( $X = 0, Y = 0$ ). However, the origin can be set to any location. Be careful not to specify a location which is out of the print area, as this may damage the pen or cause other problems.



X-Y coordinates after **MODE GR** has been executed. The allowable range of X is 0 to 480 and the allowable range of Y is -999 to 999.



X-Y coordinates after the origin has been moved to the center of paper.  
(**MOVE 240, -240: HSET**)

Note: See page 88 for the **HSET** statement.

### 2.7.6.1 LINE

#### Format

**LINE**  $x_1, y_1 <, x_2, y_2, \dots, x_i, y_i>$  or

**LINE** %n,  $x_1, y_1 <, x_2, y_2, \dots, x_i, y_i>$

n ..... Integer from 1 to 16

$x_i$  ..... Number indicating the X coordinate ( $x_i = -480$  to  $480$ ; the limit varies depending on the current pen location.)

$y_i$  ..... Number indicating the Y coordinate ( $y_i = -999$  to  $999$ )

#### Function

This statement draws a line from the current pen location to location ( $x_1, y_1$ ), then draws a line from ( $x_1, y_1$ ) to ( $x_2, y_2$ ), and so on. n specifies the type of line drawn as shown below.

n = 1: solid line

n = 2 to 16: dotted line

If % is omitted, the previous value of n is assumed. The initial value of n is 1 (solid line).

#### Example

(Example 1) The following program draws a square with a side length of 240 units.

```
1Ø MODE GR .....Switches to the graphic mode.
2Ø LINE 24Ø, Ø .....Draws a line from the origin to the center
                        of paper.
```

```
3Ø LINE 24Ø, -24Ø
```

```
4Ø LINE Ø, -24Ø
```

```
5Ø LINE Ø, Ø .....Draws a line to the origin.
```

```
6Ø MODE TN .....Returns to the text mode.
```

(Example 2) The following program draws the same square as the example above.

```
1Ø MODE GR
```

```
2Ø LINE 24Ø, Ø, 24Ø, -24Ø, Ø, -24Ø, Ø, Ø
```

```
3Ø MODE TN
```

(Example 3) The following program draws a rectangle with a side length of 240 units.

```
10 MODE GR
20 SQ=INT(120*SQR(3))
30 LINE %2, 240, 0, 120, -SQ, 0, 0
40 MODE TN
```

The lines indicated with n are as follows.

\*\*\* LINE 1-16 \*\*\*

-----	N=1
-----	N=2
-----	N=3
-----	N=4
-----	N=5
-----	N=6
-----	N=7
-----	N=8
-----	N=9
-----	N=10
-----	N=11
-----	N=12
-----	N=13
-----	N=14
-----	N=15
-----	N=16

## 2.7.6.2 RLINE ..... (abbreviated format: RL.)

**Format** **RLINE**  $x_1, y_1 < x_2, y_2, \dots x_i, y_i \dots$   
**RLINE** %n,  $x_1, y_1, < x_2, y_2, \dots, x_i, y_i \dots$   
n ..... Integer from 1 to 16

$x_i$  ..... Number indicating the X coordinate (-480 to 480)

$y_i$  ..... Number indicating the Y coordinate (-999 to 999)

**Function** This statement draws a line from the current pen location to the location indicated by **relative coordinates**  $x_1, y_1$ , then draws a line from that point to the location indicated by relative coordinates  $x_2, y_2$ , and so on. n is the same as for the LINE statement.

**Example** This program draws the same rectangle as example 3 above.

```
10 MODE GR
20 SQ=INT(120*SQR(3))
30 RLINE %1, 240, 0, -120, -SQ, -120, SQ
40 MODE TN
```

Initial pen location

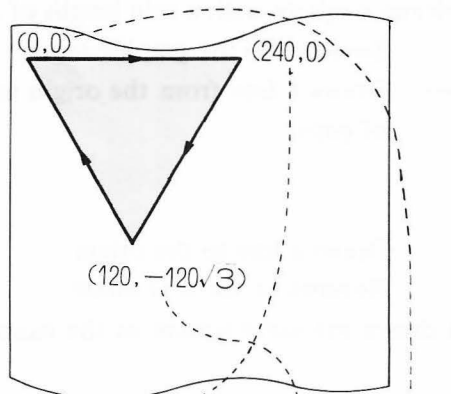


Figure drawn by LINE

Initial pen location

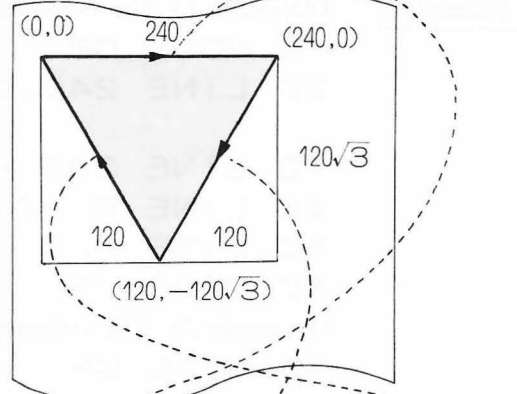


Figure drawn by RLINE



### 2.7.6.3 MOVE

Format
--------

**MOVE** x, y

x . . . . . Integer indicating the X coordinate (–480 to 480)

y . . . . . Integer indicating the Y coordinate (–999 to 999)

Function
----------

This statement **lifts the pen** and moves it to the specified location (x, y).

Example
---------

The following program draws a cross with a side length of 480 units.

```
1Ø MODE GR
```

```
2Ø LINE 48Ø, Ø
```

```
3Ø MOVE 24Ø, 24Ø.....Lifts the pen at (480, 0) and moves it to  
240, 240).
```

```
4Ø LINE 24Ø, –24Ø
```

```
5Ø MODE TN
```

Be sure to advance the paper before executing this program.

### 2.7.6.4 RMOVE . . . . . (abbreviated formed: **RM.**)

Format
--------

**RMOVE** x, y

x . . . . . Integer indicating relative X coordinate (–480 to 480)

y . . . . . Integer indicating relative Y coordinate (–999 to 999)

Function
----------

This statement lifts the pen and moves it to the location indicated by **relative coordinates** ( $\Delta x$ ,  $\Delta y$ )

Example
---------

The following program draws the same cross as the example for the MOVE statement.

```
1Ø MODE GR
```

```
2Ø LINE 48Ø, Ø
```

```
3Ø RMOVE –24Ø, 24Ø.....Lifts the pen at (480, 0), then moves it  
–240 units in the X direction and 240  
units in the Y direction.
```

```
4Ø LINE 24Ø, –24Ø
```

```
5Ø MODE TN
```

Be sure to advance the paper before executing this program.

### 2.7.6.5 PHOME . . . . . (abbreviated format: **PH.**)

Format
--------

**PHOME**

Function
----------

This statement returns the pen to the origin.

Example
---------

The following example draws the same cross in red as the example for the MOVE statement.

```
1Ø MODE GR
```

```
2Ø LINE 48Ø, Ø :MOVE 24Ø, 24Ø
```

```
3Ø LINE 24Ø, –24Ø
```

```
4Ø PHOME .....Returns the pen to the origin.
```

```
5Ø PCOLOR 3
```

```
6Ø LINE Ø, 24Ø, 48Ø, 24Ø, 48Ø, –24Ø, Ø, –24Ø, Ø,  
Ø
```

```
7Ø MODE TN
```

## 2.7.6.6 HSET ..... (abbreviated format: H.)

**Format**

**HSET**

**Function**

This statement sets the current pen location as the new origin. With this feature, the origin can be set to the location which is most appropriate for drawing figures. A MOVE statement is frequently executed before executing this command.

**Example**

```
10 MODE GR
20 MOVE 240, -240
30 HSET .....Sets the new origin.
40 FOR I=1 TO 360 STEP 30
50 LINE 240*COS (PAI(1)*I/180),240*SIN (PAI(1)*I/180)
60 PHOME
70 NEXT
80 MODE TN
```

## 2.7.6.7 GPRINT ..... (abbreviated format: GP.)

**Format**

**GPRINT [n, @] , x\$**

**GPRINT x\$**

n ..... Integer indicating the character size (0 ~ 63)

@ ..... Integer indicating the direction in which lines of characters are printed.  
(@ = 0 ~ 3)

x\$ ..... Character

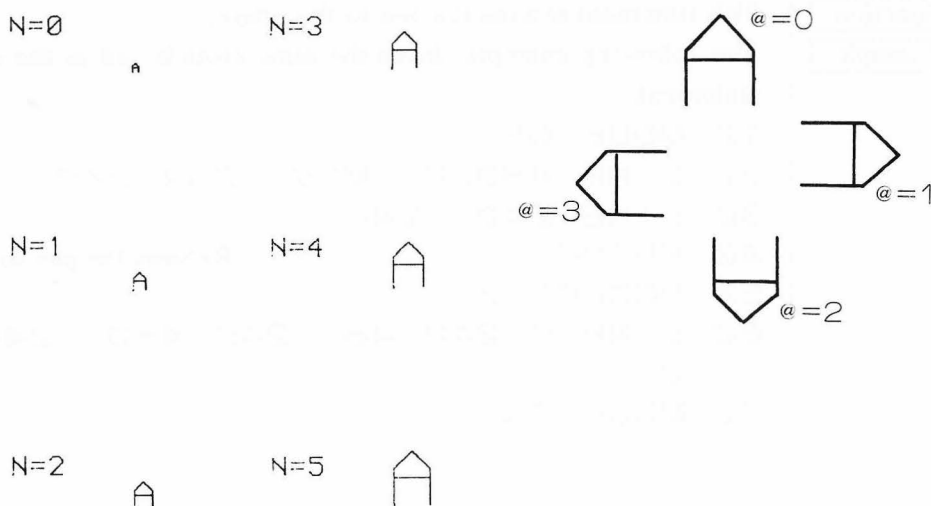
**Function**

This statement prints the specified character using the specified size and direction. 80 characters can be printed on each line when n = 0; 40 characters can be printed on each line when n = 1; and 26 characters can be printed on each line when n = 2. When n and @ are omitted, the previous settings are assumed. Their initial values are n = 1 and @ = 0.

**Example**

```
10 MODE GR
20 GPRINT "A" .....Prints "A" in the graphic mode.
30 GPRINT (2, 2) , "A" .....Prints an upside down "A" in the 26
                           characters/line mode.
```

The following figures show various examples of printout.



## 2.7.6.8 AXIS ..... (abbreviated format: AX.)

### Format

**AXIS** x, p, r

x ..... Integer specifying the axis drawn (0 or 1)

p ..... Integer specifying the scale pitch (–999 to 999)

r ..... Integer specifying the number of repetitions (1 to 255)

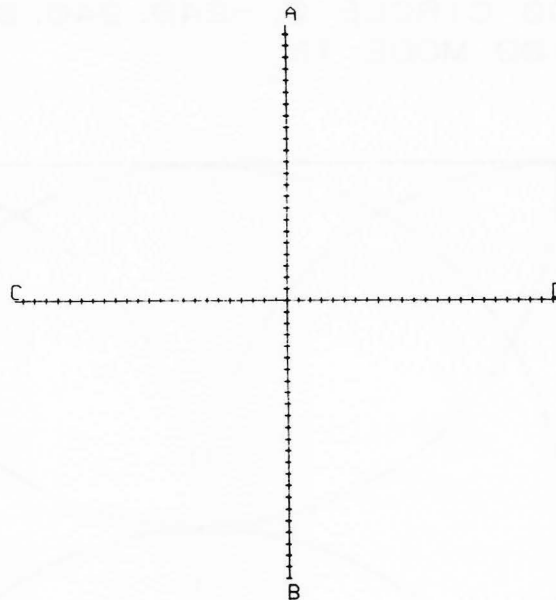
### Function

This statement draws the X-axis when x = 0 and the Y-axis when x = 1. The number of scale marks specified in r are drawn with a pitch of p.

### Example

The following example draws the X and Y axes with scale marks from –240 to 240 at 10 unit intervals.

```
1Ø MODE GR ..... Switches the printer to the graphic mode.
2Ø MOVE 24Ø, Ø ..... Lifts the pen and moves it to position A
                        (240, 0).
3Ø AXIS Ø, –1Ø, 48 ..... Draws the Y-axis from position A to posi-
                        tion B with scale marks included at 10-
                        unit interval.
4Ø MOVE Ø, –24Ø ..... Lifts the pen and moves it to position C
                        (0, –240).
5Ø AXIS 1, 1Ø, 48 ..... Draws the X-axis from position C to posi-
                        tion D with scale marks included at 10-
                        unit intervals.
6Ø MODE TN
```



The coordinates can be used in the same manner as ordinary Cartesian coordinates after setting the point of intersection of the X and Y axes as the new origin. (X = –240 to 240, Y = –240 to 240)

## 2.7.6.9 CIRCLE ..... (abbreviated format: CI.)

### Format

**CIRCLE** x, y, r, s, e, d

x, y ..... Location of the center (-999 to 999)

r ..... Radius (0 to 999)

s ..... Starting angle (in degree)

e ..... Ending angle (in degree)

d ..... Step angle (in degree)

### Function

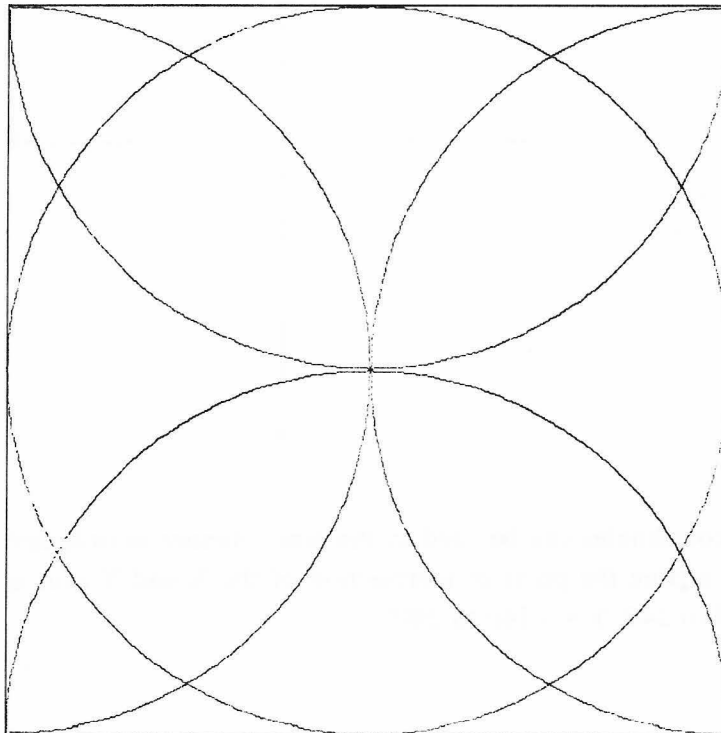
This statement draws a circle or arc with a radius of r and a step of d at location (x, y), starting at angle S and ending at angle e. A complete circle is drawn when s = 0, e = 360 and d = 0.2.

Actually this statement draws a polygon; therefore, d must be as small as possible in order to draw a smooth figure.

s must be smaller than e. When d = 0, lines connecting the center and the starting point and the center and the ending point are drawn.

### Example

```
1Ø MODE GR
2Ø LINE 48Ø, Ø, 48Ø, -48Ø, Ø, -48Ø, Ø, Ø
3Ø MOVE 24Ø, -24Ø
4Ø HSET
5Ø CIRCLE Ø, Ø, 24Ø, Ø, 36Ø, Ø. 2
6Ø CIRCLE 24Ø, Ø, 24Ø, 9Ø, 27Ø, Ø. 2
7Ø CIRCLE Ø, 24Ø, 24Ø, 18Ø, 36Ø, Ø. 2
8Ø CIRCLE -24Ø, Ø, 24Ø, 27Ø, 45Ø, Ø. 2
9Ø CIRCLE Ø, -24Ø, 24Ø, Ø, 18Ø, Ø. 2
1ØØ MODE TN
```



## 2.8 Machine Language Program Control Statements

Several machine language program control statements are supported by the MZ-700 BASIC interpreter. With these statements, machine language programs can be linked with a BASIC program.

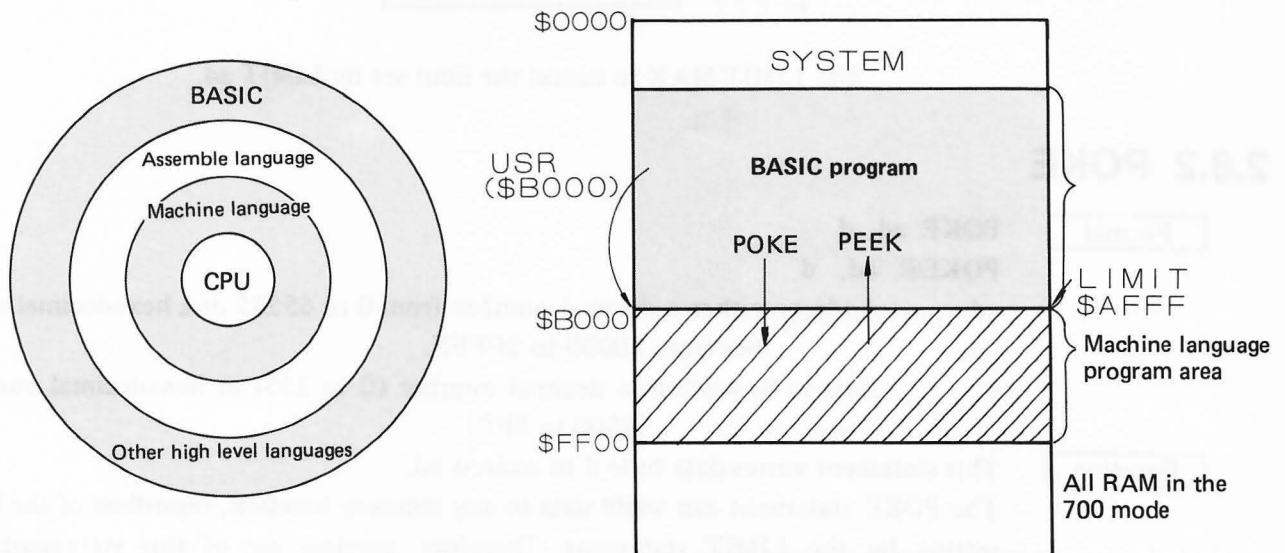
Computer programming languages form a hierarchical structure as shown below. High level languages such as BASIC automatically performs work required when lower level languages such as assembly language are used. Although high level languages are convenient and easy to use, they cannot control the CPU directly.

The lowest level language (machine language) directly controls the CPU and provides high processing speed, but considerable skill is required for coding long programs.

Machine language program control statements enable sophisticated programming techniques which make it possible to utilize the advantages of both BASIC and machine language.

Machine language programs can be generated and loaded into the machine language program area (reserved with the BASIC LIMIT statement) using the monitor or assembler and loader. Such machine language programs can be called by BASIC programs with the USR ( ) function. Machine language programs can also be loaded into memory using a BASIC program which uses the POKE statement to write each step in machine code. The resultant machine language program can then be called by BASIC programs with the USR ( ) function.

The memory map at bottom right outlines the concept of data access with POKE and PEEK, and of calling machine language programs with USR ( ).



## 2.8.1 LIMIT ..... (Abbreviated format: LIM.)

### Format

**LIMIT** ad

ad ..... Address; either a decimal number from 0 to 65279 or a 4-digit hexadecimal number from \$0000 to \$FEFF.

### Function

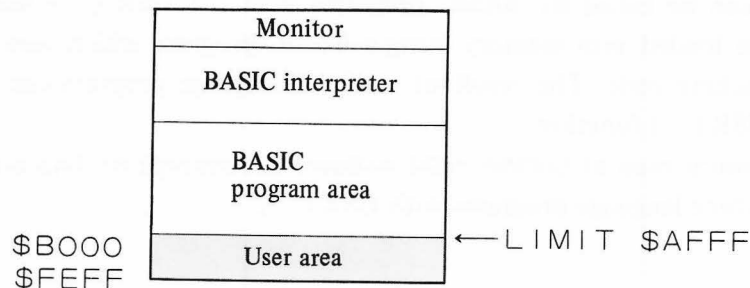
This statement limits the memory area which can be used by the BASIC interpreter. ad indicates the upper limit of the BASIC area, and the area from the following address (ad + 1) to \$FEFF (65279) can be used for machine language programs or special data.

### Example

**LIMIT** \$AFFF

Limits the BASIC program area to \$AFFF.

**Note** The area from \$FF00 to \$FFFF is used by the monitor as a work area, so it cannot be used as the user area. The **LIMIT** statement must be used at the beginning of a BASIC program.



Use **LIMIT MAX** to cancel the limit set by **LIMIT** ad.

## 2.8.2 POKE

### Format

**POKE** ad, d

**POKE@** ad, d

ad ..... Address: either a decimal number from 0 to 65535 or a hexadecimal number from \$0000 to \$FFFF.

d ..... Data to be written: a decimal number (0 to 255) or hexadecimal number (\$00 to \$FF)

### Function

This statement writes data byte d to address ad.

The **POKE** statement can write data to any memory location, regardless of the limit setting by the **LIMIT** statement. Therefore, careless use of this statement can destroy the monitor or BASIC interpreter.

The **POKE@** format is used to write data to an address in the user RAM area following 53248 (\$D000). (See page 125.)

### Example

**POKE** \$D000, \$5F

**POKE** 53248, 95

The two statements above perform the same function.

**Note** A **POKE** statement specifying an address after \$D000 writes data into the video RAM area.



---

### 2.8.3 PEEK

Format
--------

**PEEK** (ad)

**PEEK@** (ad)

ad . . . . . Address in decimal or hexadecimal notation (0 to 65535 or \$0000 to \$FFFF)

Function
----------

This function returns the contents of the specified address as a decimal number from 0 to 255. Use the PEEK@ format to PEEK a user RAM area following 53248 (\$D000).

Example
---------

The following program displays data stored in the area from 40960 (\$A000) to 40975 (\$A00F).

```
10 FOR AD= 40960 TO 40975
20 ? PEEK (AD)
30 NEXT AD
```

### 2.8.4 USR ..... (Abbreviated format: U.)

Format
--------

**USR** (ad)

**USR** (ad, x\$)

ad . . . . . Address (decimal or 4-digit hexadecimal)

x\$ . . . . . String data

Function
----------

This is a special function which transfers control to a machine language program which starts at the specified address. As with CALL ad, so control is returned to the statement following the USR function if the machine language program includes a return instruction (RET or RET\_cc).

When x\$ is specified, the starting address of the memory area containing x\$ is loaded into the DE register, then the length of x\$ is loaded into the B register before the machine language program is called. This makes it possible for a BASIC program to pass string data to a machine language program.

## 2.8.5 Preparing machine language programs

A machine language program which fills the entire display screen with the characters supported by the MZ-700 is presented in this section as an example.

The following BASIC program loads such a machine program into memory and calls it.

```
10 LIMIT $BFFF .....Limits the BASIC area to $BFFF.
20 GOSUB 50
30 USR($C000) .....Calls the machine language program.
40 END
50 FOR I =49152 TO 49181
60 READ M
70 POKE I,M
80 NEXT I
90 RETURN

100 DATA 197:REM      PUSH BC .....Beginning of data for the machine language program.
110 DATA 213:REM      PUSH DE
120 DATA 229:REM      PUSH HL
130 DATA 22,0:REM      LD D,0
140 DATA 33,0,208:REM  LD HL,D000H
150 DATA 1,232,3:REM   LD BC,1000
160 DATA 243:REM      DI
170 DATA 211,227:REM   OUT (E3H),A .....Switches the memory block to video RAM. (See page
180 DATA 114:REM      STO:LD (HL),D .....155).
190 DATA 35:REM      INC HL .....Sets a display code to video RAM.
200 DATA 20:REM      INC D
210 DATA 11:REM      DEC BC
220 DATA 120:REM      LD A,B
230 DATA 177:REM      OR C
240 DATA 194,14,192:REM JP NZ,STO
250 DATA 211,225:REM   OUT (E1H),A .....Switches the memory block to RAM. (See page 127.)
260 DATA 251:REM      EI
270 DATA 225:REM      POP HL
280 DATA 209:REM      POP DE
290 DATA 193:REM      POP BC
300 DATA 201:REM      RET .....Returns to the BASIC program.
```

If the machine language program has been generated with the monitor and saved on cassette tape under the file name DISPLAYCODE, use the following program to call the machine language program.

```
110 LIMIT $BFFF
110 LOAD "DISPLAYCODE"
120 USR($C000)
```

---

## 2.9 I/O Statements

All external devices (including floppy disk drives) are connected to the MZ-700 through an optional interface board. The optional universal interface board makes it possible for the user to connect external devices such as an X-Y plotter, paper tape punch, and music synthesizer to the MZ-700.

A port address selection switch is provided on the universal interface card to allow any port address from 0 to 239 (00H to EFH) can be assigned to any devices. Addresses 240 to 255 are reserved for optional peripheral devices supplied by Sharp.

The INP and OUT statements allow the user to transfer data from/to external devices through the optional universal I/O card. The format of these statements is as follows.

- INP #P, D . . . . .** Reads 8-bit data from port P, converts it into a decimal number and assigns it to variable D.
- OUT #P, D . . . . .** Converts a decimal number in variable D to binary format and outputs it to port D.

These statements greatly extend the range of applications of the MZ-700 series computers.

---

## 2.10 Other Statements

### 2.10.1 ON ERROR GOTO ..... (Abbreviated format: ON ERR. G.)

Format	ON ERROR GOTO Lr
--------	------------------

Lr . . . . Destination line number (entry point of an error processing routine)

Function	This statements causes execution to branch to line number Lr if an error occurs. The IF ERN and IF ERL statement can be used in a trap routine starting at that line to control subsequent processing according to the type of error and the line number in which it occurred. Including a RESUME statement at the end of the error processing routine makes it possible to return execution to the line at which the error occurred. Executing an ON ERROR GOTO statement cancels the error trap line number defined by the previous ON ERROR GOTO statement. The error trap line number definition is also cancelled by executing a CLR statement.
----------	--

### 2.10.2 IF ERN

Format	IF relational expression using ERN THEN Lr IF relational expression using ERN THEN statement IF relational expression using ERN GOTO Lr
--------	---

Lr . . . . Destination line number

Function	This statement branches execution to the error processing (trap) routine starting at line Lr or executes the statement following THEN when the result of <relational expression using ERN> is true.
----------	---

ERN is a special function which returns a number corresponding to the type of error occurring. See page 159 for the error numbers.

Example	The following shows an error processing routine beginning on line 1000 which causes execution to branch to line 1200 if the error number is 5.
---------	--

1000 ON ERROR GOTO 1000.....Declares the line number of the  
error processing routine.

.....  
1000 IF ERN=5 THEN 1200.....Branches to 1200 if a string  
overflow error has occurred.  
.....

### 2.10.3 IF ERL

**Format** IF relational expression using ERL THEN Lr  
IF relational expression using ERL THEN statement  
IF relational expression using ERL GOTO Lr  
Lr . . . . Destination line number

**Function** This statement branches execution to the routine starting at line Lr or executes the statement following THEN when the result of <relational expression using ERL> is true.

**Example** ERL is a special function which returns the line number at which an error occurred. The following statement causes execution to branch to line 1300 if an error has occurred on line 250.

1010 IF ERL = 250 THEN 1300

The following statement returns control to line 520 in the main routine if the error number is 43 and the error line number is other than 450.

1020 IF (ERN = 43) \* (ERL < > 450) THEN RESUME 520

### 2.10.4 RESUME ..... (Abbreviated format: RESU.)

**Format** RESUME <NEXT>  
RESUME Lr

Lr . . . . Line number or 0

**Function** This statement returns control to the main routine from an error processing routine.

**Discussion** The system holds the number of the line on which the error occurred in memory and returns program execution to that line or to another specified line after the error is corrected.

The RESUME statement may be used in any of the following four forms:

RESUME ..... Returns to the error line.

RESUME NEXT ..... Returns to the line following the error line.

RESUME Lr ..... Returns to line Lr.

RESUME Ø ..... Returns to the beginning of the main routine.

If the RESUME is encountered when no error has occurred, error 21 (RESUME ERROR) occurs.

If the RESUME cannot be executed, error 20 (CAN'T RESUME ERROR) occurs.

### 2.10.5 SIZE

**Format** PRINT SIZE

**Function** This is a special function which returns the number of bytes in memory which can be used for storage of BASIC programs.

For example, PRINT SIZE displays the number of free bytes of memory area.

## 2.10.6 PLOT ON ..... (Abbreviated format: PL. ON)

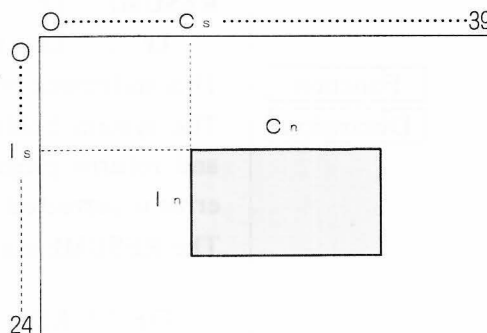
Format	PLOT ON
Function	This statement makes it possible to use the color plotter-printer as a display unit. Thus, the MZ-700 can be used without an external display screen. This statement is effective only when the color plotter-printer is installed and the MODE TN statement has been previously executed.
Example	PLOT ON
Note	A period "." is printed to represent any characters which are not stored in the color plotter-printer's character generator (see page 156). The <code>INST</code> , <code>DEL</code> and " <code>←</code> " keys are disabled by executing this statement. <code>CTRL</code> + <code>G</code> can be used to change the pen.

## 2.10.7 PLOT OFF ..... (Abbreviated format: PL. OFF)

Format	PLOT OFF
Function	This statement cancels PLOT ON made of plotter-printer operation.
Example	PLOT OFF

## 2.10.8 CONSOLE ..... (Abbreviated format: CONS.)

Format	CONSOLE <Is, In< ,Cs, Cn >>
	Is : Starting line of the scroll area In : Number of lines within the scroll area Cs : Starting column of the scroll area Cn : Number of columns in the scroll area
Example	<pre>CONSOLE 0, 25, 0, 40 CONSOLE 5, 15 CONSOLE 0, 25, 5, 30 CONSOLE 0, 10, 0, 10 CONSOLE</pre>
Function	<p>This statement specifies the size of the scroll area; i. e., the area which is cleared by PRINT "■".</p> <p>The first example specifies the entire screen as the scroll area. The second specifies the area between lines 5 and 15 as the scroll area. The third specifies the area between columns 5 and 30 as the scroll area. The fourth specifies the 10 x 10 positions at the upper left corner of the screen as the scroll area.</p> <p>This statement is useful for excluding the left and/or right edges of the image from the display area. When they are hidden behind the edges of the screen.</p> <p>The last example does not specify the scroll area. When the scroll area is not specified, it is possible to scroll the screen up or down.</p> <p>However, this makes it harder to perform screen editing because the values of Cn and In become smaller.</p>





---

## 2.11 Monitor Function

The IOCS section of the BASIC Interpreter includes a monitor program to make it easy to enter machine language programs. This monitor program uses the area from FF00H to FFFFH as a stack area.

This monitor program includes the screen editor similar to that of BASIC which makes it possible to change the contents of any address within the 64K RAM area as described below.

### 2.11.1 Editing format

: address = data data data

: (colon) ... Indicates that the line following can be edited.

address ... Indicates the starting address of the memory area whose contents can be changed.  
(4 hexadecimal digits)

= ... Separates data from the address.

data ... 2-digit hexadecimal number or a semicolon “;” plus the character which is written in the specified address. A blank is used to separate adjacent data items.

### 2.11.2 Printer switching command ..... (P command)

Format \* P

This command switches data output with the D or F command between the printer and display. If the printer is not connected to the computer, the message “ERR?” is displayed and the monitor stands by for input of another command. Check the printer connection or execute the P command again to switch the output device to the display.

### 2.11.3 Dump command ..... (D command)

Format \* D <start address < \_ end address >>

This command dumps the contents of memory from the starting address to the end address. If the end address is omitted, the contents of the 128-byte block starting at the specified address are dumped. If both addresses are omitted, it dumps the contents of the 128-byte block following memory block previously dumped. The format in which data is dumped is as follows.

: HHHH=HH\_HH\_HH HH HH HH HH HH /ABCDE. G.  
↑  
Starting address                      8 bytes (Hexadecimal code)                      8 bytes (Characters)

The contents of any location can be changed by moving the cursor to the corresponding byte, entering the new data, and pressing the CR key.

**Note** Control codes are displayed as a period (.) in the character data field. Pressing the BREAK key stops dump output, and pressing the SHIFT and BREAK keys simultaneously returns the monitor to the command input mode.

---

## 2.11.4 Memory set command ..... (M command)

Format      \* M [starting address]

This command is used to change the contents of memory. If the starting address is omitted, the address currently indicated by the program counter is assumed. Press the SHIFT and BREAK keys together to terminate this command.

When this command is entered, the starting address of the memory block and its contents are displayed in the editing format described previously and the cursor is moved to the data to be changed. Enter the new data and press the CR key; the following address and its contents are then displayed.

## 2.11.5 Fin command ..... (F command)

Format      \* F [starting address] \_ [end address] \_ [data] \_ [data] \_ .....

This command searches for the specified data string in the memory area from the starting address to the end address. When found, the address of the string and its contents are dumped to the screen. This command is terminated by simultaneously pressing the SHIFT and BREAK keys.

## 2.11.6 Subroutine call ..... (G command)

Format      \* G [call address]

This command calls the subroutine starting at the specified address. The stack pointer is located at FFEEH.

## 2.11.7 Transfer command ..... (T command)

Format      \* T [starting address] \_ [end address] \_ [destination address]

This address transfers the contents of memory between the starting address and the end address to the memory area starting at the destination address.

## 2.11.8 Save command ..... (S command)

Format      \* S[starting address] \_ [end address] \_ [execution address] : [file name]

This command saves the contents of the memory between the starting address and the end address to cassette tape under the specified file name.

---

## 2.11.9 Load command ..... (L command)

Format
--------

 \* L < load address > < : file name >

This command loads the specified file into memory, starting at the load address. If the load address is omitted, the execution address contained in the file is assumed as the load address. If the file name is omitted, the first file encountered on the tape is loaded. The message "ERR?" is displayed if a check sum error is detected or the 

BREAK
-------

 key is pressed during execution, then the monitor returns to the command wait state input mode. The command input mode wait state is entered when execution is wait state is entered when execution is completed.

## 2.11.10 Verify command ..... (V command)

Format
--------

 \* V < file name >

This command reads the specified file from cassette tape and compares it with the contents of memory. This makes it possible to confirm that a program has been properly recorded with the SAVE command. If any difference is found between data read from the tape and that contained in memory, the message "Err ? " is displayed.

## 2.11.11 Return command ..... (R command)

Format
--------

 \* R

This command returns control to the system program which called the monitor program and restores the SP (stack pointer) and HL register to the values which they contained when the monitor program was called. Execution resumes with the command following BYE is executed.

This command cannot return control if the monitor has been called by a system program whose stack pointer is between FF00H to FFFFH, or if the stack pointer does not contain a return address. In such cases, use the G command to call the warm start entry point.



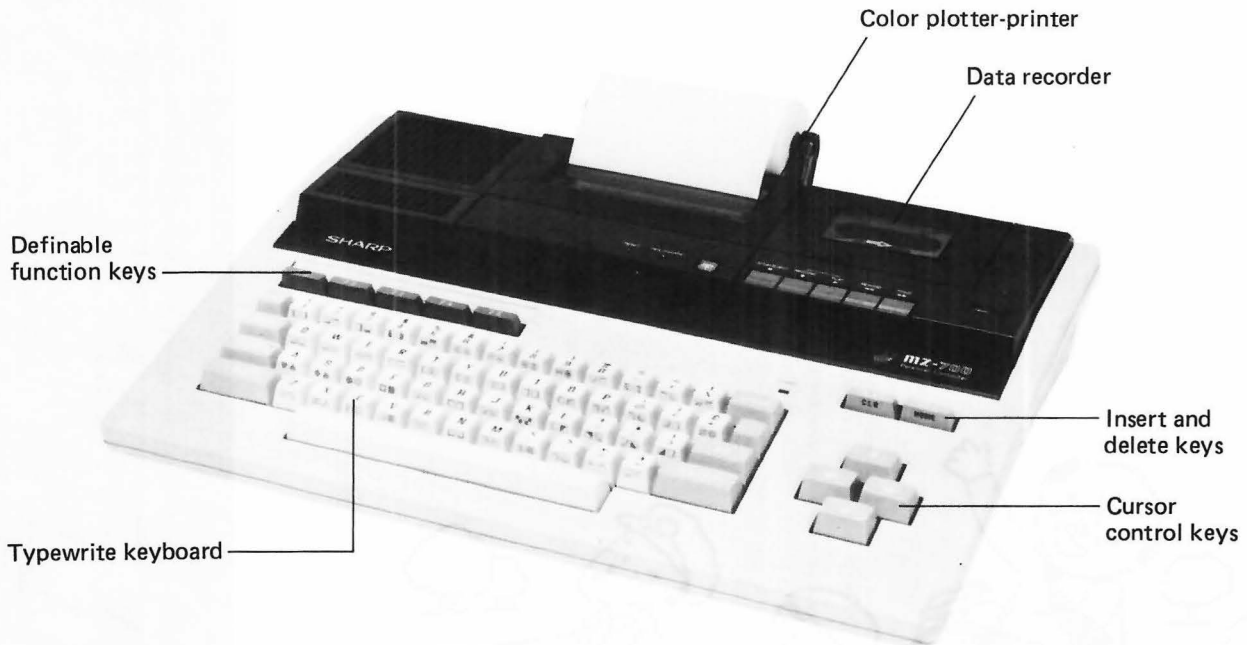
# Operating the MZ-700



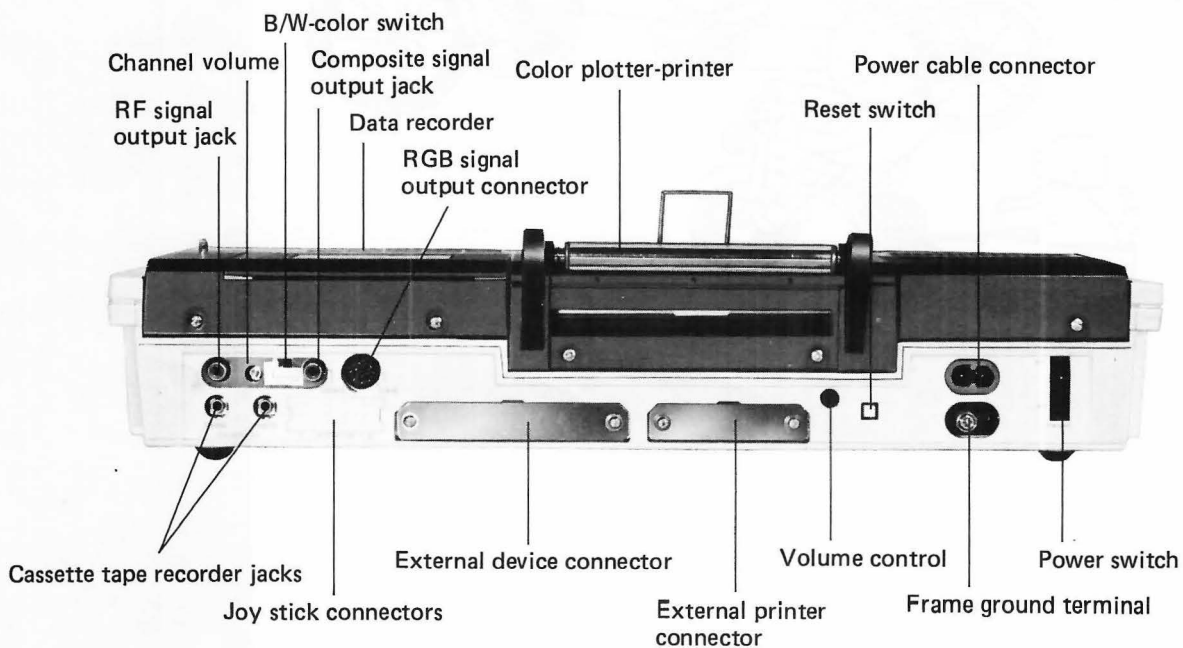
## 3.1 Appearance of the MZ-700 Series Personal Computers

### 3.1.1 MZ-731

#### ■ Front view



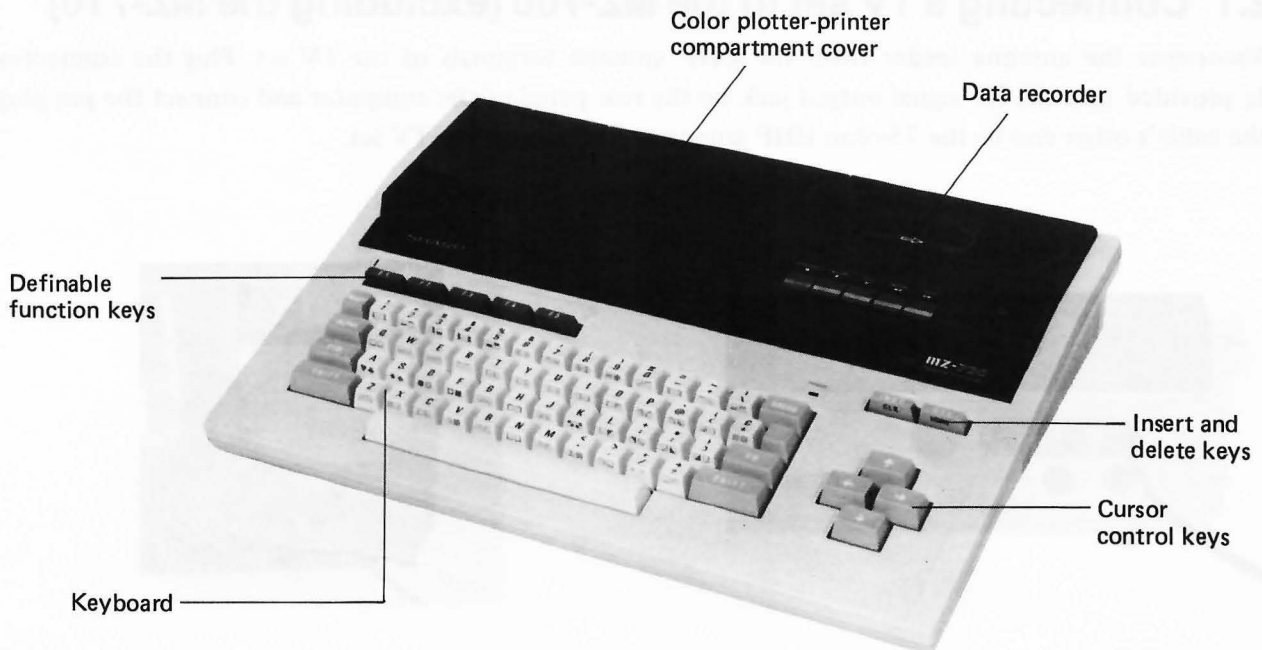
#### ■ Rear view





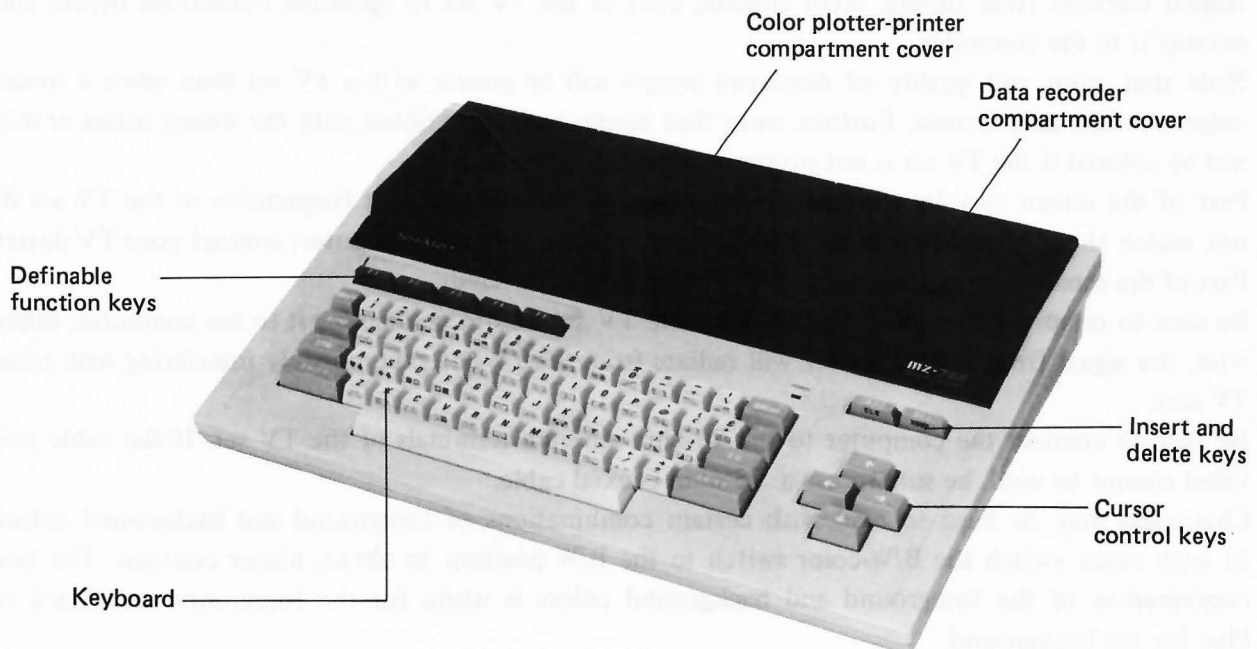
### 3.1.2 MZ-721

#### ■ Front view



### 3.1.3 MZ-711

#### ■ Front view



### 3.1.4 MZ-710

The MZ-710 is the same as the MZ-711 except that it does not include an RF signal output jack and Composite signal output jack on the rear panel.

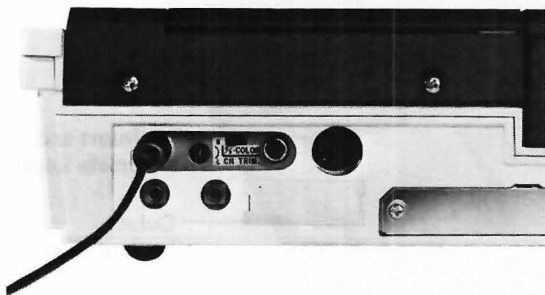
---

## 3.2 Connection to Display Unit

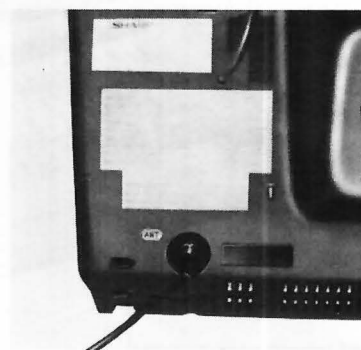
Be sure to turn off both the computer and display unit before connecting them.

### 3.2.1 Connecting a TV set to the MZ-700 (excluding the MZ-710)

Disconnect the antenna feeder from the UHF antenna terminals of the TV set. Plug the connection cable provided into the RF signal output jack on the rear panel of the computer and connect the pin plugs on the cable's other end to the 75-ohm UHF antenna terminals on the TV set.



Back view of MZ-700



Back view of Home TV

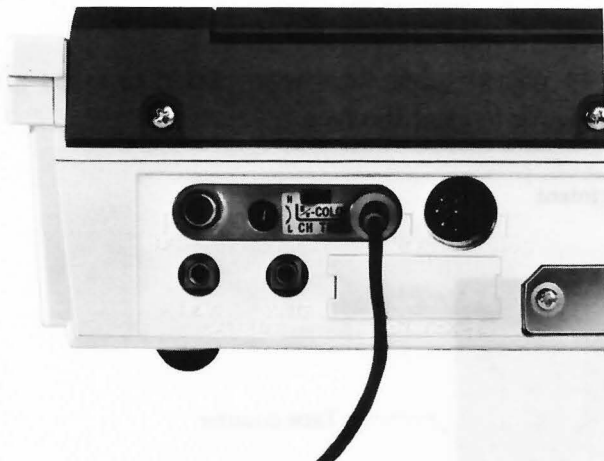
Set the channel selection switch to the  $36 \pm 3$  ch position, depending on which is not used in your area.

