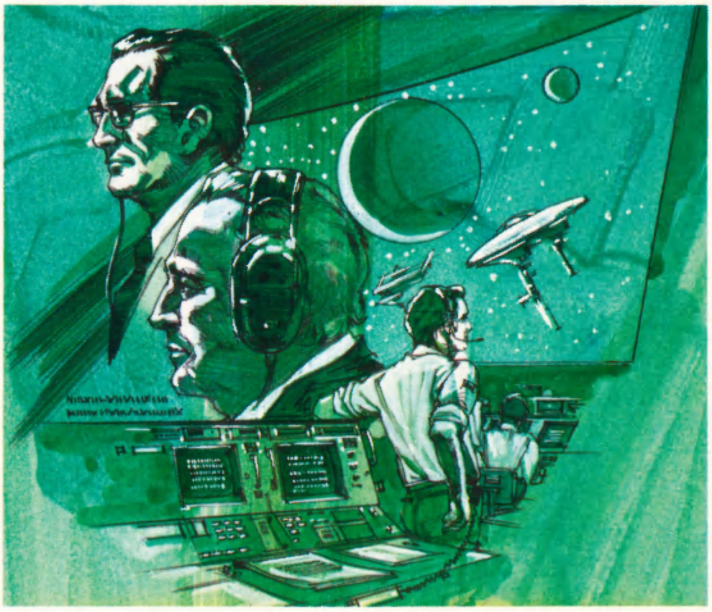
Personal Computer

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.

This apparatus complies with requirements of EEC directive 76/889/EEC.

Das Gerät stimmt mit den Bedingungen der EG-Richtlinien 76/889/EWG überein.

Cet appareil répond aux spécifications de la directive CCE 76/889/CCE.

Dit apparaat voldoet aan de vereisten van EEG-reglementen 76/889/EEG.

Apparatet opfylder kravene i EF direktivet 76/889/EF.

Quest'apparecchio è stato prodotto in conformità alle direttive CEE 76/889/CEE.

Personal Computer

MZ-700

Owner's Manual

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.

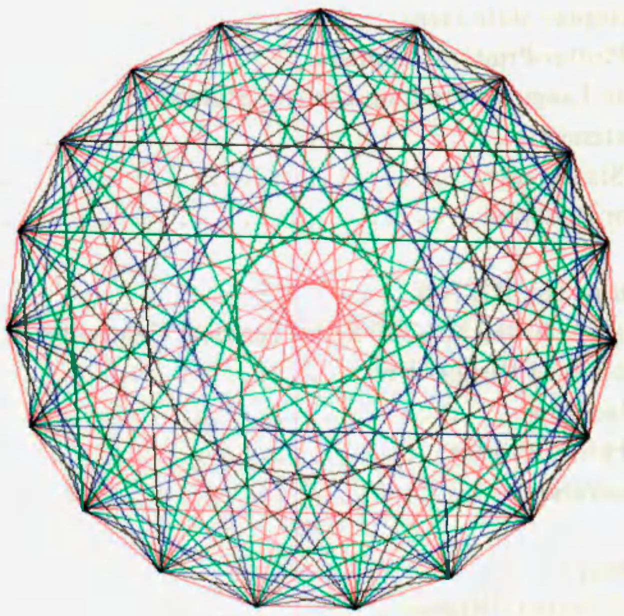
(2) All system software for the Sharp MZ-700 series personal computer has been developed by the Sharp Corporation, and all rights to such software are reserved. Reproduction of the system software or the contents of this book is prohibited.

(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.
Not withstanding 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.

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

APPENDICES

INDEX

[BASIC COMMANDS] ( ) is abbreviated format

THE WORLD OF MZ-700 SERIES PERSONAL COMPUTER

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.

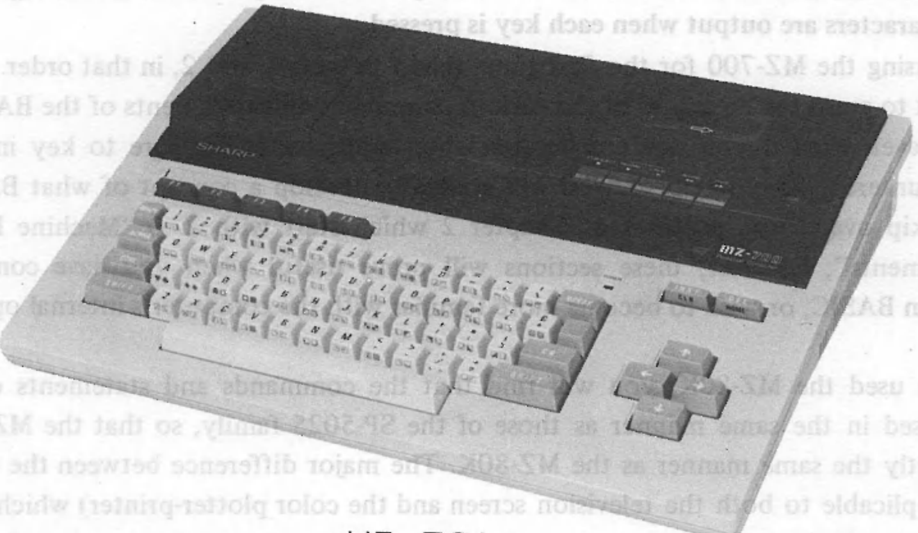
People use words and expressions to convey meanings.

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-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.**

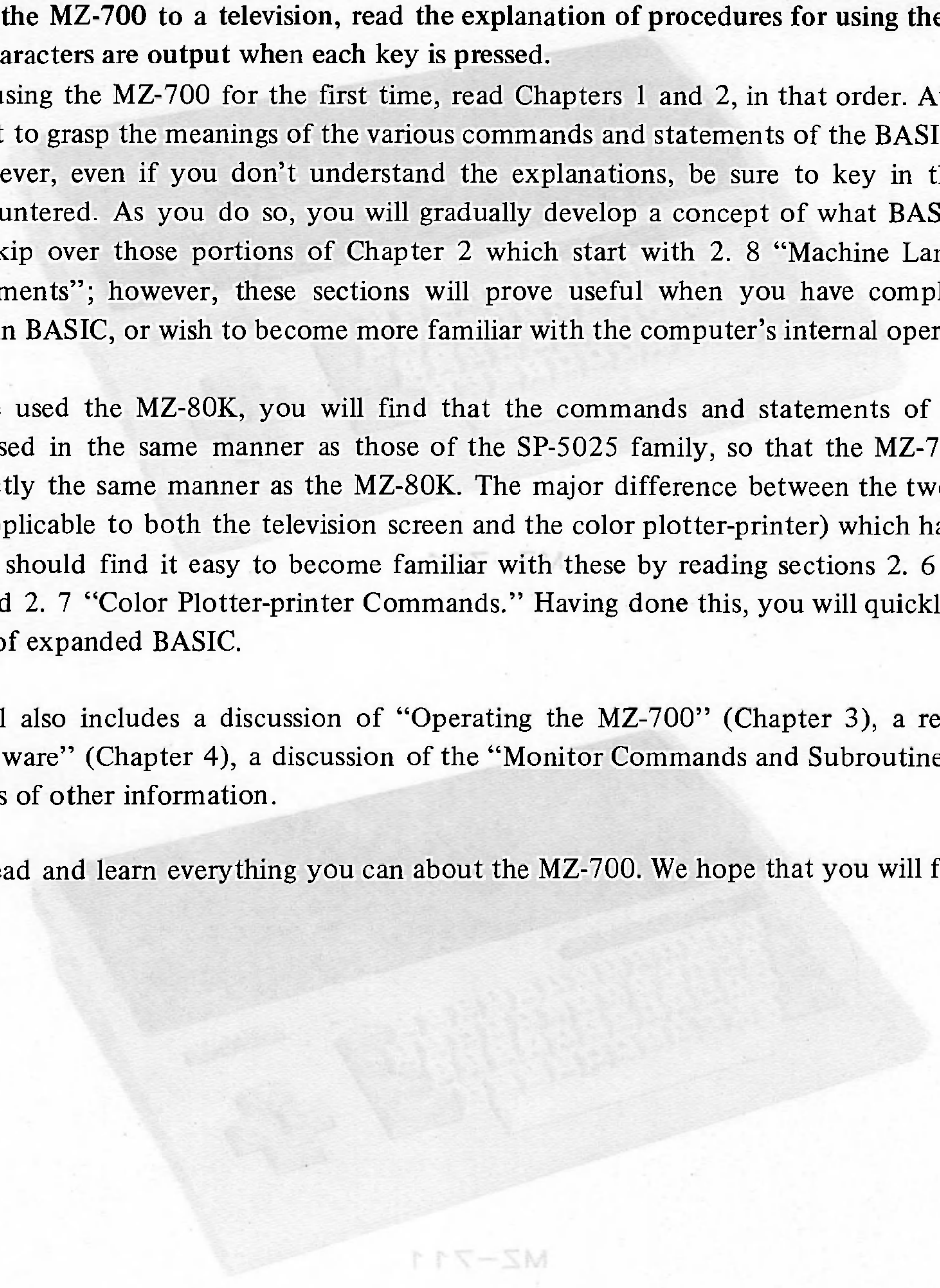
If you are using the MZ-700 for the first time, read Chapters 1 and 2, in that order. At first, you may find it difficult to grasp the meanings of the various commands and statements of the BASIC programming language; however, even if you don't understand the explanations, be sure to key in the examples 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 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 followiŋg 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[※1] and press the CR[※2] key. After doing this, the message ⊥ PLAY appears on the screen.

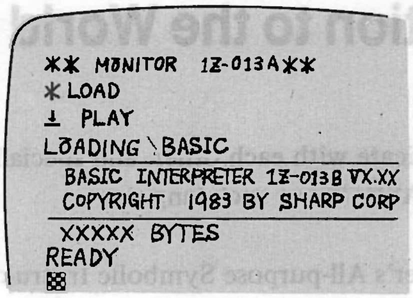
(6) Press the data recorder's PLAY button; the cassette tape starts moving and loading of the BASIC language begins.

(7) After loading has been completed, the message READY is displayed and the cursor starts to blink again.

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.



```
** 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.

1.3.3 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.

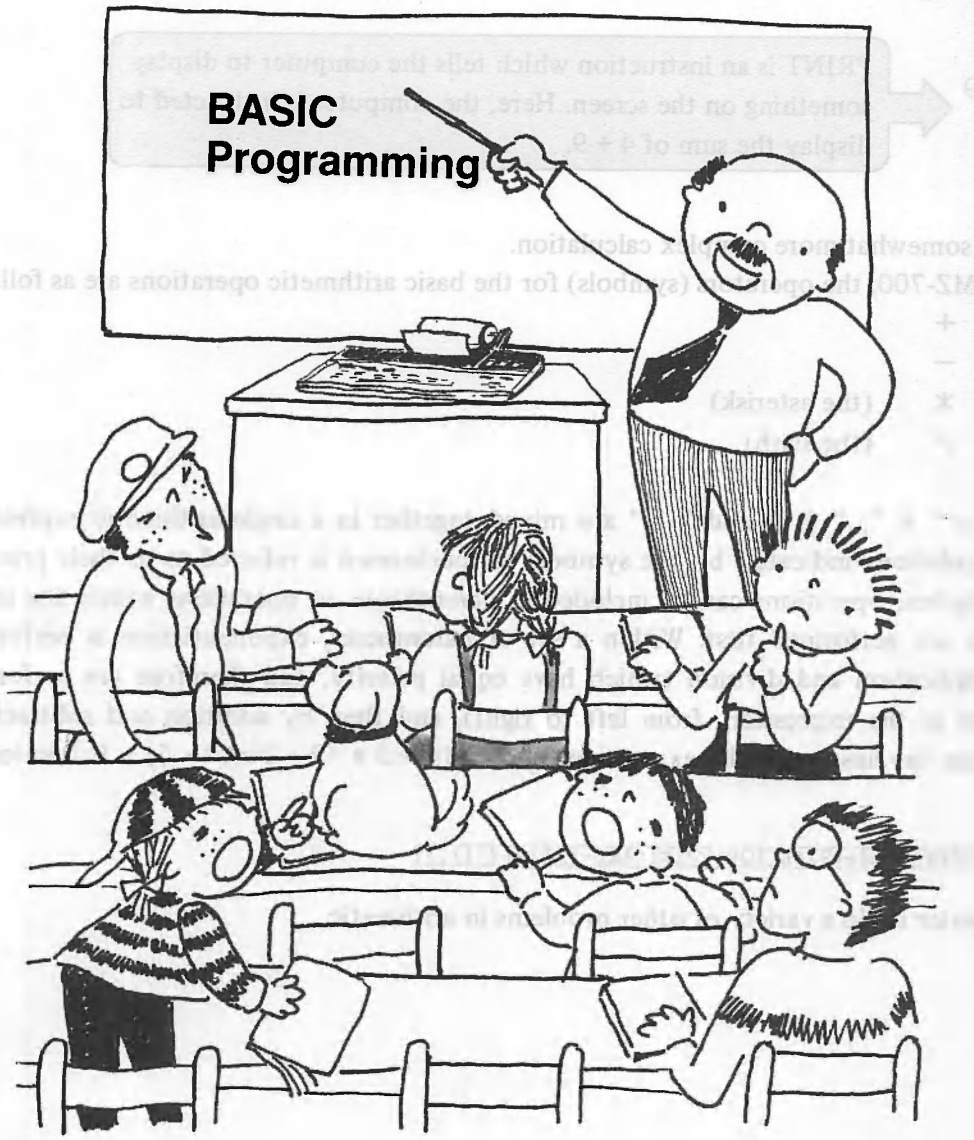
(5) Type in LIST and press the CR key. This lists the lines of the program on the screen one after another. (Output of the list can be temporarily stopped at any time by pressing the space bar.)

(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.

Chapter 2

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

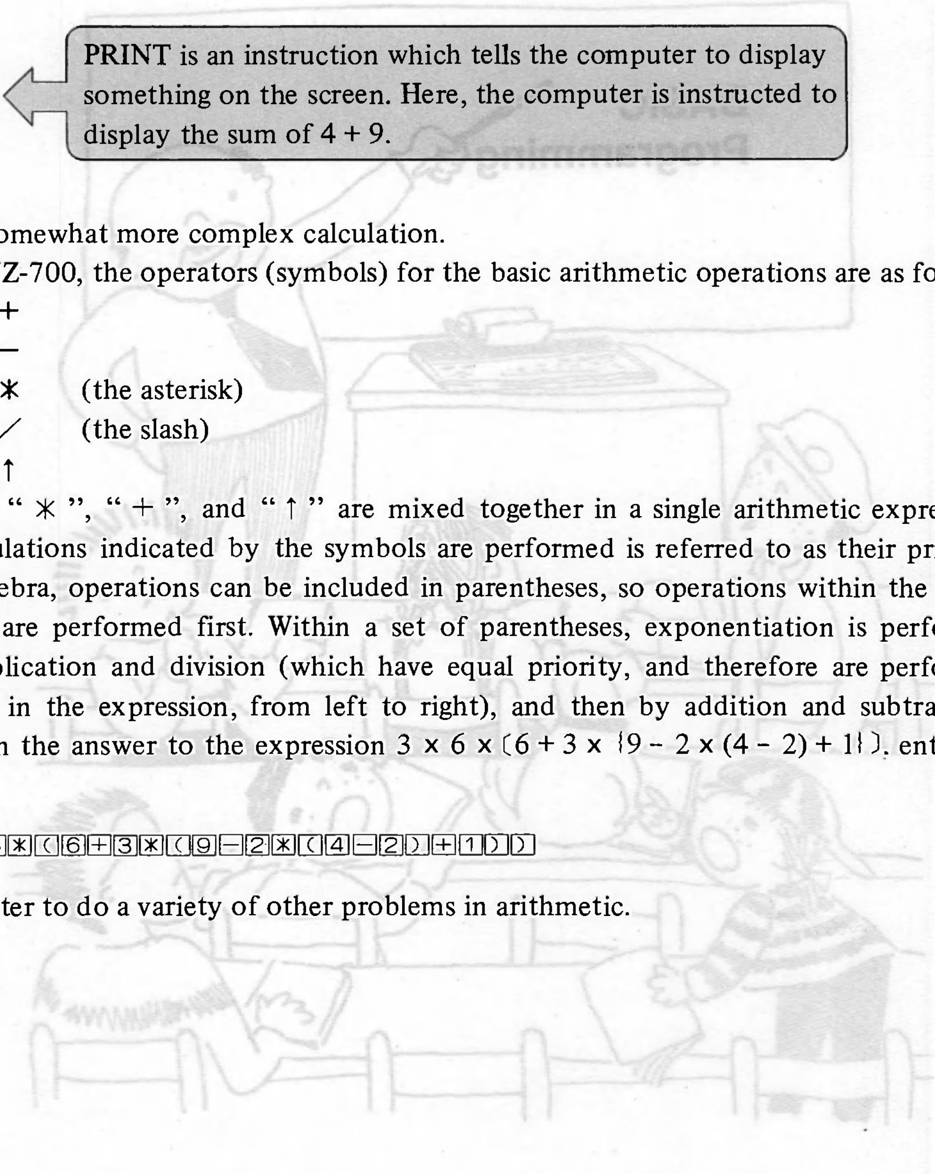
As you can see, the computer doesn't do anything when it is presented with a problem in this form; your computer and an ordinary calculator are completely different in this respect, and instructions must be entered in a form which can be understood by the computer (i.e, in the form prescribed by the 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 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 x 6 x 〔6 + 3 x {9 – 2 x (4 – 2) + 1}〕. enter the following.

PRINT 3✱6✱(6+3✱(9−2✱(4−2)+1))

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

[EXERCISE]	[ANSWER]
1. $\frac{6+4}{6-4}$	PRINT (6+4)/(6-4) 5
2. $3\times\{5+9\times(9-2)-\frac{6}{4-2}\}+5$	PRINT 3*(5+9*(9-2)-6/(4-2)) +5 200
3. $(3+4)\times(5+6)$	PRINT (3+4)*(5+6) 77
4. $\frac{10+20}{6}\times(2+3)$	PRINT (10+20) /6*(2+3) 25
5. $\frac{10+20}{6\times(2+3)}$	PRINT (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"ABCDEFG"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=␣13................ (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.)

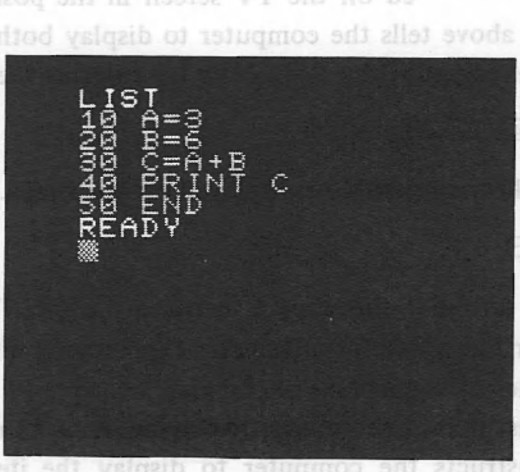
Type in the following program exactly as shown.

```
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.

[R][U][N][CR]

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 [3][0][∗] 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 connot 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.

LIST	-30CR	Lists all lines of the program to line 30.
LIST	30-CR	Lists all lines from line 30 to the end of the program.
LIST	30-50CR	Lists all lines from line 30 to line 50.
LIST	30CR	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^{12}
7. 65E−9 ····················· 7. 65×10^{-9} E indicates the exponent.
14. 8E9 ························ 14. 8×10^{9}

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.

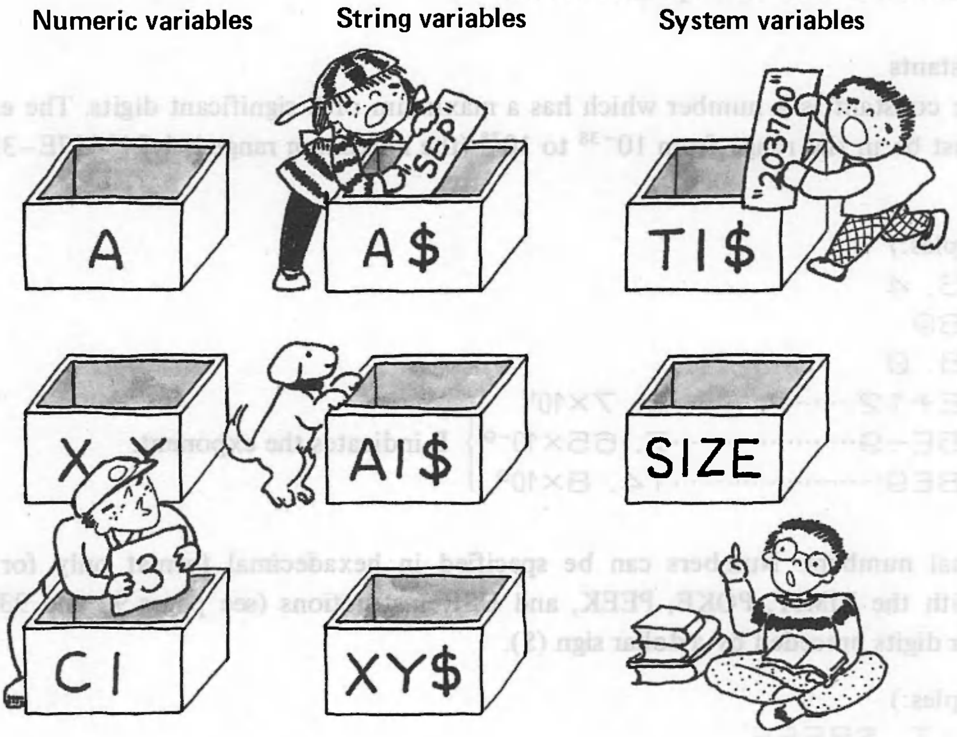
String constants

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"
"123456789Ø"
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:)

1Ø A$="ABCD"········Substitutes the character string ABCD into string variable A$.
2Ø B$="XYZ"··············Substitutes the character string XYZ into string variable B$.
3Ø C$=A$+B$··············Substitutes the sum of string variables A$ and B$ (ABCDXYZ) into string variable C$.
4Ø PRINT C$··············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:)

1Ø TI$="Ø135ØØ"··· This statement assigns the value corresponding to 1:35:00 A.M. to system variable TI$ and sets the system clock to that time.
2Ø 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.

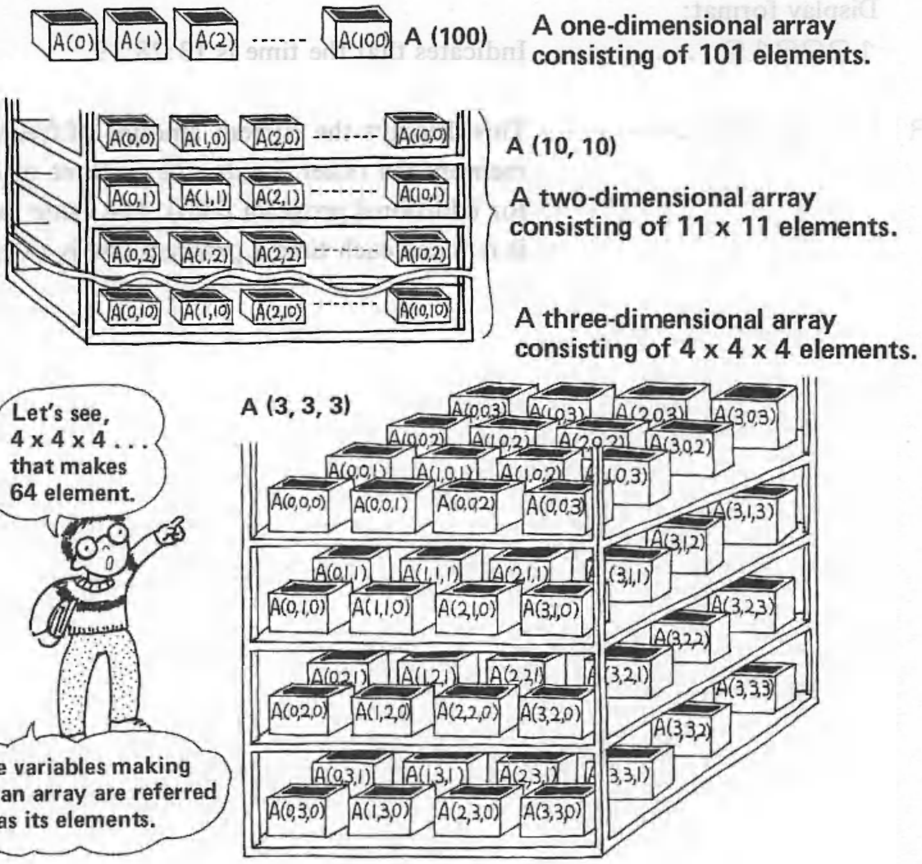
Such shelves of boxes are set up simply by executing an instruction which declares that they exist; this is referred to as making an array declaration. The array declaration specifies the number of boxes which are to be included in each set of shelves (i.e., the size of the shelves) and the manner in 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(0) A(1) A(2) ------ A(100) A (100)
A one-dimensional array consisting of 101 elements.
A(0,0) A(1,0) A(2,0) ------ A(10,0)
A(0,1) A(1,1) A(2,1) ------- A(10,1)
A(0,2) A(1,2) A(2,2) A(10,2)
A(0,10) A(1,10) A(2,10) ------- A(10,10)
A (10, 10)
A two-dimensional array consisting of 11 x 11 elements.
A three-dimensional array consisting of 4 x 4 x 4 elements.
A (3, 3, 3)
A(0,0,0) A(0,0,1) A(0,0,2) A(0,0,3)
A(0,1,0) A(1,1,0) A(2,1,0) A(3,1,0)
A(0,2,0) A(1,2,0) A(2,2,0) A(3,2,0)
A(0,3,0) A(1,3,0) A(2,3,0) A(3,3,0)
A(3,0,3) A(3,1,3) A(3,2,3) A(3,3,3)
Let's see, 4 x 4 x 4 . . . that makes 64 element.
The variables making up an array are referred to as its elements.

(Example 1)

```
1Ø  DIM  A(5)························ Declares 1-dimensional numeric array A with 6 elements.
2Ø  DIM  X$(8)····················· Declares 1-dimensional string array X$ with 9 elements.

1Ø  DIM  A(5), X$(8)········· Performs the same function as lines 10 and 20 above.
```

(Example 2)

```
1Ø  DIM  B(5, 5)················· Declares 2-dimensional numeric array B with 6 x 6
                                  elements.
2Ø  DIM  Y$(5, 8)·············· Declares 2-dimensional string array Y$ with 6 x 9 elements.
1Ø  DIM  B(5, 5), Y$(5, 8), A(5), X$(8)··········· Declares two numeric arrays
                                                   and two string arrays.
```

(Example 3)

```
1Ø  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.

```
1Ø  DIM  A(2), B$(2)
2Ø  A(Ø)=26
3Ø  A(1)=9
4Ø  A(2)=-1ØØ
5Ø  B$(Ø)="ABC"
6Ø  B$(1)="XYZ"
7Ø  B$(2)="MZ-7ØØ"
8Ø  PRINT  A(1)
9Ø  PRINT  B$(2)
1ØØ  PRINT  A(2)
11Ø  PRINT  B$(Ø)+B$(1)
12Ø  PRINT  A(Ø)
```

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	↑	Exponentiation	X ↑ 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↑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" ↲ ——→ 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 TI$ is "000000".

■ **Music function**

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

■ **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.)

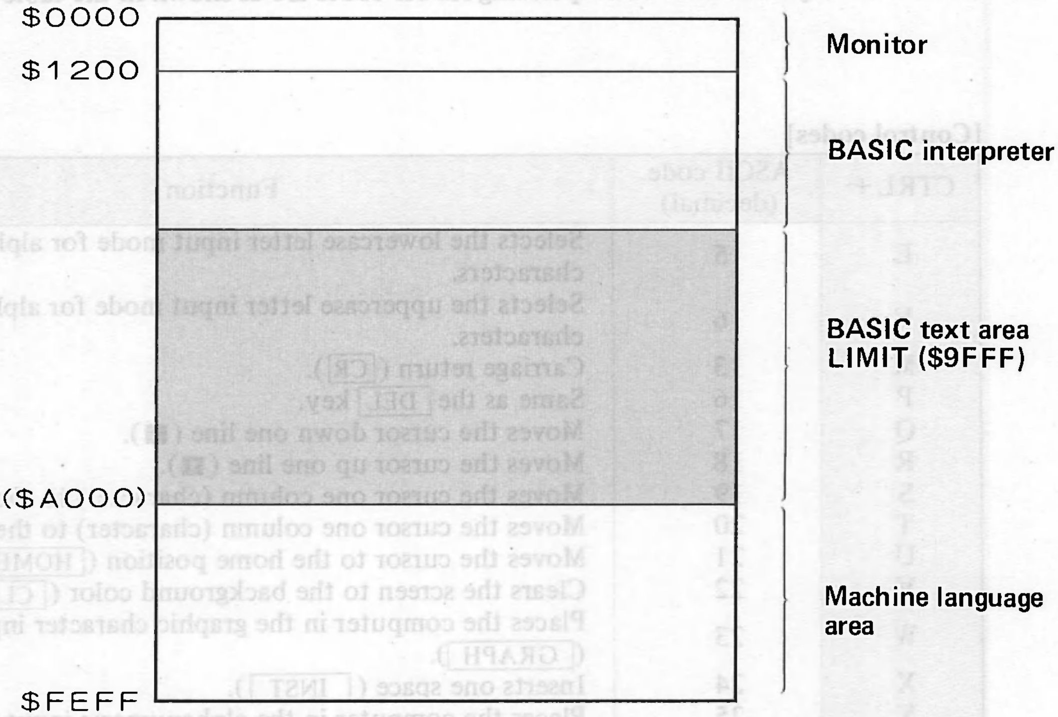
Format LOAD or LOAD "filename"

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 progam 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.

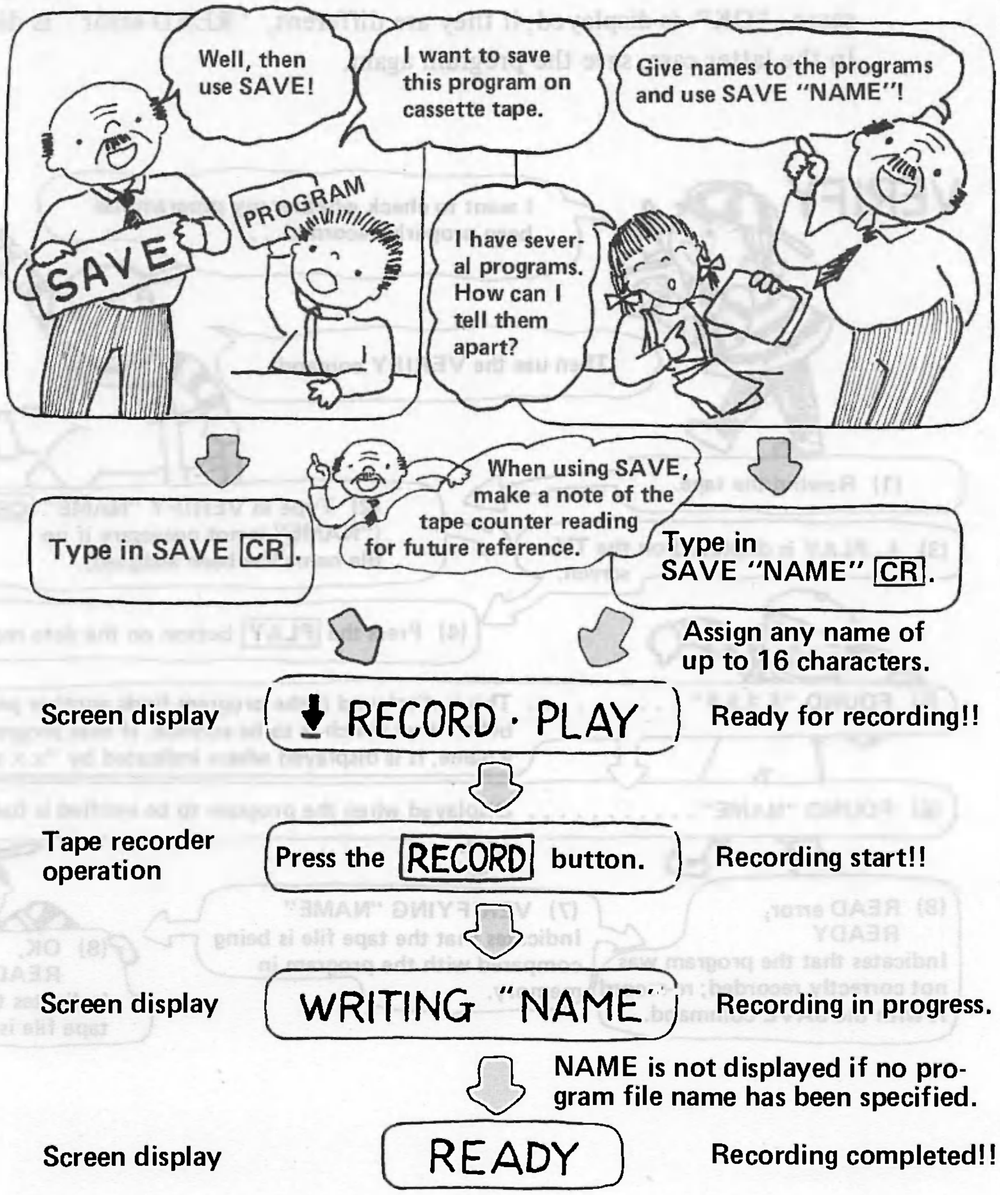


Note: The lower limit of the BASIC text area shifts according to the size the program text loaded.

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.


Well, then use SAVE!
I want to save this program on cassette tape.
Give names to the programs and use SAVE "NAME"!
SAVE
PROGRAM
I have several programs. How can I tell them apart?
When using SAVE, make a note of the tape counter reading for future reference.
Type in SAVE CR.
Type in SAVE "NAME" CR.
Assign any name of up to 16 characters.
Screen display
↓ RECORD · PLAY
Ready for recording!!
Tape recorder operation
Press the RECORD button.
Recording start!!
Screen display
WRITING "NAME"
Recording in progress.
NAME is not displayed if no program file name has been specified.
Screen display
READY
Recording completed!!

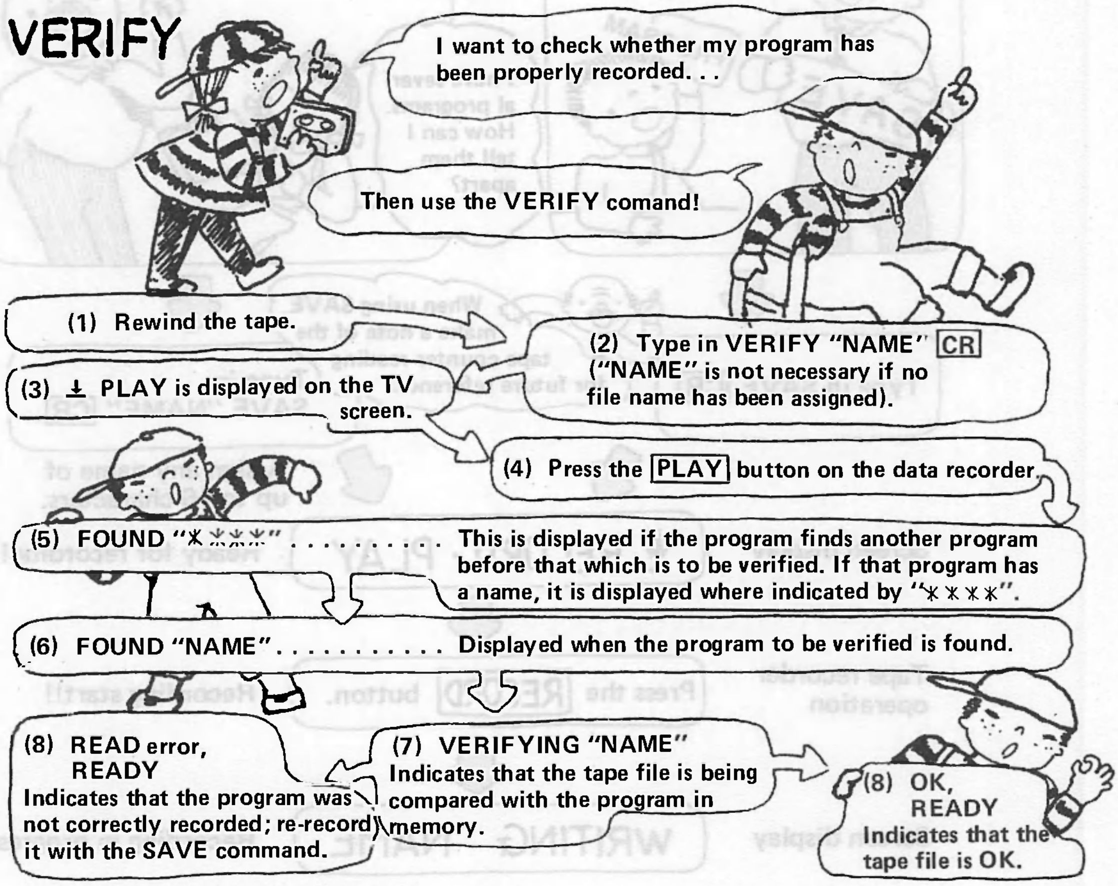
Note This command saves only the BASIC program text (i.e., the program text displayed by executing the LIST command); it does not save any machine language program in the machine language area.

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.

2. 3. 1. 3 VERIFY . . . . (abbreviated format: V.)

Format: **VERIFY** or **VERIFY** "filename"

Function: 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.


VERIFY
I want to check whether my program has been properly recorded. . .
Then use the VERIFY comand!
(1) Rewind the tape.
(2) Type in VERIFY "NAME" CR ("NAME" is not necessary if no file name has been assigned).
(3) ⊥ PLAY is displayed on the TV screen.
(4) Press the PLAY button on the data recorder.
(5) FOUND "****" . . . . This is displayed if the program finds another program before that which is to be verified. If that program has a name, it is displayed where indicated by "****".
(6) FOUND "NAME". . . . . Displayed when the program to be verified is found.
(7) VERIFYING "NAME" Indicates that the tape file is being compared with the program in memory.
(8) READ error, READY Indicates that the program was not correctly recorded; re-record it with the SAVE command.
(8) OK, READY Indicates that the tape file is OK.

2.3.2 Text editing commands

2. 3. 2. 1 AUTO .... (abbreviated format: A.)

Format

AUTO or AUTO Ls, n

Ls ······ Starting line number

n ·········· Line number increment

Function

This command automatically generates program line numbers during entry of BASIC program statements.

Example

(Example 1)

```
AUTO↵
1Ø························↵
2Ø························↵
3Ø························↵
```

(Example 2)

```
AUTO  3ØØ, 5↵
3ØØ·····················↵
3Ø5·····················↵
31Ø·····················↵
```

Automatically generates program line numbers with an increment of 5, starting with line 300.

(Example 3)

```
AUTO  1ØØ↵
1ØØ·····················↵ ⎫
11Ø·····················↵ ⎬  Generates program line numbers with an increment
12Ø·····················↵ ⎭  of 10, starting with line 100.
```

(Example 4)

```
AUTO, 2Ø↵
1Ø······················↵ ⎫
3Ø······················↵ ⎬ ......... Generates program line numbers with an increment
5Ø······················↵ ⎭           of 20, starting with line 10.
```

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

2. 3. 2. 2 DELETE .... (abbreviated format: D.)

Format

```
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.
```

Example

(Example 1)

```
DELETE  15Ø-35Ø↵ ······Deletes all program lines from 150 to 350.
```

(Example 2)

```
DELETE  -1ØØ↵ ·············Deletes all program lines up to line 100.
```

(Example 3)

```
DELETE  4ØØ-↵ ················Deletes all program lines from 400 to the end
                               of the program.
```

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

Format

```
LIST
LIST Ls-Le
LIST Ls-
LIST -Le
```

.... Ls indicates the starting line number and Le indicates the ending line number.

Function

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

```
LIST ↵ ......................... Lists the entire program.
LIST -3Ø↵ ................. Lists all lines of the program to line 30.
LIST 3Ø-↵ ................. Lists all lines of the program from line 30 to the end.
LIST 3Ø-5Ø↵ ........... Lists all lines of the program from line 30 to line 50.
LIST 3Ø↵ ..................... 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 1ØØ .... Renumbers the lines of the current program in memory so that they start with 100 and incremented in units of 10.

RENUM 1ØØ, 5Ø, 2Ø .... 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)
 5Ø  A=1             )          1ØØ  A=1
 6Ø  A=A+1           |          12Ø  A=A+1
 7Ø  PRINT  A        |------>   14Ø  PRINT  A
 1ØØ  GOTO  6Ø       )          16Ø  GOTO  12Ø
```

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
● Program execution stopped by pressing SHIFT + BREAK .	● Before a RUN command has been executed.
● Program execution stopped by a STOP command.	● "READY" displayed due to an error occurring during program execution.
● Program execution stopped by pressing SHIFT + BREAK while the program was a waiting input for an INPUT statement.	● Cassette tape operation interrupted by pressing SHIFT + BREAK .
	● Program execution stopped during execution of a MUSIC statement.
	● Program execution stopped and "READY" displayed after execution of an END statement.

2. 3. 3. 3 BYE .... (abbreviated format: B.)

Format
BYE

Function
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
KEY LIST

Function
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(1Ø)="LOAD"
READY
▩
```

2.3.4 Assignment statement

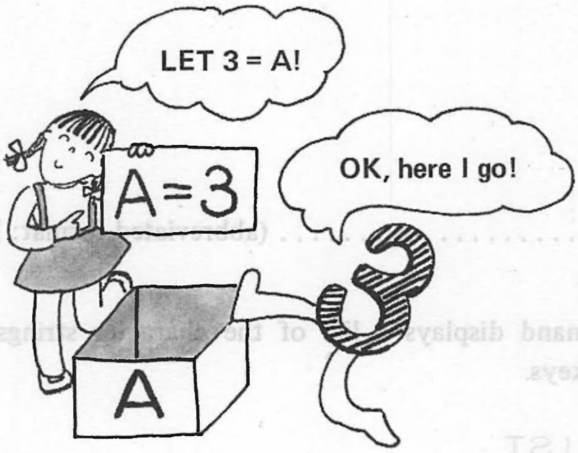
LET

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

Format

$$\left\{\begin{matrix}\textbf{PRINT}\\ \\ ?\end{matrix}\right\}\left\{\begin{matrix}\text{variable}\\ \text{constant}\\ \text{expression}\end{matrix}\right\}\left\langle\left\{\begin{matrix};\\ ,\end{matrix}\right\}\left\{\begin{matrix}\text{variable}\\ \text{constant}\\ \text{expression}\end{matrix}\right\}\cdots\cdots\right\rangle$$

Function

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.)

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.

Real number format

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

```
-1.9999
 63598757
 Ø.ØØØØØØØ1 ..............................................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 10^9
 .513606E-2Ø.............................................. 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 "C" Clears the entire screen and moves the cursor to the home position (the upper left corner of the screen).

PRINT "H" 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 "🅲⬇⬇⬇⬇⬇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 ⟨ $\left\{\begin{matrix};\\,\end{matrix}\right\}$ 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 ↲
**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 ↲
␣␣£123
```

(g) $$

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

(h) ↑↑↑↑

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

(Example:)

```
10 A = 51123
20 PRINT USING "##.###↑↑↑↑" ; A
RUN ↲
␣5.112E+04
```

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

(i) 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 ↲
␣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
```

$$\underbrace{\text{20 PRINT A; B;}}_{\text{Normal PRINT function}}\ \underbrace{\text{USING "\#\#\#\#" ; C}}_{\text{USING function}}$$

```
RUN ↵
–12␣14␣␣12
```

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

Format

```
        ⎧ numeric variable ⎫                                 ⎧ numeric variable ⎫
INPUT   ⎨ string variable  ⎬ ... or INPUT "character string" ; ⎨ string variable  ⎬ ...
        ⎩ array element    ⎭                                 ⎩ array element    ⎭
```

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

```
1Ø  INPUT A, B
2Ø  C=A+B
3Ø  PRINT C
4Ø  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.

```
1Ø  INPUT"A=";A
2Ø  INPUT"B=";B
3Ø  PRINT"A+B=";A+B
4Ø  PRINT"A-B=";A-B
5Ø  PRINT"A*B=";A*B
6Ø  PRINT"A/B=";A/B
7Ø  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:)

```
1Ø PRINT "NEXT GO? (Y OR N) "
2Ø GET A$
3Ø IF A$="Y" THEN 5Ø····· In the example above, execution
                          jumps from line 30 to line 50 if the
                          value of variable A$ is "Y".
4Ø GOTO 2Ø··············· Line 40 unconditionally transfers exe-
5Ø PRINT "PROGRAM END"    cution to line 20.
6Ø END
```

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   ⎧ numeric variable ⎫   ⟨ ⎧ numeric variable ⎫             ⟩
  ⋮    ⎨ string variable  ⎬ , ⟨ ⎨ string variable  ⎬ , ..................
  ⋮    ⎩ array element    ⎭     ⎩ array element    ⎭
  ⋮
DATA   ⎧ numeric constant ⎫ , ⟨ ⎧ numeric constant ⎫ , ..................  ⟩
       ⎩ string constant  ⎭     ⎩ string constant  ⎭
```

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)

```
1Ø READ A, B, C, D
2Ø PRINT A;B;C;D
3Ø END
4Ø DATA 1Ø, 1ØØ, 5Ø, 6Ø
RUN ↵

 1Ø  1ØØ  5Ø  6Ø
```

.... 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)

```
1Ø READ X$, A1, Z$
2Ø PRINT X$;A1;Z$
3Ø END
4Ø 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)

```
1Ø  DIM  A(2)
2Ø  READ  A(Ø), A(1), A(2)
3Ø  PRINT  A(Ø); A(1); A(2)
4Ø  END
5Ø  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)

```
1Ø  READ  A
2Ø  READ  B
3Ø  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

```
1Ø DATA 1, 2, 3
2Ø DATA "AA", "BB"
3Ø READ X, Y
4Ø READ Z, V$
.....................................
1ØØ RESTORE
11Ø READ A, B, C, D$, E$
.....................................
2ØØ READ I, J
21Ø RESTORE
22Ø READ M, N
23Ø RESTORE 26Ø
24Ø READ O, P
25Ø DATA 1, 2, 3, 4
26Ø DATA -1, -2, -3, -4
```

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

```
1Ø  X=33*RND(1)
2Ø  FOR A=1  TO  5
3Ø  READ  M$
4Ø  PRINT  TAB(Ø) ; "◆" ; TAB(X) ; M$ ;
5Ø  PRINT  TAB(37) ; "◆"
6Ø  NEXT  A
7Ø  Y=1Ø*RND(1)
8Ø  FOR  A=1  TO  Y
9Ø  PRINT  TAB(Ø) ; "◆" ;
1ØØ  PRINT  TAB(37) ; "◆" : NEXT
11Ø  RESTORE : GOTO  1Ø
12Ø  DATA"  ◢◙◣", "●▒▒▒▒●"
13Ø  DATA"  ▒▒▒", "●▒▒▒●"
14Ø  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.

```
1Ø A=Ø
2Ø FOR N=Ø  TO  1Ø  STEP  2
3Ø A=A+1
4Ø  PRINT  "N=";N,
5Ø  PRINT  "A=";A
6Ø  NEXT  N
```

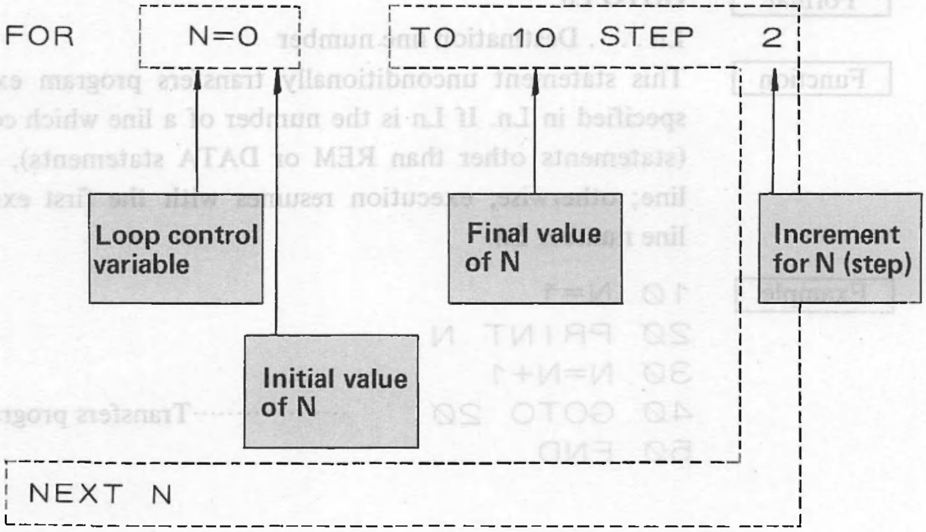
(1) In the program above, 0 is assigned to N as the initial value.

(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

```
1Ø FOR X=1 TO 9 ─────┐
2Ø FOR Y=1 TO 9 ─┐   │
3Ø PRINT X*Y;    │   │
4Ø NEXT Y ───────┘   │
5Ø PRINT             │
6Ø NEXT X ───────────┘
7Ø END
```

Inner loop

Outer loop

```
FOR A=1 TO 3 ─────┐
FOR B=1 TO 5 ───┐ │
FOR C=1 TO 7 ─┐ │ │
..............│ │ │
NEXT C ───────┘ │ │  ┐
NEXT B ─────────┘ │  │ NEXT
NEXT A ───────────┘  ┘ 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 1Ø
│┌→FOR J=K TO K+S
└┴─NEXT J
```

× Different control variables must be used in each loop.

```
┌→FOR I=1 TO 1Ø
│┌→FOR J=K TO K+5
└┼─NEXT I
 └─NEXT J
```

× 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

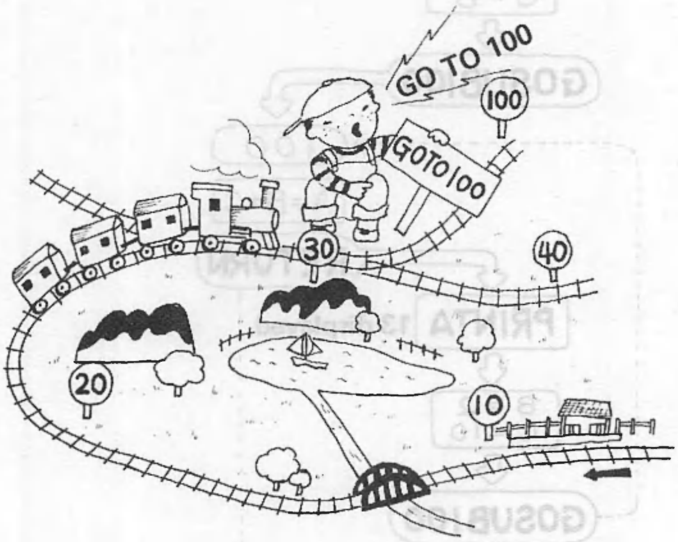
Function

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

```
1Ø N=1
2Ø PRINT N
3Ø N=N+1
4Ø GOTO 2Ø         ..................Transfers program execution to line 20.
5Ø END
```

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.

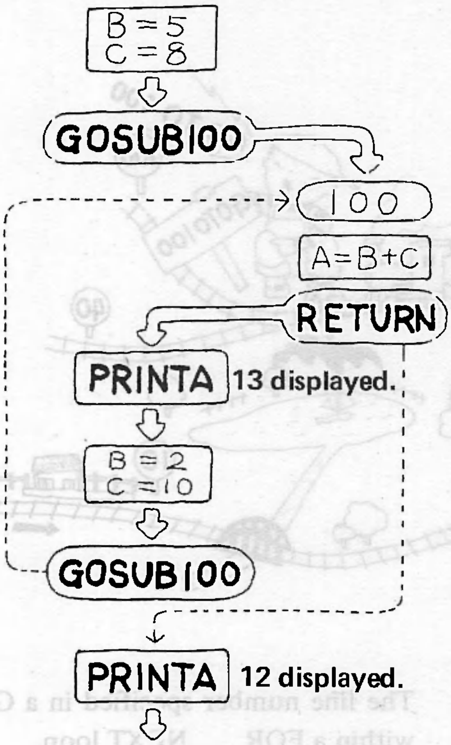
Example

```
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

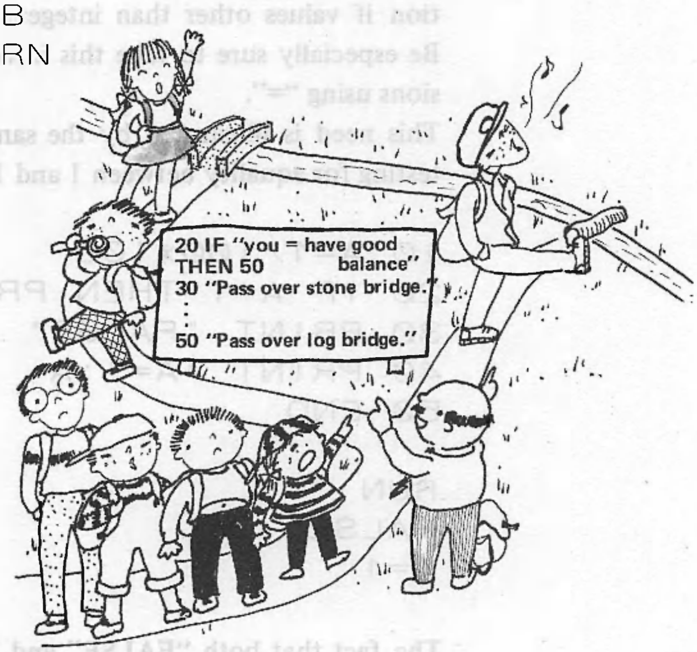
IF . . . THEN statements are used to control branching of program execution according to the result of a logical or relational expression. When the result of such an expression is true, statements following THEN are executed. If a line number is specified following THEN, program execution jumps to that 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 1ØØ
IF……THEN GOTO or IF……GOTO
IF……THEN PRINT or IF……THEN ?
IF……THEN A=5*7 assignment
IF……THEN I=1Ø:J=5Ø
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 then 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.
	<= or =<	IF A<=X THEN…	If the value of variable A is less than or equal to X, execute the statements following THEN.
Logical expressions	*	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.
	+	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 * 100.

```
1Ø A=1/1ØØ*1ØØ
2Ø IF A=1 THEN PRINT "TRUE":GOTO 4Ø
3Ø PRINT "FALSE"
4Ø PRINT "A=";A
5Ø END

RUN
FALSE
A=1
```

The fact that both "FALSE" and " A = 1" are displayed as the result of this program showns 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.

```
2Ø IF ABS(A-1)< .1E-8 THEN PRINT "TRUE":
   GOTO 4Ø
```

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

```
1Ø  G=Ø:N=Ø
2Ø  INPUT  "GRADE=";X
3Ø  IF  X=999  GOTO  1ØØ
4Ø  T=T+X:N=N+1
5Ø  GOTO  2Ø
1ØØ  PRINT  "-----------"
11Ø  PRINT  "TOTAL:";T
12Ø  PRINT  "NO. PEOPLE:";N
13Ø  PRINT  "AVERAGE:";T/N
14Ø  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

```
1Ø  INPUT  " X= ";X
2Ø  IF  X<Ø  GOSUB  1ØØ:PRINT"X<Ø"
3Ø  IF  X=Ø  GOSUB  2ØØ:PRINT"X=Ø"
4Ø  IF  X>Ø  GOSUB  3ØØ:PRINT"X>Ø"
5Ø  PRINT  "-------------------------"
6Ø  GOTO  1Ø
1ØØ  PRINT  " * PROGRAM LINE 1ØØ ":RETURN
2ØØ  PRINT  " * PROGRAM LINE 2ØØ ":RETURN
3ØØ  PRINT  " * PROGRAM LINE 3ØØ ":RETURN
```

2. 3. 6. 7 ON~GOTO .... (abbreviated format: ON~G.)