Note the following when using an ordinary TV set as a display unit.

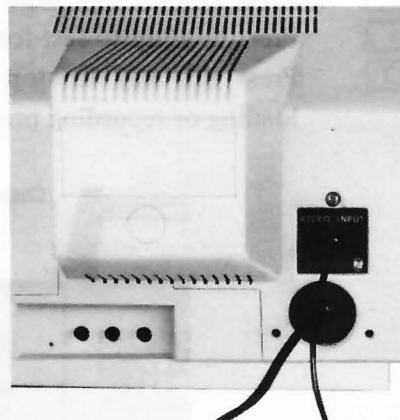
- Adjust controls (fine tuning, color control, etc.) of the TV set to optimum conditions before connecting it to the computer.
- Note that color and quality of displayed images will be poorer with a TV set than when a special color monitor unit is used. Further, note that images may be painted with the wrong colors or may not be colored if the TV set is not properly adjusted.
- Part of the screen may be omitted if vertical and horizontal scanning frequencies of the TV set do not match those of the computer. This is not a problem with the computer; contact your TV dealer.
- Part of the screen may not be visible if the image is not centered.
- Be sure to remove the antenna feeder from the TV set before connecting it to the computer; otherwise, the signal from the computer will radiate from the TV antenna, possibly interfering with other TV sets.
- Be sure to connect the computer to the 75-ohm antenna terminals of the TV set. If the cable provided cannot be used, be sure to use a **75-ohm coaxial cable**.
- Characters may be hard to read with certain combinations of foreground and background colors. In such cases, switch the B/W-color switch to the B/W position to obtain higher contrast. The best combination of the foreground and background colors is white for the foreground and black or blue for the background.
- No audio signal is included in the RF signal fed to the TV set, so sound cannot be output from the speaker of the TV set.

### 3.2.2 Connecting the MZ-1D04 12-inch green display to the computer (excluding the MZ-710)

Use the cable included with the MZ-1D04 green display to connect it to the computer. Plug the cable into the composite signal jack on the computer's rear panel, then set the B/W-COLOR switch to the B/W position.



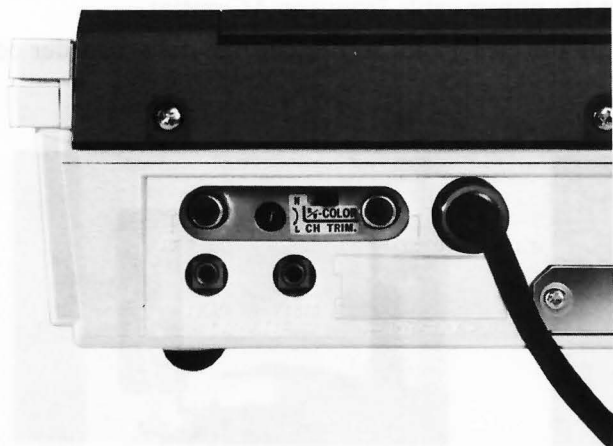
Rear panel of the MZ-700 series computer



Rear panel of the MZ-1D04

### 3.2.3 Connecting the MZ-1D05 14-inch color display to the computer

Use the cable included with the MZ-1D05 color display to connect it to the computer. Plug the cable's DIN connector into the RGB signal output connector on the MZ-700.

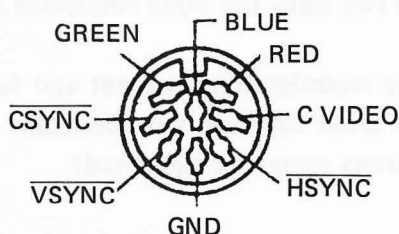


Rear panel of the MZ-700 series computer



Rear panel of the MZ-1D05

Pin assignments of the RGB signal output connector of the MZ-700 are as shown below.



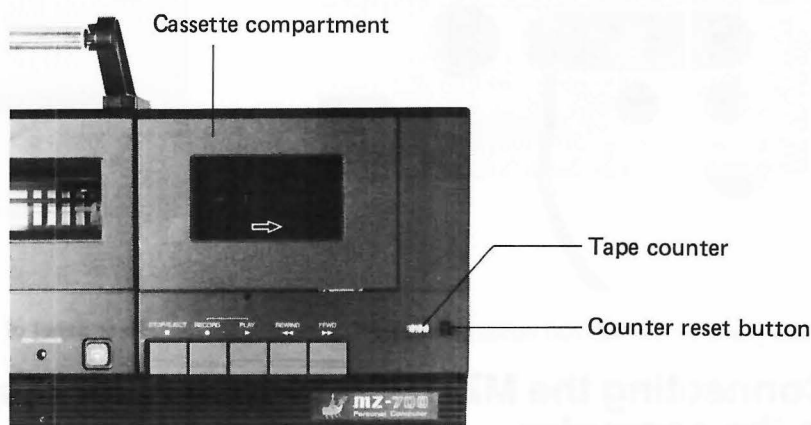
RGB signal output DIN connector  
(viewed from the rear side)

## 3.3 Data Recorder

### ■ Data recorder built into the MZ-731 and MZ-721

The built-in data recorder can be operated in the same manner as an ordinary cassette tape recorder.

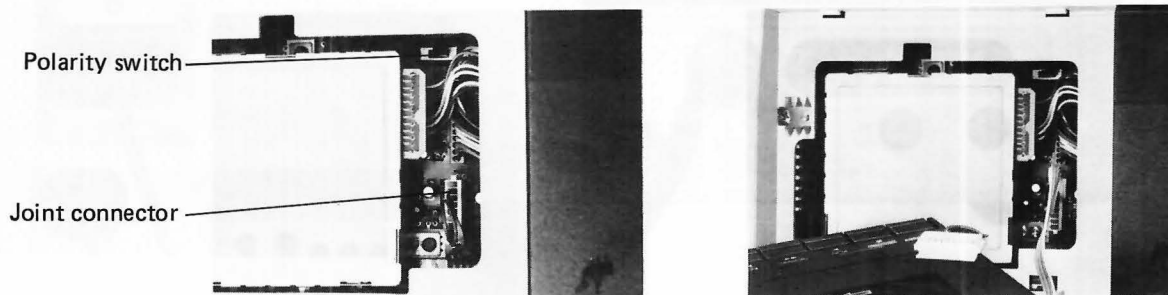
RECORD	Press this key to record programs and data.
PLAY	Press this key to load programs and data.
REWIND	Press this key to rewind the tape.
FFWD	Press this key to fast-forward the tape.
STOP/EJECT	Press this key to stop the tape, to release other keys when the tape stops after loading or recording programs or data, or to eject the tape.



### ■ MZ-1T01

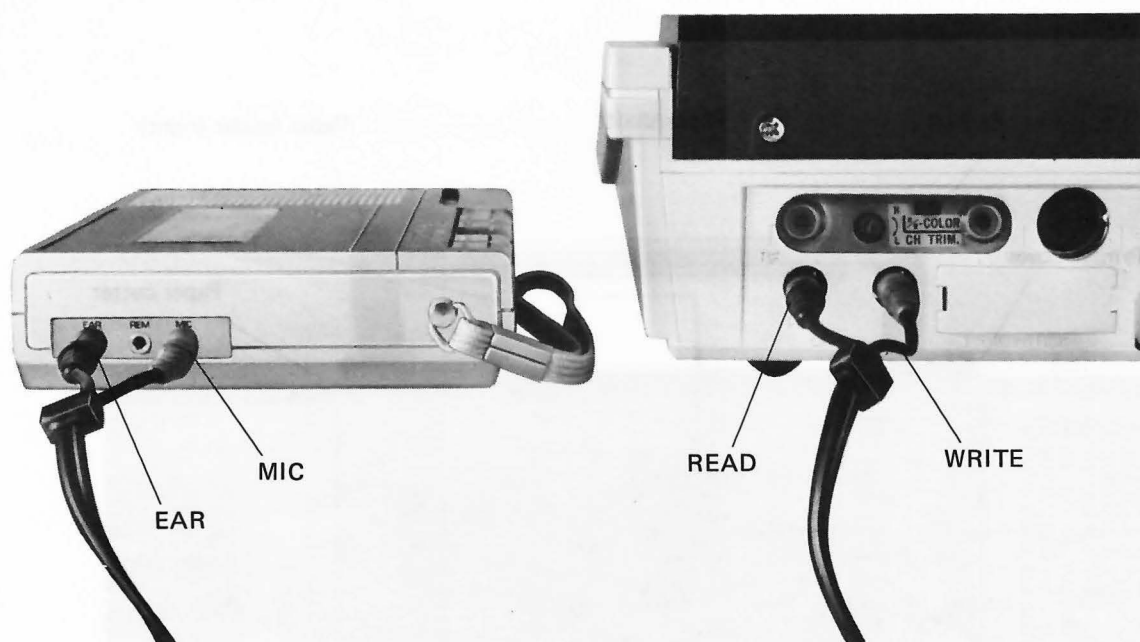
The MZ-1T01 data recorder unit can be installed in the MZ-711 (MZ-710). Installation procedures are as follows.

1. Turn off the computer's power switch and unplug the power cable from the AC outlet.
2. Remove the two screws located on the left side of the rear panel to remove the data recorder compartment cover.



3. Remove the joint connector cover.
4. Plug the connector of the MZ-1T01 onto the 9-pin connector located at the left rear of the recorder compartment of the MZ-711.
5. Position the data recorder in the recorder compartment and fasten it in place with the two screws. When doing this, be careful to avoid catching the connector cable between the data recorder and the computer, (otherwise, the screws cannot be tightened).

## ■ Ordinary cassette tape recorder

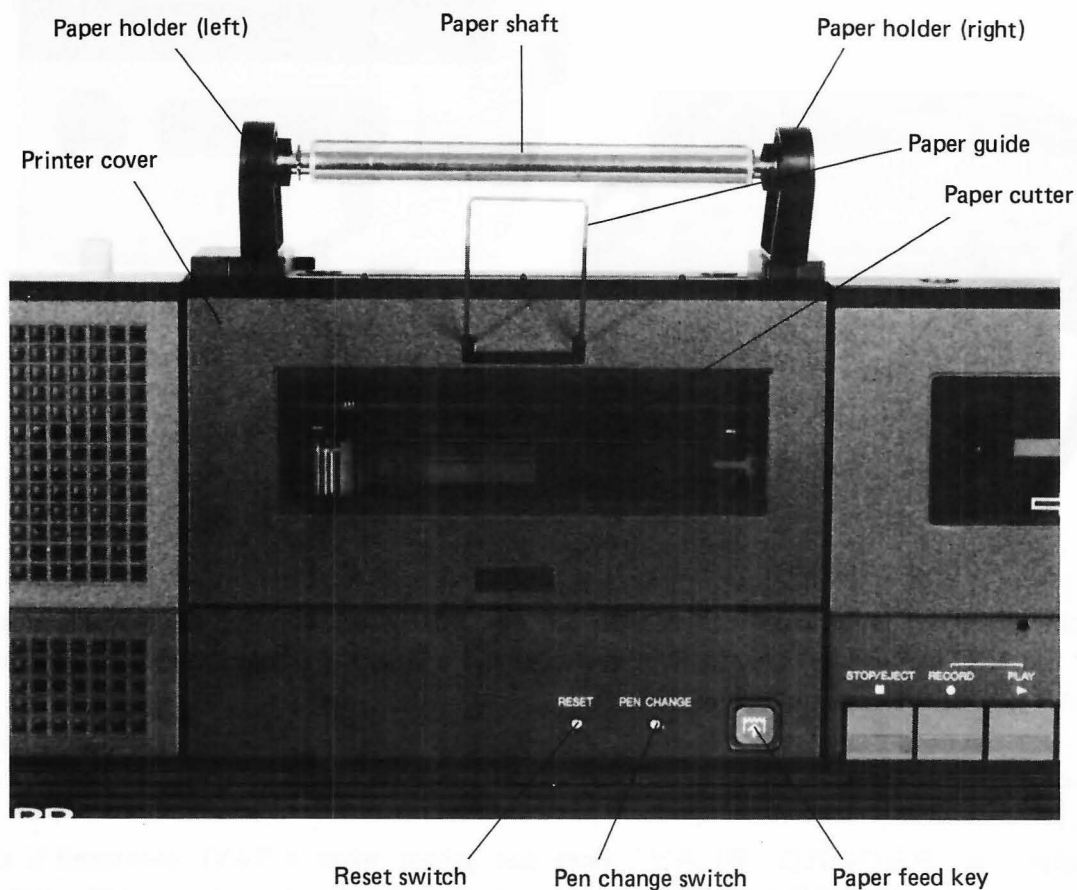


Using commercially available audio cables with 3.5 mm mini-plugs, connect the WRITE jack of the computer to the MIC jack of the cassette tape recorder and connect the computer's READ jack to the EXT SP or EAR jack of the cassette tape recorder.

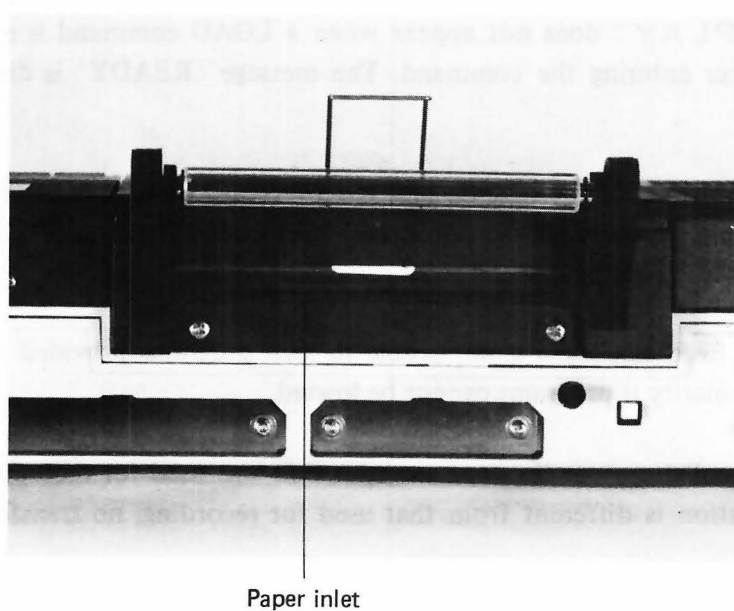
Take note of the following when using an **ordinary cassette tape recorder**.

- (1) The message "  $\perp$  RECORD. PLAY " does not appear when a SAVE command is entered. Be sure to press the RECORD key on the recorder before entering this command. Press the STOP key to stop the recorder after the message "READY" is displayed. Without depressing the STOP key, the recorder is not stopped.
- (2) The message "  $\perp$  PLAY " does not appear when a LOAD command is entered. Be sure to start playing the tape after entering the command. The message "READY" is displayed when loading is completed.
- (3) The level and tone controls of the cassette tape recorder must be adjusted to appropriate levels. Some cassette recorders (e.g. those with the automatic level control) may not be usable. In such cases, please purchase the MZ-1T01.
- (4) The polarity of the head can make it impossible to load programs provided with the computer. Try switching the head polarity if programs cannot be loaded.
- (5) For any transfer or collation, use the tape recorder that was used for recording. If the tape recorder for transfer or collation is different from that used for recording, no transfer nor collation may be possible.
- (6) Data written using an ordinary cassette recorder may not be readable with the data recorder. Therefore, use of the **MZ-1T01** is recommended.

## 3.4 Color Plotter-Printer




Plotter-printer (viewed from the top)



Plotter-printer (viewed from the rear side)



## ■ Loading roll paper

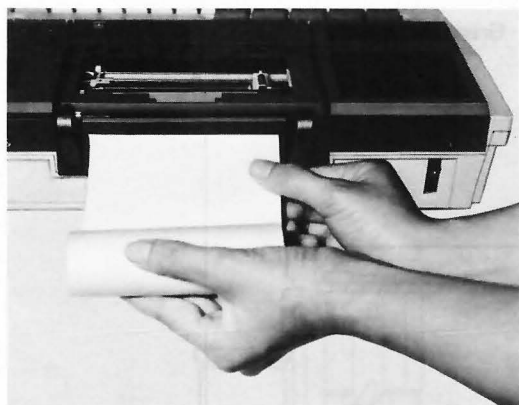
1. Remove the printer cover.
2. Cut the end of roll paper straight across and insert the end into the paper inlet. (Be careful to avoid folding or wrinkling the end of the paper when doing this.)
3. Turn on MZ-731's power switch and press the  (paper feed) key to feed paper until the top of paper is 3 to 5 cm above the outlet.
4. Insert the paper shaft into the roll and mount it to the paper holders.
5. Set the printer cover so that the end of paper comes out through the paper cutter.

- To remove the roll from the printer for replacement, cut straight across the paper at the paper inlet and press the paper feed key.

- Roll paper for the MZ-700 series computers is available at any Sharp dealer. Do not use paper other than that specified.

The length of the paper is 23 to 25 meters, and the maximum roll diameter which can be loaded is 50 mm. Paper will not feed properly if a roll with a greater diameter is used, resulting in poor print quality.

### Procedures for loading roll paper



(A) Insert paper into the paper inlet.



(B) Press the paper feed key to feed paper.

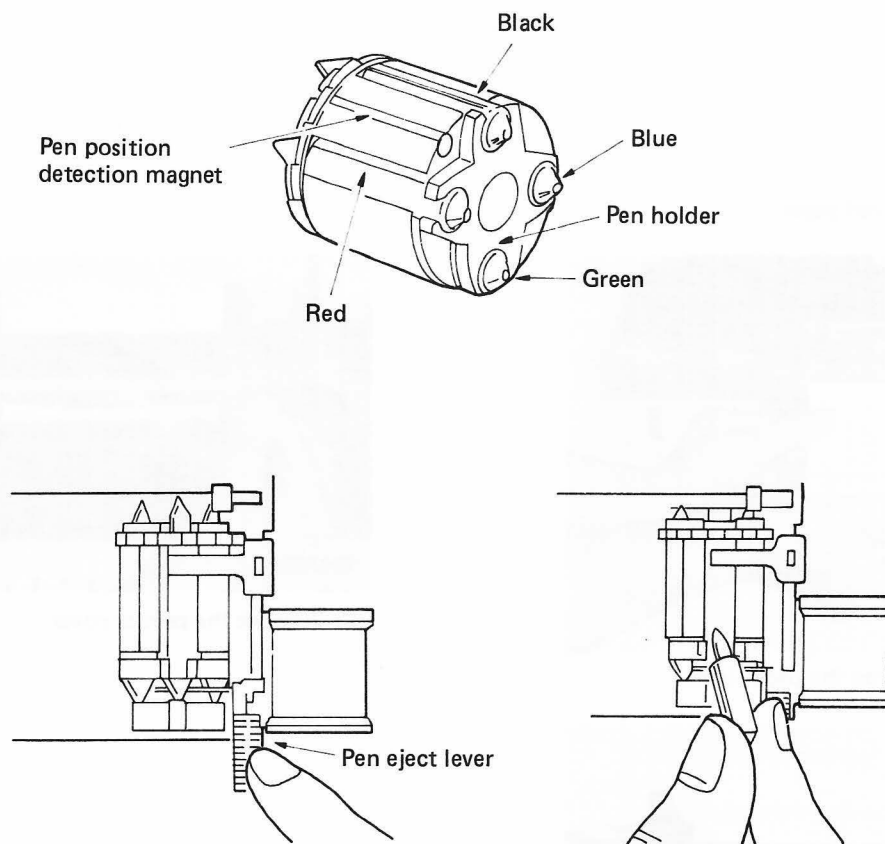


(C) Replace the printer cover.

## ■ Installing/replacing pens

1. Remove the printer cover and press the PEN CHANGE switch with a ball pen or the like; this causes the pen holder to move to the right side of the printer for pen replacement.
2. Depress the pen eject lever to eject the pen which is at the top of the holder. When doing this, rest your finger lightly on top of the pen while pushing the eject lever to prevent it from falling inside the printer.
3. Insert a new pen.
4. Press the PEN CHANGE switch again to bring another pen to the top of the holder.
5. Replace all four pens (black, blue, green and red) in the same manner. When finished, press the RESET switch to ready the printer for printing with the black pen.

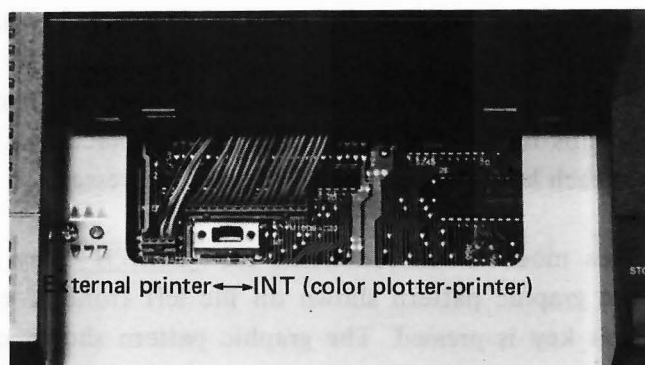
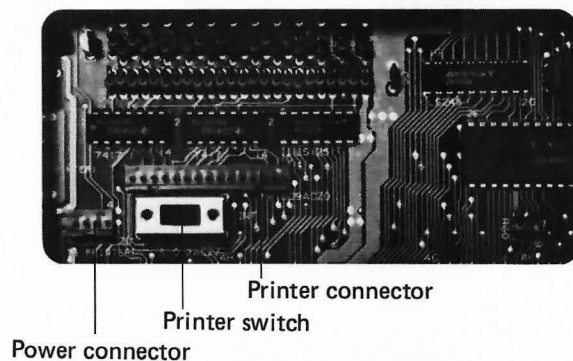
Execute the BASIC TEST command to confirm that all colors are printed correctly.



## ■ MZ-1P01

Installation of the MZ-1P01 color plotter printer (for models other than the MZ-731)

1. Turn off the computer's power switch and unplug the power cable.
2. Remove the two screws located at the center of the rear panel to remove the printer compartment cover.
3. Confirm that the printer switch on the printed circuit board is set to the INT position.
4. Plug the printer connector into the matching connector on the printed circuit board, then position the printer in the printer compartment and fasten it in place with the two screws. When doing this, be careful to avoid catching the connector cable between the data recorder and the computer (otherwise, the screws cannot be tightened).



Connection of color plotter-printer to the MZ-700

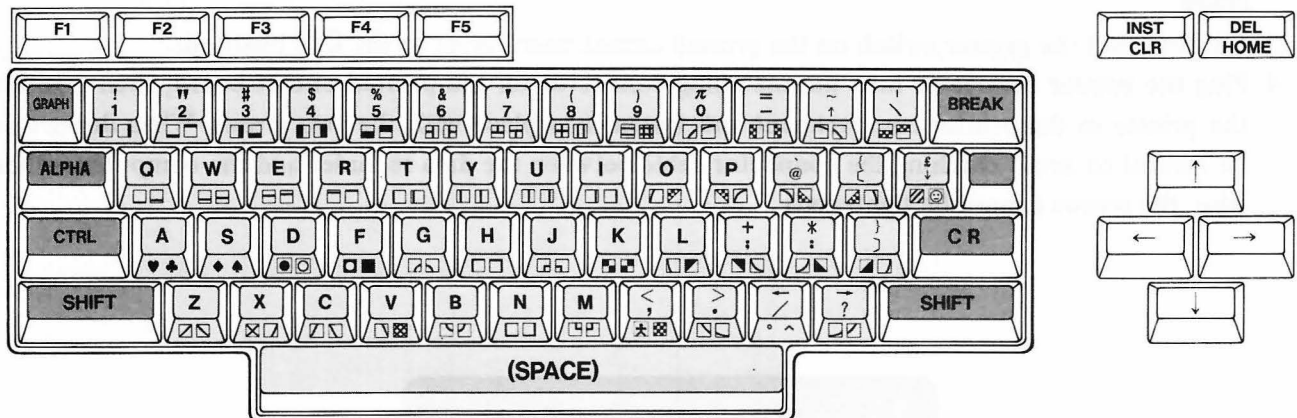
## ■ Connecting an external printer (MZ-80P5(K))

The MZ-80P5(K) printer for the MZ-80K series computers can be connected to the MZ-700's external printer connector (see page 104) without any special interface card. Use an optional connection cable for making the connection.

When using an external printer, the printer switch on the printed circuit board must be set to the external printer position. Therefore, the color plotter-printer and the external printer cannot be used simultaneously.

Note that if a program including color plotter-printer control statements is run with an external printer, meaningless characters (control codes for the plotter-printer) will be printed.

## 3.5 Key Operation



### 3.5.1 Typewriter keyboard

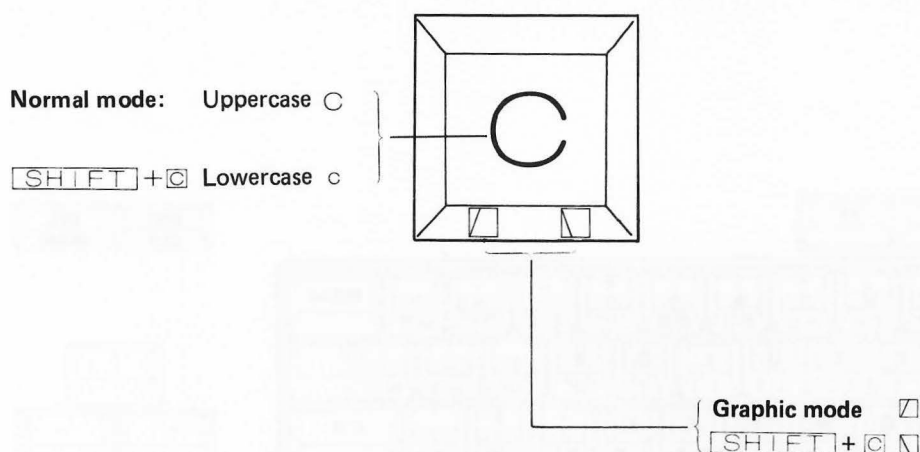
Except for the special control keys, several characters are assigned to each key on the keyboard. The character entered when a key is pressed depends on the input mode selected by the special keys.

The input modes are as follows.

- (1) **Normal mode** . . . . . This mode is automatically entered when the BASIC interpreter is loaded. In this mode, the ASCII character (uppercase or lowercase) shown on top of each key is entered when that key is pressed.
- (2) **Graphic mode** . . . . . This mode is entered when the **GRAPH** key is pressed. In this mode, the graphic pattern shown on the left front of each key is entered when that key is pressed. The graphic pattern shown on the right front of each key is entered by pressing that key together with the shift key. Pressing the **ALPHA** key returns input to the normal mode.



Pressing the space bar enters a space regardless of the input mode.

For example characters entered by the C key in different input modes are as follows.



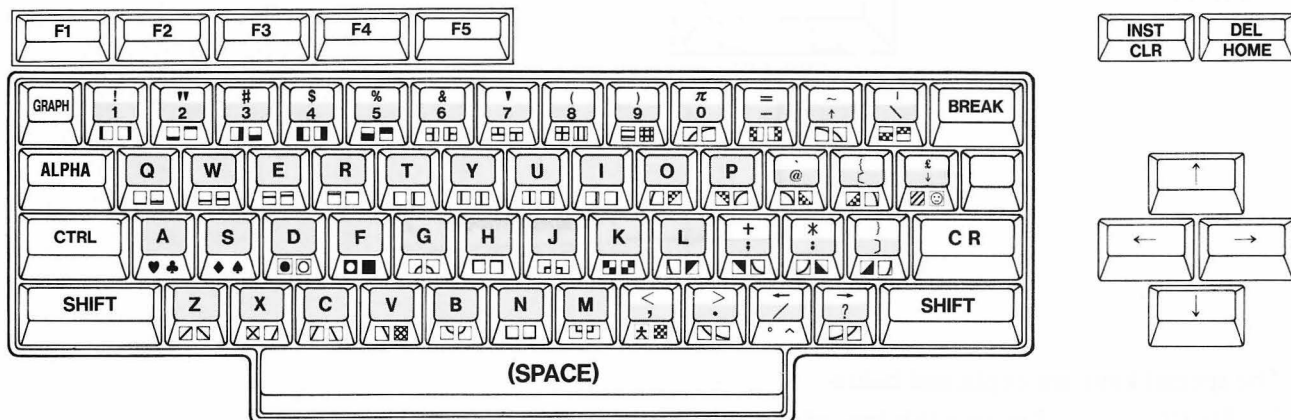
The special keys are explained below.

SHIFT	Pressing this key allows shift position characters to be entered. For alphabetic keys, the shift position characters are lowercase letters; for keys other than alphabetic keys, the shift position characters are those shown on the upper side of the key tops. In the GRAPH mode, the graphic pattern shown on the right front of each key is entered.
C R	Pressing this key enters a CR (carriage return) code, terminating the line and moving the cursor to the beginning of the next line.
BREAK	Pressing this key enters a BREAK code. Pressing it together with the SHIFT key stops execution of a program or operation of the data recorder.
GRAPH	Pressing this key changes the input mode from normal to graphic for input of the graphic patterns shown on the left front of keys.
ALPHA	Pressing this key changes the input mode from graphic to normal.

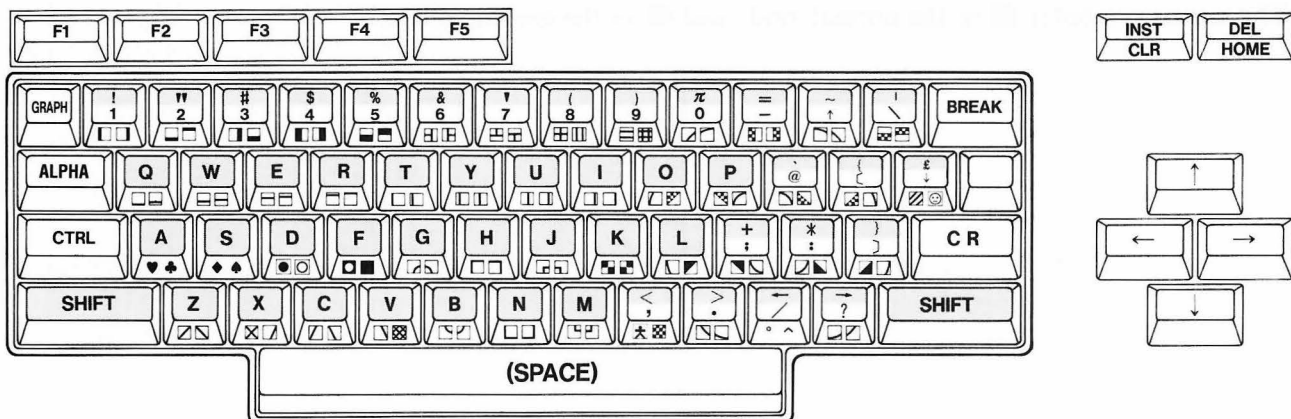
The cursor symbol is  in the normal mode and  in the graphic mode.

### (1) Normal mode (alphanumeric mode)

Character entered by each key in the normal mode are as indicated by the screened areas in the figure below.



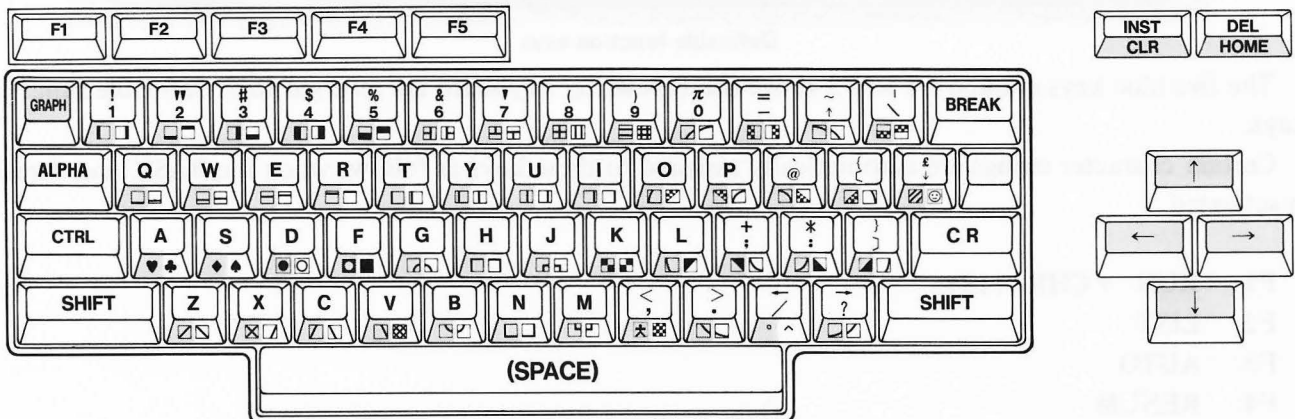
When with the **SHIFT** key is pressed together with other keys, lowercase letters (or other symbols indicated by the screen areas in the figure below) are entered.



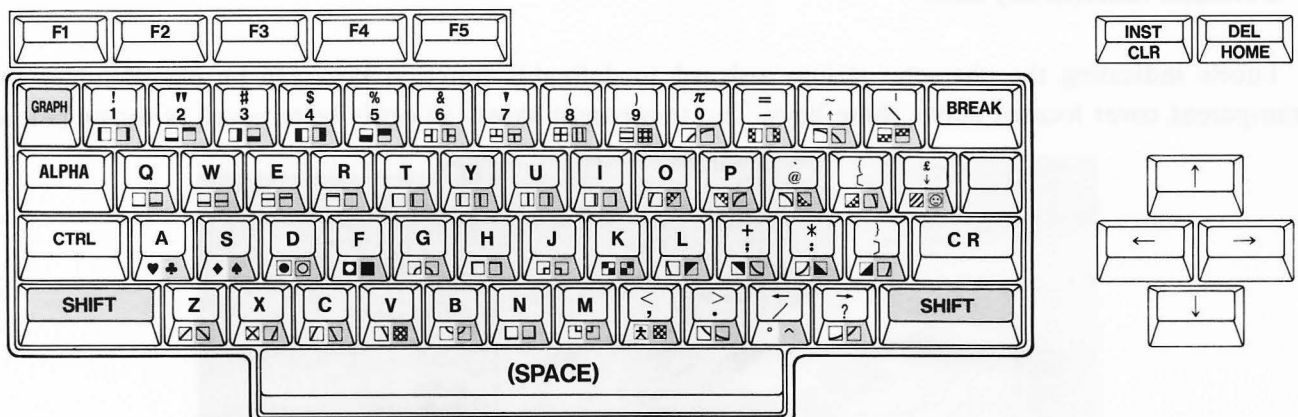


## (2) Graphic mode

Pressing the **GRAPH** key places the computer in the graphic input mode. Characters entered by each key in the graphic mode are as indicated by the screened areas in the figure below. In this mode, pressing any of the cursor control keys, the INST/CLR key or the DEL/HOME key enters the symbols **↑**, **←**, **→**, **↓**, **⊞**, or **⊟**, respectively.



When with the **SHIFT** key is pressed together with other keys, symbols indicated by the screen areas in the figure below are entered.



The cursor symbol is **⊞** in the graphic mode. To return the mode to normal, press the **ALPHA** key.

### 3.5.2 Definable function keys



Definable function keys

The five blue keys marked F1 to F5 above the typewriter keyboard are referred to as definable function keys.

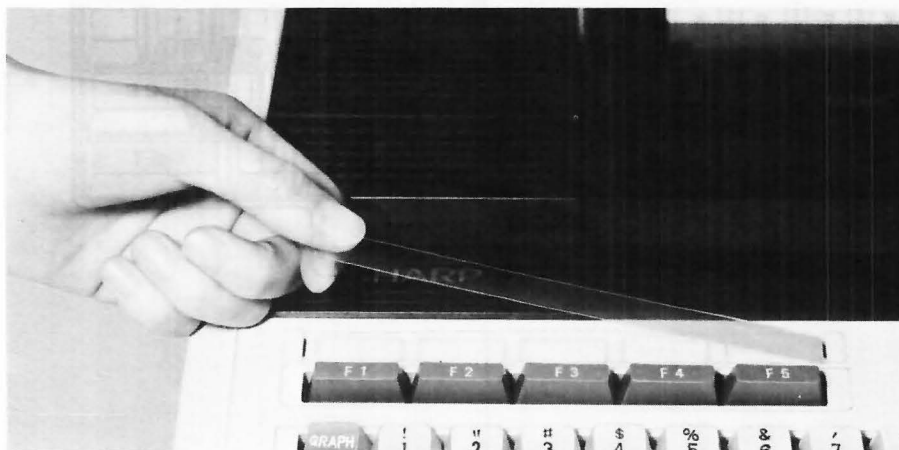
Certain character strings are automatically assigned to these keys as follows when the BASIC interpreter is activated.

F1: " RUN " + CHR\$ (13)  
F2: " LIST "  
F3: " AUTO "  
F4: " RENUM "  
F5: " COLOR "  
SHIFT + F1: " CHR\$ ("  
SHIFT + F2: " DEF KEY ("  
SHIFT + F3: " CONT "  
SHIFT + F4: " SAVE "  
SHIFT + F5: " LOAD "

When one of these keys is pressed, the character string assigned to that key is entered; thus, statements which are frequently used can be entered just by pressing one key. The character string assigned to any of the definable function keys can be changed by the DEF KEY statement. (See page 57, DEF KEY statement.)

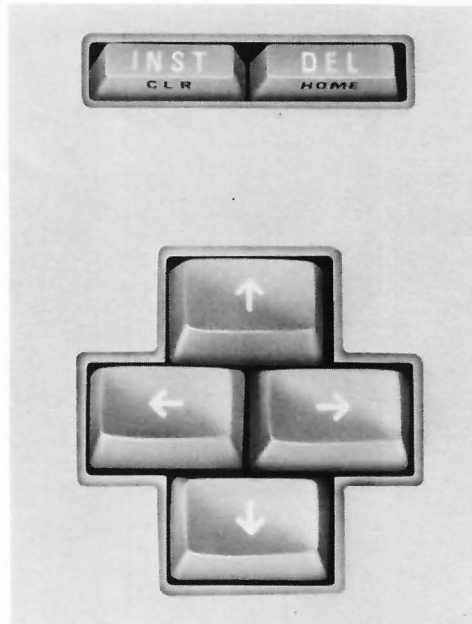
#### ■ Definable function key label

Labels indicating the character strings assigned to definable function keys can be placed under the transparent cover located above these keys. The transparent sheet can easily be removed as shown below.



---

### 3.5.3 Cursor control keys and insert and delete keys



Cursor control keys and insert and delete keys

The cursor control keys are the four yellow keys at the right of the keyboard which are marked with arrows.

Pressing these keys moves the cursor one position in the direction indicated by the arrow. These keys are used when editing programs.

The 

INST
CLR

 and 

DEL
HOME

 key have the following functions.

INST
CLR

Inserts a space at the position of the cursor and shifts all following characters one position to the right. INST: insert.

DEL
HOME

Erases the character to the left of the cursor and shifts all following characters one position to the left. DEL: delete.

SHIFT
-------

 + 

INST
CLR

Clears the entire screen and returns the cursor to the screen's upper left corner. Pressing this key does not affect the program in memory. CLR: clear.

SHIFT
-------

 + 

DEL
HOME

Returns the cursor to the upper left corner of the screen (does not affect any characters displayed).

See pages 18 and 19.



## Hardware

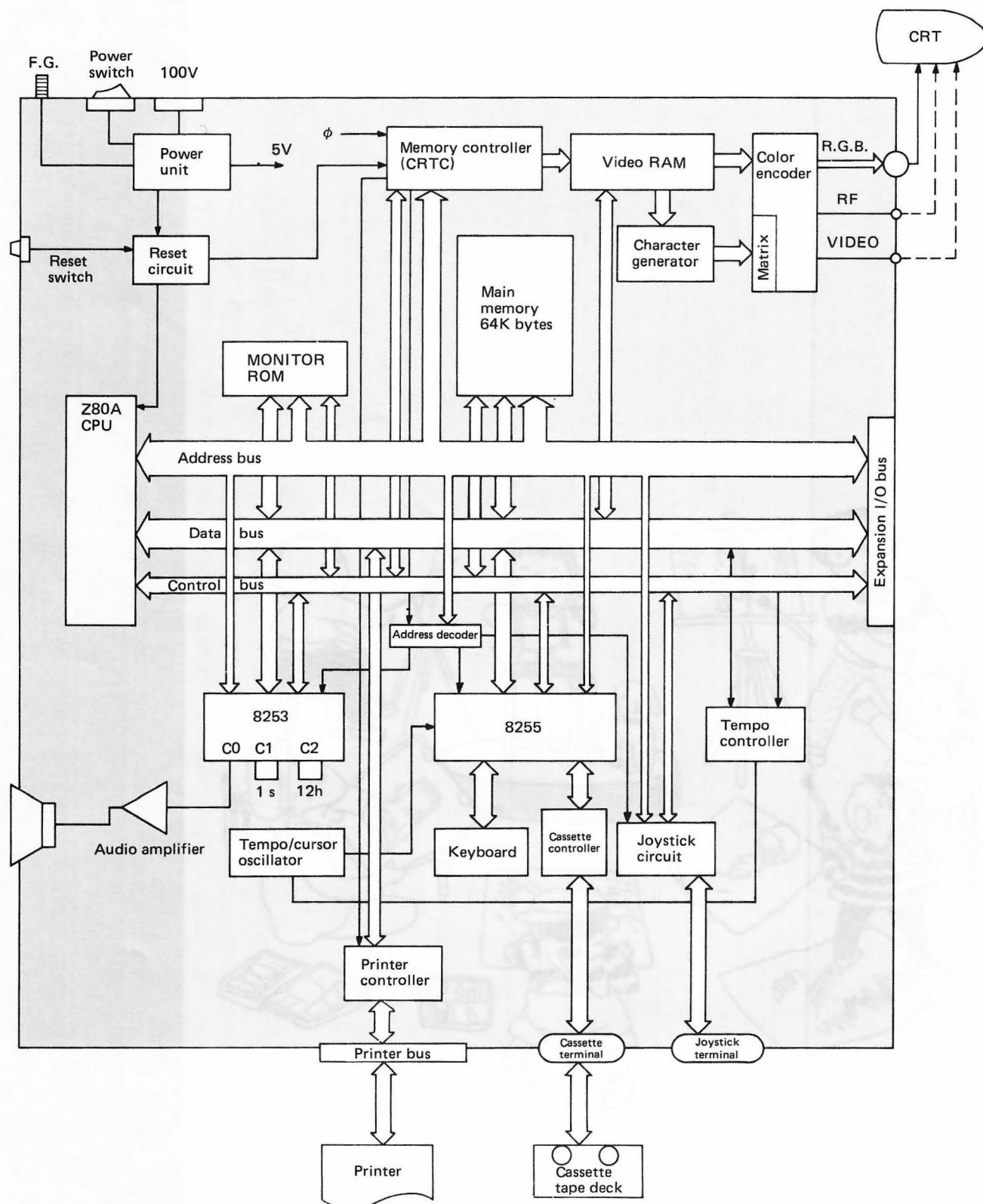


Notice: The contents of this chapter are for reference only, and Sharp cannot assume responsibility for answering any questions about its contents.

## 4.1 MZ-700 System Diagram

The figure below shows the system configuration of the MZ-700 series computers.

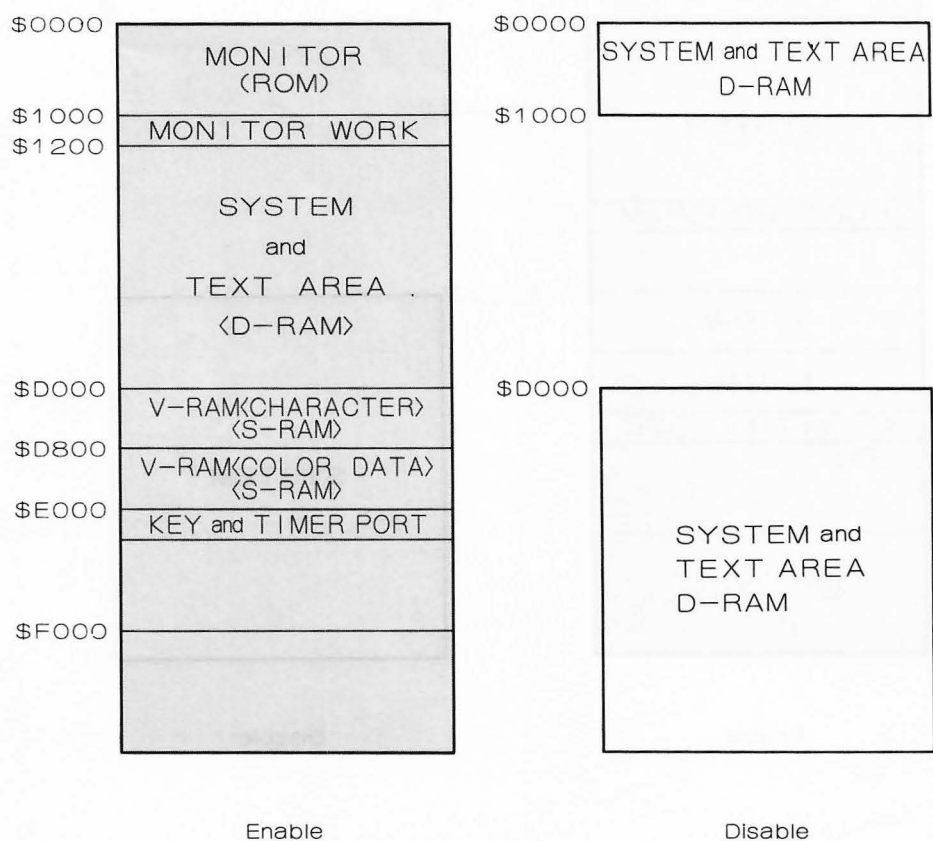
RF and VIDEO terminal is not included in the MZ-710.





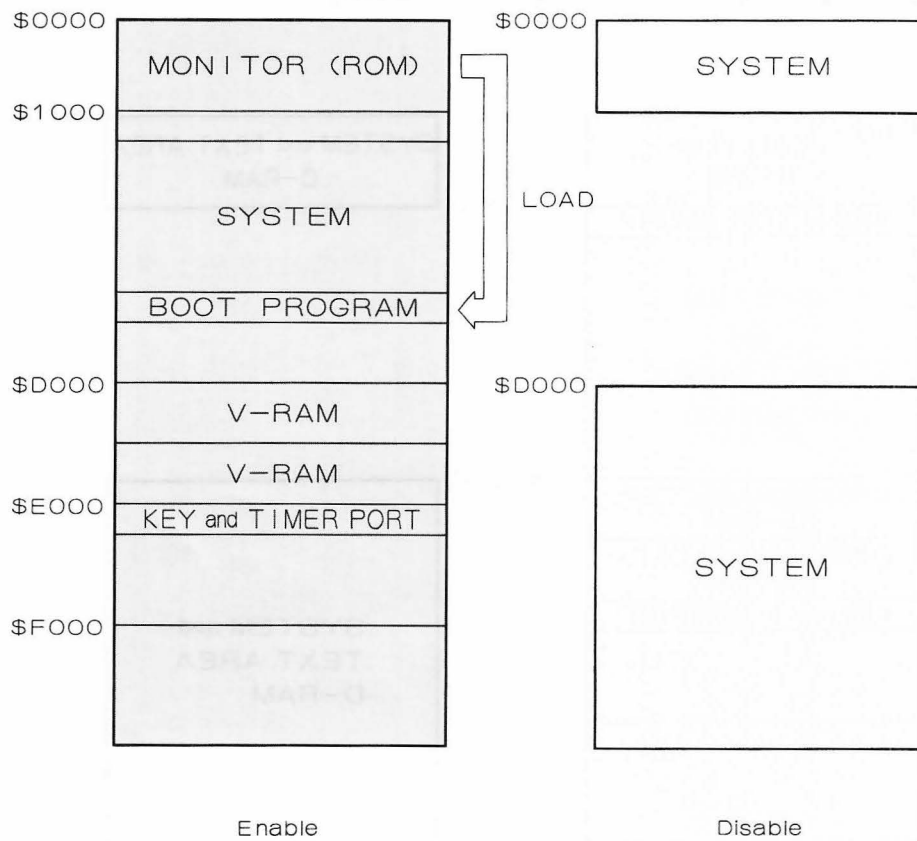
## 4.2 Memory configuration

### 4.2.1 Memory map at power-on (80k mode)



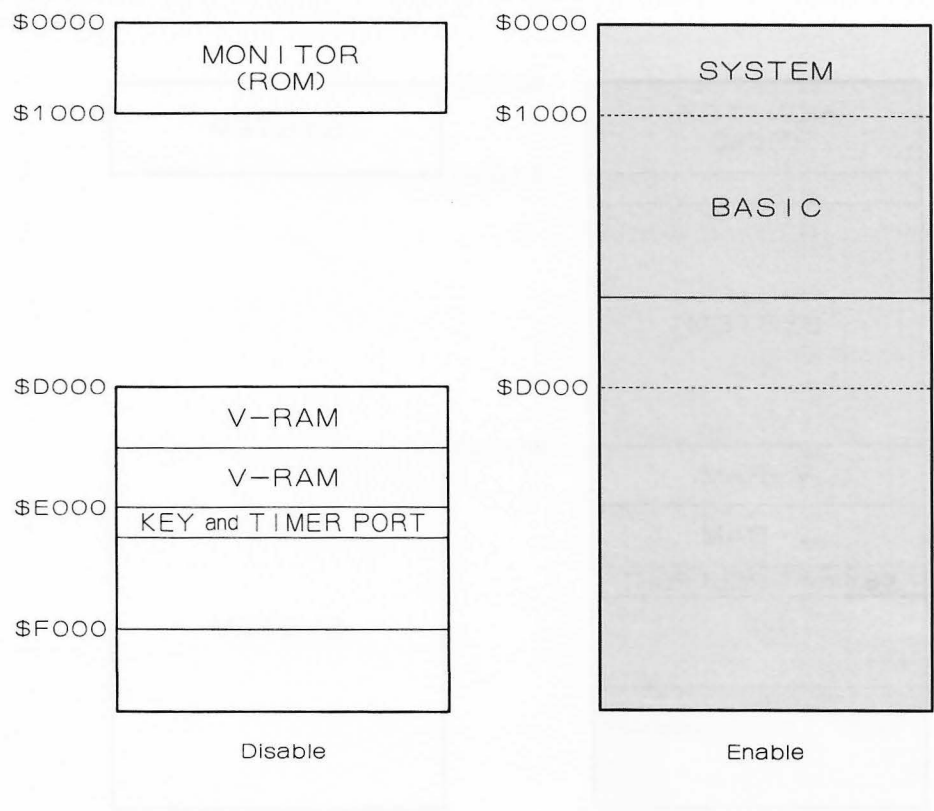
- The memory map is as shown above immediately after the power has been turned on. (The contents of the V-RAM area from \$D000 to \$DFFF are not the same as those of MZ-80K.)
- The entry point of the monitor ROM is the same as that of the MZ-80K.

## 4.2.2 Memory map while loading system program (BASIC)



- When the monitor LOAD command is entered, the bootstrap loader is loaded into the system RAM area from ROM and control is transferred to that program.
- BOOT COMMAND : L

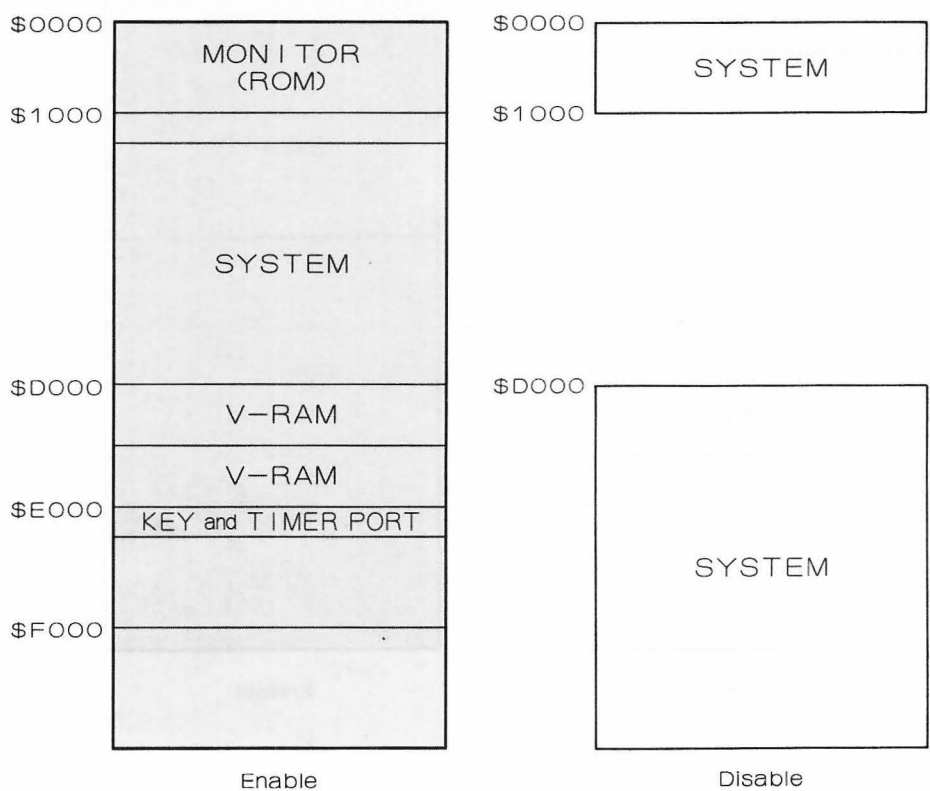
### 4.2.3 Memory map after the BASIC interpreter has been loaded (MZ-700 mode)



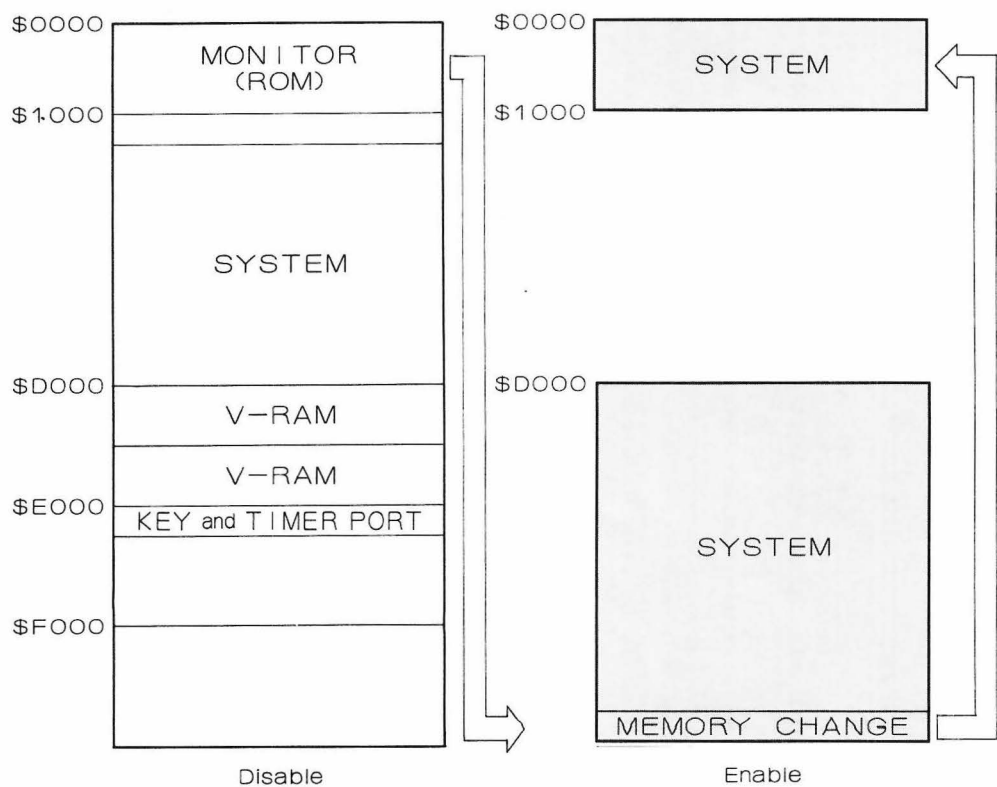
- The memory map is as shown above after the BASIC interpreter has been loaded.
- Bank switching is performed to access V-RAM or the KEY and TIMER PORT area.

### 4.2.4 Memory map after manual reset

The memory map is as shown below after the reset switch on the rear panel has been pressed.



After pressing the reset switch together with the **CTRL** key, the memory map is as shown below.



- When the reset switch is pressed together with the **CTRL** key, addresses \$0000 to \$0FFF and from \$D000 to \$FFFF are assigned to RAM.
- When the # command is entered after the reset switch has been pressed, the computer operates in the same manner as after the reset switch has been pressed together with the **CTRL** key.

## 4.2.5 Bank switching

- a) Memory blocks can be selected by outputting data to I/O ports as shown below.

SWITCHING

I/O PORT	\$0000~\$0FFF	\$D000~\$FFFF
\$ E0	SYSTEM AREA (D-RAM)	
\$ E1		SYSTEM AREA (D-RAM)
\$ E2	MONITOR (ROM)	
\$ E3		V-RAM, KEY, TIMER
\$ E4	MONITOR (ROM)	V-RAM, KEY, TIMER
\$ E5		Inhibit
\$ E6		Return to the front of condition, where being inhibited by \$ E5.

**Note:** Outputting data to I/O port \$E4 performs the same function as pressing the reset switch.

- b) Examples:

**OUT (\$E0), A**

Assigns addresses \$0000 to \$0FFF to RAM, but does not change execution address. The contents of variable A do not affect the result.

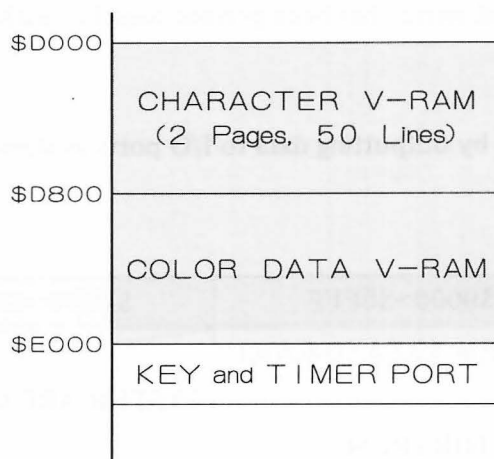
**OUT (\$E4), A**

Initializes memory to the state immediately after the power has been turned on.

**Note:** Since the program counter is not moved by the OUT statement, care must be taken when switching memory blocks if the program counter is located in the area from \$0000 to \$0FFF or from \$D000 to \$FFFF.

## 4.2.6 Memory map when V-RAM is accessed

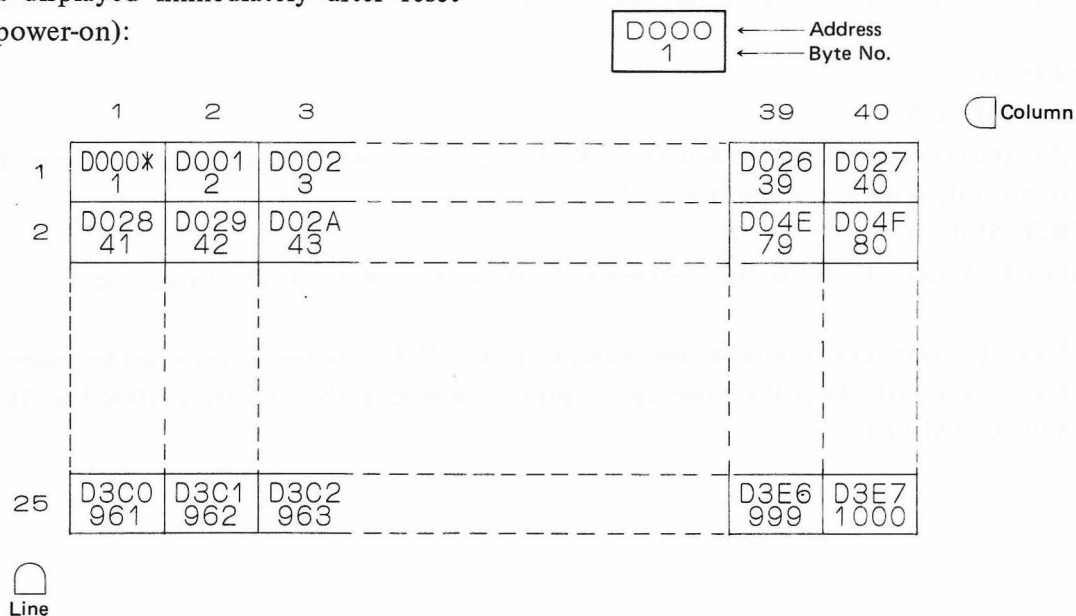
### i) V-RAM (Video RAM) memory map



### ii) Correspondence between V-RAM address and location on the screen.

The MZ-700 has a 2K byte V-RAM area, but only 1K byte of that area can be displayed on the screen at one time. The area displayed can be changed by scrolling the screen.

#### a) Area displayed immediately after reset (or power-on):





b) Area displayed after the screen has been scrolled up one line from the end of V-RAM:

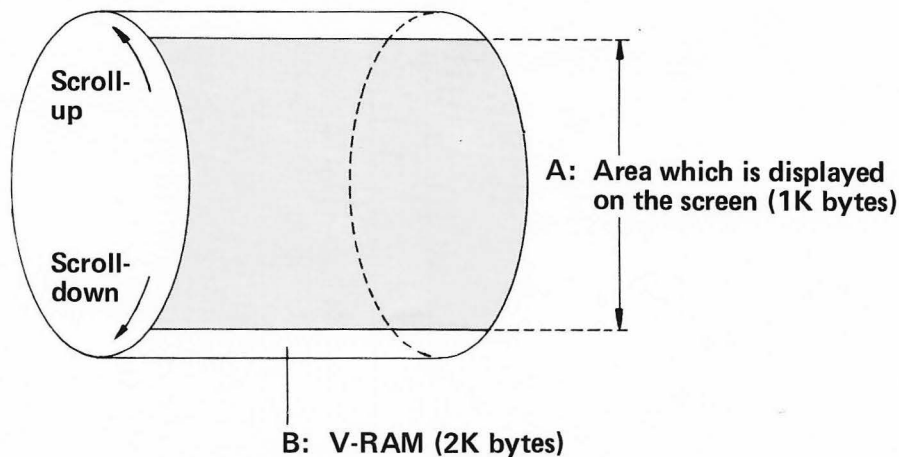
	1	2	3		39	40
1	D000 1041	D001 1042	D002 1043		D026 1079	D027 1080
2	D028 1081	D029 1082	D02A 1083		D04E 1119	D04F 1120
24	D398 1961	D399 1962	D39A 1963		D3BE 1999	D3BF 2000
25	D3C0 1	D3C1 2	D3C2 3		D3E6 39	D3E7 40

**Note:** The line consisting of bytes 1 to 40 is wrapped around to that consisting of bytes 1961 to 2000 as shown above.

iii) Scroll-up and scroll-down

a) The screen is scrolled up by pressing the **SHIFT** and **↑** keys together, and is scrolled down by pressing the **SHIFT** and **↓** keys together.

b) Scroll-up and scroll-down



- During scrolling, the area which is displayed on the screen moves through the 2K byte V-RAM area as shown above.
- The end of the V-RAM area is warped around to the beginning of V-RAM as shown above.
- The cursor does not move on the screen during scrolling.

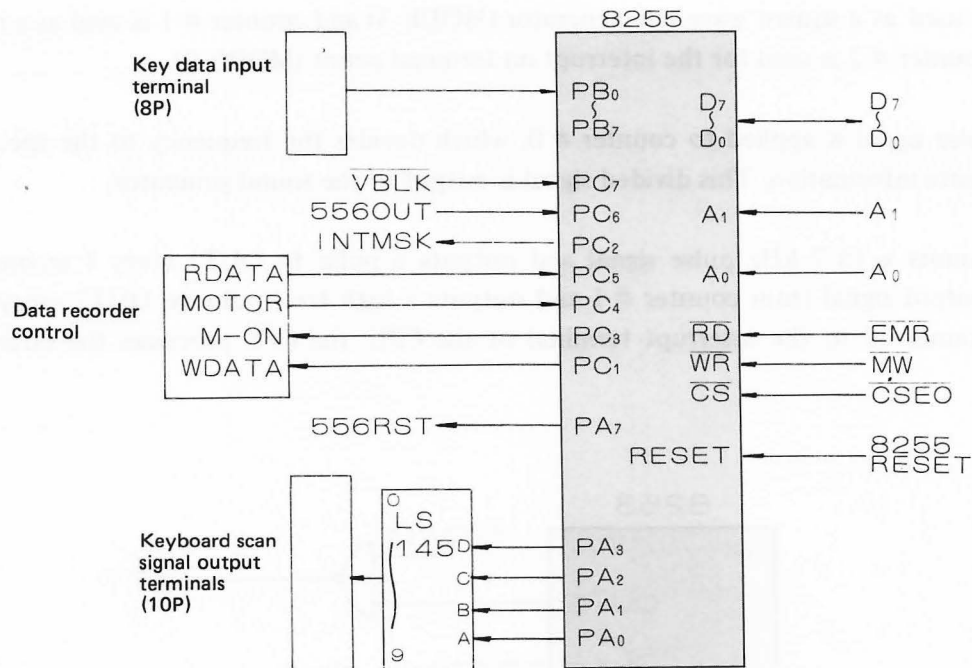
## 4.3 Memory Mapped I/O (\$E000-\$E008)

Addresses \$E000 to \$E008 are assigned to the 8255 programmable peripheral interface, 8253 programmable interval timer and other I/O control ICs so that various I/O devices (including music functions using counter #0 of the 8253) can be accessed in the same manner as memory. The memory mapped I/O chart is shown below.

CPU memory address	Controller	Operation
\$E000 \$E001 \$E002 \$E003	8255	P <sub>A</sub> : Output P <sub>B</sub> : Input P <sub>C</sub> : Input and output control by bit setting Mode control
\$E004 \$E005 \$E006 \$E007	8253	C <sub>0</sub> : Mode 3 (square wave rate generator) C <sub>1</sub> : Mode 2 (rate generator) C <sub>2</sub> : Mode 0 (terminal counter) Mode control
\$E008	LS367, etc.	Tempo, joystick and HBLNK input

### 4.3.1 Signal system of the 8255

The 8255 outputs keyboard scan signals, input key data, and controls the cassette tape deck and cursor blink timing.



Port	Terminal	I/O	Active state	Description of control	Name of signal
PA (\$E000)	PA <sub>0</sub>	OUT	H	Keyboard scan signals	556 RST
	PA <sub>1</sub>		H		
	PA <sub>2</sub>		H		
	PA <sub>3</sub>		H		
	PA <sub>7</sub>		L	Resets the cursor blink timer.	
PB (\$E001)	PB <sub>0</sub>	IN	L	Key scanning data input signals	
	PB <sub>1</sub>		L		
	PB <sub>2</sub>		L		
	PB <sub>3</sub>		L		
	PB <sub>4</sub>		L		
	PB <sub>5</sub>		L		
	PB <sub>6</sub>		L		
	PB <sub>7</sub>		L		
PC* (\$E002)	PC <sub>1</sub>	OUT	—	Cassette tape write data	WDATA
	PC <sub>2</sub>	OUT	L	Inhibits clock interrupts.	INTMSK
	PC <sub>3</sub>	OUT		Motor drive signal	M-ON
	PC <sub>4</sub>	IN	H	Indicates that the motor is on.	MOTOR
	PC <sub>5</sub>	IN	—	Cassette tape read data	RDATA
	PC <sub>6</sub>	IN	—	Cursor blink timer input signal	556 OUT
	PC <sub>7</sub>	IN	—	Vertical blanking signal	VBLK

\* Each output data bit can be independently set or reset.

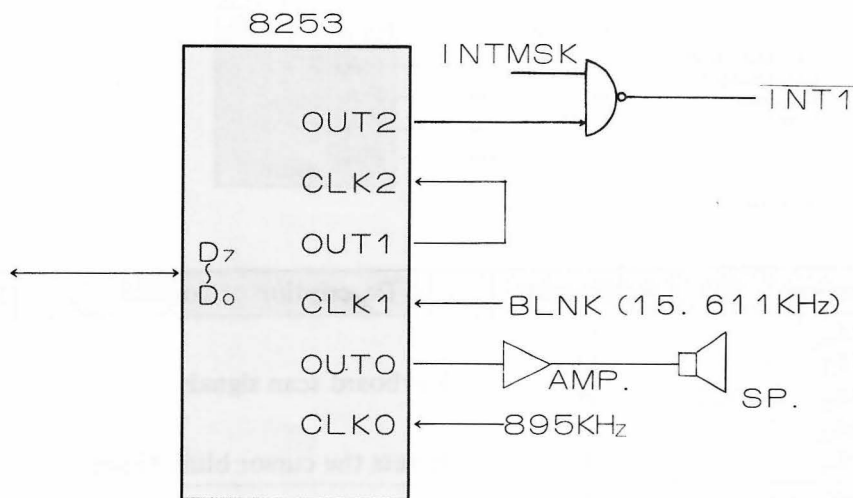
### 4.3.2 Signal system of the 8253

The 8253 includes three counters # 0, # 1 and # 2. Counter # 0 is used for sound generation, and counter # 1 and # 2 are used for the built-in clock.

Counter # 0 is used as a square wave rate generator (MODE 3) and counter # 1 is used as a rate generator (MODE 2). Counter # 2 is used for the interrupt on terminal count (MODE 0).

A 895 kHz pulse signal is applied to counter # 0, which divides the frequency to the specified value according to the note information. This divided signal is output to the sound generator.

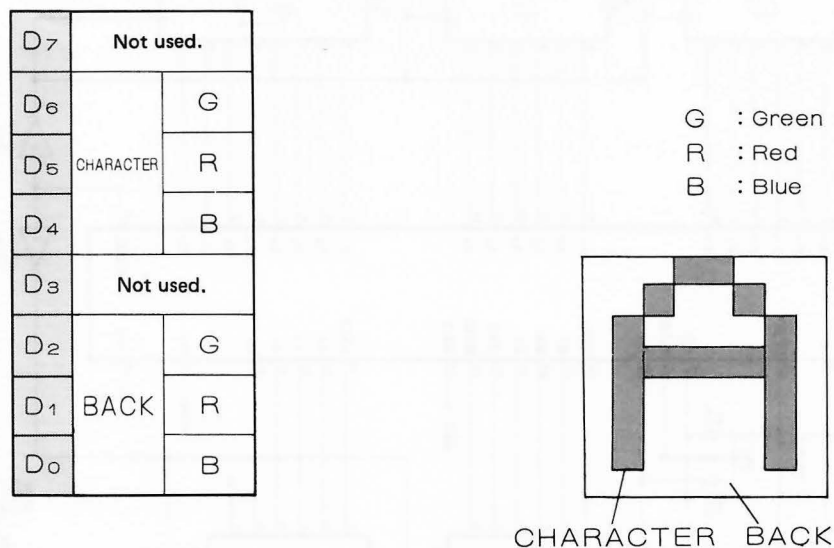
Counter # 1 counts a 15.7 kHz pulse signal and outputs a pulse to OUT1 every 1 second. Counter # 2 counts the output signal from counter # 1 and outputs a high level pulse to OUT2 every 12 hours. Since OUT2 is connected to the interrupt terminal of the CPU, the CPU processes the interrupt every 12 hours.



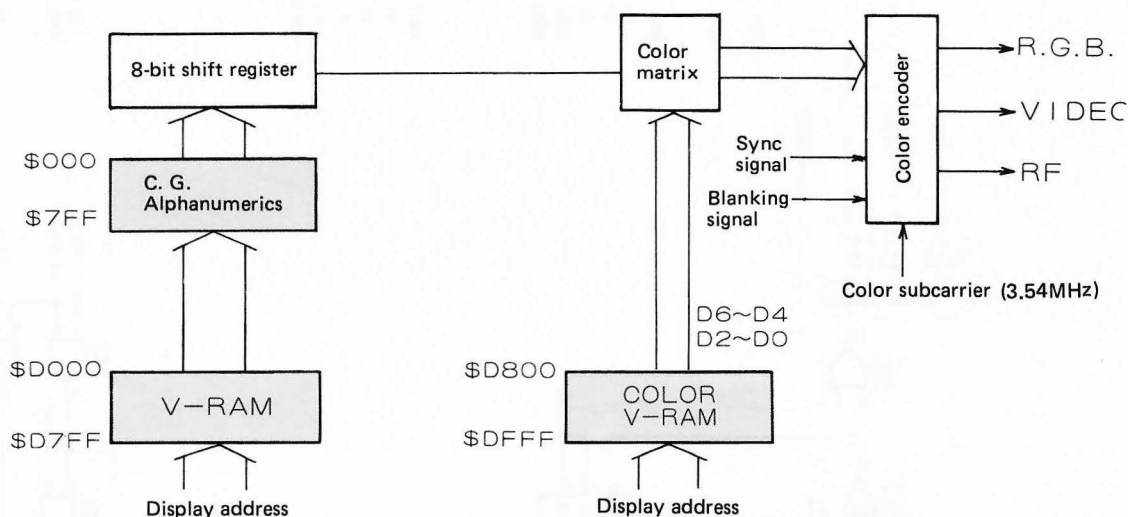
## 4.4 Signal System of Color V-RAM

Color information of the MZ-700 is controlled in character units; that is, a 1-byte color information table is assigned to each character displayed on the screen.

A color information table is shown in the figure below.



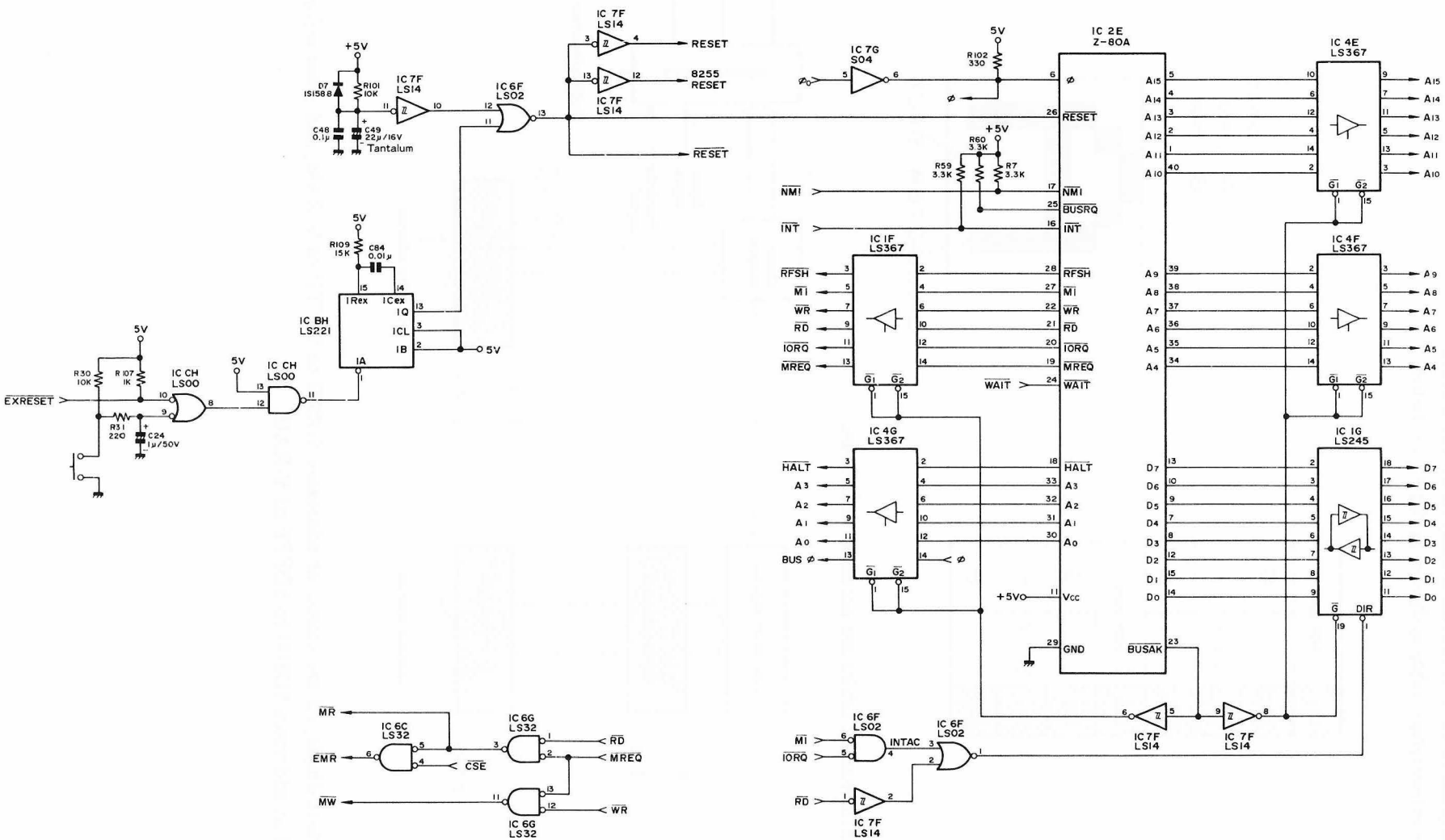
Color information tables are accessed as follows.



Characters displayed are stored at addresses \$D000 to \$D7FF of V-RAM, and color information tables are stored at addresses \$D800 to \$DFFF of V-RAM.

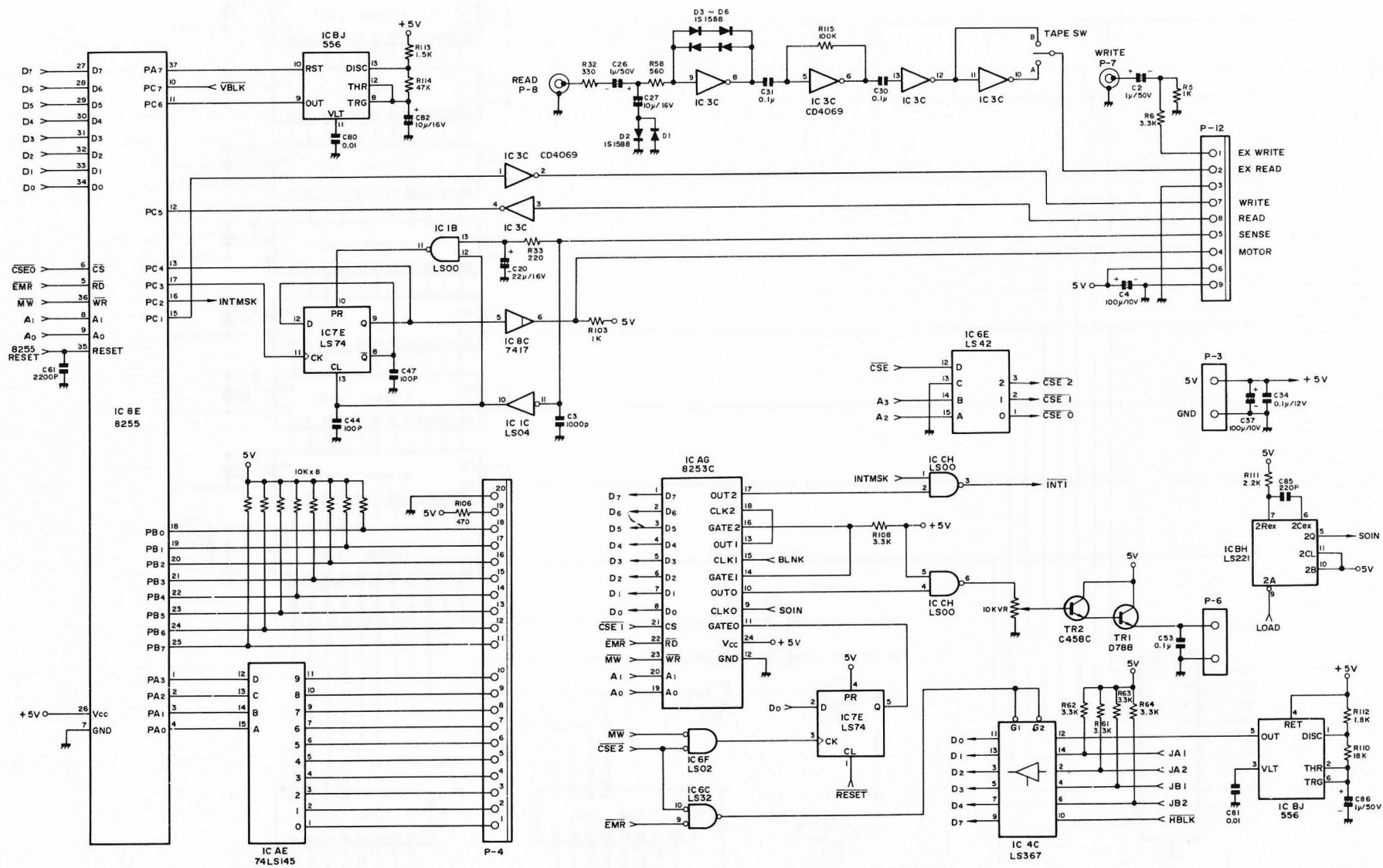
# 4.5 MZ-700 Circuit Diagrams

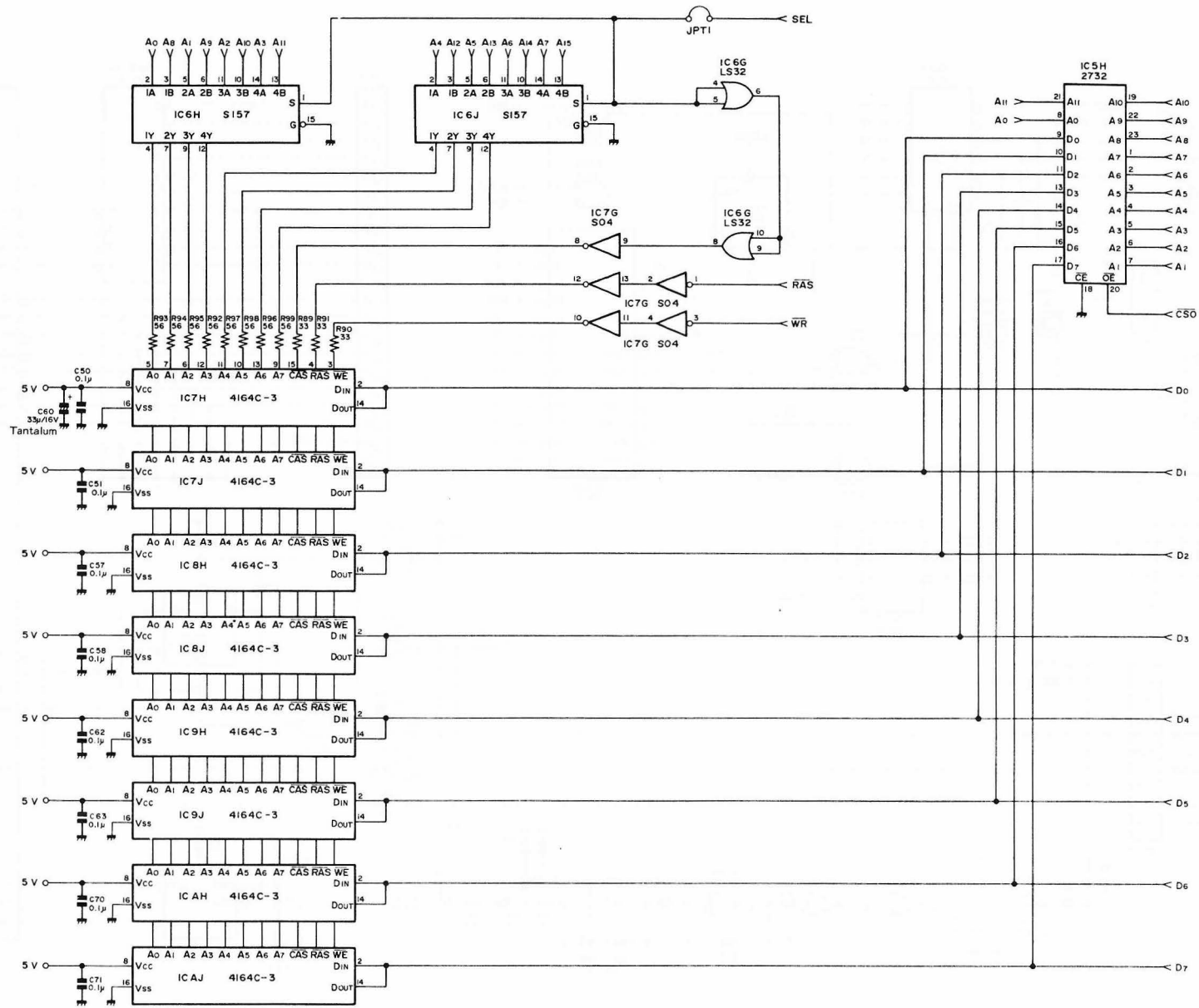
[CPU board circuit (1)]





## [CPU board circuit (2)]



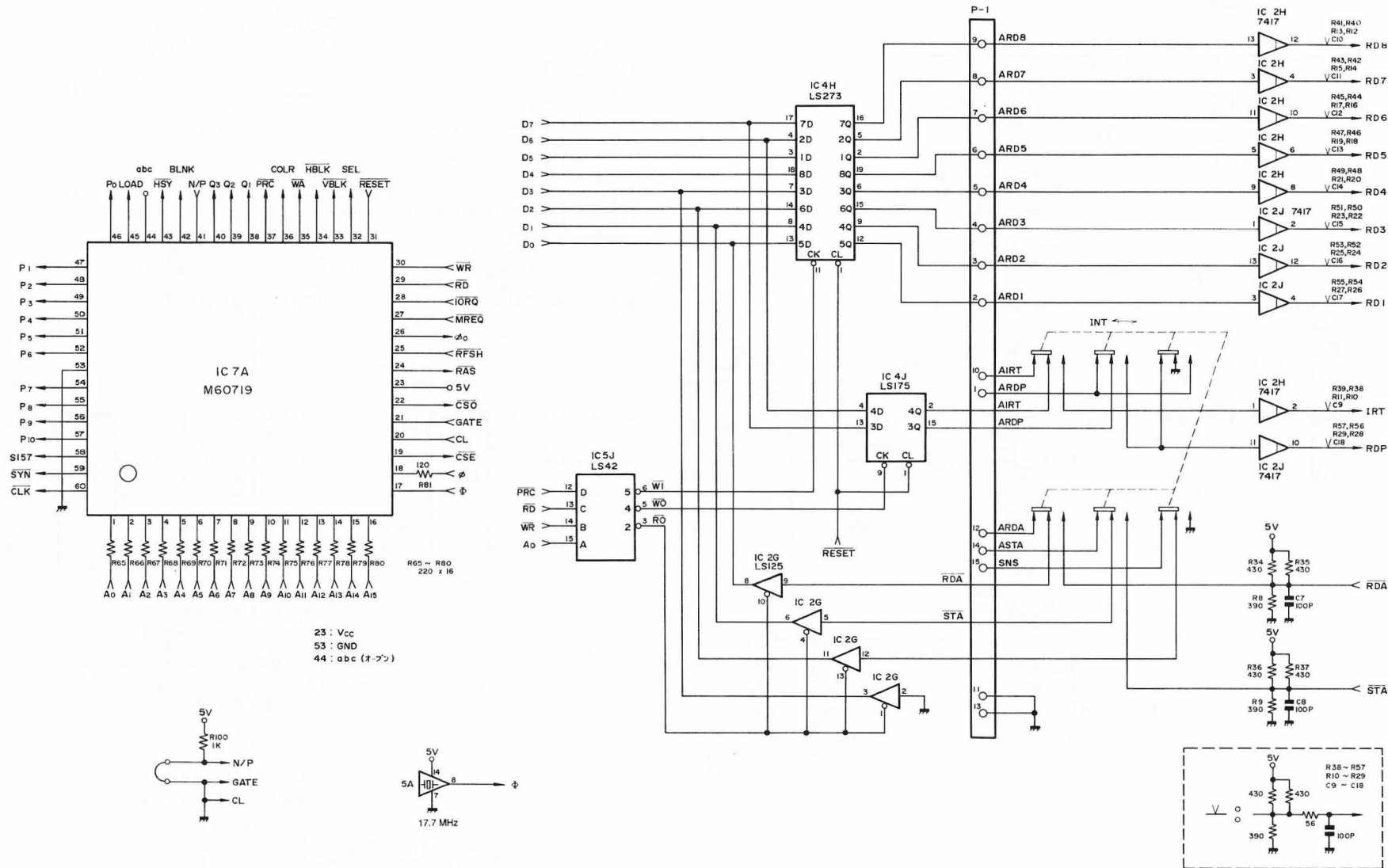


[CPU board circuit (3)]

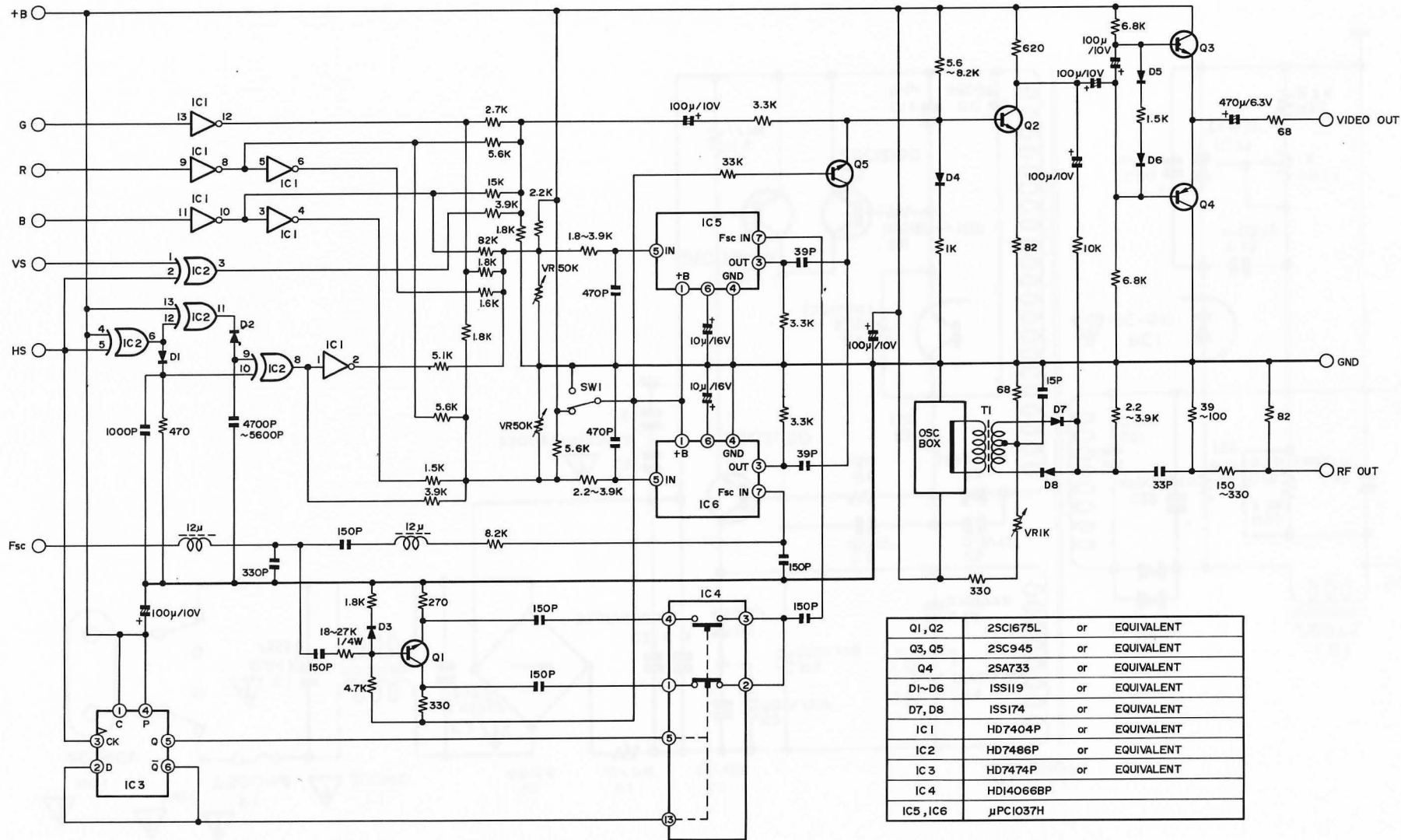
---

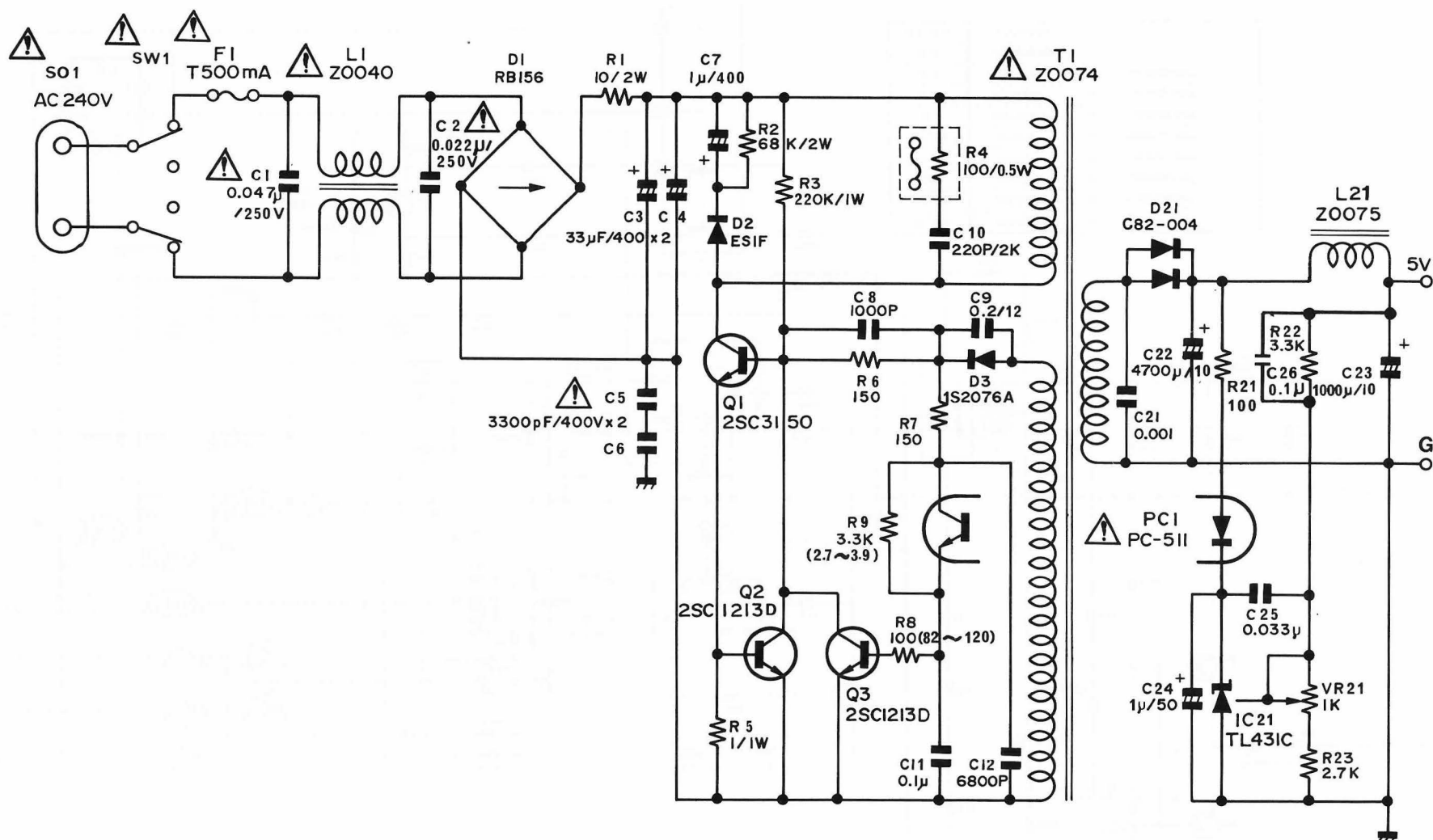


## [CPU board circuit (5)]



[Color encoder circuit]