Format

ON e **GOTO** Lr_1 < , Lr_2 , Lr_3 , ... , Lri >

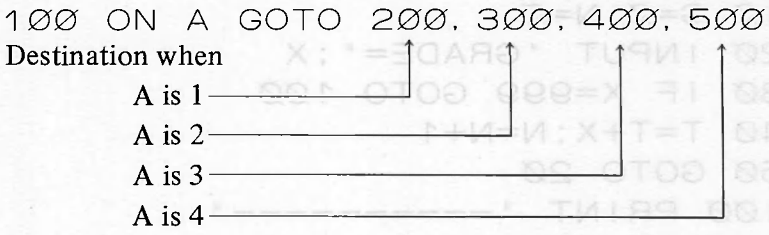
e . . . Numeric variable, array element, or expression

Lri . 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:



```
1ØØ  ON  A  GOTO  2ØØ, 3ØØ, 4ØØ, 5ØØ
```

Destination when

A is 1

A is 2

A is 3

A is 4

Example

```
1Ø   INPUT"NUMBER" ;A
2Ø  ON  A  GOTO  5Ø, 6Ø, 7Ø
5Ø  PRINT"XXX" :GOTO  1Ø
6Ø  PRINT"YYY" :GOTO  1Ø
7Ø  PRINT"ZZZ" :GOTO  1Ø
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 , ... , Lri >

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

Lri . 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$=" H RM ":PRINT"🅲"
50 INPUT"WHAT DAY?";X$
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), ... ai (im) >
DIM b_1 $(i_1, j_1) <, b_2$ (i_2, j_2), ... bi (in, jn) >

ai .... 1-dimensional array name (list)
bi .... 2-dimensional array name (table)
im, in, jn .... 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:)

```
1Ø DIM A(1ØØ)
2Ø FOR J=Ø TO 1ØØ
3Ø READ A(J)
4Ø NEXT J
5Ø DATA 5, 3Ø, 12, ......
```

(Examples:)

```
1Ø DIM A$ (1), B$ (1), C$ (1)
2Ø FOR J=Ø TO 1 : READ A$ (J), B$ (J)
3Ø C$ (J) =A$ (J) +"  "+B$ (J)
4Ø PRINT A$ (J), B$ (J), C$ (J)
5Ø NEXT J
6Ø END
7Ø 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 . . . Name assigned to the function being defined (one uppercase letter from A to Z)
x . . . Argument (variable name)
e . . . 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 2X² + 3X+ 1 as FNA
                             (X).
DEF  FNE (V) =1/2*M*V↑2............... Defines 1/2MV² as FNE (V).
1Ø  DEF  FNB (X) =TAN (X−PAI (1) /6)
2Ø  DEF  FND (X) =FNB (X)/ C+X··Defines function FNB using the
                             function defined on line 10.
```

(Incorrect definitions)

```
1Ø  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 x 2, and 3.5 x 3.

```
1Ø  DEF  FNE(V) =1/2*M*V↑2
2Ø  M=5. 5:V=3. 5
3Ø  PRINT FNE(V), FNE(V*2), FNE (V*3)
4Ø  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:)

```
1Ø  DEF  KEY (1) =" INPUT "..............Defines key F1 as INPUT
2Ø  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.

```
1Ø  A=5
2Ø  B=8                    1Ø  A=5:B=8:C=A*B:PRINT C
3Ø  C=A*B
4Ø  PRINT  C
```

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

```
1Ø  INPUT  A
2Ø  B=Ø
3Ø  IF  99<A  THEN  B=1
4Ø  PRINT  B
5Ø  END
```

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.

```
1Ø  INPUT  A:B=Ø:IF  99<A  THEN  B=1:PRINT  B
2Ø  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 ↲ .

Example

```
1Ø READ A, B
2Ø X=A*B
3Ø STOP
4Ø Y=A/B
5Ø PRINT X, Y
6Ø DATA 15, 5
7Ø END
RUN
BREAK IN 3Ø
```

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×××× 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.

```
1Ø  PRINT  "C"
2Ø  DIM  C$(1Ø), D(1Ø), E(1Ø), T$(1Ø)
3Ø  FOR  I=1  TO  1Ø:READ  C$(I), D(I) :NEXT  I
4Ø  PRINT"ENTER NEW YORK TIME  (HOUR, MINUT
E, SECOND) "
5Ø  INPUT  B$:TI$=B$:PRINT"C"
6Ø  PRINT  "H" :T$(1)=TI$
7Ø  FOR  I=1  TO  1Ø
8Ø  E(I)=VAL(LEFT$(T$(1), 2))+D(I)
9Ø  IF  E(I)=24  THEN  E(I)=Ø
1ØØ  IF  E(I)<Ø  THEN  E(I)=24+E(I)
11Ø  T$(I)=STR$(E(I))+RIGHT$(T$(1), 4)
12Ø  IF  LEN(T$(I))=5  THEN  T$(I)="Ø"+T$(I)
13Ø  PRINT  C$(I);TAB(15);LEFT$(T$(I), 2);
14Ø  PRINT": ";MID$(T$(I), 3, 2);":";RIGHT$(
T$(I), 2);
15Ø  NEXT  I:GOTO  6Ø
16Ø  DATA  NEW  YORK, Ø, MOSCOW, 8, RIO  DE  JANE
IRO, 2
17Ø  DATA  SYDNEY, 15, HONOLULU, -5, LONDON, 5,
CAIRO, 7
18Ø  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

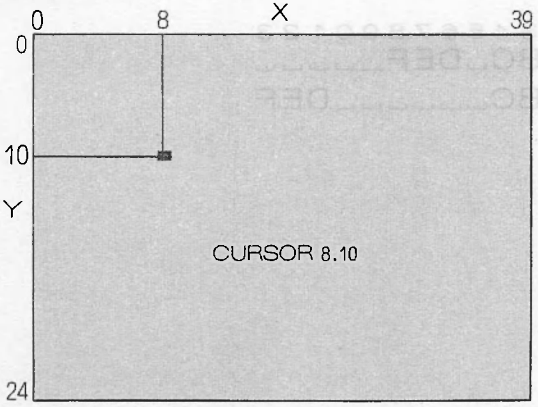
The following program moves an asterisk (∗) about on the screen as the cursor keys are pressed.

```
10  X=0:Y=0
15  PRINT"🅲"
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 (1Ø) ; "ABC"
0123456789012 ⇐ 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"
01234567890
␣␣␣␣␣XYZABC
```

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"
01234567
␣␣␣␣␣ABC
```

(Example 2)

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

```
1Ø  ?  TAB (2) ; "ABC" ; TAB (6) ; "DEF"
2Ø  ?  SPC (2) ; "ABC" ; SPC (6) ; "DEF"
01234567890123
␣␣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
SET X, Y < , C > 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 ≦ X ≦ 79 0 ≦ Y ≦ 49
RESET X, Y 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 ≦ X ≦ 79 0 ≦ Y ≦ 49

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

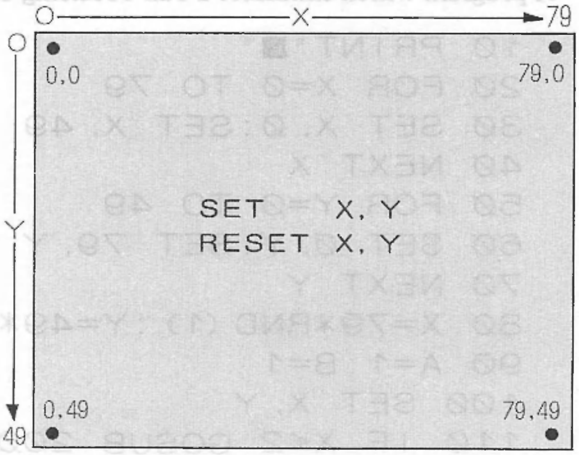
(0)	Black
(1)	Blue
(2)	Red
(3)	Purple
(4)	Green
(5)	Light blue
(6)	Yellow
(7)	White

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 use to produce a wide variety of interesting effects; some examples are introduced below.

1. Turning on one dot on the sceen.

```
1Ø  PRINT"🅲"
2Ø  X=79:Y=49
3Ø  SET  X, Y, 2  <──── Turns dots on.
4Ø  RESET  X, Y  <──── Turns dots off.
5Ø  GOTO  3Ø
```



2. Coloring the entire screen white.

```
1Ø  PRINT"🅲"
2Ø  FOR  X=Ø  TO  79
3Ø  FOR  Y=Ø  TO  49
4Ø  SET  X, Y, 7
5Ø  NEXT  Y, X
6Ø  GOTO  1Ø
```

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
```

Note   As to JOY command, refer to the instruction manual of Joy Stick.

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).

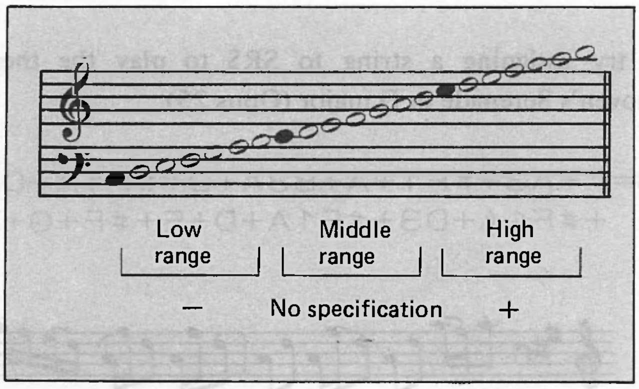
The format for specification of each note is as follows:

< 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

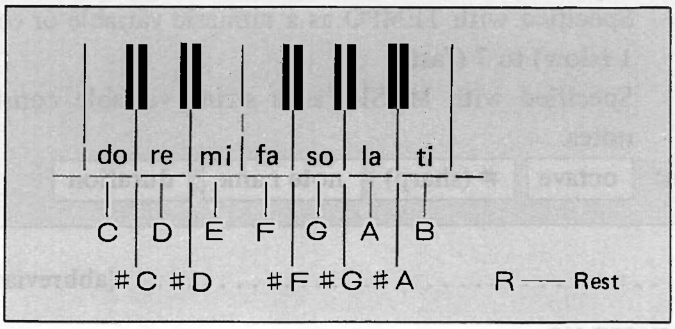


Note specification

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

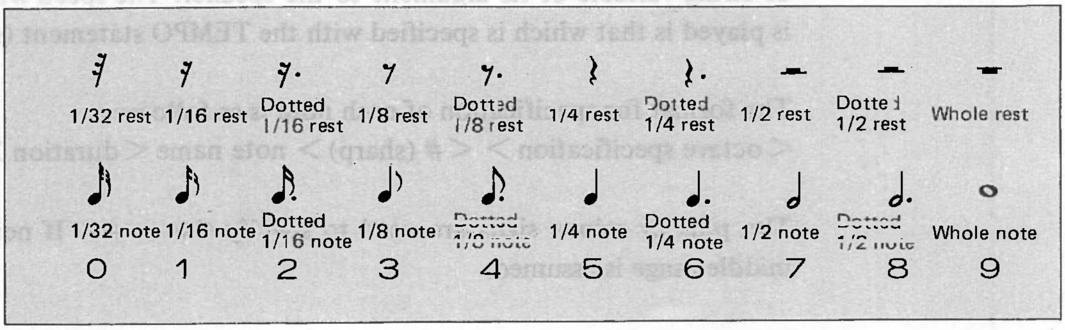
CDEFGAB # R

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	1/8 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 sucessive 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.

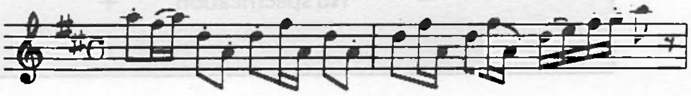
Sound volume

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.

```
3Ø  TEMPO 1   Slowest tempo . . . . (Lento, adagio)
3Ø  TEMPO 4   Medium tempo . . . . (Moderato);
                        four times as fast as TEMPO 1.
3Ø  TEMPO 7   Fastest tempo . . . . . (Motto allegro, presto);
                        seven times as fast as TEMPO 1.
```

Example

```
1Ø  REM  Chopin's mazourka
2Ø  MM$="A3":M1$="A5+#C3+D+E+#F+G+#FØ+G+#
    F4+E3+D+#CB"
3Ø  M2$="A3+D2RØ+D1+E2+D+#C3B+#C7+#C3"
4Ø  M3$="A3+#C2RØ+#C1+D2+#CB3A+D7+D3"
5Ø  TEMPO 3
6Ø  MUSIC MM$, M1$, M2$, M1$, M3$, M1$, M2$, M1
    $, M3$
7Ø  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
WOPEN (abbreviated **W.**)	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.
PRINT/T (abbreviated **?/T**)	PRINT/T d_1 < , d_2, d_3 , . . . dn > dn . . . . Numeric data or string data	Writes data to cassette tape in the same format as it would be displayed by the PRINT statement.
ROPEN (abbreviated **RO.**)	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.
INPUT/T (abbreviated **I./T**)	INPUT/T v_1 < , v_2, v_3 , . . . vn > vn . . . . 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).
CLOSE (abbreviated **CLO.**)	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.

```
1Ø  DIM N$(5)
2Ø  N$(1)="BACH"
3Ø  N$(2)="MOZART"
4Ø  N$(3)="BEETHOVEN"
5Ø  N$(4)="CHOPIN"
6Ø  N$(5)="BRAHMS"
7Ø  WOPEN"GREAT MUSICIAN"
8Ø  FOR J=1 TO 5
9Ø  PRINT/T N$(J)
1ØØ  NEXT J
11Ø  CLOSE
12Ø  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.

```
2ØØ  DIM M$(5)
21Ø  ROPEN "GREAT MUSICIAN"
22Ø  FOR K=1 TO 5
23Ø  INPUT/T M$(K)
24Ø  PRINT M$(K)
25Ø  NEXT K
26Ø  CLOSE
27Ø  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 mathemetics to a sequential data cassette file.

```
1Ø  INPUT"ENTER NO.  OF  STUDENTS";N
2Ø  DIM  N$(N), K(N), E(N)
3Ø  DIM  R(N), S(N)
4Ø  A$="GRADE  IS"
5Ø  FOR  X=1  TO  N
6Ø  PRINT:PRINT  "STUDENT  NO.";X
7Ø  INPUT"ENTER  STUDENT  NAME: ";N$(X)
8Ø  PRINT  "ENG  ";A$;:INPUT K(X)
9Ø  PRINT  "FREN";A$;: INPUT  E(X)
1ØØ  PRINT  "SCI  ";A$;: INPUT  R(X)
11Ø  PRINT  "MATH";A$;: INPUT  S(X)
12Ø  NEXT  X
13Ø  WOPEN"GRADES"              Opens data file "GRADES" for output on cassette tape.
14Ø  PRINT/T  N                 Writes the number of students in the class to the file.
15Ø  FOR  X=1  TO  N
16Ø  PRINT/T N$(X),K(X),E(X),R(X), S(X)      Writes grades
                                             to the file.
17Ø  NEXT  X
18Ø  CLOSE           Closes the cassette file.
19Ø  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.

```
1Ø  ROPEN"GRADES"                  Opens cassette file "GRADES" for input.
2Ø  INPUT/T  N                     Reads the number of people in the class.
3Ø  DIM  N$(N), K(N), E(N)
4Ø  DIM  R(N), S(N)
5Ø  FOR  X=1  TO  N                      Reads student names and the grades for
6Ø  INPUT/T  N$(X),K(X)                  English.
7Ø  INPUT/T E(X),R(X),S(X)               Reads the grades for French, science
                                         and mathematics.
8Ø  NEXT  X
9Ø  CLOSE                                Closes the file.
1ØØ  PRINT  TAB(1Ø); "ENG";
11Ø  PRINT  TAB(15);"FREN";
12Ø  PRINT  TAB(2Ø); "SCI  ';
13Ø  PRINT  TAB(25); "MATH"
14Ø  FOR  X=1  TO  N
15Ø  PRINT  N$(X);TAB(1Ø);K(X);
16Ø  PRINT  TAB(15);E(X);
17Ø  PRINT  TAB(2Ø);R(X);
18Ø  PRINT  TAB(25);S(X);
19Ø  PRINT  TAB(3Ø); (K(X)+E(X)+R(X)+S(X))/4
2ØØ  K(Ø)=K(Ø)+K(X) :E(Ø)=E(Ø)+E(X)
21Ø  R(Ø)=R(Ø)+R(X) :S(Ø)=S(Ø)+S(X)
22Ø  NEXT  X
23Ø  PRINT  TAB(1Ø);K(Ø)/N;TAB(15);E(Ø)/N;
24Ø  PRINT  TAB(2Ø);R(Ø)/N;TAB(25);S(Ø)/N
25Ø  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 vairable 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. $A = \begin{cases} 1 & (X > 0) \\ 0 & (X = 0) \\ -1 & (X < 0) \end{cases}$ Example: 1 is assigned to variable A when A = SGN (0.4) is executed.

Integer conver-sion	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

Trigono-metric functions	SIN (X)	A = SIN (X)
		A=SIN(30*PAI(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 π/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*PAI(1)/180)
			Assigns the cosine of X (where X is in radians) to variable A.

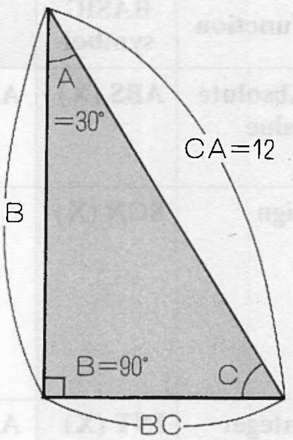
	TAN (X)	A = TAN (X)
		A=TAN(Y*PAI(1)/180)
			Assigns the tangent of X (where X is in radians) to variable A.

	ATN (X)	A = ATN (X)
		A=180/PAI(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.
Exponen-tiation	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 circum-ference 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 funcions

(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°, 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.

```
1Ø  A=3Ø:B=9Ø:CA=12
2Ø  AB=CA*COS (A*PAI(1)/18Ø)
3Ø  BC=CA*SIN (A*PAI(1)/18Ø)
4Ø  S=AB*BC/2
5Ø  C=18Ø-A-B
6Ø  PRINT "AB=";AB, "BC=";BC, "CA=";CA
7Ø  PRINT "AREAS=";S
8Ø  PRINT "A=";A, "B=";B, "C=";C
9Ø  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 fine angle C from sides AB and BC.

```
1Ø  A=3Ø:B=9Ø:CA=12
2Ø  AB=CA*COS (A*PAI(1)/18Ø)
3Ø  BC=CA*SIN (A*PAI(1)/18Ø)
4Ø  S=AB*BC/2
5Ø  C=ATN (AB/BC)*18Ø/PAI(1)
6Ø  PRINT  "AB=";AB, "BC=";BC, "CA=";CA
7Ø  PRINT  "AREAS=";S
8Ø  PRINT "A=";A, "B=";B, "C=";C
9Ø  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 ≦ 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 unpredicatable 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.

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

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

```
INT(6*RND(1))→0、1、2、3、4、5
```

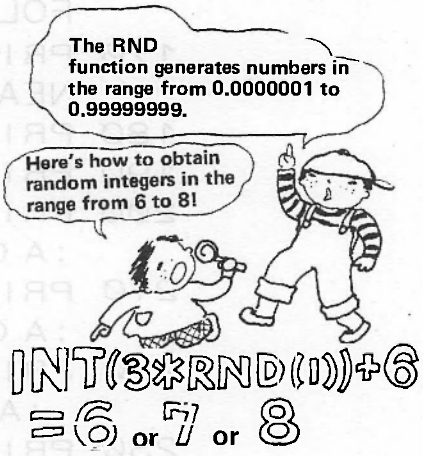
The result will be an integer between 0 and 5; now 1 is added to obtain the numbers which correspond to the number of dots on each of the 6 sides of a die.

```
INT(6*RND(1))+1→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!
INT(3*RND(1))+6
=6 or 7 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.

```
10Ø DIM A(3),L$(4)
2Ø FOR J=1 TO 4
3Ø READ L$(J):NEXT J
4Ø FOR J=1 TO 3
5Ø A(J)=INT(2Ø*RND(1))+1
6Ø NEXT J
7Ø IF A(1)>=A(2)+A(3)  GOTO 4Ø
8Ø IF A(2)>=A(1)+A(3)  GOTO 4Ø
9Ø IF A(3)>=A(1)+A(2)  GOTO 4Ø
1ØØ W=(A(1)+A(2)+A(3))/2
11Ø T=W :FOR J=1 TO 3
12Ø T=T*(W-A(J)):NEXT J
13Ø SS=SQR(T):S=INT(SS)
14Ø IF SS-S>Ø.5 THEN S=S+1
15Ø PRINT "C↓↓↓↓"
16Ø PRINT "    SOLVE FOR THE AREA OF THE
    FOLLOWING TRIANGLE"
17Ø PRINT "    ROUND YOUR ANSWER TO THE
    NEAREST WHOLE NUMBER"
18Ø PRINT
19Ø PRINT TAB(8);"A"
2ØØ PRINT TAB(8);"◪◩"        ;TAB(15);L$(1)
    ;A(1)
21Ø PRINT TAB(7);"◪  ◩"      ;TAB(15);L$(2)
    ;A(2)
22Ø PRINT TAB(6);"◪    ◩"    ;TAB(15);L$(3)
    ;A(3)
23Ø PRINT TAB(5);"◪      ◩"
24Ø PRINT TAB(3);"B◪        ◩C"
25Ø PRINT TAB(4);"□□□□□□□□□□"
26Ø PRINT "↓↓↓"
27Ø PRINT TAB(3);L$(4);
28Ø INPUT Y
29Ø IF Y=S  THEN PRINT "       OK!!":GOTO
    4Ø
3ØØ IF Y<S  THEN PRINT"  TOO SMALL!"
    :GOTO  32Ø
31Ø PRINT "  TOO LARGE!"
32Ø PRINT "↑↑";
33Ø PRINT TAB(24);SPC(25):PRINT "▌";
34Ø GOTO 27Ø
35Ø DATA LENGTH SIDE AB=,LENGTH SIDE BC=
36Ø DATA LENGTH SIDE CA=,AREAS OF TRIAN
    GLE ABC IS
```

Note

Note than 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 funcion 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)

```
1Ø  A$="ABCDEFG"
2Ø  PRINT  LEN(A$)

RUN
7
```

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

```
1Ø  ?"🅲":?"ENTER  3OR  MORE  ASTERISKS"
2Ø  INPUT  A$
3Ø  FOR  I=1  TO  LEN(A$)-2
4Ø  PRINT  TAB(2) ; "*" ; SPC(LEN(A$)-2) ; "*"
5Ø  NEXT  I
6Ø  PRINT  TAB(2) ;A$:GOTO  2Ø
```

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

```
1Ø  S$="SHARP  BASIC"
2Ø  FOR  I=1  TO  LEN(S$)
3Ø  ?  RIGHT$(S$, I)
4Ø  NEXT  I
5Ø  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$

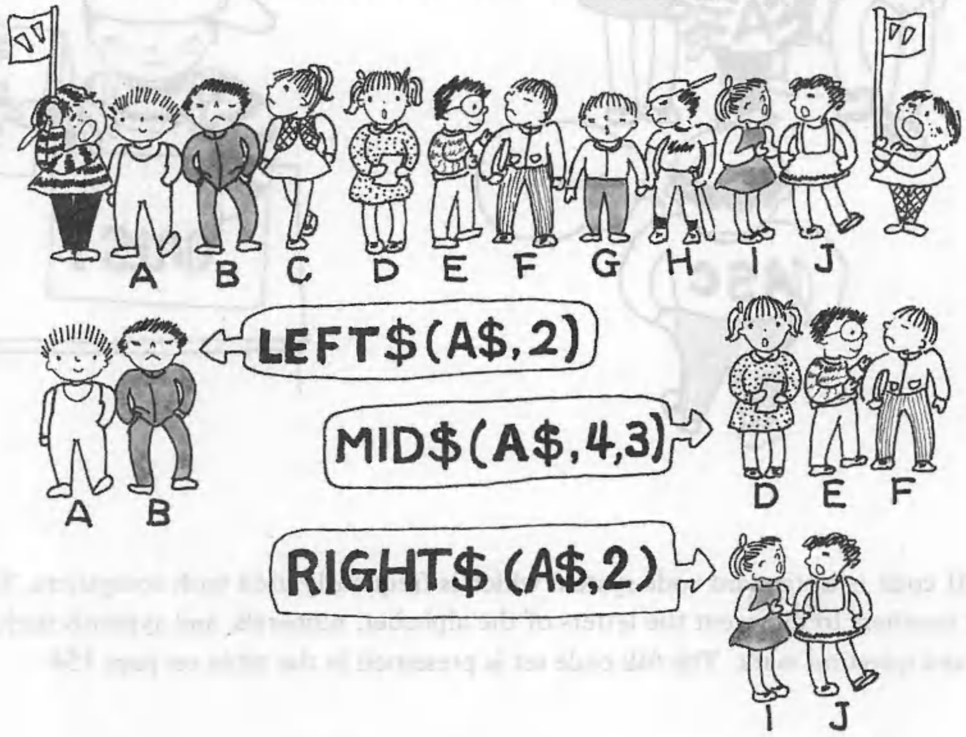
The LEFT$, MID$, and RIGHT$ functions are used to extract character strings from the left end, right end, or middle of a character expression.

Format X$: String expression m and n: Numeric expressions	Function	Example (when A$ = "ABCDEFG")	Remarks

LEFT$ (X$, n)	Returns the character string consisting of the n characters making up the left of string expression X$.	B$= LEFT$ (A$, 2)
		B$← $\boxed{AB}$CDEFG
		Substitutes 2 characters from the left end of string variable A$ into string varible B$.
		Thus, B$ = "AB".
			0≦ n≦ 255

MID$ (X$, m, n)	Returns the character string consisting of the n characters making up the n characters starting with the mth character in string expression X$.	B$=MID$ (A$, 3, 3)
		B$← AB$\boxed{CDE}$FG
		Substitutes the 3 characters starting at the 3rd character in string variable A$ into string variable B$.
			1≦ m≦ 255
			0≦ n ≦ 255

RIGHT$ (X$, n)	Returns the character string consisting of the n characters making up the right end of string expression X$.	B$ = RIGHT$ (A$, 2)
		B$← ABCDE$\boxed{FG}$
		Substitutes 2 characters from the right end of string variable A$ into string varible B$.
		Thus, B$ = "FG".
			0≦ n≦ 255

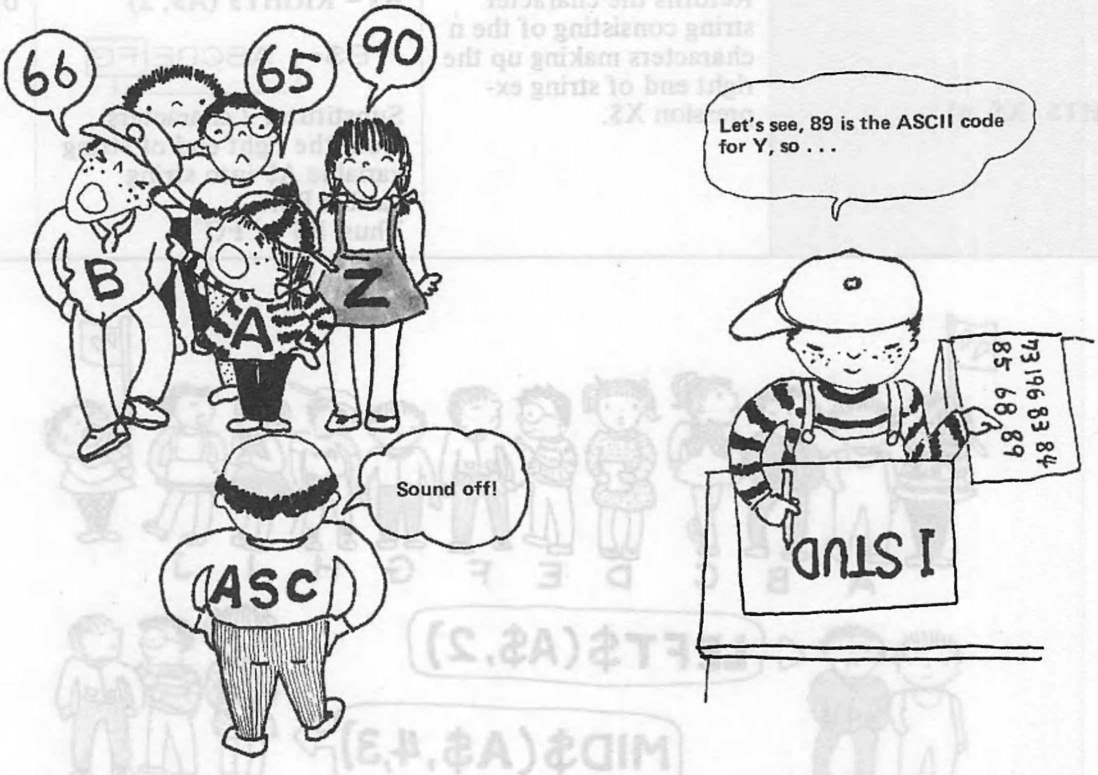


2.5.3 ASC and CHR$

Format	Function	Example

ASC (x$)
x$: 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 (" $\boxed{S}$HARP ")
		Substitutes 83 (the ASCII code for S, the first letter in the string "SHARP") into variable X.

CHR$ (x) 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 then 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) ↵ 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.

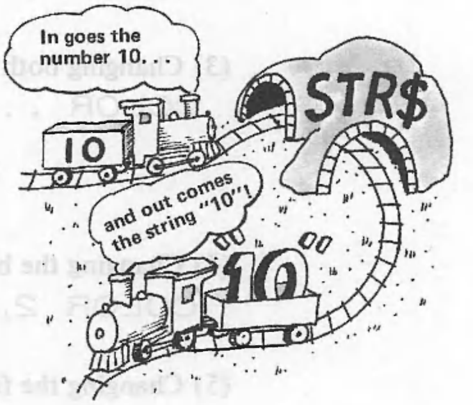
2.5.4 VAL and STR$

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

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).

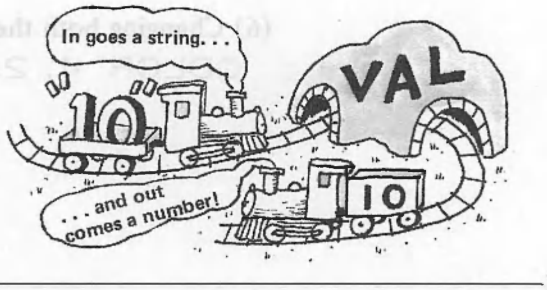
```
1.23456
12.3456
1Ø
1
1234
```

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



```
1Ø FOR X=1 TO 5
2Ø READ A
3Ø L=5-LEN(STR$(INT(A)))
4Ø PRINT TAB(L);A
5Ø NEXT:END
6Ø DATA 1.23456,12.3456
7Ø DATA 123.456,1234.56
8Ø 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, Ø` ..... (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.)

2.6.2 Adding color specifications to the PRINT statement

Format

```
{ PRINT }   [f,  b]   { variable   } < { ; }  { variable   }  { ; } ......>
{   ?   }             { constant   }   { , }  { constant   }  { , }
                      { expression }          { expression }
or
{ PRINT }   [f,  b]   USING  "format string" ; variable < { ; } variable ....>
{   ?   }                                                 { , }
```

f .... Foreground (character color) specification (a number from 0 to 7)

b .... Background color specification (a number from 0 to 7)

Function

Adding the color specifications to the PRINT and PRINT USING statements described on pages 37 and 38 makes it possible to display characters in a variety of colors. In the format above, f indicates the character foreground color, and b indicates the character background color. If only the foreground color is specified, the current background color is used for display of characters; this is done by specifying the foreground color, followed by a comma.

If only the background color is specified, the current foreground color is used for display of characters; in this case, a comma must precede the background color specification.

Example

(Example 1)

`PRINT [6, 5] "ABCDE"` . . . Displays the letters "ABCDE" in yellow against a background of light blue.

`PRINT [, 4] "FGHIJ"` . . . Displays the letters "FGHIJ" in yellow against a background of green.

`PRINT [7, ] "VWXYZ"` . . . Displays the letter "VWXYZ" in green against a background of white.

(Example 2) Let's try adding color to the automobile race program shown on page 46.

```
1Ø PRINT  [, 1] "🅲"
2Ø Q=INT (5*RND (1) ) +2 : X=33*RND (1)
3Ø FOR A=1 TO 5
4Ø READ M$
5Ø PRINT TAB (Ø) ;"◆" ; TAB (X) ;
6Ø PRINT [Q, 1] M$ ;
7Ø PRINT [7, 1] TAB (37) ; "◆"
8Ø NEXT A
9Ø Y=1Ø*RND (1)
1ØØ FOR A=1 TO Y
11Ø PRINT TAB (Ø) ; "◆" ;
12Ø PRINT TAB (37) ; "◆" : NEXT
13Ø RESTORE : GOTO 2Ø
14Ø DATA "  ◢○◣  ", "●▦▦▦●"
15Ø DATA "  ▦▦▦  ", "●▦▦▦●"
16Ø DATA "  ◣■◢  "
```

With ordinary PRINT statements (those without color specifications), the foreground and background colors used for character display are those which have been specified with the latest COLOR statement.

2. 7 Color Plotter-Printer Commands

The color plotter-printer commands described below can be used with the MZ-731 or, when the MZ1P01 color-plotter printer is connected, with the MZ-711, or MZ-721. The color plotter-printer can be used in either of two modes: The text mode (for printout of program lists, results of calculations, or other character data), or the graphic mode (for drawing figures and graphs).

Further, any of four colors (black, blue, green, or red) can be used for printout of characters and graphics. This capability is particularly useful when using the printer in the graphic mode.

2.7.1 General information about the color plotter-printer

(1) The color plotter-printer operates in either of two modes: **The text mode** (for printout of the results of calculations, program lists, and other character data) and **the graphic mode** (used for drawing figures and graphs). The printer will only operate in one mode at a time. (Graphic printer commands are ignored while the printer is in the text mode, and vice versa.)

(2) Printer parameters are reset when the printer is switched from the graphics mode to the text mode. (In other words, the pens' X and Y coordinate settings are reinitialized.)

(3) The printer runs on power supplied from the main unit of the MZ-700, and is not equipped with a separate power switch.

(4) The following switches are used to control operation of the printer.

a. Feed switch .... Advances the paper.

b. Reset switch .... Resets (reinitializes) the printer.

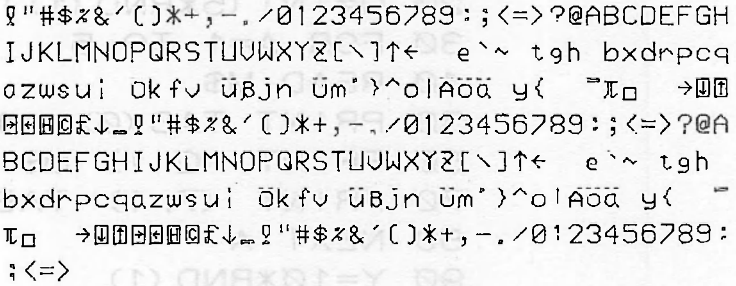
c. Pen change switch ..... Used when replacing the printer's pens.

(5) There are four pen colors: Black, blue, green, and red.

(6) When the printer is used in the text mode, any of three different sizes of characters can be printed. The largest size permits a maximum of 26 characters to be printed on one line, medium size permits a maximum of 40 characters to be printed on one line, and the smallest size allows up to 80 characters to be printed on one line.

Characters which can be printed when using the printer in the text mode are as shown below. No other letters, symbols, or graphic characters can be output while the printer is in this mode.

In most cases, hexadecimal ASCII codes will be printed in a different color if an attempt is made to print graphic characters with the PRINT/-P statement or LIST/P command.



```
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGH
IJKLMNOPQRSTUVWXYZ[\]↑←  e`~ tgh bxdrpcq
ozwsui Ökfv üBjn Üm'}^o|Aöä y{  ¯π□  →⍗⍐
⍈⍇🄷🄲£↓_!"#$%&'()*+,-./0123456789:;<=>?@A
BCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑←  e`~ tgh
bxdrpcqazwsui Ökfv üBjn Üm'}^o|Aöä y{  ¯
π□  →⍗⍐⍈⍇🄷🄲£↓_!"#$%&'()*+,-./0123456789:
;<=>
```

2.7.2 Initial Printer Settings

The initial printer settings made when the BASIC interpreter 1Z-013B is started up are as follows.

(1) Pen color: Black

(2) Pen position: Left side of the carriage. (top line of 1 page.)

(3) Mode: Text mode

(4) Print size: 40 characters/line .... (standard size)
66 lines/page

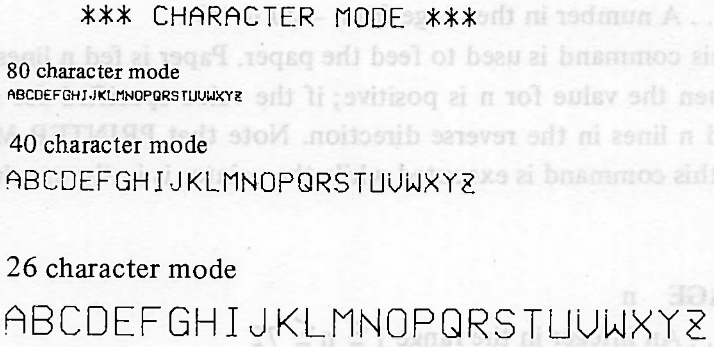
2.7.3 Mode Specification Commands

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** .... (abreviated 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

PCOLOR n
n : 0 black
n : 1 blue
n : 2 green
n : 3 red
.... (abbreviated format: **PC.**)

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 — TEST

Format — 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.)

□	□	□	□	
0	1	2	3	..... Value of n in PCOLOR n
(Black)	(Blue)	(Green)	(Red)	

2.7.5.2 SKIP

Format — SKIP n

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

Function — 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 — PAGE n

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

Function — 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 — LIST/P or LIST/P <LS–Le>

Ls . . . . . . Starting line number

Le . . . . . . Ending line number

Function — 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 — PRINT/P $< I_1,\ d_1,\ I_2,\ d_2$ ...... In, dn >

In ...... Output list (numeric or string expressions)

dn ...... Delimiter

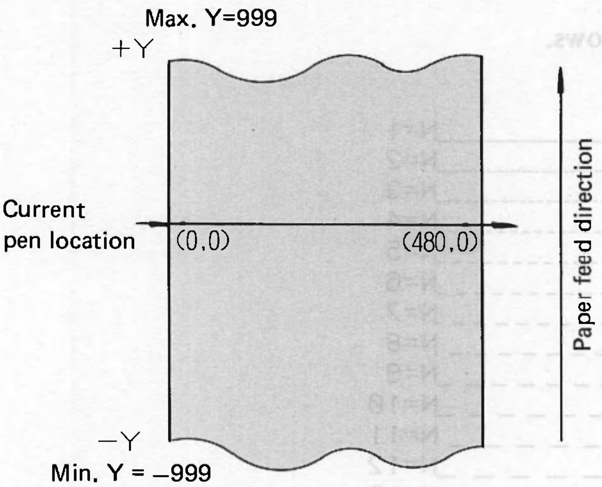
Function — This command outputs the data in the output list to the printer. For details on using this command, see the description of the PRINT command on page 37. See pages 82 for printout of graphic characters.

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

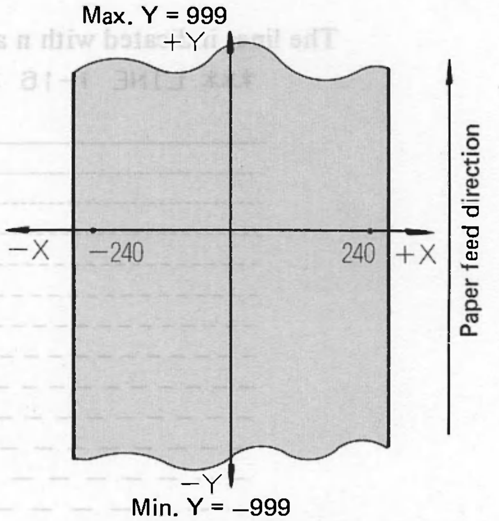
Except that output is directed to the printer, this is the same as the PRINT USING statement described on page 38.

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, . . . , xi, yi> or
LINE %n, x_1, y_1 <, x_2, y_2, . . . , xi, yi>

n . . . . . . Integer from 1 to 16

xi . . . . . Number indicating the X coordinate (xi = −480 to 480; the limit varies depending on the current pen location.)

yi . . . . . Number indicating the Y coordinate (yi = −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.

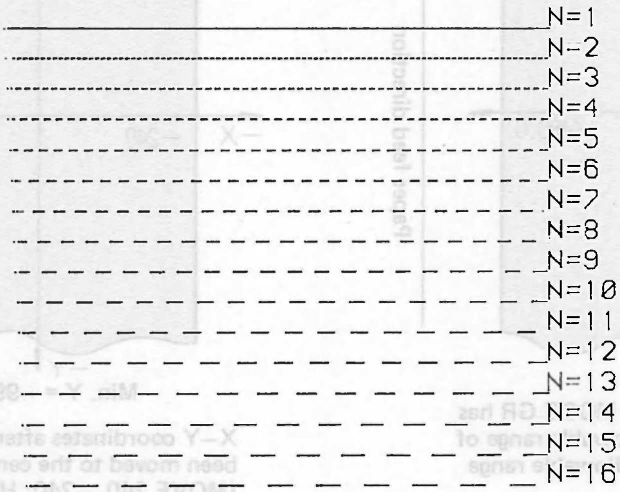
```
1Ø  MODE  GR
2Ø  LINE  24Ø, Ø, 24Ø, −24Ø, Ø, −24Ø, Ø, Ø
3Ø  MODE  TN
```

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

```
1Ø  MODE  GR
2Ø  SQ=INT (12Ø*SQR (3))
3Ø  LINE %2, 24Ø, Ø, 12Ø, −SQ, Ø, Ø
4Ø  MODE  TN
```

The lines indicated with n are as follows.

```
*** LINE 1-16 ***
```



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

Format

RLINE $x_1, y_1 <, x_2, y_2, \ldots xi, yi \ldots>$

RLINE $\%n, x_1, y_1, <, x_2, y_2, \ldots, xi, yi \ldots>$

n .... Integer from 1 to 16

xi ...... Number indicating the X coordinate (−480 to 480)

yi ...... 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.

```
1Ø  MODE  GR
2Ø  SQ=INT (12Ø*SQR (3))
3Ø  RLINE  %1, 24Ø, Ø, −12Ø, −SQ, −12Ø, SQ
4Ø  MODE  TN
```

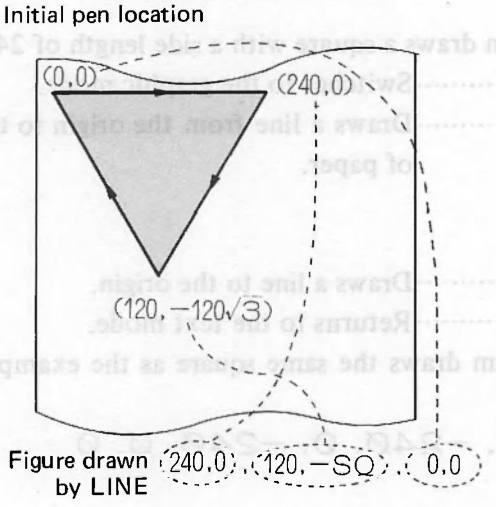


Figure drawn by LINE

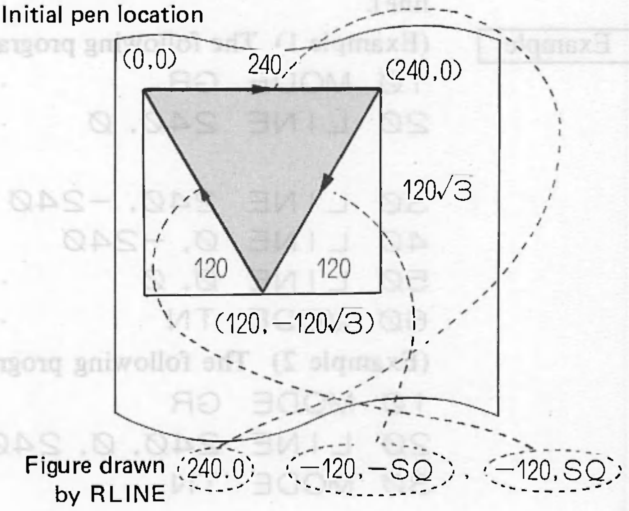


Figure drawn by RLINE

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.

```
10  MODE  GR
20  LINE  480, 0
30  MOVE  240, 240··········Lifts the pen at (480, 0) and moves it to
                            240, 240).
40  LINE  240, -240
50  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** (Δ x, Δ y)

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

```
10 MODE GR
20 LINE 480, 0
30 RMOVE-240, 240·······Lifts the pen at (480, 0), then moves it
                        –240 units in the X direction and 240
                        units in the Y direction.
40 LINE 240, -240
50 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.

```
10  MODE  GR
20  LINE  480, 0  :MOVE  240, 240
30  LINE  240, -240
40  PHOME                     ·········Returns the pen to the origin.
50  PCOLOR  3
60  LINE  0, 240, 480, 240, 480, -240, 0, -240, 0,
    0
70  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

```
1Ø MODE GR
2Ø MOVE 24Ø, -24Ø
3Ø HSET ..............................Sets the new origin.
4Ø FOR I=1 TO 36Ø STEP 3Ø
5Ø L INE 24Ø*COS (PAI(1)*I/18Ø),24Ø*SIN (PAI(1)*I/18Ø)
6Ø PHOME
7Ø NEXT
8ØMODE TN
```

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

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

GPRINTx$

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

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

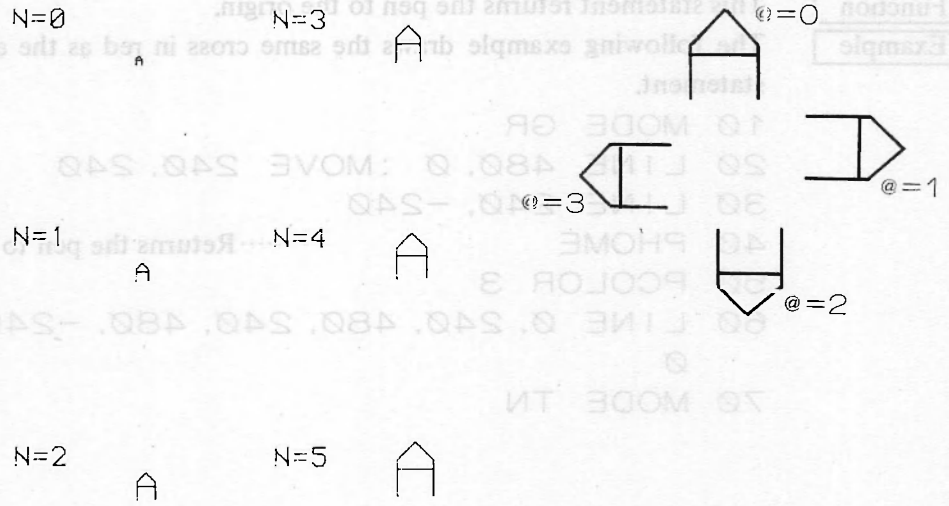
x$. . . . . . . Character

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

```
1Ø MODE GR
2Ø GPRINT "A" ..............Prints "A" in the graphic mode.
3Ø 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)

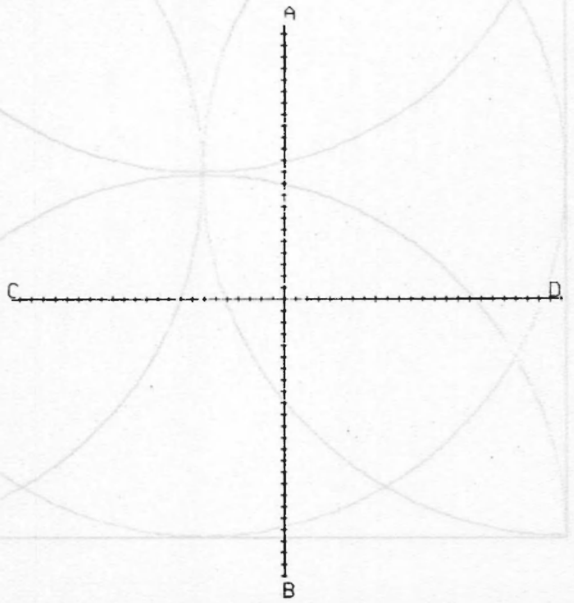
Function

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,5
3Ø GPRINT [1,Ø],"A"
4Ø MOVE 24Ø,Ø............ Lifts the pen and moves it to position
                          A (240, 0).
5Ø AXIS Ø,-1Ø,48......... Draws the Y-axis from position A to
                          position B with scale marks included
                          at 10-unit interval.
6Ø MOVE 24Ø,-5ØØ
7Ø GPRINT [1,Ø],"B"
8Ø MOVE Ø,-24Ø .......... Lifts the pen and moves it to position
                          C (0, –240).
9Ø GPRINT [1,Ø],"C"
1ØØ MOVE Ø,-24Ø
11Ø AXIS 1,1Ø,48......... Draws the X-axis from position C to
                          position D with scale marks included
                          at 10-unit intervals.
12Ø MOVE 47Ø,-24Ø
13Ø GPRINT [1,Ø],"D"
14Ø 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

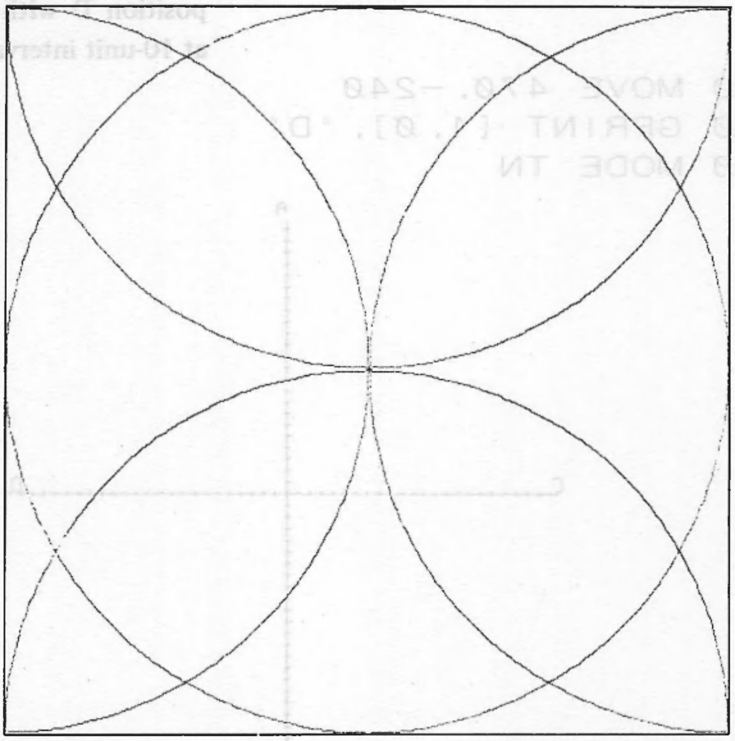
This statement draws a circle or arc with a radius of r and a step of d at location (x, y), starting at angle S and ending at angle e. A complete circle is drawn when s = 0, e = 360 and d = 0.2.

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

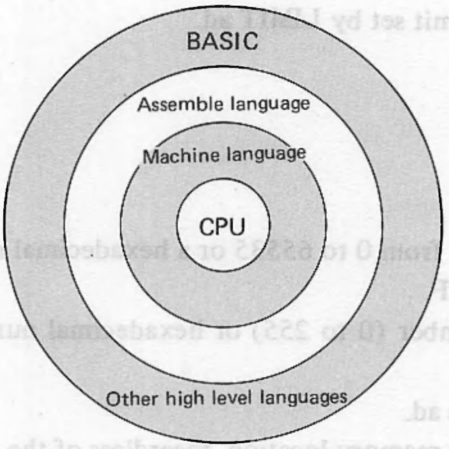
Computer programming languages form a hierarchical structure as shown below. High level languages such as BASIC automatically performs work required when lower level languages such as assembly language are used. Although high level languages are convenient and easy to use, they cannot control the CPU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

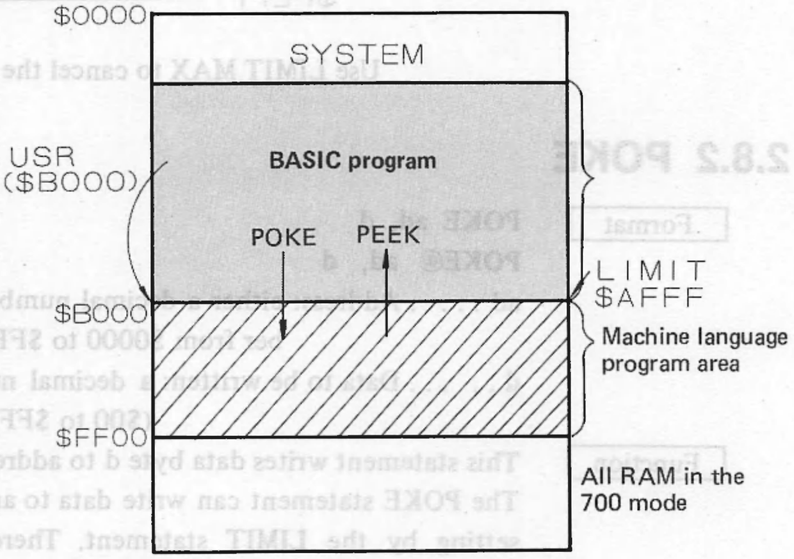
The lowest level language (machine language) directly controls the CPU and provides high processing speed, but considerable skill is required for coding long programs.

Machine language program control statements enable sophisticated programming techiques 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

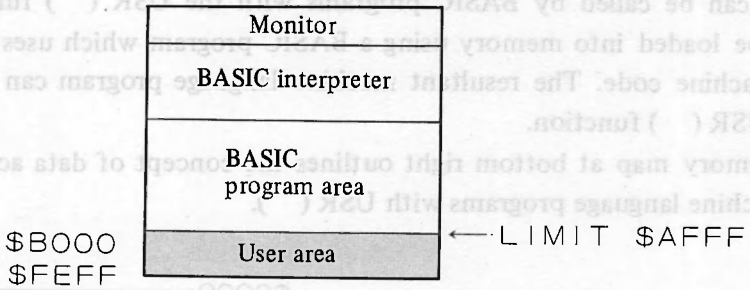
This statement limits the memory area which can be used by the BASIC interpreter. ad indicates the upper limit of the BASIC area, and the area from the following address (ad + 1) to $FEFF (65279) can be used for machine language programs or special data.

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  $DØØØ, $5F
POKE  53248, 95
```

The two statements above perform the same funcition.

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).

```
1Ø FOR AD= 4Ø96Ø TO 4Ø975
2Ø ? PEEK (AD)
3Ø 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                         :.................. Reads data for the machine language program from DATA
80 NEXT I                           :                   statements and writes it into the machine language area.
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.

```
11Ø  LIMIT  $BFFF
11Ø  LOAD  "DISPLAYCODE"
12Ø  USR ($CØØØ)
```

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 definied 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.

```
1Ø ON ERROR GOTO 1ØØØ·······Declares the line number of the
                            error processing routine.
................................................
1ØØØ IF ERN=5 THEN 12ØØ·········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.

ERL is a special function which returns the line number at which an error occurred.

Example

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 then 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 INST , DEL and " ← " keys are disabled by executing this statement. CTRL + G 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.)

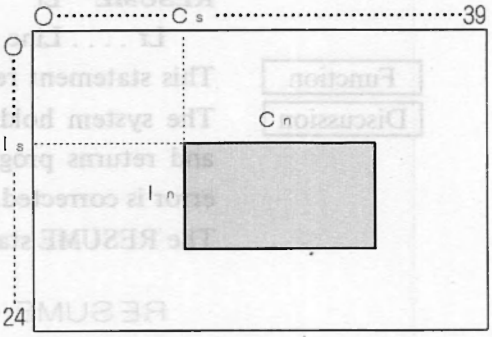
Format — CONSOLE <I_s, I_n< ,C_s, C_n >>

I_s : Starting line of the scroll area

I_n : Number of lines within the scroll area

C_s : Starting column of the scroll area

C_n : Number of columns in the scroll area



Example

```
CONSOLE Ø, 25, Ø, 4Ø
CONSOLE 5, 15
CONSOLE Ø, 25, 5, 3Ø
CONSOLE Ø, 1Ø, Ø, 1Ø
CONSOLE
```

Function — This statement specifies the size of the scroll area; i. e., the area which is cleared by PRINT " C ".

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.