[Power unit]

[CPU board terminal configuration]

P-1

1	ARDP
2	ARD <sub>1</sub>
3	ARD <sub>2</sub>
4	ARD <sub>3</sub>
5	ARD <sub>4</sub>
6	ARD <sub>5</sub>
7	ARD <sub>6</sub>
8	ARD <sub>7</sub>
9	ARD <sub>8</sub>
10	AIRT
11	GND
12	ARDA
13	GND
14	ASTA
15	ALPS

P-5

1	+ 5 V
2	+ 5 V
3	GND
4	GND

P-10

1	RDP	2	GND
3	RD <sub>1</sub>	4	GND
5	RD <sub>2</sub>	6	GND
7	RD <sub>3</sub>	8	GND
9	RD <sub>4</sub>	10	GND
11	RD <sub>5</sub>	12	GND
13	RD <sub>6</sub>	14	GND
15	RD <sub>7</sub>	16	GND
17	RD <sub>8</sub>	18	GND
19	IRT	20	GND
21	RDA	22	GND
23	STA	24	GND
25	FG	26	FG

P-11

49	A <sub>15</sub>	NMI	50
47	A <sub>14</sub>	EXINT	48
45	A <sub>13</sub>	GND	46
43	A <sub>12</sub>	MREQ	44
41	A <sub>11</sub>	GND	42
39	A <sub>10</sub>	IORQ	40
37	A <sub>9</sub>	GND	38
35	A <sub>8</sub>	RD	36
33	A <sub>7</sub>	GND	34
31	A <sub>6</sub>	WR	32
29	A <sub>5</sub>	EXWAIT	30
27	A <sub>4</sub>	M <sub>1</sub>	28
25	A <sub>3</sub>	GND	26
23	A <sub>2</sub>	HALT	24
21	A <sub>1</sub>	EXRESET	22
19	A <sub>0</sub>	RESET	20
17	BUS $\phi$	GND	18
15	D <sub>7</sub>	GND	16
13	D <sub>6</sub>	GND	14
11	D <sub>5</sub>	GND	12
9	D <sub>4</sub>	GND	10
7	D <sub>3</sub>	GND	8
5	D <sub>2</sub>	GND	6
3	D <sub>1</sub>	GND	4
1	D <sub>0</sub>	GND	2

P-13

1	5 V
2	VBLK
3	JA 1
4	JA 2
5	GND

P-14

1	5 V
2	VBLK
3	JB 1
4	JB 2
5	GND

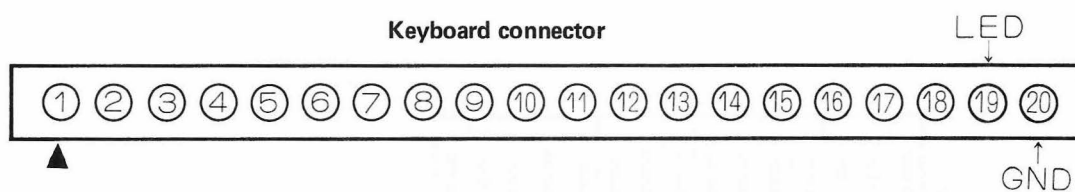
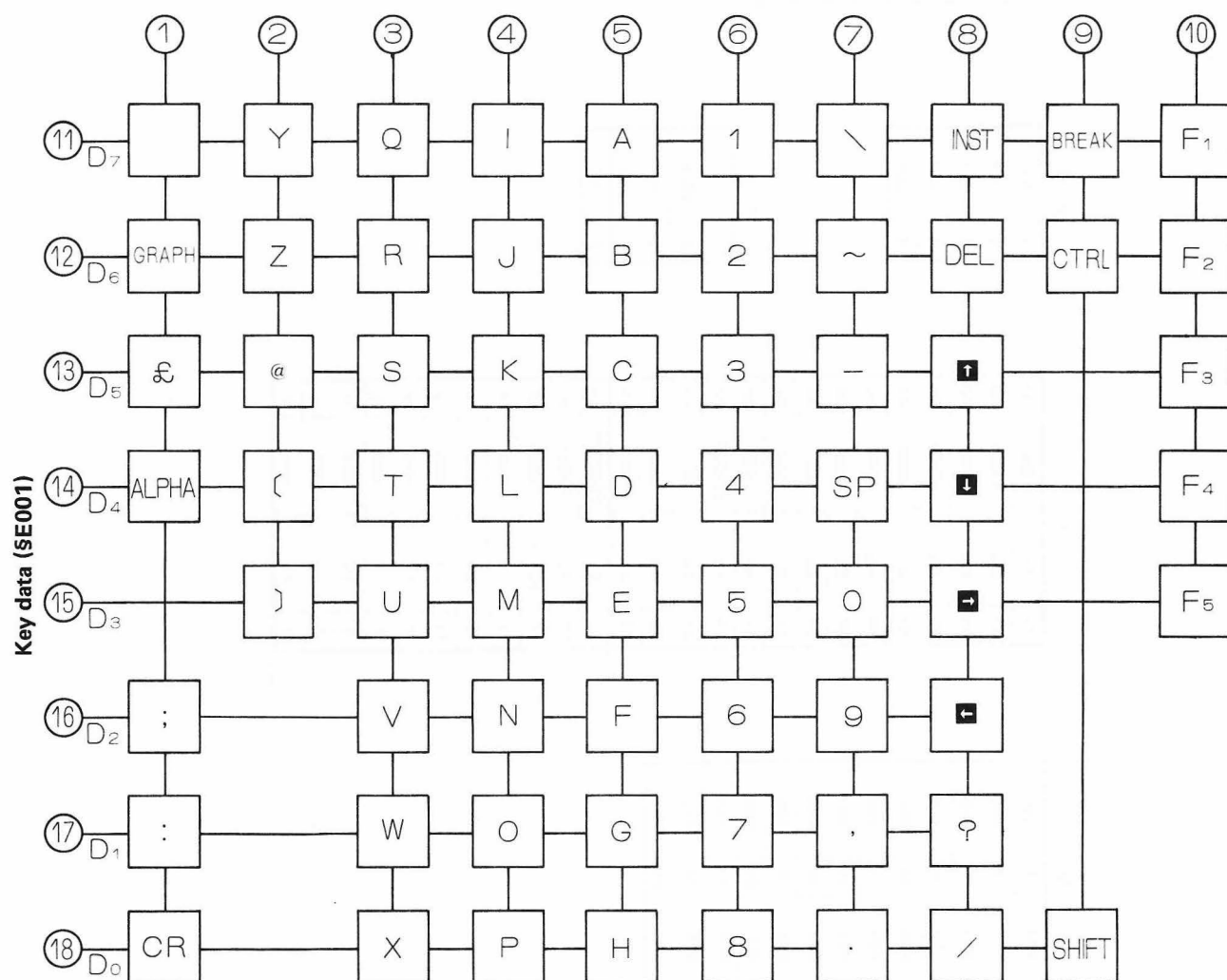
P-9

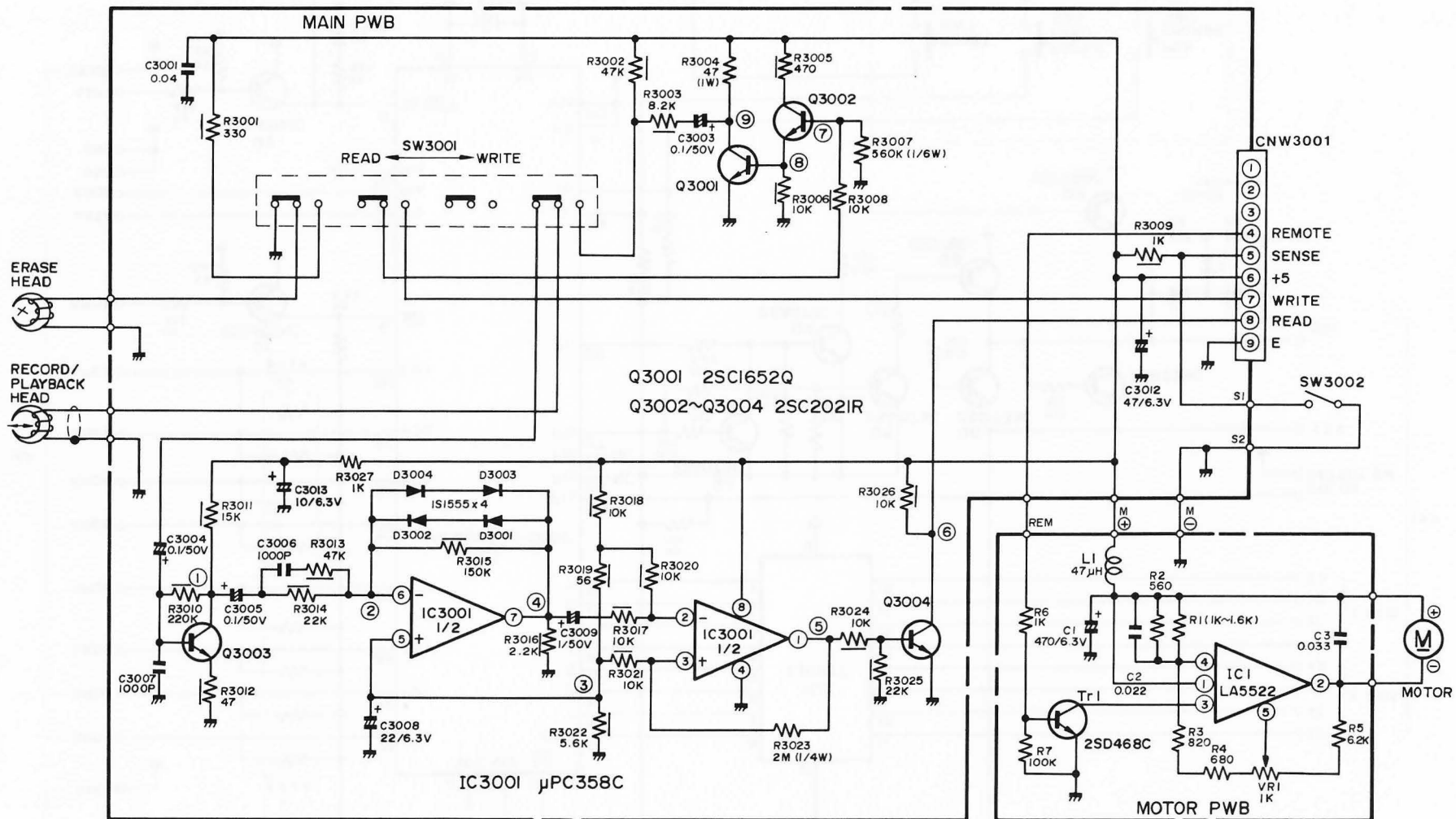
1	○	GND
2	○	C SYNC
3	○	C VIDEO
4	○	H SYNC
5	○	V SYNC
6	○	GND
7	○	+5V
8	○	G
9	○	B
10	○	R
11	○	COLR
12	○	GND



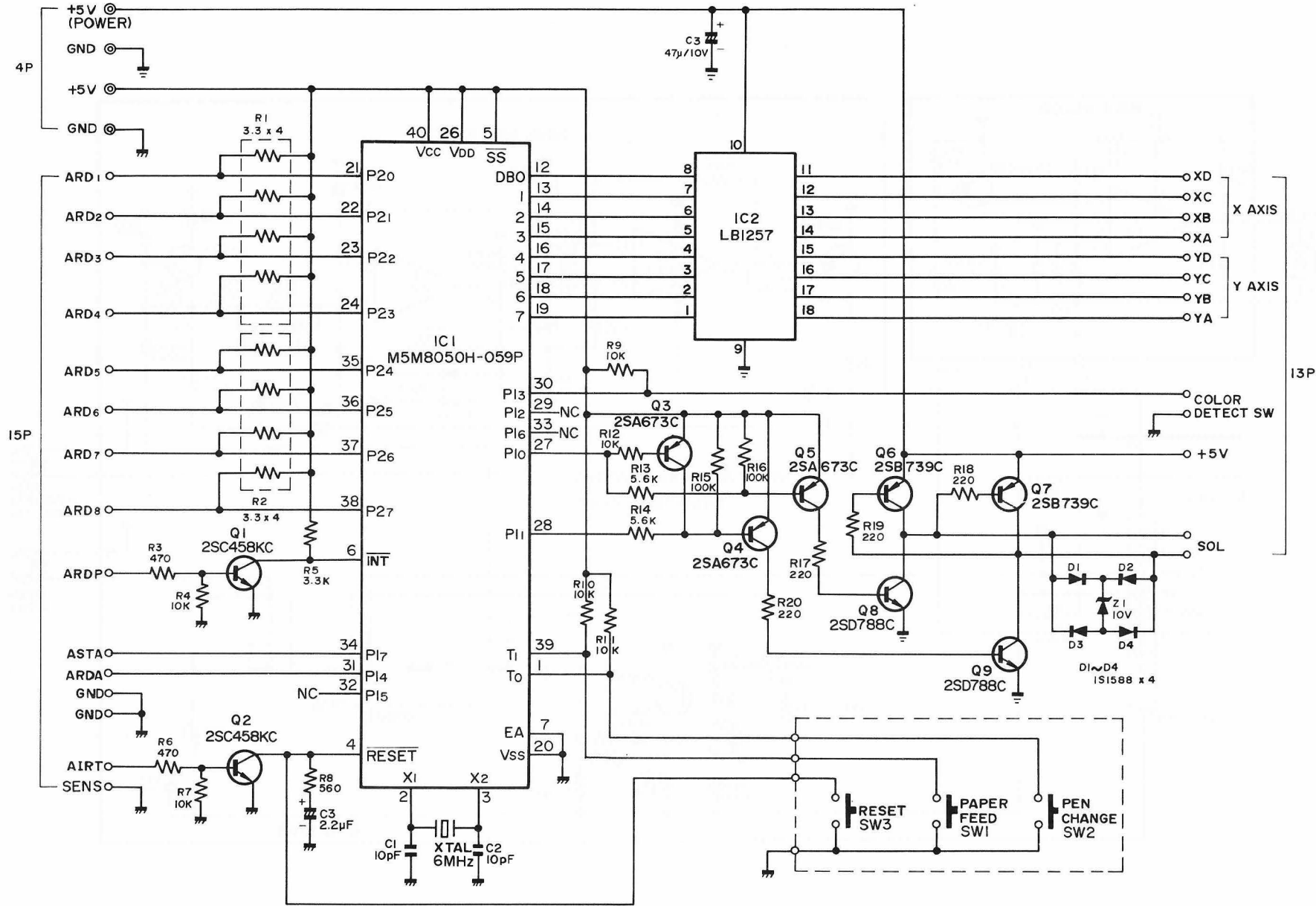
### [Keyboard matrix circuit]

8255 outputs keyboard scan signals from port PA to the keyboard and reads key data from port PB. The figure below shows the key matrix.





[Color plotter-printer circuit]



# Monitor Commands and Subroutines

## Chapter 5



## 5.1 Monitor Commands

The monitor program starts immediately after the power is turned on and awaits input of a monitor command. The monitor commands are listed below. In this chapter, CR indicates that the carriage return key is to be pressed.

- L command . . . . . Loads cassette tape files into memory.
- P command . . . . . Outputs the specified character string to the printer. (Print)
- M command . . . . . Changes the contents of memory. (Memory correction)
- J command . . . . . Transfers control to the specified address. (Jump)
- S command . . . . . Saves the contents of the specified memory block to cassette tape. (Save)
- V command . . . . . Compares the contents of cassette tape with the contents of memory.
- # command . . . . . Transfers control to the RAM area.
- B command . . . . . Makes the bell sound every time a key is pressed. Executing this command again stops the bell.

### ■ Configuration of the monitor work area

The configuration of the monitor work area from \$1000 to \$11FF is shown below.

\$0000	Monitor
\$1000	Stack area
\$10F0	Cassette tape header area
\$1170	Variable area
\$11A3	Key input data area
\$1200	Free area

**Note:** The ROM monitor described in this chapter is not the same as the monitor function of the BASIC interpreter.

---

## 5.2 Functions and Use of Monitor Commands

This section describes the functions and use of the eight monitor commands.

- Commands are executed when the **CR** key is pressed. Characters must be entered in the correct order. If illegal characters (such as spaces) are included in a command string, the monitor rejects the command.
- All numeric data must be entered in hexadecimal form at, and all data is displayed in hexadecimal form at. Therefore, 1-byte data is represented with two hexadecimal digits and 2-byte data is represented with a four hexadecimal digits. For example, the decimal number 21 is displayed as 15 and the decimal number 10 must be typed in as 0A. The upper digit "0" cannot be omitted.
- If the number of characters typed as an operand exceeds the specified number, excess characters are discarded.
- Each command can access any location of memory. Therefore, the monitor program may be changed if the commands are used carelessly. **Since this can result in loss of control over the system, be careful to avoid changing the contents of the monitor program.**

### 5.2.1 L command

Format
--------

L

Function
----------

This command loads the first machine language file encountered on the cassette tape into memory. After the L command is entered, the display changes as follows.

\*L J

⏮ PLAY

Press the **PLAY** key of the data recorder. When a machine language program is found, the message "LOADING program-name" is displayed. For example, the following message is displayed during loading of the BASIC interpreter.

LOADING BASIC

---

## 5.2.2 P command (P : Printer)

Function
----------

This command is used as follows to control the plotter printer:

\*PABC J

Prints the letters "ABC".

\*P&T J

Prints the test pattern.

\*P&S J

Sets the line width (character size) to 80 characters/line.

\*P&L J

Sets the line width (character size) to 40 characters/line.

\*P&G J

Switches the printer to the graphic mode.

\*P&C J

Changes the pen color.

## 5.2.3 M command (M : Memory modification)

Format
--------

M h h h h

h h h h . . . . . starting address

Function
----------

This command is used to change the contents of memory a byte at a time, starting at the specified address.

\*MC000 J

C000 00 FF

C001 00 FF

C002 00 FF

C003 00 FF

C004 00 

SHIFT
-------

 + 

BREAK
-------

\*MC010 J

C010 00 88

C011 00 88

C012 00 88

C013 00 88

C014 00 

SHIFT
-------

 + 

BREAK
-------

\*

To terminate the M command, simultaneously press the 

SHIFT
-------

 and 

BREAK
-------

 keys.



## 5.2.4 J command (J : Jump)

**Format** J h h h h

h h h h . . . . . destination address

**Function**

This command transfers control to the specified address; i.e., it sets the specified address in the program counter.

\*J1200J ..... Jumps to address \$1200.

## 5.2.5 S command (S : Save)

**Format**

S h h h h h' h' h' h' h' h' h' h' h'

h h h h .....starting address

h' h' h' h' .....end address

h" h" h" h" ... execution address

**Function**

Upon execution, this command prompts for entry of a file name, then saves the contents of memory from h h h h to h' h' h' h' on cassette tape under the specified file name. Assume that a machine language program in the area from \$6000 to \$60A3 whose execution address is at \$6050 is to be saved under file name "MFILE"; the command is then entered as follows.

```
*$6000$60A3$6050J
FILENAME? MFILEJ
⏏ RECORD.PLAY
```

Confirm that a blank cassette tape is loaded in the data recorder and press the **RECORD** key.

If the write protect tab of the cassette tape is removed, the **RECORD** key cannot be pressed. Replace it with another cassette.

This command can only be used to save machine language programs.

WRITING MFILE

OK!



**Note:** To abort recording, hold down both the **SHIFT** and **BREAK** keys until the prompt " \* " appears.

---

## 5.2.6 V command (V : Verify)

Format
Function

V

Compares a machine language cassette file saved using the S command with the original program in memory.

\*V J

↓ PLAY

OK

Press the PLAY key to read the cassette tape file when the prompt "↓ PLAY" is displayed. The message "OK" is displayed when the contents of the cassette file matches that of the original program; otherwise, the message "CHECK SUM ER." is displayed.

It is recommended that this command be executed immediately after recording a program with the S command.

## 5.2.7 # command

Format
Function

#

After pressing the RESET switch, executing this command produces the same effect as simultaneously pressing the RESET switch and the CTRL key.

\*# J

## 5.2.8 B command (B : Bell)

Format
Function

B

\*B J

Executing this command once causes the bell to ring each time a key is pressed. Executing it again disables the bell.

## 5.3 Monitor Subroutines

The following subroutines are provided for **Monitor 1Z-013A**. Each subroutine name symbolically represents the function of the corresponding subroutine. These subroutines can be called from user programs.

Registers saved are those whose contents are restored when control is returned to the calling program. The contents of other registers are changed by execution of the subroutine.

Name and entry point (hex.)	Function	Register saved
<b>CALL LETNL</b> (0006)	Moves the cursor to the beginning of the next line.	Other than AF
<b>CALL PRINTS</b> (000C)	Displays a space at the cursor position.	Other than AF
<b>CALL PRINTS</b> (0012)	Displays the character corresponding to the ASCII code stored in ACC at the cursor position. See Appendix A. 1 for the ASCII codes. No character is displayed when code 0D (carriage return) or 11 to 16 (the cursor control codes) is entered, but the corresponding function is performed (a carriage return for 0D and cursor movement for 11 to 16).	Other than AF
<b>CALL MSG</b> (0015)	Displays a message, starting at the position of the cursor. The starting address of the area in which the message is stored must be set in the DE register before calling this subroutine, and the message must end with a carriage return code (0D). The carriage return is not executed. The cursor is moved if any cursor control codes (11 to 16) are included in the message.	All registers
<b>CALL BELL</b> (003E)	Briefly sounds high A (about 880 Hz).	Other AF
<b>CALL MELDY</b> (0030)	Plays music according to music data stored in the memory area starting at the address indicated in the DE register. The music data must be in the same format as that for the MUSIC statement of the BASIC, and must end with 0D or C8. When play is completed, control is returned to the calling program with the C flag set to 0; when play is interrupted with the <b>BREAK</b> key, control is returned with the C flag set to 1.	Other than AF
<b>CALL XTEMP</b> (0041)	Sets the musical tempo according to the tempo data stored in the accumulator (ACC). ACC ← 01    Slowest speed ACC ← 04    Middle speed ACC ← 07    Highest speed Note that the data in the accumulator is not the ASCII code corresponding to 1 to 7 but the binary code.	All registers
<b>CALL MSTA</b> (0044)	Generates a continuous sound of the specified frequency. The frequency is given by the following equation. $\text{freq.} = 895 \text{ kHz}/nn'$ Here, $nn'$ is a 2-byte number stored in addresses 11A1 and 11A2 (n in 11A2 and n' in 11A1).	BC and DE

Name and entry point (hex.)	Function	Register saved
<b>CALL MSTP</b> (0047)	Stops the sound generated with the CALL MSTA subroutine.	Other than AF
<b>CALL TIMST</b> (0033)	Sets and starts the built-in clock. Registers must be set as follows before this routine is called. ACC ← 0 (AM), ACC ← 1 (PM) DE ← 4-digit hexadecimal number representing the time in seconds.	Other than AF
<b>CALL TIMRD</b> (003B)	Reads the built-in clock and returns the time as follows. ACC ← 0 (AM), ACC ← 1 (PM) DE ← 4-digit hexadecimal number representing the time in seconds.	Other than AF and DE
<b>CALL BRKEY</b> (001E)	Checks whether the <b>SHIFT</b> and <b>BREAK</b> keys are both being pressed. The Z flag is set when they are being pressed simultaneously; otherwise, it is reset.	Other than AF
<b>CALL GETL</b> (0003)	Reads one line of data from the keyboard and stores it in the memory area starting at the address indicated in the DE register. This routine stops reading data when the RETURN key is pressed, then appends a carriage return code (0D) to the end of the data read. A maximum of 80 characters (including the carriage return code) can be entered in one line. Characters keyed in are echoed back to the display, and cursor control codes can be included in the line. When the <b>SHIFT</b> and <b>BREAK</b> keys are pressed simultaneously, BREAK code is stored in the address indicated in the DE register and a carriage return code is stored in the subsequent address.	All registers
<b>CALL GETKY</b> (001B)	Reads a character code (ASCII) from the keyboard. If no key is pressed, control is returned to the calling program with 00 set in ACC. No provision is made to avoid data read errors due to key chatter, and characters entered are not echoed back to the display. When any of the special keys (such as <b>DEL</b> or <b>CR</b> ) are pressed, this subroutine returns a code to ACC which is different from the corresponding ASCII code as shown below. Here, display codes are used to address characters stored in the cahracter generator, and are different from the ASCII codes.	Other than AF
<b>Special key read with GETKY</b>	<b>Special key</b>	<b>Code set in ACC</b>
	<b>DEL</b>	60
	<b>INST</b>	61
	<b>ALPHA</b>	62
	<b>BREAK</b>	64
	<b>CR</b>	66
	<b>↓</b>	11
	<b>↑</b>	12
	<b>→</b>	13
	<b>←</b>	14
	<b>HOME</b>	15
	<b>CLR</b>	16
		<b>Display code</b>
		C7
		C8
		C9
		CB
		CD
		C1
		C2
		C3
		C4
		C5
		C6

Name and entry point (hex.)	Function	Register saved																												
CALL ASC (03DA)	Sets the ASCII character corresponding to the hexadecimal number represented by the lower 4 bits of data in ACC.	Other than AF																												
CALL HEX (03F9)	Converts the 8 data bits stored in ACC into a hexadecimal number (assuming that the data is an ASCII character), then sets the hexadecimal number in the lower 4 bits of ACC. The C flag is set to 0 when a hexadecimal number is set in ACC; otherwise, it is set to 1.	Other than AF																												
CALL HLHEX (0410)	Converts a string of 4 ASCII characters into a hexadecimal number and sets it in the HL register. The call and return conditions are as follows. DE ← Starting address of the memory area which contains the ASCII character string (e.g., "3" "1" "A" "5" ) CALL HLHEX CF = 0 HL ← hexadecimal number (e.g., HL = 31A5H) CF = 1 The contents of HL are not assured.	Other than AF and HL																												
CALL 2HEX (041F)	Converts a string of 2 ASCII characters into a hexadecimal number and sets it in ACC. The call and return conditions are as follows. DE ← Starting address of the memory area which contains the ASCII character string. (e.g., "3" "A" ) CALL 2HEX CF = 0 ACC ← hexadecimal number (e.g., ACC = 3AH) CF = 1 The contents of the ACC are not assured.	Other than AF and DE																												
CALL ??KEY (09B3)	Blinks the cursor to prompt for key input. When a key is pressed, the corresponding display code is set in ACC and control is returned to the calling program.	Other than AF																												
CALL ?ADCN (0BB9)	Converts ASCII codes into display codes. The call and return conditions are as follows. ACC ← ASCII code CALL ?ADCN ACC ← Display code	Other than AF																												
CALL ?DACN (0BCE)	Converts display codes into ASCII codes. The call and return conditions are as follows. ACC ← Display code CALL ?DACN ACC ← ASCII code	Other than AF																												
CALL ?BLNK (0DA6)	Detects the vertical blanking period. Control is returned to the calling program when the vertical blanking period is entered.	All registers																												
CALL ?DPCT (0DDC)	Controls display as follows. <table><tr><th>ACC</th><th>Control</th><th>ACC</th><th>Control</th></tr><tr><td>C0H</td><td>Scrolling</td><td>C6H</td><td>Same as the CLR key.</td></tr><tr><td>C1H</td><td>Same as the F1 key.</td><td>C7H</td><td>Same as the DEL key.</td></tr><tr><td>C2H</td><td>Same as the F2 key.</td><td>C8H</td><td>Same as the INST key.</td></tr><tr><td>C3H</td><td>Same as the F3 key.</td><td>C9H</td><td>Same as the ALPHA key.</td></tr><tr><td>C4H</td><td>Same as the F4 key.</td><td>CDH</td><td>Same as the CR key.</td></tr><tr><td>C5H</td><td>Same as the HOME key.</td><td></td><td></td></tr></table>	ACC	Control	ACC	Control	C0H	Scrolling	C6H	Same as the CLR key.	C1H	Same as the F1 key.	C7H	Same as the DEL key.	C2H	Same as the F2 key.	C8H	Same as the INST key.	C3H	Same as the F3 key.	C9H	Same as the ALPHA key.	C4H	Same as the F4 key.	CDH	Same as the CR key.	C5H	Same as the HOME key.			All registers
ACC	Control	ACC	Control																											
C0H	Scrolling	C6H	Same as the CLR key.																											
C1H	Same as the F1 key.	C7H	Same as the DEL key.																											
C2H	Same as the F2 key.	C8H	Same as the INST key.																											
C3H	Same as the F3 key.	C9H	Same as the ALPHA key.																											
C4H	Same as the F4 key.	CDH	Same as the CR key.																											
C5H	Same as the HOME key.																													
CALL ?PONT (0FB1)	Sets the current cursor location in the HL register. The return conditions are as follows. CALL ?PONT HL ← Cursor location (binary)	Other than AF and HL																												



## A. 1 Code Tables

### ■ ASCII code table

MSD is an abbreviation for most significant digit, and represents the upper 4 bits of each code; LSD is an abbreviation for least significant digit, and represents the lower 4 bits of each code. Codes 11<sub>H</sub> to 16<sub>H</sub> are cursor control codes. For example, executing CALL PRNT (a monitor subroutine) with 15<sub>H</sub> set in ACC returns the cursor to the home position. ("H" is not displayed.)

MSD \ LSD		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0	0000			SP	O	@	P	☼	☐	}	—	q	n		⌋	—	□
1	0001		↓	!	I	A	Q	H	☐	☐	☐	a	⌈	☐	⌈	♠	●
2	0010		↑	"	2	B	R	I	☐	☐	e	z	Ü	☐	⌈	⌈	—
3	0011		→	#	3	C	S	♠	☐	☐	`	w	m	☐	⌈	—	♥
4	0100		←	\$	4	D	T	♠	☐	☐	~	s	⌈	☐	⌈	—	⌈
5	0101		H	%	5	E	U	✈	☐	☐	☐	u	⌈	☐	⌈	—	⌈
6	0110		©	&	6	F	V	¥	☐	☐	t	i	⌈	→	☐	—	⊗
7	0111			'	7	G	W	☐	☐	☐	g	≡	o	☐	☐	⌈	○
8	1000			(	8	H	X	☐	☐	☐	h	Ö	l	☐	☐	⌈	♣
9	1001			)	9	I	Y	☐	☐	☐	k	Ä	☐	☐	☐	☐	⌈
A	1010			*	:	J	Z	✈	☐	☐	b	f	ö	☐	☐	☐	♦
B	1011			+	;	K	☐	☐	°	^	x	v	ä	⌈	☐	☐	£
C	1100			,	<	L	☐	☐	☐	☐	d	☐	☐	☐	☐	☐	↓
D	1101	CR		—	=	M	☐	☐	☐	☐	r	ü	y	☐	☐	☐	⌈
E	1110			.	>	N	↑	☐	☐	☐	p	β	☐	☐	☐	☐	☐
F	1111			/	?	O	←	☐	☐	☐	c	j	☐	☐	☐	☐	π

## ■ Display code table

The display codes are used to address character patterns stored in the character generator. These codes must be transferred to video-RAM to display characters.

Monitor subroutines PRNT (0012<sub>H</sub>) and MSG (0015<sub>H</sub>) convert ASCII codes into display codes and transfer them to the V-RAM location indicated for the cursor.

Codes C1<sub>H</sub> to C6<sub>H</sub> are for controlling the cursor.

MSD LSD		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0	0000	SP	P	O	▢	}	↑	π	□		p	▤	▥	↓	▦	▧	SP
1	0001	A	Q	I	▢	♠	<	!	□	a	q	▤	▥	↓	▦	▧	▨
2	0010	B	R	2	▢	▤	▥	"	□	b	r	▤	▥	↑	▦	▧	▨
3	0011	C	S	3	▢	■	♥	#	□	c	s	▤	▥	→	▦	▧	▨
4	0100	D	T	4	▢	♦	▥	\$	▢	d	t	▤	▥	←	▦	▧	▨
5	0101	E	U	5	▢	←	@	%	▢	e	u	~	▤	H	▦	▧	▨
6	0110	F	V	6	▢	♣	▤	&	▤	f	v	▤	▥	☾	▦	▧	▨
7	0111	G	W	7	▢	●	>	'	▤	g	w	▤	▥	☼	▦	▧	▨
8	1000	H	X	8	▢	○	↓	(	▢	h	x	▤	▥	H	▦	▧	▨
9	1001	I	Y	9	▢	?	▤	)	▢	i	y	▤	▥	I	▦	▧	▨
A	1010	J	Z	—	▢	◐	→	+	▢	j	z	β	▤	人	▦	▧	▨
B	1011	K	£	=	▢	▤	▥	*	▢	k	ä	ü	▤	✱	°	Y	▨
C	1100	L	▢	;	▢	▤	▥	▢	▢	l	▤	ö	{	✱	▦	▧	▨
D	1101	M	▢	▤	▢	▤	▥	✕	▢	m	▤	Ü	▤	✱	▦	▧	▨
E	1110	N	H	.	▢	▤	▥	▢	▢	n	▤	Ä	^	☉	▦	▧	▨
F	1111	O	H	,	▢	:	▥	▢	▢	o	▤	Ö	▢	☺	▦	▧	▨



## ■ ASCII code table for color plotter-printer

Graphic characters other than those shown above cannot be printed, but the corresponding hexadecimal code is printed in a different pen color.

MSD LSD	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0			SP	Ø	@	P			}		q	n				
1		↓	∇	1	A	Q					a					
2		↑	"	2	B	R				e	Z	ü				
3		→	#	3	C	S				`	w	m				
4		←	\$	4	D	T				~	s					
5		H	%	5	E	U					u					
6		©	&	6	F	V				t	i		→			
7			/	7	G	W				g		o		=		
8			[	8	H	X				h	ö	l				
9			]	9	I	Y					k	ä				
A			*	:	J	z				b	f	ö				
B			+	;	K	[		°	^	x	v	ä				£
C			,	<	L	\				d						↓
D			-	=	M	]				^	ü	y				
E			·	>	N	↑				p	ß	{				
F			/	?	O	←				c	j		=			π

---

## A. 2 MZ-700 Series Computer Specifications

### A.2.1 MZ-700

CPU:	SHARP LH0080A (Z80A)
Clock:	3.5 MHz
Memory:	ROM 4K bytes (ROM) 2K bytes (character generator) RAM 64K bytes (program area) 4K bytes (video RAM)
Video output:	(Except MZ-710) PAL system RGB signal Composite signal (B/W) RF signal (UHF $36 \pm 3$ CH, B/W) MZ-710 is available only for connecting with RGB signal.
Screen size:	40 characters x 25 lines 8 x 8 dot character matrix
Colors:	8 colors for characters 8 colors for background
Music function:	Built in (500 mW max. output)
Clock:	Built in (24 hour clock, no backup)
Keys:	69 keys ASCII standard Definable function keys, cursor control keys
Editing function:	Screen editor (cursor control, home, clear, insert, and delete)
Temperature:	Operating; 0 ~ 35°C Storage; -20 ~ 60°C
Humidity:	Operating; 85% or less Storage; 85% or less
Dimensions:	MZ-731; 400 (W) x 305 (D) x 102 (H) mm MZ-721; 440 (W) x 305 (D) x 86 (H) mm MZ-711; 440 (W) x 305 (D) x 86 (H) mm
Weight:	MZ-731; 4.6 kg MZ-721; 4.0 kg MZ-711; 3.6 kg
Accessories:	Cassette tape (BASIC (side A) Application programs (side B)) Owners manual, function labels, power cable, TV connection cable (except MZ-710) Attachments for the color plotter-printer are listed later.

---

## A.2.2 CPU board specifications

CPU:	LH0080A (Z80A)	.....	1	
PPI:	8255	.....	1	
PIT:	8253	.....	1	
Memory controller				
(CRTC):	M60719	.....	1	
ROM:	Monitor	4K byte ROM	.....	1
	Character generator	2K byte ROM	.....	1
RAM:	64K bits	D-RAM	.....	8
	2K byte	S-RAM	.....	2
I/O bus:	Expansion I/O bus	.....	1	
	Printer I/O bus	.....	2 (Cannot be used at the same time)	
	Cassette READ/WRITE terminals	.....	2	
	Joystick terminal	.....	2	

## A.2.3 Color plotter-printer specifications

Printing system:	4 selectable colors using ball point pens
Colors:	1. Black, 2. Blue, 3. Green, 4. Red
Printing speed:	Average 10 characters/second when printing with the smallest size characters.
Line width:	80 columns, 40 columns, or 26 columns (selected by software)
Number of characters:	115 (including ASCII characters)
Resolution:	0.2 mm
Accessories:	Roll paper (1), Ball pens (black, blue, green red) Paper holders (left and right) Roll shaft (1), Paper guide (1)

## A.2.4 Data recorder specifications

Type:	IEC standard compact cassette mechanism
Recording/ playback system:	2 track, 1 channel monophonic
Rated speed:	4.8 cm/s $\pm 3.5\%$
Type of control switches:	Piano type
Control switches:	PLAY, FF, REW, STOP/EJECT, and REC keys and counter reset button
Data transfer method:	Sharp PWM method
Data transfer rate:	1200 bps (typ.)
Tape:	Ordinary audio cassette tape

## A.2.5 Power supply specifications

(Supplies power to the color plotter-printer and data recorder, as well as to the main unit.)	
Input:	240/220 V $\pm 10\%$ , 50/60 Hz, 20 W
Output:	5 V

## A.3 BASIC Error Message List

The BASIC interpreter displays an error message in one of the following formats when an error occurs during operation.

1. <error type> error (Direct mode error)
2. <error type> error in line number (Run mode error)

Error messages in format 1 are issued when an error is detected during execution of a direct command or entry of a program. Error messages in format 2 are issued when an error is detected during program execution.

Error messages which may be displayed are shown below.

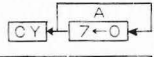
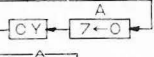
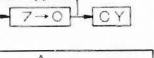
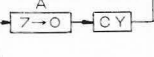
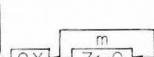
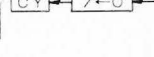
### SYNTAX

Error No.	Message displayed	Description
1	SYNTAX ERROR	Syntax error
2	OVER FLOW ERROR	Numeric data used is out of the specified range, or an overflow occurred.
3	ILLEGAL DATA ERROR	Illegal constant or variable was used.
5	STRING LENGTH ERROR	String length exceeded 255 characters.
6	MEMORY CAPACITY ERROR	Memory capacity is insufficient.
7	ARRAY DEF. ERROR	An attempt was made to redefine an array to a size greater than that defined previously.
8	LINELENGTH ERROR	The length of a line was too long.
10	GOSUB NESTING ERROR	The number of levels of GOSUB nesting exceeded the limit determined by the usable memory space.
11	FOR~NEXT ERROR	The number of levels of FOR~NEXT loops exceeded the limit determined by the usable memory area.
12	DEF FN NESTING ERROR	The number of levels of DEF FN nesting exceeded the limit.
13	NEXT ERROR	NEXT was used without a corresponding FOR.
14	RETURN ERROR	RETURN was used without a corresponding GOSUB.
15	UN DEF. FUNCTION ERROR	An undefined function was called.
16	UN DEF. LINE NUM. ERROR	An unused line number was referenced.
17	CAN'T CONTINUE	CONT command cannot be executed.
18	MEMORY PROTECTION	An attempt was made to write data to the BASIC control area.
19	INSTRUCTION ERROR	Direct mode commands and statements are mixed together.
20	CAN'T RESUME ERROR	RESUME cannot be executed.
21	RESUME ERROR	An attempt was made to execute RESUME when no error had occurred.
24	READ ERROR	READ was used without a corresponding DATA statement.
43	ALREADY OPEN ERROR	An OPEN statement was issued to a file which was already open.
63	OUT OF FILE ERROR	Out of file during file read.
65	PRINTER IS NOT READY	Printer is not connected.
68	PRINTER MODE ERROR	Color plotter-printer mode error.
70	CHECK SUM ERROR	Check sum error (during tape read).

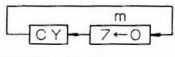
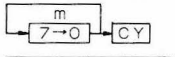
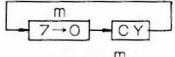
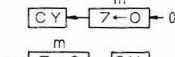
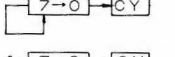
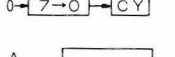
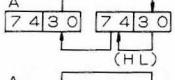
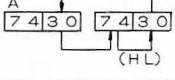
## A. 4 Z80A Instruction Set

A summary of the Z80A instructions are given below for reference.

Mnemonic	Symbolic operation	Op-code	Mnemonic	Symbolic operation	Op-code
<b>8-bit load group</b>					
LD r, r'	$r \leftarrow r'$	01 r r'	LD HL, (nn)	$H \leftarrow (nn+1)$ $L \leftarrow (nn)$	00 101 010 $\leftarrow n \rightarrow$
LD r, n	$r \leftarrow n$	00 r 110 $\leftarrow n \rightarrow$	LD dd, (nn)	$dd_H \leftarrow (nn+1)$ $dd_L \leftarrow (nn)$	$\leftarrow n \rightarrow$ 11 101 101 01 dd1 011 $\leftarrow n \rightarrow$
LD r, (HL)	$r \leftarrow (HL)$	01 r 110	LD IX, (nn)	$IX_H \leftarrow (nn+1)$ $IX_L \leftarrow (nn)$	$\leftarrow n \rightarrow$ 11 011 101 00 011 010 $\leftarrow n \rightarrow$
LD r, (IX+d)	$r \leftarrow (IX+d)$	01 r 110 $\leftarrow d \rightarrow$	LD IY, (nn)	$IY_H \leftarrow (nn+1)$ $IY_L \leftarrow (nn)$	$\leftarrow n \rightarrow$ 11 111 101 00 101 010 $\leftarrow n \rightarrow$
LD r, (IY+d)	$r \leftarrow (IY+d)$	11 111 101 01 r 110 $\leftarrow d \rightarrow$	LD (nn), HL	$(nn+1) \leftarrow H$ $(nn) \leftarrow L$	$\leftarrow n \rightarrow$ 00 100 010 $\leftarrow n \rightarrow$
LD (HL), r	$(HL) \leftarrow r$	01 110 r	LD (nn), dd	$(nn+1) \leftarrow dd_H$ $(nn) \leftarrow dd_L$	$\leftarrow n \rightarrow$ 11 101 101 01 dd0 011 $\leftarrow n \rightarrow$
LD (IX+d), r	$(IX+d) \leftarrow r$	11 011 101 01 110 r $\leftarrow d \rightarrow$	LD (nn), IX	$(nn+1) \leftarrow IX_H$ $(nn) \leftarrow IX_L$	$\leftarrow n \rightarrow$ 11 011 101 00 100 010 $\leftarrow n \rightarrow$
LD (IY+d), r	$(IY+d) \leftarrow r$	11 111 101 01 110 r $\leftarrow d \rightarrow$	LD (nn), IY	$(nn+1) \leftarrow IY_H$ $(nn) \leftarrow IY_L$	$\leftarrow n \rightarrow$ 11 111 101 00 100 010 $\leftarrow n \rightarrow$
LD (HL), n	$(HL) \leftarrow n$	00 110 110 $\leftarrow n \rightarrow$	LD SP, HL	$SP \leftarrow HL$	11 111 001
LD (IX+d), n	$(IX+d) \leftarrow n$	11 011 101 00 110 110 $\leftarrow d \rightarrow$	LD SP, IX	$SP \leftarrow IX$	11 011 101 11 111 001
LD (IY+d), n	$(IY+d) \leftarrow n$	11 111 101 00 110 110 $\leftarrow d \rightarrow$	LD SP, IY	$SP \leftarrow IY$	11 111 101 11 111 001
LD A, (BC)	$A \leftarrow (BC)$	00 001 010	PUSH qq	$(SP-2) \leftarrow qq_L$	11 qq0 101
LD A, (DE)	$A \leftarrow (DE)$	00 011 010	PUSH IX	$(SP-1) \leftarrow qq_H$ $(SP-2) \leftarrow IX_L$	11 011 101 11 100 101
LD A, (nn)	$A \leftarrow (nn)$	00 111 010 $\leftarrow n \rightarrow$	PUSH IY	$(SP-1) \leftarrow IX_H$ $(SP-2) \leftarrow IY_L$	11 111 101 11 100 101
LD (BC), A	$(BC) \leftarrow A$	00 000 010	POP qq	$(SP-1) \leftarrow IY_H$ $qq_H \leftarrow (SP+1)$	11 qq0 001
LD (DE), A	$(DE) \leftarrow A$	00 010 010	POP IX	$qq_L \leftarrow (SP)$ $IX_H \leftarrow (SP+1)$	11 011 101 11 100 001
LD (nn), A	$(nn) \leftarrow A$	00 110 010 $\leftarrow n \rightarrow$	POP IY	$IX_L \leftarrow (SP)$ $IY_H \leftarrow (SP+1)$	11 111 101 11 100 001
LD A, I	$A \leftarrow I$	11 101 101		$IY_L \leftarrow (SP)$	
LD A, R	$A \leftarrow R$	01 010 111			
LD I, A	$I \leftarrow A$	11 101 101			
LD R, A	$R \leftarrow A$	01 000 111			
		01 001 111			
<b>16-bit load group</b>			<b>Exchange group and block transfer and search group</b>		
LD dd, nn	$dd \leftarrow nn$	00 dd0 001 $\leftarrow n \rightarrow$	EX DE, HL	$DE \leftrightarrow HL$	11 101 011
LD IX, nn	$IX \leftarrow nn$	11 011 101 00 100 001 $\leftarrow n \rightarrow$	EX AF, AF'	$AF \leftrightarrow AF'$	00 001 000
LD IY, nn	$IY \leftarrow nn$	11 111 101 00 100 001 $\leftarrow n \rightarrow$	EXX	$(BC) \leftrightarrow (BC')$ $(DE) \leftrightarrow (DE')$ $(HL) \leftrightarrow (HL')$	11 011 001
		$\leftarrow n \rightarrow$	EX (SP), HL	$H \leftrightarrow (SP+1)$ $L \leftrightarrow (SP)$	11 100 011
		$\leftarrow n \rightarrow$	EX (SP), IX	$IX_H \leftrightarrow (SP+1)$ $IX_L \leftrightarrow (SP)$	11 011 101 11 100 011
		$\leftarrow n \rightarrow$	EX (SP), IY	$IY_H \leftrightarrow (SP+1)$ $IY_L \leftrightarrow (SP)$	11 111 101 11 100 011