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.

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.

However, this makes it harder to perform screen editing because the values of Cn and In become smaller.

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 adress         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 dispalyed 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 adress] ␣ [end adress] ␣ [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 adress]

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 adress] ␣ [execution adress] : [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.

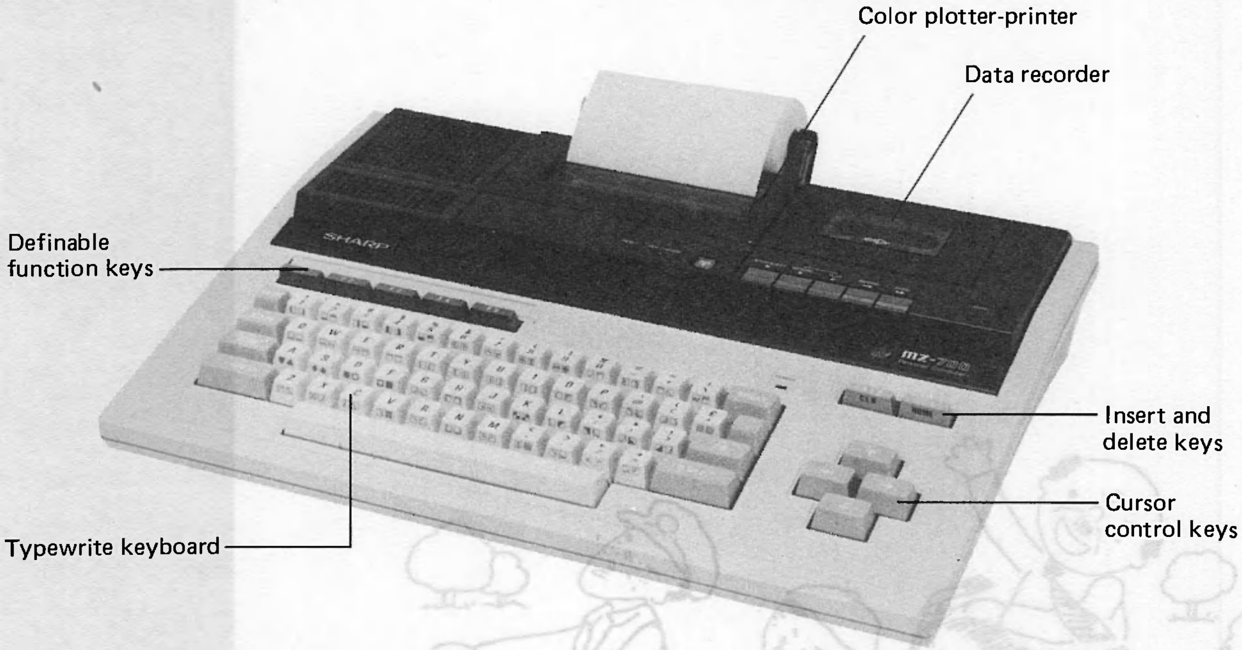
Chapter 3

Operating the MZ-700

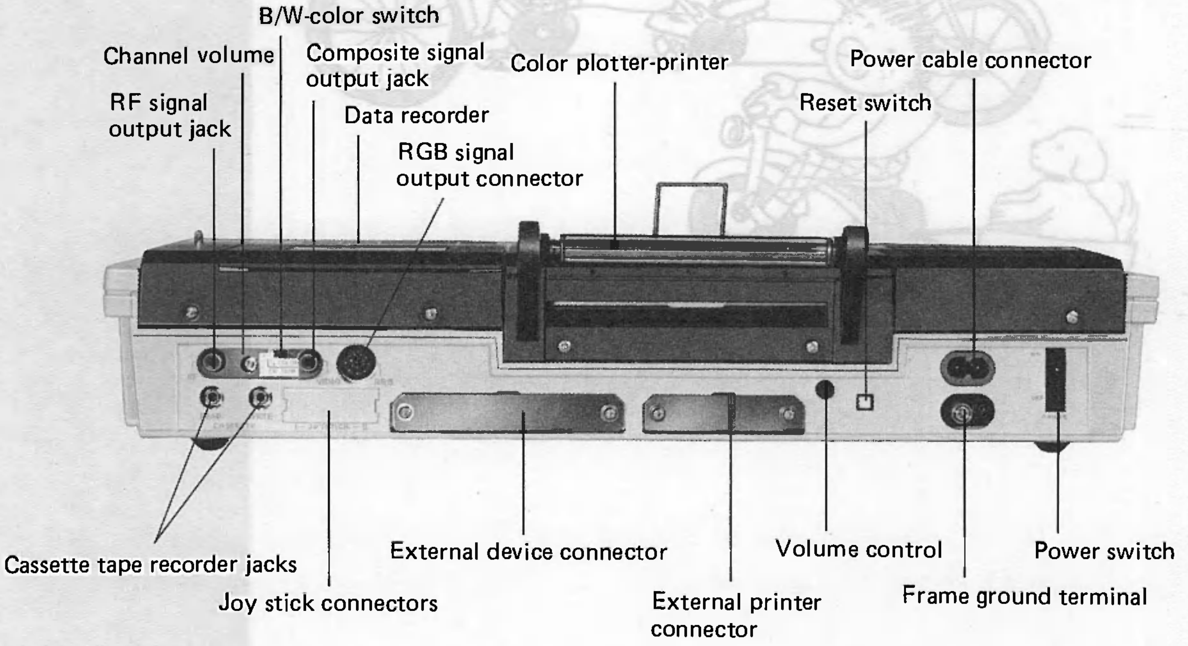
3. 1 Appearance of the MZ-700 Series Personal Computers

3.1.1 MZ-731

- **Front view**

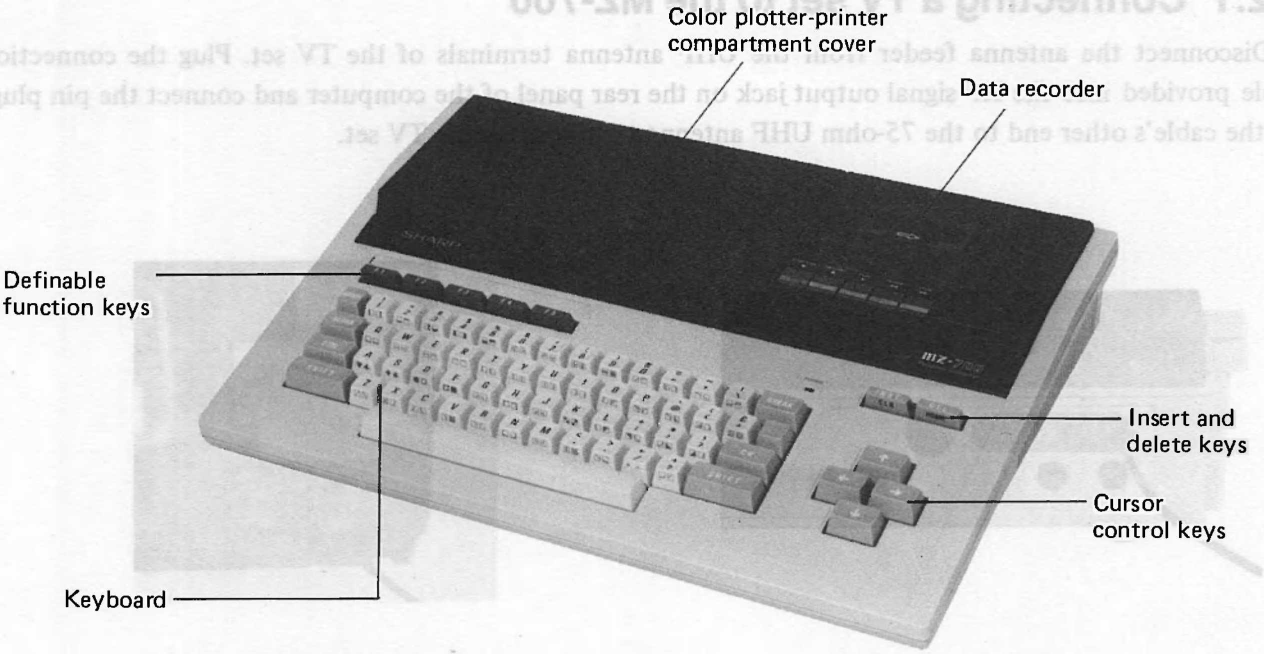


- **Rear view**



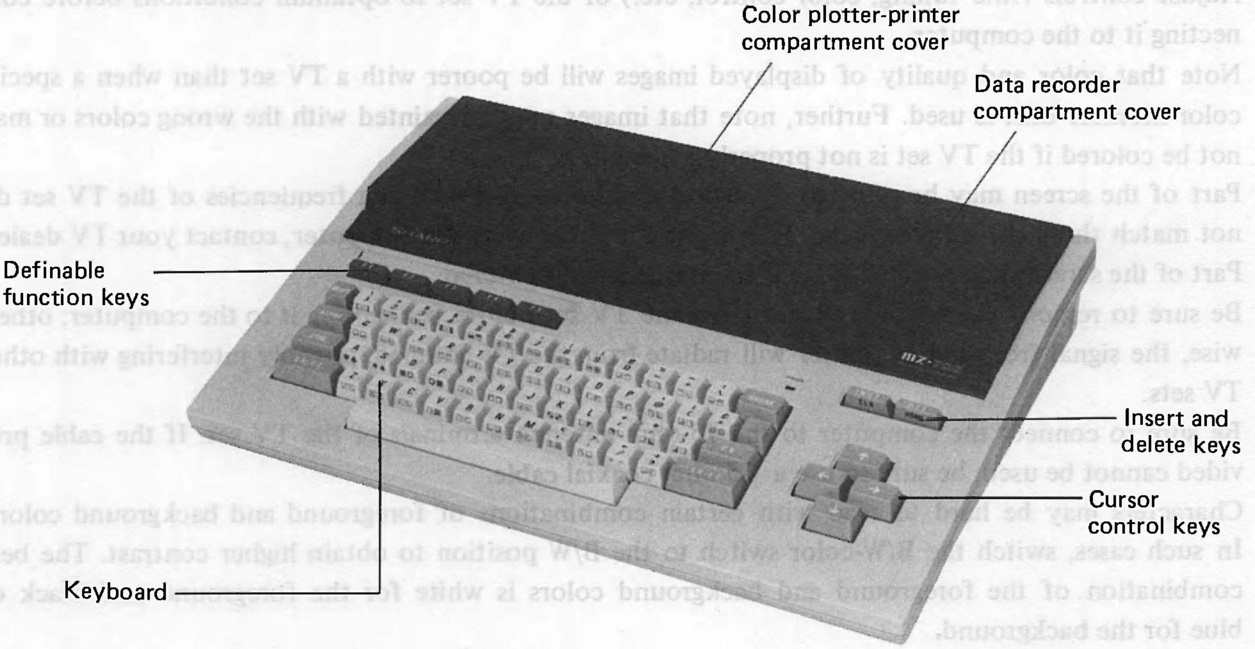
3.1.2 MZ-721

- **Front view**



3.1.3 MZ-711

- **Front view**

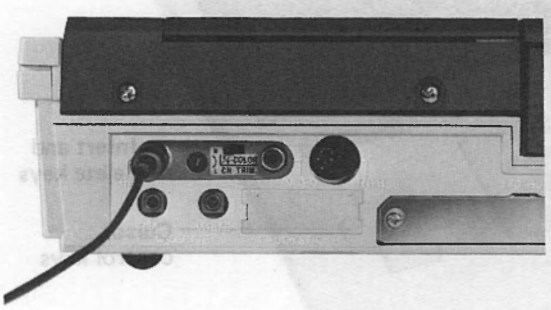


3. 2 Connection to Display Unit

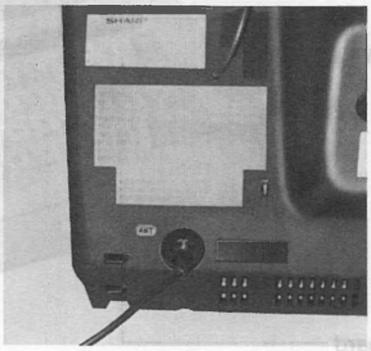
Be sure to turn off both the computer and display unit before connecting them.

3.2.1 Connecting a TV set to the MZ-700

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 ± 3 ch position, depending on which is not used in your area.

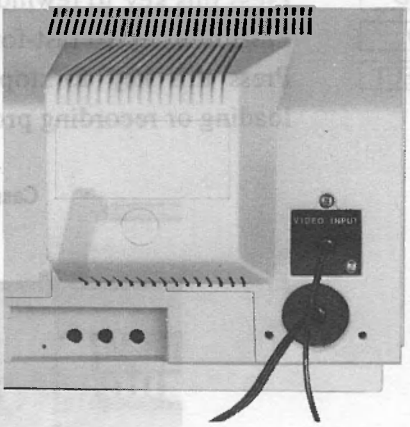
Note the following when using an ordinary TV set as a display unit.

- 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

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.

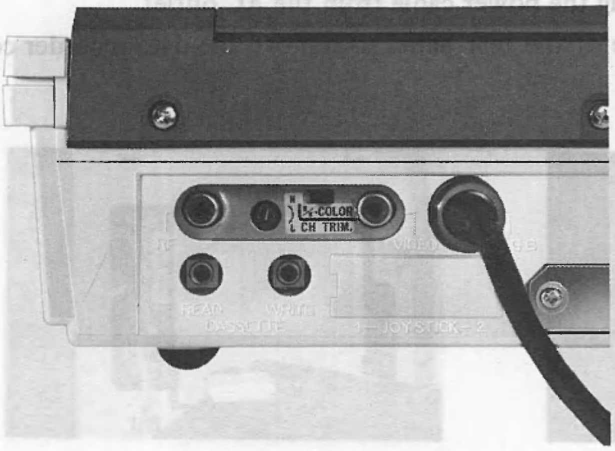
Rear panel of the MZ-700 series computer



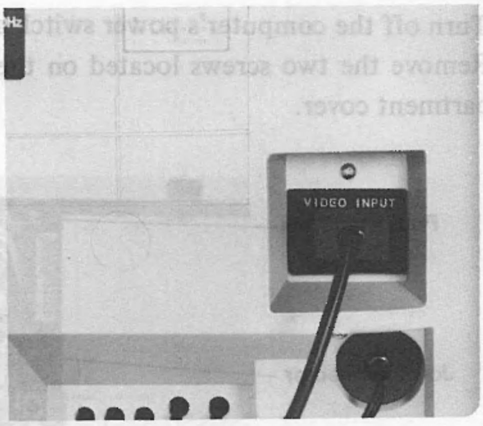
Rear panel of the MZ-1D04

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.

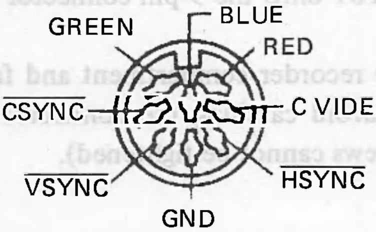


Rear panel of the MZ-700 series computer



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.

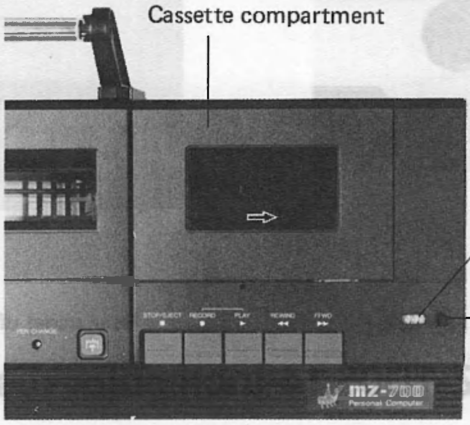
RECORD Press this key to record programs and data.

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.

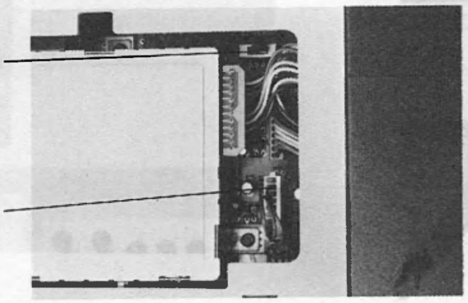
STOP/EJECT Press this key to stop the tape, to release other keys when the tape stops after loading or recording programs or data, or to eject the tape.

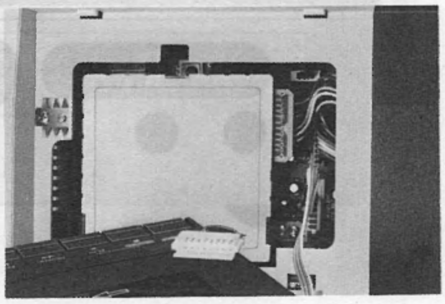


- MZ-1T01

The MZ-1T01 data recorder unit can be installed in the MZ-711 (MZ-710). Installation procedures are as follows.

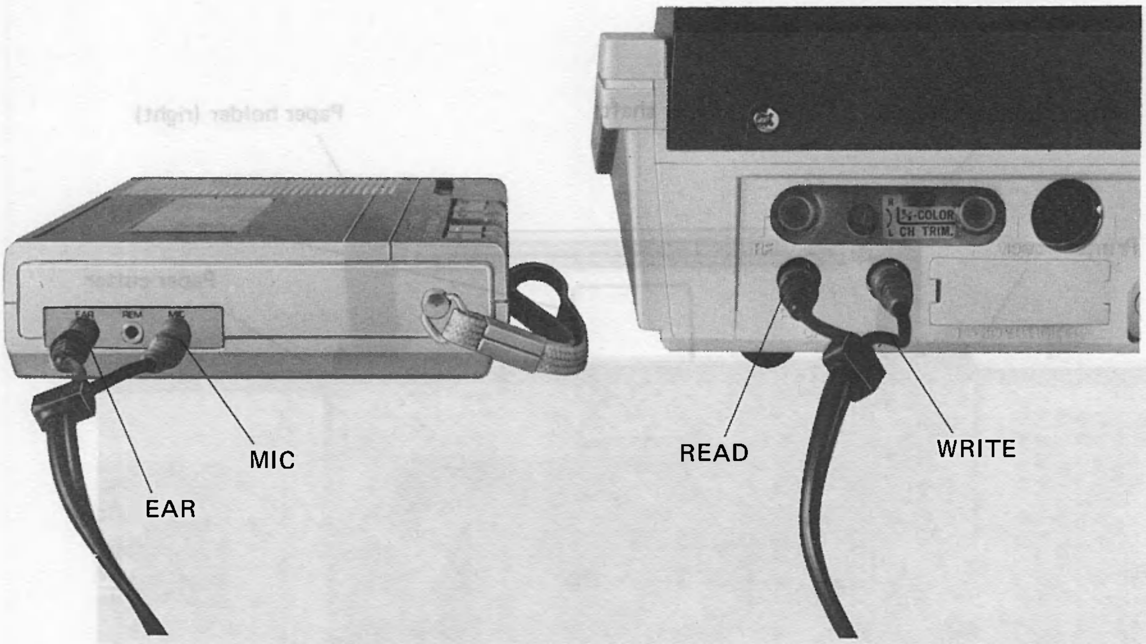
1. Turn off the computer's power switch and unplug the power cable from the AC outlet.
2. Remove the two screws located on the left side of the rear panel to remove the data recorder compartment cover.





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 " ⤓ 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 " ⤓ 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.

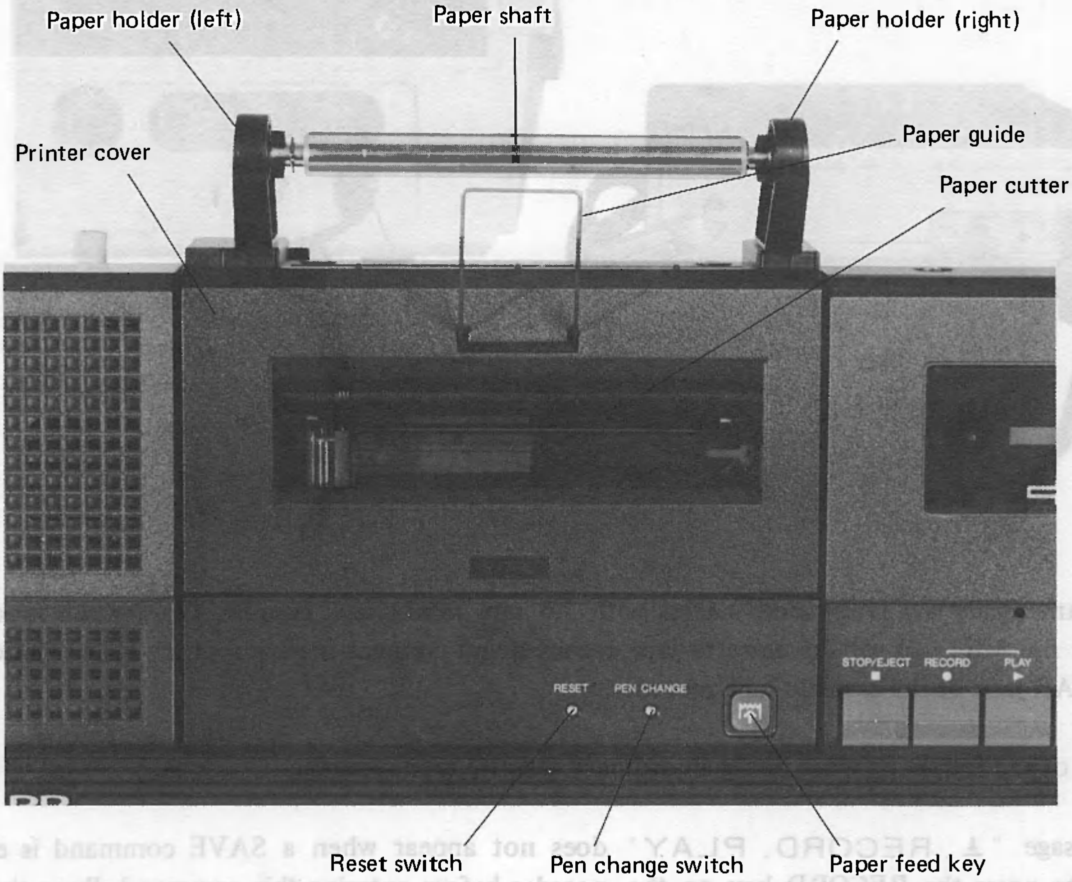
(3) The level and tone controls of the cassette tape reocrder must be adjusted to appropriate levels. Some cassette recorders (e.g. those with the automatic level control) may not be usable. In such cases, please purchase the MZ-1T01.

(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.

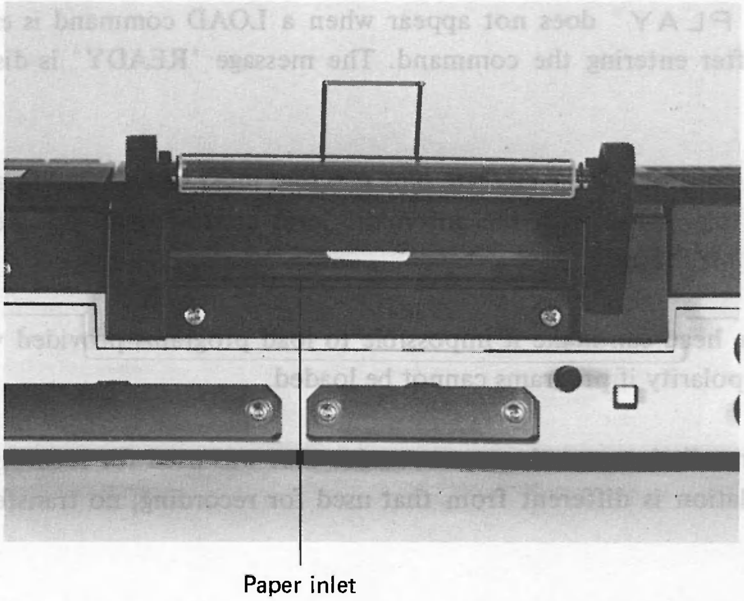
(5) For any transfer or collation, use the tape recorder that was used for recording. If the tape recorder for transfer or collation is different from that used for recording, no transfer nor collation may be possible.

(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**

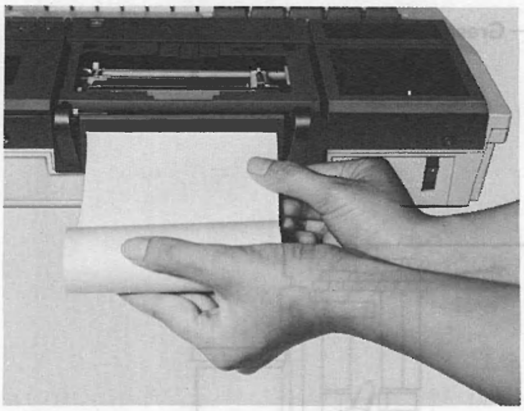
1. Remove the printer cover.
2. Cut the end of roll paper straight across and insert the end into the paper inlet. (Be careful to avoid folding or wrinkling the end of the paper when doing this.)
3. Turn on MZ-731's power switch and press the ⍐ (paper feed) key to feed paper until the top of paper is 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.

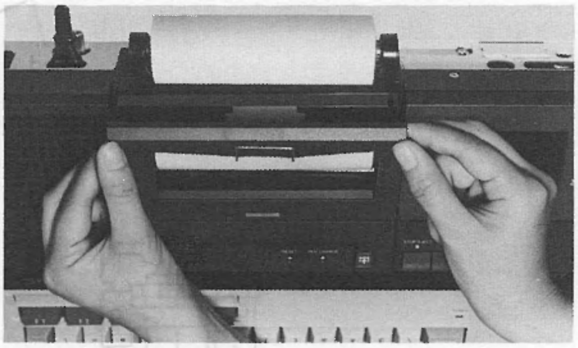
- Roll paper for the MZ-700 series computers is available at any Sharp dealer. Do not use paper other than that specified.

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.

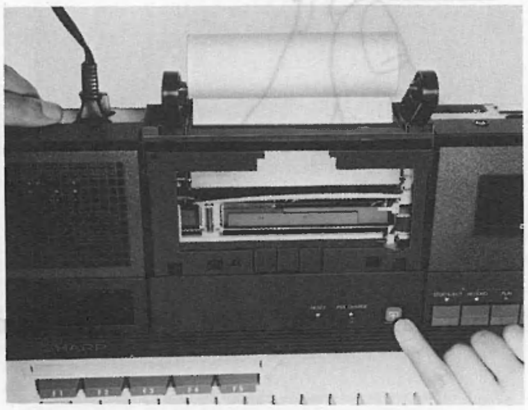
Procedures for loading roll paper



(A) Insert paper into the paper inlet.



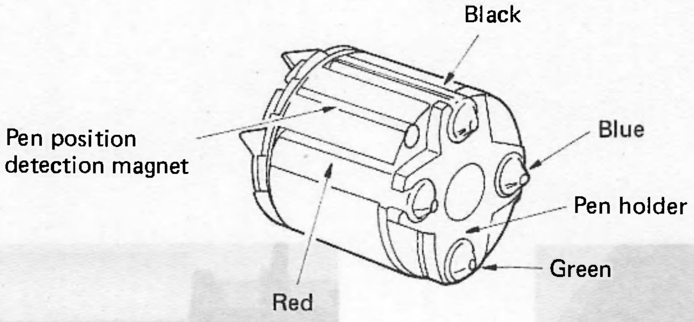
(C) Replace the printer cover.

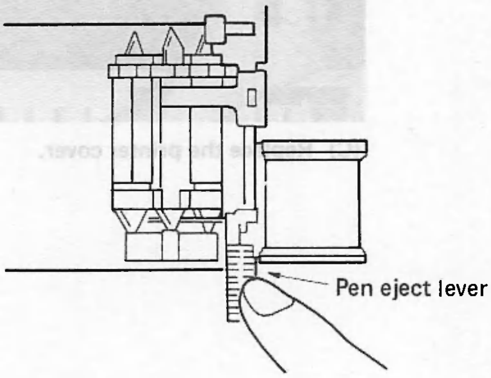


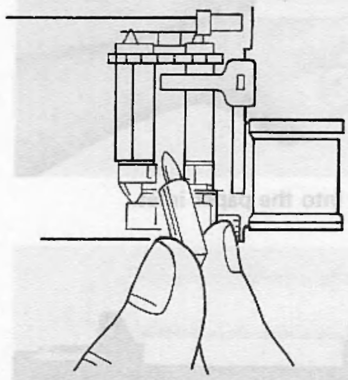
(B) Press the paper feed key to feed paper.

■ 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.







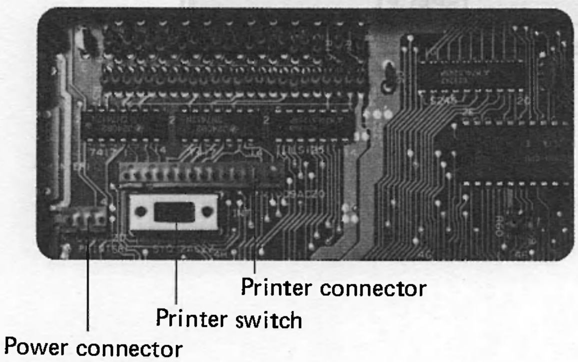
■ Replacements for the printer pens (ballpoint pens) can be purchased at the dealer where the printer was purchased.

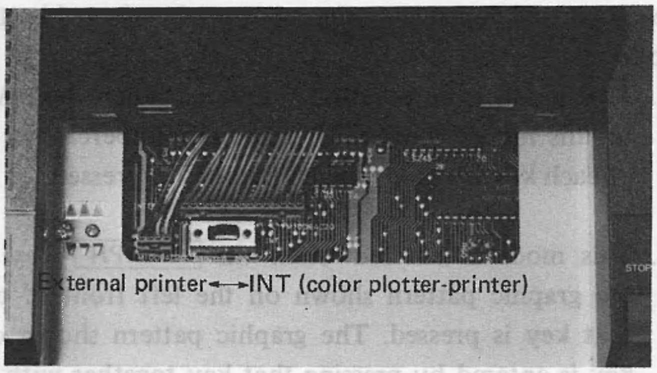
- EA-850B (black; 4 pens)
- EA-850C (black, blue, green, red; 4 pens, 1 of each color)

- **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

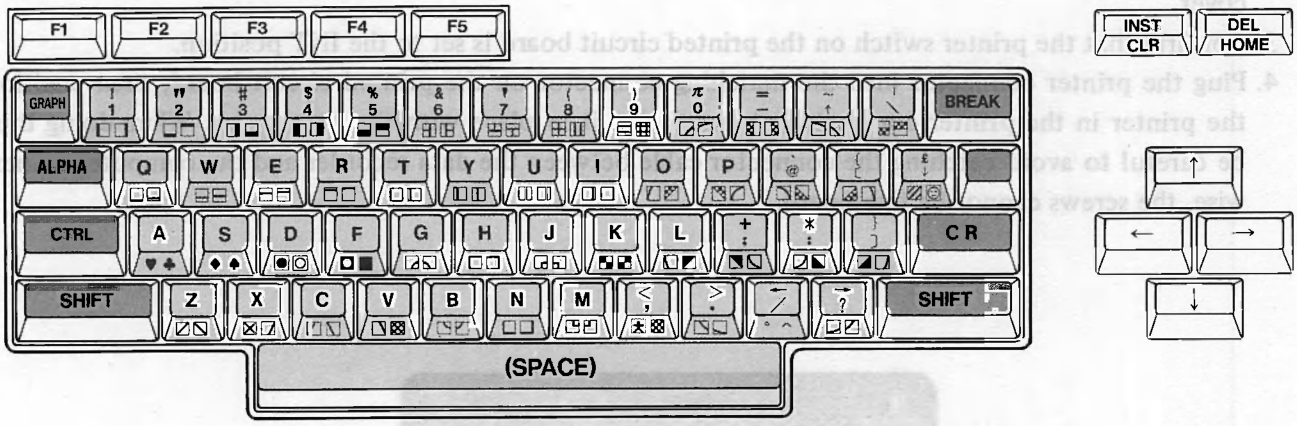
- **Connecting an external printer (MZ-80P5(K))**

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.

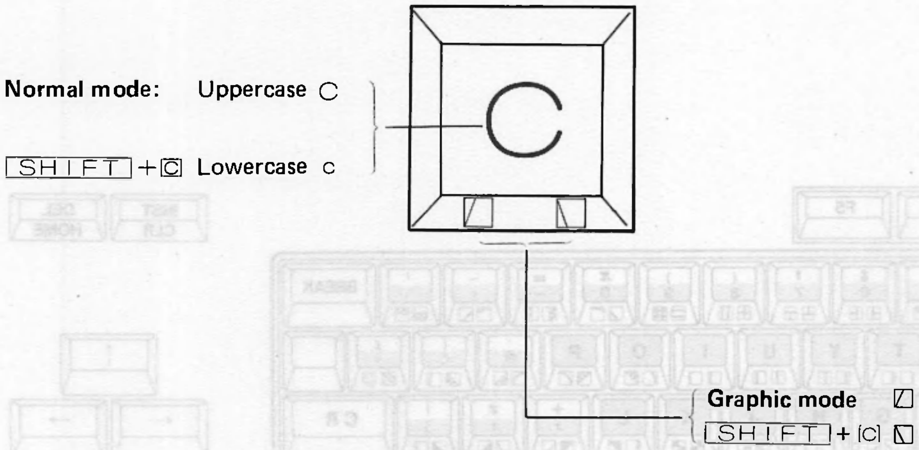
The input modes are as follows.

(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.

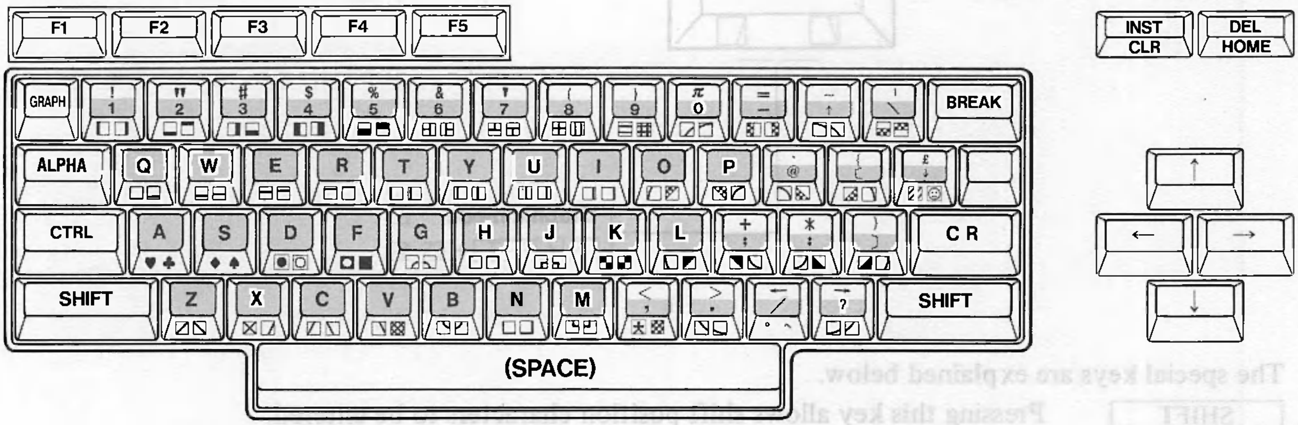
GRAPH Pressing this key changes the input mode from normal to graphic for input of the graphic patterns shown on the left front of keys.

ALPHA **Pressing this key changes the input mode from graphic to normal.**

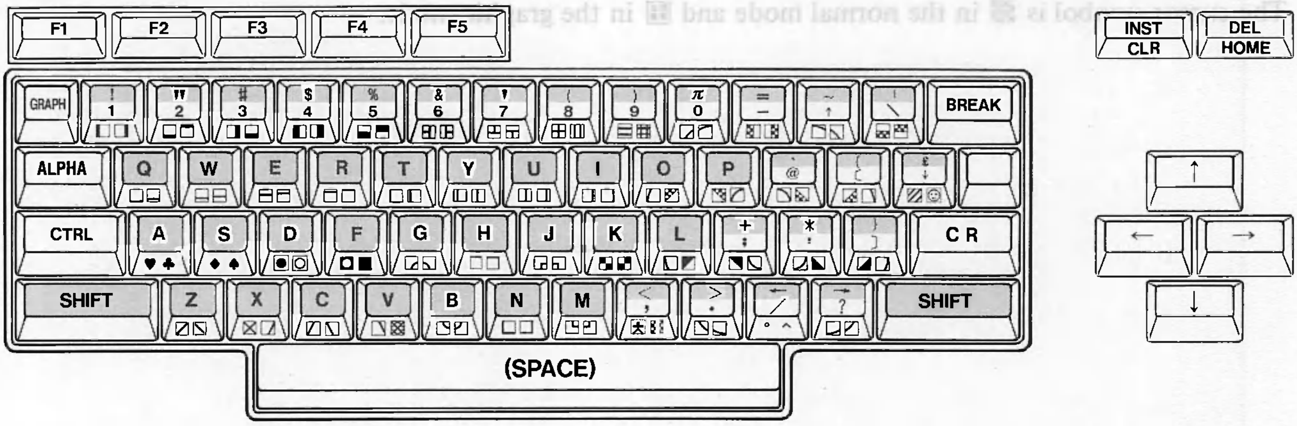
The cursor symbol is ▩ in the normal mode and ▦ in the graphic mode.

(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.

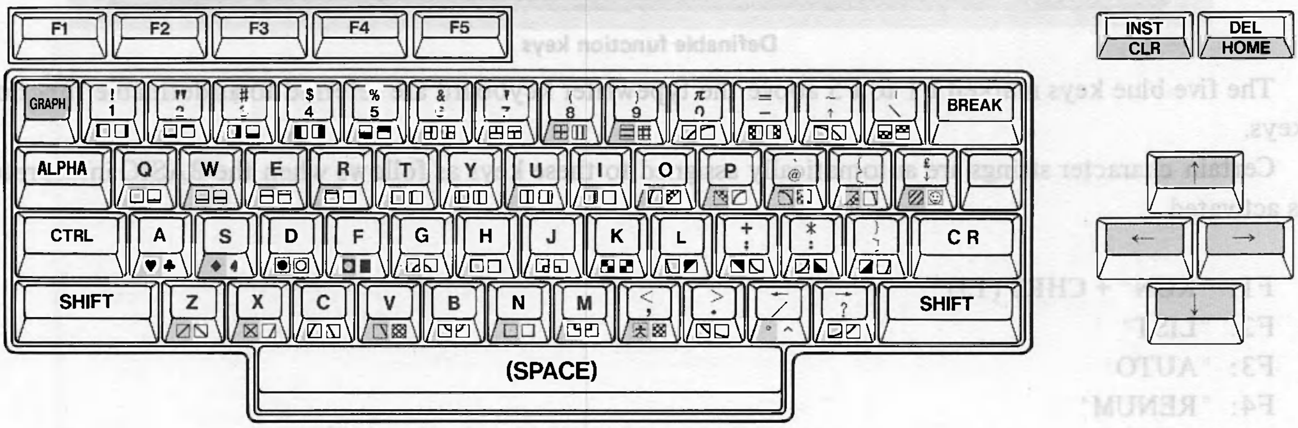


When with the SHIFT key is pressed together with other keys, lowercase letters (or other symbols indicated by the screen areas in the figure below) are entered.

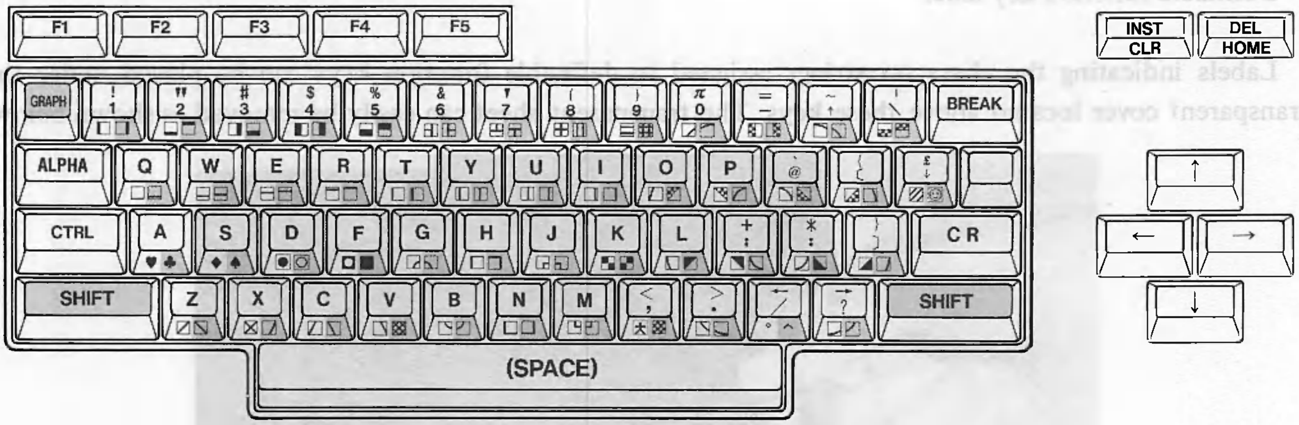


(2) Graphic mode

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.

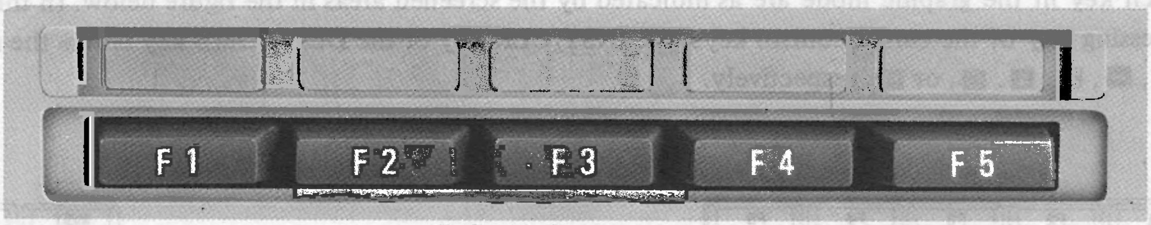


When with the SHIFT key is pressed together with other keys, symbols indicated by the screen areas in the figure below are entered.



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.

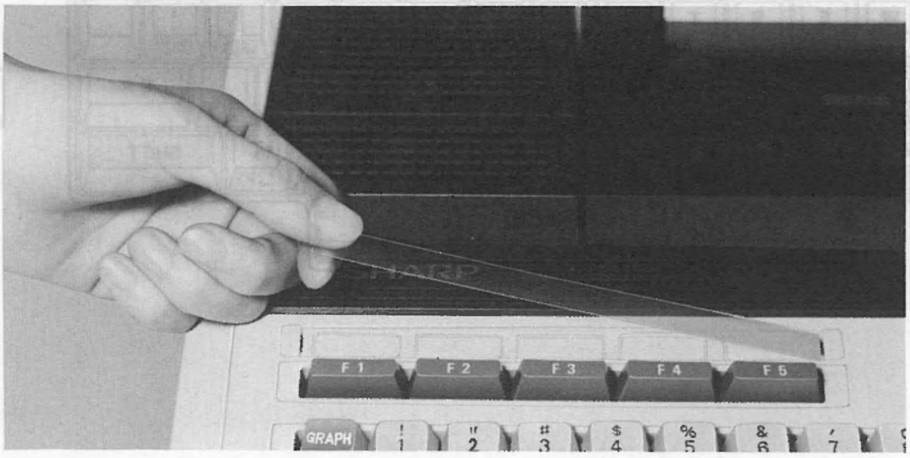
Certain character strings are automatically assigned to these keys as follows when the BASIC interpreter is activated.

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"

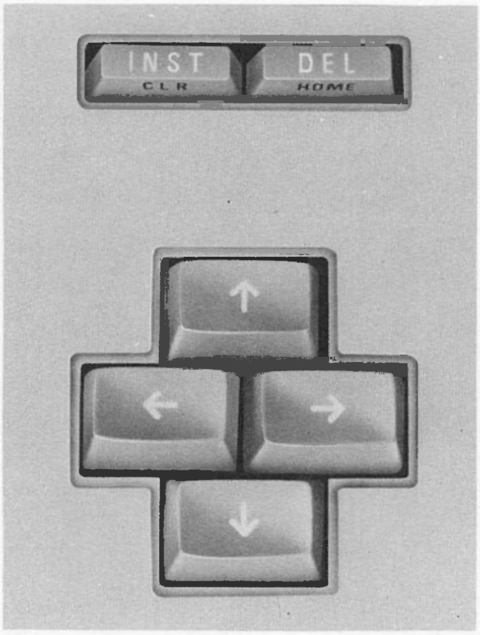
When one of these keys is pressed, the character string assigned to that key is entered; thus, statements which are frequently used can be entered just by pressing one key. The character string assigned to any of the definable function keys can be changed by the DEF KEY statement. (See page 57, DEF KEY 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.

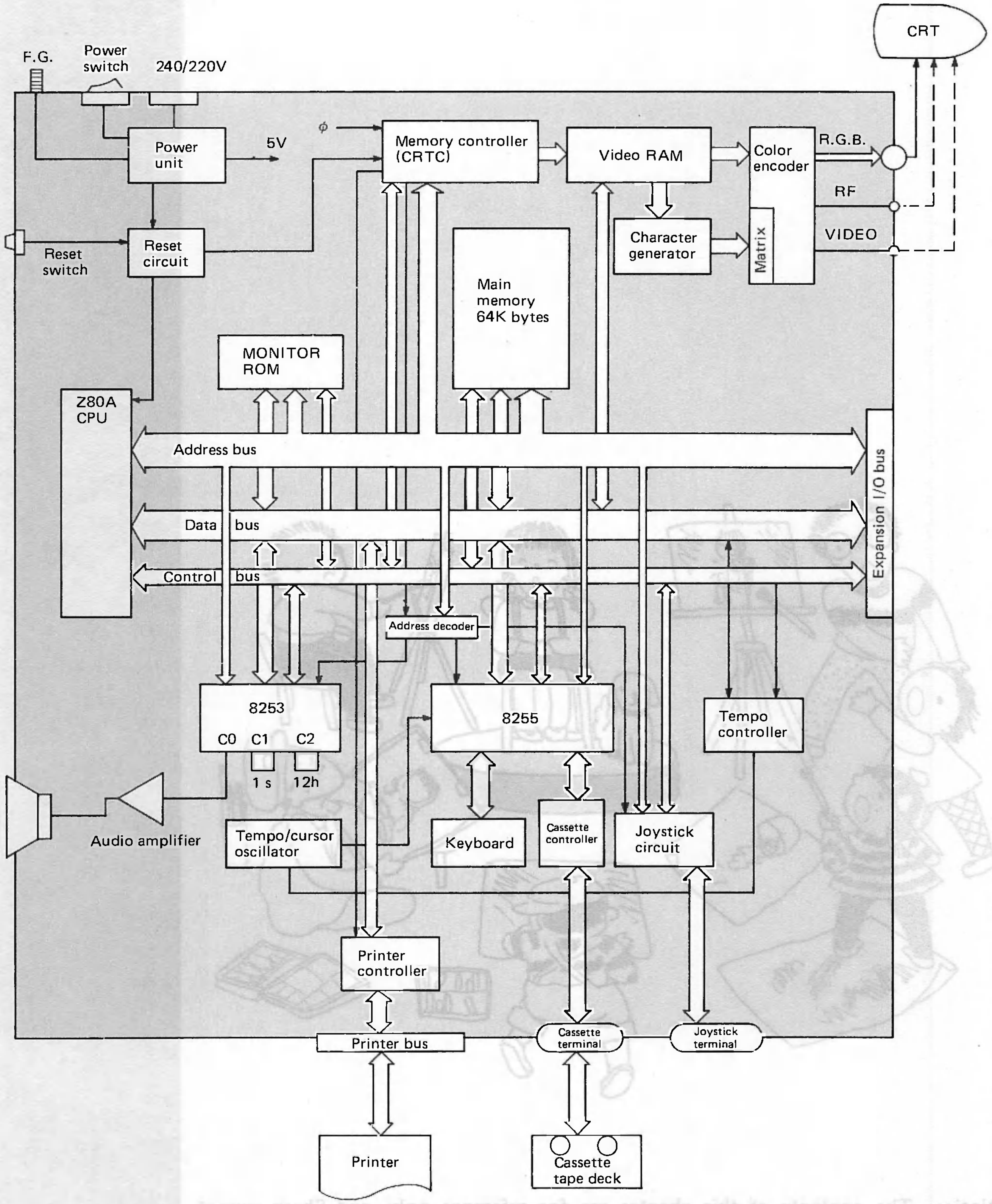
Chapter 4

Hardware

Notice: The contents of this chapter are for reference only, and Sharp cannot assume responsibility for answering any questions about its contents.

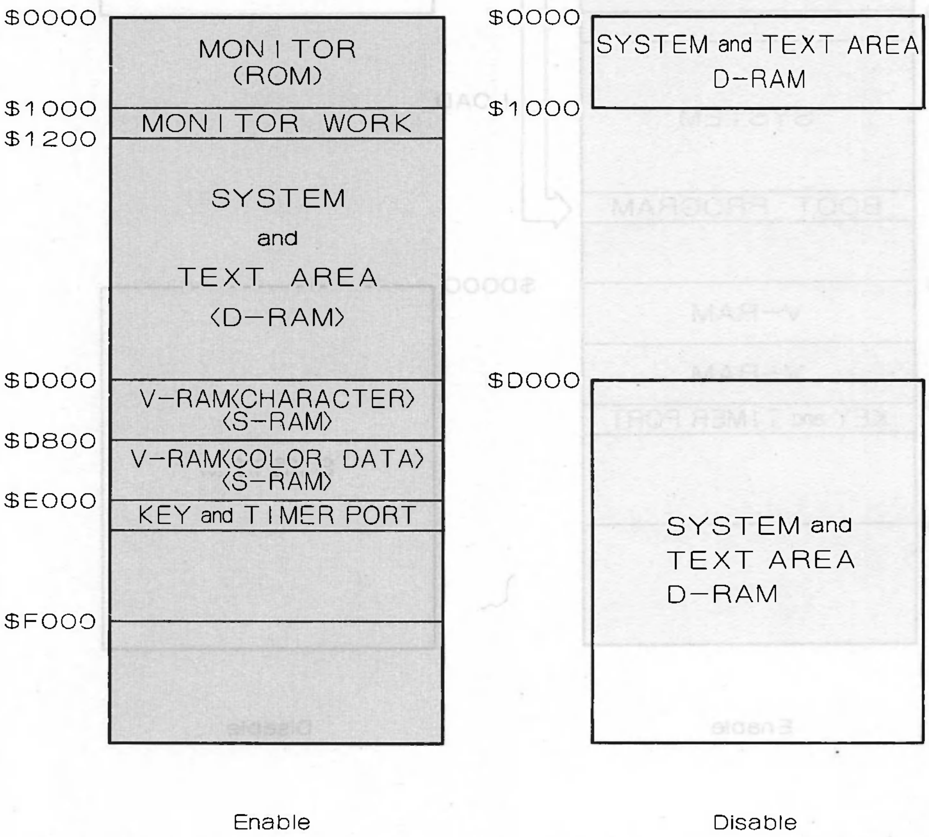
4. 1 MZ-700 System Diagram

The figure below shows the system configuration of the MZ-700 series computers.



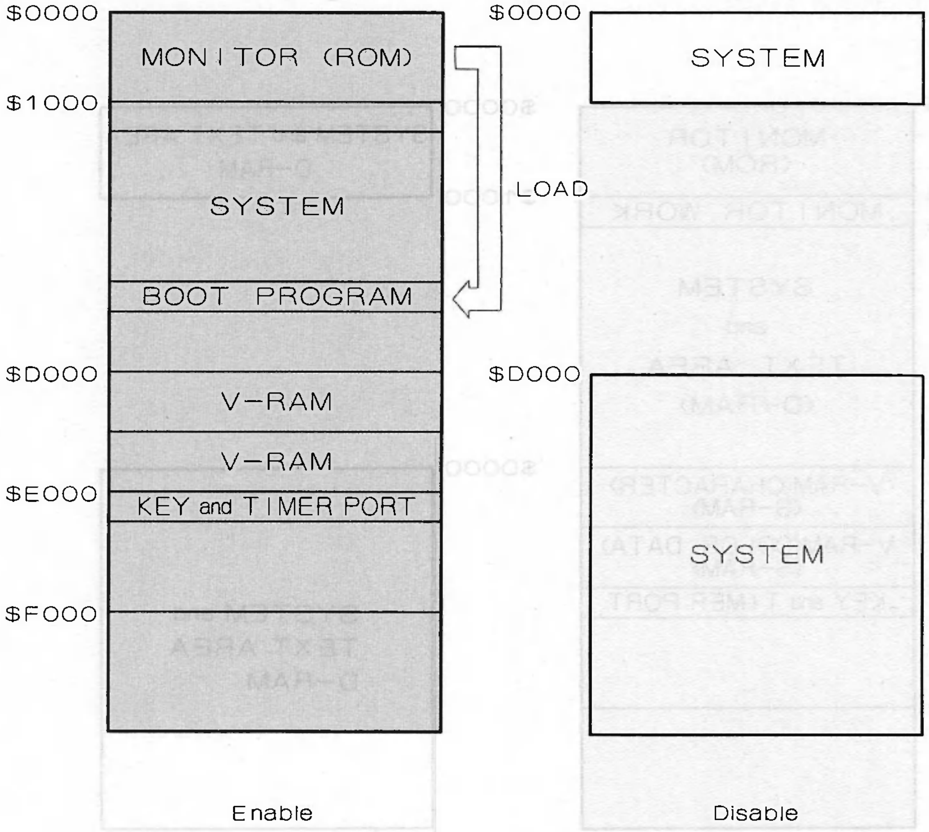
4. 2 Memory configuration

4.2.1 Memory map at power-on (80k mode)



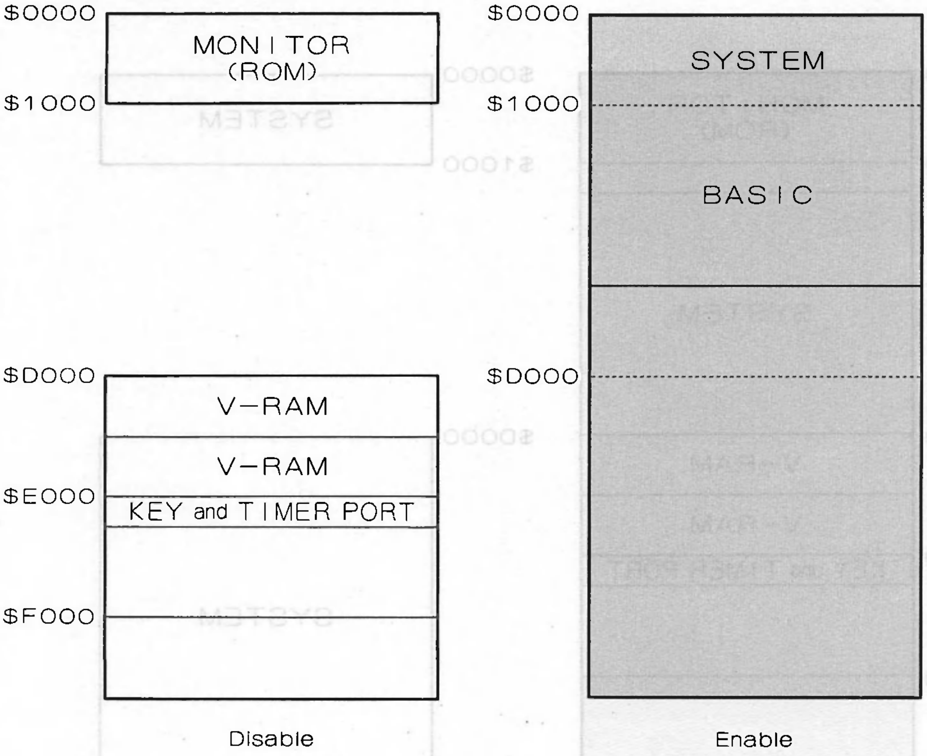
- The memory map is as shown above immediately after the power has been turned on. (The contents of the V-RAM area from $D000 to $DFFF are not the same as those of MZ-80K.)
- The entry point of the monitor ROM is the same as that of the MZ-80K.