Mnemonic	Symbolic operation	Op-code	Mnemonic	Symbolic operation	Op-code
LDI	$(DE) \leftarrow (HL)$ $DE \leftarrow DE + 1$ $HL \leftarrow HL + 1$ $BC \leftarrow BC - 1$	11 101 101 10 100 000	DEC m	$m \leftarrow m - 1$	$\leftarrow d \rightarrow$ 101
LDIR	$(DE) \leftarrow (HL)$ $DE \leftarrow DE + 1$ $HL \leftarrow HL + 1$ $BC \leftarrow BC - 1$ Repeat until $BC = 0$	11 101 101 10 110 000	General purpose arithmetic and control group		
LDD	$(DE) \leftarrow (HL)$ $DE \leftarrow DE - 1$ $HL \leftarrow HL - 1$ $BC \leftarrow BC - 1$	11 101 101 10 101 000	DAA	Decimal adjustment upon contents of A after add or subtract	00 100 111
LDDR	$(DE) \leftarrow (HL)$ $DE \leftarrow DE - 1$ $HL \leftarrow HL - 1$ $BC \leftarrow BC - 1$ Repeat until $BC = 0$	11 101 101 10 111 000	CPL	$A \leftarrow \overline{A}$	00 101 111
CPI	$A - (HL)$ $HL \leftarrow HL + 1$ $BC \leftarrow BC - 1$	11 101 101 10 100 001	NEG	$A \leftarrow \overline{A} + 1$	11 101 101 01 000 100
CPIR	$A - (HL)$ $HL \leftarrow HL + 1$ $BC \leftarrow BC - 1$ Repeat until $A = (HL)$ or $BC = 0$	11 101 101 10 110 001	CCF	$CY \leftarrow \overline{CY}$	00 111 111
CPD	$A - (HL)$ $HL \leftarrow HL - 1$ $BC \leftarrow BC - 1$	11 101 101 10 101 001	SCF	$CY \leftarrow 1$	00 110 111
CPDR	$A - (HL)$ $HL \leftarrow HL - 1$ $BC \leftarrow BC - 1$ Repeat until $A = (HL)$ or $BC = 0$	11 101 001 10 111 001	NOP	No operation, but PC is incremented.	00 000 000
8-bit arithmetic and logical group			HALT	CPU halted	01 110 110
ADD A, r	$A \leftarrow A + r$	10 <u>000</u> r	DI	$IFF \leftarrow 0$	11 110 011
AD A, n	$A \leftarrow A + n$	11 <u>000</u> 110 $\leftarrow n \rightarrow$	EI	$IFF \leftarrow 1$	11 111 011
ADD A, (HL)	$A \leftarrow A + (HL)$	10 <u>000</u> 110	IM0	Set interrupt mode 0	11 001 101
ADD A, (IX+d)	$A \leftarrow A, (IX+d)$	11 011 101 10 <u>000</u> 110 $\leftarrow d \rightarrow$	IM1	Set interrupt mode 1	01 010 110
ADD A, (IY+d)	$A \leftarrow A + (IY+d)$	11 111 101 10 <u>000</u> 110 $\leftarrow d \rightarrow$	IM2	Set interrupt mode 2	01 011 110
ADC A, s	$A \leftarrow A + s + CY$	<u>001</u>	16-bit arithmetic group		
SUB s	$A \leftarrow A - s$	<u>010</u>	ADD HL, ss	$HL \leftarrow HL + ss$	00 ss1 001
SBC A, s	$A \leftarrow A - s - CY$	<u>011</u>	ADC HL, ss	$HL \leftarrow HL + ss + CY$	11 101 101 01 ss1 010
AND s	$A \leftarrow A \wedge s$	<u>100</u>	SBC HL, ss	$HL \leftarrow HL - ss - CY$	11 101 101 01 ss0 010
OR s	$A \leftarrow A \vee s$	<u>110</u>	ADD IX, pp	$IX \leftarrow IX + pp$	11 011 101 00 pp1 001
XOR s	$A \leftarrow A \oplus s$	<u>101</u>	ADD IY, rr	$IY \leftarrow IY + rr$	11 111 101 00 rr1 001
CP s	$A - s$	<u>111</u>	INC ss	$ss \leftarrow ss + 1$	00 ss0 011
INC r	$r \leftarrow r + 1$	00 r <u>100</u>	INC IX	$IX \leftarrow IX + 1$	11 011 101 00 100 011
INC (HL)	$(HL) \leftarrow (HL) + 1$	00 110 <u>100</u>	INC IY	$IY \leftarrow IY + 1$	11 111 101 00 100 011
INC (IX+d)	$(IX+d) \leftarrow (IX+d) + 1$	11 011 101 00 110 <u>100</u> $\leftarrow d \rightarrow$	DEC ss	$ss \leftarrow ss - 1$	00 ss1 011
INC (IY+d)	$(IY+d) \leftarrow (IY+d) + 1$	11 111 101 00 110 <u>100</u> $\leftarrow d \rightarrow$	DEC IX	$IX \leftarrow IX - 1$	11 011 101 00 101 011
			DEC IY	$IY \leftarrow IY - 1$	11 111 101 00 101 011
			Rotate and shift group		
			RLCA		00 000 111
			RLA		00 010 111
			RRCA		00 001 111
			RRA		00 011 111
			RLC r		11 001 011 00 <u>000</u> r
			RLC (HL)		11 001 011 00 <u>000</u> 110



Mnemonic	Symbolic operation	Op-code	Mnemonic	Symbolic operation	Op-code
RLC (IX+d)		11 011 101 11 001 011 ← d → 00 000 110	Jump group		
RLC (IY+d)		11 111 011 11 001 011 ← d → 00 000 110	JP nn	PC←nn	11 000 011 ← n → ← n →
RL m		010	JP cc, nn	If condition cc is true, PC←-nn; otherwise, continue	11 cc 010 ← n → ← n →
RRC m		001	JR e	PC←PC+e	00 011 000 ← e-2 →
RR m		011	JR C, e	If C=0, continue. If C=1, PC←PC+e	00 111 000 ← e-2 →
SLA m		100	JR Z, e	If Z=0, continue. If C=1, PC←PC+e	00 101 000 ← e-2 →
SRA m		101	JR NC, e	If C=1, continue. If C=0, PC←PC+e	00 110 000 ← e-2 →
SRL m		111	JR NZ, e	If Z=1, continue. If Z=0, PC←PC+e	00 100 000 ← e-2 →
RLD		11 101 101 01 101 111	JP (HL)	PC←HL	11 101 001
RRD		11 101 101 01 100 111	JP (IX)	PC←IX	11 011 101 11 101 001 11 111 101 11 101 001
Bit set, reset and test group			JP (IY)	PC←IY	11 101 001 00 010 000 ← e-2 →
BIT b, r	$Z \leftarrow \overline{r}b$	11 001 011 01 b r 11 011 011 01 b 110	Call and return group		
BIT b, (HL)	$Z \leftarrow \overline{(HL)}b$	11 011 101 11 001 011 ← d → 01 b 110	CALL nn	(SP-1)←PC <sub>H</sub> (SP-2)←PC <sub>L</sub> PC←nn	11 001 101 ← n → ← n →
BIT b, (IX+d)	$Z \leftarrow \overline{(IX+d)}b$	11 011 101 11 001 011 ← d → 01 b 110	CALL cc, nn	If condition cc is false, continue; otherwise same as CALL nn.	11 cc 100 ← n → ← n →
BIT b, (IY+d)	$Z \leftarrow \overline{(IY+d)}b$	11 111 101 11 001 011 ← d → 01 b 110	RET	PC <sub>L</sub> ←(SP) PC <sub>H</sub> ←(SP+1)	11 001 001
SET b, r	rb←1	11 001 011 11 b r 11 001 011 11 b 110	RET cc	If condition cc is false, continue; otherwise same as RET.	11 cc 000
SET b, (HL)	(HL)b←1	11 001 011 11 001 101 11 001 011 ← d → 11 b 110	RETI	Return from interrupt	11 101 101 01 001 101
SET b, (IX+d)	(IX+d)b←1	11 001 011 11 001 011 ← d → 11 b 110	RETN	Return from NMI.	11 101 101 01 000 101 11 t 111
SET b, (IY+d)	(IY+d)b←1	11 111 101 11 001 011 ← d → 11 b 110	RST p	(SP-1)←PC <sub>H</sub> (SP-2)←PC <sub>L</sub> PC <sub>H</sub> ←0 PC <sub>L</sub> ←p	
RES b, m	mb←0	11 001 011 11 b 110 10			



Mnemonic	Symbolic operation	Op-code	Mnemonic	Symbolic operation	Op-code
Input and output group					
IN A, (n)	$A \leftarrow (n)$	11 011 011 $\leftarrow n \rightarrow$	OUT (n), A	$(n) \leftarrow A$	11 010 011 $\leftarrow n \rightarrow$
IN r, (C)	$r \leftarrow (C)$	11 101 101 01 r 000	OUT (C), r	$(C) \leftarrow r$	11 101 101 01 r 001
INI	$(HL) \leftarrow (C)$ $B \leftarrow B - 1$ $HL \leftarrow HL + 1$	11 101 101 10 100 010	OUTI	$(C) \leftarrow (HL)$ $B \leftarrow B - 1$ $HL \leftarrow HL + 1$	11 101 101 10 100 011
INIR	$(HL) \leftarrow (C)$ $B \leftarrow B - 1$ $HL \leftarrow HL + 1$ Repeat until B=0	11 101 101 10 110 010	OTIR	$(C) \leftarrow (HL)$ $B \leftarrow B - 1$ $HL \leftarrow HL + 1$ Repeat until B=0	11 101 101 10 110 011
IND	$(HL) \leftarrow (C)$ $B \leftarrow B - 1$ $HL \leftarrow HL - 1$	11 101 101 10 101 010	OUTD	$(C) \leftarrow (HL)$ $B \leftarrow B - 1$ $HL \leftarrow HL - 1$	11 101 101 10 101 011
INDR	$(HL) \leftarrow (C)$ $B \leftarrow B - 1$ $HL \leftarrow HL - 1$ Repeat until B=0	11 101 101 10 111 010	OTDR	$(C) \leftarrow (HL)$ $B \leftarrow B - 1$ $HL \leftarrow HL - 1$ Repeat until B=0	11 101 101 10 111 011

(Note) The meanings of symbols used in the above table are as follows.

r, r'	Register	dd, ss	Register pair	qq	Register pair	pp	Register pair
000	B	00	B C	00	B C	00	B C
001	C	01	D E	01	D E	01	D E
010	D	10	H L	10	H L	10	I X
011	E	11	S P	11	A F	11	S P
100	H						
101	L						
111	A						

rr	Register pair	b	Bit set	cc	Condition	t	p
00	B C	000	0	000	N Z non zero	000	00H
01	D E	001	1	001	Z zero	001	08H
10	I Y	010	2	010	N C non carry	010	10H
11	S P	011	3	011	C carry	011	18H
		100	4	100	P O parity odd	100	20H
$\wedge$ : AND operation		101	5	101	P E parity even	101	28H
$\vee$ : OR operation		110	6	110	P sign positive	110	30H
$\oplus$ : Exclusive OR operation		111	7	111	M sign negative	111	38H

s: r, n, (HL), (IX + d), (IY + d)

CY: Carry flip-flop

(register pair)H: Upper 8 bits of register pair

m : r, (HL), (IX + d), (IY + d)

m<sub>b</sub> : Bit b or location m

(register pair)L: Lower 8 bits of register pair

For op-codes ADC, SUB, SBC, AND, OR, XOR and CP, the bits in ☐ replace ☐ in the ADD set.

For op-code DEC, ☐ replaces ☐ in the INC set.

Similar operations apply to op-codes of the rotate and shift group and bit set, reset and test group.

## A. 5 Monitor Program Assembly List

An assembly listing of the MONITOR 1Z-013A is provided on the following pages.

This assembly list was produced with the Z80 assembler contained in the floppy DOS. The meanings of symbols in the list are as follows.

Relative address	Relocatable object code	Assembler message	Label	Mnemonic (op-code)	Operand	Comment
20	02A7 13			INC	DE	
21	02A8 13			INC	DE	
22	02A9 13			INC	DE	
23	02AA C9			RET		
24	02AB					
25	02AB					
26	02AB					
27	02AB					
28	02AB					
29	02AB					
30	02AB					
31	02AB					
32	02AB 2AA111					
33	02AE 7C					
34	02AF B7					
35	02B0 280C					
36	02B2 D5					
37	02B3 EB					
38	02B4 2104E0					
39	02B7 73					
40	02B8 72					
41	02B9 3E01					
42	02BB D1					
43	02BC 1806					
44	02BE					
45	02BE					
46	02BE 3E36					
47	02C0 3207E0					
48	02C3 AF					
49	02C4 3208E0					
50	02C7 C9					

Since the starting address of Monitor 1Z-013A is set to \$0000, relocatable addresses and object codes in the assembly list can be assumed as absolute addresses and object code, respectively.

This assembly list is provided for reference, only and the Sharp Corporation can assume no responsibility for answering any question about it.

Note that this monitor differs from the monitor program included in the BASIC interpreter.

```

01 0000      ;
02 0000      ;
03 0000      ;   MONITOR PROGRAM 1Z-013A
04 0000      ;
05 0000      ;       (MZ-70Q) FOR PAL
06 0000      ;
07 0000      ;       REV. 83.4.7
08 0000      ;
09 0000      ;
10 0000      ;   MONIT:  ENT
11 0000 C34A00      ;   JP      START      ; MONITOR ON
12 0003      ;   GETL:  ENT
13 0003 C3E607      ;   JP      ?GETL      ; GET LINE (END 'CR')
14 0006      ;   LETNL: ENT
15 0006 C30E09      ;   JP      ?LTNL      ; NEW LINE
16 0009      ;   NL:    ENT
17 0009 C31809      ;   JP      ?NL        ;
18 000C      ;   PRNTS: ENT
19 000C C32009      ;   JP      ?PRTS     ; PRINT SPACE
20 000F      ;   PRNTT: ENT
21 000F C32409      ;   JP      ?PRTT     ; PRINT TAB
22 0012      ;   PRNT:  ENT
23 0012 C33509      ;   JP      ?PRNT     ; 1 CHARACTER PRINT
24 0015      ;   MSG:   ENT
25 0015 C39308      ;   JP      ?MSG      ; 1 LINE PRINT (END 'ODH')
26 0018      ;
27 0018 C3A108      ;   MSGX: ENT
28 001B      ;   JP      ?MSGX     ; RST 3
29 001B C3BD08      ;   GETKY: ENT
30 001E      ;   JP      ?GET      ; GET KEY
31 001E C3320A      ;   BRKEY: ENT
32 0021      ;   JP      ?BRK      ; GET BREAK
33 0021 C33604      ;   WRINF: ENT
34 0024      ;   JP      ?WRI      ; WRITE INFORMATION
35 0024 C37504      ;   WRDAT: ENT
36 0027      ;   JP      ?WRD      ; WRITE DATA
37 0027 C3D804      ;   RDINF: ENT
38 002A      ;   JP      ?RDI      ; READ INFORMATION
39 002A C3F804      ;   RDDAT: ENT
40 002D      ;   JP      ?RDD      ; READ DATA
41 002D C38805      ;   VERFY: ENT
42 0030      ;   JP      ?VRFY     ; VERIFYING CMT
43 0030 C3C701      ;   MELDY: ENT
44 0033      ;   JP      ?MLDY     ; RST 6
45 0033 C30803      ;   TIMST: ENT
46 0036 00          ;   JP      ?TMST     ; TIME SET
47 0037 00          ;   NOP
48 0038 C33810      ;   JP      103BH      ; INTERRUPT ROUTINE
49 003B      ;   TIMRD: ENT
50 003B C35803      ;   JP      ?TMRD     ; TIME READ
51 003E      ;   BELL:  ENT
52 003E C37705      ;   JP      ?BEL      ; BELL ON
53 0041      ;   XTEMP: ENT
54 0041 C3E502      ;   JP      ?TEMP     ; TEMPO SET (1+7)
55 0044      ;   MSTA:  ENT
56 0044 C3AB02      ;   JP      MLDST     ; MELODY START
57 0047      ;   MSTP:  ENT
58 0047 C3BE02      ;   JP      MLDSP     ; MELODY STOP
59 004A      ;
60 004A      ;

```

```

01 004A      ;
02 004A      ;   START: ENT
03 004A 31F010      ;   LD      SP,SP      ; STACK SET (10FOH)
04 004D ED56        ;   IM      1          ; IM 1 SET
05 004F CD3E07      ;   CALL   ?MODE      ; 8255,8253 MODE SET
06 0052 CD320A      ;   CALL   ?BRK       ; CTRL ?
07 0055 3019        ;   JR      NC,ST0     ;
08 0057 FE20        ;   CP      20H        ; KEY IS CTRL KEY
09 0059 2015        ;   JR      NZ,ST0     ;
10 005B      ;   CMY0:  ENT
11 005B D3E1        ;   OUT     (E1H),A     ; D000H-FFFFH IS DRAM
12 005D 11F0FF      ;   LD      DE,FFFF0H  ; TRANS. ADR.
13 0060 216B00      ;   LD      HL,$MCP     ; MEMORY CHANG PROGRAM
14 0063 010500      ;   LD      BC,05      ; BYTE SIZE
15 0066 EDB0        ;   LDIR    JP         ;
16 0068 C3F0FF      ;   JP      FFF0H      ; JUMP $FFFF
17 006B      ;
18 006B      ;   $MCP:  ENT
19 006B D3E0        ;   DEFW    E0D3H      ; 0000H-OFFFH IS DRAM
20 006D C300        ;   DEFW    00C3H      ; OUT (E0H),A
21 006F 00          ;   DEFB    00H        ; JP 0000H
22 0070      ;
23 0070      ;   ST0:   ENT
24 0070 06FF        ;   LD      B,FFH      ; BUFFER CLEAR
25 0072 21F110      ;   LD      HL,NAME    ; 10F1H-11F0H CLEAR
26 0075 CDD80F      ;   CALL   ?CLR        ;
27 0078 3E16        ;   LD      A,16H      ; LASTER CLR.
28 007A CD1200      ;   CALL   PRNT        ;
29 007D 3E71        ;   LD      A,71H      ; BACK:BLUE CHA.:WRITE
30 007F 2100D8      ;   LD      HL,D800H   ; COLOR ADDRESS
31 0082 CDD509      ;   CALL   #CLR8       ;
32 0085 218D03      ;   LD      HL,TIMIN   ; INTERRUPT JUMP ROUTINE
33 0088 3EC3        ;   LD      A,C3H      ;
34 008A 323810      ;   LD      (103BH),A  ;
35 008D 223910      ;   LD      (1039H),HL ;
36 0090 3E04        ;   LD      A,04       ; NORMAL TEMPO
37 0092 329E11      ;   LD      (TEMPW),A  ;
38 0095 CDBE02      ;   CALL   MLDSP       ; MELODY STOP
39 0098 CD0900      ;   CALL   NL          ;
40 009B 11E706      ;   LD      DE,MSG?3   ; ** MONITOR 1Z-013A **
41 009E DF          ;   RST      3         ; CALL MGX
42 009F CD7705      ;   CALL   ?BEL        ;
43 00A2      ;   SS:   ENT
44 00A2 3E01        ;   LD      A,01H      ;
45 00A4 329D11      ;   LD      (SWRK),A   ; KEY IN SILENT
46 00A7 2100E8      ;   LD      HL,E800H   ; USR ROM ?
47 00AA 77          ;   LD      (HL),A     ; ROM CHECK
48 00AB 1855        ;   JR      FD2        ;
49 00AD      ;
50 00AD CD0900      ;   CALL   NL          ;
51 00B0 3E2A        ;   LD      A,2AH      ; '* ' PRINT
52 00B2 CD1200      ;   CALL   PRNT        ;
53 00B5 11A311      ;   LD      DE,BUFER   ; GET LINE WORK (11A3H)
54 00B8 CD0300      ;   CALL   GETL        ;
55 00BB 1A          ;   LD      A,(DE)     ;
56 00BC 13          ;   INC     DE         ;
57 00BD FE0D        ;   CP      0DH        ;
58 00BF 28EC        ;   JR      Z,ST1      ;
59 00C1 FE4A        ;   CP      'J'        ; JUMP
60 00C3 282E        ;   JR      Z,GOTO     ;

```

\*\* Z80 ASSEMBLER SB-7201 &lt;1Z-013A&gt; PAGE 03

04.07.83

```

01 00C5 FE4C          CP      'L'      ; LOAD PROGRAM
02 00C7 2848          JR      Z,LOAD
03 00C9 FE46          CP      'F'      ; FLOPPY ACCESS
04 00CB 2832          JR      Z,FD
05 00CD FE42          CP      'B'      ; KEY IN BELL
06 00CF 2826          JR      Z,SG
07 00D1 FE23          CP      'H'      ; CHANG MEMORY
08 00D3 2886          JR      Z,CMYO
09 00D5 FE50          CP      'P'      ; PRINTER TEST
10 00D7 287C          JR      Z,PTEST
11 00D9 FE4D          CP      'M'      ; MEMORY CORRECTION
12 00DB CAAB07        JP      Z,MCOR
13 00DE FE53          CP      'S'      ; SAVED DATA
14 00E0 CA5E0F        JP      Z,SAVE
15 00E3 FE56          CP      'V'      ; VERIFYING DATA
16 00E5 CACB0F        JP      Z,VRFY
17 00E8 FE44          CP      'D'      ; DUMP DATA
18 00EA CA290D        JP      Z,DUMP
19 00ED              ;
20 00ED              ;
21 00ED              ;      DEFS      +4
22 00F1              ;
23 00F1 18C8          JR      ST2      ; NOT COMMAND
24 00F3              ;
25 00F3              ;      JUMP COMMAND
26 00F3              ;
27 00F3 CD3D01        GOTO:  CALL  HEXIY
28 00F6 E9            JP      (HL)
29 00F7              ;
30 00F7              ;      KEY SOUND ON OFF
31 00F7              ;
32 00F7 3A9D11        SG:      LD      A,(SWRK)      ; D0 = SOUND WORK
33 00FA 1F            RRA
34 00FB 3F            CCF
35 00FC 17            RLA
36 00FD 18A5          JR      SS+2
37 00FF              ;
38 00FF              ;      FLOPPY
39 00FF              ;
40 00FF 2100F0        FD:      LD      HL,F000H      ; FLOPPY I/O CHECK
41 0102 7E            FD2:     LD      A,(HL)
42 0103 B7            OR      A
43 0104 20A7          JR      NZ,ST1
44 0106 E9            FD1:     JP      (HL)
45 0107              ;
46 0107              ;
47 0107              ;      ERROR (LOADING)
48 0107              ;
49 0107              ;
50 0107 FE02          ?ER:     ENT     CP      02H      ; A=02H : BREAK IN
51 0109 28A2          JR      Z,ST1
52 010B 114701        LD      DE,MSGE1
53 010E DF            RST      3
54 010F 189C          JR      ST1
55 0111              ;
56 0111              ;
57 0111              ;      LOAD COMMAND
58 0111              ;
59 0111 CDD804        LOAD:     CALL  ?RDI
60 0114 3BF1          JR      C,?ER

```

\*\* Z80 ASSEMBLER SB-7201 &lt;1Z-013A&gt; PAGE 04

04.07.83

```

01 0116 CD0900        LOA0:   CALL  NL
02 0119 11A009        LD      DE,MSG?2      ; LOADING
03 011C DF            RST      3              ; CALL MSGX
04 011D 11F110        LD      DE,NAME      ; FILE NAME
05 0120 DF            RST      3              ; CALL MSGX
06 0121 CDF804        CALL  ?RDD
07 0124 38E1          JR      C,?ER
08 0126 2A0611        LD      HL,(EXADR)      ; EXECUTE ADDRESS
09 0129 7C            LD      A,H
10 012A FE12          CP      12H              ; EXECUTE CHECK
11 012C 38E1          JR      C,LOAD-2
12 012E E9            JP      (HL)
13 012F              ;
14 012F              ;
15 012F              ;
16 012F              ;      GETLINE AND BREAK IN CHECK
17 012F              ;
18 012F              ;      EXIT BREAK IN THEN JUMP (ST1)
19 012F              ;      ACC=TOP OF LINE DATA
20 012F              ;
21 012F              ;
22 012F E3            BGETL:  ENT     EX      (SP),HL
23 0130 C1            POP     BC              ; STACK LOAD
24 0131 11A311        LD      DE,BUFER      ; MONITOR GETLINE BUFF
25 0134 CD0300        CALL  GETL
26 0137 1A            LD      A,(DE)
27 0138 FE1B          CP      1BH              ; BREAK CODE
28 013A 28D3          JR      Z,LOAD-2      ; JP Z,ST1
29 013C E9            JP      (HL)
30 013D              ;
31 013D              ;      ASCII TO HEX CONVERT
32 013D              ;      INPUT (DE)=ASCII
33 013D              ;      CY=1 THEN JUMP (ST1)
34 013D              ;
35 013D              ;
36 013D FDE3          HEXIY:  ENT     EX      (SP),IY
37 013F F1            POP     AF
38 0140 CD1004        CALL  HLHEX
39 0143 38CA          JR      C,LOAD-2      ; JP C,ST1
40 0145 FDE9          JP      (IY)
41 0147              ;
42 0147              ;
43 0147              ;
44 0147              ;
45 0147 43484543      MSGE1:  ENT     DEFM  'CHECK SUM ER.'
46 014B 4B205355
47 014F 4D204552
48 0153 2E
49 0154 0D            DEFB  0DH
50 0155              ;
51 0155              ;
52 0155              ;      PLOTTER PRINTER TEST COMMAND
53 0155              ;      (DPG23)
54 0155              ;      &=CONTROL COMMANDS GROUP
55 0155              ;      C=PEN CHENGE
56 0155              ;      G=GRAPH MODE
57 0155              ;      S=80 CHA. IN 1 LINE
58 0155              ;      L=40 CHA. IN 1 LINE
59 0155              ;      T=PLOTTER TEST
60 0155              ;      IN  (DE)=PRINT DATA

```

```

01 0155      ;
02 0155      PTEST: ENT
03 0155 1A      LD      A,(DE)
04 0156 FE26      CP      '8'
05 0158 2016      JR      NZ,PTST1
06 015A 13      PTST0: INC      DE
07 015B 1A      LD      A,(DE)
08 015C FE4C      CP      'L'
09 015E 2816      JR      Z,.LPT
10 0160 FE53      CP      'S'
11 0162 2817      JR      Z,..LPT
12 0164 FE43      CP      'C'
13 0166 2823      JR      Z,PEN
14 0168 FE47      CP      'G'
15 016A 2818      JR      Z,PLOT
16 016C FE54      CP      'T'
17 016E 2810      JR      Z,PTRN
18 0170      ;
19 0170 CDA501      PTST1: CALL    PMSG
20 0173 C3AD00      JP      ST1
21 0176      ;
22 0176 117004      .LPT: LD      DE,LLPT
23 0179 18F5      JR      PTST1
24 017B      ;
25 017B 11D503      ..LPT: LD      DE,SLPT
26 017E 18F0      JR      PTST1
27 0180      ;
28 0180 3E04      PTRN: LD      A,04H
29 0182 1802      JR      PLOT+2
30 0184      ;
31 0184 3E02      PLOT: LD      A,02H
32 0186 CDBF01      CALL    LPRNT
33 0189 18CF      JR      PTST0
34 018B      ;
35 018B 3E1D      PEN: LD      A,1DH
36 018D 18F7      JR      PLOT+2
37 018F      ;
38 018F      ;
39 018F      ; 1CHA. PRINT TO $LPT
40 018F      ;
41 018F      ; IN: ACC PRINT DATA
42 018F      ;
43 018F      ;
44 018F 0E00      LPRNT: LD      C,0
45 0191 47      LD      B,A
46 0192 CDB601      CALL    RDA
47 0195 7B      LD      A,B
48 0196 D3FF      OUT      (FFH),A
49 0198 3E80      LD      A,80H
50 019A D3FE      OUT      (FEH),A
51 019C 0E01      LD      C,01H
52 019E CDB601      CALL    RDA
53 01A1 AF      XOR      A
54 01A2 D3FE      OUT      (FEH),A
55 01A4 C9      RET
56 01A5      ;
57 01A5      ; $LPT MSG.
58 01A5      ; IN: DE DATA LOW ADR.
59 01A5      ; 0DH MSG. END
60 01A5      ;

```

```

01 01A5 D5      PMSG: PUSH    DE
02 01A6 C5      PUSH    BC
03 01A7 F5      PUSH    AF
04 01A8 1A      PMSG1: LD      A,(DE)
05 01A9 CDBF01      CALL    LPRNT
06 01AC 1A      LD      A,(DE)
07 01AD 13      INC      DE
08 01AE FE0D      CP      0DH
09 01B0 20F6      JR      NZ,PMSG1
10 01B2 F1      POP      AF
11 01B3 C1      POP      BC
12 01B4 D1      POP      DE
13 01B5 C9      RET
14 01B6      ;
15 01B6      ; RDA CHECK
16 01B6      ;
17 01B6      ; BRKEY IN TO MONITOR RETURN
18 01B6      ; IN: C RDA CODE
19 01B6      ;
20 01B6 DBFE      RDA: IN      A,(FE)
21 01B8 E60D      AND      0DH
22 01BA B9      CP      C
23 01BB C8      RET      Z
24 01BC CD1E00      CALL    BRKEY
25 01BF 20F5      JR      NZ,RDA
26 01C1 31F010      LD      SP,SP
27 01C4 C3AD00      JP      ST1
28 01C7      ;
29 01C7      ;
30 01C7      ; ORG 01C7H
31 01C7      ;
32 01C7      ; MELODY
33 01C7      ;
34 01C7      ; DE=DATA LOW ADR.
35 01C7      ; EXIT CF=1 BREAK
36 01C7      ; CF=0 OK
37 01C7      ;
38 01C7      ; ?MLDY: ENT
39 01C7 C5      PUSH    BC
40 01C8 D5      PUSH    DE
41 01C9 E5      PUSH    HL
42 01CA 3E02      LD      A,02H
43 01CC 32A011      LD      (OCTV),A
44 01CF 0601      LD      B,01
45 01D1 1A      LD      A,(DE)
46 01D2 FE0D      CP      0DH
47 01D4 283B      JR      Z,MLD4
48 01D6 FEC8      CP      0BH
49 01D8 2837      JR      Z,MLD4
50 01DA FE0F      CP      CFH
51 01DC 2827      JR      Z,MLD2
52 01DE FE2D      CP      2DH
53 01E0 2823      JR      Z,MLD2
54 01E2 FE2B      CP      2BH
55 01E4 2827      JR      Z,MLD3
56 01E6 FED7      CP      D7H
57 01E8 2823      JR      Z,MLD3
58 01EA FE23      CP      23H
59 01EC 216C02      LD      HL,MTBL
60 01EF 2004      JR      NZ,+6

```

\*\* Z80 ASSEMBLER SB-7201 &lt;1Z-013A&gt; PAGE 07

04.07.83

```

01 01F1 21B402      LD      HL,M#TBL
02 01F4 13          INC      DE
03 01F5 CD1C02      CALL    ONPU      ; ONTYO SET
04 01F8 3BD7        JR       C,MLD1
05 01FA CDC802      CALL    RYTHM
06 01FD 3815        JR       C,MLD5
07 01FF CDAB02      CALL    MLDST    ; MELODY START
08 0202 41          LD       B,C
09 0203 18CC        JR       MLD1
10 0205 3E03        MLD2: LD      A,+3
11 0207 32A011      LD      (OCTV),A
12 020A 13          INC      DE
13 020B 18C4        JR       MLD1
14 020D 3E01        MLD3: LD      A,1
15 020F 18F6        JR       MLD2+2
16 0211 CDC802      MLD4: CALL    RYTHM
17 0214 F5          MLD5: PUSH    AF
18 0215 CDBE02      CALL    MLDSP
19 0218 F1          POP      AF
20 0219 C39B06      JP       RET3
21 021C            ;
22 021C            ; ONPU TO RATIO CONV
23 021C            ;
24 021C            ; EXIT (RATIO)=RATIO VALUE
25 021C            ; C=ONTYO*TEMPO
26 021C            ;
27 021C            ONPU: ENT
28 021C C5          PUSH    BC
29 021D 060B        LD      B,B
30 021F 1A          ONP1: LD      A,(DE)
31 0220 BE          CP      (HL)
32 0221 2809        JR       Z,ONP2
33 0223 23          INC      HL
34 0224 23          INC      HL
35 0225 23          INC      HL
36 0226 10F8        DJNZ    -6
37 0228 37          SCF
38 0229 13          INC      DE
39 022A C1          POP      BC
40 022B C9          RET
41 022C 23          ONP2: INC      HL
42 022D D5          PUSH    DE
43 022E 5E          LD      E,(HL)
44 022F 23          INC      HL
45 0230 56          LD      D,(HL)
46 0231 EB          EX      DE,HL
47 0232 7C          LD      A,H
48 0233 B7          OR      A
49 0234 2809        JR       Z,+11
50 0236 3AA011      LD      A,(OCTV) ; 11A0H OCTAVE WORK
51 0239 3D          DEC      A
52 023A 2803        JR       Z,+5
53 023C 29          ADD     HL,HL
54 023D 18FA        JR       -4
55 023F 22A111      LD      (RATIO),HL ; 11A1H ONPU RATIO
56 0242 21A011      LD      HL,OCTV
57 0245 3602        LD      (HL),2
58 0247 2B          DEC      HL
59 0248 D1          POP      DE
60 0249 13          INC      DE

```

\*\* Z80 ASSEMBLER SB-7201 &lt;1Z-013A&gt; PAGE 08

04.07.83

```

01 024A 1A          LD      A,(DE)
02 024B 47          LD      B,A
03 024C E6F0        AND     FOH      ; ONTYO ?
04 024E FE30        CP      30H
05 0250 2803        JR       Z,+5
06 0252 7E          LD      A,(HL) ; HL=ONTYO
07 0253 1805        JR       +7
08 0255 13          INC      DE
09 0256 78          LD      A,B
10 0257 E60F        AND     OFH
11 0259 77          LD      (HL),A ; HL=ONTYO
12 025A 219C02      LD      HL,OPTBL
13 025D 85          ADD     A,L
14 025E 6F          LD      L,A
15 025F 4E          LD      C,(HL)
16 0260 3A9E11      LD      A,(TEMPW)
17 0263 47          LD      B,A
18 0264 AF          XOR      A
19 0265 81          ONP3: ADD     A,C
20 0266 10FD        DJNZ    -1
21 0268 C1          POP      BC
22 0269 4F          LD      C,A
23 026A AF          XOR      A
24 026B C9          RET
25 026C            ;
26 026C            ;
27 026C            MTBL: ENT
28 026C 43          DEFB    43H      ; C
29 026D 4608        DEFW    0B46H
30 026F 44          DEFB    44H      ; D
31 0270 5F07        DEFW    075FH
32 0272 45          DEFB    45H      ; E
33 0273 9106        DEFW    0691H
34 0275 46          DEFB    46H      ; F
35 0276 3306        DEFW    0633H
36 0278 47          DEFB    47H      ; G
37 0279 8605        DEFW    0586H
38 027B 41          DEFB    41H      ; A
39 027C EC04        DEFW    04ECH
40 027E 42          DEFB    42H      ; B
41 027F 6404        DEFW    0464H
42 0281 52          DEFB    52H      ; R
43 0282 0000        DEFW    0
44 0284            M#TBL: ENT
45 0284 43          DEFB    43H      ; #C
46 0285 CF07        DEFW    07CFH
47 0287 44          DEFB    44H      ; #D
48 0288 F506        DEFW    06F5H
49 028A 45          DEFB    45H      ; #E
50 028B 3306        DEFW    0633H
51 028D 46          DEFB    46H      ; #F
52 028E DA05        DEFW    05DAH
53 0290 47          DEFB    47H      ; #G
54 0291 3705        DEFW    0537H
55 0293 41          DEFB    41H      ; #A
56 0294 A504        DEFW    04A5H
57 0296 42          DEFB    42H      ; #B
58 0297 2304        DEFW    0423H
59 0299 52          DEFB    52H      ; #R
60 029A

```

```

01 029A 0000      DEFW 0
02 029C          OPTBL: ENT
03 029C 01        DEFB 1
04 029D 02        DEFB 2
05 029E 03        DEFB 3
06 029F 04        DEFB 4
07 02A0 06        DEFB 6
08 02A1 08        DEFB 8
09 02A2 0C        DEFB 0CH
10 02A3 10        DEFB 10H
11 02A4 18        DEFB 18H
12 02A5 20        DEFB 20H
13 02A6          ;
14 02A6          ;
15 02A6          ;
16 02A6          ; INCREMENT DE REG.
17 02A6          ;
18 02A6          ;
19 02A6 13        .4DE: ENT INC DE
20 02A7 13        INC DE
21 02A8 13        INC DE
22 02A9 13        INC DE
23 02AA C9        RET
24 02AB          ;
25 02AB          ;
26 02AB          ;
27 02AB          ; ORG 02ABH ; MLDST
28 02AB          ;
29 02AB          ; MELODY START & STOP
30 02AB          ;
31 02AB          MLDST: ENT
32 02AB 2AA111    LD HL,(RATIO)
33 02AE 7C        LD A,H
34 02AF B7        OR A
35 02B0 280C      JR Z,MLDSP
36 02B2 D5        PUSH DE
37 02B3 EB        EX DE,HL
38 02B4 2104E0    LD HL,CONTO
39 02B7 73        LD (HL),E
40 02B8 72        LD (HL),D
41 02B9 3E01      LD A,1
42 02BB D1        POP DE
43 02BC 1806      JR MLDS1
44 02BE          ;
45 02BE          MLDSP: ENT
46 02BE 3E36      LD A,36H ; MODE SET (8253 C0)
47 02C0 3207E0    LD (CONTF),A ; E007H
48 02C3 AF        XOR A
49 02C4 3208E0    MLDSP: LD (SUNDG),A ; E008H
50 02C7 C9        RET ; TEHRO RESET
51 02C8          ;
52 02C8          ; RHYTHM
53 02C8          ;
54 02C8          ; B=COUNT DATA
55 02C8          ; IN
56 02C8          ; EXIT CF=1 BREAK
57 02C8          ; CF=0 OK
58 02C8          ;
59 02C8          RYTHM: ENT
60 02C8 2100E0    LD HL,KEYPA ; E000H

```

```

01 02CB 36FB      LD (HL),FBH
02 02CD 23        INC HL
03 02CE 7E        LD A,(HL)
04 02CF E681      AND B1H ; BREAK IN CHECK
05 02D1 2002      JR NZ,+4
06 02D3 37        SCF
07 02D4 C9        RET
08 02D5 3A08E0    LD A,(TEMP) ; E008H
09 02D8 0F        RRCA ; TEMPO OUT
10 02D9 38FA      JR C,-4
11 02DB 3A08E0    LD A,(TEMP)
12 02DE 0F        RRCA
13 02DF 30FA      JR NC,-4
14 02E1 10F2      DJNZ -12
15 02E3 AF        XOR A
16 02E4 C9        RET
17 02E5          ;
18 02E5          ;
19 02E5          ; TEMPO SET
20 02E5          ;
21 02E5          ; ACC=VALUE (1-7)
22 02E5          ;
23 02E5          ; ?TEMP: ENT
24 02E5 F5        PUSH AF
25 02E6 C5        PUSH BC
26 02E7 E60F      AND 0FH
27 02E9 47        LD B,A
28 02EA 3E08      LD A,B
29 02EC 90        SUB B
30 02ED 329E11    LD (TEMPW),A
31 02F0 C1        POP BC
32 02F1 F1        POP AF
33 02F2 C9        RET
34 02F3          ;
35 02F3          ; CRT MANAGMENT
36 02F3          ;
37 02F3          ;
38 02F3          ; EXIT HL:DSPXY H=Y,L=X
39 02F3          ; DE:MANG ADR. (ON DSPXY)
40 02F3          ; A:MANG DATA
41 02F3          ; CY:MANG=1
42 02F3          ;
43 02F3 217311    .MANG: ENT LD HL,MANG ; CRT MANG. POINTER
44 02F6 3A7211    LD A,(1172H) ; DSPXY+1
45 02F9 85        ADD A,L
46 02FA 6F        LD L,A
47 02FB 7E        LD A,(HL)
48 02FC 23        INC HL
49 02FD CB16      RL (HL)
50 02FF B6        OR (HL)
51 0300 CB1E      RR (HL)
52 0302 0F        RRCA
53 0303 EB        EX DE,HL
54 0304 2A7111    LD HL,(DSPXY)
55 0307 C9        RET
56 0308          ;
57 0308          ;
58 0308          ;
59 0308          ; ORG 0308H
60 0308          ;

```



\*\* Z80 ASSEMBLER SB-7201 &lt;1Z-013A&gt; PAGE 11

04.07.83

```

01 0308      ; TIME SET
02 0308      ;
03 0308      ; ACC=0 : AM
04 0308      ; =1 : PM
05 0308      ; DE=SEC: BINARY
06 0308      ;
07 0308      ?TMS1: ENT
08 0308 F3      DI
09 0309 C5      PUSH BC
10 030A D5      PUSH DE
11 030B E5      PUSH HL
12 030C 329B11  LD (AMPM),A ; AMPM DATA
13 030F 3EF0      LD A,FOH
14 0311 329C11  LD (TIMFG),A ; TIME FLAG
15 0314 21C0AB  LD HL,ABCOH ; 12H
16 0317 AF      XOR A
17 031B ED52      SBC HL,DE ; COUNT DATA = 12H-IN DA
18 031A E5      PUSH HL
19 031B 00      NOP
20 031C EB      EX DE,HL
21 031D 2107E0  LD HL,CONTF ; E007H
22 0320 3674      LD (HL),74H
23 0322 36B0      LD (HL),80H
24 0324 2B      DEC HL ; CONT2
25 0325 73      LD (HL),E
26 0326 72      LD (HL),D
27 0327 2B      DEC HL ; CONT1
28 0328 360A      LD (HL),0AH
29 032A 3600      LD (HL),0
30 032C 23      INC HL
31 032D 23      INC HL ; CONTF
32 032E 3680      LD (HL),80H
33 0330 2B      DEC HL ; CONT2
34 0331 4E      LD C,(HL)
35 0332 7E      LD A,(HL)
36 0333 BA      CP D
37 0334 20FB      JR NZ,?TMS1
38 0336 79      LD A,C
39 0337 BB      CP E
40 0338 20F7      JR NZ,?TMS1
41 033A 2B      DEC HL
42 033B 00      NOP
43 033C 00      NOP
44 033D 00      NOP
45 033E 36FB      LD (HL),FBH ; 1SEC
46 0340 363C      LD (HL),3CH
47 0342 23      INC HL
48 0343 D1      POP DE
49 0344 4E      LD C,(HL)
50 0345 7E      LD A,(HL)
51 0346 BA      CP D
52 0347 20FB      JR NZ,?TMS2
53 0349 79      LD A,C
54 034A BB      CP E
55 034B 20F7      JR NZ,?TMS2
56 034D E1      POP HL
57 034E D1      POP DE
58 034F C1      POP BC
59 0350 FB      EI
60 0351 C9      RET

```

\*\* Z80 ASSEMBLER SB-7201 &lt;1Z-013A&gt; PAGE 12

04.07.83

```

01 0352      ;
02 0352      ;
03 0352      ; BELL DATA
04 0352      ;
05 0352      ?BELD: ENT
06 0352 D7      DEFB D7H
07 0353 4130      DEFM 'A0'
08 0355 0D      DEFB 0DH
09 0356      ;
10 0356      ;
11 0356      ;
12 0356      DEFS +2
13 0358      ;ORG 0358H
14 0358      ;
15 0358      ; TIME READ
16 0358      ;
17 0358      ; EXIT ACC=0 : AM
18 0358      ; =1 : PM
19 0358      ; DE=SEC. BINARY
20 0358      ;
21 0358      ?TMRD: ENT
22 0358 E5      PUSH HL
23 0359 2107E0  LD HL,CONTF
24 035C 3680      LD (HL),80H
25 035E 2B      DEC HL ; CONT2
26 035F F3      DI
27 0360 5E      LD E,(HL)
28 0361 56      LD D,(HL)
29 0362 FB      EI
30 0363 7B      LD A,E
31 0364 B2      OR D
32 0365 280E      JR Z,?TMR1
33 0367 AF      XOR A
34 0368 21C0AB  LD HL,ABCOH
35 036B ED52      SBC HL,DE
36 036D 3B10      JR C,?TMR2
37 036F EB      EX DE,HL
38 0370 3A9B11  LD A,(AMPM)
39 0373 E1      POP HL
40 0374 C9      RET
41 0375 11C0AB  ?TMR1: LD DE,ABCOH ; 12H
42 0378 3A9B11  LD A,(AMPM)
43 037B EE01      XOR 1
44 037D E1      POP HL
45 037E C9      RET
46 037F F3      DI
47 0380 2106E0  ?TMR2: LD HL,CONTF
48 0383 7E      LD A,(HL)
49 0384 2F      CPL
50 0385 5F      LD E,A
51 0386 7E      LD A,(HL)
52 0387 2F      CPL
53 0388 57      LD D,A
54 0389 FB      EI
55 038A 13      INC DE
56 038B 18EB      JR ?TMR1+3
57 038D      ;
58 038D      ; TIME INTERRUPT
59 038D      ;
60 038D      TIMIN: ENT

```

```

01 038D F5      PUSH AF
02 038E C5      PUSH BC
03 038F D5      PUSH DE
04 0390 E5      PUSH HL
05 0391 219B11  LD HL,AMPM
06 0394 7E      LD A,(HL)
07 0395 EE01    XOR 1
08 0397 77      LD (HL),A
09 0398 2107E0  LD HL,CONTF
10 039B 3680    LD (HL),BOH
11 039D 2B      DEC HL
12 039E E5      PUSH HL
13 039F 5E      LD E,(HL)
14 03A0 56      LD D,(HL)
15 03A1 21C0A8  LD HL,ABCOH
16 03A4 19      ADD HL,DE
17 03A5 2B      DEC HL
18 03A6 2B      DEC HL
19 03A7 EB      EX DE,HL
20 03A8 E1      POP HL
21 03A9 73      LD (HL),E
22 03AA 72      LD (HL),D
23 03AB E1      POP HL
24 03AC D1      POP DE
25 03AD C1      POP BC
26 03AE F1      POP AF
27 03AF FB      EI
28 03B0 C9      RET

;
; SPACE PRINT AND DISP ACC
;
; INPUT:HL=DISP. ADR.
;
SPHEX: ENT
CALL ?PRTS ; SP.PRINT
LD A,(HL)
CALL PRTHX ; DSP OF ACC (ASCII)
LD A,(HL)
RET

;
;
; ORG 03BAH
;
; (ASCII PRINT) FOR HL
;
PRTHL: ENT
LD A,H
CALL PRTHX
LD A,L
JR PRTHX

;
; DEFS +2
; ORG 03C3H;PRTHX
;
; (ASCII PRINT) FOR ACC
;
PRTHX: ENT
PUSH AF
RRCA

```

```

01 03C5 0F      RRCA
02 03C6 0F      RRCA
03 03C7 0F      RRCA
04 03C8 CDDA03  CALL ASC
05 03CB CD1200  CALL PRNT
06 03CE F1      POP AF
07 03CF CDDA03  CALL ASC
08 03D2 C31200  JP PRNT
09 03D5
10 03D5
11 03D5
12 03D5
13 03D5
14 03D5
15 03D5
SLPT: ENT
DEFB 01H ; TEXT MODE
DEFB 09H
DEFB 09H
DEFB 09H
DEFB 0DH
;
; ORG 03DAH;ASC
;
; HEXADECIMAL TO ASCII
; IN : ACC (D3-D0)=HEXADECIMAL
; EXIT: ACC = ASCII
;
ASC: ENT
AND 0FH
CP 0AH
JR C,NOADD
ADD A,7
NOADD: ENT
ADD A,30H
RET
;
; ASCII TO HEXADECIMAL
; IN : ACC = ASCII
; EXIT : ACC = HEXADECIMAL
; CY = 1 ERROR
;
HEXJ: ENT
SUB 30H
RET C
CP 0AH
CCF
RET NC
SUB 7
CP 10H
CCF
RET C
CP 0AH
RET
;
; DEFS +4
; ORG 03F9H;HEX
HEX: ENT
JR HEXJ
;

```

\*\* Z80 ASSEMBLER SB-7201 &lt;1Z-013A&gt; PAGE 15

04.07.83

```

01 03FB      ; PRASS PLAY MESSAGE
02 03FB      ;
03 03FB      MSG#1: ENT
04 03FB 7F20      DEFW 207FH
05 03FD      MSG#2: ENT
06 03FD 504C4159  DEFM 'PLAY'
07 0401 0D      DEFB 0DH
08 0402      MSG#3: ENT
09 0402 7F20      DEFW 207FH
10 0404 5245434F  DEFM 'RECORD.' ; PRESS RECORD
11 0408 52442E      DEFB 0DH
12 040B 0D      DEFB 0DH
13 040C      ;
14 040C      ;
15 040C      DEFS +4
16 0410      ;ORG 0410H;HLHEX
17 0410      ;
18 0410      ;
19 0410      ; 4 ASCII TO (HL)
20 0410      ;
21 0410      ; IN DE=DATA LOW ADR.
22 0410      ; EXIT CF=0 : OK
23 0410      ; =1 : OUT
24 0410      ;
25 0410      HLHEX: ENT
26 0410 D5      PUSH DE
27 0411 CD1F04   CALL 2HEX
28 0414 3807     JR C,+9
29 0416 67      LD H,A
30 0417 CD1F04   CALL 2HEX
31 041A 3801     JR C,+3
32 041C 6F      LD L,A
33 041D D1      POP DE
34 041E C9      RET
35 041F      ;
36 041F      ;ORG 041FH;2HEX
37 041F      ;
38 041F      ;
39 041F      ; 2 ASCII TO (ACC)
40 041F      ;
41 041F      ; IN DE=DATA LOW ADR.
42 041F      ;
43 041F      ; EXIT CF=0 : OK
44 041F      ; =1 : OUT
45 041F      ;
46 041F      ;
47 041F      2HEX: ENT
48 041F C5      PUSH BC
49 0420 1A      LD A,(DE)
50 0421 13      INC DE
51 0422 CDF903   CALL HEX
52 0425 380D     JR C,+15
53 0427 0F      RRCA
54 0428 0F      RRCA
55 0429 0F      RRCA
56 042A 0F      RRCA
57 042B 4F      LD C,A
58 042C 1A      LD A,(DE)
59 042D 13      INC DE
60 042E CDF903   CALL HEX
61 0431 3801     JR C,+3