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

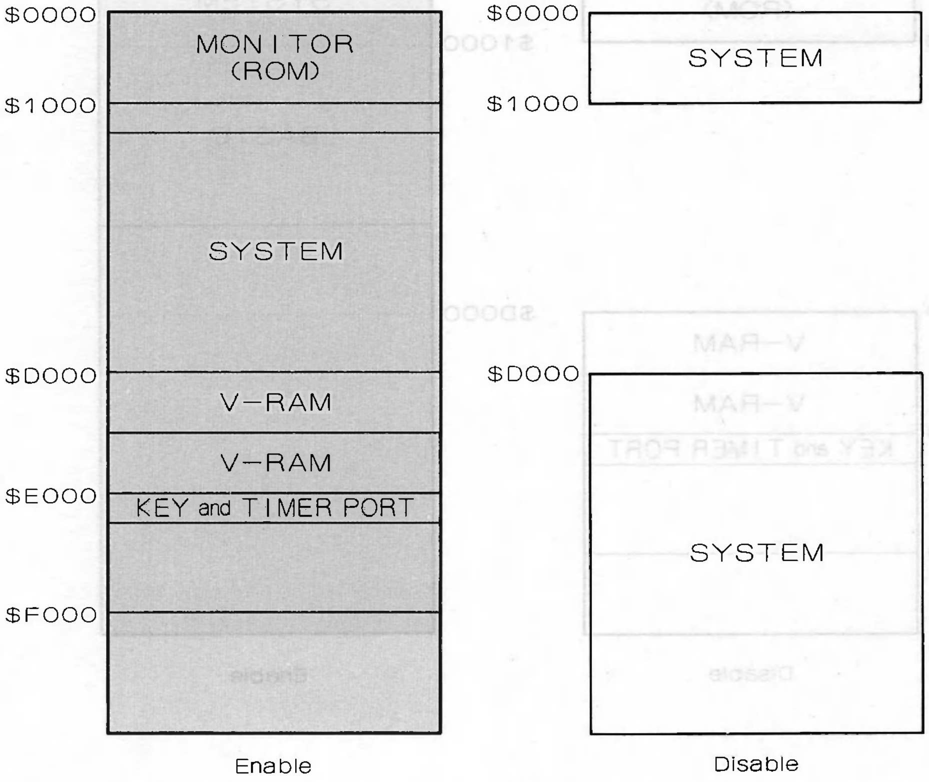
4.2.3 Memory map after the BASIC interpreter has been loaded (MZ-700 mode)



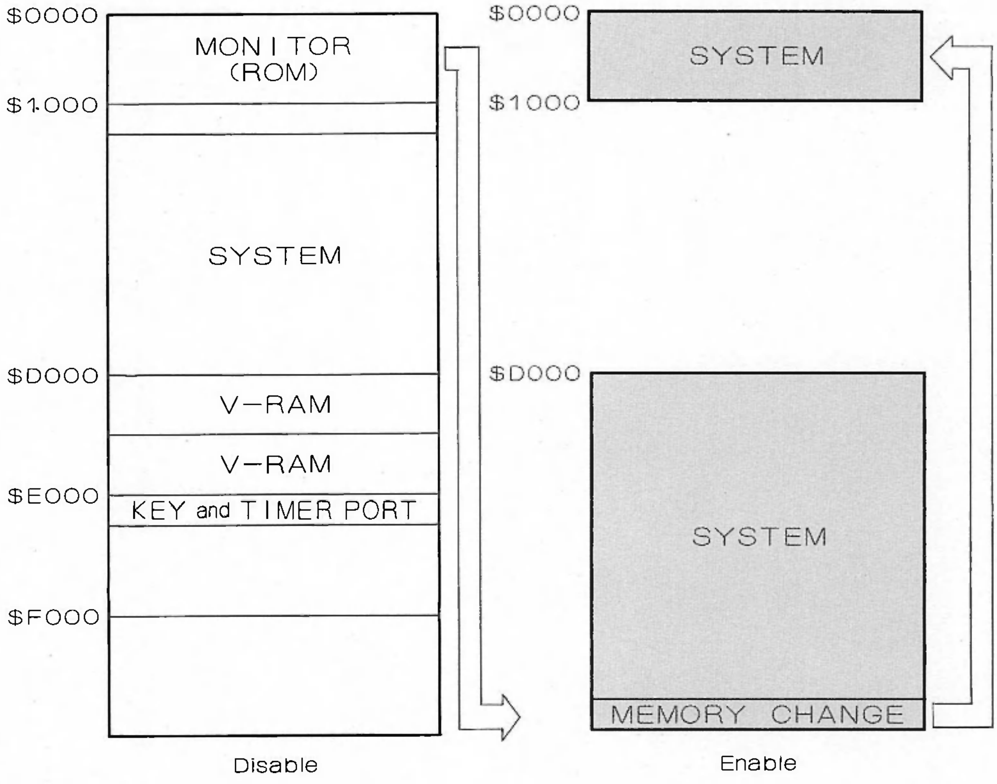
- The memory map is as shown above after the BASIC interpreter has been loaded.
- Bank switching is performed to access V-RAM or the KEY and TIMER PORT area.

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 inhibitted 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.

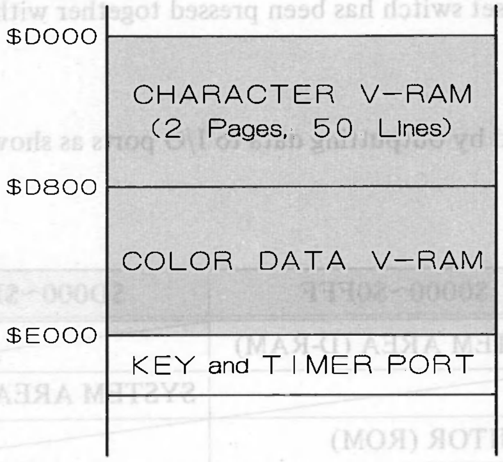
OUT ($E4), A

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):

D000 ← Address
1 ← Byte No.

Column

Line	1	2	3		39	40
1	D000* 1	D001 2	D002 3		D026 39	D027 40
2	D028 41	D029 42	D02A 43		D04E 79	D04F 80
25	D3C0 961	D3C1 962	D3C2 963		D3E6 999	D3E7 1000

b) Area displayed after the screen has been scrolled up one line from the end of V-RAM:

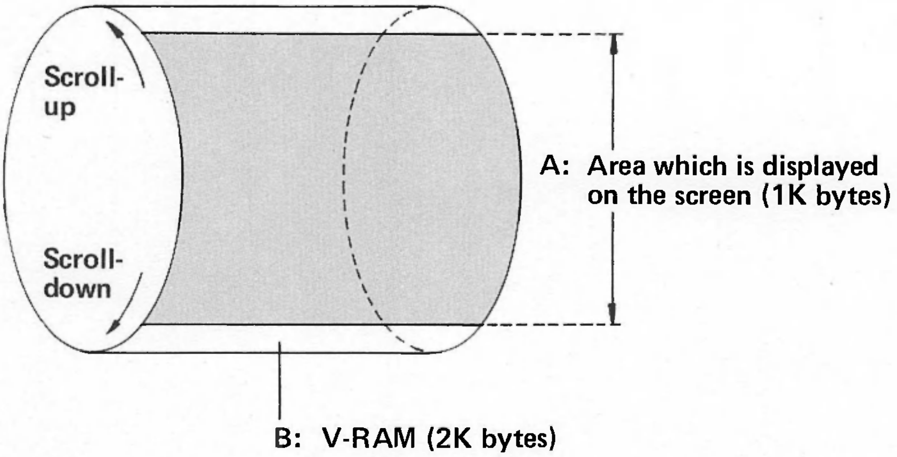
	1	2	3	39	40
1	D000 1041	D001 1042	D002 1043	D026 1079	D027 1080
2	D028 1081	D029 1082	D02A 1083	D04E 1119	D04F 1120
24	D398 1961	D399 1962	D39A 1963	D3BE 1999	D3BF 2000
25	D3C0 1	D3C1 2	D3C2 3	D3E6 39	D3E7 40

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 warpped around to the beginning of V-RAM as shown above.
- The cursor does not move on the screen during scrolling.

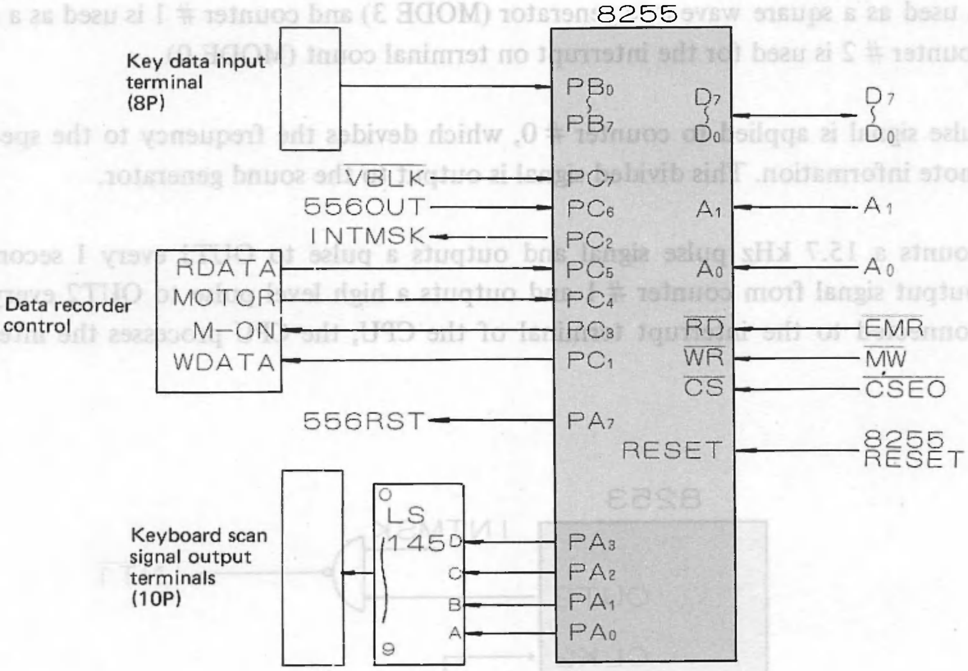
4. 3 Memory Mapped I/O ($E000-$E008)

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_A: Output P_B: Input P_C: Input and output control by bit setting Mode control
$E004 $E005 $E006 $E007	8253	C_0: Mode 3 (square wave rate generator) C_1: Mode 2 (rate generator) C_2: 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 ($E000)	PA_0	OUT	H	Keyboard scan signals	
	PA_1		H		
	PA_2		H		
	PA_3		H		
	PA_7		L	Resets the cursor blink timer.	556 RST
PB ($E001)	PB_0	IN	L	Key scanning data input signals	
	PB_1		L		
	PB_2		L		
	PB_3		L		
	PB_4		L		
	PB_5		L		
	PB_6		L		
	PB_7		L		
PC* ($E002)	PC_1	OUT	–	Cassette tape write data	WDATA
	PC_2	OUT	L	Inhibits clock interrupts.	INTMSK
	PC_3	OUT	⎍	Motor drive signal	M–ON
	PC_4	IN	H	Indicates that the motor is on.	MOTOR
	PC_5	IN	–	Cassette tape read data	RDATA
	PC_6	IN	–	Cursor blink timer input signal	556 OUT
	PC_7	IN	–	Vertical blanking signal	$\overline{\text{VBLK}}$

∗ Each output data bit can be independently set or reset.

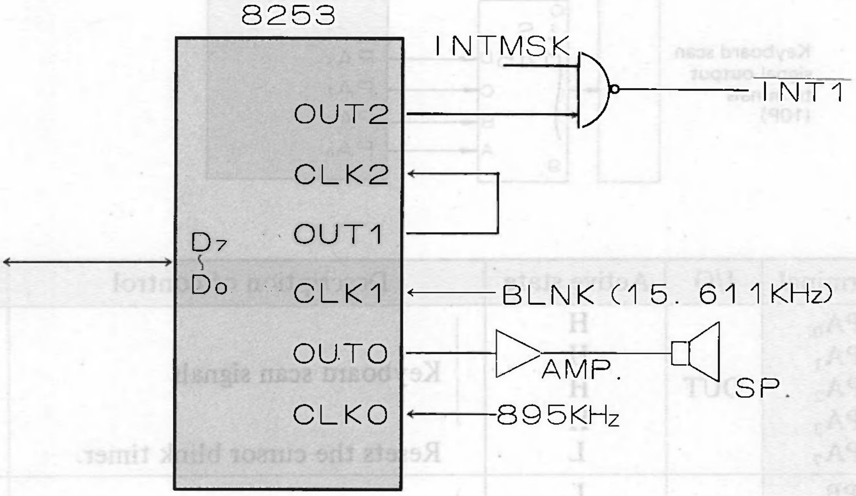
4.3.2 Signal system of the 8253

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 devides 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.



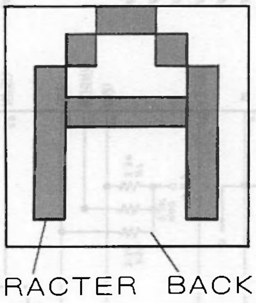
4. 4 Signal System of Color V-RAM

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.

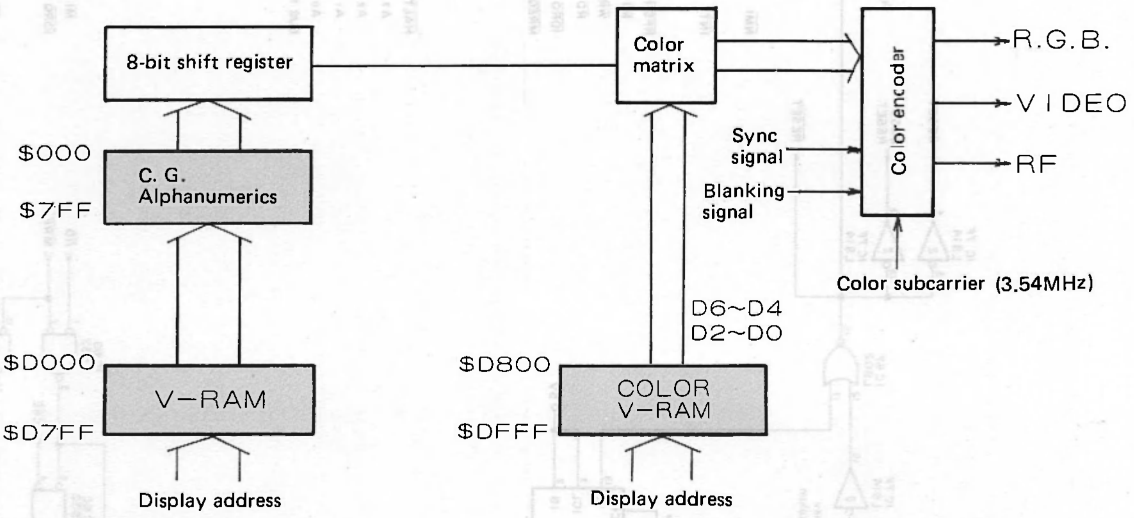
A color information table is shown in the figure below.

D7	Not used.	
D6	CHARACTER	G
D5		R
D4		B
D3	Not used.	
D2	BACK	G
D1		R
D0		B

G : Green
R : Red
B : Blue



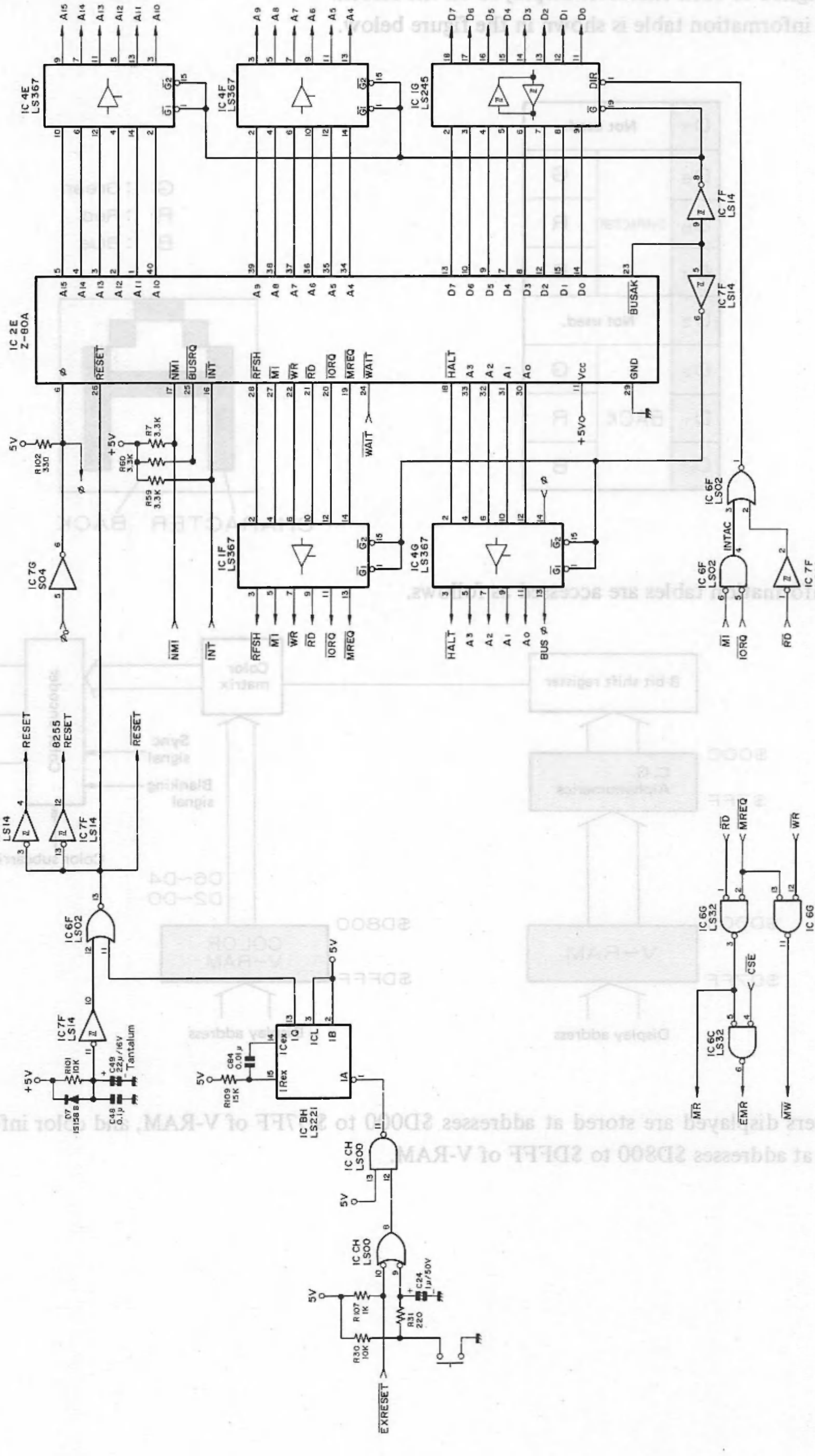
Color information tables are accessed as follows.



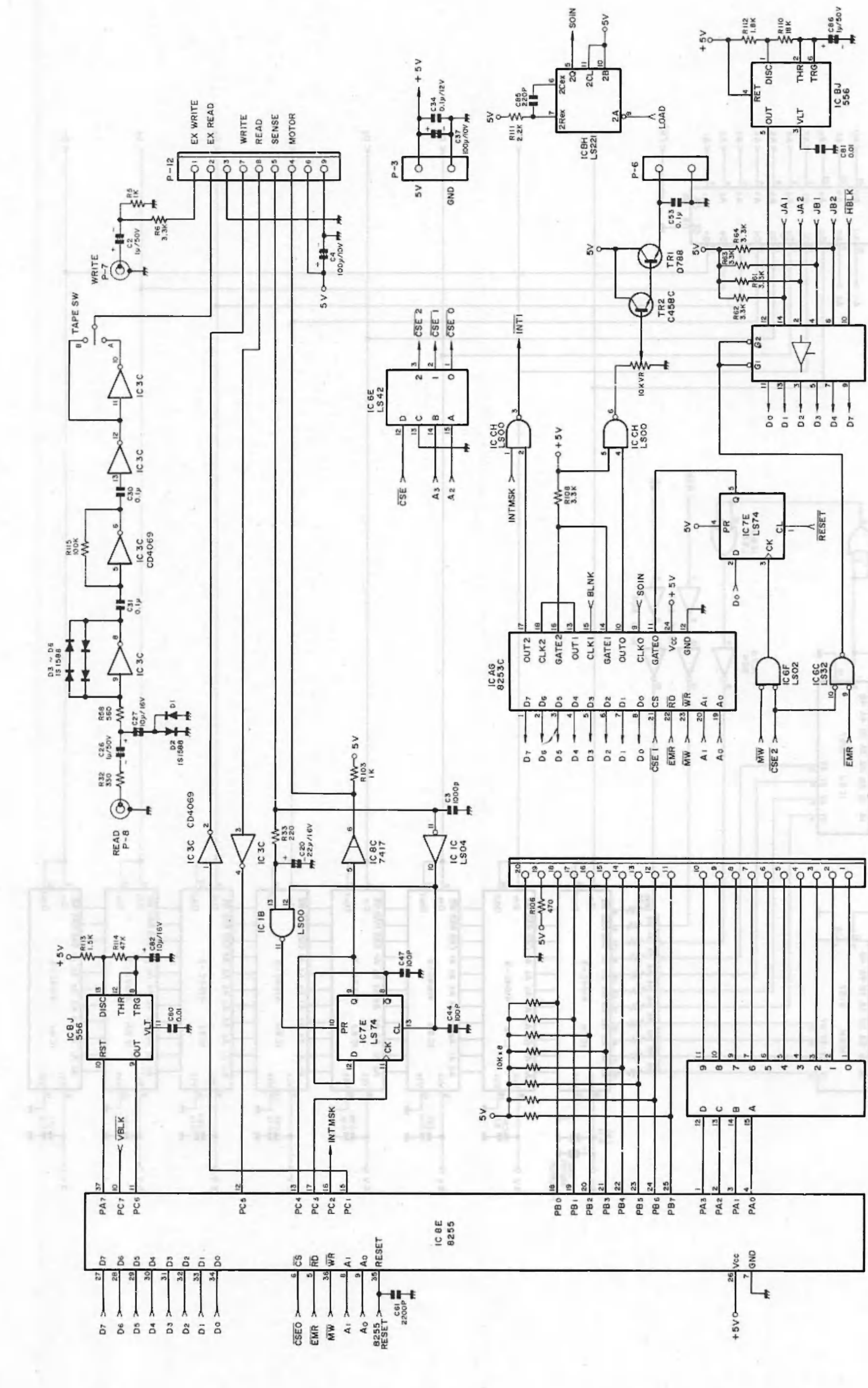
Characters displayed are stored at addresses $D000 to $D7FF of V-RAM, and color information tables are stored at addresses $D800 to $DFFF of V-RAM.

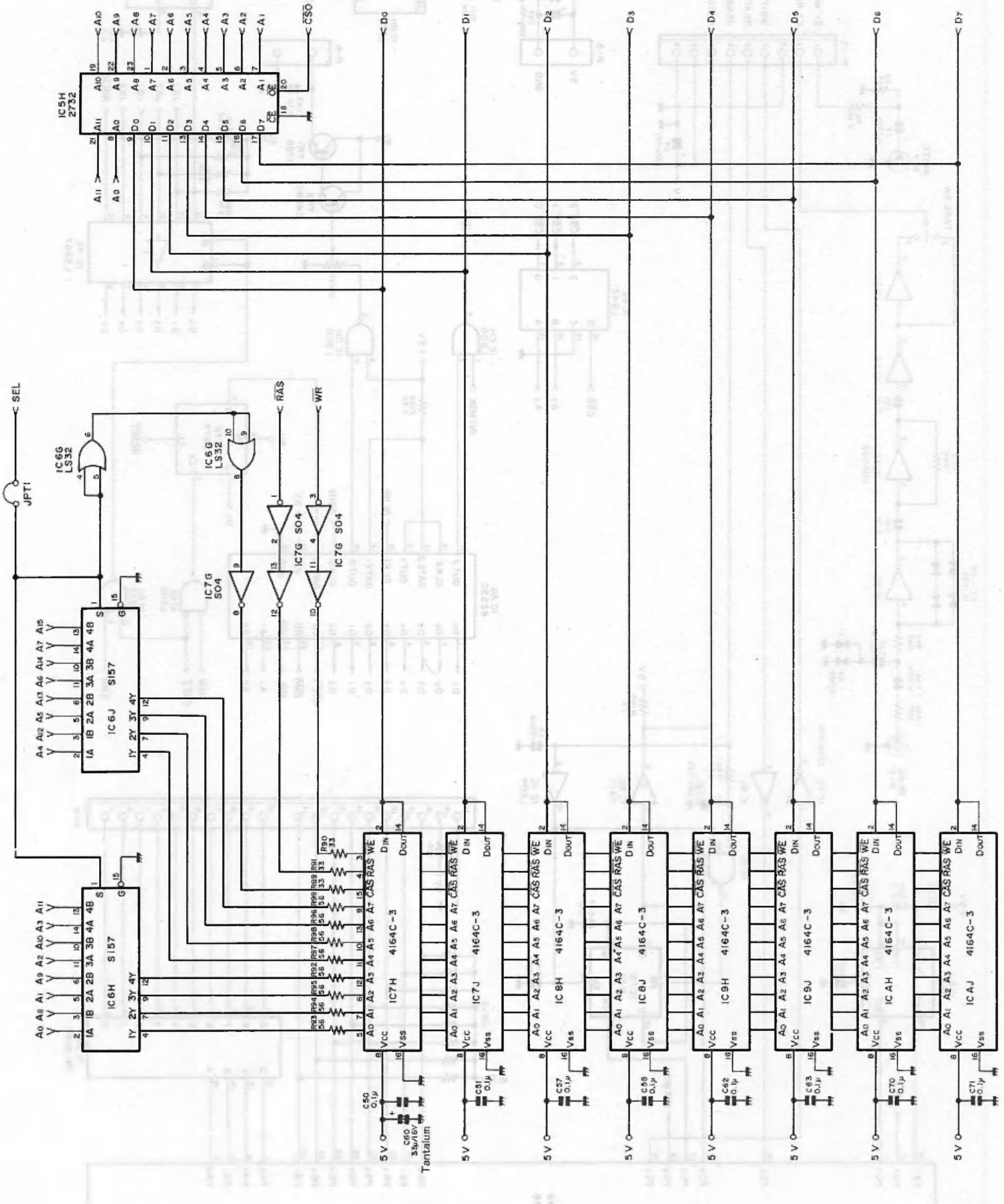
4. 5 MZ-700 Circuit Diagrams

[CPU board circuit (1)]

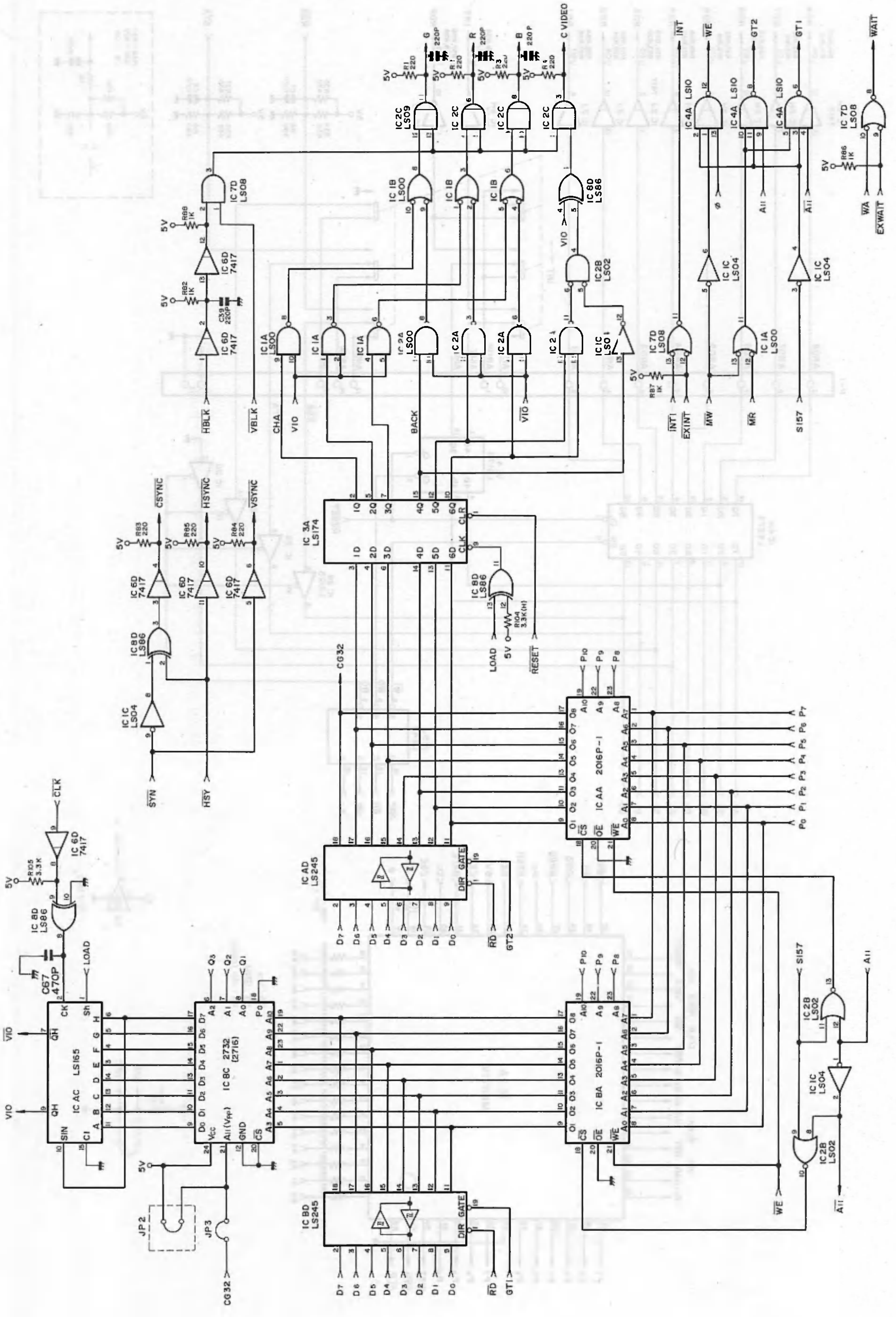


[CPU board circuit (2)]

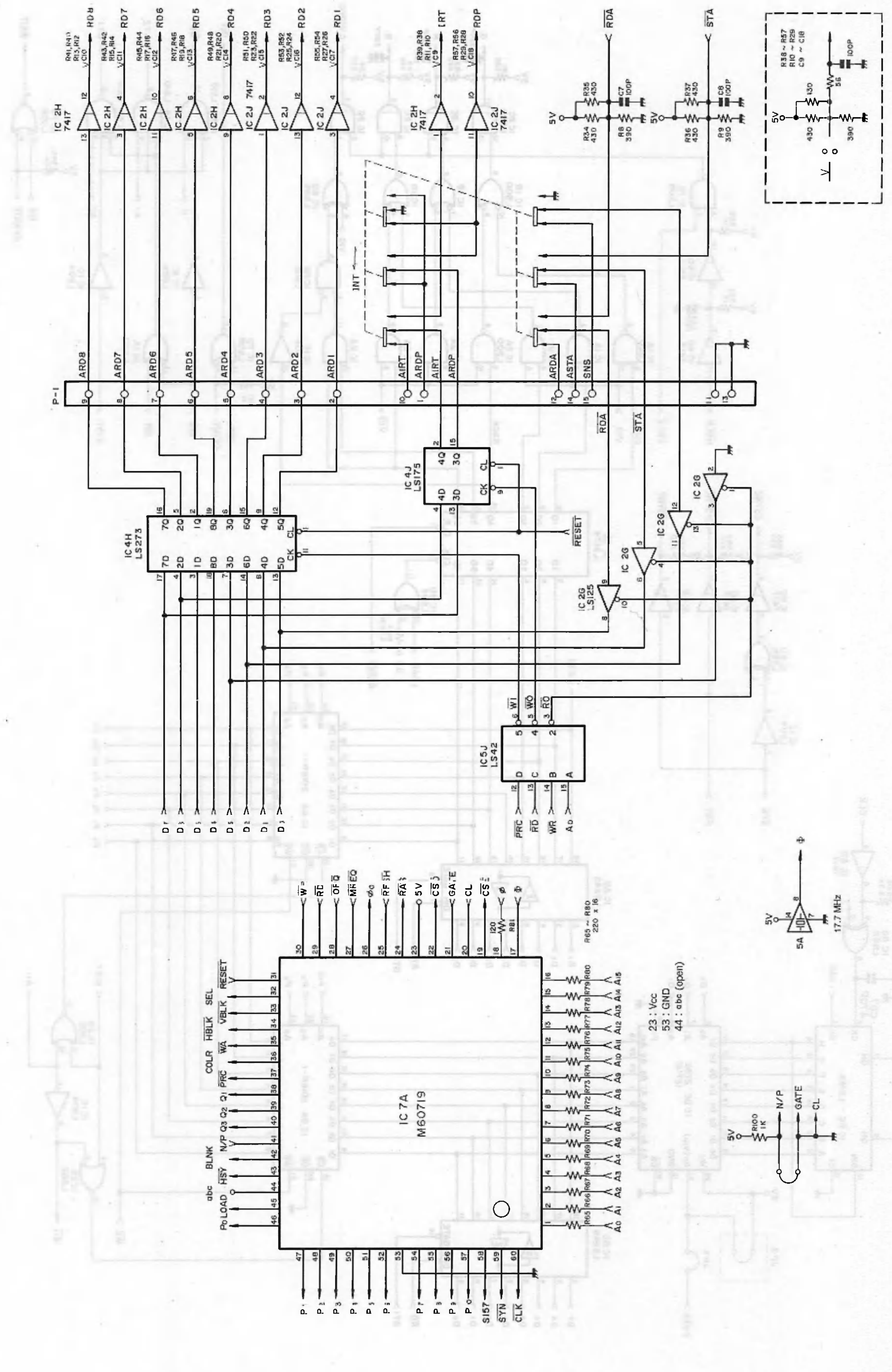


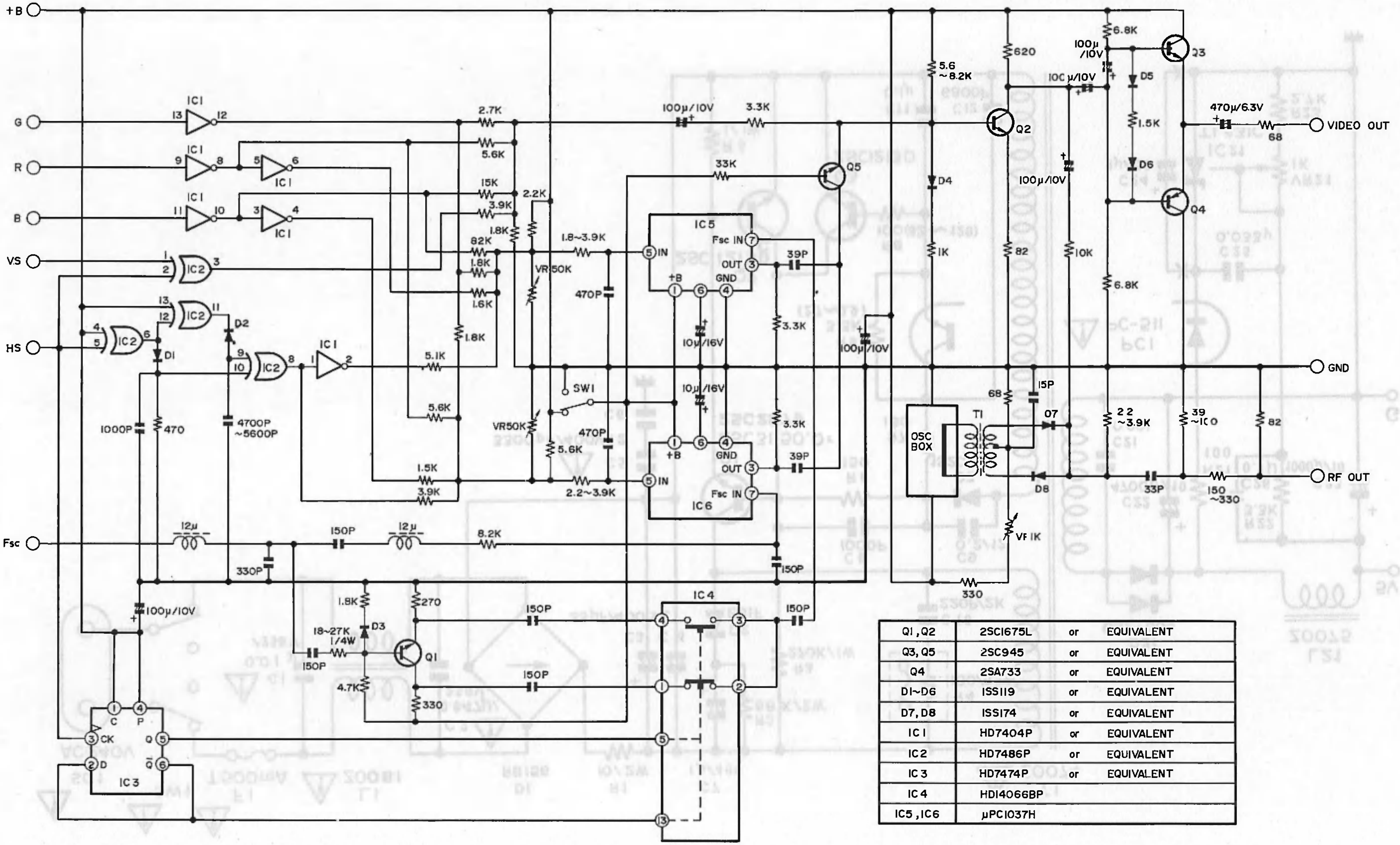

IC5H
2732
A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11
D0 D1 D2 D3 D4 D5 D6 D7
CE OE
CS0
SEL
JPT1
IC6G LS32
RAS
WR
IC7G S04
IC6J S157
IC6H S157
1A 1B 2A 2B 3A 3B 4A 4B
1Y 2Y 3Y 4Y
S G
R90 33
R91 33
R89 33
R99 56
R96 56
R97 56
R92 56
R95 56
R94 56
R93 56
A0 A1 A2 A3 A4 A5 A6 A7 CAS RAS WE
DIN DOUT
Vcc Vss
4164C-3
IC7H
IC7J
IC8H
IC8J
IC9H
IC9J
ICAH
ICAJ
5V
C50 0.1µ
C60 33µ/16V Tantalum
C51 0.1µ
C57 0.1µ
C58 0.1µ
C62 0.1µ
C63 0.1µ
C70 0.1µ
C71 0.1µ

[CPU board circuit (4)]



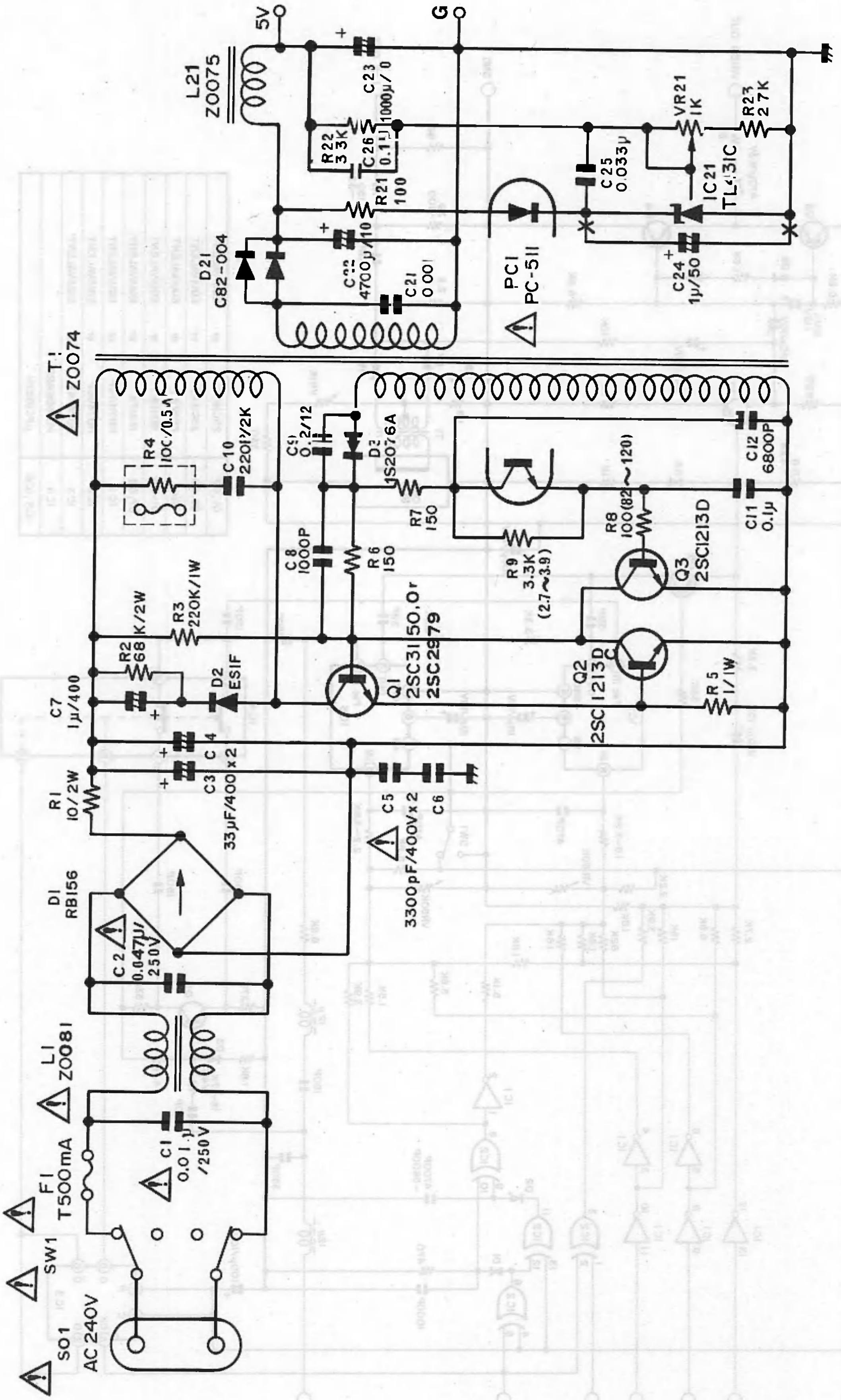
[CPU board circuit (5)]





Q1, Q2	2SC1675L	or	EQUIVALENT
Q3, Q5	2SC945	or	EQUIVALENT
Q4	2SA733	or	EQUIVALENT
D1~D6	1SS119	or	EQUIVALENT
D7, D8	1SS174	or	EQUIVALENT
IC1	HD7404P	or	EQUIVALENT
IC2	HD7486P	or	EQUIVALENT
IC3	HD7474P	or	EQUIVALENT
IC4	HD14066BP		
IC5, IC6	μPC1037H		

[Power unit]



SO1
AC 240V
SW1
F1
T500mA
C1
0.01μ
/250V
L1
Z0081
C2
0.047μ
250V
D1
RB156
R1
10/2W
C3 C4
33μF/400×2
C5
C6
3300pF/400V×2
C7
1μ/400
R2
68K/2W
R3
220K/1W
D2
ES1F
R4
10C/0.5A
C10
220P/2K
T1
Z0074
Q1
2SC3150,Or
2SC2979
R6
150
C8
1000P
C9
0.2/12
D3
1S2076A
R7
150
R9
3.3K
(2.7~3.9)
R8
100(82~120)
Q3
2SC1213D
Q2
2SC1213D
R5
1/1W
C11
0.1μ
C12
6800P
D21
C82-004
C22
4700μ/10
C21
0.001
R21
100
R22
3.3K
C26
0.1μ
C23
1000μ/10
L21
Z0075
5V
G
PC1
PC-511
C25
0.033μ
VR21
1K
R23
2.7K
IC21
TL431C
C24
1μ/50

P-1

▷ mark	1	ARDP
	2	ARD$_1$
	3	ARD$_2$
	4	ARD$_3$
	5	ARD$_4$
	6	ARD$_5$
	7	ARD$_6$
	8	ARD$_7$
	9	ARD$_8$
	10	AIRT
	11	GND
	12	ARDA
	13	GND
	14	ASTA
	15	ALPS

P-5

▷ mark	1	+ 5 V
	2	+ 5 V
	3	GND
	4	GND

P -10

▷ mark	1	RDP	2	GND
	3	RD$_1$	4	GND
	5	RD$_2$	6	GND
	7	RD$_3$	8	GND
	9	RD$_4$	10	GND
	11	RD$_5$	12	GND
	13	RD$_6$	14	GND
	15	RD$_7$	16	GND
	17	RD$_8$	18	GND
	19	IRT	20	GND
	21	RDA	22	GND
	23	STA	24	GND
	25	FG	26	FG

[CPU board terminal configuration]

P-11

49	A_{15}	$\text{N}\overline{\text{MI}}$	50
47	A_{14}	EXINT	48
45	A_{13}	GND	46
43	A_{12}	MREQ	44
41	A_{11}	GND	42
39	A_{10}	IORQ	40
37	A_{9}	GND	38
35	A_{8}	RD	36
33	A_{7}	GND	34
31	A_{6}	WR	32
29	A_{5}	$\overline{\text{EXWAIT}}$	30
27	A_{4}	$\text{M}\overline{1}$	28
25	A_{3}	GND	26
23	A_{2}	HALT	24
21	A_{1}	$\overline{\text{EXRESET}}$	22
19	A_{0}	RESET	20
17	BUS ⌀	GND	18
15	D_{7}	GND	16
13	D_{6}	GND	14
11	D_{5}	GND	12
9	D_{4}	GND	10
7	D_{3}	GND	8
5	D_{2}	GND	6
3	D_{1}	GND	4
1	D_{0}	GND	2

▷ mark

P-13

1	5V
2	VBLK
3	JA1
4	JA2
5	GND

P-14

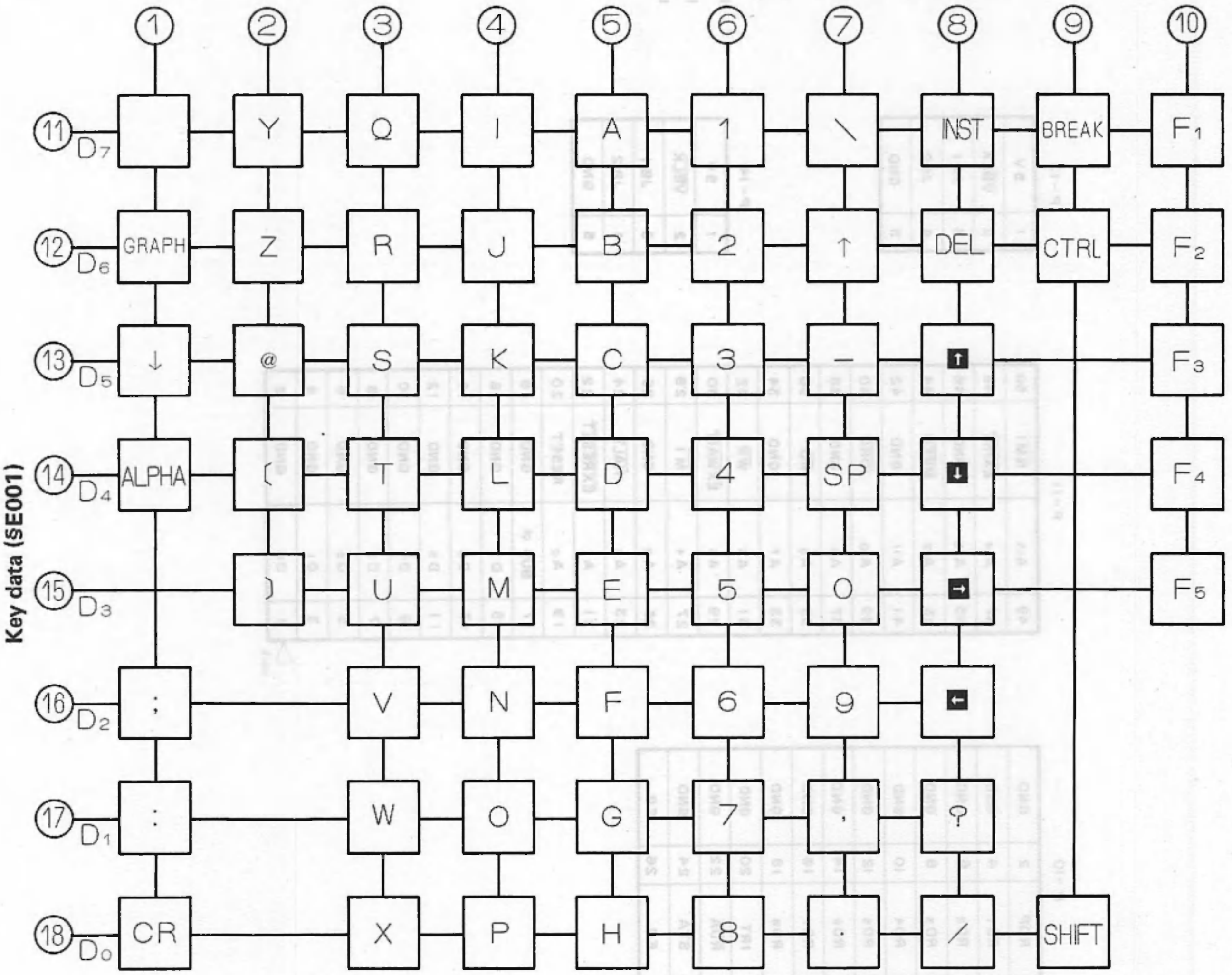
1	5V
2	VBLK
3	JB1
4	JB2
5	GND

P-9

1	○	GND
2	○	$\overline{\text{C SYNC}}$
3	○	C VIDEO
4	○	$\overline{\text{H SYNC}}$
5	○	$\overline{\text{V SYNC}}$
6	○	GND
7	○	+5V
8	○	G
9	○	B
10	○	R
11	○	COLR
12	○	GND

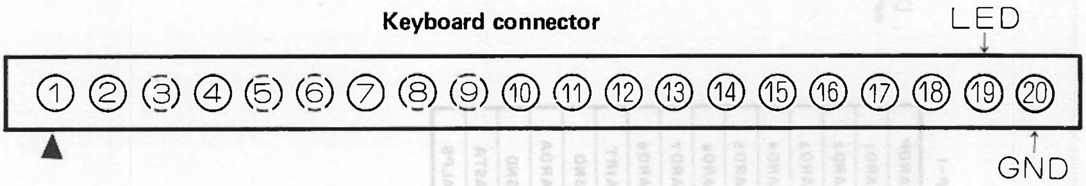
[Keyboard matrix circuit]

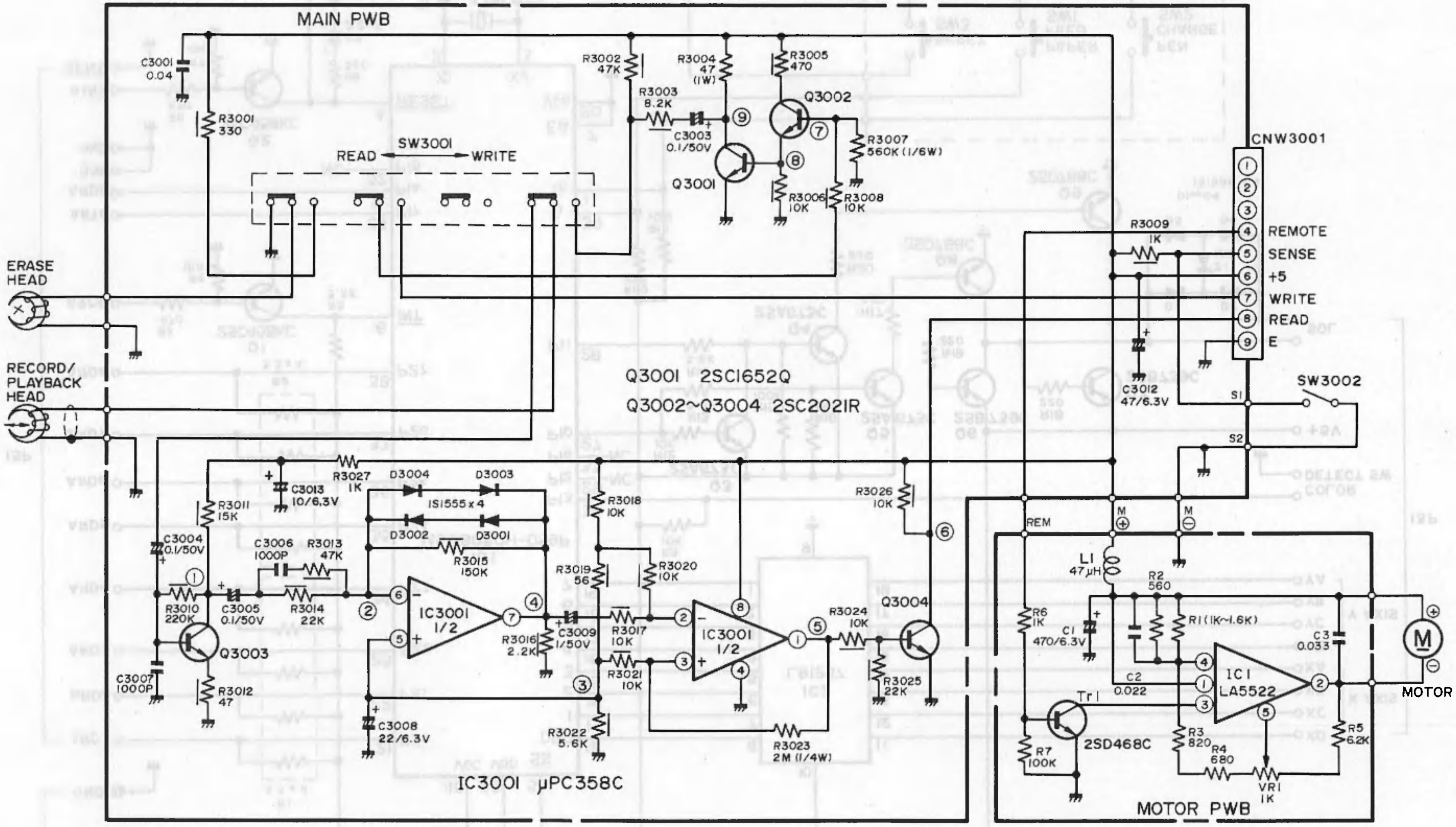
8255 outputs keyboard scan signals from port PA to the keyboard and reads key data from port PB. The figure below snows the key matrix.



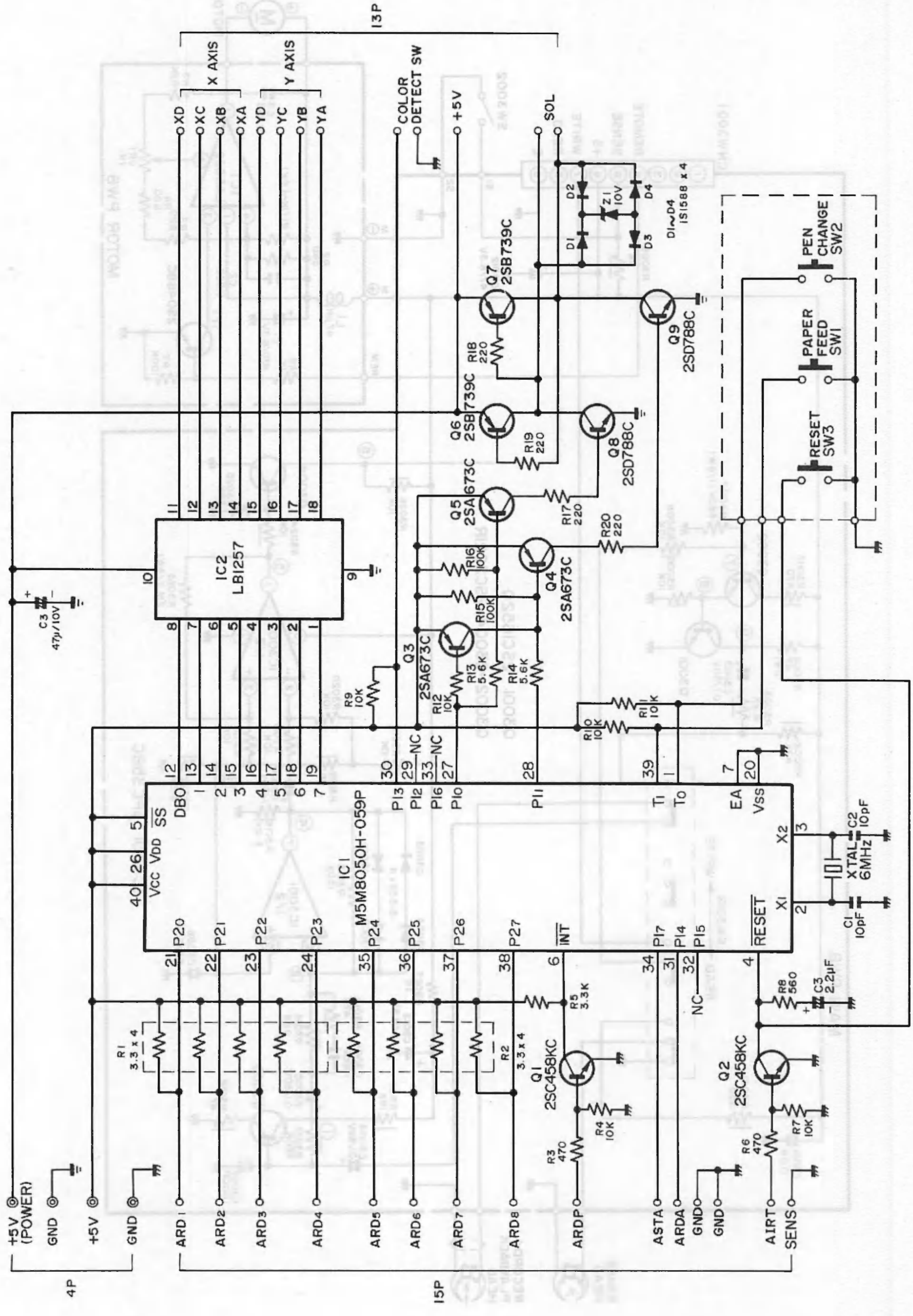
	①	②	③	④	⑤	⑥	⑦	⑧	⑨	⑩
⑪ D7		Y	Q	I	A	1	\	INST	BREAK	F1
⑫ D6	GRAPH	Z	R	J	B	2	↑	DEL	CTRL	F2
⑬ D5	↓	@	S	K	C	3	−	⬆		F3
⑭ D4	ALPHA	[	T	L	D	4	SP	⬇		F4
⑮ D3		]	U	M	E	5	0	➡		F5
⑯ D2	;		V	N	F	6	9	⬅		
⑰ D1	:		W	O	G	7	,	?		
⑱ D0	CR		X	P	H	8	.	/	SHIFT	

Keyboard connector




MAIN PWB
C3001 0.04
R3001 330
R3002 47K
R3003 8.2K
C3003 0.1/50V
R3004 47 (1W)
R3005 470
Q3002
Q3001
R3006 10K
R3007 560K (1/6W)
R3008 10K
SW3001
READ ← → WRITE
ERASE HEAD
RECORD/ PLAYBACK HEAD
CNW3001
REMOTE
SENSE
+5
WRITE
READ
E
R3009 1K
C3012 47/6.3V
SW3002
S1
S2
Q3001 2SC1652Q
Q3002~Q3004 2SC2021R
R3027 1K
C3013 10/6.3V
D3004
D3003
1S1555 x 4
D3002
D3001
R3015 150K
R3011 15K
C3004 0.1/50V
C3006 1000P
R3013 47K
R3010 220K
C3005 0.1/50V
R3014 22K
Q3003
C3007 1000P
R3012 47
IC3001 1/2
C3008 22/6.3V
R3016 2.2K
C3009 1/50V
R3018 10K
R3019 56
R3020 10K
R3017 10K
R3021 10K
R3022 5.6K
IC3001 μPC358C
R3023 2M (1/4W)
R3024 10K
R3025 22K
Q3004
R3026 10K
REM
M ⊕
M ⊖
L1 47μH
R2 560
R6 1K
C1 470/6.3V
R1 (1K~1.6K)
C3 0.033
C2 0.022
IC1 LA5522
Tr1
2SD468C
R7 100K
R3 820
R4 680
VR1 1K
R5 6.2K
M
MOTOR
MOTOR PWB

[Color plotter-printer circuit]



Chapter 5

Monitor Commands and Subroutines

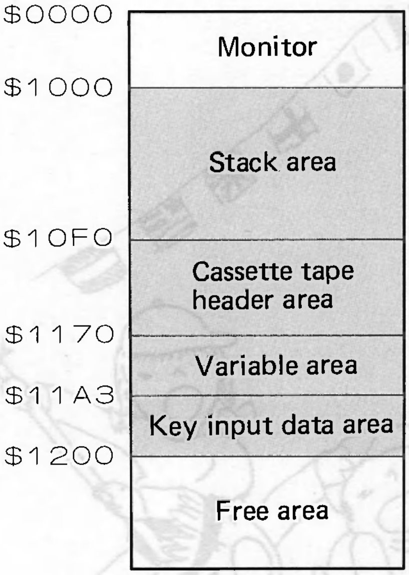
5.1 Monitor Commands

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.



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↵
⤓ 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↵
```

Prints the letters "ABC".

```
*P&T↵
```

Prints the test pattern.

```
*P&S↵
```

Sets the line width (character size) to 80 characters/line.

```
*P&L↵
```

Sets the line width (character size) to 40 characters/line.

```
*P&G↵
```

Switches the printer to the graphic mode.

```
*P&C↵
```

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.

```
*MCØØØ↵
CØØØ  ØØ  FF
CØØ1  ØØ  FF
CØØ2  ØØ  FF
CØØ3  ØØ  FF
CØØ4  ØØ      SHIFT + BREAK
*MCØ1Ø↵
CØ1Ø  ØØ  88
CØ11  ØØ  88
CØ12  ØØ  88
CØ13  ØØ  88
CØ14  ØØ      SHIFT + BREAK
*
```

To terminate the M command, simultaneously press the SHIFT and BREAK keys.

5.2.4 J command (J : Jump)

Format J hhhh

hhhh ...... destination address

Function This command transfers control to the specified address; i.e., it sets the specified address in the program counter.

```
*J12ØØ↲                 ..................... Jumps to address $1200.
```

5.2.5 S command (S : Save)

Format S hhhhh' h' h' h' h'' h'' h'' h''

hhhh.........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 hhhh 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.

```
*S600060A36050↲
FILENAME? MFILE↲
⊥ 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 " * " appeas.

5.2.6 V command (V : Verify)

Format	V
Function	Compares a machine language cassette file saved using the S command with the original program in memory.

```
*V↵
⤓ 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 to 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 simultaneoulsy pressing the RESET switch and the CTRL key.

```
*#↵
```

5.2.8 B command (B : Bell)

Format	B
Function	*B↵

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
CALL LETNL (0006)	Moves the cursor to the beginning of the next line.	Other than AF
CALL PRINTS (000C)	Displays a space at the cursor position.	Other than AF
CALL PRINTS (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
CALL MSG (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
CALL BELL (003E)	Briefly sounds high A (about 880 Hz).	Other AF
CALL MELDY (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 BREAK key. control is returned with the C flag set to 1.	Other than AF

CALL XTEMP
(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

CALL MSTA (0044)	Generates a continuous sound of the specified frequency. The frequency is given by the following equation. freq. = 895 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
CALL MSTP (0047)	Stops the sound generated with the CALL MSTA subroutine.	Other than AF
CALL TIMST (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
CALL TIMRD (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
CALL BRKEY (001E)	Checks whether the SHIFT and BREAK keys are both being pressed. The Z flag is set when they are being pressed simultaneoulsy; otherwise, it is reset.	Other than AF
CALL GETL (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 SHIFT and BREAK 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
CALL GETKY (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 DEL or CR) 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

	Special key	Code set in ACC	Display code
Special key read with GETKY	DEL	60	C7
	INST	61	C8
	ALPHA	62	C9
	BREAK	64	CB
	CR	66	CD
	↓	11	C1
	↑	12	C2
	→	13	C3
	←	14	C4
	HOME	15	C5
	CLR	16	C6

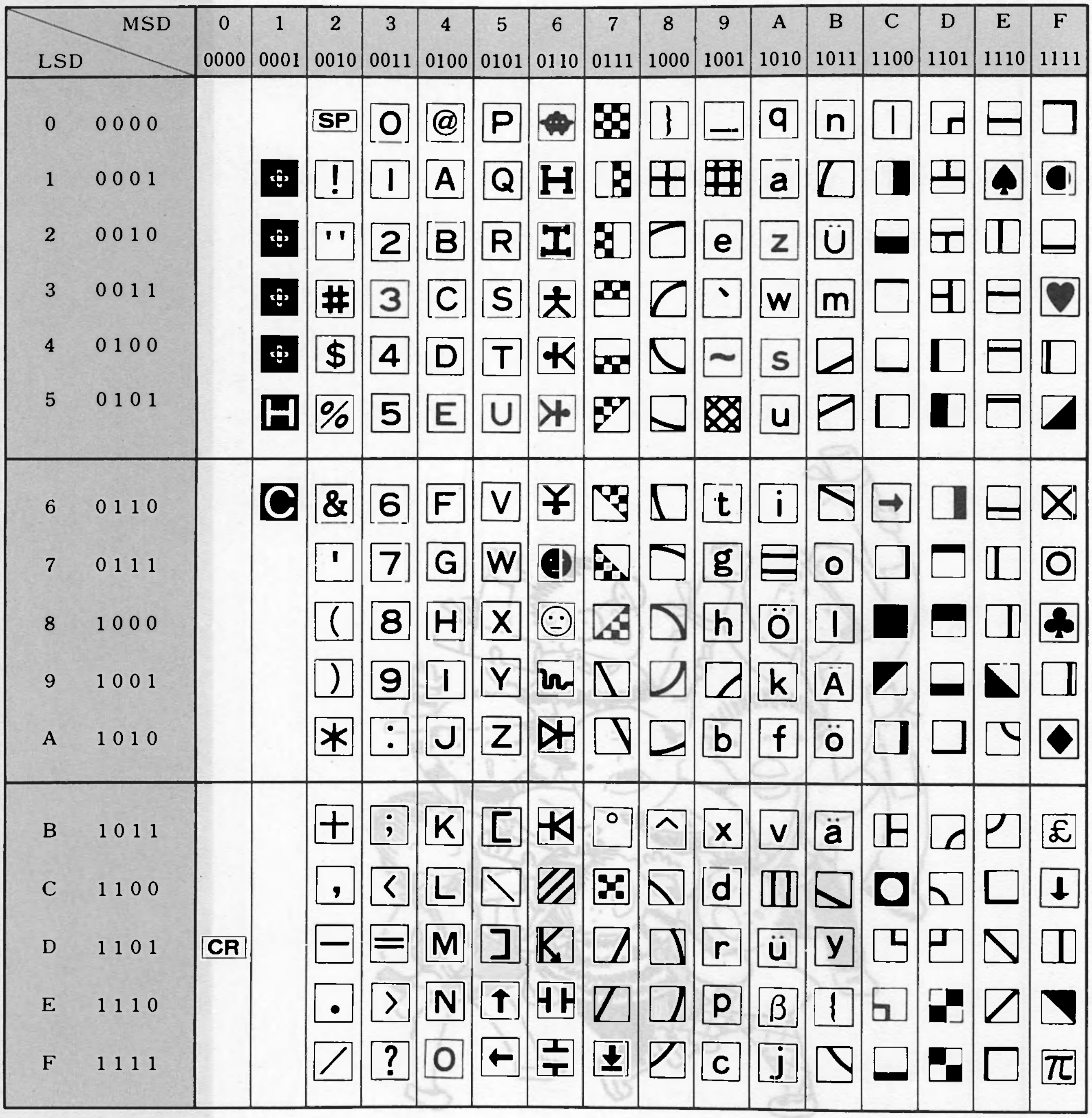
<table>
<tr><th>Name and entry point (hex.)</th><th>Function</th><th>Register saved</th></tr>
<tr><td>CALL ASC
(03DA)</td><td>Sets the ASCII character corresponding to the hexadecimal number represented by the lower 4 bits of data in ACC.</td><td>Other than AF</td></tr>
<tr><td>CALL HEX
(03F9)</td><td>Converts 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.</td><td>Other than AF</td></tr>
<tr><td>CALL HLHEX
(0410)</td><td>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 adress of the memory area which contains the ASCII character string
(e.g., "3" "1" "A" "5")
CALL HLHEX └ DE
CF = 0 HL ← hexadecimal number (e.g., HL = $31A5_H$)
CF = 1 The contents of HL are not assured.</td><td>Other than AF and HL</td></tr>
<tr><td>CALL 2HEX
(041F)</td><td>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 adress of the memory area which contains the ASCII character string. (e.g., "3" "A")
CALL 2HEX └ DE
CF = 0 ACC ← hexadecimal number (e.g., ACC = $3A_H$)
CF = 1 The contents of the ACC are not assured.</td><td>Other than AF and DE</td></tr>
<tr><td>CALL ??KEY
(09B3)</td><td>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.</td><td>Other than AF</td></tr>
<tr><td>CALL ?ADCN
(0BB9)</td><td>Converts ASCII codes into display codes. The call and return conditions are as follows.
ACC ← ASCII code
CALL ? ADCN
ACC ← Display code</td><td>Other than AF</td></tr>
<tr><td>CALL ?DACN
(0BCE)</td><td>Converts display codes into ASCII codes. The call and return conditions are as follows.
ACC ← Display code
CALL ? DACN
ACC ← ASCII code</td><td>Other than AF</td></tr>
<tr><td>CALL ?BLNK
(0DA6)</td><td>Detects the vertical blanking period. Control is returned to the calling program when the vertical blanking period is entered.</td><td>All registers</td></tr>
<tr><td>CALL ?DPCT
(0DDC)</td><td>Controls display as follows.
<table>
<tr><th>ACC</th><th>Control</th><th>ACC</th><th>Control</th></tr>
<tr><td>$C0_H$</td><td>Scrolling</td><td>$C6_H$</td><td>Same as the CLR key.</td></tr>
<tr><td>$C1_H$</td><td>Same as the ⬇ key.</td><td>$C7_H$</td><td>Same as the DEL key.</td></tr>
<tr><td>$C2_H$</td><td>Same as the ⬆ key.</td><td>$C8_H$</td><td>Same as the INST key.</td></tr>
<tr><td>$C3_H$</td><td>Same as the ➡ key.</td><td>$C9_H$</td><td>Same as the ALPHA key.</td></tr>
<tr><td>$C4_H$</td><td>Same as the ⬅ key.</td><td>CD_H</td><td>Same as the CR key.</td></tr>
<tr><td>$C5_H$</td><td>Same as the HOME key.</td><td></td><td></td></tr>
</table></td><td>All registers</td></tr>
<tr><td>CALL ?PONT
(0FB1)</td><td>Sets the current cursor location in the HL register. The return conditions are as follows.
CALL ? PONT
HL ← Cursor location (binary)</td><td>Other than AF and HL</td></tr>
</table>

APPENDICES

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 11H to 16H are cursor control codes. For example, executing CALL PRNT (a monitor subroutine) with 15H set in ACC returns the cursor to the home position. (" H " is not displayed.)



LSD \ MSD	0 0000	1 0001	2 0010	3 0011	4 0100	5 0101	6 0110	7 0111	8 1000	9 1001	A 1010	B 1011	C 1100	D 1101	E 1110	F 1111
0 0000			SP	0	@	P	◓	▦	}	_	q	n	│	▗	━	□
1 0001		■	!	1	A	Q	H	▐	┼	▦	a	⟋	▐	┴	♠	◖
2 0010		■	"	2	B	R	I	▌	◜	e	z	Ü	▄	┬	┃	▁
3 0011		■	#	3	C	S	人	▀	◜	`	w	m	▔	┤	━	♥
4 0100		■	$	4	D	T	K	▄	◟	~	s	⟋	▁	▏	▔	▏
5 0101		H	%	5	E	U	⊁	◩	◡	▩	u	⟋	□	▌	▔	◢
6 0110		C	&	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	[	K	°	^	x	v	ä	┣	◞	◜	£
C 1100			,	<	L	\	▨	⁙	⟍	d	▥	⟍	◙	◟	□	↓
D 1101	CR		-	=	M	]	K	⟋	)	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 (0012H) and MSG (0015H) convert ASCII codes into display codes and transfer them to the V-RAM location indicated for the cursor.

Codes C1H to C6H are for controlling the cursor.

LSD \ MSD		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	1	▏	♠	<	!	□	a	q	☰	◜	◈	▐	⊣	▘
2	0010	B	R	2	▁	◥	[	"	▔	b	r	Ⅲ	◝	◈	▌	⊥	▝
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	▩	◞	C	◪	≻	▞
7	0111	G	W	7	▌	●	>	'	╲	g	w	╱	)	👾	◣	⊳	▛
8	1000	H	X	8	▁	○	↓	(	━	h	x	╱	◝	H	◢	⊲	▗
9	1001	I	Y	9	▕	?	╲	)	▕	i	y	╲	◡	I	╲	K	▚
A	1010	J	Z	—	▄	◙	→	+	▀	j	z	ß	◡	🯅	╲	K	▐
B	1011	K	£	=	▐	◞	▚	*	▌	k	ä	ü	╱	⊸	°	≻	▜
C	1100	L	▝	;	▁	◟	▗	▞	▁	l	◞	ö	{	⟜	⁙	⊣⊢	▄
D	1101	M	▘	/	▕	◣	▖	☒	▕	m	◜	Ü	⊞	¥	╱	÷	▙
E	1110	N	├	.	▂	◢	┤	◜	▃	n	◝	Ä	^	☻	╱	Σ	▟
F	1111	O	┴	,	▐	:	┬	◝	▐	o	◟	Ö	_	☺	∿	▒	█

The character patterns on the former page are contained in the 2K bytes which make up the first half of CG-ROM. Character patterns for the second half of CG-ROM are shown on the latter page. However, character patterns in the second 2K bytes of the CG-ROM are not supported by BASIC, and cannot be entered directly from the keyboard. Although they can be displayed using the POKE statement as shown in the example below, they cannot be output to any printer (either the built-in printer or an external printer).

< Examples >

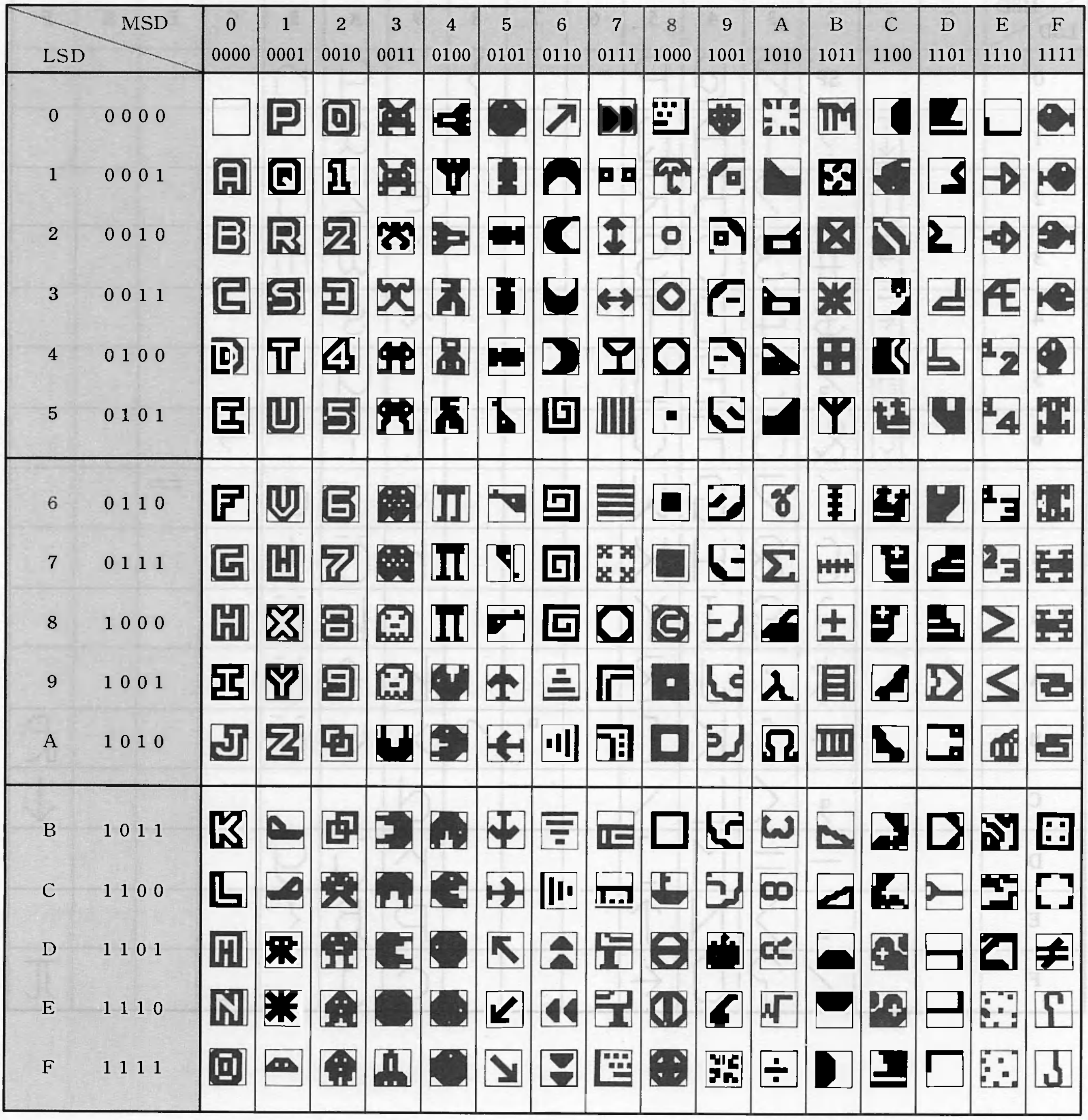
(1) The following program example displays character patterns from the second half of CG-ROM on the CRT screen.

```
10  COLOR, , 7, 0
20  PRINT"🅲";
30  FOR J=55296 TO 56296 ——→ 55296 = $D800
40  POKE J, 24Ø ——————————→ Specifies the second 2K-byte
50  NEXT J                    half of CG-ROM. 240 = $F0
60  A=53248:I=O:H=O ——————→ 53248 = $D000
70  POKE A, I
80  A=A+2
90  I=I+1:IF I=256 THEN GOTO 12Ø
100 H=H+1:IF H=2Ø THEN A=A+4Ø:H=Ø
110 GOTO 7Ø
120 GOTO 12Ø
```

(2) The example below illustrates using machine language to display character patterns from the second half of CG-ROM on the CRT screen.

```
LD HL, DØØØH              DISP  :XOR A
CALL DISP                        LD B, ØØH
LD HL, D2Ø8H              DISP2 :LD (HL), A
CALL DISP                        INC HL
LD A, F1H                        INC A
LD HL, DAØ8H                     DEC B
LD DE, DAØ9H                     JP NZ, DISP2
LD BC, ØØFFH                     RET
LD (HL), A
LDIR
END
```

■ MZ-700 Display code table (second 2K-byte half)



MSD / LSD		0 0000	1 0001	2 0010	3 0011	4 0100	5 0101	6 0110	7 0111	8 1000	9 1001	A 1010	B 1011	C 1100	D 1101	E 1110	F 1111
0	0000		P	0				↗									
1	0001	A	Q	1													
2	0010	B	R	2					↕								
3	0011	C	S	3					↔								
4	0100	D	T	4													
5	0101	E	U	5													
6	0110	F	V	6													
7	0111	G	W	7								Σ					
8	1000	H	X	8						©			±			≥	
9	1001	I	Y	9												≤	
A	1010	J	Z									Ω					
B	1011	K															
C	1100	L															
D	1101	M					↖										≠
E	1110	N					↙										
F	1111	O					↘					÷					

- **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.

LSD \ MSD	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		C	&	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	]				r	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:	PAL system
	RGB signal
	Composite signal (B/W)
	RF signal (UHF 36 ± 3 CH, B/W)
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
	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 ±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 ±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
8-bit load group		
LD r, r'	r←r'	01 r r'
LD r, n	r←n	00 r 110 ← n →
LD r, (HL)	r←(HL)	01 r 110
LD r, (IX+d)	r←(IX+d)	11 011 101 01 r 110 ← d →
LD r, (IY+d)	r←(IY+d)	11 111 101 01 r 110 ← d →
LD (HL), r	(HL)←r	01 110 r
LD (IX+d), r	(IX+d)←r	11 011 101 01 110 r ← d →
LD (IY+d), r	(IY+d)←r	11 111 101 01 110 r ← d →
LD (HL), n	(HL)←n	00 110 110 ← n →
LD (IX+d), n	(IX+d)←n	11 011 101 00 110 110 ← d → ← n →
LD (IY+d), n	(IY+d)←n	11 111 101 00 110 110 ← d → ← n →
LD A, (BC)	A←(BC)	00 001 010
LD A, (DE)	A←(DE)	00 011 010
LD A, (nn)	A←(nn)	00 111 010 ← n → ← n →
LD (BC), A	(BC)←A	00 000 010
LD (DE), A	(DE)←A	00 010 010
LD (nn), A	(nn)←A	00 110 010 ← n → ← n →
LD A, I	A←I	11 101 101 01 010 111
LD A, R	A←R	11 101 101 01 011 111
LD I, A	I←A	11 101 101 01 000 111
LD R, A	R←A	11 101 101 01 001 111
16-bit load group		
LD dd, nn	dd←nn	00 dd0 001 ← n → ← n →
LD IX, nn	IX←nn	11 011 101 00 100 001 ← n → ← n →
LD IY, nn	IY←nn	11 111 101 00 100 001 ← n → ← n →

Mnemonic	Symbolic operation	Op-code

LD HL, (nn)	H←(nn+1)
	L←(nn)
		00 101 010
		← n →
		← n →

LD dd, (nn)	dd_H←(nn+1) dd_L←(nn)	11 101 101 01 dd1 011 ← n → ← n →
LD IX, (nn)	IX_H←(nn+1) IX_L←(nn)	11 011 101 00 101 010 ← n → ← n →
LD IY, (nn)	IY_H←(nn+1) IY_L←(nn)	11 111 101 00 101 010 ← n → ← n →
LD (nn), HL	(nn+1)←H (nn)←L	00 100 010 ← n → ← n →
LD (nn), dd	(nn+1)←dd_H (nn)←dd_L	11 101 101 01 dd0 011 ← n → ← n →
LD (nn), IX	(nn+1)←IX_H (nn)←IX_L	11 011 101 00 100 010 ← n → ← n →
LD (nn), IY	(nn+1)←IY_H (nn)←IY_L	11 111 101 00 100 010 ← n → ← n →
LD SP, HL	SP←HL	11 111 001
LD SP, IX	SP←IX	11 011 101 11 111 001
LD SP, IY	SP←IY	11 111 101 11 111 001
PUSH qq	(SP−2)←qq_L (SP−1)←qq_H	11 qq0 101
PUSH IX	(SP−2)←IX_L (SP−1)←IX_H	11 011 101 11 100 101
PUSH IY	(SP−2)←IY_L (SP−1)←IY_H	11 111 101 11 100 101
POP qq	qq_H←(SP+1) qq_L←(SP)	11 qq0 001
POP IX	IX_H←(SP+1) IX_L←(SP)	11 011 101 11 100 001
POP IY	IY_H←(SP+1) IY_L←(SP)	11 111 101 11 100 001
Exchange group and block transfer and search group		
EX DE, HL	DE↔HL	11 101 011
EX AF, AF'	AF↔AF'	00 001 000
EXX	(BC)↔(BC') (DE)↔(DE') (HL)↔(HL')	11 011 001
EX (SP), HL	H↔(SP+1) L↔(SP)	11 100 011
EX (SP), IX	IX_H↔(SP+1) IX_L↔(SP)	11 011 101 11 100 011
EX (SP), IY	IY_H↔(SP+1) IY_L↔(SP)	11 111 101 11 100 011

Mnemonic	Symbolic operation	Op-code

LDI	(DE)←(HL)
	DE←DE+1
	HL←HL+1
	BC←BC−1
		11 101 101
		10 100 000

LDIR	(DE)←(HL)
	DE←DE+1
	HL←HL+1
	BC←BC−1
	Repeat until BC=0
		11 101 101
		10 110 000

LDD	(DE)←(HL)
	DE←DE−1
	HL←HL−1
	BC←BC−1
		11 101 101
		10 101 000

LDDR	(DE)←(HL)
	DE←DE−1
	HL←HL−1
	BC←BC−1
	Repeat until BC=0
		11 101 101
		10 111 000

CPI	A−(HL)
	HL←HL+1
	BC←BC−1
		11 101 101
		10 100 001

CPIR	A−(HL) HL←HL+1 BC←BC−1 Repeat until A= (HL) or BC=0	11 101 101 10 110 001

CPD	A−(HL)
	HL←HL−1
	BC←BC−1
		11 101 101
		10 101 001

CPDR	A−(HL) HL←HL−1 BC←BC−1 Repeat until A= (HL) or BC=0	11 101 001 10 111 001

8-bit arithmetic and logical group

Mnemonic	Symbolic operation	Op-code
ADD A, r	A←A+r	10 $\boxed{000}$ r
AD A, n	A←A+n	11 $\boxed{000}$ 110 ← n →
ADD A, (HL)	A←A+(HL)	10 $\boxed{000}$ 110
ADD A, (IX+d)	A←A, (IX+d)	11 011 101 10 $\boxed{000}$ 110 ← d →
ADD A, (IY+d)	A←A+(IY+d)	11 111 101 10 $\boxed{000}$ 110 ← d →
ADC A, s	A←A+s+CY	$\boxed{001}$
SUB s	A←A−s	$\boxed{010}$
SBC A, s	A←A−s−CY	$\boxed{011}$
AND s	A=A∧s	$\boxed{100}$
OR s	A←A∨s	$\boxed{110}$
XOR s	A←A⊕s	$\boxed{101}$
CP s	A−s	$\boxed{111}$
INC r	r←r+1	00 r $\boxed{100}$
INC (HL)	(HL)←(HL)+1	00 110 $\boxed{100}$
INC (IX+d)	(IX+d) ←(IX+d)+1	11 011 101 00 110 $\boxed{100}$ ← d →
INC (IY+d)	(IY+d) ←(IY+d)+1	11 111 101 00 110 $\boxed{100}$

Mnemonic	Symbolic operation	Op-code
DEC m	m←m−1	← d → $\boxed{101}$

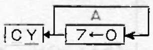
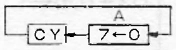
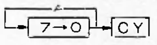
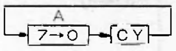
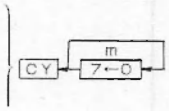
General purpose arithmetic and control group

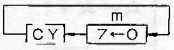
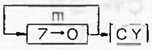
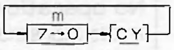
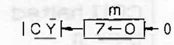
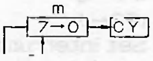
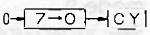
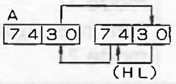
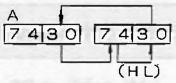
Mnemonic	Symbolic operation	Op-code
DAA	Decimal adjustment upon contents of A after add or subtract	00 100 111
CPL	A←$\overline{A}$	00 101 111
NEG	A←$\overline{A}$+1	11 101 101 01 000 100
CCF	CY←C$\overline{Y}$	00 111 111
SCF	CY←1	00 110 111
NOP	No operation, but PC is incremented.	00 000 000
HALT	CPU halted	01 110 110
DI	IFF←0	11 110 011
EI	IFF←1	11 111 011
IM0	Set interrupt mode 0	11 101 101 01 000 110
IM1	Set interrupt mode 1	11 101 101 01 010 110
IM2	Set interrupt mode·2	11 101 101 01 011 110

16-bit arithmetic group

Mnemonic	Symbolic operation	Op-code
ADD HL, ss	HL←HL+ss	00 ss1 001
ADC HL, ss	HL←HL+ss+CY	11 101 101 01 ss1 010
SBC HL, ss	HL←HL−ss−CY	11 101 101 01 ss0 010
ADD IX, pp	IX←IX+pp	11 011 101 00 pp1 001
ADD IY, rr	IY←IY+rr	11 111 101 00 rr1 001
INC ss	ss←ss+1	00 ss0 011
INC IX	IX←IX+1	11 011 101 00 100 011
INC IY	IY←IY+1	11 111 101 00 100 011
DEC ss	ss←ss−1	00 ss1 011
DEC IX	IX←IX−1	11 011 101 00 101 011
DEC IY	IY←IY−1	11 111 101 00 101 011

Rotate and shift group

Mnemonic	Symbolic operation	Op-code
RLCA		00 000 111
RLA		00 010 111
RRCA		00 001 111
RRA		00 011 111
RLC r		11 001 011 00 $\boxed{000}$ r
RLC (HL)		11 001 011 00 $\boxed{000}$ 110

Mnemonic	Symbolic operation	Op-code
RLC (IX+d)		11 011 101 11 001 011 ← d → 00 $\boxed{000}$ 110
RLC (IY+d)		11 111 011 11 001 011 ← d → 00 $\boxed{000}$ 110
RL m		$\boxed{010}$
RRC m		$\boxed{001}$
RR m		$\boxed{011}$
SLA m		$\boxed{100}$
SRA m		$\boxed{101}$
SRL m		$\boxed{111}$
RLD		11 101 101 01 101 111
RRD		11 101 101 01 100 111
Bit set, reset and test group		
BIT b, r	Z←$\overline{rb}$	11 001 011 01 b r
BIT b, (HL)	Z←$\overline{(HL)b}$	11 011 011 01 b 110
BIT b, (IX+d)	Z←$\overline{(IX+d)b}$	11 011 101 11 001 011 ← d → 01 b 110
BIT b, (IY+d)	Z←$\overline{(IY+d)b}$	11 111 101 11 001 011 ← d → 01 b 110
SET b, r	rb←1	11 001 011 $\boxed{11}$ b r
SET b, (HL)	(HL)b←1	11 001 011 $\boxed{11}$ b 110
SET b, (IX+d)	(IX+d)b←1	11 001 101 11 001 011 ← d → $\boxed{11}$ b 110
SET b, (IY+d)	(IY+d)b←1	11 111 101 11 001 011 ← d → $\boxed{11}$ b 110
RES b, m	mb←0	$\boxed{10}$

Mnemonic	Symbolic operation	Op-code
Jump group		
JP nn	PC←nn	11 000 011 ← n → ← n →
JP cc, nn	If condition cc is true, PC< −nn; otherwise, continue	11cc 010 ← n → ← n →
JR e	PC←PC+e	00 011 000 ← e-2 →
JR C, e	If C=0, continue. If C=1, PC←PC+e	00 111 000 ← e-2 →
JR Z, e	If Z=0, continue. If C=1, PC←PC+e	00 101 000 ← e-2 →
JR NC, e	If C=1, continue. If C=0, PC←PC+e	00 110 000 ← e-2 →
JR NZ, e	If Z=1, continue. If Z=0, PC←PC+e	00 100 000 ← e-2 →
JP (HL)	PC←HL	11 101 001
JP (IX)	PC←IX	11 011 101 11 101 001
JP (IY)	PC←IY	11 111 101 11 101 001
DJNZ e	B←B−1 If B=0, continue; otherwise, PC←PC+e	00 010 000 ← e-2 →
Call and return group		
CALL nn	(SP−1)←PC_H (SP−2)←PC_L PC←nn	11 001 101 ← n → ← n →
CALL cc, nn	If condition cc is false, continue; otherwise same as CALL nn.	11 cc 100 ← n → ← n →
RET	PC_L←(SP) PC_H←(SP+1)	11 001 001
RET cc	If condition cc is false, continue; otherwise same as RET.	11 cc 000
RETI	Return from interrupt	11 101 101 01 001 101
RETN	Return from NMI.	11 101 101 01 000 101
RST p	(SP−1)←PC_H (SP−2)←PC_L PC_H←0 PC_L←p	11 t 111

Mnemonic	Symbolic operation	Op-code
Input and output group		
IN A, (n)	A←(n)	11 011 011 ← n →
IN r, (C)	r←(C)	11 101 101 01 r 000

INI	(HL)←(C)
	B←B−1
	HL←HL+1
		11 101 101
		10 100 010

INIR	(HL)←(C)
	B←B−1
	HL←HL+1
	Repeat until B=0
		11 101 101
		10 110 010

IND	(HL)←(C)
	B←B−1
	HL←HL−1
		11 101 101
		10 101 010

INDR	(HL)←(C)
	B←B−1
	HL←HL−1
	Repeat until B=0
		11 101 101
		10 111 010

OUT (n), A	(n)←A	11 010 011 ← n →
OUT (C), r	(C)←r	11 101 101 01 r 001

OUTI	(C)←(HL)
	B←B−1
	HL←HL+1
		11 101 101
		10 100 011

OTIR	(C)←(HL)
	B←B−1
	HL←HL+1
	Repeat until B=0
		11 101 101
		10 110 011

OUTD	(C)←(HL)
	B←B−1
	HL←HL−1
		11 101 101
		10 101 011

OTDR	(C)←(HL)
	B←B−1
	HL←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
000	B
001	C
010	D
011	E
100	H
101	L
111	A

dd, ss	Register pair
00	BC
01	DE
10	HL
11	SP

qq	Register pair
00	BC
01	DE
10	HL
11	AF

pp	Register pair
00	BC
01	DE
10	IX
11	SP

rr	Register pair
00	BC
01	DE
10	IY
11	SP

b	Bit set
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

cc	Condition	
000	NZ	non zero
001	Z	zero
010	NC	non carry
011	C	carry
100	PO	parity odd
101	PE	parity even
110	P	sign positive
111	M	sign negative

t	p
000	00H
001	08H
010	10H
011	18H
100	20H
101	28H
110	30H
111	38H

∧ : AND operation
∨ : OR operation
⊕ : Exclusive OR operation

s: r, n, (HL), (IX + d), (IY + d)
CY: Carry flip-flop
(register pair)$_H$: Upper 8 bits of register pair

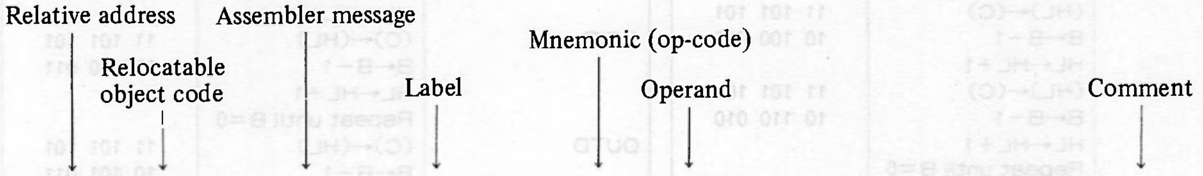
m : r, (HL), (IX + d), (IY + d)
m$_b$: Bit b or location m
(register pair)$_L$: Lower 8 bits of register pair

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.



```
02A7 13                 INC   DE
02A8 13                 INC   DE
02A9 13                 INC   DE
02AA C9                 RET
02AB            ;
02AB            ;
02AB            ;
02AB            ;ORG 02ABH ; MLDST
02AB            ;
02AB            ;   MELODY START & STOP
02AB            ;
02AB            MLDST:  ENT
02AB 2AA111             LD    HL,(RATIO)
02AE 7C                 LD    A,H
02AF B7                 OR    A
02B0 280C               JR    Z,MLDSP
02B2 D5                 PUSH  DE
02B3 EB                 EX    DE,HL
02B4 2104E0             LD    HL,CONTO
02B7 73                 LD    (HL),E
02B8 72                 LD    (HL),D
02B9 3E01               LD    A,1
02BB D1                 POP   DE
02BC 1806               JR    MLDS1
02BE            ;
02BE            MLDSP:  ENT
02BE 3E36               LD    A,36H          ; MODE SET (8253 CO)
02C0 3207E0             LD    (CONTF),A      ; E007H
02C3 AF                 XOR   A
02C4 3208E0     MLDS1:  LD    (SUNDG),A      ; E008H
02C7 C9                 RET                  ; TEHRO RESET
```

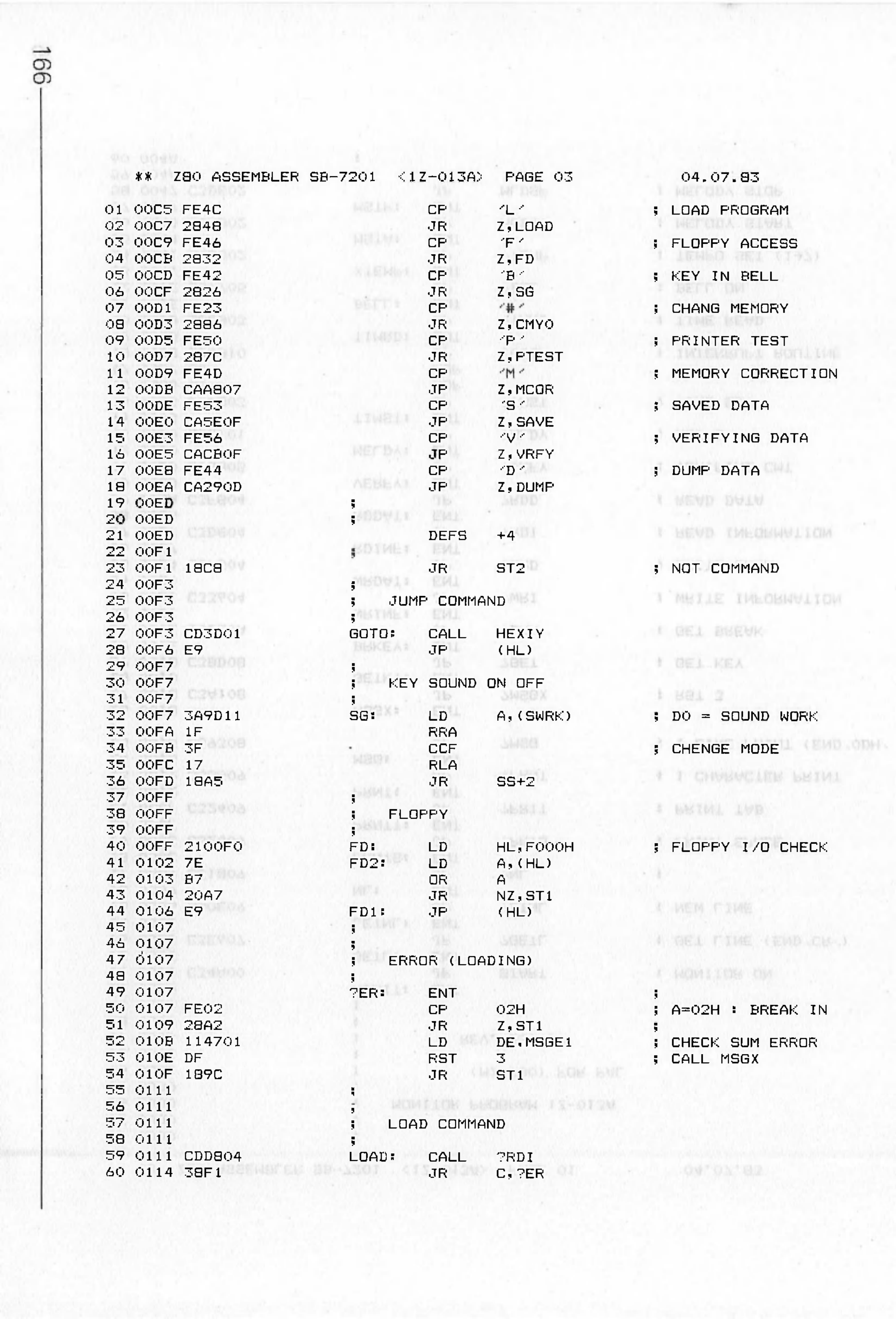
Since the starting address of Monitor 1Z-013A is set to $0000, relocatable addresses and object codes in the assembly list can be assumed as absolute addresses and object code, respectively.

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.

```
  ** Z80 ASSEMBLER SB-7201  <1Z-013A>  PAGE 01           04.07.83
01 0000                    ;
02 0000                    ;
03 0000                    ;   MONITOR PROGRAM 1Z-013A
04 0000                    ;
05 0000                    ;       (MZ-700) 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´OD
)
26 0018                   MSGX:   ENT
27 0018 C3A108                    JP      ?MSGX             ; RST 3
28 001B                   GETKY:  ENT
29 001B C3BD08                    JP      ?GET              ; GET KEY
30 001E                   BRKEY:  ENT
31 001E C3320A                    JP      ?BRK              ; GET BREAK
32 0021                   WRINF:  ENT
33 0021 C33604                    JP      ?WRI              ; WRITE INFORMATION
34 0024                   WRDAT:  ENT
35 0024 C37504                    JP      ?WRD              ; WRITE DATA
36 0027                   RDINF:  ENT
37 0027 C3D804                    JP      ?RDI              ; READ INFORMATION
38 002A                   RDDAT:  ENT
39 002A C3F804                    JP      ?RDD              ; READ DATA
40 002D                   VERFY:  ENT
41 002D C38805                    JP      ?VRFY             ; VERIFING CMT
42 0030                   MELDY:  ENT
43 0030 C3C701                    JP      ?MLDY             ; RST 6
44 0033                   TIMST:  ENT
45 0033 C30803                    JP      ?TMST             ; TIME SET
46 0036 00                        NOP
47 0037 00                        NOP
48 0038 C33810                    JP      1038H             ; 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                    ;
```

```
004A                  ;
004A          START:  ENT
004A 31F010           LD    SP,SP        ; STACK SET (10F0H)
004D ED56             IM    1            ; IM 1 SET
004F CD3E07           CALL  ?MODE        ; 8255,8253 MODE SET
0052 CD320A           CALL  ?BRK         ; CTRL ?
0055 3019             JR    NC,STO
0057 FE20             CP    20H          ; KEY IS CTRL KEY
0059 2015             JR    NZ,STO
005B          CMYO:   ENT
005B D3E1             OUT   (E1H),A      ; D000H-FFFFH IS DRAM
005D 11F0FF           LD    DE,FFF0H     ; TRANS. ADR.
0060 216B00           LD    HL,$MCP      ; MEMORY CHANG PROGRAM
0063 010500           LD    BC,05        ; BYTE SIZE
0066 EDB0             LDIR
0068 C3F0FF           JP    FFF0H        ; JUMP $FFF0
006B                  ;
006B          $MCP:   ENT                ; 0000H-0FFFH IS DRAM
006B D3E0             DEFW  E0D3H        ; OUT (E0H),A
006D C300             DEFW  00C3H        ; JP 0000H
006F 00               DEFB  00H
0070                  ;
0070          STO:    ENT
0070 06FF             LD    B,FFH        ; BUFFER CLEAR
0072 21F110           LD    HL,NAME      ; 10F1H-11F0H CLEAR
0075 CDD80F           CALL  ?CLER
0078 3E16             LD    A,16H        ; LASTER CLR.
007A CD1200           CALL  PRNT
007D 3E71             LD    A,71H        ; BACK:BLUE CHA.:WRITE
007F 2100D8           LD    HL,D800H     ; COLOR ADDRESS
0082 CDD509           CALL  #CLR8
0085 218D03           LD    HL,TIMIN     ; INTERRUPT JUMP ROUTINE
0088 3EC3             LD    A,C3H        ;
008A 323810           LD    (1038H),A
008D 223910           LD    (1039H),HL
0090 3E04             LD    A,04         ; NORMAL TEMPO
0092 329E11           LD    (TEMPW),A
0095 CDBE02           CALL  MLDSP        ; MELODY STOP
0098 CD0900           CALL  NL
009B 11E706           LD    DE,MSG?3     ; ** MONITOR 1Z-013A **
009E DF               RST   3            ; CALL MGX
009F CD7705           CALL  ?BEL
00A2          SS:     ENT
00A2 3E01             LD    A,01H
00A4 329D11           LD    (SWRK),A     ; KEY IN SILENT
00A7 2100E8           LD    HL,E800H     ; USR ROM ?
00AA 77               LD    (HL),A       ; ROM CHECK
00AB 1855             JR    FD2
00AD          ST1:    ENT
00AD CD0900           CALL  NL
00B0 3E2A             LD    A,2AH        ; ´*´ PRINT
00B2 CD1200           CALL  PRNT
00B5 11A311           LD    DE,BUFER     ; GET LINE WORK (11A3H)
00B8 CD0300           CALL  GETL
00BB 1A       ST2:    LD    A,(DE)
00BC 13               INC   DE
00BD FE0D             CP    0DH
00BF 28EC             JR    Z,ST1
00C1 FE4A             CP    ´J´          ; JUMP
00C3 282E             JR    Z,GOTO
```



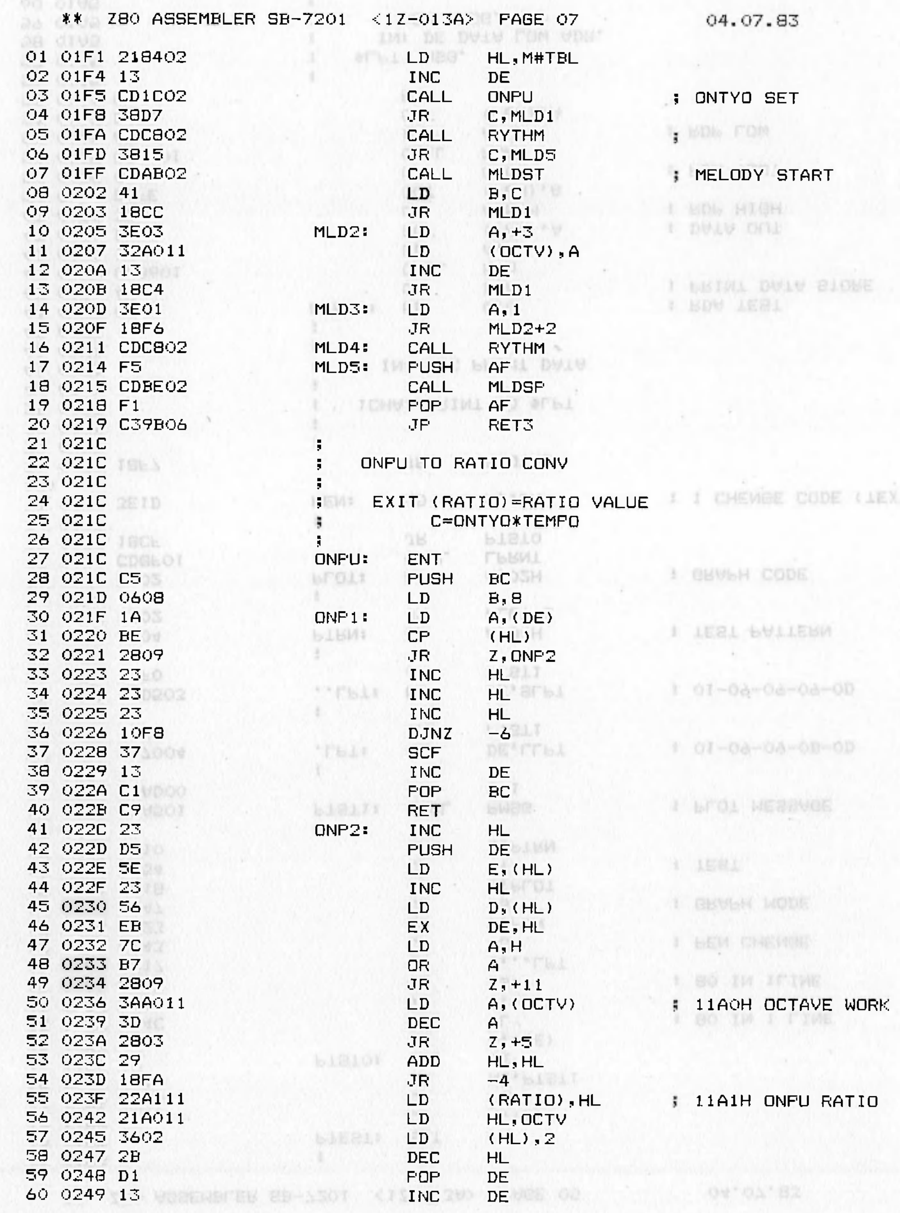
```
   **  Z80 ASSEMBLER SB-7201  <1Z-013A>  PAGE 03          04.07.83

0OC5 FE4C                     CP     'L'            ; LOAD PROGRAM
00C7 2848                     JR     Z,LOAD
00C9 FE46                     CP     'F'            ; FLOPPY ACCESS
00CB 2832                     JR     Z,FD
00CD FE42                     CP     'B'            ; KEY IN BELL
00CF 2826                     JR     Z,SG
00D1 FE23                     CP     '#'            ; CHANG MEMORY
00D3 2886                     JR     Z,CMYO
00D5 FE50                     CP     'P'            ; PRINTER TEST
00D7 287C                     JR     Z,PTEST
00D9 FE4D                     CP     'M'            ; MEMORY CORRECTION
00DB CAA807                   JP     Z,MCOR
00DE FE53                     CP     'S'            ; SAVED DATA
00E0 CA5E0F                   JP     Z,SAVE
00E3 FE56                     CP     'V'            ; VERIFYING DATA
00E5 CACB0F                   JP     Z,VRFY
00E8 FE44                     CP     'D'            ; DUMP DATA
00EA CA290D                   JP     Z,DUMP
00ED                 ;
00ED                 ;
00ED                          DEFS   +4
00F1                 ;
00F1 18C8                     JR     ST2            ; NOT COMMAND
00F3                 ;
00F3                 ;    JUMP COMMAND
00F3                 ;
00F3 CD3D01          GOTO:    CALL   HEXIY
00F6 E9                       JP     (HL)
00F7                 ;
00F7                 ;    KEY SOUND ON OFF
00F7                 ;
00F7 3A9D11          SG:      LD     A,(SWRK)       ; D0 = SOUND WORK
00FA 1F                       RRA
00FB 3F                       CCF                   ; CHENGE MODE
00FC 17                       RLA
00FD 18A5                     JR     SS+2
00FF                 ;
00FF                 ;    FLOPPY
00FF                 ;
00FF 2100F0          FD:      LD     HL,F000H       ; FLOPPY I/O CHECK
0102 7E              FD2:     LD     A,(HL)
0103 B7                       OR     A
0104 20A7                     JR     NZ,ST1
0106 E9              FD1:     JP     (HL)
0107                 ;
0107                 ;
0107                 ;    ERROR (LOADING)
0107                 ;
0107                 ?ER:     ENT                   ;
0107 FE02                     CP     02H            ; A=02H : BREAK IN
0109 28A2                     JR     Z,ST1          ;
010B 114701                   LD     DE.MSGE1       ; CHECK SUM ERROR
010E DF                       RST    3              ; CALL MSGX
010F 189C                     JR     ST1
0111                 ;
0111                 ;
0111                 ;    LOAD COMMAND
0111                 ;
0111 CDD804          LOAD:    CALL   ?RDI
0114 38F1                     JR     C,?ER
```

```
0116 CD0900           LOAD:   CALL   NL
0119 11A009                   LD     DE,MSG?2         ; LOADING
011C DF                       RST    3                ; CALL MSGX
011D 11F110                   LD     DE,NAME          ; FILE NAME
0120 DF                       RST    3                ; CALL MSGX
0121 CDF804                   CALL   ?RDD
0124 38E1                     JR     C,?ER
0126 2A0611                   LD     HL,(EXADR)       ; EXECUTE ADDRESS
0129 7C                       LD     A,H
012A FE12                     CP     12H              ; EXECUTE CHECK
012C 38E1                     JR     C,LOAD-2
012E E9                       JP     (HL)
012F                  ;
012F                  ;
012F                  ;
012F                  ;     GETLINE AND BREAK IN CHECK
012F                  ;
012F                  ;    EXIT BREAK IN THEN JUMP (ST1)
012F                  ;        ACC=TOP OF LINE DATA
012F                  ;
012F                  BGETL:  ENT
012F E3                       EX     (SP),HL
0130 C1                       POP    BC               ; STACK LOAD
0131 11A311                   LD     DE,BUFER         ; MONITOR GETLINE BUFF
0134 CD0300                   CALL   GETL
0137 1A                       LD     A,(DE)
0138 FE1B                     CP     1BH              ; BREAK CODE
013A 28D3                     JR     Z,LOAD-2         ; JP Z,ST1
013C E9                       JP     (HL)
013D                  ;
013D                  ;   ASCII TO HEX CONVERT
013D                  ;     INPUT (DE)=ASCII
013D                  ;     CY=1 THEN JUMP (ST1)
013D                  ;
013D                  HEXIY:  ENT
013D FDE3                     EX     (SP),IY
013F F1                       POP    AF
0140 CD1004                   CALL   HLHEX
0143 38CA                     JR     C,LOAD-2         ; JP C,ST1
0145 FDE9                     JP     (IY)
0147                  ;
0147                  ;
0147                  ;
0147                  MSGE1:  ENT
0147 43484543                 DEFM   'CHECK SUM ER.'
014B 4B205355
014F 4D204552
0153 2E
0154 0D                       DEFB   0DH
0155                  ;
0155                  ;
0155                  ;   PLOTTER PRINTER TEST COMMAND
0155                  ;             (DPG23)
0155                  ;       &=CONTROL COMMANDS GROUP
0155                  ;           C=PEN CHENGE
0155                  ;           G=GRAPH MODE
0155                  ;           S=80 CHA. IN 1 LINE
0155                  ;           L=40 CHA. IN 1 LINE
0155                  ;           T=PLOTTER TEST
0155                  ;    IN  (DE)=PRINT DATA
```