```

\*\* Z80 ASSEMBLER SB-7201 &lt;1Z-013A&gt; PAGE 16

04.07.83

```

01 0433 B1      OR C
02 0434 C1      2HE1: POP BC
03 0435 C9      RET
04 0436      ;
05 0436      ;
06 0436      ; WRITE INFORMATION
07 0436      ;
08 0436      ?WRI: ENT
09 0436 F3      DI
10 0437 D5      PUSH DE
11 0438 C5      PUSH BC
12 0439 E5      PUSH HL
13 043A 16D7     LD D,D7H ; 'W'
14 043C 1ECC     LD E,CCH ; 'L'
15 043E 21F010   LD HL,IBUFE ; 10F0H
16 0441 018000   LD BC,80H ; WRITE BYTE SIZE
17 0444 CD1A07   CALL CKSUM ; CHECK SUM
18 0447 CD9F06   CALL MOTOR ; MOTOR ON
19 044A 3818     JR C,WRI3
20 044C 7B      LD A,E
21 044D FECC     CP CCH ; 'L'
22 044F 200D     JR NZ,WRI2
23 0451 CD0900   CALL NL
24 0454 D5      PUSH DE
25 0455 116704   LD DE,MSG#7 ; WRITING
26 0458 DF      RST 3 ; CALL MSGX
27 0459 11F110   LD DE,NAME ; FILE NAME
28 045C DF      RST 3 ; CALL MSGX
29 045D D1      POP DE
30 045E CD7A07   WRI2: CALL GAP
31 0461 CD8A04   CALL WTAPE
32 0464 C35405   WRI3: JP RET2
33 0467      ;
34 0467      ;
35 0467 57524954 MSG#7: ENT
36 046B 494E4720 DEFM 'WRITING'
37 046F 0D      DEFB 0DH
38 0470      ;
39 0470      ;
40 0470      ;
41 0470      ; 40 CHA. IN 1 LINE CODE (DATA)
42 0470      ;
43 0470      ;
44 0470 01      LLPT: ENT
45 0471 09      DEFB 01H ; TEXT MODE
46 0472 09      DEFB 09H
47 0473 0B      DEFB 0BH
48 0474 0D      DEFB 0DH
49 0475      ;
50 0475      ;ORG 0475H
51 0475      ;
52 0475      ;
53 0475      ; WRITE DATA
54 0475      ;
55 0475      ; EXIT CF=0 : OK
56 0475      ; =1 : BREAK
57 0475      ;
58 0475      ;
59 0475 F3      ?WRD: ENT
60 0476 D5      DI

```

```

01 0477 C5          PUSH BC
02 0478 E5          PUSH HL
03 0479 16D7        LD D,D7H      ; 'W'
04 047B 1E53        LD E,53H      ; 'S'
05 047D ED4B0211    LD BC,(SIZE)  ; WRITE DATA BYTE SIZE
06 0481 2A0411      LD HL,(DTADR) ; WRITE DATA ADDRESS
07 0484 78          LD A,B
08 0485 B1          OR C
09 0486 2B4A        JR Z,RET1
10 0488 18BA        JR WRI1
11 048A            ;
12 048A            ;
13 048A            ; TAPE WRITE
14 048A            ;
15 048A            ; BC=BYTE SIZE
16 048A            ; HL=DATA LOW ADR.
17 048A            ;
18 048A            ; EXIT CF=0 : OK
19 048A            ; =1 : BREAK
20 048A            ;
21 048A D5          WTAPE: PUSH DE
22 048B C5          PUSH BC
23 048C E5          PUSH HL
24 048D 1602        LD D,2
25 048F 3EF8        LD A,F8H
26 0491 3200E0      LD (KEYPA),A   ; E000H
27 0494 7E          LD A,(HL)
28 0495 CD6707      CALL WBYTE      ; 1 BYTE WRITE
29 0498 3A01E0      LD A,(KEYPB)   ; E001H
30 049B E681        AND B1H        ; SHIFT & BREAK
31 049D C2A504      JP NZ,WTAP2    ; BREAK IN CODE
32 04A0 3E02        LD A,02H
33 04A2 37          SCF
34 04A3 182D        JR WTAP3
35 04A5 23          WTAP2: INC HL
36 04A6 0B          DEC BC
37 04A7 78          LD A,B
38 04A8 B1          OR C
39 04A9 C29404      JP NZ,WTAP1
40 04AC 2A9711      LD HL,(SUMDT)  ; SUM DATA SET
41 04AF 7C          LD A,H
42 04B0 CD6707      CALL WBYTE
43 04B3 7D          LD A,L
44 04B4 CD6707      CALL WBYTE
45 04B7 CD1A0A      CALL LONG
46 04BA 15          DEC D
47 04BB C2C204      JP NZ,+7
48 04BE B7          OR A
49 04BF C3D204      JP WTAP3
50 04C2 0600        LD B,0
51 04C4 CD010A      CALL SHORT
52 04C7 05          DEC B
53 04C8 C2C404      JP NZ,-4
54 04CB E1          POP HL
55 04CC C1          POP BC
56 04CD C5          PUSH BC
57 04CE E5          PUSH HL
58 04CF C39404      JP WTAP1
59 04D2            ;
60 04D2 E1          RET1: POP HL

```

```

01 04D3 C1          POP BC
02 04D4 D1          POP DE
03 04D5 C9          RET
04 04D6            ;
05 04D6            ;
06 04D6            ;
07 04D6            ;
08 04D6            ;
09 04D8            ; ORG 04D8H
10 04D8            ;
11 04D8            ;
12 04D8            ; READ INFORMATION (FROM $CMT)
13 04D8            ;
14 04D8            ; EXIT ACC=0 : OK CF=0
15 04D8            ; =1 : ER CF=1
16 04D8            ; =2 : BREAK CF=1
17 04D8            ;
18 04D8            ; ?RDI: ENT
19 04D8 F3          DI
20 04D9 D5          PUSH DE
21 04DA C5          PUSH BC
22 04DB E5          PUSH HL
23 04DC 16D2        LD D,D2H      ; 'R'
24 04DE 1ECC        LD E,CCH      ; 'L'
25 04E0 018000      LD BC,80H
26 04E3 21F010      LD HL,IBUFE
27 04E6            ; RD1: ENT
28 04E6 CD9F06      CALL MOTOR
29 04E9 DA7205      JP C,RTP6
30 04EC CD5B06      CALL TMARK
31 04EF DA7205      JP C,RTP6
32 04F2 CD0E05      CALL RTAPE
33 04F5 C35405      JP RTP4
34 04F8            ;
35 04F8            ;
36 04F8            ;
37 04F8            ; ORG 04F8H
38 04F8            ;
39 04F8            ;
40 04F8            ; READ DATA (FROM $CMT)
41 04F8            ;
42 04F8            ; EXIT SAME UP
43 04F8            ;
44 04F8            ; ?RDD: ENT
45 04F8 F3          DI
46 04F9 D5          PUSH DE
47 04FA C5          PUSH BC
48 04FB E5          PUSH HL
49 04FC 16D2        LD D,D2H      ; 'R'
50 04FE 1E53        LD E,53H      ; 'S'
51 0500 ED4B0211    LD BC,(SIZE)
52 0504 2A0411      LD HL,(DTADR)
53 0507 78          LD A,B
54 0508 B1          OR C
55 0509 CA5405      JP Z,RTP4
56 050C 18D8        JR RD1
57 050E            ;
58 050E            ;
59 050E            ; READ TAPE
60 050E            ;

```

\*\* Z80 ASSEMBLER SB-7201 &lt;1Z-013A&gt; PAGE 19

04.07.83

```

01 050E      ; IN BC=SIZE
02 050E      ; DE=LOAD ADR.
03 050E      ;
04 050E      ; EXIT ACC=0 : OK CF=0
05 050E      ; =1 : ER =1
06 050E      ; =2 : BREAK=1
07 050E      ;
08 050E      RTAPE: ENT
09 050E D5    PUSH DE
10 050F C5    PUSH BC
11 0510 E5    PUSH HL
12 0511 2602  LD H,2 ; TWICE WRITE
13 0513      RTP1: ENT
14 0513 0101E0 LD BC,KEYPB
15 0516 1102E0 LD DE,CSTR
16 0519      RTP2: ENT
17 0519 CD0106 CALL EDGE ; 1+0 EDGE DETECT
18 051C 3B54 JR C,RTP6
19 051E CD4A0A CALL DLY3 ; CALL DLY2*3
20 0521 1A LD A,(DE) ; DATA (1BIT) READ
21 0522 E620 AND 20H
22 0524 CA1905 JP Z,RTP2
23 0527 54 LD D,H
24 0528 210000 LD HL,0
25 052B 229711 LD (SUMDT),HL
26 052E E1 POP HL ;
27 052F C1 POP BC
28 0530 C5 PUSH BC
29 0531 E5 PUSH HL
30 0532      RTP3: ENT
31 0532 CD2406 CALL RBYTE ; 1BYTE READ
32 0535 3B3B JR C,RTP6
33 0537 77 LD (HL),A
34 0538 23 HL INC
35 0539 0B DEC BC
36 053A 78 LD A,B
37 053B B1 OR C
38 053C 20F4 JR NZ,RTP3
39 053E 2A9711 LD HL,(SUMDT) ; CHECK SUM
40 0541 CD2406 CALL RBYTE ; CHECK SUM DATA
41 0544 3B2C JR C,RTP6
42 0546 5F LD E,A
43 0547 CD2406 CALL RBYTE ; CHECK SUM DATA
44 054A 3B26 JR C,RTP6
45 054C BD CP L
46 054D 2016 JR NZ,RTP5
47 054F 7B LD A,E
48 0550 BC CP H
49 0551 2012 JR NZ,RTP5
50 0553      RTP8: ENT
51 0553 AF XOR A
52 0554      RTP4: ENT
53 0554      RET2: ENT
54 0554 E1 POP HL
55 0555 C1 POP BC
56 0556 D1 POP DE
57 0557 CD0007 CALL MSTOP
58 055A F5 PUSH AF
59 055B 3A9C11 LD A,(TIMFG) ; INT. CHECK
60 055E FEFO CP FOH

```

\*\* Z80 ASSEMBLER SB-7201 &lt;1Z-013A&gt; PAGE 20

04.07.83

```

01 0560 2001 JR NZ,+3
02 0562 FB EI
03 0563 F1 POP AF
04 0564 C9 RET
05 0565      ;
06 0565      RTP5: ENT
07 0565 15 DEC D
08 0566 2B06 JR Z,RTP7
09 0568 62 LD H,D
10 0569 CDE20F CALL GAPCK
11 056C 18A5 JR RTP1
12 056E      RTP7: ENT
13 056E 3E01 LD A,1
14 0570 1B02 JR RTP9
15 0572      RTP6: ENT
16 0572 3E02 LD A,2
17 0574      RTP9: ENT
18 0574 37 SCF
19 0575 1BDD JR RTP4
20 0577      ;
21 0577      ; BELL
22 0577      ;
23 0577      ;
24 0577      ?BEL: ENT
25 0577 D5 PUSH DE
26 057B 115203 LD DE,?BELD
27 057B F7 RST 6 ; CALL MELDY
28 057C D1 POP DE
29 057D C9 RET
30 057E      ;
31 057E      ; FLASING AND KEYIN
32 057E      ; EXIT:ACC INPUT KEY DATA(DSP.CODE)
33 057E      ; H=FOH THEN NO KEYIN(Z FLG.)
34 057E      ;
35 057E      FLKEY: ENT
36 057E CDFF09 CALL ?FLAS
37 05B1 CDCA0B CALL ?KEY
38 05B4 FEFO CP FOH
39 05B6 C9 RET
40 05B7      ;
41 05B7      ;
42 05B7      ;
43 05B7      ;
44 05B7      ;
45 05B7      DEFS +1
46 05B8      ;ORG 05B8H
47 05B8      ;
48 05B8      ;
49 05B8      ; VERIFY (FROM $CMT)-
50 05B8      ;
51 05B8      ; EXIT ACC =0 : OK CF=0
52 05B8      ; =1 : ER CF=1
53 05B8      ; =2 : BREAK CF=1
54 05B8      ;
55 05B8      ?VRFY: ENT
56 05B8 F3 DI
57 05B9 D5 PUSH DE
58 05BA C5 PUSH BC
59 05BB E5 PUSH HL
60 05BC ED4B0211 LD BC,(SIZE)

```

```

01 0590 2A0411      LD    HL,(DTADR)
02 0593 16D2        LD    D,D2H
03 0595 1E53        LD    E,53H
04 0597 78          LD    A,B
05 0598 B1          OR    C
06 0599 28B9        JR    Z,RTP4
07 059B CD1A07      CALL  CKSUM
08 059E CD9F06      CALL  MOTOR
09 05A1 38CF        JR    C,RTP6
10 05A3 CD5B06      CALL  TMARK
11 05A6 38CA        JR    C,RTP6
12 05A8 CDAD05      CALL  TVRFY
13 05AB 18A7        JR    RTP4
14 05AD             ;
15 05AD             ;
16 05AD             ; DATA VERIFY
17 05AD             ;
18 05AD             ; BC=SIZE
19 05AD             ; HL=DATA LOW ADR
20 05AD             ; CSMDT=CHECK SUM
21 05AD             ; EXIT ACC=0 : OK CF=0
22 05AD             ;      =1 : ER  =1
23 05AD             ;      =2 : BREAK=1
24 05AD             ;
25 05AD             ;
26 05AD TVRFY: ENT
27 05AD             PUSH  DE
28 05AE C5          PUSH  BC
29 05AF E5          PUSH  HL
30 05B0 2602        LD    H,2
31 05B2 TVF1: ENT
32 05B2 0101E0      LD    BC,KEYPB
33 05B5 1102E0      LD    DE,CSTR
34 05B8 TVF2: ENT
35 05B8 CD0106      CALL  EDGE
36 05BB DA7205      JP    C,RTP6
37 05BE CD4A0A      CALL  DLY3
38 05C1 1A          LD    A,(DE)
39 05C2 E620        AND    20H
40 05C4 CAB805      JP    Z,TVF2
41 05C7 54          LD    D,H
42 05C8 E1          POP   HL
43 05C9 C1          POP   BC
44 05CA C5          PUSH  BC
45 05CB E5          PUSH  HL
46 05CC TVF3: ENT
47 05CC CD2406      CALL  RBYTE
48 05CF 38A1        JR    C,RTP6
49 05D1 BE          CP    (HL)
50 05D2 209A        JR    NZ,RTP7
51 05D4 23          INC   HL
52 05D5 0B          DEC   BC
53 05D6 78          LD    A,B
54 05D7 B1          OR    C
55 05D8 20F2        JR    NZ,TVF3
56 05DA 2A9911      LD    HL,(CSMDT)
57 05DD CD2406      CALL  RBYTE
58 05E0 BC          CP    H
59 05E1 208B        JR    NZ,RTP7
60 05E3 CD2406      CALL  RBYTE

```

```

01 05E6 BD          CP    L
02 05E7 2085        JR    NZ,RTP7
03 05E9 15          DEC   D
04 05EA CA5305      JP    Z,RTP8
05 05ED 62          LD    H,D
06 05EE 18C2        JR    TVF1
07 05F0             ;
08 05F0             ; FLASHING DATA LOAD
09 05F0             ;
10 05F0 ?LOAD: ENT
11 05F0 F5          PUSH  AF
12 05F1 3ABE11      LD    A,(FLASH)
13 05F4 CDB10F      CALL  ?PONT
14 05F7 77          LD    (HL),A
15 05F8 F1          POP   AF
16 05F9 C9          RET
17 05FA             ;
18 05FA             ;
19 05FA             ; NEW LINE AND PRINT HL REG.(ASCII)
20 05FA             ;
21 05FA NLPHL: ENT
22 05FA CD0900      CALL  NL
23 05FD CDBA03      CALL  PRTHL
24 0600 C9          RET
25 0601             ;
26 0601             ;
27 0601             ; ORG 0601H;EDGE
28 0601             ;
29 0601             ;
30 0601             ; EDGE (TAPE DATA EDGE DETECT)
31 0601             ;
32 0601             ; BC=KEYPB ($E001)
33 0601             ; DE=CSTR ($E002)
34 0601             ; EXIT CF=0 OK : CF=1 BREAK
35 0601             ;
36 0601 EDGE: ENT
37 0601 3EF8        LD    A,F8H
38 0603 3200E0      LD    (KEYPA),A
39 0606 00          NOP
40 0607 EDG1: ENT
41 0607 0A          LD    A,(BC)
42 0608 E6B1        AND    B1H
43 060A 2002        JR    NZ,+4
44 060C 37          SCF
45 060D C9          RET
46 060E 1A          LD    A,(DE)
47 060F E620        AND    20H
48 0611 20F4        JR    NZ,EDG1
49 0613 EDG2: ENT
50 0613 0A          LD    A,(BC)
51 0614 E6B1        AND    B1H
52 0616 2002        JR    NZ,+4
53 0618 37          SCF
54 0619 C9          RET
55 061A 1A          LD    A,(DE)
56 061B E620        AND    20H
57 061D 2BF4        JR    Z,EDG2
58 061F C9          RET
59 0620             ;
60 0620             ;

```

\*\* Z80 ASSEMBLER SB-7201 &lt;1Z-013A&gt; PAGE 23

04.07.83

```

01 0620          DEFS  +4
02 0624          ;ORG 0624H;RBYTE
03 0624          ;
04 0624          ;
05 0624          ; 1 BYTE READ
06 0624          ;
07 0624          ; EXIT SUMDT=STORE
08 0624          ; CF=1 : BREAK
09 0624          ;
10 0624          ; CF=0 : DATA=ACC
11 0624          ;
12 0624          RBYTE: ENT
13 0624 C5        PUSH BC
14 0625 D5        PUSH DE
15 0626 E5        PUSH HL
16 0627 210008    LD HL,0800H
17 062A 0101E0    LD BC,KEYPB ; KEY DATA $E001
18 062D 1102E0    LD DE,CSTR ; $TAPE DATA $E002
19 0630          RBY1: ENT
20 0630 CD0106    CALL EDGE ; 41 OR 101
21 0633 DA5406    JP C,RBY3 ; 13
22 0636 CD4A0A    CALL DLY3 ; 20+18*63+33
23 0639 1A        LD A,(DE) ; DATA READ :8
24 063A E620      AND 20H
25 063C CA4906    JP Z,RBY2
26 063F E5        PUSH HL
27 0640 2A9711    LD HL,(SUMDT)
28 0643 23        INC HL
29 0644 229711    LD (SUMDT),HL
30 0647 E1        POP HL
31 0648 37        SCF
32 0649          RBY2: ENT
33 0649 7D        LD A,L
34 064A 17        RLA
35 064B 6F        LD L,A
36 064C 25        DEC H
37 064D C23006    JP NZ,RBY1
38 0650 CD0106    CALL EDGE
39 0653 7D        LD A,L
40 0654          RBY3: ENT
41 0654 E1        POP HL
42 0655 D1        POP DE
43 0656 C1        POP BC
44 0657 C9        RET
45 0658          ;
46 0658          ;
47 0658          ; TAPE MARK DETECT
48 0658          ;
49 0658          ; E=3L0 : INFORMATION
50 0658          ; =0S0 : DATA
51 0658          ; EXIT CF=0 : OK
52 0658          ; =1 :BREAK
53 0658          ;
54 0658          DEFS  +3
55 065B          ;
56 065B          TMARK: ENT
57 065B          ;
58 065B          ;ORG 065BH
59 065B          ;
60 065B CDE20F    CALL GAPCK

```

\*\* Z80 ASSEMBLER SB-7201 &lt;1Z-013A&gt; PAGE 24

04.07.83

```

01 065E C5        PUSH BC ; ORG 065EH
02 065F D5        PUSH DE
03 0660 E5        PUSH HL
04 0661 212828    LD HL,2828H
05 0664 7B        LD A,E
06 0665 FECC      CP CCH ; 'L'
07 0667 2803      JR Z,+5
08 0669 211414    LD HL,1414H
09 066C 229511    LD (TMCNT),HL
10 066F 0101E0    LD BC,KEYPB
11 0672 1102E0    LD DE,CSTR
12 0675          TM1: ENT
13 0675 2A9511    LD HL,(TMCNT)
14 0678          TM2: ENT
15 0678 CD0106    CALL EDGE
16 067B 381E      JR C,TM4
17 067D CD4A0A    CALL DLY3 ; CALL DLY2*3
18 0680 1A        LD A,(DE)
19 0681 E620      AND 20H
20 0683 28F0      JR Z,TM1
21 0685 25        DEC H
22 0686 20F0      JR NZ,TM2
23 0688          TM3: ENT
24 0688 CD0106    CALL EDGE
25 068B 380E      JR C,TM4
26 068D CD4A0A    CALL DLY3 ; CALL DLY2*3
27 0690 1A        LD A,(DE)
28 0691 E620      AND 20H
29 0693 20E0      JR NZ,TM1
30 0695 2D        DEC L
31 0696 20F0      JR NZ,TM3
32 0698 CD0106    CALL EDGE
33 069B          RET3: ENT
34 069B          TM4: ENT
35 069B E1        POP HL
36 069C D1        POP DE
37 069D C1        POP BC
38 069E C9        RET
39 069F          ;
40 069F          ;
41 069F          ; MOTOR ON
42 069F          ;
43 069F          ; IN D=0W0 :WRITE
44 069F          ; =0R0 :READ
45 069F          ; EXIT CF=0 :OK
46 069F          ; =1 :BREAK
47 069F          ;
48 069F C5        PUSH BC
49 06A0 D5        PUSH DE
50 06A1 E5        PUSH HL
51 06A2 060A      LD B,10
52 06A4          MOT1: ENT
53 06A4 3A02E0    LD A,(CSTR)
54 06A7 E610      AND 10H
55 06A9 280E      JR Z,MOT4
56 06AB          MOT2: ENT
57 06AB 06FF      LD B,FFH ; 2 SEC DELAY
58 06AD CD9609    CALL DLY12 ; 7 MSEC DELAY
59 06B0 1802      JR +4 ;MOTOR ENTRY ADJUST
60 06B2 18EB      JR MOTOR ; ORG 06B2H

```



```

01 06B4 10F7      DJNZ  -7
02 06B6 AF        XOR   A
03 06B7           MOT7: ENT
04 06B7 18E2      JR    RET3
05 06B9           MOT4: ENT
06 06B9 3E06      LD     A,06H
07 06BB 2103E0    LD     HL,CSTPT
08 06BE 77        LD     (HL),A
09 06BF 3C        INC    A
10 06C0 77        LD     (HL),A
11 06C1 10E1      DJNZ  MOT1
12 06C3 CD0900    CALL   NL
13 06C6 7A        LD     A,D
14 06C7 FED7      CP     D7H      ; 'W'
15 06C9 2805      JR     Z,MOT8
16 06CB 11FB03    LD     DE,MSG#1  ; PLAY MARK
17 06CE 1807      JR     MOT9
18 06D0           MOT8: ENT
19 06D0 110204    LD     DE,MSG#3  ; "RECORD."
20 06D3 DF        RST     3        ; CALL MSGX
21 06D4 11FD03    LD     DE,MSG#2  ; "PLAY"
22 06D7           MOT9: ENT
23 06D7 DF        RST     3        ; CALL MSGX
24 06D8           MOT5: ENT
25 06D8 3A02E0    LD     A,(CSTR)
26 06DB E610      AND     10H
27 06DD 20CC      JR     NZ,MOT2
28 06DF CD320A    CALL   ?BRK
29 06E2 20F4      JR     NZ,MOT5
30 06E4 37        SCF
31 06E5 18D0      JR     MOT7
32 06E7           ;
33 06E7           ; INITIAL MESSAGE
34 06E7           ;
35 06E7           MSG#3: ENT
36 06E7 2A2A2020  DEFM    '*** MONITOR 1Z-013A ***'
37 06EB 4D4F4E49
38 06EF 544F5220
39 06F3 315A2D30
40 06F7 31334120
41 06FB 202A2A
42 06FE 0D        DEFB    0DH
43 06FF           ;
44 06FF           ;
45 06FF           DEFS    +1
46 0700           ;
47 0700           ;
48 0700           ; ORG 0700H;MSTOP
49 0700           ;
50 0700           ;
51 0700           ; MOTOR STOP
52 0700           ;
53 0700           MSTOP: ENT
54 0700 F5        PUSH   AF
55 0701 C5        PUSH   BC
56 0702 D5        PUSH   DE
57 0703 060A      LD     B,10
58 0705           MST1: ENT
59 0705 3A02E0    LD     A,(CSTR)
60 0708 E610      AND     10H

```

```

01 070A 280B      JR     Z,MST3
02 070C           MST2: ENT
03 070C 3E06      LD     A,06H
04 070E 3203E0    LD     (CSTPT),A
05 0711 3C        INC    A
06 0712 3203E0    LD     (CSTPT),A
07 0715 10EE      DJNZ  MST1
08 0717           MST3: ENT
09 0717 C3E60E    JP     ?RSTR1
10 071A           ;
11 071A           ;
12 071A           ;
13 071A           ; CHECK SUM
14 071A           ;
15 071A           ; IN BC=SIZE
16 071A           ; HL=DATA ADR.
17 071A           ; EXIT SUMDT=STORE
18 071A           ; CSMDDT=STORE
19 071A           ;
20 071A           ;
21 071A           CKSUM: ENT
22 071A C5        PUSH   BC
23 071B D5        PUSH   DE
24 071C E5        PUSH   HL
25 071D 110000    LD     DE,0
26 0720           CKS1: ENT
27 0720 78        LD     A,B
28 0721 B1        OR     C
29 0722 200B      JR     NZ,CKS2
30 0724 EB        EX     DE,HL
31 0725 229711    LD     (SUMDT),HL
32 0728 229911    LD     (CSMDDT),HL
33 072B E1        POP    HL
34 072C D1        POP    DE
35 072D C1        POP    BC
36 072E C9        RET
37 072F           CKS2: ENT
38 072F 7E        LD     A,(HL)
39 0730 C5        PUSH   BC
40 0731 0608      LD     B,+8
41 0733           CKS3: ENT
42 0733 07        RLCA
43 0734 3001      JR     NC,+3
44 0736 13        INC    DE
45 0737 10FA      DJNZ  CKS3
46 0739 C1        POP    BC
47 073A 23        INC    HL
48 073B 0B        DEC    BC
49 073C 18E2      JR     CKS1
50 073E           ;
51 073E           ; MODE SET OF KEYPORT
52 073E           ;
53 073E           ;
54 073E 2103E0    LD     HL,KEYPF
55 0741 36BA      LD     (HL),BAH      ; 10001010
56 0743 3607      LD     (HL),07H      ; PC3=1
57 0745 3605      LD     (HL),05H      ; PC2=1
58 0747           VGOFF: ENT
59 0747           ;
60 0747 C9        RET

```

\*\* Z80 ASSEMBLER SB-7201 &lt;1Z-013A&gt; PAGE 27

04.07.83

```

01 0748 ;
02 0748 ;
03 0748     DEFS    +17
04 0759 ;
05 0759 ;ORG 0759H;DLY1
06 0759 ;
07 0759 ;    107 MICRO SEC DELY
08 0759 ;
09 0759 DLY1:  ENT
10 0759 3E15    LD    A,15H    ; 18*21+20
11 075B 3D      DEC    A
12 075C C25B07  JP     NZ,-1
13 075F C9      RET
14 0760 ;
15 0760 ;ORG 0760H;DLY2
16 0760 ;
17 0760 DLY2:  ENT
18 0760 3E13    LD    A,13H    ; 18*19+20
19 0762 3D      DEC    A
20 0763 C26207  JP     NZ,-1
21 0766 C9      RET
22 0767 ;
23 0767 ;
24 0767 ;
25 0767 ;
26 0767 ;
27 0767 ;    1 BYTE WRITE
28 0767 ;
29 0767 WBYTE: ENT
30 0767 C5      PUSH   BC
31 076B 060B    LD     B,+8
32 076A CD1A0A  CALL    LONG
33 076D         ENT
34 076D 07      RLCA
35 076E DC1A0A  CALL    C, LONG
36 0771 D4010A  CALL    NC, SHORT
37 0774 05      DEC     B
38 0775 C26D07  JP     NZ, WBY1
39 077B C1      POP     BC
40 0779 C9      RET
41 077A ;
42 077A ;
43 077A ;    GAP + TAPEMARK
44 077A ;
45 077A ;    E=0L0 LONG GAP
46 077A ;    =0S0 SHORT GAP
47 077A ;
48 077A GAP:    ENT
49 077A C5      PUSH   BC
50 077B D5      PUSH   DE
51 077C 7B      LD     A,E
52 077D 01F055  LD     BC,55F0H
53 0780 112B2B  LD     DE,2B2BH
54 0783 FECC    CP     CCH
55 0785 CABE07  JP     Z, GAP1
56 078B 01F82A  LD     BC,2AF8H
57 078B 111414  LD     DE,1414H
58 078E         ENT
59 078E CD010A  CALL    SHORT
60 0791 0B      DEC     BC

```

\*\* Z80 ASSEMBLER SB-7201 &lt;1Z-013A&gt; PAGE 28

04.07.83

```

01 0792 '78      LD     A,B
02 0793 B1       OR     C
03 0794 20F8     JR     NZ,-6
04 0796         ENT
05 0796 CD1A0A  CALL    LONG
06 0799 15      DEC     D
07 079A 20FA     JR     NZ,-4
08 079C         ENT
09 079C CD010A  CALL    SHORT
10 079F 1D      DEC     E
11 07A0 20FA     JR     NZ,-4
12 07A2 CD1A0A  CALL    LONG
13 07A5 D1      POP     DE
14 07A6 C1      POP     BC
15 07A7 C9      RET
16 07AB ;
17 07AB ;    MEMORY CORRECTION
18 07AB ;    COMMAND 'M'
19 07AB ;
20 07AB MCR0:  ENT
21 07AB CD3D01  CALL    HEXIV    ; CRRECTION ADR.
22 07AB MCR1:  ENT
23 07AB CDFA05  CALL    NLPHL    ; COR. ADR. PRINT
24 07AE CDB103  CALL    SPHEX    ; ACC ← ASCII DISP.
25 07B1 CD2009  CALL    ?PRTS    ; SPACE PRINT
26 07B4 CD2F01  CALL    BGETL    ; GET DATA & CHECK DATA
27 07B7 CD1004  CALL    HLHEX    ; HL←ASCII(DE)
28 07BA 3B1B     JR     C,MCR3
29 07BC CDA602  CALL    .4DE    ; (INC DE)*4
30 07BF 13      INC     DE
31 07C0 CD1F04  CALL    2HEX    ; DATA CHECK
32 07C3 3BE6     JR     C,MCR1
33 07C5 BE      CP     (HL)
34 07C6 20E3     JR     NZ,MCR1
35 07C8 13      INC     DE
36 07C9 1A      LD     A,(DE)
37 07CA FE0D     CP     0DH    ; NOT CORRECTION ?
38 07CC 2B06     JR     Z,MCR2
39 07CE CD1F04  CALL    2HEX    ; ACC←HL(ASCII)
40 07D1 3BD8     JR     C,MCR1
41 07D3 77      LD     (HL),A    ; DATA CORRECT
42 07D4 MCR2:  ENT
43 07D4 23      INC     HL
44 07D5 1BD4     JR     MCR1
45 07D7 ;
46 07D7 60      LD     H,B
47 07D8 69      LD     L,C
48 07D9 1BD0     JR     MCR1
49 07DB ;
50 07DB ;
51 07DB ;
52 07DB ;
53 07DB ;
54 07E4         ORG     07E6H
55 07E6 ;
56 07E6 ;
57 07E6 ;
58 07E6 ;
59 07E6 ;    GET 1 LINE STATEMENT *
60 07E6 ;    DE = DATA STORE LOW ADR.

```

```

01 07E6      ;      (END =CR )
02 07E6      ;
03 07E6      ;
04 07E6      ?GETL: ENT
05 07E6 F5    PUSH  AF
06 07E7 C5    PUSH  BC
07 07E8 E5    PUSH  HL
08 07E9 D5    PUSH  DE
09 07EA      GETL1: ENT
10 07EA CDB309 CALL  ??KEY      ; ENTRY KEY
11 07ED      AUTO3: ENT
12 07ED F5    PUSH  AF      ; IN KEY DATA SAVE
13 07EE 47    LD     B,A
14 07EF 3A9D11 LD     A,(SWRK)      ; BELL WORK
15 07F2 0F    RRCA
16 07F3 D47705 CALL  NC,?BEL      ; ENTRY BELL
17 07F6 78    LD     A,B
18 07F7 217011 LD     HL,KANAF      ; KANA & GRAPH FLAG
19 07FA E6F0   AND    FOH
20 07FC FEC0   CP     COH
21 07FE D1    POP    DE      ; Ereg=FLAGreg
22 07FF 78    LD     A,B
23 0800 2016   JR     NZ,GETL2
24 0802 FEC0   CP     CDH      ; CR
25 0804 2855   JR     Z,GETL3
26 0806 FECB   CP     CBH      ; BREAK
27 0808 CA2208 JP     Z,GETLC
28 080B FECF   CP     CFH      ; NIKO MARK WH.
29 080D 2809   JR     Z,GETL2
30 080F FEC7   CP     C7H      ; CRT EDITION
31 0811 300A   JR     NC,GETL5
32 0813 CB1B   RR     E      ; CY ?
33 0815 78    LD     A,B
34 0816 3005   JR     NC,GETL5
35 0818      GETL2: ENT
36 0818 CDB50D CALL  ?DSP
37 081B 18CD   JR     GETL1
38 081D      GETL5: ENT
39 081D CDDC0D CALL  ?DPCT      ; CRT CONTROL
40 0820 18CB   JR     GETL1
41 0822      ;
42 0822      ; BREAK IN
43 0822      ;
44 0822 E1    GETLC: POP    HL
45 0823 E5    PUSH    HL
46 0824 361B   LD     (HL),1BH      ; BREAK CODE
47 0826 23    INC     HL
48 0827 360D   LD     (HL),0DH
49 0829 1853   JR     GETLR
50 082B      ; GETLA
51 082B      ;
52 082B 0F    GETLA: RRCA
53 082C 3037   JR     NC,GETL6
54 082E 1833   JR     GETLB
55 0830      ;
56 0830      ;
57 0830      ;
58 0830      ; DELAY 7M SEC AND SWEP
59 0830      ;
60 0830 CD9609 DSWEP: CALL  DLY12

```

```

01 0833 CD500A CALL  ?SWEP
02 0836 C9    RET
03 0837      ;
04 0837      ;
05 0837      ; DEFS 36
06 085B      ;
07 085B      ;
08 085B      ;
09 085B      ; ORG 085BH;GETL3
10 085B      ;
11 085B CDF302 GETL3: CALL  .MANG      ; CR
12 085E 0628   LD     B,40      ; 1LINE
13 0860 30C9   JR     NC,GETLA
14 0862 25    DEC     H      ; BEFORE LINE
15 0863 0650   LD     B,80      ; 2 LINE
16 0865 2E00   LD     L,0
17 0867 CDB40F CALL  ?PNT1
18 086A D1    POP    DE      ; STORE TOP ADR.
19 086B D5    PUSH    DE
20 086C 7E    LD     A,(HL)
21 086D CDCE0B CALL  ?DACN
22 0870 12    LD     (DE),A
23 0871 23    INC     HL
24 0872 13    INC     DE
25 0873 10F7   DJNZ   GETLZ
26 0875 EB    EX      DE,HL
27 0876 360D   LD     (HL),0DH
28 0878 2B    DEC     HL
29 0879 7E    LD     A,(HL)
30 087A FE20   CP     20H      ; SPACE THEN CR
31 087C      ;
32 087C      ;
33 087C      ; CR AND NEW LINE
34 087C      ;
35 087C 28F8   JR     Z,GETLU
36 087E      ;
37 087E      ; NEW LINE RETURN
38 087E      ;
39 087E CD0E09 GETLR: CALL  ?LTNL
40 0881 D1    POP    DE
41 0882 E1    POP    HL
42 0883 C1    POP    BC
43 0884 F1    POP    AF
44 0885 C9    RET
45 0886      ;
46 0886      ;
47 0886      ; DEFS +13
48 0886      ; ORG 0893H
49 0893      ;
50 0893      ; MESSAGE PRINT
51 0893      ;
52 0893      ; DE PRINT DATA LOW ADR.
53 0893      ; END=CR
54 0893      ;
55 0893      ;
56 0893      ; MSG: ENT
57 0893 F5    PUSH    AF
58 0894 C5    PUSH    BC
59 0895 D5    PUSH    DE
60 0896 1A    LD     A,(DE)

```

\*\* Z80 ASSEMBLER SB-7201 &lt;1Z-013A&gt; PAGE 31

04.07.83

```

01 0897 FE0D          CP    0DH          ; CR
02 0899 280C          JR    Z,MSGX2
03 089B CD3509        CALL ?PRNT
04 089E 13           INC    DE
05 089F 18F5          JR    MSG1
06 08A1              ;
07 08A1              ;
08 08A1              ;ORG 08A1H
09 08A1              ;
10 08A1              ; ALL PRINT MESSAGE
11 08A1              ;
12 08A1              ?MSGX: ENT
13 08A1 F5           PUSH  AF
14 08A2 C5           PUSH  BC
15 08A3 D5           PUSH  DE
16 08A4 1A          MSGX1: LD    A,(DE)
17 08A5 FE0D          CP    0DH
18 08A7 CAE60E        MSGX2: JP    Z,?RSTR1
19 08AA CDB90B        CALL  ?ADCN
20 08AD CD6C09        CALL  PRNT3
21 08B0 13           INC    DE
22 08B1 18F1          JR    MSGX1
23 08B3              ;
24 08B3              ; TOP OF KEYTBLS
25 08B3              ;
26 08B3 112A0C        ?KYSM: LD    DE,KTBL
27 08B6 1842          JR    ?KYS
28 08B8              ;
29 08B8              ; BREAK CODE IN
30 08B8              ;
31 08B8 3ECB          #BRK: LD    A,CBH
32 08BA B7           OR     A
33 08BB 1819          JR    ?KY1
34 08BD              ;
35 08BD              ;
36 08BD              ;ORG 08BDH
37 08BD              ;
38 08BD              ; GETKEY
39 08BD              ;
40 08BD              ; NOT ECHO BACK
41 08BD              ;
42 08BD              ; EXIT:ACC=ASCII CODE
43 08BD              ;
44 08BD              ?GET: ENT
45 08BD CDCA08        CALL  ?KEY
46 08C0 D6F0          SUB    FOH
47 08C2 C8           RET     Z
48 08C3 C6F0          ADD    A,FOH
49 08C5 C3CE0B        JP     ?DACN
50 08C8              ;
51 08C8              ;
52 08C8              ; DEFS +2
53 08CA              ;
54 08CA              ;
55 08CA              ;
56 08CA              ;
57 08CA              ;ORG 08CAH;?KEY
58 08CA              ;
59 08CA              ; 1KEY INPUT
60 08CA              ; IN      B = KEY MODE(SHIFT,CTRL,BREAK)

```

\*\* Z80 ASSEMBLER SB-7201 &lt;1Z-013A&gt; PAGE 32

04.07.83

```

01 08CA              ; C = KEY DATA (COLUMN & ROW)
02 08CA              ; EXIT  ACC=DISPLAY CODE
03 08CA              ; IF NO KEY  ACC=FOH
04 08CA              ; IF CY=1 THEN ATTRIBUTE ON
05 08CA              ; (SMALL,HIRAKANA)
06 08CA              ;
07 08CA              ?KEY: ENT
08 08CA C5           PUSH  BC
09 08CB D5           PUSH  DE
10 08CC E5           PUSH  HL
11 08CD CD300B        CALL  DSWEF
12 08D0 78           LD     A,B
13 08D1 07           RLCA
14 08D2 3806          JR     C,?KY2
15 08D4 3EF0          LD     A,FOH
16 08D6              ?KY1: ENT
17 08D6 E1           POP    HL
18 08D7 D1           POP    DE
19 08D8 C1           POP    BC
20 08D9 C9           RET
21 08DA              ;
22 08DA              ?KY2: ENT
23 08DA 11EA0B        LD     DE,KTBL
24 08DD 78           LD     A,B
25 08DE FE98          CP     98H
26 08E0 28D6          JR     Z,#BRK
27 08E2 2600          LD     H,0
28 08E4 69           LD     L,C
29 08E5 CB6F          BIT     5,A
30 08E7 200E          JR     NZ,?KYS-3
31 08E9 3A7011        LD     A,(KANAF)
32 08EC 0F           RRCA
33 08ED DAFE08        JP     C,?KYGRP
34 08F0 78           LD     A,B
35 08F1 17           RLA
36 08F2 17           RLA
37 08F3 38BE          JR     C,?KYSM
38 08F5 1803          JR     ?KYS
39 08F7 11AA0C        LD     DE,KTBL
40 08FA              ?KYS: ENT
41 08FA 19           ADD     HL,DE
42 08FB              ?KY55: ENT
43 08FB 7E           LD     A,(HL)
44 08FC 18D8          JR     ?KY1
45 08FE              ?KYGRP: ENT
46 08FE CB70          BIT     6,B
47 0900 2807          JR     Z,?KYGRS
48 0902 11E90C        LD     DE,KTBL
49 0905 19           ADD     HL,DE
50 0906 37           SCF
51 0907 18F2          JR     ?KY55
52 0909              ;
53 0909 116A0C        ?KYGRS: LD  DE,KTBL
54 090C 18EC          JR     ?KYS
55 090E              ;
56 090E              ;
57 090E              ;
58 090E              ;
59 090E              ;
60 090E              ;ORG 090EH

```

```

01 090E      ;
02 090E      ; NEWLINE
03 090E      ;
04 090E      ;LTNL: ENT
05 090E AF    XDR      A
06 090F 329411 LD      (DPRNT),A      ; ROW POINTER
07 0912 3ECD   LD      A,CDH      ; CR
08 0914 1843   JR      PRNT5
09 0916       DEFS     +2
10 0918      ;ORG 0918H
11 0918      ;
12 0918      ;NL: ENT
13 0918 3A9411 LD      A,(DPRNT)
14 091B B7     OR      A
15 091C C8     RET     Z
16 091D 18EF   JR      ?LTNL
17 091F       DEFS     +1
18 0920      ;ORG 0920H
19 0920      ;
20 0920      ; PRINT SPACE
21 0920      ;
22 0920      ;PRTS: ENT
23 0920 3E20   LD      A,20H
24 0922 1811   JR      ?PRNT
25 0924      ;
26 0924      ; PRINT TAB
27 0924      ;
28 0924      ;PRTT: ENT
29 0924 CD0C00 CALL     PRNTS
30 0927 3A9411 LD      A,(DPRNT)
31 092A B7     OR      A
32 092B C8     RET     Z
33 092C D60A   SUB     +10
34 092E 38F4   JR      C,-10
35 0930 20FA   JR      NZ,-4
36 0932       DEFS     +3
37 0935      ;ORG 0935H
38 0935      ;
39 0935      ; PRINT
40 0935      ;
41 0935      ; IN ACC = PRINT DATA (ASCII)
42 0935      ;
43 0935      ;PRNT: ENT
44 0935 FE0D   CP      0DH      ; CR
45 0937 28D5   JR      Z,?LTNL
46 0939 C5     PUSH    BC
47 093A 4F     LD      C,A
48 093B 47     LD      B,A
49 093C CD4609 CALL     ?PRT
50 093F 78     LD      A,B
51 0940 C1     POP     BC
52 0941 C9     RET
53 0942      ;
54 0942      ;
55 0942      ;MSGOK: ENT
56 0942 4F4B21 DEFM     'OK!'
57 0945 0D     DEFB     0DH
58 0946      ;ORG 0946H
59 0946      ;
60 0946      ; PRINT ROUTINE

```

```

01 0946      ; 1 CHA.
02 0946      ; INPUT:C=ASCII DATA (?DSP+?DPCT)
03 0946      ;
04 0946      ;PRT: ENT
05 0946 79     LD      A,C
06 0947 CDB90B CALL     ?ADCN      ; ASCII TO DISPLAY
07 094A 4F     LD      C,A
08 094B FEFO   CP      FOH
09 094D C8     RET     Z      ; ZERO=ILLEGAL DATA
10 094E E6F0   AND     FOH      ; MSD CHECK
11 0950 FEC0   CP      COH
12 0952 79     LD      A,C
13 0953 2017   JR      NZ,PRNT3
14 0955 FEC7   CP      C7H
15 0957 3013   JR      NC,PRNT3      ; CRT EDITOR
16 0959      ;PRNT5: ENT
17 0959 CDDC0D CALL     ?DPCT
18 095C FEC3   CP      C3H
19 095E 280F   JR      Z,PRNT4
20 0960 FEC5   CP      C5H      ; HOME
21 0962 2803   JR      Z,PRNT2
22 0964 FEC6   CP      C6H      ; CLR
23 0966 C0     RET     NZ
24 0967 AF     XOR     A
25 0968 329411 LD      (DPRNT),A
26 096B C9     RET
27 096C      ;PRNT3: ENT
28 096C CDB50D CALL     ?DSP
29 096F 3A9411 LD      A,(DPRNT)      ; TAB POINT+1
30 0972 3C     INC     A
31 0973 FE50   CP      +80
32 0975 38F1   JR      C,PRNT2+1
33 0977 D650   SUB     +80
34 0979 18ED   JR      PRNT2+1
35 097B      ;
36 097B      ;
37 097B      ;
38 097B      ;
39 097B      ;
40 097B      ; FLASSING BYPASS 1
41 097B      ;
42 097B      ;
43 097B 3ABE11 LD      A,(FLASH)
44 097E 186F   JR      FLAS2
45 0980      ;
46 0980      ; BREAK SUBROUTINE BYPASS 1
47 0980      ;
48 0980      ; CTRL OR NOT KEY
49 0980      ;
50 0980      ;BRK2: ENT
51 0980 CB6F   BIT     S,A      ; NOT OR CTRL
52 0982 2802   JR      Z,?BRK3      ; CTRL
53 0984 B7     OR      A      ; NOTKEY A=7FH
54 0985 C9     RET
55 0986      ;
56 0986 3E20   LD      A,20H      ; CTRL D5=1
57 0988 B7     OR      A      ; ZERO FLG. CLR
58 0989 37     SCF
59 098A C9     RET
60 098B      ;

```

\*\* Z80 ASSEMBLER SB-7201 &lt;1Z-013A&gt; PAGE 35

04.07.83

```

01 098B MSGSV: ENT
02 098B 46494C45 DEFM 'FILENAME? '
03 098F 4E414D45
04 0993 3F20
05 0995 0D DEFB 0DH
06 0996 ;
07 0996 ; DLY 7 MSEC
08 0996 ;
09 0996 DLY12: ENT
10 0996 C5 PUSH BC
11 0997 0615 LD B,15H
12 0999 CD4A0A CALL DLY3
13 099C 10FB DJNZ -3
14 099E C1 POP BC
15 099F C9 RET
16 09A0 ;
17 09A0 ;
18 09A0 ;
19 09A0 ; LOADING MESSAGE
20 09A0 ;
21 09A0 MSG?2: ENT
22 09A0 4C4F4144 DEFM 'LOADING '
23 09A4 494E4720
24 09A8 0D DEFB 0DH
25 09A9 ;
26 09A9 ;
27 09A9 ;
28 09A9 ; DELAY FOR LONG PULSE
29 09A9 ;
30 09A9 DLY4: ENT
31 09A9 3E59 LD A,59H ; 18*89+20
32 09AB 3D DEC A
33 09AC C2AB09 JP NZ,-1
34 09AF C9 RET
35 09B0 ;
36 09B0 ;
37 09B0 DEFS +3
38 09B3 ;
39 09B3 ;
40 09B3 ; ORG 09B3H;??KEY
41 09B3 ;
42 09B3 ; KEY BOAD SEARCH
43 09B3 ; & DISPLAY CODE CONV.
44 09B3 ;
45 09B3 ; EXIT A = DISPLAY CODE
46 09B3 ; CY= GRAPH MODE
47 09B3 ; WITH CURSOR DISPLAY
48 09B3 ;
49 09B3 ??KEY: ENT
50 09B3 E5 PUSH HL
51 09B4 CD920B CALL ?SAVE
52 09B7 ;
53 09B7 CD7E05 KSL1: ENT
54 09BA 20FB CALL FLKEY ; KEY
55 09BC ENT JR NZ,KSL1 ; KEY IN THEN JUMP
56 09BC CD7E05 KSL2: ENT
57 09BF 28FB CALL FLKEY ; NOT KEY IN THEN JUMP
58 09C1 67 LD H,A
59 09C2 CD9609 CALL DLY12 ; DELAY CHATTER
60 09C5 CDCA08 CALL ?KEY