```
    ** Z80 ASSEMBLER SB-7201  <1Z-013A>  PAGE 05            04.07.83
01 0155                        ;
02 0155                        PTEST:   ENT
03 0155 1A                              LD     A,(DE)
04 0156 FE26                            CP     '&'
05 0158 2016                            JR     NZ,PTST1
06 015A 13                     PTSTO:   INC    DE
07 015B 1A                              LD     A,(DE)
08 015C FE4C                            CP     'L'               ; 80 IN 1 LINE
09 015E 2816                            JR     Z,.LPT
10 0160 FE53                            CP     'S'               ; 80 IN 1LINE
11 0162 2817                            JR     Z,..LPT
12 0164 FE43                            CP     'C'               ; PEN CHENGE
13 0166 2823                            JR     Z,PEN
14 0168 FE47                            CP     'G'               ; GRAPH MODE
15 016A 2818                            JR     Z,PLOT
16 016C FE54                            CP     'T'               ; TEST
17 016E 2810                            JR     Z,PTRN
18 0170                        ;
19 0170 CDA501                 PTST1:   CALL   PMSG              ; PLOT MESSAGE
20 0173 C3AD00                          JP     ST1
21 0176                        ;
22 0176 117004                 .LPT:    LD     DE,LLPT           ; 01-09-09-0B-0D
23 0179 18F5                            JR     PTST1
24 017B                        ;
25 017B 11D503                 ..LPT:   LD     DE,SLPT           ; 01-09-09-09-0D
26 017E 18F0                            JR     PTST1
27 01S0                        ;
28 0180 3E04                   PTRN:    LD     A,04H             ; TEST PATTERN
29 0182 1802                            JR     PLOT+2
30 0184                        ;
31 0184 3E02                   PLOT:    LD     A,02H             ; GRAPH CODE
32 0186 CD8F01                          CALL   LPRNT
33 0189 18CF                            JR     PT8TO
34 018B                        ;
35 018B 3E1D                   PEN:     LD     A,1DH             ; 1 CHENGE CODE (TEXT MO
DE)
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               ; RDA TEST
45 0191 47                              LD     B,A               ; PRINT DATA STORE
46 0192 CDB601                          CALL   RDA
47 0195 78                              LD     A,B
48 0196 D3FF                            OUT    (FFH),A           ; DATA OUT
49 0198 3E80                            LD     A,80H             ; RDP HIGH
50 019A D3FE                            OUT    (FEH),A
51 019C 0E01                            LD     C,01H             ; RDA TEST
52 019E CDB601                          CALL   RDA
53 01A1 AF                              XOR    A                 ; RDP LOW
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                        ;
```

```
01A5 D5               PMSG:  PUSH   DE
01A6 C5                      PUSH   BC
01A7 F5                      PUSH   AF
01A8 1A               PMSG1: LD     A,(DE)          ; ACC=DATA
01A9 CDBF01                  CALL   LPRNT
01AC 1A                      LD     A,(DE)
01AD 13                      INC    DE
01AE FE0D                    CP     0DH             ; END ?
01B0 20F6                    JR     NZ,PMSG1
01B2 F1                      POP    AF
01B3 C1                      POP    BC
01B4 D1                      POP    DE
01B5 C9                      RET
01B6                  ;
01B6                  ;   RDA CHECK
01B6                  ;
01B6                  ;      BRKEY IN TO MONITOR RETURN
01B6                  ;     IN: C RDA CODE
01B6                  ;
01B6 DBFE             RDA:   IN     A,(FE)
01B8 E60D                    AND    0DH
01BA B9                      CP     C
01BB C8                      RET    Z
01BC CD1E00                  CALL   BRKEY
01BF 20F5                    JR     NZ,RDA
01C1 31F010                  LD     SP,SP
01C4 C3AD00                  JP     ST1
01C7                  ;
01C7                  ;
01C7                  ;ORG 01C7H
01C7                  ;
01C7                  ;   MELODY
01C7                  ;
01C7                  ;    DE=DATA LOW ADR.
01C7                  ;   EXIT  CF=1 BREAK
01C7                  ;         CF=0 OK
01C7                  ;
01C7                  ?MLDY: ENT
01C7 C5                      PUSH   BC
01CS D5                      PUSH   DE
01C9 E5                      PUSH   HL
01CA 3E02                    LD     A,02H
01CC 32A011                  LD     (OCTV),A
01CF 0601                    LD     B,01
01D1 1A               MLD1:  LD     A,(DE)
01D2 FE0D                    CP     0DH             ; CR
01D4 283B                    JR     Z,MLD4
01D6 FEC8                    CP     C8H             ; END MARK
01D8 2837                    JR     Z,MLD4
01DA FECF                    CP     CFH             ; UNDER OCTAVE
01DC 2827                    JR     Z,MLD2
01DE FE2D                    CP     2DH             ; ´-´
01E0 2823                    JR     Z,MLD2
01E2 FE2B                    CP     2BH             ; ´+´
01E4 2827                    JR     Z,MLD3
01E6 FED7                    CP     D7H             ; UPPER OCTAVE
01E8 2823                    JR     Z,MLD3
01EA FE23                    CP     23H             ; "#" HANON
01EC 216C02                  LD     HL,MTBL
01EF 2004                    JR     NZ,+6
```

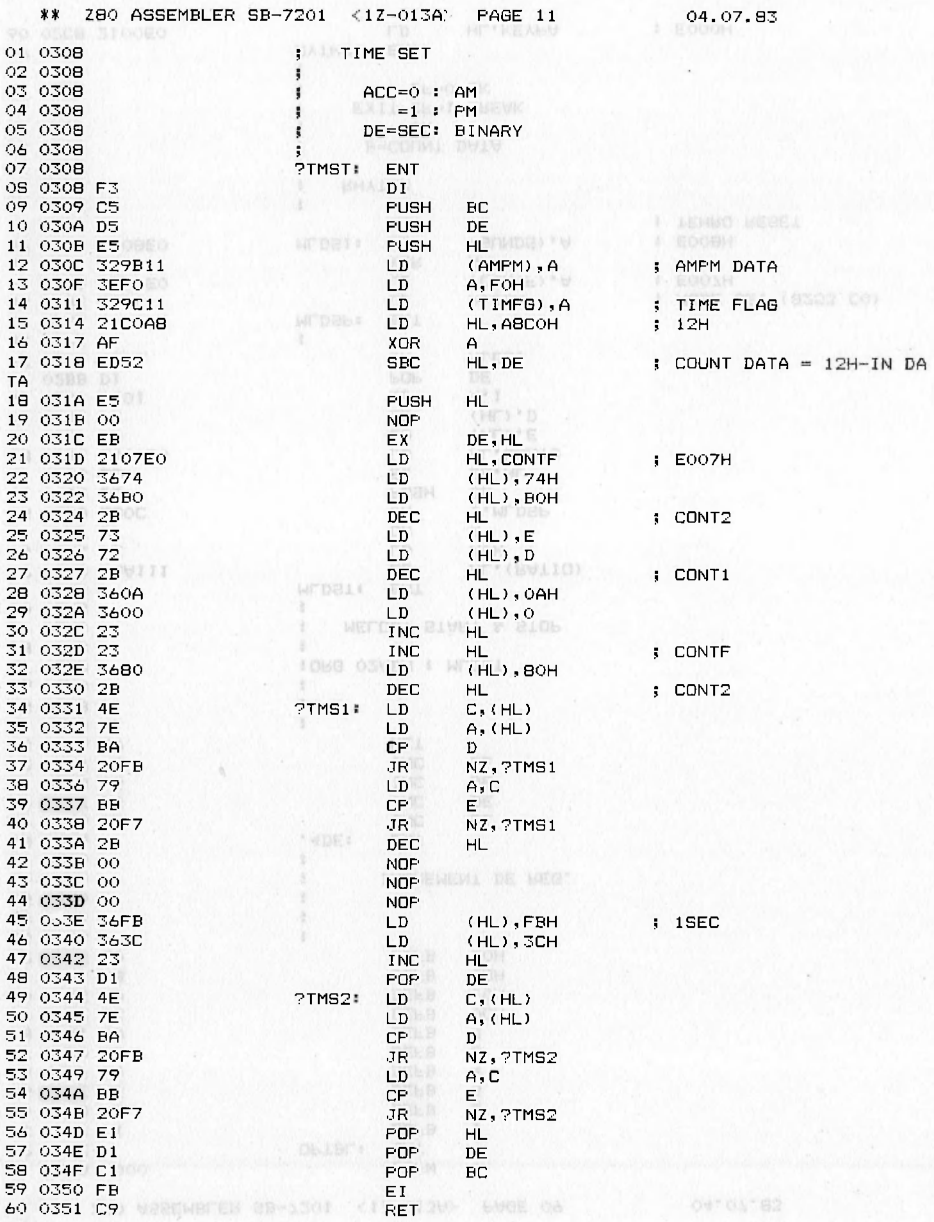


```
** Z80 ASSEMBLER SB-7201  <1Z-013A> PAGE 07          04.07.83
01F1 218402                LD    HL,M#TBL
01F4 13                    INC   DE
01F5 CD1C02                CALL  ONPU            ; ONTYO SET
01F8 38D7                  JR    C,MLD1
01FA CDC802                CALL  RYTHM           ;
01FD 3815                  JR    C,MLD5
01FF CDAB02                CALL  MLDST           ; MELODY START
0202 41                    LD    B,C
0203 18CC                  JR    MLD1
0205 3E03          MLD2:   LD    A,+3
0207 32A011                LD    (OCTV),A
020A 13                    INC   DE
020B 18C4                  JR    MLD1
020D 3E01          MLD3:   LD    A,1
020F 18F6                  JR    MLD2+2
0211 CDC802        MLD4:   CALL  RYTHM
0214 F5            MLD5:   PUSH  AF
0215 CDBE02                CALL  MLDSP
0218 F1                    POP   AF
0219 C39B06                JP    RET3
021C               ;
021C               ;   ONPU TO RATIO CONV
021C               ;
021C               ;    EXIT (RATIO)=RATIO VALUE
021C               ;         C=ONTYO*TEMPO
021C               ;
021C               ONPU:   ENT
021C C5                    PUSH  BC
021D 0608                  LD    B,8
021F 1A            ONP1:   LD    A,(DE)
0220 BE                    CP    (HL)
0221 2809                  JR    Z,ONP2
0223 23                    INC   HL
0224 23                    INC   HL
0225 23                    INC   HL
0226 10F8                  DJNZ  -6
0228 37                    SCF
0229 13                    INC   DE
022A C1                    POP   BC
022B C9                    RET
022C 23            ONP2:   INC   HL
022D D5                    PUSH  DE
022E 5E                    LD    E,(HL)
022F 23                    INC   HL
0230 56                    LD    D,(HL)
0231 EB                    EX    DE,HL
0232 7C                    LD    A,H
0233 B7                    OR    A
0234 2809                  JR    Z,+11
0236 3AA011                LD    A,(OCTV)        ; 11A0H OCTAVE WORK
0239 3D                    DEC   A
023A 2803                  JR    Z,+5
023C 29                    ADD   HL,HL
023D 18FA                  JR    -4
023F 22A111                LD    (RATIO),HL      ; 11A1H ONPU RATIO
0242 21A011                LD    HL,OCTV
0245 3602                  LD    (HL),2
0247 2B                    DEC   HL
0248 D1                    POP   DE
0249 13                    INC   DE
```

```
024A 1A                   LD      A,(DE)
024B 47                   LD      B,A
024C E6F0                 AND     F0H            ; ONTYO ?
024E FE30                 CP      30H
0250 2803                 JR      Z,+5
0252 7E                   LD      A,(HL)         ; HL=ONTYO
0253 1805                 JR      +7
0255 13                   INC     DE
0256 78                   LD      A,B
0257 E60F                 AND     0FH
0259 77                   LD      (HL),A         ; HL=ONTYO
025A 219C02               LD      HL,OPTBL
025D 85                   ADD     A,L
025E 6F                   LD      L,A
025F 4E                   LD      C,(HL)
0260 3A9E11               LD      A,(TEMPW)
0263 47                   LD      B,A
0264 AF                   XOR     A
0265 81          ONP3:    ADD     A,C
0266 10FD                 DJNZ    -1
0268 C1                   POP     BC
0269 4F                   LD      C,A
026A AF                   XOR     A
026B C9                   RET
026C             ;
026C             ;
026C             MTBL:    ENT
026C 43                   DEFB    43H            ; C
026D 4608                 DEFW    0846H
026F 44                   DEFB    44H            ; D
0270 5F07                 DEFW    075FH
0272 45                   DEFB    45H            ; E
0273 9106                 DEFW    0691H
0275 46                   DEFB    46H            ; F
0276 3306                 DEFW    0633H
0278 47                   DEFB    47H            ; G
0279 8605                 DEFW    0586H
027B 41                   DEFB    41H            ; A
027C EC04                 DEFW    04ECH
027E 42                   DEFB    42H            ; B
027F 6404                 DEFW    0464H
0281 52                   DEFB    52H            ; R
0282 0000                 DEFW    0
0284             M#TBL:   ENT
0284 43                   DEFB    43H            ; #C
0285 CF07                 DEFW    07CFH
0287 44                   DEFB    44H            ; #D
0288 F506                 DEFW    06F5H
028A 45                   DEFB    45H            ; #E
028B 3306                 DEFW    0633H
028D 46                   DEFB    46H            ; #F
028E DA05                 DEFW    05DAH
0290 47                   DEFB    47H            ; #G
0291 3705                 DEFW    0537H
0293 41                   DEFB    41H            ; #A
0294 A504                 DEFW    04A5H
0296 42                   DEFB    42H            ; #B
0297 2304                 DEFW    0423H
0299 52                   DEFB    52H            ; #R
029A
```

```
   **  Z80 ASSEMBLER SB-7201  <1Z-013A>  PAGE 09           04.07.83
029A 0000                 DEFW   0
029C            OPTBL:    ENT
029C 01                   DEFB   1
029D 02                   DEFB   2
029E 03                   DEFB   3
029F 04                   DEFB   4
02A0 06                   DEFB   6
02A1 08                   DEFB   8
02A2 0C                   DEFB   0CH
02A3 10                   DEFB   10H
02A4 18                   DEFB   18H
02A5 20                   DEFB   20H
02A6            ;
02A6            ;
02A6            ;        INCREMENT DE REG.
02A6            ;
02A6            .4DE:     ENT
02A6 13                   INC    DE
02A7 13                   INC    DE
02A8 13                   INC    DE
02A9 13                   INC    DE
02AA C9                   RET
02AB            ;
02AB            ;
02AB            ;
02AB            ;ORG 02ABH ; MLDST
02AB            ;
02AB            ;     MELODY START & STOP
02AB            ;
02AB            MLDST:    ENT
02AB 2AA111               LD     HL,(RATIO)
02AE 7C                   LD     A,H
02AF B7                   OR     A
02B0 280C                 JR     Z,MLDSP
02B2 D5                   PUSH   DE
02B3 EB                   EX     DE,HL
02B4 2104E0               LD     HL,CONTO
02B7 73                   LD     (HL),E
02B8 72                   LD     (HL),D
02B9 3E01                 LD     A,1
02BB D1                   POP    DE
02BC 1806                 JR     MLDS1
02BE            ;
02BE            MLDSP:    ENT
02BE 3E36                 LD     A,36H           ; MODE SET (8253 C0)
02C0 3207E0               LD     (CONTF),A       ; E007H
02C3 AF                   XOR    A
02C4 3208E0     MLDS1:    LD     (SUNDG),A       ; E008H
02C7 C9                   RET                    ; TEHRO RESET
02C8            ;
02C8            ;     RHYTHM
02C8            ;
02C8            ;       B=COUNT DATA
02C8            ;            IN
02C8            ;      EXIT CF=1 BREAK
02C8            ;           CF=0 OK
02C8            ;
02C8            RYTHM:    ENT
02C8 2100E0               LD     HL,KEYPA        ; E000H
```

```
02CB 36F8                LD     (HL),F8H
02CD 23                  INC    HL
02CE 7E                  LD     A,(HL)
02CF E681                AND    81H               ; BREAK IN CHECK
02D1 2002                JR     NZ,+4
02D3 37                  SCF
02D4 C9                  RET
02D5 3A08E0              LD     A,(TEMP)          ; E008H
02D8 0F                  RRCA                     ; TEMPO OUT
02D9 38FA                JR     C,-4
02DB 3A08E0              LD     A,(TEMP)
02DE 0F                  RRCA
02DF 30FA                JR     NC,-4
02E1 10F2                DJNZ   -12
02E3 AF                  XOR    A
02E4 C9                  RET
02E5                ;
02E5                ;
02E5                ;   TEMPO SET
02E5                ;
02E5                ;     ACC=VALUE (1-7)
02E5                ;
02E5                ?TEMP:   ENT
02E5 F5                  PUSH   AF
02E6 C5                  PUSH   BC
02E7 E60F                AND    0FH
02E9 47                  LD     B,A
02EA 3E08                LD     A,8
02EC 90                  SUB    B
02ED 329E11              LD     (TEMPW),A
02F0 C1                  POP    BC
02F1 F1                  POP    AF
02F2 C9                  RET
02F3                ;
02F3                ;          CRT MANAGMENT
02F3                ;
02F3                ;     EXIT  HL:DSPXY H=Y,L=X
02F3                ;           DE:MANG ADR. (ON DSPXY)
02F3                ;           A :MANG DATA
02F3                ;           CY:MANG=1
02F3                ;
02F3                .MANG:   ENT
02F3 217311              LD     HL,MANG           ; CRT MANG. POINTER
02F6 3A7211              LD     A,(1172H)         ; DSPXY+1
02F9 85                  ADD    A,L
02FA 6F                  LD     L,A
02FB 7E                  LD     A,(HL)
02FC 23                  INC    HL
02FD CB16                RL     (HL)
02FF B6                  OR     (HL)
0300 CB1E                RR     (HL)
0302 0F                  RRCA
0303 EB                  EX     DE,HL
0304 2A7111              LD     HL,(DSPXY)
0307 C9                  RET
0308                ;
0308                ;
0308                ;
0308                ;ORG 030SH
0308                ;
```

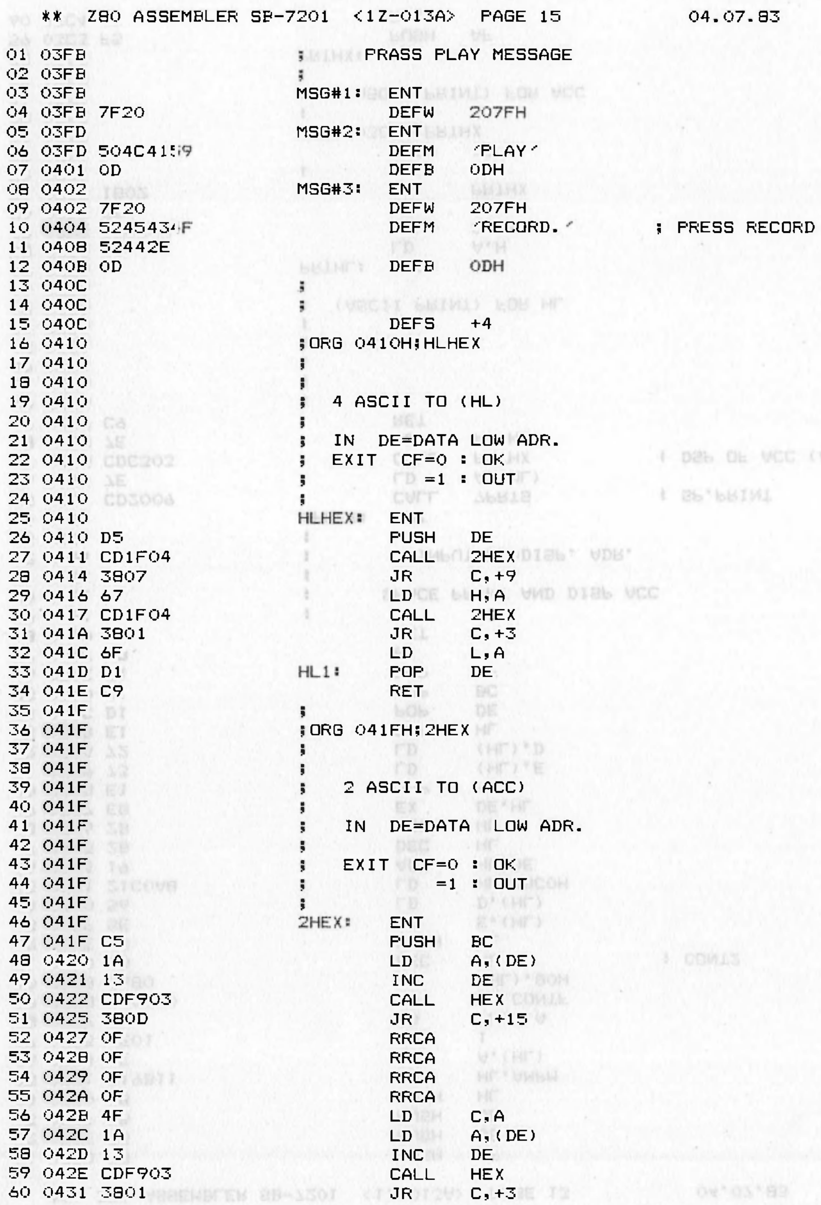


```
 **  Z80 ASSEMBLER SB-7201  <1Z-013A>   PAGE 11            04.07.83
0308                 ; TIME SET
0308                 ;
0308                 ;     ACC=0 : AM
0308                 ;        =1 : PM
0308                 ;     DE=SEC: BINARY
0308                 ;
0308                 ?TMST:  ENT
0308 F3                      DI
0309 C5                      PUSH   BC
030A D5                      PUSH   DE
030B E5                      PUSH   HL
030C 329B11                  LD     (AMPM),A         ; AMPM DATA
030F 3EF0                    LD     A,F0H
0311 329C11                  LD     (TIMFG),A        ; TIME FLAG
0314 21C0A8                  LD     HL,A8C0H         ; 12H
0317 AF                      XOR    A
0318 ED52                    SBC    HL,DE            ; COUNT DATA = 12H-IN DA
TA
031A E5                      PUSH   HL
031B 00                      NOP
031C EB                      EX     DE,HL
031D 2107E0                  LD     HL,CONTF         ; E007H
0320 3674                    LD     (HL),74H
0322 36B0                    LD     (HL),B0H
0324 2B                      DEC    HL               ; CONT2
0325 73                      LD     (HL),E
0326 72                      LD     (HL),D
0327 2B                      DEC    HL               ; CONT1
0328 360A                    LD     (HL),0AH
032A 3600                    LD     (HL),0
032C 23                      INC    HL
032D 23                      INC    HL               ; CONTF
032E 3680                    LD     (HL),80H
0330 2B                      DEC    HL               ; CONT2
0331 4E              ?TMS1:  LD     C,(HL)
0332 7E                      LD     A,(HL)
0333 BA                      CP     D
0334 20FB                    JR     NZ,?TMS1
0336 79                      LD     A,C
0337 BB                      CP     E
0338 20F7                    JR     NZ,?TMS1
033A 2B                      DEC    HL
033B 00                      NOP
033C 00                      NOP
033D 00                      NOP
033E 36FB                    LD     (HL),FBH         ; 1SEC
0340 363C                    LD     (HL),3CH
0342 23                      INC    HL
0343 D1                      POP    DE
0344 4E              ?TMS2:  LD     C,(HL)
0345 7E                      LD     A,(HL)
0346 BA                      CP     D
0347 20FB                    JR     NZ,?TMS2
0349 79                      LD     A,C
034A BB                      CP     E
034B 20F7                    JR     NZ,?TMS2
034D E1                      POP    HL
034E D1                      POP    DE
034F C1                      POP    BC
0350 FB                      EI
0351 C9                      RET
```

```
0352                  ;
0352                  ;
0352                  ;   BELL DATA
0352                  ;
0352                  ?BELD:  ENT
0352 D7                       DEFB   D7H
0353 4130                     DEFM   'A0'
0355 0D                       DEFB   0DH
0356                  ;
0356                  ;
0356                  ;
0356                          DEFS   +2
0358                  ;ORG 0358H
0358                  ;
0358                  ;   TIME READ
0358                  ;
0358                  ;     EXIT  ACC=0 :AM
0358                  ;              =1 :PM
0358                  ;           DE=SEC. BINARY
0358                  ;
0358                  ?TMRD:  ENT
0358 E5                       PUSH   HL
0359 2107E0                   LD     HL,CONTF
035C 3680                     LD     (HL),80H
035E 2B                       DEC    HL              ; CONT2
035F F3                       DI
0360 5E                       LD     E,(HL)
0361 56                       LD     D,(HL)
0362 FB                       EI
0363 7B                       LD     A,E
0364 B2                       OR     D
0365 280E                     JR     Z,?TMR1
0367 AF                       XOR    A
0368 21C0A8                   LD     HL,A8C0H
036B ED52                     SBC    HL,DE
036D 3810                     JR     C,?TMR2
036F EB                       EX     DE,HL
0370 3A9B11                   LD     A,(AMPM)
0373 E1                       POP    HL
0374 C9                       RET
0375 11C0A8           ?TMR1:  LD     DE,A8C0H        ; 12H
0378 3A9B11                   LD     A,(AMPM)
037B EE01                     XOR    1
037D E1                       POP    HL
037E C9                       RET
037F F3               ?TMR2:  DI
0380 2106E0                   LD     HL,CONT2
0383 7E                       LD     A,(HL)
0384 2F                       CPL
0385 5F                       LD     E,A
0386 7E                       LD     A,(HL)
0387 2F                       CPL
0388 57                       LD     D,A
0389 FB                       EI
038A 13                       INC    DE
038B 18EB                     JR     ?TMR1+3
038D                  ;
038D                  ;   TIME INTERRUPT
038D                  ;
038D                  TIMIN:  ENT
```

```
      **  Z80  ASSEMBLER SB-7201  <1Z-013A>  PAGE 13            04.07.83
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),80H
11 039D 2B                         DEC     HL                ; CONT2
12 039E E5                         PUSH    HL
13 039F 5E                         LD      E,(HL)
14 03A0 56                         LD      D,(HL)
15 03A1 21C0A8                     LD      HL,A8C0H
16 03A4 19                         ADD     HL,DE
17 03A5 2B                         DEC     HL
1S 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
29 03B1                 ;
30 03B1                 ;       SPACE PRINT AND DISP ACC
31 03B1                 ;
32 03B1                 ;           INPUT:HL=DISP. ADR.
33 03B1                 ;
34 03B1                 SPHEX:     ENT
35 03B1 CD2009                     CALL    ?PRTS             ; SP.PRINT
36 03B4 7E                         LD      A,(HL)
37 03B5 CDC303                     CALL    PRTHX             ; DSP OF ACC (ASCII)
38 03B8 7E                         LD      A,(HL)
39 03B9 C9                         RET
40 03BA                 ;
41 03BA                 ;
42 03BA                 ;
43 03BA                 ;ORB 03BAH
44 03BA                 ;
45 03BA                 ;   (ASCII PRINT) FOR HL
46 03BA                 ;
47 03BA                 PRTHL:     ENT
48 03BA 7C                         LD      A,H
49 03BB CDC303                     CALL    PRTHX
50 03BE 7D                         LD      A,L
51 03BF 1802                       JR      PRTHX
52 03C1                 ;
53 03C1                            DEFS    +2
54 03C3                 ;ORG 03C3H;PRTHX
55 03C3                 ;
56 03C3                 ;   (ASCII PRINT) FOR ACC
57 03C3                 ;
58 03C3                 PRTHX:     ENT
59 03C3 F5                         PUSH    AF
60 03C4 0F                         RRCA
```

```
03C5 0F                  RRCA
03C6 0F                  RRCA
03C7 0F                  RRCA
03C8 CDDA03              CALL   ASC
03CB CD1200              CALL   PRNT
03CE F1                  POP    AF
03CF CDDA03              CALL   ASC
03D2 C31200              JP     PRNT
03D5             ;
03D5             ;
03D5             ;
03D5             ;
03D5             ;    80 CHA. 1 LINE CODE (DATA)
03D5             ;
03D5             SLPT:   ENT
03D5 01                  DEFB   01H             ; TEXT MODE
03D6 09                  DEFB   09H
03D7 09                  DEFB   09H
03D8 09                  DEFB   09H
03D9 0D                  DEFB   0DH
03DA             ;
03DA             ;ORG 03DAH;ASC
03DA             ;
03DA             ;  HEXADECIMAL TO ASCII
03DA             ;    IN : ACC (D3-D0)=HEXADECIMAL
03DA             ;   EXIT: ACC = ASCII
03DA             ;
03DA             ASC:    ENT
03DA E60F                AND    0FH
03DC FE0A                CP     0AH
03DE 3802                JR     C,NOADD
03E0 C607                ADD    A,7
03E2             NOADD:  ENT
03E2 C630                ADD    A,30H
03E4 C9                  RET
03E5             ;
03E5             ;  ASCII TO HEXADECIMAL
03E5             ;  IN   : ACC = ASCII
03E5             ; EXIT  : ACC = HEXADECIMAL
03E5             ;         CY  = 1 ERROR
03E5             ;
03E5             HEXJ:   ENT
03E5 D630                SUB    30H
03E7 D8                  RET    C
03E8 FE0A                CP     0AH
03EA 3F                  CCF
03EB D0                  RET    NC
03EC D607                SUB    7
03EE FE10                CP     10H
03F0 3F                  CCF
03F1 D8                  RET    C
03F2 FE0A                CP     0AH
03F4 C9                  RET
03F5             ;
03F5             ;
03F5                     DEFS   +4
03F9             ;ORG 03F9H;HEX
03F9             HEX:    ENT
03F9 18EA                JR     HEXJ
03FB             ;
```



```
**  Z80 ASSEMBLER SP-7201  <1Z-013A>  PAGE 15            04.07.83
03FB                  ;PRASS PLAY MESSAGE
03FB                  ;
03FB                  MSG#1: ENT
03FB 7F20                    DEFW   207FH
03FD                  MSG#2: ENT
03FD 504C4159                DEFM   'PLAY'
0401 0D                      DEFB   0DH
0402                  MSG#3: ENT
0402 7F20                    DEFW   207FH
0404 5245434F                DEFM   'RECORD.'        ; PRESS RECORD
0408 52442E
040B 0D                      DEFB   0DH
040C                  ;
040C                  ;
040C                         DEFS   +4
0410                  ;ORG 0410H;HLHEX
0410                  ;
0410                  ;
0410                  ;  4 ASCII TO (HL)
0410                  ;
0410                  ;  IN  DE=DATA LOW ADR.
0410                  ;  EXIT  CF=0 : OK
0410                  ;         =1 : OUT
0410                  ;
0410                  HLHEX: ENT
0410 D5                      PUSH   DE
0411 CD1F04                  CALL   2HEX
0414 3807                    JR     C,+9
0416 67                      LD     H,A
0417 CD1F04                  CALL   2HEX
041A 3801                    JR     C,+3
041C 6F                      LD     L,A
041D D1               HL1:   POP    DE
041E C9                      RET
041F                  ;
041F                  ;ORG 041FH;2HEX
041F                  ;
041F                  ;
041F                  ;   2 ASCII TO (ACC)
041F                  ;
041F                  ;   IN  DE=DATA  LOW ADR.
041F                  ;
041F                  ;   EXIT  CF=0 : OK
041F                  ;            =1 : OUT
041F                  ;
041F                  2HEX:  ENT
041F C5                      PUSH   BC
0420 1A                      LD     A,(DE)
0421 13                      INC    DE
0422 CDF903                  CALL   HEX
0425 380D                    JR     C,+15
0427 0F                      RRCA
0428 0F                      RRCA
0429 0F                      RRCA
042A 0F                      RRCA
042B 4F                      LD     C,A
042C 1A                      LD     A,(DE)
042D 13                      INC    DE
042E CDF903                  CALL   HEX
0431 3801                    JR     C,+3
```

```
0433 B1                       OR     C
0434 C1              2HE1:    POP    BC
0435 C9                       RET
0436                 ,
0436                 ;
0436                 ;   WRITE INFORMATION
0436                 ;
0436                 ?WRI:    ENT
0436 F3                       DI
0437 D5                       PUSH   DE
0438 C5                       PUSH   BC
0439 E5                       PUSH   HL
043A 16D7                     LD     D,D7H            ; ´W´
043C 1ECC                     LD     E,CCH            ; ´L´
043E 21F010                   LD     HL,IBUFE         ; 10F0H
0441 018000                   LD     BC,80H           ; WRITE BYTE SIZE
0444 CD1A07          WRI1:    CALL   CKSUM            ; CHECK SUM
0447 CD9F06                   CALL   MOTOR            ; MOTOR ON
044A 3818                     JR     C,WRI3
044C 7B                       LD     A,E
044D FECC                     CP     CCH              ; ´L´
044F 200D                     JR     NZ,WRI2
0451 CD0900                   CALL   NL
0454 D5                       PUSH   DE
0455 116704                   LD     DE,MSG#7         ; WRITING
0458 DF                       RST    3                ; CALL MSGX
0459 11F110                   LD     DE,NAME          ; FILE NAME
045C DF                       RST    3                ; CALL MSGX
045D D1                       POP    DE
045E CD7A07          WRI2:    CALL   GAP
0461 CD8A04                   CALL   WTAPE
0464 C35405          WRI3:    JP     RET2
0467                 ;
0467                 MSG#7:   ENT
0467 57524954                 DEFM   ´WRITING ´
046B 494E4720
046F 0D                       DEFB   0DH
0470                 ;
0470                 ;
0470                 ;
0470                 ;    40 CHA. IN 1 LINE CODE (DATA)
0470                 ;
0470                 LLPT:    ENT
0470 01                       DEFB   01H              ; TEXT MODE
0471 09                       DEFB   09H
0472 09                       DEFB   09H
0473 0B                       DEFB   0BH
0474 0D                       DEFB   0DH
0475                 ;
0475                 ;ORG 0475H
0475                 ;
0475                 ;   WRITE DATA
0475                 ;
0475                 ;    EXIT CF=0 : OK
0475                 ;           =1 : BREAK
0475                 ;
0475                 ?WRD:    ENT
0475 F3                       DI
0476 D5                       PUSH   DE
```

```
0477 C5                  PUSH  BC
0478 E5                  PUSH  HL
0479 16D7                LD    D,D7H           ; 'W'
047B 1E53                LD    E,53H           ; 'S'
047D ED4B0211            LD    BC,(SIZE)       ; WRITE DATA BYTE SIZE
0481 2A0411              LD    HL,(DTADR)      ; WRITE DATA ADDRESS
0484 78                  LD    A,B
0485 B1                  OR    C
0486 284A                JR    Z,RET1
0488 18BA                JR    WRI1
048A           ;
048A           ;
048A           ;   TAPE WRITE
048A           ;
048A           ;     BC=BYTE SIZE
048A           ;     HL=DATA LOW ADR.
048A           ;
048A           ;    EXIT  CF=0 : OK
048A           ;             =1 : BREAK
048A           ;
048A D5        WTAPE:    PUSH  DE
048B C5                  PUSH  BC
048C E5                  PUSH  HL
048D 1602                LD    D,2
048F 3EF8                LD    A,F8H
0491 3200E0              LD    (KEYPA),A       ; E000H
0494 7E        WTAP1:    LD    A,(HL)
0495 CD6707              CALL  WBYTE           ; 1 BYTE WRITE
0498 3A01E0              LD    A,(KEYPB)       ; E001H
049B E681                AND   81H             ; SHIFT & BREAK
049D C2A504              JP    NZ,WTAP2
04A0 3E02                LD    A,02H           ; BREAK IN CODE
04A2 37                  SCF
04A3 182D                JR    WTAP3
04A5 23        WTAP2:    INC   HL
04A6 0B                  DEC   BC
04A7 78                  LD    A,B
04A8 B1                  OR    C
04A9 C29404              JP    NZ,WTAP1
04AC 2A9711              LD    HL,(SUMDT)      ; SUM DATA SET
04AF 7C                  LD    A,H
04B0 CD6707              CALL  WBYTE
04B3 7D                  LD    A,L
04B4 CD6707              CALL  WBYTE
04B7 CD1A0A              CALL  LONG
04BA 15                  DEC   D
04BB C2C204              JP    NZ,+7
04BE B7                  OR    A
04BF C3D204              JP    WTAP3
04C2 0600                LD    B,0
04C4 CD010A              CALL  SHORT
04C7 05                  DEC   B
04C8 C2C404              JP    NZ,-4
04CB E1                  POP   HL
04CC C1                  POP   BC
04CD C5                  PUSH  BC
04CE E5                  PUSH  HL
04CF C39404              JP    WTAP1
04D2           WTAP3:
04D2 E1        RET1:     POP   HL
```

```
04D3 C1                       POP    BC
04D4 D1                       POP    DE
04D5 C9                       RET
04D6                  ;
04D6                  ;
04D6                  ;
04D6                  ;
04D6                  ;
04D8                          ORG    04D8H
04D8                  ;
04D8                  ;
04D8                  ;   READ INFORMATION (FROM $CMT)
04D8                  ;
04D8                  ;    EXIT ACC=0 : OK CF=0
04D8                  ;            =1 : ER CF=1
04D8                  ;            =2 : BREAK CF=1
04D8                  ;
04D8                  ?RDI:   ENT
04D8 F3                       DI
04D9 D5                       PUSH   DE
04DA C5                       PUSH   BC
04DB E5                       PUSH   HL
04DC 16D2                     LD     D,D2H            ;  'R'
04DE 1ECC                     LD     E,CCH            ;  'L'
04E0 018000                   LD     BC,80H
04E3 21F010                   LD     HL,IBUFE
04E6                  RD1:    ENT
04E6 CD9F06                   CALL   MOTOR
04E9 DA7205                   JP     C,RTP6
04EC CD5B06                   CALL   TMARK
04EF DA7205                   JP     C,RTP6
04F2 CD0E05                   CALL   RTAPE
04F5 C35405                   JP     RTP4
04F8                  ;
04F8                  ;
04F8                  ;
04F8                  ;ORG 04F8H
04F8                  ;
04F8                  ;
04F8                  ;   READ DATA (FROM $CMT)
04F8                  ;
04F8                  ;      EXIT SAME UP
04F8                  ;
04F8                  ?RDD:   ENT
04F8 F3                       DI
04F9 D5                       PUSH   DE
04FA C5                       PUSH   BC
04FB E5                       PUSH   HL
04FC 16D2                     LD     D,D2H            ;  'R'
04FE 1E53                     LD     E,53H            ;  'S'
0500 ED4B0211                 LD     BC,(SIZE)
0504 2A0411                   LD     HL,(DTADR)
0507 78                       LD     A,B
0508 B1                       OR     C
0509 CA5405                   JP     Z,RTP4
050C 18D8                     JR     RD1
050E                  ;
050E                  ;
050E                  ;   READ TAPE
050E                  ;
```

```
**  Z80  ASSEMBLER SB-7201  <1Z-013A>  PAGE 19              04.07.83
050E                    ;      IN  BC=SIZE
050E                    ;            DE=LOAD ADR.
050E                    ;
050E                    ;      EXIT  ACC=0 : OK CF=0
050E                    ;                =1 : ER    =1
050E                    ;                =2 : BREAK=1
050E                    ;
050E                    RTAPE:  ENT
050E D5                         PUSH    DE
050F C5                         PUSH    BC
0510 E5                         PUSH    HL
0511 2602                       LD      H,2                 ; TWICE WRITE
0513                    RTP1:   ENT
0513 0101E0                     LD      BC,KEYPB
0516 1102E0                     LD      DE,CSTR
0519                    RTP2:   ENT
0519 CD0106                     CALL    EDGE                ; 1→0 EDGE DETECT
051C 3854                       JR      C,RTP6
051E CD4A0A                     CALL    DLY3                ; CALL DLY2*3
0521 1A                         LD      A,(DE)              ; DATA (1BIT) READ
0522 E620                       AND     20H
0524 CA1905                     JP      Z,RTP2
0527 54                         LD      D,H
0528 210000                     LD      HL,0
052B 229711                     LD      (SUMDT),HL
052E E1                         POP     HL                  ;
052F C1                         POP     BC
0530 C5                         PUSH    BC
0531 E5                         PUSH    HL
0532                    RTP3:   ENT
0532 CD2406                     CALL    RBYTE               ; 1BYTE READ
0535 383B                       JR      C,RTP6
0537 77                         LD      (HL),A
0538 23                         INC     HL
0539 0B                         DEC     BC
053A 78                         LD      A,B
053B B1                         OR      C
053C 20F4                       JR      NZ,RTP3
053E 2A9711                     LD      HL,(SUMDT)          ; CHECK SUM
0541 CD2406                     CALL    RBYTE               ; CHECK SUM DATA
0544 382C                       JR      C,RTP6
0546 5F                         LD      E,A
0547 CD2406                     CALL    RBYTE               ; CHECK SUM DATA
054A 3826                       JR      C,RTP6
054C BD                         CP      L
054D 2016                       JR      NZ,RTP5
054F 7B                         LD      A,E
0550 BC                         CP      H
0551 2012                       JR      NZ,RTP5
0553                    RTP8:   ENT
0553 AF                         XOR     A
0554                    RTP4:   ENT
0554                    RET2:   ENT
0554 E1                         POP     HL
0555 C1                         POP     BC
0556 D1                         POP     DE
0557 CD0007                     CALL    MSTOP
055A F5                         PUSH    AF
055B 3A9C11                     LD      A,(TIMFG)           ; INT. CHECK
055E FEF0                       CP      F0H
```

```
0560 2001                     JR     NZ,+3
0562 FB                       EI
0563 F1                       POP    AF
0564 C9                       RET
0565                  ;
0565                  RTP5:   ENT
0565 15                       DEC    D
0566 2806                     JR     Z,RTP7
0568 62                       LD     H,D
0569 CDE20F                   CALL   GAPCK
056C 18A5                     JR     RTP1
056E                  RTP7:   ENT
056E 3E01                     LD     A,1
0570 1802                     JR     RTP9
0572                  RTP6:   ENT
0572 3E02                     LD     A,2
0574                  RTP9:   ENT
0574 37                       SCF
0575 18DD                     JR     RTP4
0577                  ;
0577                  ;
0577                  ;BELL
0577                  ;
0577                  ?BEL:   ENT
0577 D5                       PUSH   DE
0578 115203                   LD     DE,?BELD
057B F7                       RST    6                ; CALL MELDY
057C D1                       POP    DE
057D C9                       RET
057E                  ;
057E                  ;    FLASING AND KEYIN
057E                  ;       EXIT:ACC INPUT KEY DATA(DSP.CODE)
057E                  ;          H=FOH THEN NO KEYIN(Z FLG.)
057E                  ;
057E                  FLKEY:  ENT
057E CDFF09                   CALL   ?FLAS
0581 CDCA08                   CALL   ?KEY
0584 FEF0                     CP     FOH
0586 C9                       RET
0587                  ;
0587                  ;
0587                  ;
0587                  ;
0587                  ;
0587                          DEFS   +1
0588                  ;ORG 0588H
0588                  ;
0588                  ;
0588                  ;   VERIFY (FROM $CMT)
0588                  ;
0588                  ;    EXIT ACC =0 : OK CF=0
0588                  ;             =1 : ER CF=1
0588                  ;             =2 : BREAK CF=1
0588                  ;
0588                  ?VRFY:  ENT
0588 F3                       DI
0589 D5                       PUSH   DE
058A C5                       PUSH   BC
058B E5                       PUSH   HL
058C ED4B0211                 LD     BC,(SIZE)
```

```
**  Z80  ASSEMBLER SB-7201  <1Z-013A>  PAGE 21             04.07.83
0590 2A0411                LD      HL,(DTADR)
0593 16D2                  LD      D,D2H             ;  'R'
0595 1E53                  LD      E,53H             ;  'S'
0597 78                    LD      A,B
0598 B1                    OR      C
0599 28B9                  JR      Z,RTP4
059B CD1A07                CALL    CKSUM
059E CD9F06                CALL    MOTOR
05A1 38CF                  JR      C,RTP6
05A3 CD5B06                CALL    TMARK             ; TAPE MARK DETECT
05A6 38CA                  JR      C,RTP6
05A8 CDAD05                CALL    TVRFY
05AB 18A7                  JR      RTP4
05AD              ;
05AD              ;
05AD              ;     DATA VERIFY
05AD              ;
05AD              ;       BC=SIZE
05AD              ;       HL=DATA LOW ADR
05AD              ;       CSMDT=CHECK SUM
05AD              ;      EXIT ACC=0 : OK CF=0
05AD              ;              =1 : ER   =1
05AD              ;              =2 : BREAK-1
05AD              ;
05AD              ;
05AD              TVRFY:   ENT
05AD D5                    PUSH    DE
05AE C5                    PUSH    BC
05AF E5                    PUSH    HL
05B0 2602                  LD      H,2
05B2              TVF1:    ENT
05B2 0101E0                LD      BC,KEYPB
05B5 1102E0                LD      DE,CSTR
05B8              TVF2:    ENT
05B8 CD0106                CALL    EDGE
05BB DA7205                JP      C,RTP6
05BE CD4A0A                CALL    DLY3              ; CALL DLY2*3
05C1 1A                    LD      A,(DE)
05C2 E620                  AND     20H
05C4 CAB805                JP      Z,TVF2
05C7 54                    LD      D,H
05C8 E1                    POP     HL
05C9 C1                    POP     BC
05CA C5                    PUSH    BC
05CB E5                    PUSH    HL
05CC              TVF3:    ENT
05CC CD2406                CALL    RBYTE
05CF 38A1                  JR      C,RTP6
05D1 BE                    CP      (HL)
05D2 209A                  JR      NZ,RTP7
05D4 23                    INC     HL
05D5 0B                    DEC     BC
05D6 78                    LD      A,B
05D7 B1                    OR      C
05D8 20F2                  JR      NZ,TVF3
05DA 2A9911                LD      HL,(CSMDT)
05DD CD2406                CALL    RBYTE
05E0 BC                    CP      H
05E1 208B                  JR      NZ,RTP7
05E3 CD2406                CALL    RBYTE
```

```
05E6 BD                   CP     L
05E7 2085                 JR     NZ,RTP7
05E9 15                   DEC    D
05EA CA5305               JP     Z,RTP8
05ED 62                   LD     H,D
05EE 18C2                 JR     TVF1
05F0
05F0            ;    FLASHING DATA LOAD
05F0            ;
05F0            ?LOAD:    ENT
05F0 F5                   PUSH   AF
05F1 3A8E11               LD     A,(FLASH)
05F4 CDB10F               CALL   ?PONT
05F7 77                   LD     (HL),A
05F8 F1                   POP    AF
05F9 C9                   RET
05FA            ;
05FA            ;
05FA            ;    NEW LINE AND PRINT HL REG.(ASCII)
05FA            ;
05FA            NLPHL:    ENT
05FA CD0900               CALL   NL
05FD CDBA03               CALL   PRTHL
0600 C9                   RET
0601            ;
0601            ;
0601            ;ORG 0601H;EDGE
0601            ;
0601            ;
0601            ;    EDGE      (TAPE DATA EDGE DETECT)
0601            ;
0601            ;    BC=KEYPB  ($E001)
0601            ;    DE=CSTR   ($E002)
0601            ;    EXIT CF=0 OK :    CF=1 BREAK
0601            ;
0601            EDGE:     ENT
0601 3EF8                 LD     A,F8H               ; BREAK KEY IN
0603 3200E0               LD     (KEYPA),A
0606 00                   NOP
0607            EDG1:     ENT
0607 0A                   LD     A,(BC)
0608 E681                 AND    81H                 ; SHIFT & BREAK
060A 2002                 JR     NZ,+4
060C 37                   SCF
060D C9                   RET
060E 1A                   LD     A,(DE)
060F E620                 AND    20H
0611 20F4                 JR     NZ,EDG1             ; CSTR D5 = 0
0613            EDG2:     ENT
0613 0A                   LD     A,(BC)              ; 8
0614 E681                 AND    81H                 ; 9
0616 2002                 JR     NZ,+4               ; 10/14
0618 37                   SCF
0619 C9                   RET
061A 1A                   LD     A,(DE)              ; 8
061B E620                 AND    20H                 ; 9
061D 28F4                 JR     Z,EDG2              ; CSTR D5 = 1   :10/14
061F C9                   RET                        ; 11
0620            ;
0620            ;
```

```
** Z80 ASSEMBLER SB-7201  <1Z-013A>  PAGE 23            04.07.S3
0620                       DEFS   +4
0624                 ;ORG 0624H;RBYTE
0624                 ;
0624                 ;
0624                 ;   1 BYTE READ
0624                 ;
0624                 ;     EXIT  SUMDT=STORE
0624                 ;      CF=1 : BREAK
0624                 ;
0624                 ;      CF=0 : DATA=ACC
0624                 ;
0624                 RBYTE:  ENT
0624 C5                      PUSH   BC
0625 D5                      PUSH   DE
0626 E5                      PUSH   HL
0627 210008                  LD     HL,0800H
062A 0101E0                  LD     BC,KEYPB           ; KEY DATA $E001
062D 1102E0                  LD     DE,CSTR            ; $TAPE DATA $E002
0630                 RBY1:   ENT
0630 CD0106                  CALL   EDGE               ; 41 OR 101
0633 DA5406                  JP     C,RBY3             ; 13
0636 CD4A0A                  CALL   DLY3               ; 20+18*63+33
0639 1A                      LD     A,(DE)             ; DATA READ :8
063A E620                    AND    20H
063C CA4906                  JP     Z,RBY2
063F E5                      PUSH   HL
0640 2A9711                  LD     HL,(SUMDT)
0643 23                      INC    HL
0644 229711                  LD     (SUMDT),HL
0647 E1                      POP    HL
0648 37                      SCF
0649                 RBY2:   ENT
0649 7D                      LD     A,L
064A 17                      RLA
064B 6F                      LD     L,A
064C 25                      DEC    H
064D C23006                  JP     NZ,RBY1
0650 CD0106                  CALL   EDGE
0653 7D                      LD     A,L
0654                 RBY3:   ENT
0654 E1                      POP    HL
0655 D1                      POP    DE
0656 C1                      POP    BC
0657 C9                      RET
0658                 ;
0658                 ;
0658                 ;   TAPE MARK DETECT
0658                 ;
0658                 ;     E=@L@ :INFORMATION
0658                 ;      =@S@ :DATA
0658                 ;    EXIT CF=0 :OK
0658                 ;           =1 :BREAK
0658                 ;
0658                         DEFS   +3
065B                 ;
065B                 TMARK:  ENT
065B                 ;
065B                 ;ORG 065BH
065B                 ;
065B CDE20F                  CALL   GAPCK
```

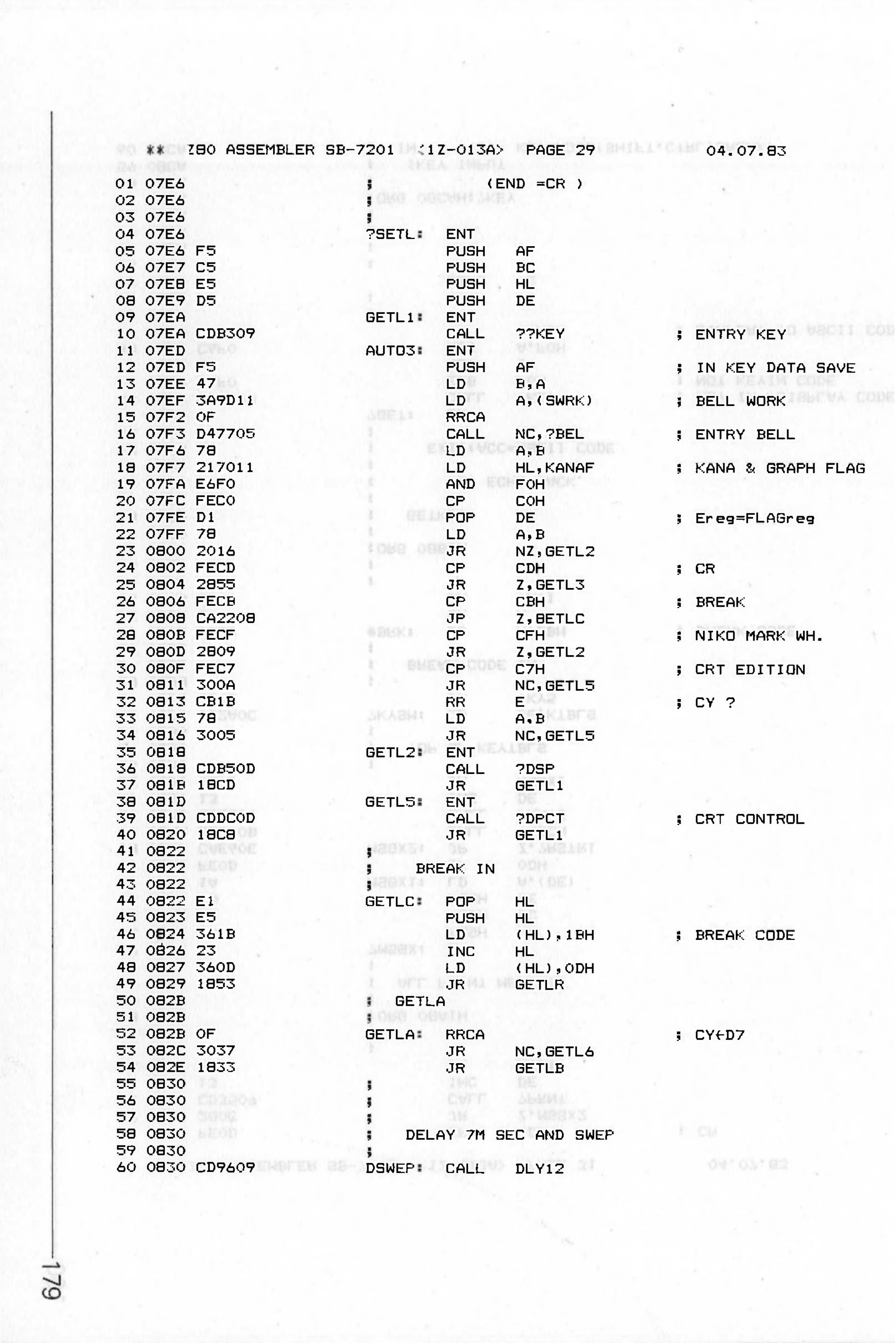
```
0665E C5                        PUSH   BC              ; ORG 065EH
065F D5                         PUSH   DE
0660 E5                         PUSH   HL
0661 212828                     LD     HL,2828H
0664 7B                         LD     A,E
0665 FECC                       CP     CCH             ;'L'
0667 2803                       JR     Z,+5
0669 211414                     LD     HL,1414H
066C 229511                     LD     (TMCNT),HL
066F 0101E0                     LD     BC,KEYPB
0672 1102E0                     LD     DE,CSTR
0675                   TM1:     ENT
0675 2A9511                     LD     HL,(TMCNT)
0678                   TM2:     ENT
0678 CD0106                     CALL   EDGE
067B 381E                       JR     C,TM4
067D CD4A0A                     CALL   DLY3            ; CALL DLY2*3
0680 1A                         LD     A,(DE)
0681 E620                       AND    20H
0683 28F0                       JR     Z,TM1
0685 25                         DEC    H
0686 20F0                       JR     NZ,TM2
0688                   TM3:     ENT
0688 CD0106                     CALL   EDGE
068B 380E                       JR     C,TM4
068D CD4A0A                     CALL   DLY3            ; CALL DLY2*3
0690 1A                         LD     A,(DE)
0691 E620                       AND    20H
0693 20E0                       JR     NZ,TM1
0695 2D                         DEC    L
0696 20F0                       JR     NZ,TM3
0698 CD0106                     CALL   EDGE
069B                   RET3:    ENT
069B                   TM4:     ENT
069B E1                         POP    HL
069C D1                         POP    DE
069D C1                         POP    BC
069E C9                         RET
069F                   ;
069F                   ;
069F                   ;   MOTOR ON
069F                   ;
069F                   ;     IN  D=@W@ :WRITE
069F                   ;           =@R@ :READ
069F                   ;     EXIT CF=0 :OK
069F                   ;            =1 :BREAK
069F                   MOTOR:   ENT
069F C5                         PUSH   BC
06A0 D5                         PUSH   DE
06A1 E5                         PUSH   HL
06A2 060A                       LD     B,10
06A4                   MOT1:    ENT
06A4 3A02E0                     LD     A,(CSTR)
06A7 E610                       AND    10H
06A9 280E                       JR     Z,MOT4
06AB                   MOT2:    ENT
06AB 06FF                       LD     B,FFH           ; 2 SEC DELAY
06AD CD9609                     CALL   DLY12           ; 7 MSEC DELAY
06B0 1802                       JR     +4              ;MOTOR ENTRY ADJUST
06B2 18EB                       JR     MOTOR           ; ORG 06B2H
```

```
06B4 10F7                DJNZ   -7
06B6 AF                  XOR    A
06B7             MOT7:   ENT
06B7 18E2                JR     RET3
06B9             MOT4:   ENT
06B9 3E06                LD     A,06H
06BB 2103E0              LD     HL,CSTPT
06BE 77                  LD     (HL),A
06BF 3C                  INC    A
06C0 77                  LD     (HL),A
06C1 10E1                DJNZ   MOT1
06C3 CD0900              CALL   NL
06C6 7A                  LD     A,D
06C7 FED7                CP     D7H             ; ´W´
06C9 2805                JR     Z,MOTS
06CB 11FB03              LD     DE,MSG#1        ; PLAY MARK
06CE 1807                JR     MOT9
06D0             MOT8:   ENT
06D0 110204              LD     DE,MSG#3        ; "RECORD."
06D3 DF                  RST    3               ; CALL MSGX
06D4 11FD03              LD     DE,MSG#2        ; "PLAY"
06D7             MOT9:   ENT
06D7 DF                  RST    3               ; CALL MSGX
06D8             MOT5:   ENT
06D8 3A02E0              LD     A,(CSTR)
06DB E610                AND    10H
06DD 20CC                JR     NZ,MOT2
06DF CD320A              CALL   ?BRK
06E2 20F4                JR     NZ,MOT5
06E4 37                  SCF
06E5 18D0                JR     MOT7
06E7             ;
06E7             ;    INITIAL MESSAGE
06E7             ;
06E7             MSG?3:  ENT
06E7 2A2A2020            DEFM    ´**  MONITOR 1Z-013A  **´
06EB 4D4F4E49
06EF 544F5220
06F3 315A2D30
06F7 31334120
06FB 202A2A
06FE 0D                  DEFB   0DH
06FF             ;
06FF             ;
06FF                     DEFS   +1
0700             ;
0700             ;
0700             ;ORG 0700H;MSTOP
0700             ;
0700             ;
0700             ;  MOTOR STOP
0700             ;
0700             MSTOP:  ENT
0700 F5                  PUSH   AF
0701 C5                  PUSH   BC
0702 D5                  PUSH   DE
0703 060A                LD     B,10
0705             MST1:   ENT
0705 3A02E0              LD     A,(CSTR)
0708 E610                AND    10H
```

```
070A 280B                   JR      Z,MST3
070C               MST2:    ENT
070C 3E06                   LD      A,06H
070E 3203E0                 LD      (CSTPT),A
0711 3C                     INC     A
0712 3203E0                 LD      (CSTPT),A
0715 10EE                   DJNZ    MST1
0717               MST3:    ENT
0717 C3E60E                 JP      ?RSTR1
071A               ;
071A               ;
071A               ;
071A               ;
071A               ;    CHECK SUM
071A               ;
071A               ;     IN  BC=SIZE
071A               ;         HL=DATA ADR.
071A               ;    EXIT SUMDT=STORE
071A               ;         CSMDT=STORE
071A               ;
071A               CKSUM:   ENT
071A C5                     PUSH    BC
071B D5                     PUSH    DE
071C E5                     PUSH    HL
071D 110000                 LD      DE,0
0720               CKS1:    ENT
0720 78                     LD      A,B
0721 B1                     OR      C
0722 200B                   JR      NZ,CKS2
0724 EB                     EX      DE,HL
0725 229711                 LD      (SUMDT),HL
0728 229911                 LD      (CSMDT),HL
072B E1                     POP     HL
072C D1                     POP     DE
072D C1                     POP     BC
072E C9                     RET
072F               CKS2:    ENT
072F 7E                     LD      A,(HL)
0730 C5                     PUSH    BC
0731 0608                   LD      B,+8
0733               CKS3:    ENT
0733 07                     RLCA
0734 3001                   JR      NC,+3
0736 13                     INC     DE
0737 10FA                   DJNZ    CKS3
0739 C1                     POP     BC
073A 23                     INC     HL
073B 0B                     DEC     BC
073C 18E2                   JR      CKS1
073E               ;
073E               ;    MODE SET OF KEYPORT
073E               ;
073E               ?MODE:   ENT
073E 2103E0                 LD      HL,KEYPF
0741 368A                   LD      (HL),8AH          ; 10001010
0743 3607                   LD      (HL),07H          ; PC3=1
0745 3605                   LD      (HL),05H          ; PC2=1
0747               VGOFF:   ENT
0747               ;
0747 C9                     RET
```

```
**  Z80 ASSEMBLER SB-7201  <1Z-013A>  PAGE 27            04.07.83
0748         ;
0748         ;
0748                 DEFS   +17
0759         ;
0759         ;ORG 0759H;DLY1
0759         ;
0759         ;    107 MICRO SEC DELY
0759         ;
0759         DLY1:   ENT
0759 3E15            LD     A,15H           ; 18*21+20
075B 3D              DEC    A
075C C25B07          JP     NZ,-1
075F C9              RET
0760         ;
0760         ;ORG 0760H;DLY2
0760         ;
0760         DLY2:   ENT
0760 3E13            LD     A,13H           ; 18*19+20
0762 3D              DEC    A
0763 C26207          JP     NZ,-1
0766 C9              RET
0767         ;
0767         ;
0767         ;
0767         ;
0767         ;
0767         ;  1 BYTE WRITE
0767         ;
0767         WBYTE:  ENT
0767 C5              PUSH   BC
0768 0608            LD     B,+8
076A CD1A0A          CALL   LONG
076D         WBY1:   ENT
076D 07              RLCA
076E DC1A0A          CALL   C,LONG
0771 D4010A          CALL   NC,SHORT
0774 05              DEC    B
0775 C26D07          JP     NZ,WBY1
0778 C1              POP    BC
0779 C9              RET
077A         ;
077A         ;
077A         ;   GAP + TAPEMARK
077A         ;
077A         ;    E=@L@ LONG GAP
077A         ;     =@S@ SHORT GAP
077A         ;
077A         GAP:    ENT
077A C5              PUSH   BC
077B D5              PUSH   DE
077C 7B              LD     A,E
077D 01F055          LD     BC,55F0H
0780 112828          LD     DE,2828H
0783 FECC            CP     CCH             ;'L'
0785 CA8E07          JP     Z,GAP1
0788 01F82A          LD     BC,2AF8H
078B 111414          LD     DE,1414H
078E         GAP1:   ENT
078E CD010A          CALL   SHORT
0791 0B              DEC    BC
```

```
0792 78                   LD     A,B
0793 B1                   OR     C
0794 20F8                 JR     NZ,-6
0796              GAP2:   ENT
0796 CD1A0A               CALL   LONG
0799 15                   DEC    D
079A 20FA                 JR     NZ,-4
079C              GAP3:   ENT
079C CD010A               CALL   SHORT
079F 1D                   DEC    E
07A0 20FA                 JR     NZ,-4
07A2 CD1A0A               CALL   LONG
07A5 D1                   POP    DE
07A6 C1                   POP    BC
07A7 C9                   RET
07A8              ;
07A8              ;    MEMORY CORRECTION
07A8              ;       COMMAND 'M'
07A8              ;
07A8              MCOR.   ENT
07A8 CD3D01               CALL   HEXIY              ; CRRECTION ADR.
07AB              MCR1:   ENT
07AB CDFA05               CALL   NLPHL              ; COR. ADR. PRINT
07AE CDB103               CALL   SPHEX              ; ACC ⇒ ASCII DISP.
07B1 CD2009               CALL   ?PRTS              ; SPACE PRINT
07B4 CD2F01               CALL   BGETL              ; GET DATA & CHECK DATA
07B7 CD1004               CALL   HLHEX              ; HL←ASCII(DE)
07BA 381B                 JR     C,MCR3
07BC CDA602               CALL   .4DE               ; (INC DE)*4
07BF 13                   INC    DE
07C0 CD1F04               CALL   2HEX               ; DATA CHECK
07C3 38E6                 JR     C,MCR1
07C5 BE                   CP     (HL)
07C6 20E3                 JR     NZ,MCR1
07C8 13                   INC    DE
07C9 1A                   LD     A,(DE)
07CA FE0D                 CP     0DH                ; NOT CORRECTION ?
07CC 2806                 JR     Z,MCR2
07CE CD1F04               CALL   2HEX               ; ACC←HL(ASCII)
07D1 38D8                 JR     C,MCR1
07D3 77                   LD     (HL),A             ; DATA CORRECT
07D4              MCR2:   ENT
07D4 23                   INC    HL
07D5 18D4                 JR     MCR1
07D7              ;
07D7 60           MCR3:   LD     H,B                ; MEMORY ADR.
07D8 69                   LD     L,C
07D9 18D0                 JR     MCR1
07DB              ;
07DB              ;
07DB              ;
07DB              ;
07DB              ;
07E6                      ORG    07E6H
07E6              ;
07E6              ;
07E6              ;
07E6              ;   GET 1 LINE STATEMENT  *
07E6              ;
07E6              ;      DE = DATA STORE LOW ADR.
```



```
** Z80 ASSEMBLER SB-7201 <1Z-013A>  PAGE 29             04.07.83

07E6                ;            (END =CR )
07E6                ;
07E6                ;
07E6                ?SETL:  ENT
07E6 F5                     PUSH    AF
07E7 C5                     PUSH    BC
07E8 E5                     PUSH    HL
07E9 D5                     PUSH    DE
07EA                GETL1:  ENT
07EA CDB309                 CALL    ??KEY           ; ENTRY KEY
07ED                AUTO3:  ENT
07ED F5                     PUSH    AF              ; IN KEY DATA SAVE
07EE 47                     LD      B,A
07EF 3A9D11                 LD      A,(SWRK)        ; BELL WORK
07F2 0F                     RRCA
07F3 D47705                 CALL    NC,?BEL         ; ENTRY BELL
07F6 78                     LD      A,B
07F7 217011                 LD      HL,KANAF        ; KANA & GRAPH FLAG
07FA E6F0                   AND     F0H
07FC FEC0                   CP      C0H
07FE D1                     POP     DE              ; Ereg=FLAGreg
07FF 78                     LD      A,B
0800 2016                   JR      NZ,GETL2
0802 FECD                   CP      CDH             ; CR
0804 2855                   JR      Z,GETL3
0806 FECB                   CP      CBH             ; BREAK
0808 CA2208                 JP      Z,BETLC
080B FECF                   CP      CFH             ; NIKO MARK WH.
080D 2809                   JR      Z,GETL2
080F FEC7                   CP      C7H             ; CRT EDITION
0811 300A                   JR      NC,GETL5
0813 CB1B                   RR      E               ; CY ?
0815 78                     LD      A.B
0816 3005                   JR      NC,GETL5
0818                GETL2:  ENT
0818 CDB50D                 CALL    ?DSP
081B 18CD                   JR      GETL1
081D                GETL5:  ENT
081D CDDC0D                 CALL    ?DPCT           ; CRT CONTROL
0820 18C8                   JR      GETL1
0822                ;
0822                ;     BREAK IN
0822                ;
0822 E1             GETLC:  POP     HL
0823 E5                     PUSH    HL
0824 361B                   LD      (HL),1BH        ; BREAK CODE
0826 23                     INC     HL
0827 360D                   LD      (HL),0DH
0829 1853                   JR      GETLR
082B                ;  GETLA
082B                ;
082B 0F             GETLA:  RRCA                    ; CY←D7
082C 3037                   JR      NC,GETL6
082E 1833                   JR      GETLB
0830                ;
0830                ;
0830                ;
0830                ;    DELAY 7M SEC AND SWEP
0830                ;
0830 CD9609         DSWEP:  CALL    DLY12
```

```
0833 CD500A                    CALL    ?SWEP
0836 C9                        RET
0837                  ;
0837                  ;
0837                           DEFS    36
085B                  ;
085B                  ;
085B                  ;
085B                  ; ORB 085BH;GETL3
085B                  ;
085B CDF302           GETL3:   CALL    .MANS          ; CR
085E 0628                      LD      B,40           ; 1LINE
0S60 30C9                      JR      NC,GETLA
0862 25                        DEC     H              ; BEFORE LINE
0863 0650             GETLB:   LD      B,80           ; 2 LINE
0865 2E00             GETL6:   LD      L,0
0867 CDB40F                    CALL    ?PNT1
086A D1                        POP     DE             ; STORE TOP ADR.
086B D5                        PUSH    DE
086C 7E               GETLZ:   LD      A,(HL)
086D CDCE0B                    CALL    ?DACN
0870 12                        LD      (DE),A
0871 23                        INC     HL
0872 13                        INC     DE
0873 10F7                      DJNZ    GETLZ
0875 EB                        EX      DE,HL
0876 360D             GETLU:   LD      (HL),0DH
0878 2B                        DEC     HL
0879 7E                        LD      A,(HL)
087A FE20                      CP      20H            ; SPACE THEN CR
087C                  ;
087C                  ;
087C                  ;   CR AND NEW LINE
087C                  ;
087C 28F8                      JR      Z,GETLU
087E                  ;
087E                  ;   NEW LINE RETURN
087E                  ;
087E CD0E09           GETLR:   CALL    ?LTNL
0881 D1                        POP     DE
0882 E1                        POP     HL
0883 C1                        POP     BC
0884 F1                        POP     AF
0885 C9                        RET
0886                  ;
0886                  ;
0886                  ;
0886                           DEFS    +13
0893                  ;ORG 0893H
0893                  ;
0893                  ;   MESSAGE PRINT
0893                  ;
0893                  ;     DE PRINT DATA LOW ADR.
0893                  ;         END=CR
0893                  ;
0893                  ?MSG:    ENT
0893 F5                        PUSH    AF
0894 C5                        PUSH    BC
0895 D5                        PUSH    DE
0896 1A               MSG1:    LD      A,(DE)
```

```
** Z80 ASSEMBLER SB-7201 <1Z-013A>  PAGE 31             04.07.83
0897 FEOD                        CP      ODH             ; CR
0899 280C                        JR      Z,MSGX2
089B CD3509                      CALL    ?PRNT
089E 13                          INC     DE
089F 18F5                        JR      MSG1
08A1                     ;
08A1                     ;
08A1                     ;ORS 08A1H
08A1                     ;
08A1                     ;  ALL PRINT MESSABE
08A1                     ;
08A1                     ?MSGX:  ENT
08A1 F5                          PUSH    AF
08A2 C5                          PUSH    BC
08A3 D5                          PUSH    DE
08A4 1A                  MSGX1:  LD      A,(DE)
08A5 FEOD                        CP      ODH
08A7 CAE60E              MSGX2:  JP      Z,?RSTR1
08AA CDB90B                      CALL    ?ADCN
08AD CD6C09                      CALL    PRNT3
08B0 13                          INC     DE
08B1 18F1                        JR      MSGX1
08B3                     ;
08B3                     ;   TOP OF KEYTBLS
08B3                     ;
08B3 112A0C              ?KYSM:  LD      DE,KTBLS
08B6 1842                        JR      ?KY5
08B8                     ;
08B8                     ;   BREAK CODE IN
08B8                     ;
08B8 3ECB                #BRK:   LD      A,CBH           ; BREAK CODE
08BA B7                          OR      A
08BB 1819                        JR      ?KY1
08BD                     ;
08BD                     ;
08BD                     ;ORG 08BDH
08BD                     ;
08BD                     ;   GETKEY
08BD                     ;
08BD                     ;       NOT ECHO BACK
08BD                     ;
08BD                     ;     EXIT:ACC=ASCII CODE
08BD                     ;
08BD                     ?GET:   ENT
08BD CDCA08                      CALL    ?KEY            ; KEY IN (DISPLAY CODE)
08C0 D6F0                        SUB     FOH             ; NOT KEYIN CODE
08C2 C8                          RET     Z
08C3 C6F0                        ADD     A,FOH
08C5 C3CE0B                      JP      ?DACN           ; DIAPLAY TO ASCII CODE
08C8                     ;
08C8                     ;
08C8                             DEFS    +2
08CA                     ;
08CA                     ;
08CA                     ;
08CA                     ;
08CA                     ;ORG 08CAH;?KEY
08CA                     ;
08CA                     ;   1KEY INPUT
08CA                     ;  IN       B = KEY MODE(SHIFT,CTRL,BREAK)
```

```
08CA                ;        C = KEY DATA (COLUMN & ROW)
08CA                ;  EXIT  ACC=DISPLAY CODE
08CA                ;        IF NO KEY   ACC=F0H
08CA                ;      IF CY=1 THEN ATTRIBUTE ON
08CA                ;                (SMALL,HIRAKANA)
08CA                ;
08CA                ?KEY:   ENT
08CA C5                     PUSH   BC
08CB D5                     PUSH   DE
08CC E5                     PUSH   HL
08CD CD3008                 CALL   DSWEP            ; DELAY AND KEY SWEP
08D0 78                     LD     A,B
08D1 07                     RLCA
08D2 3806                   JR     C,?KY2
08D4 3EF0                   LD     A,F0H
08D6                ?KY1:   ENT
08D6 E1                     POP    HL
08D7 D1                     POP    DE
08D8 C1                     POP    BC
08D9 C9                     RET
08DA                ;
08DA                ?KY2:   ENT
08DA 11EA0B                 LD     DE,KTBL          ; NORMAL KEY TABLE
08DD 78                     LD     A,B
08DE FE88                   CP     88H              ; BREAK IN
08E0 28D6                   JR     Z,#BRK
08E2 2600                   LD     H,0              ; HL=ROW & COLUMN
08E4 69                     LD     L,C
08E5 CB6F                   BIT    5,A              ; CTRL CHECK
08E7 200E                   JR     NZ,?KY5-3
08E9 3A7011                 LD     A,(KANAF)        ; 0=NR.,1=GRAPH
08EC 0F                     RRCA
08ED DAFE08                 JP     C,?KYGRP         ; GRAPH MODE
08F0 78                     LD     A,B              ; CTRL KEY CHECK
08F1 17                     RLA
08F2 17                     RLA
08F3 38BE                   JR     C,?KYBM
08F5 1803                   JR     ?KY5
08F7 11AA0C                 LD     DE,KTBLC         ; CONTROL KEY TABLE
08FA                ?KY5:   ENT
08FA 19                     ADD    HL,DE            ; TABLE
08FB                ?KY55:  ENT
08FB 7E                     LD     A,(HL)
08FC 18D8                   JR     ?KY1
08FE                ?KYGRP: ENT
08FE CB70                   BIT    6,B
0900 2807                   JR     Z,?KYGRB
0902 11E90C                 LD     DE,KTBLG
0905 19                     ADD    HL,DE
0906 37                     SCF
0907 18F2                   JR     ?KY55
0909                ;
0909 116A0C         ?KYGRS: LD     DE,KTBLGS
090C 18EC                   JR     ?KY5
090E                ;
090E                ;
090E                ;
090E                ;
090E                ;
090E                ;ORG 090EH
```

```
 **  Z80 ASSEMBLER SB-7201  <1Z-013A>  PA:E 33              04.07.83
090E                ;
090E                ;   NEWLINE
090E                ;
090E                ?LTNL:    ENT
090E AF                       XOR     A
090F 329411                   LD      (DPRNT),A          ; ROW POINTER
0912 3ECD                     LD      A,CDH              ; CR
0914 1843                     JR      PRNT5
0916                          DEFS    +2
0918                ;ORG 0918H
0918                ;
0918                ?NL:      ENT
0918 3A9411                   LD      A,(DPRNT)
091B B7                       OR      A
091C C8                       RET     Z
091D 18EF                     JR      ?LTNL
091F                          DEFS    +1
0920                ;ORG 0920H
0920                ;
0920                ;   PRINT SPACE
0920                ;
0920                ?PRTS:    ENT
0920 3E20                     LD      A,20H
0922 1811                     JR      ?PRNT
0924                ;
0924                ;   PRINT TAB
0924                ;
0924                ?PRTT:    ENT
0924 CD0C00                   CALL    PRNTS
0927 3A9411                   LD      A,(DPRNT)
092A B7                       OR      A
092B C8                       RET     Z
092C D60A                     SUB     +10
092E 38F4                     JR      C,-10
0930 20FA                     JR      NZ,-4
0932                          DEFS    +3
0935                ;ORG 0935H
0935                ;
0935                ;   PRINT
0935                ;
0935                ;      IN ACC = PRINT DATA (ASCII)
0935                ;
0935                ?PRNT:    ENT
0935 FE0D                     CP      0DH                ; CR
0937 28D5                     JR      Z,?LTNL
0939 C5                       PUSH    BC
093A 4F                       LD      C,A
093B 47                       LD      B,A
093C CD4609                   CALL    ?PRT
093F 78                       LD      A,B
0940 C1                       POP     BC
0941 C9                       RET
0942                ;
0942                ;
0942                MSGOK:    ENT
0942 4F4B21                   DEFM    'OK!'
0945 0D                       DEFB    0DH
0946                ;ORG 0946H
0946                ;
0946                ;  PRINT ROUTINE
```

```
0946                ;        1 CHA.
0946                ;   INPUT:C=ASCII DATA (?DSP+?DPCT)
0946                ;
0946                ?PRT:   ENT
0946 79                     LD      A,C
0947 CDB90B                 CALL    ?ADCN           ; ASCII TO DSPLAY
094A 4F                     LD      C,A
094B FEF0                   CP      F0H
094D C8                     RET     Z               ; ZERO=ILLEGAL DATA
094E E6F0                   AND     F0H             ; MSD CHECK
0950 FEC0                   CP      C0H
0952 79                     LD      A,C
0953 2017                   JR      NZ,PRNT3
0955 FEC7                   CP      C7H
0957 3013                   JR      NC,PRNT3        ; CRT EDITOR
0959                PRNT5:  ENT
0959 CDDC0D                 CALL    ?DPCT
095C FEC3                   CP      C3H
095E 280F                   JR      Z,PRNT4
0960 FEC5                   CP      C5H             ; HOME
0962 2803                   JR      Z,PRNT2
0964 FEC6                   CP      C6H             ; CLR
0966 C0                     RET     NZ
0967 AF             PRNT2:  XOR     A
0968 329411                 LD      (DPRNT),A
096B C9                     RET
096C                PRNT3:  ENT
096C CDB50D                 CALL    ?DSP
096F 3A9411         PRNT4:  LD      A,(DPRNT)       ; TAB POINT+1
0972 3C                     INC     A
0973 FE50                   CP      +80
0975 38F1                   JR      C,PRNT2+1
0977 D650                   SUB     +80
0979 18ED                   JR      PRNT2+1
097B                ;
097B                ;
097B                ;
097B                ;
097B                ;
097B                ;   FLASSING BYPASS 1
097B                ;
097B                FLAS1:  ENT
097B 3A8E11                 LD      A,(FLASH)
097E 186F                   JR      FLAS2
0980                ;
0980                ;   BREAK SUBROUTINE BYPASS 1
0980                ;
0980                ;       CTRL OR NOT KEY
0980                ;
0980                ?BRK2:  ENT
0980 CB6F                   BIT     5,A             ; NOT OR CTRL
0982 2802                   JR      Z,?BRK3         ; CTRL
0984 B7                     OR      A               ; NOTKEY A=7FH
0985 C9                     RET
0986                ;
0986 3E20           ?BRK3:  LD      A,20H           ; CTRL D5=1
0988 B7                     OR      A               ; ZERO FLS. CLR
0989 37                     SCF
098A C9                     RET
098B                ;
```

```
** Z80 ASSEMBLER SB-7201 <1Z-013A>  PAGE 35          04.07.83
0988                    MSGSV:  ENT
098B 46494C45                   DEFM    'FILENAME? '
098F 4E414D45
0993 3F20
0995 0D                         DEFB    0DH
0996                    ;
0996                    ;   DLY 7  MSEC
0996                    ;
0996                    DLY12:  ENT
0996 C5                         PUSH    BC
0997 0615                       LD      B,15H
0999 CD4A0A                     CALL    DLY3
099C 10FB                       DJNZ    -3
099E C1                         POP     BC
099F C9                         RET
09A0                    ;
09A0                    ;
09A0                    ;
09A0                    ;   LOADING MESSAGE
09A0                    ;
09A0                    MSG?2:  ENT
09A0 4C4F4144                   DEFM    'LOADING '
09A4 494E4720
09A8 0D                         DEFB    0DH
09A9                    ;
09A9                    ;
09A9                    ;
09A9                    ;   DELAY FOR LONG PULSE
09A9                    ;
09A9                    DLY4:   ENT
09A9 3E59                       LD      A,59H           ; 18*89+20
09AB 3D                         DEC     A
09AC C2AB09                     JP      NZ,-1
09AF C9                         RET
09B0                    ;
09B0                    ;
09B0                            DEFS    +3
09B3                    ;
09B3                    ;
09B3                    ;ORG 09B3H;??KEY
09B3                    ;
09B3                    ;   KEY BOAD SEARCH
09B3                    ;      & DISPLAY CODE CONV.
09B3                    ;
09B3                    ;      EXIT A = DISPLAY CODE
09B3                    ;           CY= GRAPH MODE
09B3                    ;      WITH CURSOR DISPLAY
09B3                    ;
09B3                    ??KEY:  ENT
09B3 E5                         PUSH    HL
09B4 CD920B                     CALL    ?SAVE
09B7                    KSL1:   ENT
09B7 CD7E05                     CALL    FLKEY           ; KEY
09BA 20FB                       JR      NZ,KSL1         ; KEY IN THEN JUMP
09BC                    KSL2:   ENT
09BC CD7E05                     CALL    FLKEY
09BF 28FB                       JR      Z,KSL2          ; NOT KEY IN THEN JUMP
09C1 67                         LD      H,A
09C2 CD9609                     CALL    DLY12           ; DELAY CHATTER
09C5 CDCA08                     CALL    ?KEY
```

```
09C8 F5                 PUSH   AF
09C9 BC                 CP     H              ; CHATER CHECK
09CA E1                 POP    HL
09CB 20EF               JR     NZ,KSL2
09CD E5                 PUSH   HL
09CE F1                 POP    AF             ; IN KEY DATA
09CF CDF005             CALL   ?LOAD          ; FLSHING DATA LOAD
09D2 E1                 POP    HL
09D3 C9                 RET
09D4           ;
09D4           ;
09D4           ;    CLEAR 2
09D4           ;
09D4           #CLR08:  ENT                   ; CY FLG.
09D4 AF                 XOR    A
09D5           #CLR8:   ENT
09D5 010008             LD     BC,0800H
09D8           CLEAR:   ENT                   ; BC = CLR BYTE SIZE
09D8 D5                 PUSH   DE             ; A = CLR DATA
09D9 57                 LD     D,A
09DA           CLEAR1:  ENT
09DA 72                 LD     (HL),D
09DB 23                 INC    HL
09DC 0B                 DEC    BC
09DD 78                 LD     A,B
09DE B1                 OR     C
09DF 20F9               JR     NZ,CLEAR1
09E1 D1                 POP    DE
09E2 C9                 RET
09E3           ;
09E3           ;
09E3           ;
09E3           ;
09E3           ;    FLASHING 2
09E3           ;
09E3           ?FLS:    ENT
09E3 F5                 PUSH   AF
09E4 E5                 PUSH   HL
09E5 3A02E0             LD     A,(KEYPC)
09E8 07                 RLCA
09E9 07                 RLCA
09EA 388F               JR     C,FLAS1
09EC 3A9211             LD     A,(FLSDT)
09EF           FLAS2:   ENT
09EF CDB10F             CALL   ?PONT          ; DISPLAY POSITION
09F2 77                 LD     (HL),A
09F3           FLAS3:   ENT
09F3 E1                 POP    HL
09F4 F1                 POP    AF
09F5 C9                 RET
09F6           ;
09F6           ;
09F6           ;
09F6                    DEFS   +9
09FF           ;
09FF           ;
09FF           ;ORG 09FF ; ?FLAS
09FF           ;
09FF           ?FLAS:   ENT
09FF 18E2               JR     ?FLS
```

```
** Z80 ASSEMBLER SB-7201  <1Z-013A>  PAGE 37          04.07.83

0A01                  ;
0A01                  ;
0A01                  ;
0A01                  ;  SHORT AND LONG PULSE FOR 1 BIT WRITE
0A01                  ;
0A01                  SHORT:  ENT
0A01 F5                       PUSH   AF              ; 12
0A02 3E03                     LD     A,03H           ; 9
0A04 3203E0                   LD     (CSTPT),A       ; $E003 PC3=1:16
0A07 CD5907                   CALL   DLY1            ; 20+18*21+20
0A0A CD5907                   CALL   DLY1            ; 20+18*21+20
0A0D 3E02                     LD     A,02H           ; 9
0A0F 3203E0                   LD     (CSTPT),A       ; $E003 PC3=0:16
0A12 CD5907                   CALL   DLY1            ; 20+18*21+20
0A15 CD5907                   CALL   DLY1            ; 20+18*21+20
0A18 F1                       POP    AF              ; 11
0A19 C9                       RET                    ; 11
0A1A                  ;
0A1A                  ;
0A1A                  LONG:   ENT
0A1A F5                       PUSH   AF              ; 11
0A1B 3E03                     LD     A,03H           ; 9
0A1D 3203E0                   LD     (CSTPT),A       ; 16
0A20 CDA909                   CALL   DLY4            ; 20+18*89+20
0A23 3E02                     LD     A,02H           ; 9
0A25 3203E0                   LD     (CSTPT),A       ; 16
0A28 CDA909                   CALL   DLY4            ; 20+18*89+20
0A2B F1                       POP    AF              ; 11
0A2C C9                       RET                    ; 11
0A2D                  ;
0A2D                  ;
0A2D                          DEFS   +5
0A32                  ;
0A32                  ;
0A32                  ;ORG 0A32H
0A32                  ;
0A32                  ;   BREAK KEY CHECK
0A32                  ;     AND SHIFT,CTNL KEY CHECK
0A32                  ;
0A32                  ;     EXIT BREAK ON : ZERO=1
0A32                  ;                OFF: ZERO=0
0A32                  ;          NO KEY   : CY  =0
0A32                  ;          KEY IN   : CY  =1
0A32                  ;           A D6=1  : SHIFT ON
0A32                  ;               =0  :      OFF
0A32                  ;             D5=1  : CTRL ON
0A32                  ;               =0  :      OFF
0A32                  ;             D4=1  : SFT+CNT ON
0A32                  ;               =0            OFF
0A32
0A32                  ?BRK:   ENT
0A32 3EF8                     LD     A,F8H           ; LINE 8SWEEP
0A34 3200E0                   LD     (KEYPA),A
0A37 00                       NOP
0A38 3A01E0                   LD     A,(KEYPB)
0A3B B7                       OR     A
0A3C 1F                       RRA
0A3D DA8009                   JP     C,?BRK2         ; SHIFT ?
0A40 17                       RLA
0A41 17                       RLA
```

```
0A42 3004                JR     NC,?BRK1         ; BREAK ?
0A44 3E40                LD     A,40H            ; SHIFT D6=1
0A46 37                  SCF
0A47 C9                  RET
0A48            ;
0A48            ;
0A48 AF         ?BRK1:   XOR    A                ; SHIFT ?
0A49 C9                  RET
0A4A            ;
0A4A            ;
0A4A            ;    320 U SEC DELAY
0A4A            ;
0A4A            DLY3:    ENT
0A4A 3E3F                LD     A,3FH            ; 18*63+33
0A4C C36207              JP     0762H            ; JP DLY2+2
0A4F            ;
0A4F            ;
0A4F                     DEFS   +1
0A50            ;
0A50            ;
0A50            ;
0A50            ;ORG 0A50H ; ?SWEP
0A50            ;
0A50            ;
0A50            ;    KEY BOAD SWEEP
0A50            ;
0A50            ;    EXIT B,D7=0  NO DATA
0A50            ;            =1  DATA
0A50            ;         D6=0  SHIFT OFF
0A50            ;            =1  SHIFT ON
0A50            ;         D5=0  CTRL OFF
0A50            ;            =1  CTRL ON
0A50            ;         D4-0  SHIFT+CTRL OFF
0A50            ;            =1  SHIFT+CTRL ON
0A50            ;       C   = ROW & COLOUMN
0A50            ;         7 6 5 4 3 2 1 0
0A50            ;         * * ↑ ↑ ↑ ← ← ←
0A50            ;
0A50            ?SWEP:   ENT
0A50 D5                  PUSH   DE
0A51 E5                  PUSH   HL
0A52 AF                  XOR    A
0A53 06F8                LD     B,F8H
0A55 57                  LD     D,A
0A56 CD320A              CALL   ?BRK
0A59 2004                JR     NZ,SWEP6
0A5B 1688                LD     D,88H            ; BREAK ON
0A5D 1814                JR     SWEP9
0A5F            SWEP6:   ENT
0A5F 3005                JR     NC,SWEP0
0A61 57                  LD     D,A
0A62 1802                JR     SWEP0
0A64            SWEP01:  ENT
0A64 CBFA                SET    7,D
0A66            SWEP0:   ENT
0A66 05                  DEC    B
0A67 78                  LD     A,B
0A68 3200E0              LD     (KEYPA),A
0A6B FEEF                CP     EFH              ; MAP SWEEP END ?
0A6D 2008                JR     NZ,SWEP3
```

```
** Z80 ASSEMBLER SB-7201  <1Z-013A>  PAGE 39           04.07.83
0A6F FEF8                    CP     F8H                ; BREAK KEY ROW
0A71 28F3                    JR     Z,SWEP0
0A73               SWEP9:    ENT
0A73 42                      LD     B,D
0A74 E1                      POP    HL
0A75 D1                      POP    DE
0A76 C9                      RET
0A77               ;
0A77               SWEP3:    ENT
0A77 3A01E0                  LD     A,(KEYPB)
0A7A 2F                      CPL
0A7B B7                      OR     A
0A7C 28E8                    JR     Z,SWEP0
0A7E 5F                      LD     E,A
0A7F               SWEP2:    ENT
0A7F 2608                    LD     H,8
0A81 78                      LD     A,B
0A82 E60F                    AND    0FH
0A84 07                      RLCA
0A85 07                      RLCA
0A86 07                      RLCA
0A87 4F                      LD     C,A
0A88 7B                      LD     A,E
0A89 25                      DEC    H
0A8A 0F                      RRCA
0A8B 30FC                    JR     NC,-2
0A8D 7C                      LD     A,H
0A8E 81                      ADD    A,C
0A8F 4F                      LD     C,A
0A90 18D2                    JR     SWEP01
0A92               ;
0A92               ;
0A92               ;    ASCII TO DISPLAY CODE TABL   ;
0A92               ;
0A92               ATBL:
0A92               ;  00 - 0F  ;
0A92 F0                      DEFB   F0H                ; ↑@
0A93 F0                      DEFB   F0H                ; ↑A
0A94 F0                      DEFB   F0H                ; ↑B
0A95 F3                      DEFB   F3H                ; ↑C
0A96 F0                      DEFB   F0H                ; ↑D
0A97 F5                      DEFB   F5H                ; ↑E
0A98 F0                      DEFB   F0H                ; ↑F
0A99 F0                      DEFB   F0H                ; ↑G
0A9A F0                      DEFB   F0H                ; ↑H
0A9B F0                      DEFB   F0H                ; ↑I
0A9C F0                      DEFB   F0H                ; ↑J
0A9D F0                      DEFB   F0H                ; ↑K
0A9E F0                      DEFB   F0H                ; ↑L
0A9F F0                      DEFB   F0H                ; ↑M
0AA0 F0                      DEFB   F0H                ; ↑N
0AA1 F0                      DEFB   F0H                ; ↑O
0AA2               ;10 - 1F
0AA2 F0                      DEFB   F0H                ; ↑P
0AA3 C1                      DEFB   C1H                ; ↑Q CUR. DOWN
0AA4 C2                      DEFB   C2H                ; ↑R CUR. UP
0AA5 C3                      DEFB   C3H                ; ↑S CUR. RIGHT
0AA6 C4                      DEFB   C4H                ; ↑T CUR. LEFT
0AA7 C5                      DEFB   C5H                ; ↑U HOME
0AA8 C6                      DEFB   C6H                ; ↑V CLEAR
```

```
0AA9 F0                       DEFB   F0H              ; ↑W
0AAA F0                       DEFB   F0H              ; ↑X
0AAB F0                       DEFB   F0H              ; ↑Y
0AAC F0                       DEFB   F0H              ; ↑Z SEP.
0AAD F0                       DEFB   F0H              ; ↑[
0AAE F0                       DEFB   F0H              ; ↑\
0AAF F0                       DEFB   F0H              ; ↑]
0AB0 F0                       DEFB   F0H              ; ↑^
0AB1 F0                       DEFB   F0H              ; ↑-
0AB2                  ;20 - 2F ;
0AB2 00                       DEFB   00H              ; SPACE
0AB3 61                       DEFB   61H              ; !
0AB4 62                       DEFB   62H              ; "
0AB5 63                       DEFB   63H              ; #
0AB6 64                       DEFB   64H              ; $
0AB7 65                       DEFB   65H              ; %
0AB8 66                       DEFB   66H              ; &
0AB9 67                       DEFB   67H              ; '
0ABA 68                       DEFB   68H              ; (
0ABB 69                       DEFB   69H              ; )
0ABC 6B                       DEFB   6BH              ; *
0ABD 6A                       DEFB   6AH              ; +
0ABE 2F                       DEFB   2FH              ; ,
0ABF 2A                       DEFB   2AH              ; -
0AC0 2E                       DEFB   2EH              ; .
0AC1 2D                       DEFB   2DH
0AC2                  ;  30 -3F  ;
0AC2 20                       DEFB   20H              ; 0
0AC3 21                       DEFB   21H              ; 1
0AC4 22                       DEFB   22H              ; 2
0AC5 23                       DEFB   23H              ; 3
0AC6 24                       DEFB   24H              ; 4
0AC7 25                       DEFB   25H              ; 5
0AC8 26                       DEFB   26H              ; 6
0AC9 27                       DEFB   27H              ; 7
0ACA 28                       DEFB   28H              ; 8
0ACB 29                       DEFB   29H              ; 9
0ACC 4F                       DEFB   4FH              ; :
0ACD 2C                       DEFB   2CH              ; ;
0ACE 51                       DEFB   51H              ; <
0ACF 2B                       DEFB   2BH              ; =
0AD0 57                       DEFB   57H              ; >
0AD1 49                       DEFB   49H              ; ?
0AD2                  ;  40 - 4F  ;
0AD2 55                       DEFB   55H              ; @
0AD3 01                       DEFB   01H              ; A
0AD4 02                       DEFB   02H              ; B
0AD5 03                       DEFB   03H              ; C
0AD6 04                       DEFB   04H              ; D
0AD7 05                       DEFB   05H              ; E
0AD8 06                       DEFB   06H              ; F
0AD9 07                       DEFB   07H              ; G
0ADA 08                       DEFB   08H              ; H
0ADB 09                       DEFB   09H              ; I
0ADC 0A                       DEFB   0AH              ; J
0ADD 0B                       DEFB   0BH              ; K
0ADE 0C                       DEFB   0CH              ; L
0ADF 0D                       DEFB   0DH              ; M
0AE0 0E                       DEFB   0EH              ; U
0AE1 0F                       DEFB   0FH              ; O
```

```
0AE2                 ;  50 - 5F ;
0AE2 10                     DEFB    10H          ; P
0AE3 11                     DEFB    11H          ; Q
0AE4 12                     DEFB    12H          ; R
0AE5 13                     DEFB    13H          ; S
0AE6 14                     DEFB    14H          ; T
0AE7 15                     DEFB    15H          ; U
0AE8 16                     DEFB    16H          ; V
0AE9 17                     DEFB    17H          ; W
0AEA 18                     DEFB    18H          ; X
0AEB 19                     DEFB    19H          ; Y
0AEC 1A                     DEFB    1AH          ; Z
0AED 52                     DEFB    52H          ; [
0AEE 59                     DEFB    59H          ; \
0AEF 54                     DEFB    54H          ; ]
0AF0 50                     DEFB    50H          ; ^
0AF1 45                     DEFB    45H          ; _
0AF2                 ;  60 - 6F ;
0AF2 C7                     DEFB    C7H          ; UFO
0AF3 C8                     DEFB    C8H
0AF4 C9                     DEFB    C9H
0AF5 CA                     DEFB    CAH
0AF6 CB                     DEFB    CBH
0AF7 CC                     DEFB    CCH
0AF8 CD                     DEFB    CDH
0AF9 CE                     DEFB    CEH
0AFA CF                     DEFB    CFH
0AFB DF                     DEFB    DFH
0AFC E7                     DEFB    E7H
0AFD E8                     DEFB    E8H
0AFE E5                     DEFB    E5H
0AFF E9                     DEFB    E9H
0B00 EC                     DEFB    ECH
0B01 ED                     DEFB    EDH
0B02                 ;  70 - 7F ;
0B02 D0                     DEFB    D0H
0B03 D1                     DEFB    D1H
0B04 D2                     DEFB    D2H
0B05 D3                     DEFB    D3H
0B06 D4                     DEFB    D4H
0B07 D5                     DEFB    D5H
0B08 D6                     DEFB    D6H
0B09 D7                     DEFB    D7H
0B0A D8                     DEFB    D8H
0B0B D9                     DEFB    D9H
0B0C DA                     DEFB    DAH
0B0D DB                     DEFB    DBH
0B0E DC                     DEFB    DCH
0B0F DD                     DEFB    DDH
0B10 DE                     DEFB    DEH
0B11 C0                     DEFB    C0H
0B12                 ;  80 - 8F ;
0B12 80                     DEFB    80H          ; }
0B13 BD                     DEFB    BDH          ;
0B14 9D                     DEFB    9DH
0B15 B1                     DEFB    B1H
0B16 B5                     DEFB    B5H
0B17 B9                     DEFB    B9H
0B18 B4                     DEFB    B4H          ;
0B19 9E                     DEFB    9EH          ;
```

```
0B1A B2                     DEFB   B2H            ;
0B1B B6                     DEFB   B6H            ;
0B1C BA                     DEFB   BAH            ;
0B1D BE                     DEFB   BEH            ;
0B1E 9F                     DEFB   9FH            ;
0B1F B3                     DEFB   B3H            ;
0B20 B7                     DEFB   B7H            ;
0B21 BB                     DEFB   BBH            ;
0B22                 ;  90 - 9F  ;
0B22 BF                     DEFB   BFH            ; —
0B23 A3                     DEFB   A3H            ;
0B24 85                     DEFB   85H            ;
0B25 A4                     DEFB   A4H            ; `
0B26 A5                     DEFB   A5H            ; ~
0B27 A6                     DEFB   A6H            ;
0B28 94                     DEFB   94H            ;
0B29 87                     DEFB   87H            ;
0B2A 88                     DEFB   88H            ;
0B2B 9C                     DEFB   9CH            ;
0B2C 82                     DEFB   82H            ;
0B2D 98                     DEFB   98H            ;
0B2E 84                     DEFB   84H            ;
0B2F 92                     DEFB   92H            ;
0B30 90                     DEFB   90H            ;
0B31 83                     DEFB   83H            ;
0B32                 ;  A0 - AF  ;
0B32 91                     DEFB   91H            ;
0B33 81                     DEFB   81H            ;
0B34 9A                     DEFB   9AH            ;
0B35 97                     DEFB   97H            ;
0B36 93                     DEFB   93H            ;
0B37 95                     DEFB   95H            ;
0B38 89                     DEFB   89H            ;
0B39 A1                     DEFB   A1H            ;
0B3A AF                     DEFB   AFH            ;
0B3B 8B                     DEFB   8BH            ;
0B3C 86                     DEFB   86H            ;
0B3D 96                     DEFB   96H            ;
0B3E A2                     DEFB   A2H            ;
0B3F AB                     DEFB   ABH            ;
0B40 AA                     DEFB   AAH            ;
0B41 8A                     DEFB   8AH            ;
0B42                 ;  B0 - BF ;
0B42 8E                     DEFB   8EH            ;
0B43 B0                     DEFB   B0H            ;
0B44 AD                     DEFB   ADH            ;
0B45 8D                     DEFB   8DH            ;
0B46 A7                     DEFB   A7H            ;
0B47 A8                     DEFB   A8H            ;
0B48 A9                     DEFB   A9H            ;
0B49 8F                     DEFB   8FH            ;
0B4A 8C                     DEFB   8CH            ;
0B4B AE                     DEFB   AEH            ;
0B4C AC                     DEFB   ACH            ;
0B4D 9B                     DEFB   9BH            ;
0B4E A0                     DEFB   A0H            ;
0B4F 99                     DEFB   99H            ;
0B50 BC                     DEFB   BCH            ; {
0B51 B8                     DEFB   B8H
0B52                 ; C0 - CF;
```

```
**  Z80 ASSEMBLER SB-7201  <1Z-013A>  PAGE 43          04.07.83

01 0B52 40                       DEFB    40H              ; I
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                 ;  D0 - 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                 ;  E0 - 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                 ;  F0 - 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
```

```
0B8B 7D                        DEFB   7DH
0B8C 44                        DEFB   44H
0B8D 1B                        DEFB   1BH
0B8E 58                        DEFB   58H
0B8F 79                        DEFB   79H
0B90 42                        DEFB   42H
0B91 60                        DEFB   60H
0B92                  ;
0B92                  ;
0B92                  ;    FLASHING DATA SAVE
0B92                  ;
0B92                  ?SAVE:   ENT
0B92 219211                    LD     HL,FLSDT
0B95 36EF                      LD     (HL),EFH         ; NOMAL CURSOR
0B97 3A7011                    LD     A,(KANAF)
0B9A 0F                        RRCA
0B9B 3803                      JR     C,SV0-2          ; GRAPH MODE
0B9D 0F                        RRCA
0B9E 3002                      JR     NC,SV0           ; NORMAL MODE
0BA0 36FF                      LD     (HL),FFH         ; GRAPH CURSOR
0BA2                  SV0:     ENT
0BA2 7E                        LD     A,(HL)
0BA3 F5                        PUSH   AF
0BA4 CDB10F                    CALL   ?PONT            ; FLASING POSITION
0BA7 7E                        LD     A,(HL)
0BA8 328E11                    LD     (FLASH),A
0BAB F1                        POP    AF
0BAC 77                        LD     (HL),A
0BAD AF                        XOR    A
0BAE 2100E0                    LD     HL,KEYPA
0BB1 77                        LD     (HL),A
0BB2 2F                        CPL
0BB3 77                        LD     (HL),A
0BB4 C9                        RET
0BB5                  SV1:     ENT
0BB5 3643                      LD     (HL),43H         ; KANA CURSOR
0BB7 18E9                      JR     SV0
0BB9                  ;
0BB9                  ;ORG 0BB9H;?ADCN
0BB9                  ;
0BB9                  ;
0BB9                  ;    ASCII TO DISPLAY CODE CONVERTE
0BB9                  ;
0BB9                  ;        IN   ACC:ASCII
0BB9                  ;        EXIT ACC:DISPLAY CODE
0BB9                  ;
0BB9                  ?ADCN:   ENT
0BB9 C5                        PUSH   BC
0BBA E5                        PUSH   HL
0BBB 21920A                    LD     HL,ATBL
0BBE 4F                        LD     C,A
0BBF 0600                      LD     B,0
0BC1 09                        ADD    HL,BC
0BC2 7E                        LD     A,(HL)
0BC3 181B                      JR     DACN3
0BC5                  ;
0BC5 56312E30         VRNS:    DEFM   "V1.0A"          ; VERSION MANAGEMENT
0BC9 41
0BCA 0D                        DEFB   0DH
0BCB                           DEFS   +3
```

```
** Z8( ASSEMBLER SB-7201  <1Z-013A>  PAGE 45              04.07.83
OBCE                      ;
OBCE                      ;
OBCE                      ;ORG OBCEH;?DACN
OBCE                      ;
OBCE                      ;    DISPLAY CODE TO ASCII CONV.   ;
OBCE                      ;
OBCE                      ;      IN   ACC = DISPLAY CODE
OBCE                      ;      EXIT ACC = ASCII
OBCE                      ;
OBCE                      ?DACN:  ENT
OBCE C5                           PUSH   BC
OBCF E5                           PUSH   HL
OBDO D5                           PUSH   DE
OBD1 21920A                       LD     HL,ATBL
OBD4 54                           LD     D,H
OBD5 5D                           LD     E,L
OBD6 010001                       LD     BC,0100H
OBD9 EDB1                         CPIR
OBDB 2806                         JR     Z,DACN1
OBDD 3EFO                         LD     A,FOH
OBDF                      DACN2:  ENT
OBDF D1                           POP    DE
OBEO                      DACN3:  ENT
OBEO E1                           POP    HL
OBE1 C1                           POP    BC
OBE2 C9                           RET
OBE3                      ;
OBE3                      DACN1:  ENT
OBE3 B7                           OR     A
OBE4 2B                           DEC    HL
OBE5 ED52                         SBC    HL,DE
OBE7 7D                           LD     A,L
OBE8 18F5                         JR     DACN2
OBEA                      ;
OBEA                      ;
OBEA                      ;
OBEA                      ;    KEY MATRIX TO DISPLAY CODE TABL
OBEA                      ;
OBEA                      KTBL:   ENT
OBEA                      ;SO   00 - 07  ;
OBEA BF                           DEFB   BFH             ; SPARE
OBEB CA                           DEFB   CAH             ; GRAPH
OBEC 58                           DEFB   58H             ; ↓
OBED C9                           DEFB   C9H             ; ALPHA
OBEE FO                           DEFB   FOH             ; NO
OBEF 2C                           DEFB   2CH             ; ;
OBFO 4F                           DEFB   4FH             ; :
OBF1 CD                           DEFB   CDH             ; CR
OBF2                      ;S1   08 - OF  ;
OBF2 19                           DEFB   19H             ; Y
OBF3 1A                           DEFB   1AH             ; Z
OBF4 55                           DEFB   55H             ; @
OBF5 52                           DEFB   52H             ; [
OBF6 54                           DEFB   54H             ; ]
OBF7 FO                           DEFB   FOH             ; NULL
OBF8 FO                           DEFB   FOH             ; NULL
OBF9 FO                           DEFB   FOH             ; NULL
OBFA                      ;S2   0 - 17   ;
OBFA 11                           DEFB   11H             ; Q
OBFB 12                           DEFB   12H             ; R
```

```
0BFC 13                      DEFB    13H            ; S
0BFD 14                      DEFB    14H            ; T
0BFE 15                      DEFB    15H            ; U
0BFF 16                      DEFB    16H            ; V
0C00 17                      DEFB    17H            ; W
0C01 18                      DEFB    18H            ; X
0C02                 ;S3   18 - 1F   ;
0C02 09                      DEFB    09H            ; I
0C03 0A                      DEFB    0AH            ; J
0C04 0B                      DEFB    0BH            ; K
0C05 0C                      DEFB    0CH            ; L
0C06 0D                      DEFB    0DH            ; M
0C07 0E                      DEFB    0EH            ; N
0C08 0F                      DEFB    0FH            ; O
0C09 10                      DEFB    10H            ; P
0C0A                 ;S4  20 - 27   ;
0C0A 01                      DEFB    01H            ; A
0C0B 02                      DEFB    02H            ; B
0C0C 03                      DEFB    03H            ; C
0C0D 04                      DEFB    04H            ; D
0C0E 05                      DEFB    05H            ; E
0C0F 06                      DEFB    06H            ; F
0C10 07                      DEFB    07H            ; G
0C11 08                      DEFB    08H            ; H
0C12                 ;S5  28 - 2F   ;
0C12 21                      DEFB    21H            ; 1
0C13 22                      DEFB    22H            ; 2
0C14 23                      DEFB    23H            ; 3
0C15 24                      DEFB    24H            ; 4
0C16 25                      DEFB    25H            ; 5
0C17 26                      DEFB    26H            ; 6
0C18 27                      DEFB    27H            ; 7
0C19 28                      DEFB    2SH            ; 8
0C1A                 ;S6   30 - 37   ;
0C1A 59                      DEFB    59H            ; \
0C1B 50                      DEFB    50H            ; ↑
0C1C 2A                      DEFB    2AH            ; -
0C1D 00                      DEFB    00H            ; SPACE
0C1E 20                      DEFB    20H            ; 0
0C1F 29                      DEFB    29H            ; 9
0C20 2F                      DEFB    2FH            ; ,
0C21 2E                      DEFB    2EH            ; .
0C22                 ;S7   38 - 3F   ;
0C22 C8                      DEFB    C8H            ; INST.
0C23 C7                      DEFB    C7H            ; DEL.
0C24 C2                      DEFB    C2H            ; CURSOR UP
0C25 C1                      DEFB    C1H            ; CURSOR DOWN
0C26 C3                      DEFB    C3H            ; CURSOR RIGHT
0C27 C4                      DEFB    C4H            ; CURSOR LEFT
0C28 49                      DEFB    49H            ; ?
0C29 2D                      DEFB    2DH            ; /
0C2A                 ;
0C2A                 ;    KTBL SHIFT ON
0C2A                 ;
0C2A                 KTBLS:  ENT
0C2A                 ;S0    00-07
0C2A BF                      DEFB    BFH            ; SPARE
0C2B CA                      DEFB    CAH            ; GRAPH
0C2C 1B                      DEFB    1BH            ; POND
0C2D C9                      DEFB    C9H            ; ALPHA
```

```
** Z80 ASSEMBLER SB-7201  <1Z-013A>  PAGE 47          04.07.83
0C2E F0                      DEFB   F0H             ; NO
0C2F 6A                      DEFB   6AH             ; +
0C30 6B                      DEFB   6BH             ; *
0C31 CD                      DEFB   CDH             ; CR
0C32               ;S1   08-0F
0C32 99                      DEFB   99H             ; y
0C33 9A                      DEFB   9AH             ; z
0C34 A4                      DEFB   A4H             ; `
0C35 BC                      DEFB   BCH             ; {
0C36 40                      DEFB   40H             ; }
0C37 F0                      DEFB   F0H             ; NULL
0C38 F0                      DEFB   F0H             ; NULL
0C39 F0                      DEFB   F0H             ; NULL
0C3A               ;S2   10-17
0C3A 91                      DEFB   91H             ; q
0C3B 92                      DEFB   92H             ; r
0C3C 93                      DEFB   93H             ; s
0C3D 94                      DEFB   94H             ; t
0C3E 95                      DEFB   95H             ; u
0C3F 96                      DEFB   96H             ; v
0C40 97                      DEFB   97H             ; w
0C41 98                      DEFB   98H             ; x
0C42               ;S3   18-1F
0C42 89                      DEFB   89H             ; i
0C43 8A                      DEFB   8AH             ; j
0C44 8B                      DEFB   8BH             ; k
0C45 8C                      DEFB   8CH             ; l
0C46 8D                      DEFB   8DH             ; m
0C47 8E                      DEFB   8EH             ; n
0C48 8F                      DEFB   8FH             ; o
0C49 90                      DEFB   90H             ; p
0C4A               ;S4   20-27
0C4A 81                      DEFB   81H             ; a
0C4B 82                      DEFB   82H             ; b
0C4C 83                      DEFB   83H             ; c
0C4D 84                      DEFB   84H             ; d
0C4E 85                      DEFB   85H             ; e
0C4F 86                      DEFB   86H             ; f
0C50 87                      DEFB   87H             ; g
0C51 88                      DEFB   88H             ; h
0C52               ;S5   28-2F
0C52 61                      DEFB   61H             ; !
0C53 62                      DEFB   62H             ; "
0C54 63                      DEFB   63H             ; #
0C55 64                      DEFB   64H             ; $
0C56 65                      DEFB   65H             ; %
0C57 66                      DEFB   66H             ; &
0C58 67                      DEFB   67H             ; ´
0C59 68                      DEFB   68H             ; (
0C5A               ;S6   30-37
0C5A 80                      DEFB   80H             ; \
0C5B A5                      DEFB   A5H             ; POND MARK
0C5C 2B                      DEFB   2BH             ; YEN
0C5D 00                      DEFB   00H             ; SPACE
0C5E 60                      DEFB   60H             ; π
0C5F 69                      DEFB   69H             ; )
0C60 51                      DEFB   51H             ; <
0C61 57                      DEFB   57H             ; >
0C62               ;S7      38-3F
0C62 C6                      DEFB   C6H             ; CLR
```