```

\*\* Z80 ASSEMBLER SB-7201 &lt;1Z-013A&gt; PAGE 36

04.07.83

```

01 09CB F5 PUSH AF
02 09C9 BC CP H ; CHATER CHECK
03 09CA E1 POP HL
04 09CB 20EF JR NZ,KSL2
05 09CD E5 PUSH HL
06 09CE F1 POP AF ; IN KEY DATA
07 09CF CDF005 CALL ?LOAD ; FLSHING DATA LOAD
08 09D2 E1 POP HL
09 09D3 C9 RET
10 09D4 ;
11 09D4 ;
12 09D4 ; CLEAR 2
13 09D4 ;
14 09D4 #CLR08: ENT
15 09D4 AF XOR A ; CY FLG.
16 09D5 #CLR8: ENT
17 09D5 01000B LD BC,0800H
18 09D8 CLEAR: ENT ; BC = CLR BYTE SIZE
19 09D8 D5 PUSH DE ; A = CLR DATA
20 09D9 57 LD D,A
21 09DA CLEAR1: ENT
22 09DA 72 LD (HL),D
23 09DB 23 INC HL
24 09DC 0B DEC BC
25 09DD 78 LD A,B
26 09DE B1 OR C
27 09DF 20F9 JR NZ,CLEAR1
28 09E1 D1 POP DE
29 09E2 C9 RET
30 09E3 ;
31 09E3 ;
32 09E3 ;
33 09E3 ;
34 09E3 ; FLASHING 2
35 09E3 ;
36 09E3 ?FLS: ENT
37 09E3 F5 PUSH AF
38 09E4 E5 PUSH HL
39 09E5 3A02E0 LD A,(KEYPC)
40 09E8 07 RLCA
41 09E9 07 RLCA
42 09EA 388F JR C,FLAS1
43 09EC 3A9211 LD A,(FLSDT)
44 09EF FLAS2: ENT
45 09EF CDB10F CALL ?FONT ; DISPLAY POSITION
46 09F2 77 LD (HL),A
47 09F3 FLAS3: ENT
48 09F3 E1 POP HL
49 09F4 F1 POP AF
50 09F5 C9 RET
51 09F6 ;
52 09F6 ;
53 09F6 ;
54 09F6 DEFS +9
55 09FF ;
56 09FF ; ORG 09FF ; ?FLAS
57 09FF ;
58 09FF ?FLAS: ENT
59 09FF JR ?FLS
60 09FF 18E2

```

```

01 0A01      ;
02 0A01      ;
03 0A01      ;
04 0A01      ; SHORT AND LONG PULSE FOR 1 BIT WRITE
05 0A01      ;
06 0A01      ;
07 0A01 F5   SHORT: ENT
08 0A02 3E03  PUSH    AF          ; 12
09 0A04 3203E0 LD      A,03H        ; 9
10 0A07 CD5907 CALL    (CSTPT),A      ; $E003 PC3=1:16
11 0A0A CD5907 CALL    DLY1       ; 20+18*21+20
12 0A0D 3E02  LD      A,02H        ; 9
13 0A0F 3203E0 LD      (CSTPT),A    ; $E003 PC3=0:16
14 0A12 CD5907 CALL    DLY1       ; 20+18*21+20
15 0A15 CD5907 CALL    DLY1       ; 20+18*21+20
16 0A18 F1    POP     AF          ; 11
17 0A19 C9    RET          ; 11
18 0A1A      ;
19 0A1A      ;
20 0A1A      ;
21 0A1A F5   LONG: ENT
22 0A1B 3E03  PUSH    AF          ; 11
23 0A1D 3203E0 LD      A,03H        ; 9
24 0A20 CDA909 CALL    (CSTPT),A      ; 16
25 0A23 3E02  LD      A,02H        ; 9
26 0A25 3203E0 LD      (CSTPT),A    ; 16
27 0A28 CDA909 CALL    DLY4       ; 20+18*89+20
28 0A2B F1    POP     AF          ; 11
29 0A2C C9    RET          ; 11
30 0A2D      ;
31 0A2D      ;
32 0A2D      ; DEFS +5
33 0A32      ;
34 0A32      ;
35 0A32      ;
36 0A32      ;
37 0A32      ; BREAK KEY CHECK
38 0A32      ; AND SHIFT,CTNL KEY CHECK
39 0A32      ;
40 0A32      ; EXIT BREAK ON : ZERO=1
41 0A32      ; OFF: ZERO=0
42 0A32      ; NO KEY : CY =0
43 0A32      ; KEY IN : CY =1
44 0A32      ; A D6=1 : SHIFT ON
45 0A32      ; =0 : OFF
46 0A32      ; D5=1 : CTRL ON
47 0A32      ; =0 : OFF
48 0A32      ; D4=1 : SFT+CNT ON
49 0A32      ; =0 : OFF
50 0A32      ;
51 0A32      ;
52 0A32 3EF8  ?BRK: ENT
53 0A34 3200E0 LD      A,F8H        ; LINE BSWEPT
54 0A37 00    LD      (KEYPA),A
55 0A38 3A01E0 LD      A,(KEYPB)
56 0A3B B7    OR      A
57 0A3C 1F    RRA
58 0A3D DA8009 JP      C,?BRK2     ; SHIFT ?
59 0A40 17    RLA
60 0A41 17    RLA

```

```

01 0A42 3004      JR      NC,?BRK1  ; BREAK ?
02 0A44 3E40      LD      A,40H     ; SHIFT D6=1
03 0A46 37        SCF
04 0A47 C9        RET
05 0A48          ;
06 0A48          ;
07 0A48 AF       ?BRK1: XOR      A    ; SHIFT ?
08 0A49 C9        RET
09 0A4A          ;
10 0A4A          ;
11 0A4A          ; 320 U SEC DELAY
12 0A4A          ;
13 0A4A          ;
14 0A4A 3E3F      DLY3: ENT
15 0A4C C36207    LD      A,3FH     ; 18*63+33
16 0A4F          JP      0762H      ; JP DLY2+2
17 0A4F          ;
18 0A4F          ; DEFS +1
19 0A50          ;
20 0A50          ;
21 0A50          ;
22 0A50          ; ORG 0A50H : ?SWEP
23 0A50          ;
24 0A50          ;
25 0A50          ; KEY BOAD SWEEP
26 0A50          ;
27 0A50          ; EXIT B,D7=0 NO DATA
28 0A50          ; =1 DATA
29 0A50          ; D6=0 SHIFT OFF
30 0A50          ; =1 SHIFT ON
31 0A50          ; D5=0 CTRL OFF
32 0A50          ; =1 CTRL ON
33 0A50          ; D4=0 SHIFT+CTRL OFF
34 0A50          ; =1 SHIFT+CTRL ON
35 0A50          ; C = ROW & COLOUMN
36 0A50          ; 7 6 5 4 3 2 1 0
37 0A50          ; * * * * *
38 0A50          ;
39 0A50          ;
40 0A50 D5        ?SWEP: ENT
41 0A51 E5        PUSH    DE
42 0A52 AF        PUSH    HL
43 0A53 06F8      XOR      A
44 0A55 57        LD      B,F8H
45 0A56 CD320A    LD      D,A
46 0A59 2004      CALL    ?BRK
47 0A5B 1688      JR      NZ,SWEP6
48 0A5D 1814      LD      D,88H
49 0A5F          JR      SWEP9
50 0A5F 3005      SWEP6: ENT
51 0A61 57        JR      NC,SWEP0
52 0A62 1802      LD      D,A
53 0A64          JR      SWEP0
54 0A64 CBFA      SWEP01: ENT
55 0A66          SET      7,D
56 0A66 05        SWEP0: ENT
57 0A67 78        DEC      B
58 0A68 3200E0    LD      A,B
59 0A6B FEEF      LD      (KEYPA),A
60 0A6D 200B      CP      EFH

```



\*\* Z80 ASSEMBLER SB-7201 &lt;1Z-013A&gt; PAGE 39

04.07.83

```

01 0A6F FEF8      CP      FBH      ; BREAK KEY ROW
02 0A71 28F3      JR      Z,SWEP0
03 0A73           SWEP9: ENT
04 0A73 42        LD      B,B
05 0A74 E1        POP     HL
06 0A75 D1        POP     DE
07 0A76 C9        RET
08 0A77           ;
09 0A77           SWEP3: ENT
10 0A77 3A01E0     LD      A,(KEYPB)
11 0A7A 2F        CPL
12 0A7B B7        OR      A
13 0A7C 28EB      JR      Z,SWEP0
14 0A7E 5F        LD      E,A
15 0A7F           SWEP2: ENT
16 0A7F 2608      LD      H,B
17 0A81 78        LD      A,B
18 0A82 E60F      AND     0FH
19 0A84 07        RLCA
20 0A85 07        RLCA
21 0A86 07        RLCA
22 0A87 4F        LD      C,A
23 0A88 7B        LD      A,E
24 0A89 25        DEC     H
25 0A8A 0F        RRCA
26 0A8B 30FC      JR      NC,-2
27 0A8D 7C        LD      A,H
28 0A8E 81        ADD     A,C
29 0A8F 4F        LD      C,A
30 0A90 18D2      JR      SWEP01
31 0A92           ;
32 0A92           ;
33 0A92           ; ASCII TO DISPLAY CODE TABL ;
34 0A92           ;
35 0A92           ATBL:
36 0A92           ; 00 - 0F ;
37 0A92 F0        DEFB    F0H      ; ↑@
38 0A93 F0        DEFB    F0H      ; ↑A
39 0A94 F0        DEFB    F0H      ; ↑B
40 0A95 F3        DEFB    F3H      ; ↑C
41 0A96 F0        DEFB    F0H      ; ↑D
42 0A97 F5        DEFB    F5H      ; ↑E
43 0A98 F0        DEFB    F0H      ; ↑F
44 0A99 F0        DEFB    F0H      ; ↑G
45 0A9A F0        DEFB    F0H      ; ↑H
46 0A9B F0        DEFB    F0H      ; ↑I
47 0A9C F0        DEFB    F0H      ; ↑J
48 0A9D F0        DEFB    F0H      ; ↑K
49 0A9E F0        DEFB    F0H      ; ↑L
50 0A9F F0        DEFB    F0H      ; ↑M
51 0AA0 F0        DEFB    F0H      ; ↑N
52 0AA1 F0        DEFB    F0H      ; ↑O
53 0AA2           ; 10 - 1F
54 0AA2 F0        DEFB    F0H      ; ↑P
55 0AA3 C1        DEFB    C1H      ; ↑Q CUR. DOWN
56 0AA4 C2        DEFB    C2H      ; ↑R CUR. UP
57 0AA5 C3        DEFB    C3H      ; ↑S CUR. RIGHT
58 0AA6 C4        DEFB    C4H      ; ↑T CUR. LEFT
59 0AA7 C5        DEFB    C5H      ; ↑U HOME
60 0AA8 C6        DEFB    C6H      ; ↑V CLEAR

```

\*\* Z80 ASSEMBLER SB-7201 &lt;1Z-013A&gt; PAGE 40

04.07.83

```

01 0AA9 F0        DEFB    F0H      ; ↑W
02 0AAA F0        DEFB    F0H      ; ↑X
03 0AAB F0        DEFB    F0H      ; ↑Y
04 0AAC F0        DEFB    F0H      ; ↑Z SEP.
05 0AAD F0        DEFB    F0H      ; ↑[
06 0AAE F0        DEFB    F0H      ; ↑\
07 0AAF F0        DEFB    F0H      ; ↑]
08 0AB0 F0        DEFB    F0H      ; ↑^
09 0AB1 F0        DEFB    F0H      ; ↑_
10 0AB2           ; 20 - 2F ;
11 0AB2 00        DEFB    00H      ; SPACE
12 0AB3 61        DEFB    61H      ; !
13 0AB4 62        DEFB    62H      ; "
14 0AB5 63        DEFB    63H      ; #
15 0AB6 64        DEFB    64H      ; $
16 0AB7 65        DEFB    65H      ; %
17 0AB8 66        DEFB    66H      ; &
18 0AB9 67        DEFB    67H      ; '
19 0ABA 68        DEFB    68H      ; (
20 0ABB 69        DEFB    69H      ; )
21 0ABC 6B        DEFB    6BH      ; *
22 0ABD 6A        DEFB    6AH      ; +
23 0ABE 2F        DEFB    2FH      ; ,
24 0ABF 2A        DEFB    2AH      ; -
25 0AC0 2E        DEFB    2EH      ; .
26 0AC1 2D        DEFB    2DH      ; /
27 0AC2           ; 30 - 3F ;
28 0AC2 20        DEFB    20H      ; 0
29 0AC3 21        DEFB    21H      ; 1
30 0AC4 22        DEFB    22H      ; 2
31 0AC5 23        DEFB    23H      ; 3
32 0AC6 24        DEFB    24H      ; 4
33 0AC7 25        DEFB    25H      ; 5
34 0AC8 26        DEFB    26H      ; 6
35 0AC9 27        DEFB    27H      ; 7
36 0ACA 28        DEFB    28H      ; 8
37 0ACB 29        DEFB    29H      ; 9
38 0ACC 4F        DEFB    4FH      ; :
39 0ACD 2C        DEFB    2CH      ; ;
40 0ACE 51        DEFB    51H      ; <
41 0ACF 2B        DEFB    2BH      ; =
42 0AD0 57        DEFB    57H      ; >
43 0AD1 49        DEFB    49H      ; ?
44 0AD2           ; 40 - 4F ;
45 0AD2 55        DEFB    55H      ; @
46 0AD3 01        DEFB    01H      ; A
47 0AD4 02        DEFB    02H      ; B
48 0AD5 03        DEFB    03H      ; C
49 0AD6 04        DEFB    04H      ; D
50 0AD7 05        DEFB    05H      ; E
51 0AD8 06        DEFB    06H      ; F
52 0AD9 07        DEFB    07H      ; G
53 0ADA 08        DEFB    08H      ; H
54 0ADB 09        DEFB    09H      ; I
55 0ADC 0A        DEFB    0AH      ; J
56 0ADD 0B        DEFB    0BH      ; K
57 0ADE 0C        DEFB    0CH      ; L
58 0ADF 0D        DEFB    0DH      ; M
59 0AE0 0E        DEFB    0EH      ; N
60 0AE1 0F        DEFB    0FH      ; O

```

```

01 0AE2      ; 50 - 5F ;
02 0AE2 10   DEFB 10H   ; P
03 0AE3 11   DEFB 11H   ; Q
04 0AE4 12   DEFB 12H   ; R
05 0AE5 13   DEFB 13H   ; S
06 0AE6 14   DEFB 14H   ; T
07 0AE7 15   DEFB 15H   ; U
08 0AE8 16   DEFB 16H   ; V
09 0AE9 17   DEFB 17H   ; W
10 0AEA 18   DEFB 18H   ; X
11 0AEB 19   DEFB 19H   ; Y
12 0AEC 1A   DEFB 1AH   ; Z
13 0AED 52   DEFB 52H   ; [
14 0AEE 59   DEFB 59H   ; \
15 0AEF 54   DEFB 54H   ; ]
16 0AF0 50   DEFB 50H   ; ^
17 0AF1 45   DEFB 45H   ; _
18 0AF2      ; 60 - 6F ;
19 0AF2 C7   DEFB C7H   ;
20 0AF3 C8   DEFB C8H   ; UFO
21 0AF4 C9   DEFB C9H   ;
22 0AF5 CA   DEFB CAH   ;
23 0AF6 CB   DEFB CBH   ;
24 0AF7 CC   DEFB CCH   ;
25 0AF8 CD   DEFB CDH   ;
26 0AF9 CE   DEFB CEH   ;
27 0AFA CF   DEFB CFH   ;
28 0AFB DF   DEFB DFH   ;
29 0AFC E7   DEFB E7H   ;
30 0AFD E8   DEFB E8H   ;
31 0AFE E5   DEFB E5H   ;
32 0AFF E9   DEFB E9H   ;
33 0B00 EC   DEFB ECH   ;
34 0B01 ED   DEFB EDH   ;
35 0B02      ; 70 - 7F ;
36 0B02 D0   DEFB D0H   ;
37 0B03 D1   DEFB D1H   ;
38 0B04 D2   DEFB D2H   ;
39 0B05 D3   DEFB D3H   ;
40 0B06 D4   DEFB D4H   ;
41 0B07 D5   DEFB D5H   ;
42 0B08 D6   DEFB D6H   ;
43 0B09 D7   DEFB D7H   ;
44 0B0A D8   DEFB D8H   ;
45 0B0B D9   DEFB D9H   ;
46 0B0C DA   DEFB DAH   ;
47 0B0D DB   DEFB DBH   ;
48 0B0E DC   DEFB DCH   ;
49 0B0F DD   DEFB DDH   ;
50 0B10 DE   DEFB DEH   ;
51 0B11 CO   DEFB COH   ;
52 0B12      ; 80 - 8F ;
53 0B12 80   DEFB 80H   ;
54 0B13 8D   DEFB 8DH   ;
55 0B14 9D   DEFB 9DH   ;
56 0B15 B1   DEFB B1H   ;
57 0B16 B5   DEFB B5H   ;
58 0B17 B9   DEFB B9H   ;
59 0B18 B4   DEFB B4H   ;
60 0B19 9E   DEFB 9EH   ;

```

```

01 0B1A B2   DEFB B2H   ;
02 0B1B B6   DEFB B6H   ;
03 0B1C BA   DEFB BAH   ;
04 0B1D BE   DEFB BEH   ;
05 0B1E 9F   DEFB 9FH   ;
06 0B1F B3   DEFB B3H   ;
07 0B20 B7   DEFB B7H   ;
08 0B21 BB   DEFB BBH   ;
09 0B22      ; 90 - 9F ;
10 0B22 BF   DEFB BFH   ;
11 0B23 A3   DEFB A3H   ;
12 0B24 85   DEFB 85H   ;
13 0B25 A4   DEFB A4H   ;
14 0B26 A5   DEFB A5H   ;
15 0B27 A6   DEFB A6H   ;
16 0B28 94   DEFB 94H   ;
17 0B29 87   DEFB 87H   ;
18 0B2A 88   DEFB 88H   ;
19 0B2B 9C   DEFB 9CH   ;
20 0B2C 82   DEFB 82H   ;
21 0B2D 98   DEFB 98H   ;
22 0B2E 84   DEFB 84H   ;
23 0B2F 92   DEFB 92H   ;
24 0B30 90   DEFB 90H   ;
25 0B31 83   DEFB 83H   ;
26 0B32      ; A0 - AF ;
27 0B32 91   DEFB 91H   ;
28 0B33 81   DEFB 81H   ;
29 0B34 9A   DEFB 9AH   ;
30 0B35 97   DEFB 97H   ;
31 0B36 93   DEFB 93H   ;
32 0B37 95   DEFB 95H   ;
33 0B38 89   DEFB 89H   ;
34 0B39 A1   DEFB A1H   ;
35 0B3A AF   DEFB AFH   ;
36 0B3B 8B   DEFB 8BH   ;
37 0B3C 86   DEFB 86H   ;
38 0B3D 96   DEFB 96H   ;
39 0B3E A2   DEFB A2H   ;
40 0B3F AB   DEFB ABH   ;
41 0B40 AA   DEFB AAH   ;
42 0B41 8A   DEFB 8AH   ;
43 0B42      ; B0 - BF ;
44 0B42 8E   DEFB 8EH   ;
45 0B43 B0   DEFB B0H   ;
46 0B44 AD   DEFB ADH   ;
47 0B45 BD   DEFB BDH   ;
48 0B46 A7   DEFB A7H   ;
49 0B47 AB   DEFB ABH   ;
50 0B48 A9   DEFB A9H   ;
51 0B49 8F   DEFB 8FH   ;
52 0B4A 8C   DEFB 8CH   ;
53 0B4B AE   DEFB AEH   ;
54 0B4C AC   DEFB ACH   ;
55 0B4D 9B   DEFB 9BH   ;
56 0B4E A0   DEFB A0H   ;
57 0B4F 99   DEFB 99H   ;
58 0B50 BC   DEFB BCH   ;
59 0B51 B8   DEFB BBH   ;
60 0B52      ; C0 - CF ;

```

\*\* Z80 ASSEMBLER SB-7201 &lt;1Z-013A&gt; PAGE 43

04.07.83

```

01 0B52 40      DEFB 40H
02 0B53 3B      DEFB 3BH
03 0B54 3A      DEFB 3AH
04 0B55 70      DEFB 70H
05 0B56 3C      DEFB 3CH
06 0B57 71      DEFB 71H
07 0B58 5A      DEFB 5AH
08 0B59 3D      DEFB 3DH
09 0B5A 43      DEFB 43H
10 0B5B 56      DEFB 56H
11 0B5C 3F      DEFB 3FH
12 0B5D 1E      DEFB 1EH
13 0B5E 4A      DEFB 4AH
14 0B5F 1C      DEFB 1CH
15 0B60 5D      DEFB 5DH
16 0B61 3E      DEFB 3EH
17 0B62          ; DO - DF ;
18 0B62 5C      DEFB 5CH
19 0B63 1F      DEFB 1FH
20 0B64 5F      DEFB 5FH
21 0B65 5E      DEFB 5EH
22 0B66 37      DEFB 37H
23 0B67 7B      DEFB 7BH
24 0B68 7F      DEFB 7FH
25 0B69 36      DEFB 36H
26 0B6A 7A      DEFB 7AH
27 0B6B 7E      DEFB 7EH
28 0B6C 33      DEFB 33H
29 0B6D 4B      DEFB 4BH
30 0B6E 4C      DEFB 4CH
31 0B6F 1D      DEFB 1DH
32 0B70 6C      DEFB 6CH
33 0B71 5B      DEFB 5BH
34 0B72          ; EO - EF ;
35 0B72 78      DEFB 78H
36 0B73 41      DEFB 41H
37 0B74 35      DEFB 35H
38 0B75 34      DEFB 34H
39 0B76 74      DEFB 74H
40 0B77 30      DEFB 30H
41 0B78 38      DEFB 38H
42 0B79 75      DEFB 75H
43 0B7A 39      DEFB 39H
44 0B7B 4D      DEFB 4DH
45 0B7C 6F      DEFB 6FH
46 0B7D 6E      DEFB 6EH
47 0B7E 32      DEFB 32H
48 0B7F 77      DEFB 77H
49 0B80 76      DEFB 76H
50 0B81 72      DEFB 72H
51 0B82          ; FO - FF ;
52 0B82 73      DEFB 73H
53 0B83 47      DEFB 47H
54 0B84 7C      DEFB 7CH
55 0B85 53      DEFB 53H
56 0B86 31      DEFB 31H
57 0B87 4E      DEFB 4EH
58 0B88 6D      DEFB 6DH
59 0B89 48      DEFB 48H
60 0B8A 46      DEFB 46H

```

; I

\*\* Z80 ASSEMBLER SB-7201 &lt;1Z-013A&gt; PAGE 44

04.07.83

```

01 0B8B 7D      DEFB 7DH
02 0B8C 44      DEFB 44H
03 0B8D 1B      DEFB 1BH
04 0B8E 58      DEFB 58H
05 0B8F 79      DEFB 79H
06 0B90 42      DEFB 42H
07 0B91 60      DEFB 60H
08 0B92          ;
09 0B92          ;
10 0B92          ; FLASHING DATA SAVE
11 0B92          ;
12 0B92          ?SAVE: ENT
13 0B92 219211  LD HL,FLSDT
14 0B95 36EF    LD (HL),EFH ; NOMAL CURSOR
15 0B97 3A7011  LD A,(KANAF)
16 0B9A 0F      RRCA
17 0B9B 3803    JR C,SVO-2 ; GRAPH MODE
18 0B9D 0F      RRCA
19 0B9E 3002    JR NC,SVO ; NORMAL MODE
20 0BA0 36FF    LD (HL),FFH ; GRAPH CURSOR
21 0BA2          SVO: ENT
22 0BA2 7E      LD A,(HL)
23 0BA3 F5      PUSH AF
24 0BA4 CDB10F  CALL ?PONT ; FLASING POSITION
25 0BA7 7E      LD A,(HL)
26 0BA8 328E11  LD (FLASH),A
27 0BAB F1      POP AF
28 0BAC 77      LD (HL),A
29 0BAD AF      XOR A
30 0BAE 2100E0  LD HL,KEYPA
31 0BB1 77      LD (HL),A
32 0BB2 2F      CPL
33 0BB3 77      LD (HL),A
34 0BB4 C9      RET
35 0BB5          SV1: ENT
36 0BB5 3643    LD (HL),43H ; KANA CURSOR
37 0BB7 18E9    JR SVO
38 0BB9          ;
39 0BB9          ; ORG 0BB9H;?ADCN
40 0BB9          ;
41 0BB9          ;
42 0BB9          ; ASCII TO DISPLAY CODE CONVERTE
43 0BB9          ;
44 0BB9          ; IN ACC:ASCII
45 0BB9          ; EXIT ACC:DISPLAY CODE
46 0BB9          ;
47 0BB9          ?ADCN: ENT
48 0BB9 C5      PUSH BC
49 0BBA E5      PUSH HL
50 0BBB 21920A  LD HL,ATBL
51 0BBE 4F      LD C,A
52 0BBF 0600    LD B,0
53 0BC1 09      ADD HL,BC
54 0BC2 7E      LD A,(HL)
55 0BC3 181B    JR DACN3
56 0BC5          ;
57 0BC5 56312E30 VRNS: DEFM "V1.0A" ; VERSION MANAGEMENT
58 0BC9 41          DEFB 0DH
59 0BCA 0D          DEFS +3
60 0BCB

```

```

01 OBCE ;
02 OBCE ;
03 OBCE ;ORG OBCEH;?DACN
04 OBCE ;
05 OBCE ; DISPLAY CODE TO ASCII CONV. ;
06 OBCE ;
07 OBCE ; IN ACC = DISPLAY CODE
08 OBCE ; EXIT ACC = ASCII
09 OBCE ;
10 OBCE ?DACN: ENT
11 OBCE C5 PUSH BC
12 OBCE E5 PUSH HL
13 OBCE D5 PUSH DE
14 OBCE 21920A LD HL,ATBL
15 OBCE 54 LD D,H
16 OBCE 5D LD E,L
17 OBCE 010001 LD BC,0100H
18 OBCE EDB1 CPIR
19 OBCE 2806 JR Z,DACN1
20 OBCE 3EFO LD A,FOH
21 OBCE DACN2: ENT
22 OBCE D1 POP DE
23 OBCE DACN3: ENT
24 OBCE E1 POP HL
25 OBCE C1 POP BC
26 OBCE C9 RET
27 OBCE3 ;
28 OBCE3 DACN1: ENT
29 OBCE3 B7 OR A
30 OBCE4 2B DEC HL
31 OBCE5 ED52 SBC HL,DE
32 OBCE7 7D LD A,L
33 OBCE8 18F5 JR DACN2
34 OBCEA ;
35 OBCEA ;
36 OBCEA ;
37 OBCEA ; KEY MATRIX TO DISPLAY CODE TABL
38 OBCEA ;
39 OBCEA KTBLS: ENT
40 OBCEA ;S0 00 - 07 ;
41 OBCEA BF DEFB BFH ; SPARE
42 OBCEB CA DEFB CAH ; GRAPH
43 OBCEC 58 DEFB 58H ; ↓
44 OBCEC C9 DEFB C9H ; ALPHA
45 OBCEE F0 DEFB F0H ; NO
46 OBCEF 2C DEFB 2CH ; :
47 OBFF0 4F DEFB 4FH ; ;
48 OBFF1 CD DEFB CDH ; CR
49 OBFF2 ;S1 08 - 0F ;
50 OBFF2 19 DEFB 19H ; Y
51 OBFF3 1A DEFB 1AH ; Z
52 OBFF4 55 DEFB 55H ; @
53 OBFF5 52 DEFB 52H ; [
54 OBFF6 54 DEFB 54H ; ]
55 OBFF7 F0 DEFB F0H ; NULL
56 OBFF8 F0 DEFB F0H ; NULL
57 OBFF9 F0 DEFB F0H ; NULL
58 OBFFA ;S2 0 - 17 ;
59 OBFFA 11 DEFB 11H ; Q
60 OBFFB 12 DEFB 12H ; R

```

```

01 OBFC 13 DEFB 13H ; S
02 OBFD 14 DEFB 14H ; T
03 OBFE 15 DEFB 15H ; U
04 OBFF 16 DEFB 16H ; V
05 OC00 17 DEFB 17H ; W
06 OC01 18 DEFB 18H ; X
07 OC02 ;S3 1B - 1F ;
08 OC02 09 DEFB 09H ; I
09 OC03 0A DEFB 0AH ; J
10 OC04 0B DEFB 0BH ; K
11 OC05 0C DEFB 0CH ; L
12 OC06 0D DEFB 0DH ; M
13 OC07 0E DEFB 0EH ; N
14 OC08 0F DEFB 0FH ; O
15 OC09 10 DEFB 10H ; P
16 OC0A ;S4 20 - 27 ;
17 OC0A 01 DEFB 01H ; A
18 OC0B 02 DEFB 02H ; B
19 OC0C 03 DEFB 03H ; C
20 OC0D 04 DEFB 04H ; D
21 OC0E 05 DEFB 05H ; E
22 OC0F 06 DEFB 06H ; F
23 OC10 07 DEFB 07H ; G
24 OC11 08 DEFB 08H ; H
25 OC12 ;S5 28 - 2F ;
26 OC12 21 DEFB 21H ; 1
27 OC13 22 DEFB 22H ; 2
28 OC14 23 DEFB 23H ; 3
29 OC15 24 DEFB 24H ; 4
30 OC16 25 DEFB 25H ; 5
31 OC17 26 DEFB 26H ; 6
32 OC18 27 DEFB 27H ; 7
33 OC19 28 DEFB 28H ; 8
34 OC1A ;S6 30 - 37 ;
35 OC1A 59 DEFB 59H ; \
36 OC1B 50 DEFB 50H ; ↑
37 OC1C 2A DEFB 2AH ; -
38 OC1D 00 DEFB 00H ; SPACE
39 OC1E 20 DEFB 20H ; 0
40 OC1F 29 DEFB 29H ; 9
41 OC20 2F DEFB 2FH ; ,
42 OC21 2E DEFB 2EH ; .
43 OC22 ;S7 38 - 3F ;
44 OC22 C8 DEFB C8H ; INST.
45 OC23 C7 DEFB C7H ; DEL.
46 OC24 C2 DEFB C2H ; CURSOR UP
47 OC25 C1 DEFB C1H ; CURSOR DOWN
48 OC26 C3 DEFB C3H ; CURSOR RIGHT
49 OC27 C4 DEFB C4H ; CURSOR LEFT
50 OC28 49 DEFB 49H ; ?
51 OC29 2D DEFB 2DH ; /
52 OC2A ;
53 OC2A ; KTBLS SHIFT ON
54 OC2A ;
55 OC2A KTBLS: ENT
56 OC2A ;S0 00-07
57 OC2A BF DEFB BFH ; SPARE
58 OC2B CA DEFB CAH ; GRAPH
59 OC2C 1B DEFB 1BH ; POND
60 OC2D C9 DEFB C9H ; ALPHA

```

\*\* Z80 ASSEMBLER SB-7201 &lt;1Z-013A&gt; PAGE 47

04.07.83

```

01 0C2E F0      DEFB F0H      ; NO
02 0C2F 6A      DEFB 6AH      ; +
03 0C30 6B      DEFB 6BH      ; *
04 0C31 CD      DEFB CDH      ; CR
05 0C32          ;S1 0B-0F
06 0C32 99      DEFB 99H      ; y
07 0C33 9A      DEFB 9AH      ; z
08 0C34 A4      DEFB A4H      ; \
09 0C35 BC      DEFB BCH      ; {
10 0C36 40      DEFB 40H      ; }
11 0C37 F0      DEFB F0H      ; NULL
12 0C38 F0      DEFB F0H      ; NULL
13 0C39 F0      DEFB F0H      ; NULL
14 0C3A          ;S2 10-17
15 0C3A 91      DEFB 91H      ; q
16 0C3B 92      DEFB 92H      ; r
17 0C3C 93      DEFB 93H      ; s
18 0C3D 94      DEFB 94H      ; t
19 0C3E 95      DEFB 95H      ; u
20 0C3F 96      DEFB 96H      ; v
21 0C40 97      DEFB 97H      ; w
22 0C41 98      DEFB 98H      ; x
23 0C42          ;S3 1B-1F
24 0C42 89      DEFB 89H      ; i
25 0C43 8A      DEFB 8AH      ; j
26 0C44 8B      DEFB 8BH      ; k
27 0C45 8C      DEFB 8CH      ; l
28 0C46 8D      DEFB 8DH      ; m
29 0C47 8E      DEFB 8EH      ; n
30 0C48 8F      DEFB 8FH      ; o
31 0C49 90      DEFB 90H      ; p
32 0C4A          ;S4 20-27
33 0C4A 81      DEFB 81H      ; a
34 0C4B 82      DEFB 82H      ; b
35 0C4C 83      DEFB 83H      ; c
36 0C4D 84      DEFB 84H      ; d
37 0C4E 85      DEFB 85H      ; e
38 0C4F 86      DEFB 86H      ; f
39 0C50 87      DEFB 87H      ; g
40 0C51 88      DEFB 88H      ; h
41 0C52          ;S5 2B-2F
42 0C52 61      DEFB 61H      ; !
43 0C53 62      DEFB 62H      ; "
44 0C54 63      DEFB 63H      ; #
45 0C55 64      DEFB 64H      ; $
46 0C56 65      DEFB 65H      ; %
47 0C57 66      DEFB 66H      ; &
48 0C58 67      DEFB 67H      ; '
49 0C59 68      DEFB 68H      ; (
50 0C5A          ;S6 30-37
51 0C5A 80      DEFB 80H      ; \
52 0C5B A5      DEFB A5H      ; POND MARK
53 0C5C 2B      DEFB 2BH      ; YEN
54 0C5D 00      DEFB 00H      ; SPACE
55 0C5E 60      DEFB 60H      ; π
56 0C5F 69      DEFB 69H      ; )
57 0C60 51      DEFB 51H      ; <
58 0C61 57      DEFB 57H      ; >
59 0C62          ;S7 3B-3F
60 0C62 C6      DEFB C6H      ; CLR

```

\*\* Z80 ASSEMBLER SB-7201 &lt;1Z-013A&gt; PAGE 48

04.07.83

```

01 0C63 C5      DEFB C5H      ; HOME
02 0C64 C2      DEFB C2H      ; CURSOR UP
03 0C65 C1      DEFB C1H      ; CURSOR DOWN
04 0C66 C3      DEFB C3H      ; CURSOR RIGHT
05 0C67 C4      DEFB C4H      ; CURSOR LEFT
06 0C68 5A      DEFB 5AH      ; ⇐
07 0C69 45      DEFB 45H      ; ⇨
08 0C6A          ;
09 0C6A          ; GRAPHIC
10 0C6A          ;
11 0C6A          ; KTBLS: ENT
12 0C6A          ;S0 00-07
13 0C6A BF      DEFB BFH      ; SPARE
14 0C6B F0      DEFB F0H      ; GRAPH BUT NULL
15 0C6C E5      DEFB E5H      ; #4
16 0C6D C9      DEFB C9H      ; ALPHA
17 0C6E F0      DEFB F0H      ; NO
18 0C6F 42      DEFB 42H      ; #;
19 0C70 B6      DEFB B6H      ; #;
20 0C71 CD      DEFB CDH      ; CR
21 0C72          ;S1 0B-0F
22 0C72 75      DEFB 75H      ; #Y
23 0C73 76      DEFB 76H      ; #Z
24 0C74 B2      DEFB B2H      ; #0
25 0C75 D8      DEFB D8H      ; #I
26 0C76 4E      DEFB 4EH      ; #J
27 0C77 F0      DEFB F0H      ; #NULL
28 0C78 F0      DEFB F0H      ; #NULL
29 0C79 F0      DEFB F0H      ; #NULL
30 0C7A          ;S2 10-17
31 0C7A 3C      DEFB 3CH      ; #Q
32 0C7B 30      DEFB 30H      ; #R
33 0C7C 44      DEFB 44H      ; #S
34 0C7D 71      DEFB 71H      ; #T
35 0C7E 79      DEFB 79H      ; #U
36 0C7F DA      DEFB DAH      ; #V
37 0C80 3B      DEFB 3BH      ; #W
38 0C81 6D      DEFB 6DH      ; #X
39 0C82          ;S3 1B-1F
40 0C82 7D      DEFB 7DH      ; #I
41 0C83 5C      DEFB 5CH      ; #J
42 0C84 5B      DEFB 5BH      ; #K
43 0C85 B4      DEFB B4H      ; #L
44 0C86 1C      DEFB 1CH      ; #M
45 0C87 32      DEFB 32H      ; #N
46 0C88 B0      DEFB B0H      ; #O
47 0C89 D6      DEFB D6H      ; #P
48 0C8A          ;S4 20-27
49 0C8A 53      DEFB 53H      ; #A
50 0C8B 6F      DEFB 6FH      ; #B
51 0C8C DE      DEFB DEH      ; #C
52 0C8D 47      DEFB 47H      ; #D
53 0C8E 34      DEFB 34H      ; #E
54 0C8F 4A      DEFB 4AH      ; #F
55 0C90 4B      DEFB 4BH      ; #G
56 0C91 72      DEFB 72H      ; #H
57 0C92          ;S5 2B-2F
58 0C92 37      DEFB 37H      ; #1
59 0C93 3E      DEFB 3EH      ; #2
60 0C94 7F      DEFB 7FH      ; #3

```

```

01 0C95 7B      DEFB 7BH      ; #4
02 0C96 3A      DEFB 3AH      ; #5
03 0C97 5E      DEFB 5EH      ; #6
04 0C98 1F      DEFB 1FH      ; #7
05 0C99 BD      DEFB BDH      ; #8
06 0C9A          ;S6 30-3F
07 0C9A D4      DEFB D4H      ; #YEN
08 0C9B 9E      DEFB 9EH      ; #+
09 0C9C D2      DEFB D2H      ; #-
10 0C9D 00      DEFB 00H      ; SPACE
11 0C9E 9C      DEFB 9CH      ; #0
12 0C9F A1      DEFB A1H      ; #9
13 0CA0 CA      DEFB CAH      ; #.
14 0CA1 B8      DEFB B8H      ; #.
15 0CA2          ;S7 38-3F
16 0CA2 C8      DEFB C8H      ; INST
17 0CA3 C7      DEFB C7H      ; DEL.
18 0CA4 C2      DEFB C2H      ; CURSOR UP
19 0CA5 C1      DEFB C1H      ; CURSOR DOWN
20 0CA6 C3      DEFB C3H      ; CURSOR RIGHT
21 0CA7 C4      DEFB C4H      ; CURSOR LEFT
22 0CAB BA      DEFB BAH      ; #?
23 0CA9 DB      DEFB DBH      ; #/
24 0CAA          ;
25 0CAA          ; CONTROL CODE
26 0CAA          ;
27 0CAA          ; KTBLC: ENT
28 0CAA          ;S0 00-07N
29 0CAA F0      DEFB F0H
30 0CAB F0      DEFB F0H
31 0CAC F0      DEFB F0H
32 0CAD F0      DEFB F0H
33 0CAE F0      DEFB F0H
34 0CAF F0      DEFB F0H
35 0CB0 F0      DEFB F0H
36 0CB1 F0      DEFB F0H
37 0CB2          ;S1 08-0F
38 0CB2 F0      DEFB F0H
39 0CB3 5A      DEFB 5AH
40 0CB4 F0      DEFB F0H
41 0CB5 F0      DEFB F0H
42 0CB6 F0      DEFB F0H
43 0CB7 F0      DEFB F0H
44 0CB8 F0      DEFB F0H
45 0CB9 F0      DEFB F0H
46 0CBA          ;S2 10-17
47 0CBA C1      DEFB C1H
48 0CBB C2      DEFB C2H
49 0CBC C3      DEFB C3H
50 0CBD C4      DEFB C4H
51 0CBE C5      DEFB C5H
52 0CBF C6      DEFB C6H
53 0CC0 F0      DEFB F0H
54 0CC1 F0      DEFB F0H
55 0CC2          ;S3 18-1F
56 0CC2 F0      DEFB F0H
57 0CC3 F0      DEFB F0H
58 0CC4 F0      DEFB F0H
59 0CC5 F0      DEFB F0H
60 0CC6 F0      DEFB F0H

```

```

; #4
; #5
; #6
; #7
; #8
; #YEN
; #+
; #-
; SPACE
; #0
; #9
; #.
; #.
; INST
; DEL.
; CURSOR UP
; CURSOR DOWN
; CURSOR RIGHT
; CURSOR LEFT
; #?
; #/
;
;
; #
;
; #Y E3
; #Z E4 (CHECKER)
; #0
; #E E5
; #J E7
;
; #Q
; #R
; #S
; #T
; #U
; #V
; #W E1
; #X E2
;
; #I F9
; #J FA
; #K FB
; #L FC
; #M FD

```

```

01 0CC7 F0      DEFB F0H      ; #N FE
02 0CC8 F0      DEFB F0H      ; #O FF
03 0CC9 F0      DEFB F0H      ; #P E0
04 0CCA          ;S4 20-27
05 0CCA F0      DEFB F0H
06 0CCB F0      DEFB F0H
07 0CCC F0      DEFB F0H
08 0CCD F0      DEFB F0H
09 0CCE F0      DEFB F0H
10 0CCF F0      DEFB F0H
11 0CD0 F0      DEFB F0H
12 0CD1 F0      DEFB F0H
13 0CD2          ;S5 28-2F
14 0CD2 F0      DEFB F0H
15 0CD3 F0      DEFB F0H
16 0CD4 F0      DEFB F0H
17 0CD5 F0      DEFB F0H
18 0CD6 F0      DEFB F0H
19 0CD7 F0      DEFB F0H
20 0CD8 F0      DEFB F0H
21 0CD9 F0      DEFB F0H
22 0CDA          ;S6 30-37
23 0CDA F0      DEFB F0H
24 0CDB F0      DEFB F0H
25 0CDC F0      DEFB F0H
26 0CDD F0      DEFB F0H
27 0CDE F0      DEFB F0H
28 0CDF F0      DEFB F0H
29 0CE0 F0      DEFB F0H
30 0CE1          ;S7 38-3F
31 0CE1 F0      DEFB F0H
32 0CE2 F0      DEFB F0H
33 0CE3 F0      DEFB F0H
34 0CE4 F0      DEFB F0H
35 0CE5 F0      DEFB F0H
36 0CE6 F0      DEFB F0H
37 0CE7 F0      DEFB F0H
38 0CE8 F0      DEFB F0H
39 0CE9          ;
40 0CE9          ; KANA
41 0CE9          ;
42 0CE9          ; KTBLC: ENT
43 0CE9          ;S0 00-07
44 0CE9 BF      DEFB BFH
45 0CEA F0      DEFB F0H
46 0CEB CF      DEFB CFH
47 0CEC C9      DEFB C9H
48 0CED F0      DEFB F0H
49 0CEE B5      DEFB B5H
50 0CEF 4D      DEFB 4DH
51 0CF0 CD      DEFB CDH
52 0CF1          ;S1 08-0F
53 0CF1 35      DEFB 35H
54 0CF2 77      DEFB 77H
55 0CF3 D7      DEFB D7H
56 0CF4 B3      DEFB B3H
57 0CF5 B7      DEFB B7H
58 0CF6 F0      DEFB F0H
59 0CF7 F0      DEFB F0H
60 0CF8 F0      DEFB F0H

```

```

; #N FE
; #O FF
; #P E0
;
; #A F1
; #B F2
; #C F3
; #D F4
; #E F5
; #F F6
; #G F7
; #H F8
;
; #YEN E6
;
; #, EF
;
; #/ EE
;
; SPARE
; GRAPH BUT NULL
; NIKO WH.
; ALPHA
; NO
; MO
; DAKU TEN
; CR
;
; HA
; TA
; WA
; YO
; HANDAKU

```

\*\* Z80 ASSEMBLER SB-7201 &lt;1Z-013A&gt; PAGE 51

04.07.83

```

01 0CF9          ;S2      10-17
02 0CF9 7C      DEFB 7CH      ; KA
03 0CFA 70      DEFB 70H      ; KE
04 0CFB 41      DEFB 41H      ; SHI
05 0CFC 31      DEFB 31H      ; KO
06 0CFD 39      DEFB 39H      ; HI
07 0CFE A6      DEFB A6H      ; TE
08 0CFF 78      DEFB 78H      ; KI
09 0D00 DD      DEFB DDH      ; CHI
10 0D01          ;S3      18-1F
11 0D01 3D      DEFB 3DH      ; FU
12 0D02 5D      DEFB 5DH      ; MI
13 0D03 6C      DEFB 6CH      ; MU
14 0D04 56      DEFB 56H      ; ME
15 0D05 1D      DEFB 1DH      ; RHI
16 0D06 33      DEFB 33H      ; RA
17 0D07 D5      DEFB D5H      ; HE
18 0D08 B1      DEFB B1H      ; HO
19 0D09          ;S4      20-27
20 0D09 46      DEFB 46H      ; SA
21 0D0A 6E      DEFB 6EH      ; TO
22 0D0B D9      DEFB D9H      ; THU
23 0D0C 48      DEFB 48H      ; SU
24 0D0D 74      DEFB 74H      ; KU
25 0D0E 43      DEFB 43H      ; SE
26 0D0F 4C      DEFB 4CH      ; SO
27 0D10 73      DEFB 73H      ; MA
28 0D11          ;S5      28-2F
29 0D11 3F      DEFB 3FH      ; A
30 0D12 36      DEFB 36H      ; I
31 0D13 7E      DEFB 7EH      ; U
32 0D14 3B      DEFB 3BH      ; E
33 0D15 7A      DEFB 7AH      ; O
34 0D16 1E      DEFB 1EH      ; NA
35 0D17 5F      DEFB 5FH      ; NI
36 0D18 A2      DEFB A2H      ; NU
37 0D19          ;S6      30-37
38 0D19 D3      DEFB D3H      ; YO
39 0D1A 9F      DEFB 9FH      ; YU
40 0D1B D1      DEFB D1H      ; YA
41 0D1C 00      DEFB 00H      ; SPACE
42 0D1D 9D      DEFB 9DH      ; NO
43 0D1E A3      DEFB A3H      ; NE
44 0D1F D0      DEFB D0H      ; RU
45 0D20 B9      DEFB B9H      ; RE
46 0D21          ;S7      38-3F
47 0D21 C6      DEFB C6H      ; ?CLR ®
48 0D22 C5      DEFB C5H      ; ?HOME □
49 0D23 C2      DEFB C2H      ; ?CURSOR UP
50 0D24 C1      DEFB C1H      ; ?CURSOR DOWN
51 0D25 C3      DEFB C3H      ; ?CURSOR RIGHT
52 0D26 C4      DEFB C4H      ; ?CURSOR LEFT
53 0D27 BB      DEFB BBH      ; DASH
54 0D28 BE      DEFB BEH      ; RO
55 0D29          ;
56 0D29          ; MEMORY DUMP
57 0D29          ; COMMAND 'D'
58 0D29          ;
59 0D29          ; DUMP: ENT
60 0D29 CD3D01  CALL HEXIY      ; START ADR.

```

\*\* Z80 ASSEMBLER SB-7201 &lt;1Z-013A&gt; PAGE 52

04.07.83

```

01 0D2C CDA602  CALL .4DE
02 0D2F E5      PUSH HL
03 0D30 CD1004  CALL HLHEX      ; END ADR.
04 0D33 D1      POP DE
05 0D34 3852    JR C,DUM1      ; DATA ER. THEN
06 0D36 EB      EX DE,HL
07 0D37 0608    DUM3: LD B,08H      ; DISP 8BYTES
08 0D39 0E17    LD C,23      ; CHA. PRINT BIAS
09 0D3B CDFAO5  CALL NLPHL      ; NEWLINE PRINT
10 0D3E CDB103  DUM2: CALL SPHEX      ; SP. PRT.+ACC PRT.
11 0D41 23      INC HL
12 0D42 F5      PUSH AF
13 0D43 3A7111  LD A,(DSPXY)      ; DISPLAY POINT
14 0D46 81      ADD A,C
15 0D47 327111  LD (DSPXY),A      ; X AXIS.=X+Creg
16 0D4A F1      POP AF
17 0D4B FE20    CP 20H
18 0D4D 3002    JR NC,+4
19 0D4F 3E2E    LD A,2EH      ; ' '
20 0D51 CDB908  CALL ?ADCN      ; ASCII TO DISPLAY CODE
21 0D54 CD6C09  CALL PRNT3
22 0D57 3A7111  LD A,(DSPXY)
23 0D5A 0C      INC C
24 0D5B 91      SUB C      ; ASCII DSP POSITION
25 0D5C 327111  LD (DSPXY),A
26 0D5F 0D      DEC C
27 0D60 0D      DEC C
28 0D61 0D      DEC C
29 0D62 E5      PUSH HL
30 0D63 ED52    SBC HL,DE
31 0D65 E1      POP HL
32 0D66 281D    JR Z,DUM1-3
33 0D68 3EFB    LD A,FBH
34 0D6A 3200E0  LD (KEYPA),A
35 0D6D 00      NOP
36 0D6E 3A01E0  LD A,(KEYPB)
37 0D71 FEFE    CP FEH      ; SHIFT KEY ?
38 0D73 2003    JR NZ,+5
39 0D75 CDA60D  CALL ?BLNK      ; 64MSEC DELAY
40 0D78 10C4    DJNZ DUM2
41 0D7A CDCA08  CALL ?KEY      ; STOP DISP
42 0D7D B7      OR A
43 0D7E 28FA    JR Z,-4      ; SPACE KEY THEN STOP
44 0D80 CD320A  CALL ?BRK      ; BREAK IN ?
45 0D83 20B2    JR NZ,DUM3
46 0D85 C3AD00  JP ST1      ; COMMAND IN !
47 0D88 21A000  LD HL,160      ; 20*8 BYTE
48 0D8B 19      ADD HL,DE
49 0DBC 18AB    JR DUM3-1
50 0D8E          ;
51 0D8E          ;
52 0D8E          ;
53 0D8E          ;
54 0D8E          ; DEFS +24
55 0DA6          ;
56 0DA6          ;
57 0DA6          ;
58 0DA6          ; ORG 0DA6H; ?BLNK
59 0DA6          ;
60 0DA6          ;

```



```

01 ODA6      ; V-BLANK CHECK ;
02 ODA6      ;
03 ODA6      ?BLNK: ENT
04 ODA6 F5    PUSH AF
05 ODA7 3A02E0 LD A,(KEYPC) ; V-BLNK
06 ODA8 07    RLCA
07 ODA8 30FA  JR NC,-4
08 ODA8 3A02E0 LD A,(KEYPC)
09 ODA8 07    RLCA
10 ODB1 38FA  JR C,-4
11 ODB3 F1    POP AF
12 ODB4 C9    RET
13 ODB5      ;
14 ODB5      ;ORG ODB5H;?DSP
15 ODB5      ;
16 ODB5      ;
17 ODB5      ;
18 ODB5      ; DISPLAY ON POINTER ;
19 ODB5      ;
20 ODB5      ; ACC = DISPLAY CODE
21 ODB5      ; EXCEPT FOH
22 ODB5      ;
23 ODB5      ?DSP: ENT
24 ODB5 F5    PUSH AF
25 ODB6 C5    PUSH BC
26 ODB7 D5    PUSH DE
27 ODB8 E5    PUSH HL
28 ODB9      DSP01: ENT
29 ODB9 CDB10F CALL ?PONT ; DSPLY POSITION
30 ODB9 77    LD (HL),A
31 ODB9 2A7111 LD HL,(DSPXY)
32 ODC0 7D    LD A,L
33 ODC1 FE27  CP +39
34 ODC3 200B  JR NZ,DSP04
35 ODC5 CDF302 CALL .MANG
36 ODC8 3806  JR C,DSP04
37 ODC8 EB    EX DE,HL
38 ODCB 3601  LD (HL),+1 ; LOGICAL 1ST COLUMN
39 ODCD 23    INC HL
40 ODCE 3600  LD (HL),0 ; LOGICAL 2ND COLUMN
41 ODD0      DSP04: ENT
42 ODD0 3EC3  LD A,C3H ; CURSL
43 ODD2 180C  JR ?DPCT+4
44 ODD4      ;
45 ODD4      ;
46 ODD4      ;
47 ODD4      ;
48 ODD4      ; GRAPHIC STATUS CHECK
49 ODD4      ;
50 ODD4 3A7011 GRSTAS: LD A,(KANAF)
51 ODD7 FE01  CP 01H
52 ODD9 3ECA  LD A,CAH
53 ODD8 C9    RET
54 ODDC      ;
55 ODDC      ;
56 ODDC      ;
57 ODDC      ;
58 ODDC      ;
59 ODDC      ;
60 ODDC      ;ORG ODDCH;?DPCT

```

```

01 ODDC      ;
02 ODDC      ;
03 ODDC      ; DISPLAY CONTROL ;
04 ODDC      ;
05 ODDC      ; ACC = CONTROL CODE
06 ODDC      ;
07 ODDC      ?DPCT: ENT
08 ODDC F5    PUSH AF
09 ODDC C5    PUSH BC
10 ODDC D5    PUSH DE
11 ODDC E5    PUSH HL
12 ODE0 47    LD B,A
13 ODE1 E6F0  AND FOH
14 ODE3 FEC0  CP COH
15 ODE5 201B  JR NZ,CURS5
16 ODE7 A8    XOR B
17 ODE8 07    RLCA
18 ODE9 4F    LD C,A
19 ODEA 0600  LD B,+0
20 ODEC 21AA0E LD HL,CTBL ; PAGE MODE1
21 ODEF 09    ADD HL,BC
22 ODF0 5E    LD E,(HL)
23 ODF1 23    INC HL
24 ODF2 56    LD D,(HL)
25 ODF3 2A7111 LD HL,(DSPXY)
26 ODF6 EB    EX DE,HL
27 ODF7 E9    JP (HL)
28 ODF8      ;
29 ODF8      ;
30 ODF8      ;
31 ODF8      ;
32 ODF8 EB    EX DE,HL ; LD HL,(DSPXY)
33 ODF9 7C    LD A,H
34 ODF8 FE18  CP +24
35 ODFC 2825  JR Z,CURS4
36 ODFE 24    INC H
37 ODF8      ;
38 ODF8      ;
39 ODF8      ;
40 ODF8      ;
41 ODF8      ;
42 ODF8      ;
43 ODF8      ;
44 ODF8      ;
45 ODF8      ;
46 ODF8      ;
47 ODF8      ;
48 ODF8      ;
49 ODF8      ;
50 ODF8      ;
51 ODF8      ;
52 ODF8      ;
53 ODF8      ;
54 ODF8      ;
55 ODF8      ;
56 ODF8      ;
57 ODF8      ;
58 ODF8      ;
59 ODF8      ;
60 ODF8      ;

```

\*\* Z80 ASSEMBLER SB-7201 &lt;1Z-013A&gt; PAGE 55

04.07.83

```

01 0E16 2E00          LD      L,+0
02 0E18 24           INC      H
03 0E19 7C           LD      A,H
04 0E1A FE19          CP      +25
05 0E1C 38E1          JR      C,CURS1
06 0E1E 2618          LD      H,+24
07 0E20 227111        LD      (DSPXY),HL
08 0E23              CURS4: ENT
09 0E23 1848          JR      SCROL
10 0E25              ;
11 0E25              CURSL: ENT
12 0E25 EB           EX      DE,HL          ; LD HL,(DSPXY)
13 0E26 7D           LD      A,L
14 0E27 B7           OR      A
15 0E28 2803          JR      Z,+5
16 0E2A 2D           DEC      L
17 0E2B 18D2          JR      CURS3
18 0E2D 2E27          LD      L,+39
19 0E2F 25           DEC      H
20 0E30 F20B0E        JP      P,CURSU1
21 0E33 2600          LD      H,0
22 0E35 227111        LD      (DSPXY),HL
23 0E38 18C8          JR      CURS5
24 0E3A              ;
25 0E3A              CLRS: ENT
26 0E3A 217311        LD      HL,MANG
27 0E3D 061B          LD      B,27
28 0E3F CDD80F        CALL    ?CLER
29 0E42 2100D0        LD      HL,D000H          ; SCRN TOP
30 0E45 CDD409        CALL    #CLR08
31 0E48 3E71          LD      A,71H          ; COLOR DATA
32 0E4A CDD509        CALL    #CLR8          ; D800H-DFFFH CLR.
33 0E4D              HOME: ENT
34 0E4D 210000        LD      HL,0          ; DSPXY:0 X=0,Y=0
35 0E50 18AD          JR      CURS3
36 0E52              ;
37 0E52              DEFS    +8
38 0E5A              ;
39 0E5A              ; CR
40 0E5A              ;
41 0E5A              CR: ENT
42 0E5A CDF302        CALL    .MANG
43 0E5D 0F           RRCA
44 0E5E 30B6          JR      NC,CURS2
45 0E60 2E00          LD      L,0
46 0E62 24           INC      H
47 0E63 FE18          CP      +24
48 0E65 2803          JR      Z,CR1
49 0E67 24           INC      H
50 0E68 1895          JR      CURS1
51 0E6A              CR1: ENT
52 0E6A 227111        LD      (DSPXY),HL
53 0E6D              ;
54 0E6D              ; SCROL
55 0E6D              ;
56 0E6D              SCROL: ENT
57 0E6D 01C003        LD      BC,03C0H
58 0E70 1100D0        LD      DE,SCRN          ; TOP OF $CRT ADR.
59 0E73 2128D0        LD      HL,SCRN+40      ; 1 COLUMN
60 0E76 C5           PUSH    BC          ; 1000 STORE

```

\*\* Z80 ASSEMBLER SB-7201 &lt;1Z-013A&gt; PAGE 56

04.07.83

```

01 0E77 EDB0          LDIR
02 0E79 C1           POP      BC
03 0E7A D5           PUSH    DE
04 0E7B 1100D8        LD      DE,SCRN+800H      ; COLOR RAM SCROLL
05 0E7E 2128D8        LD      HL,SCRN+828H      ; SCROLL TOP + 40
06 0E81 EDB0          LDIR
07 0E83 0628          LD      B,40          ; ONE LINE
08 0E85 EB           EX      DE,HL
09 0E86 3E71          LD      A,71H          ; COLOR RAM INITIAL DATA
10 0E88 CDDD0F        CALL    ?DINT
11 0E8B E1           POP      HL
12 0E8C 0628          LD      B,40
13 0E8E CDD80F        CALL    ?CLER          ; LAST LINE CLEAR
14 0E91 011A00        LD      BC,26          ; ROW NUMBER +1
15 0E94 117311        LD      DE,MANG          ; LOGICAL MANAGEMENT
16 0E97 217411        LD      HL,MANG+1
17 0E9A EDB0          LDIR
18 0E9C 3600          LD      (HL),0
19 0E9E 3A7311        LD      A,(MANG)
20 0EA1 B7           OR      A
21 0EA2 2841          JR      Z,?RSTR
22 0EA4 217211        LD      HL,DSPXY+1
23 0EA7 35           DEC      (HL)
24 0EA8 18C3          JR      SCROL
25 0EAA              ;
26 0EAA              ; CONTROL CODE TABLE
27 0EAA              ;
28 0EAA              CTBL: ENT
29 0EAA 6D0E          DEFW    SCROL          ; SCROLLING
30 0EAC F80D          DEFW    CURSD          ; CURSOR
31 0EAE 050E          DEFW    CURSU
32 0EB0 0D0E          DEFW    CURSR
33 0EB2 250E          DEFW    CURSL
34 0EB4 4D0E          DEFW    HOME
35 0EB6 3A0E          DEFW    CLRS
36 0EB8 F80E          DEFW    DEL
37 0EBA 380F          DEFW    INST
38 0EBC E10E          DEFW    ALPHA
39 0EBE EE0E          DEFW    KANA
40 0EC0 E50E          DEFW    ?RSTR
41 0EC2 E50E          DEFW    ?RSTR
42 0EC4 5A0E          DEFW    CR
43 0EC6 E50E          DEFW    ?RSTR
44 0EC8 E50E          DEFW    ?RSTR
45 0ECA              ;
46 0ECA              ;
47 0ECA              ; INST BYPASS
48 0ECA              ;
49 0ECA              ;
50 0ECA CBDC          INST2: SET    3,H          ; COLOR RAM
51 0EEC 7E           LD      A,(HL)          ; FROM
52 0ECD 23           INC      HL
53 0ECE 77           LD      (HL),A          ; TO
54 0ECF 2B           DEC      HL          ; ADR ADJ.
55 0ED0 CB9C          RES     3,H
56 0ED2 EDA8          LDD     A,C          ; CHA. TRNS.
57 0ED4 79           LD      B,A,C
58 0ED5 80           OR      B          ; BC=0 ?
59 0ED6 20F2          JR      NZ,INST2
60 0ED8 EB           EX      DE,HL

```

```

01 0ED9 3600      LD      (HL),0
02 0EDB CBDC      SET     3,H      ; COLOR RAM
03 0EDD 3671      LD      (HL),71H
04 0EDF 1804      JR      ?RSTR
05 0EE1           ;
06 0EE1           ;
07 0EE1           ;
08 0EE1           ;
09 0EE1           ; ORG 0EE1H;ALPHA
10 0EE1           ;
11 0EE1           ALPHA: ENT
12 0EE1 AF        XOR      A
13 0EE2           ALPH1: ENT
14 0EE2 327011    LD      (KANAF),A
15 0EE5           ;
16 0EE5           ;
17 0EE5           ; RESTORE ;
18 0EE5           ;
19 0EE5           ?RSTR: ENT
20 0EE5 E1        POP     HL
21 0EE6           ?RSTR1: ENT
22 0EE6 D1        POP     DE
23 0EE7 C1        POP     BC
24 0EE8 F1        POP     AF
25 0EE9 C9        RET
26 0EEA           ;
27 0EEA           ; MONITOR WORK AREA ;
28 0EEA           ;
29 0E00 P        SCRN: EQU D000H
30 0E03 P        KANST: EQU E003H      ; KANA STATUS PORT
31 0EEA           ;
32 0EEA           ;
33 0EEA           ;
34 0EEA           DEFS    +4
35 0EEE           ; ORG 0EEEH;KANA
36 0EEE           ;
37 0EEE           KANA: ENT
38 0EEE CDD40D    CALL    GRSTAS
39 0EF1 CAB90D    JP      Z,DSP01      ; NOT GRAPH KEY THEN JUM
40 0EF4 3E01      LD      A,+1
41 0EF6 18EA      JR      ALPH1
42 0EF8           ;
43 0EF8           ;
44 0EF8           ;
45 0EF8 EB        DEL: ENT DE,HL      ; LD HL,(DSPXY)
46 0EF9 7C        EX      A,H      ; HOME ?
47 0EFA B5        LD      A,H
48 0EFB 28EB      OR      L
49 0EFD 7D        JR      Z,?RSTR
50 0EFE B7        LD      A,L
51 0EFF 200D      OR      A
52 0F01 CDF302    JR      NZ,DEL1      ; LEFT SIDE ?
53 0F04 3B0B      CALL    .MANG
54 0F06 CDB10F    JR      C,DEL1
55 0F09 2B        CALL    ?PONT
56 0F0A 3600      DEC     HL
57 0F0C 1B25      LD      (HL),+0
58 0F0E           JR      INST-5      ; JP CURSL
59 0F0E CDF302    DEL1: ENT .MANG
60 0F11 0F        CALL    RRCA

```

```

01 0F12 5E2B      LD      A,40
02 0F14 3001      JR      NC,+3
03 0F16 07        RLCA
04 0F17 95        SUB     L      ; ACC=80
05 0F18 47        LD      B,A      ; TRNS. BYTE
06 0F19 CDB10F    CALL    ?PONT      ; CHA. FROM ADR
07 0F1C 7E        LD      A,(HL)
08 0F1D 2B        DEC     HL      ; TO
09 0F1E 77        LD      (HL),A
10 0F1F 23        INC     HL      ; COLOR RAM
11 0F20 CBDC      SET     3,H
12 0F22 7E        LD      A,(HL)
13 0F23 2B        DEC     HL
14 0F24 77        LD      (HL),A
15 0F25 CB9C      RES     3,H      ; CHA.
16 0F27 23        INC     HL      ; NEXT
17 0F28 23        INC     HL
18 0F29 10F1      DJNZ    DEL2
19 0F2B 2B        DEC     HL      ; ADR.ADJUST
20 0F2C 3600      LD      (HL),0
21 0F2E CBDC      SET     3,H
22 0F30 217100    LD      HL,71H      ; BLUE + WHITE
23 0F33 3EC4      LD      A,C4H      ; JP CURSL
24 0F35 C3E00D    JP      ?DPCT+4
25 0F3B           ;
26 0F3B           ; INST: ENT
27 0F3B CDF302    CALL    .MANG
28 0F3B 0F        RRCA
29 0F3C 2E27      LD      L,+39
30 0F3E 7D        LD      A,L
31 0F3F 3001      JR      NC,+3
32 0F41 24        INC     H
33 0F42 CDB40F    CALL    ?PNT1
34 0F45 E5        PUSH    HL
35 0F46 2A7111    LD      HL,(DSPXY)
36 0F49 3002      JR      NC,+4
37 0F4B 3E4F      LD      A,+79
38 0F4D 95        SUB     L
39 0F4E 0600      LD      B,0
40 0F50 4F        LD      C,A
41 0F51 D1        POP     DE
42 0F52 2B91      JR      Z,?RSTR
43 0F54 1A        LD      A,(DE)
44 0F55 B7        OR      A
45 0F56 20BD      JR      NZ,?RSTR
46 0F58 62        LD      H,D      ; HL←DE
47 0F59 6B        LD      L,E
48 0F5A 2B        DEC     HL
49 0F5B C3CA0E    JP      INST2      ; JUMP NEXT (BYPASS)
50 0F5E           ;
51 0F5E           ;
52 0F5E           ; PROGRAM SAVE
53 0F5E           ;
54 0F5E           ; CMD. 'S'
55 0F5E           ;
56 0F5E           ;
57 0F5E CD3D01    SAVE: ENT
58 0F61 220411    CALL    HEXIY      ; START ADR.
59 0F64 44        LD      (DTADR),HL ; DATA ADR. BUFFER
60 0F65 4D        LD      B,H

```

\*\* Z80 ASSEMBLER SB-7201 &lt;1Z-013A&gt; PAGE 59

04.07.83

```

01 0F66 CDA602      CALL .4DE
02 0F69 CD3D01      CALL HEXIY      ; END ADR.
03 0F6C ED42        SBC HL,BC      ; BYTE SIZE
04 0F6E 23          INC HL
05 0F6F 220211      LD (SIZE),HL  ; BYTE SIZE BUFFER
06 0F72 CDA602      CALL .4DE
07 0F75 CD3D01      CALL HEXIY      ; EXECUTE ADR.
08 0F78 220611      LD (EXADR),HL  ; BUFFER
09 0F7B CD0900      CALL NL
10 0F7E 118B09      LD DE,MSGSV    ; SAVED FILENAME
11 0FB1 DF          RST 3          ; CALL MSGX
12 0FB2 CD2F01      CALL BGETL      ; FILENAME INPUT
13 0FB5 CDA602      CALL .4DE
14 0FB8 CDA602      CALL .4DE
15 0FBB 21F110      LD HL,NAME      ; NAME BUFFER
16 0FBE             SAV1: ENT
17 0FBF 13          INC DE
18 0FBF 1A          LD A,(DE)
19 0F90 77          LD (HL),A      ; FILENAME TRANS.
20 0F91 23          INC HL
21 0F92 FE0D        CP ODH        ; END CODE
22 0F94 20F8        JR NZ,SAV1
23 0F96 3E01        LD A,01H      ; ATTRIBUTE:OBJ.
24 0F98 32F010      LD (ATRB),A
25 0F9B CD3604      CALL ?WRI
26 0F9E DA0701      JP C,?ER      ; WRITE ERROR
27 0FA1 CD7504      CALL ?WRD      ; DATA
28 0FA4 DA0701      JP C,?ER
29 0FA7 CD0900      CALL NL
30 0FAA 114209      LD DE,MSGOK    ; OK MESSAGE
31 0FAD DF          RST 3          ; CALL MSGX
32 0FAE C3AD00      JP ST1
33 0FB1             ;
34 0FB1             ;
35 0FB1             ; ORG 0FB1H;?PONT
36 0FB1             ;
37 0FB1             ;
38 0FB1             ; COMPUTE POINT ADR . ;
39 0FB1             ;
40 0FB1             ; HL = SCREEN COORDINATE
41 0FB1             ; EXIT
42 0FB1             ; HL = POINT ADR. ON SCREEN
43 0FB1             ;
44 0FB1             ?PONT: ENT
45 0FB1 2A7111      LD HL,(DSPXY)
46 0FB4             ;
47 0FB4             ; ORG 0FB4H;?PNT1
48 0FB4             ;
49 0FB4             ?PNT1: ENT
50 0FB4 F5          PUSH AF
51 0FB5 C5          PUSH BC
52 0FB6 D5          PUSH DE
53 0FB7 E5          PUSH HL
54 0FB8 C1          POP BC
55 0FB9 112800      LD DE,0028H    ; 40
56 0FBC 21D8CF      LD HL,SCRN-40
57 0FBF             ?PNT2: ENT
58 0FBF 19          ADD HL,DE
59 0FC0 05          DEC B
60 0FC1 F2BF0F      JP P,-2

```

\*\* Z80 ASSEMBLER SB-7201 &lt;1Z-013A&gt; PAGE 60

04.07.83

```

01 0FC4 0600      LD B,+0
02 0FC6 09        ADD HL,BC
03 0FC7 D1        POP DE
04 0FC8 C1        POP BC
05 0FC9 F1        POP AF
06 0FCA C9        RET
07 0FCB             ;
08 0FCB             ; VERIFYING
09 0FCB             ;
10 0FCB             ; COMMAND 'V'
11 0FCB             ;
12 0FCB             VRFY: ENT
13 0FCB CD8B05      CALL ?VRFY
14 0FCE DA0701      JP C,?ER
15 0FD1 114209      LD DE,MSGOK
16 0FD4 DF          RST 3
17 0FD5 C3AD00      JP ST1
18 0FD8             ;
19 0FD8             ;
20 0FD8             ; ORG 0FD8H;?CLER
21 0FD8             ;
22 0FD8             ;
23 0FD8             ; CLER ;
24 0FD8             ; B=SIZE
25 0FD8             ; HL=LOW ADR.
26 0FD8             ;
27 0FD8             ?CLER: ENT
28 0FD8 AF          XOR A
29 0FD8 AF          JR +4
30 0FD9 1802      ?CLRFF: ENT
31 0FDB             LD A,FFH
32 0FDB 3EFF      ?DINT: ENT
33 0FDD             LD (HL),A
34 0FDD 77          INC HL
35 0FDE 23          DJNZ -2
36 0FDF 10FC      RET
37 0FE1 C9
38 0FE2             ;
39 0FE2             ; GAP CHECK
40 0FE2             ;
41 0FE2             ;
42 0FE2             GAPCK: ENT
43 0FE2 C5          PUSH BC
44 0FE3 D5          PUSH DE
45 0FE4 E5          PUSH HL
46 0FE5 0101E0      LD BC,KEYPB
47 0FE8 1102E0      LD DE,CSTR
48 0FEB             GAPCK1: ENT
49 0FEB 2664        LD H,100
50 0FED             GAPCK2: ENT
51 0FED CD0106      CALL EDGE
52 0FF0 380B        JR C,GAPCK3
53 0FF2 CD4A0A      CALL DLY3      ; CALL DLY2*3
54 0FF5 1A          LD A,(DE)
55 0FF6 E620        AND 20H
56 0FF8 20F1        JR NZ,GAPCK1
57 0FFA 25          DEC H
58 0FFB 20F0        JR NZ,GAPCK2
59 0FFD             GAPCK3: ENT
60 0FFD C39B06      JP RET3

```

01 1000                    \*                    SKP                    H

```

01 1000      ;
02 1000      ;
03 1000      ;   MONITOR WORK AREA   ;
04 1000      ;   (MZ-700)           ;
05 1000      ;
06 1000      ;
07 1000      ;
08 10F0      ORG   10F0H
09 10F0      SP:   ENT
10 10F0      IBUF: ENT           ; TAPE BUFFER(128B)
11 10F0      ATRB: ENT           ; ATTRIBUTE
12 10F0      DEFS  +1
13 10F1      NAME: ENT           ; FILE NAME
14 10F1      DEFS  +17
15 1102      SIZE: ENT           ; BYTE SIZE
16 1102      DEFS  +2
17 1104      DTADR: ENT          ; DATA ADR
18 1104      DEFS  +2
19 1106      EXADR: ENT          ; EXECUTION ADR
20 1106      DEFS  +2
21 1108      COMNT: ENT          ; COMMENT
22 1108      DEFS  104
23 1170      KANAF: ENT          ; KANA FLAG
24 1170      DEFS  +1
25 1171      DSPXY: ENT          ; DISPLAY CO-ORDINATES
26 1171      DEFS  +2
27 1173      MANG: ENT          ; COLOUMN MANAGEMENT
28 1173      DEFS  +27
29 118E      FLASH: ENT          ; FLASHING DATA
30 118E      DEFS  +1
31 118F      FLPST: ENT          ; FLASSING POSITION
32 118F      DEFS  +2
33 1191      FLSST: ENT          ; FLASING STATUS
34 1191      DEFS  +1
35 1192      FLSDT: ENT          ; CURSOR DATA
36 1192      DEFS  +1
37 1193      STRGF: ENT          ; STRING FLAG
38 1193      DEFS  +1
39 1194      DPRNT: ENT          ; TAB COUNTER
40 1194      DEFS  +1
41 1195      TMCNT: ENT          ; TAPE MARK COUNTER
42 1195      DEFS  +2
43 1197      SUMDT: ENT          ; CHECK SUM DATA
44 1197      DEFS  +2
45 1199      CSMDT: ENT          ; FOR COMPARE SUM DATA
46 1199      DEFS  +2
47 119B      AMPM: ENT          ; AMPM DATA
48 119B      DEFS  +1
49 119C      TIMFG: ENT          ; TIME FLAG
50 119C      DEFS  +1
51 119D      SWRK: ENT          ; KEY SOUND FLAG
52 119D      DEFS  +1
53 119E      TEMPW: ENT          ; TEMPO WORK
54 119E      DEFS  +1
55 119F      ONTYO: ENT          ; ONTYO WORK
56 119F      DEFS  +1
57 11A0      OCTV: ENT          ; OCTAVE WORK
58 11A0      DEFS  +1
59 11A1      RATIO: ENT          ; ONPU RATIO
60 11A1      DEFS  +2

```

\*\* Z80 ASSEMBLER SB-7201 &lt;1Z-013A&gt; PAGE 63

04.07.83

```

01 11A3      BUFR:  ENT      ; GET LINE BUFFER
02 11A3      DEFS    +81
03 11F4      ;
04 11F4      ;
05 11F4      ;      EQU TABLE I/O PORT
06 11F4      ;
07 11F4      ;
08 E000 P    KEYPA: EQU     E000H
09 E001 P    KEYPB: EQU     E001H
10 E002 P    KEYPC: EQU     E002H
11 E003 P    KEYPF: EQU     E003H
12 E002 P    CSTR:  EQU     E002H
13 E003 P    CSTPT: EQU     E003H
14 E004 P    CONTO: EQU     E004H
15 E005 P    CONT1: EQU     E005H
16 E006 P    CONT2: EQU     E006H
17 E007 P    CONTF: EQU     E007H
18 E008 P    SUNDG: EQU     E008H
19 E008 P    TEMP:  EQU     E008H
20 11F4      ;
21 11F4      ;      END

```

\*\* Z80 ASSEMBLER SB-7201 &lt;1Z-013A&gt; PAGE 64

04.07.83

```

#BRK 08B8 #CLR08 09D4 #CLR8 09D5 #MCP 006B ..LPT 017B
.LDE 02A6 .LPT 0176 .MANG 02F3 2HE1 0434 2HEX 041F
??KEY 09B3 ?ADCN 0BB9 ?BEL 0577 ?BELD 0352 ?BLNK 0DA6
?BRK 0A32 ?BRK1 0A48 ?BRK2 0980 ?BRK3 0986 ?CLER 0FD8
?CLRFF 0FDB ?DACL 0BCE ?DINT 0FDD ?DPCT 0DDC ?DSP 0DB5
?ER 0107 ?FLAS 09FF ?FLS 09E3 ?GET 08BD ?GETL 07E6
?KEY 08CA ?KY1 08D6 ?KY2 08DA ?KY5 08FA ?KY55 08FB
?KYGRP 08FE ?KYGRS 0909 ?KYSM 08B3 ?LOAD 05F0 ?LTNL 090E
?MLDY 01C7 ?MODE 073E ?MSG 0893 ?MSGX 08A1 ?NL 0918
?PNT1 0FB4 ?PNT2 0FBF ?PONT 0FB1 ?PRNT 0935 ?PRT 0946
?PRTS 0920 ?PRTT 0924 ?RDD 04F8 ?RDI 04DB ?RSTR 0EE5
?RSTR1 0EE6 ?SAVE 0B92 ?SWEP 0A50 ?TEMP 02E5 ?TMR1 0375
?TMR2 037F ?TMRD 035B ?TMS1 0331 ?TMS2 0344 ?TMS2 0344
?VRFY 0588 ?WRD 0475 ?WRI 0436 ALPH1 0EE2 ALPHA 0EE1
AMPM 119B ASC 03DA ATBL 0A92 ATRB 10F0 AUTO3 07ED
BELL 003E BGETL 012F BRKEY 001E BUFR 11A3 CKS1 0720
CKS2 072F CKS3 0733 CKSUM 071A CLEAR 09DB CLEAR1 09DA
CLRS 0E3A CMY0 005B COMNT 1108 CONTO E004 CONT1 E005
CONT2 E006 CONTF E007 CR 0E5A CR1 0E6A CSMDT 1199
CSTPT E003 CSTR E002 CTBL 0EAA CURS1 0DFF CURS2 0E16
CURS3 0DFF CURS4 0E23 CURS5 0E02 CURSD 0DF8 CURSL 0E25
CURSR 0E0D CURSU 0E05 CURSU1 0E0B DACN1 0BE3 DACN2 0BDF
DACN3 0BE0 DEL 0EF8 DEL1 0F0E DEL2 0F1C DLY1 0759
DLY12 0996 DLY2 0760 DLY3 0A4A DLY4 09A9 DPRNT 1194
DSP01 0DB9 DSP04 0DD0 DSPXY 1171 DSWEP 0B30 DTADR 1104
DUM1 0D88 DUM2 0D3E DUM3 0D37 DUMP 0D29 EDG1 0607
EDG2 0613 EDGE 0601 EXADR 1106 FD 00FF FD1 0106
FD2 0102 FLAS1 097B FLAS2 09EF FLAS3 09F3 FLASH 118E
FLKEY 057E FLPST 118F FLSDT 1192 FLSS1 1191 GAP 077A
GAP1 078E GAP2 0796 GAP3 079C GAPCK 0FE2 GAPCK1 0FEB
GAPCK2 0FED GAPCK3 0FFD GETKY 001B GETL 0003 GETL1 07EA
GETL2 0818 GETL3 085B GETL5 081D GETL6 0865 GETLA 082B
GETLB 0863 GETLC 0822 GETLR 087E GETLU 0876 GETLZ 086C
GOTO 00F3 GRSTAS 0DD4 HEX 03F9 HEX1Y 013D HEXJ 03E5
HL1 041D HLHEX 0410 HOME 0E4D IBUFE 10F0 INST 0F38
INST2 0ECA KANA 0EEE KANAF 1170 KANST E003 KEYPA E000
KEYPB E001 KEYPC E002 KEYPF E003 KSL1 09B7 KSL2 09BC
KTBL 0BEA KTBLC 0CAA KTBLG 0CE9 KTBLGS 0C6A KTBL5 0C2A
LETNL 0006 LLPT 0470 LDAO 0116 LOAD 0111 LONG 0A1A
LPRNT 018F M#TBL 0284 MANG 1173 MCR 07A8 MCR1 07AB
MCR2 07D4 MCR3 07D7 MELDY 0030 MLD1 01D1 MLD2 0205
MLD3 020D MLD4 0211 MLD5 0214 MLDS1 02C4 MLDSP 02BE
MLDST 02AB MONIT 0000 MOT1 06A4 MOT2 06AB MOT4 06B9
MOT5 06D8 MOT7 06B7 MOT8 06D0 MOT9 06D7 MOTOR 069F
MSG 0015 MSG#1 03FB MSG#2 03FD MSG#3 0402 MSG#7 0467
MSG1 0896 MSG#2 09A0 MSG#3 06E7 MSGE1 0147 MSGOK 0942
MSGSV 098B MSGX 0018 MSGX1 08A4 MSGX2 08A7 MST1 0705
MST2 070C MST3 0717 MSTA 0044 MSTOP 0700 MSTP 0047
MTBL 026C NAME 10F1 NL 0009 NLPHL 05FA NOADD 03E2
OCTV 11A0 ONP1 021F ONP2 022C ONP3 0265 ONPU 021C
ONTYD 119F OPTBL 029C PEN 018B PLOT 0184 PMSG 01A5
PMSG1 01A8 PRNT 0012 PRNT2 0967 PRNT3 096C PRNT4 096F
PRNT5 0959 PRNTS 000C PRNTT 000F PRTHL 03BA PRTHX 03C3
PTEST 0155 PTRN 0180 PTST0 015A PTST1 0170 RATI0 11A1
RBY1 0630 RBY2 0649 RBY3 0654 RBYTE 0624 RD1 04E6
RDA 01B6 RDDAT 002A RDINF 0027 RET1 04D2 RET2 0554
RET3 069B RTAPE 050E RTP1 0513 RTP2 0519 RTP3 0532
RTP4 0554 RTP5 0565 RTP6 0572 RTP7 056E RTP8 0553
RTP9 0574 RYTHM 02CB SAV1 0F8E SAVE 0F5E SCRIN 0000
SCR0L 0E6D SG 00F7 SHORT 0A01 SIZE 1102 SLPT 03D5

```

\*\* Z80 ASSEMBLER SB-7201 <1Z-013A> PAGE 65 04.07.83

SP	10F0	SPHEX	03B1	SS	00A2	ST0	0070	ST1	00AD
ST2	00BB	START	004A	STRGF	1193	SUMDT	1197	SUNDG	E008
SV0	0BA2	SV1	0BB5	SWEPO	0A66	SWEPO1	0A64	SWEPO2	0A7F
SWEPO3	0A77	SWEPO4	0A5F	SWEPO9	0A73	SWRK	119D	TEMP	E008
TEMPW	119E	TIMFG	119C	TIMIN	038D	TIMRD	003B	TIMST	0033
TM1	0675	TM2	0678	TM3	0688	TM4	069B	TMARK	065B
TMCNT	1195	TVF1	05B2	TVF2	05B8	TVF3	05CC	TVRFY	05AD
VERFY	002D	VGDF	0747	VRFY	0FCB	VRNS	0BC5	WBY1	076D
WBYTE	0767	WRDAT	0024	WRI1	0444	WRI2	045E	WRI3	0464
WRINF	0021	WTAP1	0494	WTAP2	04A5	WTAP3	04D2	WTAPE	048A
XTEMP	0041								

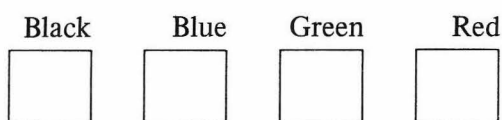


---

## A. 6 Color Plotter-Printer Control Codes

### A.6.1 Control codes used in the text mode

- Text code (\$01)  
Sets the printer in the text mode.
- Graphic code (\$02) ..... Same as the BASIC MODE GR statement.  
Sets the printer in the graphic mode.
- Line up (\$03) ..... Same as the BASIC SKIP-1 statement.  
Moves the paper one line in the reverse direction. The line counter is decremented by 1.
- Pen test (\$04) ..... Same as the BASIC TEST statement.  
Writes the following patterns to start ink flowing from the pens, then sets scale = 1 (40 chr/line), color = 0.



- Reduction scale (\$09) + (\$09) + (\$09)  
Reduces the scale from 1 to 0 (80 chr/line).
- Reduction cancel (\$09) + (\$09) + (\$0B)  
Enlarges the scale from 0 to 1. (40 chr/line).
- Line counter set (\$09) + (\$09) + (ASCII)<sub>2</sub> + (ASCII)<sub>1</sub> + (ASCII)<sub>0</sub> + (\$0D)  
..... Same as the BASIC PAGE statement.  
Specifies the number of lines per page as indicated by 3 bytes of ASCII code. The maximum number of lines per page is 255. Set to 66 when the power is turned on or the system is reset.
- Line feed (\$0A) ..... Same as the BASIC SKIP 1 statement.  
Moves the paper one line in the forward direction. The line counter is incremented by 1.
- Magnify scale (\$0B)  
Enlarges the scale from 1 to 2 (26 chr/line).
- Magnify cancel (\$0C)  
Reduces the scale from 2 to 1.
- Carriage return (\$0D)  
Moves the carriage to the left side of the print area.
- Back space (\$0E)  
Moves the carriage one column to the left. This code is ignored when the carriage is at the left side of the print area.
- Form feed (\$0F)  
Moves the paper to the beginning of the next page and resets the line counter to 0.
- Next color (\$1D)  
Changes the pen to the next color.

### A.6.2 Character scale

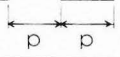
- The character scale is automatically set to 1 (40 chr/line) when the power is turned on. Afterwards, it can be changed by the control codes and commands.
- In the graphic mode, the scale can be changed in the range from 0 to 63.
- The scale is set to 1 when the mode is switched from graphic to text.

## A.6.3 Graphic mode commands

### A. 6. 3. 1 Command type

In the graphic mode, the printer can be controlled by outputting the following commands to the printer.

Words in parentheses are BASIC statements which have the same functions as the graphic mode commands.

Command name	Format	Function
LINE TYPE	$L_p$ ( $p = 0$ to 15)	Specifies the type of line (solid or dotted) and the dot pitch. $p = 0$ : solid line, $p = 1 \sim 15$ : dotted line 
ALL INITIALIZE	A	Sets the printer in the text mode.
HOME (PHOME)	H	Lifts the pen and returns it to the origin (home position).
INITIALIZE (HSET)	I	Sets the current pen location as the origin ( $x = 0$ , $y = 0$ ).
DRAW (LINE)	$Dx, y, \dots, x_n, y_n$ ( $-999 \leq x, y \leq 999$ )	Draws lines from the current pen location to coordinates ( $x_1, y_1$ ), then to coordinates ( $x_2, y_2$ ), and so forth.
RELATIVE DRAW (RLINE)	$J\Delta x, \Delta y \dots \Delta x_n, \Delta y_n$ ( $-999 \leq \Delta x, \Delta y \leq 999$ )	Draws lines from the current pen location to relative coordinates ( $\Delta x_1, \Delta y_1$ ), then to relative coordinates ( $\Delta x_2, \Delta y_2$ ) and so forth.
MOVE (MOVE)	$Mx, y$ ( $-999 \leq x, y \leq 999$ )	Lifts the pen and moves it to coordinates ( $x, y$ ).
RELATIVE MOVE (RMOVE)	$R\Delta x, \Delta y$ ( $-999 \leq \Delta x, \Delta y \leq 999$ )	Lifts the pen and moves it to relative coordinates ( $\Delta x, \Delta y$ ).
COLOR CHANGE (PCOLOR)	$C_n$ ( $n = 0$ to 3)	Changes the pen color to $n$ .
SCALE SET	$S_n$ ( $n = 0$ to 63)	Specifies the character scale.
ALPHA ROTATE	$Q_n$ ( $n = 0$ to 3)	Specifies the direction in which characters are printed.
PRINT	$Pc_1 c_2 c_3 \dots c_n$ ( $n = \infty$ )	Prints characters.
AXIS (AXIS)	$Xp, q, r$ ( $p = 0$ or 1) ( $q = -999$ to 999) ( $r = 1$ to 255)	Draws an X axis when $p = 1$ and a Y axis when $p = 0$ . $q$ specifies the scale pitch and $r$ specifies the number of scale marks to be drawn.

### A. 6. 3. 2 Command format

There are 5 types of command formats as shown below.

1. Command character only (without parameters)

"A", "H", "I"

2. Command character plus one parameter

"L", "C", "S", "Q"

3. Command character plus pairs of parameters

"D", "J", "M", "R"

"," is used to separate parameters, and a CR code is used to end the parameter list.

4. Command plus character string

"P"

The character string is terminated with a CR code.

5. Command plus three parameters

"X"

"," is used to separate parameters.

- ring a CR code.

ode to the text mode.

## A. 7 Notes Concerning Operation

### ■ Data recorder

- Although the data recorder of the MZ-700 is highly reliable, the read/write head will wear out after prolonged use. Further, magnetic particles and dust will accumulate on the head, degrading read/write performance. Therefore, the head must be cleaned periodically or replaced when it becomes worn.
  1. To clean the head, open the cassette compartment, press the **PLAY** key, and wipe the head and pinch roller using a cotton swab. If they are very dirty, soak the cotton swab in alcohol.
  2. When the head becomes worn, contact your dealer. Do not attempt to replace it by yourself.

### ■ Cassette tape

- Any commercially available cassette tape can be used with the MZ-700. However, it is recommended that you use quality cassette tape produced by a reliable manufacturer.
  - Use normal type tapes.
  - Avoid using C-120 type cassette tapes.
  - Use of C-60 or shorter cassette tapes is recommended.
  - Be sure to take up any the slack in the tape with a pencil or the like as shown at right before loading the cassette tape: otherwise, the tape may break or become wound round the pinch roller.

### ● Protecting programs/data from accidental erasure

The data recorder of the MZ-700 is equipped with a write protect function which operates in the same manner as with ordinary audio cassette tape decks.

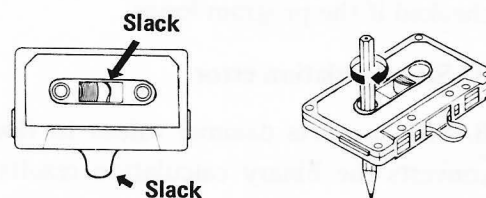
To prevent data from being accidentally erased, remove the record lock-out tab from the cassette with a screwdriver or the like. This makes it impossible to press the **RECORD** key, preventing erasure of, valuable data.

### ■ Other

- See page 109 for commercially available cassette tape decks.

### ■ Display unit

When using a display unit other than one specified for the MZ-700, the screen size must be adjusted. See page 106.



Remove record lock-out tab with a screwdriver.



---

### ■ Color plotter-printer

- Do not rotate the pen drum in the reverse direction when replacing pens.
- Be sure to remove the pens from the pen drum, replace their caps to them, and store them in the case to prevent them from drying out when the printer is not to be used for an extended period of time.
- It takes a certain amount of time for ink on the paper to dry. (The ink is water-soluble.)
- Do not rip off the paper when the printer cover is removed. Hold down the paper holder when ripping off the paper.
- Do not touch the internal mechanism when replacing the pens. Failure to observe this warning may result in damage to the printer.
- The color plotter printer generates sound for a moment when the power is turned on. This is not a problem.
- Letters printed in the 80 character line mode may be difficult to read. In this case, use the 40 character/line mode.
- In the graphic mode, lines printed repeatedly may become blurred. This is particularly liable to occur when a dotted line is printed repeatedly. Due to the characteristics of the ball pen, this is unavoidable.

### ■ Notes concerning software

- It takes about 3 minutes to load the BASIC interpreter.
- The reset switch on the rear panel is to be used in the following cases. (See 3. 1. 1.)  
To stop execution of a BASIC program during normal execution or when the program enters an infinite loop. To return to the program, use the # command. However, the program or hardware should be checked if the program loops.

### ■ BASIC calculation error

- BASIC converts decimal values to floating point binary values before performing calculations, then converts the binary calculation results into decimal numbers for display. This can result in a certain amount of error.

(Example:)

```
PRINT 817.3-810.4
6.899999          ..... Correct result is 6.9.
```

- Approximations are made during calculation of functions and exponentiation.
- The above must be considered when using IF statements.

(Example:)

```
10 A=1/100*100
20 IF A=1 THEN PRINT "TRUE" :GOTO 40
30 PRINT "FALSE"
40 PRINT "A=" ; A
50 END
RUN
FALSE
A=1
```

Although the practical result of the equation in line 10 is 1, this program prints FALSE because of error due to conversion.

---

- **Notes concerning handling**

- **Power switch**

The power switch should be left untouched for at least 10 seconds after being turned on or off. This is necessary to ensure correct operation of the computer. Do not unplug the power cable when the power switch is on: otherwise, trouble may result.

- **Power cable**

Avoid placing heavy objects such as desks on top of the power cable. This may damage the power cable, possibly resulting in a serious accident. Be sure to grasp the cable by the plug when unplugging it.

- **Power supply voltage**

The power supply voltage is 240/220 VAC. The computer may not operate properly if the voltage is too high or too low. Contact your dealer for assistance if you experience this problem.

- **Ventilation**

Many vents are provided in the cabinet to prevent overheating. Place the computer in a well ventilated place, and do not cover it with a cloth. Do not place any objects on the left side of the computer, since this is where the vents for the power supply unit are located.

- **Humidity and dust**

Do not use the computer in a damp or dusty places.

- **Temperature**

Do not place the computer near heaters or in places where it may be exposed to direct sunlight; failure to observe this precaution may result in damage to the computer's components.

- **Water and foreign substances**

Water and other foreign substances (such as pins) entering the computer will damage it. Unplug the power cable immediately and contact your dealer for assistance if such an accident occurs.

- **Shock**

Avoid subjecting the computer to shock; strong shocks will damage the computer permanently.

- **Trouble**

Stop immediately operation and contact your dealer if you note any abnormality.

- **Prolonged disuse**

Be sure to unplug the power cable if the computer is not to be used for a prolonged period of time.

- **Connection of peripheral devices**

Use only parts and components designated by Sharp when connecting any peripheral devices, otherwise, the computer may be damaged.

- **Dirt**

Wipe the cabinet with a soft cloth soaked in water or detergent when it becomes dirty. To avoid discoloration of the cabinet, do not use volatile fluids such as benzene.

---

- **Noise**

It is recommended that a line filter be used when the computer is used in a place where high level noise signals may be present in the AC power. (A line filter can be obtained from your Sharp dealer). Move the signal cables as far as possible from the power cable and other electrical appliances.

- **RF interference**

Interference with TV or radio reception may occur due to the RF signal generated by the computer if it is used near a TV or radio set. TV sets generate a strong magnetic field which may result in incorrect operation of the computer. If this occurs, move the TV set at least 2 to 3 meters away from the computer.

This apparatus complies with requirements of EEC directive 76/889/EEC.





**SHARP CORPORATION**  
**OSAKA, JAPAN**

Printed in Japan  
Gedruckt in Japan  
Imprimé au Japon

3G 105203-I  
TINSE1066ACZZ