```
0C63 C5                 DEFB   C5H          ; HOME
0C64 C2                 DEFB   C2H          ; CURSOR UP
0C65 C1                 DEFB   C1H          ; CURSOR DOWN
0C66 C3                 DEFB   C3H          ; CURSOR RIGHT
0C67 C4                 DEFB   C4H          ; CURSOR LEFT
0C68 5A                 DEFB   5AH          ; ⇨
0C69 45                 DEFB   45H          ; ⇦
0C6A            ;
0C6A            ;   GRAPHIC
0C6A            ;
0C6A            KTBLGS: ENT
0C6A            ;S0   00-07
0C6A BF                 DEFB   BFH          ; SPARE
0C6B F0                 DEFB   F0H          ; GRAPH BUT NULL
0C6C E5                 DEFB   E5H          ; #↓
0C6D C9                 DEFB   C9H          ; ALPHA
0C6E F0                 DEFB   F0H          ; NO
0C6F 42                 DEFB   42H          ; #;
0C70 B6                 DEFB   B6H          ; #:
0C71 CD                 DEFB   CDH          ; CR
0C72            ;S1   08-0F
0C72 75                 DEFB   75H          ; #Y
0C73 76                 DEFB   76H          ; #Z
0C74 B2                 DEFB   B2H          ; #@
0C75 D8                 DEFB   D8H          ; #[
0C76 4E                 DEFB   4EH          ; #]
0C77 F0                 DEFB   F0H          ; #NULL←
0C78 F0                 DEFB   F0H          ; #NULL
0C79 F0                 DEFB   F0H          ; #NULL
0C7A            ;S2   10-17
0C7A 3C                 DEFB   3CH          ; #Q
0C7B 30                 DEFB   30H          ; #R
0C7C 44                 DEFB   44H          ; #S
0C7D 71                 DEFB   71H          ; #T
0C7E 79                 DEFB   79H          ; #U
0C7F DA                 DEFB   DAH          ; #V
0C80 38                 DEFB   38H          ; #W
0C81 6D                 DEFB   6DH          ; #X
0C82            ;S3   18-1F
0C82 7D                 DEFB   7DH          ; #I
0C83 5C                 DEFB   5CH          ; #J
0C84 5B                 DEFB   5BH          ; #K
0C85 B4                 DEFB   B4H          ; #L
0CS6 1C                 DEFB   1CH          ; #M
0C87 32                 DEFB   32H          ; #N
0C88 B0                 DEFB   B0H          ; #O
0C89 D6                 DEFB   D6H          ; #P
0C8A            ;S4   20-27
0C8A 53                 DEFB   53H          ; #A
0C8B 6F                 DEFB   6FH          ; #B
0C8C DE                 DEFB   DEH          ; #C
0C8D 47                 DEFB   47H          ; #D
0C8E 34                 DEFB   34H          ; #E
0C8F 4A                 DEFB   4AH          ; #F
0C90 4B                 DEFB   4BH          ; #G
0C91 72                 DEFB   72H          ; #H
0C92            ;S5   28-2F
0C92 37                 DEFB   37H          ; #1
0C93 3E                 DEFB   3EH          ; #2
0C94 7F                 DEFB   7FH          ; #3
```

```
0C95 7B               DEFB    7BH          ; #4
0C96 3A               DEFB    3AH          ; #5
0C97 5E               DEFB    5EH          ; #6
0C98 1F               DEFB    1FH          ; #7
0C99 BD               DEFB    BDH          ; #8
0C9A          ;S6   30-3F
0C9A D4               DEFB    D4H          ; #YEN
0C9B 9E               DEFB    9EH          ; #+
0C9C D2               DEFB    D2H          ; #-
0C9D 00               DEFB    00H          ; SPACE
0C9E 9C               DEFB    9CH          ; #0
0C9F A1               DEFB    A1H          ; #9
0CA0 CA               DEFB    CAH          ; #,
0CA1 B8               DEFB    B8H          ; #.
0CA2          ;S7    38-3F
0CA2 C8               DEFB    C8H          ; INST
0CA3 C7               DEFB    C7H          ; DEL.
0CA4 C2               DEFB    C2H          ; CURSOR UP
0CA5 C1               DEFB    C1H          ; CURSOR DOWN
0CA6 C3               DEFB    C3H          ; CURSOR R1GHT
0CA7 C4               DEFB    C4H          ; CURSOR LEFT
0CA8 BA               DEFB    BAH          ; #?
0CA9 DB               DEFB    DBH          ; #/
0CAA          ;
0CAA          ;    CONTROL CODE
0CAA          ;
0CAA          KTBLC:  ENT
0CAA          ;S0     00-07N
0CAA F0               DEFB    F0H          ;
0CAB F0               DEFB    F0H
0CAC F0               DEFB    F0H          ; ↑
0CAD F0               DEFB    F0H
0CAE F0               DEFB    F0H
0CAF F0               DEFB    F0H
0CB0 F0               DEFB    F0H
0CB1 F0               DEFB    F0H
0CB2          ;S1     08-0F
0CB2 F0               DEFB    F0H          ; ↑Y   E3
0CB3 5A               DEFB    5AH          ; ↑Z   E4 (CHECKER)
0CB4 F0               DEFB    F0H          ; ↑@
0CB5 F0               DEFB    F0H          ; ↑[   E5
0CB6 F0               DEFB    F0H          ; ↑]   E7
0CB7 F0               DEFB    F0H
0CB8 F0               DEFB    F0H
0CB9 F0               DEFB    F0H
0CBA          ;S2     10-17
0CBA C1               DEFB    C1H          ; ↑Q
0CBB C2               DEFB    C2H          ; ↑R
0CBC C3               DEFB    C3H          ; ↑S
0CBD C4               DEFB    C4H          ; ↑T
0CBE C5               DEFB    C5H          ; ↑U
0CBF C6               DEFB    C6H          ; ↑V
0CC0 F0               DEFB    F0H          ; ↑W   E1
0CC1 F0               DEFB    F0H          ; ↑X   E2
0CC2          ;S3     18-1F
0CC2 F0               DEFB    F0H          ; ↑I   F9
0CC3 F0               DEFB    F0H          ; ↑J   FA
0CC4 F0               DEFB    F0H          ; ↑K   FB
0CC5 F0               DEFB    F0H          ; ↑L   FC
0CC6 F0               DEFB    F0H          ; ↑M   FD
```

```
0CC7 F0                   DEFB   F0H          ; ↑N  FE
0CC8 F0                   DEFB   F0H          ; ↑O  FF
0CC9 F0                   DEFB   F0H          ; ↑P  E0
0CCA            ;S4       20-27
0CCA F0                   DEFB   F0H          ; ↑A  F1
0CCB F0                   DEFB   F0H          ; ↑B  F2
0CCC F0                   DEFB   F0H          ; ↑C  F3
0CCD F0                   DEFB   F0H          ; ↑D  F4
0CCE F0                   DEFB   F0H          ; ↑E  F5
0CCF F0                   DEFB   F0H          ; ↑F  F6
0CD0 F0                   DEFB   F0H          ; ↑G  F7
0CD1 F0                   DEFB   F0H          ; ↑H  F8
0CD2            ;S5       28-2F
0CD2 F0                   DEFB   F0H
0CD3 F0                   DEFB   F0H
0CD4 F0                   DEFB   F0H
0CD5 F0                   DEFB   F0H
0CD6 F0                   DEFB   F0H
0CD7 F0                   DEFB   F0H
0CD8 F0                   DEFB   F0H
0CD9 F0                   DEFB   F0H
0CDA            ;S6       30-37
0CDA F0                   DEFB   F0H          ; ↑YEN E6
0CDB F0                   DEFB   F0H
0CDC F0                   DEFB   F0H
0CDD F0                   DEFB   F0H
0CDE F0                   DEFB   F0H
0CDF F0                   DEFB   F0H          ; ↑, EF
0CE0 F0                   DEFB   F0H
0CE1            ;S7          38-3F
0CE1 F0                   DEFB   F0H
0CE2 F0                   DEFB   F0H
0CE3 F0                   DEFB   F0H
0CE4 F0                   DEFB   F0H
0CE5 F0                   DEFB   F0H
0CE6 F0                   DEFB   F0H
0CE7 F0                   DEFB   F0H
0CE8 F0                   DEFB   F0H          ; ↑/ EE
0CE9            ;
0CE9            ;    KANA
0CE9            ;
0CE9            KTBLG:    ENT
0CE9            ;S0          00-07
0CE9 BF                   DEFB   BFH          ; SPARE
0CEA F0                   DEFB   F0H          ; GRAPH BUT NULL
0CEB CF                   DEFB   CFH          ; NIKO WH.
0CEC C9                   DEFB   C9H          ; ALPHA
0CED F0                   DEFB   F0H          ; NO
0CEE B5                   DEFB   B5H          ; MO
0CEF 4D                   DEFB   4DH          ; DAKU TEN
0CF0 CD                   DEFB   CDH          ; CR
0CF1            ;S1       08-0F
0CF1 35                   DEFB   35H          ; HA
0CF2 77                   DEFB   77H          ; TA
0CF3 D7                   DEFB   D7H          ; WA
0CF4 B3                   DEFB   B3H          ; YO
0CF5 B7                   DEFB   B7H          ; HANDAKU
0CF6 F0                   DEFB   F0H
0CF7 F0                   DEFB   F0H
0CF8 F0                   DEFB   F0H
```

```
** Z80 ASSEMBLER SB-7201  <1Z-013A>  PAGE 51           04.07.83
OCF9                 ;S2     10-17
OCF9 7C                     DEFB    7CH            ; KA
OCFA 70                     DEFB    70H            ; KE
OCFB 41                     DEFB    41H            ; SHI
OCFC 31                     DEFB    31H            ; KO
OCFD 39                     DEFB    39H            ; HI
OCFE A6                     DEFB    A6H            ; TE
OCFF 78                     DEFB    78H            ; KI
OD00 DD                     DEFB    DDH            ; CHI
OD01                 .S3     18-1F
OD01 3D                     DEFB    3DH            ; FU
OD02 5D                     DEFB    5DH            ; MI
OD03 6C                     DEFB    6CH            ; MU
OD04 56                     DEFB    56H            ; ME
OD05 1D                     DEFB    1DH            ; RHI
OD06 33                     DEFB    33H            ; RA
OD07 D5                     DEFB    D5H            ; HE
OD08 B1                     DEFB    B1H            ; HO
OD09                 ;S4     20-27
OD09 46                     DEFB    46H            ; SA
ODOA 6E                     DEFB    6EH            ; TO
ODOB D9                     DEFB    D9H            ; THU
ODOC 48                     DEFB    48H            ; SU
ODOD 74                     DEFB    74H            ; KU
ODOE 43                     DEFB    43H            ; SE
ODOF 4C                     DEFB    4CH            ; SO
OD10 73                     DEFB    73H            ; MA
OD11                 ;S5     28-2F
OD11 3F                     DEFB    3FH            ; A
OD12 36                     DEFB    36H            ; I
OD13 7E                     DEFB    7EH            ; U
OD14 3B                     DEFB    3BH            ; E
OD15 7A                     DEFB    7AH            ; O
OD16 1E                     DEFB    1EH            ; NA
OD17 5F                     DEFB    5FH            ; NI
OD18 A2                     DEFB    A2H            ; NU
OD19                 ;S6     30-37
OD19 D3                     DEFB    D3H            ; YO
OD1A 9F                     DEFB    9FH            ; YU
OD1B D1                     DEFB    D1H            ; YA
OD1C 00                     DEFB    00H            ; SPACE
OD1D 9D                     DEFB    9DH            ; NO
OD1E A3                     DEFB    A3H            ; NE
OD1F D0                     DEFB    D0H            ; RU
OD20 B9                     DEFB    B9H            ; RE
OD21                 ;S7     38-3F
OD21 C6                     DEFB    C6H            ; ?CLR ⊛
OD22 C5                     DEFB    C5H            ; ?HOME ⊡
OD23 C2                     DEFB    C2H            ; ?CURSOR UP
OD24 C1                     DEFB    C1H            ; ?CURSOR DOWN
OD25 C3                     DEFB    C3H            ; ?CURSOR RIGHT
OD26 C4                     DEFB    C4H            ; ?CURSOR LEFT
OD27 BB                     DEFB    BBH            ; DASH
OD28 BE                     DEFB    BEH            ; RO
OD29                 ;
OD29                 ;    MEMORY DUMP
OD29                 ;       COMMAND 'D'
OD29                 ;
OD29                 DUMP:  ENT
OD29 CD3D01                 CALL    HEXIY          ; START ADR.
```

```
0D2C CDA602             CALL   .4DE
0D2F E5                 PUSH   HL
0D30 CD1004             CALL   HLHEX          ; END ADR.
0D33 D1                 POP    DE
0D34 3852               JR     C,DUM1         ; DATA ER. THEN
0D36 EB                 EX     DE,HL
0D37 0608       DUM3:   LD     B,08H          ; DISP 8BYTES
0D39 0E17               LD     C,23           ; CHA. PRINT BIAS
0D3B CDFA05             CALL   NLPHL          ; NEWLINE PRINT
0D3E CDB103     DUM2:   CALL   SPHEX          ; SP. PRT.+ACC PRT.
0D41 23                 INC    HL
0D42 F5                 PUSH   AF
0D43 3A7111             LD     A,(DSPXY)      ; DISPLAY POINT
0D46 81                 ADD    A,C
0D47 327111             LD     (DSPXY),A      ; X AXIS.=X+Creg
0D4A F1                 POP    AF
0D4B FE20               CP     20H
0D4D 3002               JR     NC,+4
0D4F 3E2E               LD     A,2EH          ; '.'
0D51 CDB90B             CALL   ?ADCN          ; ASCII TO DSPLAY CODE
0D54 CD6C09             CALL   PRNT3
0D57 3A7111             LD     A,(DSPXY)
0D5A 0C                 INC    C
0D5B 91                 SUB    C              ; ASCII DSP POSITION
0D5C 327111             LD     (DSPXY),A
0D5F 0D                 DEC    C
0D60 0D                 DEC    C
0D61 0D                 DEC    C
0D62 E5                 PUSH   HL
0D63 ED52               SBC    HL,DE
0D65 E1                 POP    HL
0D66 281D               JR     Z,DUM1-3
0D68 3EF8               LD     A,F8H
0D6A 3200E0             LD     (KEYPA),A
0D6D 00                 NOP
0D6E 3A01E0             LD     A,(KEYPB)
0D71 FEFE               CP     FEH            ; SHIFT KEY ?
0D73 2003               JR     NZ,+5
0D75 CDA60D             CALL   ?BLNK          ; 64MSEC DELAY
0D78 10C4               DJNZ   DUM2
0D7A CDCA08             CALL   ?KEY           ; STOP DISP
0D7D B7                 OR     A
0D7E 28FA               JR     Z,-4           ; SPACE KEY THEN STOP
0D80 CD320A             CALL   ?BRK           ; BREAK IN ?
0D83 20B2               JR     NZ,DUM3
0D85 C3AD00             JP     ST1            ; COMMAND IN !
0D88 21A000     DUM1:   LD     HL,160         ; 20*8 BYTE
0D8B 19                 ADD    HL,DE
0D8C 18A8               JR     DUM3-1
0D8E            ;
0D8E            ;
0D8E            ;
0D8E            ;
0D8E                    DEFS   +24
0DA6            ;
0DA6            ;
0DA6            ;
0DA6            ;ORG 0DA6H;?BLNK
0DA6            ;
0DA6            ;
```

```
0DA6              ;    V-BLANK CHECK ;
0DA6              ;
0DA6              ?BLNK:  ENT
0DA6 F5                   PUSH   AF
0DA7 3A02E0               LD     A,(KEYPC)         ; V-BLNK
0DAA 07                   RLCA
0DAB 30FA                 JR     NC,-4
0DAD 3A02E0               LD     A,(KEYPC)
0DB0 07                   RLCA
0DB1 38FA                 JR     C,-4
0DB3 F1                   POP    AF
0DB4 C9                   RET
0DB5              ;
0DB5              ;ORG 0DB5H;?DSF
0DB5              ;
0DB5              ;
0DB5              ;
0DB5              ;  DISPLAY ON POINTER  ;
0DB5              ;
0DB5              ;    ACC = DISPLAY CODE
0DB5              ;    EXCEPT F0H
0DB5              ;
0DB5              ?DSP:   ENT
0DB5 F5                   PUSH   AF
0DB6 C5                   PUSH   BC
0DB7 D5                   PUSH   DE
0DB8 E5                   PUSH   HL
0DB9              DSP01:  ENT
0DB9 CDB10F               CALL   ?PONT             ; DSPLAY POSITION
0DBC 77                   LD     (HL),A
0DBD 2A7111               LD     HL,(DSPXY)
0DC0 7D                   LD     A,L
0DC1 FE27                 CP     +39
0DC3 200B                 JR     NZ,DSP04
0DC5 CDF302               CALL   .MANG
0DC8 3806                 JR     C,DSP04
0DCA EB                   EX     DE,HL
0DCB 3601                 LD     (HL),+1           ; LOGICAL 1ST COLUMN
0DCD 23                   INC    HL
0DCE 3600                 LD     (HL),0            ; LOGICAL 2ND COLUMN
0DD0              DSP04:  ENT
0DD0 3EC3                 LD     A,C3H             ; CURSL
0DD2 180C                 JR     ?DPCT+4
0DD4              ;
0DD4              ;
0DD4              ;
0DD4              ;
0DD4              ;  GRAPHIC STATUS CHECK
0DD4              ;
0DD4 3A7011       GRSTAS: LD     A,(KANAF)
0DD7 FE01                 CP     01H
0DD9 3ECA                 LD     A,CAH
0DDB C9                   RET
0DDC              ;
0DDC              ;
0DDC              ;
0DDC              ;
0DDC              ;
0DDC              ;
0DDC              ;ORG 0DDCH;?DPCT
```

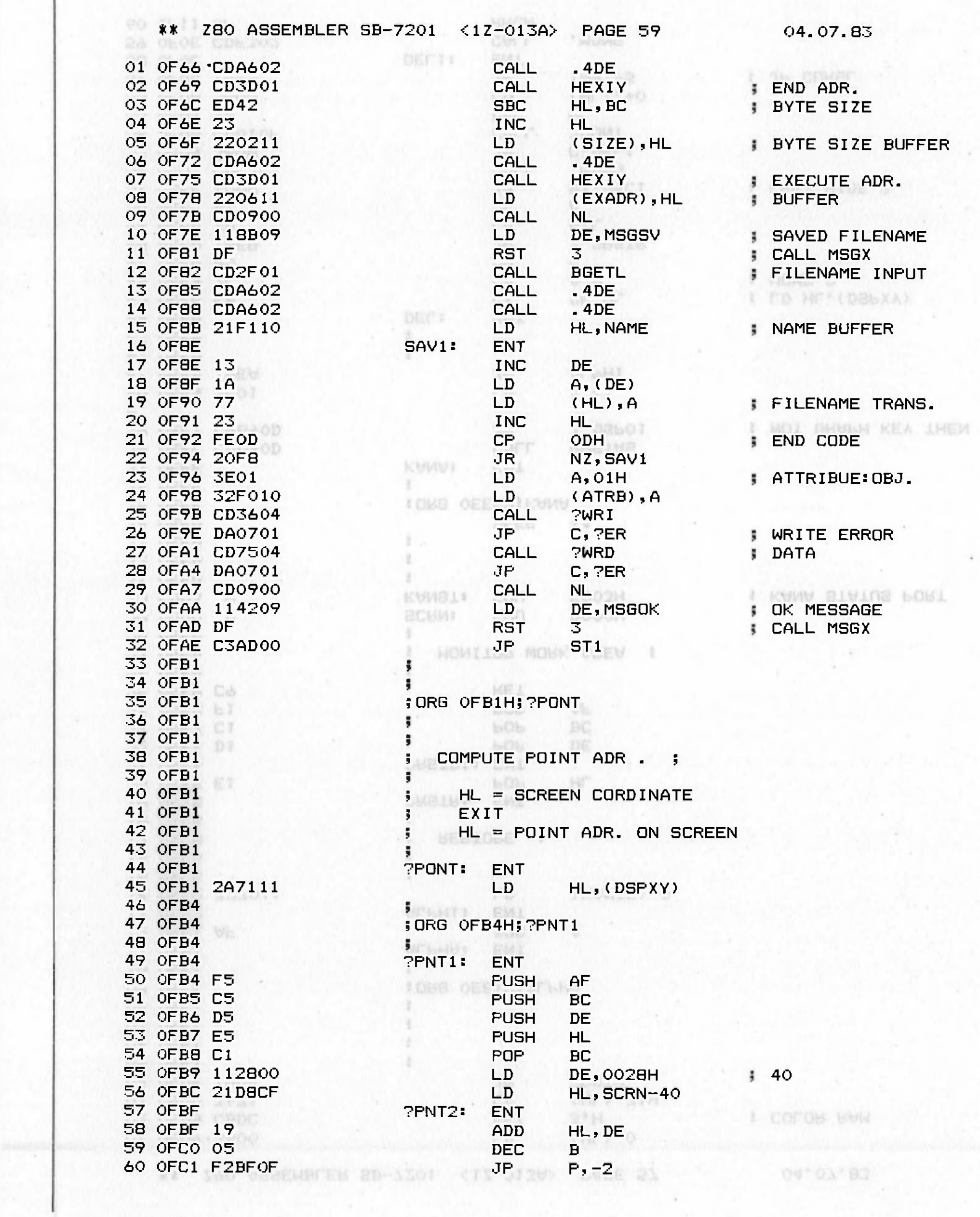
```
01 0DDC                  ;
02 0DDC                  ;
03 0DDC                  ;   DISPLAY CONTROL   ;
04 0DDC                  ;
05 0DDC                  ;     ACC = CONTROL CODE
06 0DDC                  ;
07 0DDC                  ?DPCT:   ENT
08 0DDC F5                        PUSH    AF
09 0DDD C5                        PUSH    BC
10 0DDE D5                        PUSH    DE
11 0DDF E5                        PUSH    HL
12 0DE0 47                        LD      B,A
13 0DE1 E6F0                      AND     F0H
14 0DE3 FEC0                      CP      C0H
15 0DE5 201B                      JR      NZ,CURS5
16 0DE7 A8                        XOR     B
17 0DE8 07                        RLCA
18 0DE9 4F                        LD      C,A
19 0DEA 0600                      LD      B,+0
20 0DEC 21AA0E                    LD      HL,CTBL          ; PAGE MODE1
21 0DEF 09                        ADD     HL,BC
22 0DF0 5E                        LD      E,(HL)
23 0DF1 23                        INC     HL
24 0DF2 56                        LD      D,(HL)
25 0DF3 2A7111                    LD      HL,(DSPXY)
26 0DF6 EB                        EX      DE,HL
27 0DF7 E9                        JP      (HL)
28 0DF8                  ;
29 0DF8                  ;
30 0DF8                  ;
31 0DF8                  CURSD:   ENT
32 0DF8 EB                        EX      DE,HL            ; LD HL,(DSPXY)
33 0DF9 7C                        LD      A,H
34 0DFA FE18                      CP      +24
35 0DFC 2825                      JR      Z,CURS4
36 0DFE 24                        INC     H
37 0DFF                  CURS1:   ENT
38 0DFF
39 0DFF
40 0DFF
41 0DFF                  CURS3:   ENT
42 0DFF 227111                    LD      (DSPXY),HL
43 0E02 C3E50E           CURS5:   JP      ?RSTR
44 0E05                  ;
45 0E05                  CURSU:   ENT
46 0E05 EB                        EX      DE,HL            ; LD HL,(DSPXY)
47 0E06 7C                        LD      A,H
48 0E07 B7                        OR      A
49 0E08 28F8                      JR      Z,CURS5
50 0E0A 25                        DEÇ     H
51 0E0B                  CURSU1:  ENT
52 0E0B 18F2                      JR      CURS3
53 0E0D                  CURSR:   ENT
54 0E0D EB                        EX      DE,HL            ; LD HL,(DSPXY)
55 0E0E 7D                        LD      A,L
56 0E0F FE27                      CP      +39
57 0E11 3003                      JR      NC,CURS2
58 0E13 2C                        INC     L
59 0E14 18E9                      JR      CURS3
60 0E16                  CURS2:   ENT
```

```
** Z80 ASSEMBLER SB-7201 <1Z-013A> PAGE 55            04.07.83
0E16 2E00                LD     L,+0
0E18 24                  INC    H
0E19 7C                  LD     A,H
0E1A FE19                CP     +25
0E1C 38E1                JR     C,CURS1
0E1E 2618                LD     H,+24
0E20 227111              LD     (DSPXY),HL
0E23            CURS4:   ENT
0E23 1848                JR     SCROL
0E25            ;
0E25            CURSL:   ENT
0E25 EB                  EX     DE,HL             ; LD HL,(DSPXY)
0E26 7D                  LD     A,L
0E27 B7                  OR     A
0E28 2803                JR     Z,+5
0E2A 2D                  DEC    L
0E2B 18D2                JR     CURS3
0E2D 2E27                LD     L,+39
0E2F 25                  DEC    H
0E30 F20B0E              JP     P,CURSU1
0E33 2600                LD     H,0
0E35 227111              LD     (DSPXY),HL
0E38 18C8                JR     CURS5
0E3A            ;
0E3A            CLRS:    ENT
0E3A 217311              LD     HL,MANG
0E3D 061B                LD     B,27
0E3F CDD80F              CALL   ?CLER
0E42 2100D0              LD     HL,D000H          ; SCRN TOP
0E45 CDD409              CALL   #CLR08
0E48 3E71                LD     A,71H             ; COLOR DATA
0E4A CDD509              CALL   #CLR8             ; D800H-DFFFH CLR.
0E4D            HOME:    ENT
0E4D 210000              LD     HL,0              ; DSPXY:0 X=0,Y=0
0E50 18AD                JR     CURS3
0E52            ;
0E52                     DEFS   +8
0E5A            ;
0E5A            ;    CR
0E5A            ;
0E5A            CR:      ENT
0E5A CDF302              CALL   .MANG
0E5D 0F                  RRCA
0E5E 30B6                JR     NC,CURS2
0E60 2E00                LD     L,0
0E62 24                  INC    H
0E63 FE18                CP     +24
0E65 2803                JR     Z,CR1
0E67 24                  INC    H
0E68 1895                JR     CURS1
0E6A            CR1:     ENT
0E6A 227111              LD     (DSPXY),HL
0E6D            ;
0E6D            ;    SCROL
0E6D            ;
0E6D            SCROL:   ENT
0E6D 01C003              LD     BC,03C0H
0E70 1100D0              LD     DE,SCRN           ; TOP OF $CRT ADR.
0E73 2128D0              LD     HL,SCRN+40        ; 1 COLUMN
0E76 C5                  PUSH   BC                ; 1000 STORE
```

```
0E77 EDB0                 LDIR
0E79 C1                   POP     BC
0E7A D5                   PUSH    DE
0E7B 1100D8               LD      DE,SCRN+800H      ; COLOR RAM SCROLL
0E7E 2128D8               LD      HL,SCRN+828H      ; SCROLL TOP + 40
0ES1 EDB0                 LDIR
0E83 0628                 LD      B,40              ; ONE LINE
0E85 EB                   EX      DE,HL
0E86 3E71                 LD      A,71H             ; COLOR RAM INITIAL DATA
0E88 CDDD0F               CALL    ?DINT
0E8B E1                   POP     HL
0ESC 0628                 LD      B,40
0E8E CDD80F               CALL    ?CLER             ; LAST LINE CLEAR
0E91 011A00               LD      BC,26             ; ROW NUMBER +1
0E94 117311               LD      DE,MANG           ; LOGICAL MANAGEMENT
0E97 217411               LD      HL,MANG+1
0E9A EDB0                 LDIR
0E9C 3600                 LD      (HL),0
0E9E 3A7311               LD      A,(MANG)
0EA1 B7                   OR      A
0EA2 2841                 JR      Z,?RSTR
0EA4 217211               LD      HL,DSPXY+1
0EA7 35                   DEC     (HL)
0EA8 18C3                 JR      SCROL
0EAA            ;
0EAA            ;    CONTROL CODE TABLE
0EAA            ;
0EAA            CTBL:     ENT
0EAA 6D0E                 DEFW    SCROL             ; SCROLLING
0EAC F80D                 DEFW    CURSD             ; CURSOR
0EAE 050E                 DEFW    CURSU
0EB0 0D0E                 DEFW    CURSR
0EB2 250E                 DEFW    CURSL
0EB4 4D0E                 DEFW    HOME
0EB6 3A0E                 DEFW    CLRS
0EB8 F80E                 DEFW    DEL
0EBA 380F                 DEFW    INST
0EBC E10E                 DEFW    ALPHA
0EBE EE0E                 DEFW    KANA
0EC0 E50E                 DEFW    ?RSTR
0EC2 E50E                 DEFW    ?RSTR
0EC4 5A0E                 DEFW    CR
0EC6 E50E                 DEFW    ?RSTR
0EC8 E50E                 DEFW    ?RSTR
0ECA            ;
0ECA            ;
0ECA            ;
0ECA            ;    INST BYPASS
0ECA            ;
0ECA CBDC       INST2:    SET     3,H               ; COLOR RAM
0ECC 7E                   LD      A,(HL)            ; FROM
0ECD 23                   INC     HL
0ECE 77                   LD      (HL),A            ; TO
0ECF 2B                   DEC     HL                ; ADR ADJ.
0ED0 CB9C                 RES     3,H
0ED2 EDA8                 LDD                       ; CHA. TRNS.
0ED4 79                   LD      A,C
0ED5 B0                   OR      B                 ; BC=0 ?
0ED6 20F2                 JR      NZ,INST2
0ED8 EB                   EX      DE,HL
```

```
0ED9 3600                     LD     (HL),0
0EDB CBDC                     SET    3,H              ; COLOR RAM
0EDD 3671                     LD     (HL),71H
0EDF 1804                     JR     ?RSTR
0EE1                  ;
0EE1                  ;
0EE1                  ;
0EE1                  ;
0EE1                  ;ORG 0EE1H;ALPHA
0EE1                  ;
0EE1                  ALPHA:  ENT
0EE1 AF                       XOR    A
0EE2                  ALPH1:  ENT
0EE2 327011                   LD     (KANAF),A
0EE5                  ;
0EE5                  ;
0EE5                  ;  RESTORE  ;
0EE5                  ;
0EE5                  ?RSTR:  ENT
0EE5 E1                       POP    HL
0EE6                  ?RSTR1: ENT
0EE6 D1                       POP    DE
0EE7 C1                       POP    BC
0EE8 F1                       POP    AF
0EE9 C9                       RET
0EEA                  ;
0EEA                  ;  MONITOR WORK AREA  ;
0EEA                  ;
D000  P               SCRN:   EQU    D000H
E003  P               KANST:  EQU    E003H            ; KANA STATUS PORT
0EEA                  ;
0EEA                  ;
0EEA                  ;
0EEA                          DEFS   +4
0EEE                  ;ORG 0EEEH;KANA
0EEE                  ;
0EEE                  KANA:   ENT
0EEE CDD40D                   CALL   GRSTAS
0EF1 CAB90D                   JP     Z,DSP01          ; NOT GRAPH KEY THEN JU
P
0EF4 3E01                     LD     A,+1
0EF6 18EA                     JR     ALPH1
0EF8                  ;
0EF8                  ;
0EF8                  DEL:    ENT
0EF8 EB                       EX     DE,HL            ; LD HL,(DSPXY)
0EF9 7C                       LD     A,H              ; HOME ?
0EFA B5                       OR     L
0EFB 28E8                     JR     Z,?RSTR
0EFD 7D                       LD     A,L
0EFE B7                       OR     A
0EFF 200D                     JR     NZ,DEL1          ; LEFT SIDE ?
0F01 CDF302                   CALL   .MANG
0F04 3808                     JR     C,DEL1
0F06 CDB10F                   CALL   ?PONT
0F09 2B                       DEC    HL
0F0A 3600                     LD     (HL),+0
0F0C 1825                     JR     INST-5           ; JP CURSL
0F0E                  DEL1:   ENT
0F0E CDF302                   CALL   .MANG
0F11 0F                       RRCA
```

```
0F12 5E28                    LD      A,40
0F14 3001                    JR      NC,+3
0F16 07                      RLCA                        ; ACC=80
0F17 95                      SUB     L
0F18 47                      LD      B,A                 ; TRNS. BYTE
0F19 CDB10F                  CALL    ?PONT
0F1C 7E             DEL2:    LD      A,(HL)              ; CHA. FROM ADR
0F1D 2B                      DEC     HL
0F1E 77                      LD      (HL),A              ; TO
0F1F 23                      INC     HL
0F20 CBDC                    SET     3,H                 ; COLOR RAM
0F22 7E                      LD      A,(HL)
0F23 2B                      DEC     HL
0F24 77                      LD      (HL),A
0F25 CB9C                    RES     3,H                 ; CHA.
0F27 23                      INC     HL
0F28 23                      INC     HL                  ; NEXT
0F29 10F1                    DJNZ    DEL2
0F2B 2B                      DEC     HL                  ; ADR.ADJUST
0F2C 3600                    LD      (HL),0
0F2E CBDC                    SET     3,H
0F30 217100                  LD      HL,71H              ; BLUE + WHITE
0F33 3EC4                    LD      A,C4H               ; JP CURSL
0F35 C3E00D                  JP      ?DPCT+4
0F38                ;
0F38                INST:    ENT
0F38 CDF302                  CALL    .MANB
0F3B 0F                      RRCA
0F3C 2E27                    LD      L,+39
0F3E 7D                      LD      A,L
0F3F 3001                    JR      NC,+3
0F41 24                      INC     H
0F42 CDB40F                  CALL    ?PNT1
0F45 E5                      PUSH    HL
0F46 2A7111                  LD      HL,(DSPXY)
0F49 3002                    JR      NC,+4
0F4B 3E4F                    LD      A,+79
0F4D 95                      SUB     L
0F4E 0600                    LD      B,0
0F50 4F                      LD      C,A
0F51 D1                      POP     DE
0F52 2891                    JR      Z,?RSTR
0F54 1A                      LD      A,(DE)
0F55 B7                      OR      A
0F56 208D                    JR      NZ,?RSTR
0F58 62                      LD      H,D                 ; HL←DE
0F59 6B                      LD      L,E
0F5A 2B                      DEC     HL
0F5B C3CA0E                  JP      INST2               ; JUMP NEXT (BYPASS)
0F5E                ;
0F5E                ;
0F5E                ;    PROGRAM SAVE
0F5E                ;
0F5E                ;      CMD. 'S'
0F5E                ;
0F5E                SAVE:    ENT
0F5E CD3D01                  CALL    HEXIY               ; START ADR.
0F61 220411                  LD      (DTADR),HL          ; DATA ADR. BUFFER
0F64 44                      LD      B,H
0F65 4D                      LD      C,L
```

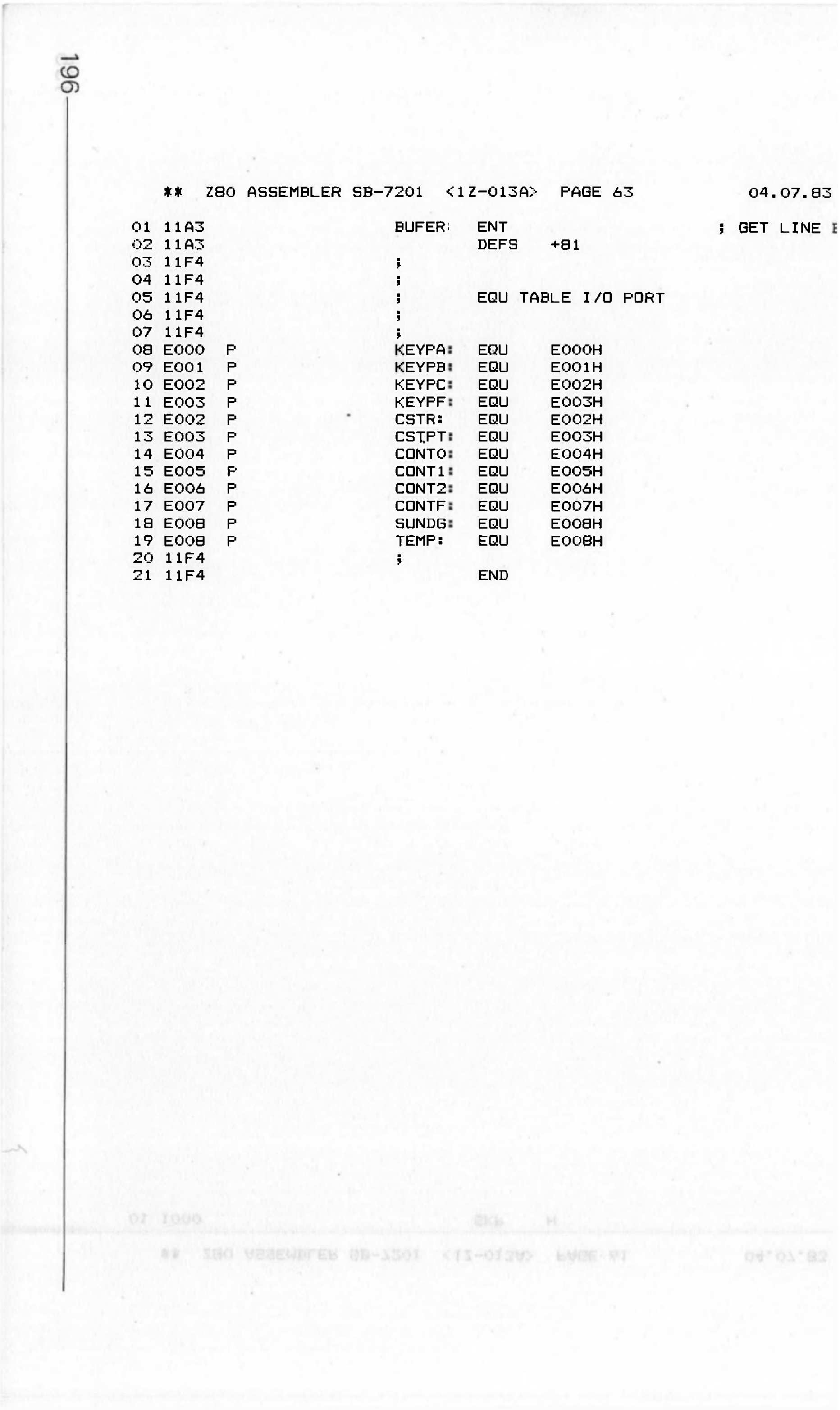


```
**  Z80 ASSEMBLER SB-7201  <1Z-013A>  PAGE 59              04.07.83
0F66 CDA602                 CALL    .4DE
0F69 CD3D01                 CALL    HEXIY             ; END ADR.
0F6C ED42                   SBC     HL,BC             ; BYTE SIZE
0F6E 23                     INC     HL
0F6F 220211                 LD      (SIZE),HL         ; BYTE SIZE BUFFER
0F72 CDA602                 CALL    .4DE
0F75 CD3D01                 CALL    HEXIY             ; EXECUTE ADR.
0F78 220611                 LD      (EXADR),HL        ; BUFFER
0F7B CD0900                 CALL    NL
0F7E 118B09                 LD      DE,MSGSV          ; SAVED FILENAME
0F81 DF                     RST     3                 ; CALL MSGX
0F82 CD2F01                 CALL    BGETL             ; FILENAME INPUT
0F85 CDA602                 CALL    .4DE
0F88 CDA602                 CALL    .4DE
0F8B 21F110                 LD      HL,NAME           ; NAME BUFFER
0F8E                SAV1:   ENT
0F8E 13                     INC     DE
0F8F 1A                     LD      A,(DE)
0F90 77                     LD      (HL),A            ; FILENAME TRANS.
0F91 23                     INC     HL
0F92 FE0D                   CP      0DH               ; END CODE
0F94 20F8                   JR      NZ,SAV1
0F96 3E01                   LD      A,01H             ; ATTRIBUE:OBJ.
0F98 32F010                 LD      (ATRB),A
0F9B CD3604                 CALL    ?WRI
0F9E DA0701                 JP      C,?ER             ; WRITE ERROR
0FA1 CD7504                 CALL    ?WRD              ; DATA
0FA4 DA0701                 JP      C,?ER
0FA7 CD0900                 CALL    NL
0FAA 114209                 LD      DE,MSGOK          ; OK MESSAGE
0FAD DF                     RST     3                 ; CALL MSGX
0FAE C3AD00                 JP      ST1
0FB1                ;
0FB1                ;
0FB1                ;ORG 0FB1H;?PONT
0FB1                ;
0FB1                ;
0FB1                ;  COMPUTE POINT ADR .  ;
0FB1                ;
0FB1                ;    HL = SCREEN CORDINATE
0FB1                ;    EXIT
0FB1                ;    HL = POINT ADR. ON SCREEN
0FB1                ;
0FB1                ?PONT:  ENT
0FB1 2A7111                 LD      HL,(DSPXY)
0FB4                ;
0FB4                ;ORG 0FB4H;?PNT1
0FB4                ;
0FB4                ?PNT1:  ENT
0FB4 F5                     PUSH    AF
0FB5 C5                     PUSH    BC
0FB6 D5                     PUSH    DE
0FB7 E5                     PUSH    HL
0FB8 C1                     POP     BC
0FB9 112800                 LD      DE,0028H          ; 40
0FBC 21D8CF                 LD      HL,SCRN-40
0FBF                ?PNT2:  ENT
0FBF 19                     ADD     HL,DE
0FC0 05                     DEC     B
0FC1 F2BF0F                 JP      P,-2
```

```
0FC4 0600                        LD     B,+0
0FC6 09                          ADD    HL,BC
0FC7 D1                          POP    DE
0FC8 C1                          POP    BC
0FC9 F1                          POP    AF
0FCA C9                          RET
0FCB                  ;
0FCB                  ;   VERIFYING
0FCB                  ;
0FCB                  ;     COMMAND  /V/
0FCB                  ;
0FCB                  VRFY:   ENT
0FCB CD8805                      CALL   ?VRFY
0FCE DA0701                      JP     C,?ER
0FD1 114209                      LD     DE,MSGOK
0FD4 DF                          RST    3
0FD5 C3AD00                      JP     ST1
0FD8                  ;
0FD8                  ;
0FD8                  ;
0FD8                  ;ORG 0FD8H;?CLER
0FD8                  ;
0FD8                  ;
0FD8                  ;  CLER  ;
0FD8                  ;    B=SIZE
0FD8                  ;    HL-LOW ADR.
0FD8                  ;
0FD8                  ?CLER:  ENT
0FD8 AF                          XOR    A
0FD9 1802                        JR     +4
0FDB                  ?CLRFF: ENT
0FDB 3EFF                        LD     A,FFH
0FDD                  ?DINT:  ENT
0FDD 77                          LD     (HL),A
0FDE 23                          INC    HL
0FDF 10FC                        DJNZ   -2
0FE1 C9                          RET
0FE2                  ;
0FE2                  ;
0FE2                  ;   GAP CHECK
0FE2                  ;
0FE2                  GAPCK:  ENT
0FE2 C5                          PUSH   BC
0FE3 D5                          PUSH   DE
0FE4 E5                          PUSH   HL
0FE5 0101E0                      LD     BC,KEYPB
0FE8 1102E0                      LD     DE,CSTR
0FEB                  GAPCK1: ENT
0FEB 2664                        LD     H,100
0FED                  GAPCK2: ENT
0FED CD0106                      CALL   EDGE
0FF0 380B                        JR     C,GAPCK3
0FF2 CD4A0A                      CALL   DLY3            ; CALL DLY2*3
0FF5 1A                          LD     A,(DE)
0FF6 E620                        AND    20H
0FF8 20F1                        JR     NZ,GAPCK1
0FFA 25                          DEC    H
0FFB 20F0                        JR     NZ,GAPCK2
0FFD                  GAPCK3: ENT
0FFD C39B06                      JP     RET3
```

```
01 1000                             SKP     H
```

```
 **  Z80 ASSEMBLER SB-7201 <1Z-013A>  PAGE 62            04.07.83
1000               ;
1000               ;
1000               ;    MONITOR WORK AREA    ;
1000               ;       (MZ-700)          ;
1000               ;
1000               ;
1000               ;
10F0                          ORG      10F0H
10F0               SP:        ENT
10F0               IBUFE:     ENT                    ; TAPE BUFFER(128B)
10F0               ATRB:      ENT                    ; ATTRIBUTE
10F0                          DEFS     +1
10F1               NAME:      ENT                    ; FILE NAME
10F1                          DEFS     +17
1102               SIZE:      ENT                    ; BYTE SIZE
1102                          DEFS     +2
1104               DTADR:     ENT                    ; DATA ADR
1104                          DEFS     +2
1106               EXADR:     ENT                    ; EXECUTION ADR
1106                          DEFS     +2
1108               COMNT:     ENT                    ; COMMENT
1108                          DEFS     104
1170               KANAF:     ENT                    ; KANA FLAG
1170                          DEFS     +1
1171               DSPXY:     ENT                    ; DISPLAY CO-ORDINATES
1171                          DEFS     +2
1173               MANG:      ENT                    ; COLOUMN MANAGEMENT
1173                          DEFS     +27
118E               FLASH:     ENT                    ; FLASHING DATA
118E                          DEFS     +1
118F               FLPST:     ENT                    ; FLASSING POSITION
118F                          DEFS     +2
1191               FLSST:     ENT                    ; FLASING STATUS
1191                          DEFS     +1
1192               FLSDT:     ENT                    ; CURSOR DATA
1192                          DEFS     +1
1193               STRGF:     ENT                    ; STRING FLAG
1193                          DEFS     +1
1194               DPRNT:     ENT                    ; TAB COUNTER
1194                          DEFS     +1
1195               TMCNT:     ENT                    ; TAPE MARK COUNTER
1195                          DEFS     +2
1197               SUMDT:     ENT                    ; CHECK SUM DATA
1197                          DEFS     +2
1199               CSMDT:     ENT                    ; FOR COMPARE SUM DATA
1199                          DEFS     +2
119B               AMPM:      ENT                    ; AMPM DATA
119B                          DEFS     +1
119C               TIMFG:     ENT                    ; TIME FLAG
119C                          DEFS     +1
119D               SWRK:      ENT                    ; KEY SOUND FLAG
119D                          DEFS     +1
119E               TEMPW:     ENT                    ; TEMPO WORK
119E                          DEFS     +1
119F               ONTYO:     ENT                    ; ONTYO WORK
119F                          DEFS     +1
11A0               OCTV:      ENT                    ; OCTAVE WORK
11A0                          DEFS     +1
11A1               RATIO:     ENT                    ; ONPU RATIO
11A1                          DEFS     +2
```



```
   **  Z80 ASSEMBLER SB-7201  <1Z-013A>  PAGE 63          04.07.83

01 11A3                   BUFER:   ENT                          ; GET LINE
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                CONT0:   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
```

```
#BRK   0BB8  #CLR08 09D4  #CLR8  09D5  $MCP   006B  ..LPT  017B
.4DE   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  ?DACN  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 05FE  ?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   04D8  ?RSTR  0EE5
?RSTR1 0EE6  ?SAVE  0B92  ?SWEP  0A50  ?TEMP  02E5  ?TMR1  0375
?TMR2  037F  ?TMRD  0358  ?TMS1  0331  ?TMS2  0344  ?TMST  0308
?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  BUFER  11A3  CKS1   0720
CKS2   072F  CKS3   0733  CKSUM  071A  CLEAR  09D8  CLEAR1 09DA
CLRS   0E3A  CMYO   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  0830  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  FLSST  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  HEXIY  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  KTBLS  0C2A
LETNL  0006  LLPT   0470  LOA0   0116  LOAD   0111  LONG   0A1A
LPRNT  018F  M#TBL  0284  MANG   1173  MCOR   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
ONTYO  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  PTSTO  015A  PTST1  0170  RATIO  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  02C8  SAV1   0F8E  SAVE   0F5E  SCRN   D000
SCROL  0E6D  SG     00F7  SHORT  0A01  SIZE   1102  SLPT   03D5
```

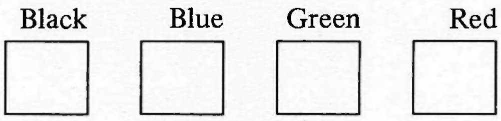
```
  **  Z80 ASSEMBLER SB-7201  <1Z-013A>  PAGE 65            04.07.83

SP      10F0  SPHEX   03B1  SS      00A2  STO     0070  ST1     00AD
ST2     00BB  START   004A  STRGF   1193  SUMDT   1197  SUNDG   E008
SVO     0BA2  SV1     0BB5  SWEPO   0A66  SWEPO1  0A64  SWEP2   0A7F
SWEP3   0A77  SWEP6   0A5F  SWEP9   0A73  SWRK    119D  TEMP    E00B
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    05BB  TVF3    05CC  TVRFY   05AD
VERFY   002D  VGOFF   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) + $(\text{ASCII})_2$ + $(\text{ASCII})_1$ + $(\text{ASCII})_0$ + ($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	Lp (p = 0 to 15)	Specifies the type of line (solid or dotted) and the dot pitch. p = 0 : solid line, p = 1 ~ 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, . . . xn, yn ($-999 \leqq x, y \leqq 999$)	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)	JΔx, Δy . . . Δxn, Δyn ($-999 \leqq \Delta x, \Delta y \leqq 999$)	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 ($-999 \leqq x, y \leqq 999$)	Lifts the pen and moves it to coordinates (x, y).
RELATIVE MOVE (RMOVE)	RΔx, Δy ($-999 \leqq \Delta x, \Delta y \leqq 999$)	Lifts the pen and moves it to relative coordinates (Δx, Δy).
COLOR CHANGE (PCOLOR)	Cn (n = 0 to 3)	Changes the pen color to n.
SCALE SET	Sn (n = 0 to 63)	Specifies the character scale.
ALPHA ROTATE	Qn (n = 0 to 3)	Specifies the direction in which characters are printed.
PRINT	P$c_1 c_2 c_3$ . . . cn ($n = \infty$)	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.

A. 6. 3. 3 Parameter specification

1. Leading blanks are ignored.
2. Any number preceded by " — " is treated as a negative number.
3. If the number of digits of a number exceeds 3, only the lower 3 digits are effective.
4. Each parameter is ended with "," or a CR code. If other than numbers are included in a parameter, subsequent characters are ignored until a comma or CR code is detected.

   Example) D␣␣ −135. 21, ……

   └—Ignored.—┘

A. 6. 3. 4 Abbreviated formats

1. Any command can be followed by a one-character command without entering a CR code.
   Ex) "HD100, 200" CR is effective and is the same as "H" CR "D100, 200" CR.
2. Any command can be followed by a command with one parameter by separating them with a comma ",".
   Ex) "L0, S1, Q0, C1, D100, 200" CR is effective.
3. A command with pairs of parameters must be terminated with a CR code.

4. 6. 3. 5 Data change due to mode switching

The following data changes when the printer is switched from the graphic mode to the text mode.

● X and Y coordinates

Y is set to 0 and the origin is placed at the left side of the printable area.

● Direction of characters

Q is set to 0.

● Character scale

Character scale is set to 1.

● The line type setting is not affected.

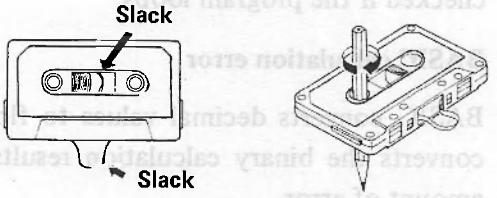
A. 7 Notes Concerning Operation

■ 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.

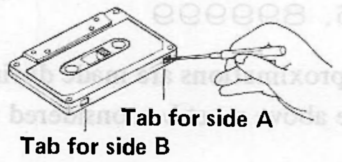


- **Protecting programs/data from accidental erasure**

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.

Remove record lock-out tab with a screwdriver.



■ 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.

■ **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 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 817.3-81Ø.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:)

```
1Ø A=1/1ØØ*1ØØ
2Ø IF A=1 THEN PRINT"TRUE" :GOTO 4Ø
3Ø PRINT "FALSE"
4Ø PRINT "A=";A
5Ø 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.

Copying/Debugging of MZ-700 Basic Interpreter

A. **Please follow the procedure below mentioned to copy the BASIC tape.**

1) Power on MZ-700 (→ monitor state)
2) Partial memory should be modified by the use of monitor command M (memory correction) as follows:

```
*MCF00
CF00    FF → CD
CF01    00 → 27
CF02    FF → 00
CF03    00 → 38
CF04    FF → 03
CF05    00 → CD
CF06    FF → 2A
CF07    00 → 00
CF08    FF → DA
CF09    00 → FE
CF0A    FF → 00
CF0B    00 → C3
CF0C    FF → AD
CF0D    00 → 00
CF0E    FF → CD
CF0F    00 → 27
CF10    FF → 00
CF11    00 → 38
CF12    FF → F5
CF13    00 → C3
CF14    FF → CB
CF15    00 → 0F
```

SHIFT + BREAK to be keyed in.

NOTE: The content of memory from CF00 to CF15 may not always be as above mentioned.

3) The cassette to be read (copyed from) should be set to the tape recorder.
4) Key in the monitor command J (Jump) as follows:

```
* JCF00   CR
⊥ PLAY
```

NOTE: If a button of the tape recorder is still pushed no play indication will appear.

5) Confirming the "⊥PLAY" indication above mentioned, push PLAY button and load the content of BASIC tape. On this occasion, no indication like FILE NAME, etc. will be shown. When ERROR occured, please restart from the item 1) again.
6) Set a new cassette to which the BASIC should be written into the recorder and execute REWIND .

7) Key in as follows:

```
⁎ J1108   [CR]
```

8) The monitor will be cleared and the following indication will appear:

```
S-BASICEX  SAVER   xx  ▚  xx  ▩
HIT  ANY  KEY?
```

9) Push any key.

```
⊥ Record Play
```

[STOP] button should be pushed beforehand.

10) Push [RECORD] button. The copy will start and the following indication will appear:

```
WRITING  S-BASIC
```

On the occasion of MZ-711, item 9) should be effectuated after setting the external tape recorder in recording state.

11) After the sound "Pit Pit", the copy will be terminated.
12) The monitor state will be recovered by pushing the rear RESET SW.
13) Rewind the tape and push [STOP] button.
14) Key in as follows:

```
⁎ JCF0E   [CR]
⊥ PLAY
```

15) Push [PLAY] button of the recorder and the "VERIFY" function will be executed. When successful verified, the indication of "OK!" will appear though no other indication like FILE NAME etc. will appear. When error occured, please restart from the item 4).
16) Please make sure to enable the write protection of the cassette by removing the nail.

B. The following procedure is requested to modify the content of BASIC interpreter.

a) Operate just as the case for copying mentioned in item 1) to 5).
b) Call up the address to be modified by using the monitor command M.
Ex. 8A in 1234H should be changed to 7A.

```
                     Key in
⁎M                   1234
1234       8A        7A  [CR]
1235       8A        [SHIFT] + [BREAK]
⁎
```

C. The operation from the item 6) onwards should be continued hereafter.

SHARP CORPORATION
OSAKA, JAPAN

Printed in Japan
Gedruckt in Japan
Imprimé au Japon

3H 5.305-I
TINSE1066ACZZ