SHARP

PERSONAL COMPUTER

MZ-80A

OWNER'S MANUAL

MACHINES SUPPLIED IN THE U.K. AND REPUBLIC OF IRELAND HAVE 48K BYTE RAM FITTED AS STANDARD

Personal Computer

MZ-80A

Owner's Manual

080311-250182

NOTICE

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

This manual is applicable to the SA-5510 BASIC interpreter used with the SHARP MZ-80A Personal Computer. The MZ-80A general-purpose personal computer is supported by system software which is filed in software packs (cassette tapes or diskettes).

All system software is subject to revision without prior notice, therefore, you are requested to pay special attention to file version numbers.

This manual has been carefully prepared and checked for completeness, accuracy and clarity. However, in the event that you should notice any errors or ambiguities, please feel free to contact your local Sharp representative for clarification.

Preface

This manual describes the Sharp MZ-80A personal computer. Read this manual thoroughly to become familiar with the operating procedures, BASIC language and precautions. This manual describes the MZ-80A and associated software.

Chapter 1 describes the features of the MZ-80A, general operating procedures–read these sections first, and language specifications and summary of the standard system software BASIC interpreter SA-5510. BASIC (an abbreviation for "Beginner's All-purpose Symbolic Instruction Code") was developed as an all purpose language to provide beginners with a means of easily programming computers to solve a diverse range of problems. Its simplicity and versatility make it well suited to personal programming applications. BASIC SA-5510 is an extended BASIC interpreter which enables the MZ-80A computer to be used to its fullest capacity.

Chapter 2 describes command and subroutines of the MONITOR SA-1510.

Chapter 3 describes the hardware. This information will be helpful to you if you intend to expand system.

Precautions

The MZ-80A is one of the finest personal computers in the world; its design incorporates all the technical knowledge accumulated by Sharp in its many years of experience in the electronics field. All units are thoroughly inspected prior to shipment so that each will operate normally when it is unpacked. However, be sure to check visually for any damage caused during transportation. If any damage is found or any parts are missing, contact your dealer immediately.

Observe the following guidelines to keep your set in optimum operating condition:

- Do not place the MZ-80A in locations where the temperature is extremely high or low or where it varies to a great extent. Avoid exposing the unit to direct sunlight, vibration or dust.
- Handle the power cable carefully to prevent it from being damaged. When removing it from the AC outlet, turn the power off first, then pull the plug (do not pull the cable).
- If the power switch is turned off then immediately turned on again, initialization may not be performed correctly. Allow a few moments after turning the power off before turning it on.
- The personal computer MZ-80A contains 32K byte RAM as standard equipment. When you use system software that requires the disk drive access (DISK BASIC, FDOS, etc.), it is necessary to expand the existing RAM area to 48K bytes.

For more detailed information, see Appendix 5.

IMPORTANT

For users in the United Kingdom:

The wires in the mains lead of this apparatus 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.

Contents

Your MZ-80A and BASIC Programming

Chapter 1

1.1 Profile of the MZ-80A

You must know the configuration of a computer to construct programs which can actually run on it. The more you know about the console, memory, processors, peripheral environment, and language processing programs, the more efficient and elaborate your programs will be because you can create your programs taking full advantage of the computer facilities. You can, however, acquire detailed such detailed knowledge only by accumulating experience in designing and running programs on a computer yourself.

This first section presents a profile of the SHARP MZ-80A personal computer to allow you to grasp an outline of its hardware configuration and basic operating procedures. In the next section, we will take our first steps in computer programming.

■ Profile

The MZ-80A is an integrated personal computer which made its debut in the fall of 1981. It is a completely new multi-purpose small computer designed with a wide range of future hardware and software applications in mind. Its greatest features are its high speed and ease of operation. When it was introduced, the MZ-80A was widely acclaimed as a system which would open a new dimension in computer programming.

Figure 1.1 is a simplified illustration of the hardware configuration of the MZ-80A. It consists of a storage unit (which stores programs and data), a central processing unit (which performs operations on data as directed by the programs in the storage unit and transfers the data to and from the storage unit), and several input/output units. The storage unit is divided into main memory, monitor program, and video RAM sections. The MZ-80A has 32 K bytes of RAM (read/write memory) in its main memory section. The main memory section can be expanded to 48 K bytes by incorporating an additional 16 K bytes of RAM. The input units include a keyboard and a cassette tape unit. The output units include CRT display, cassette tape, and audio output units.

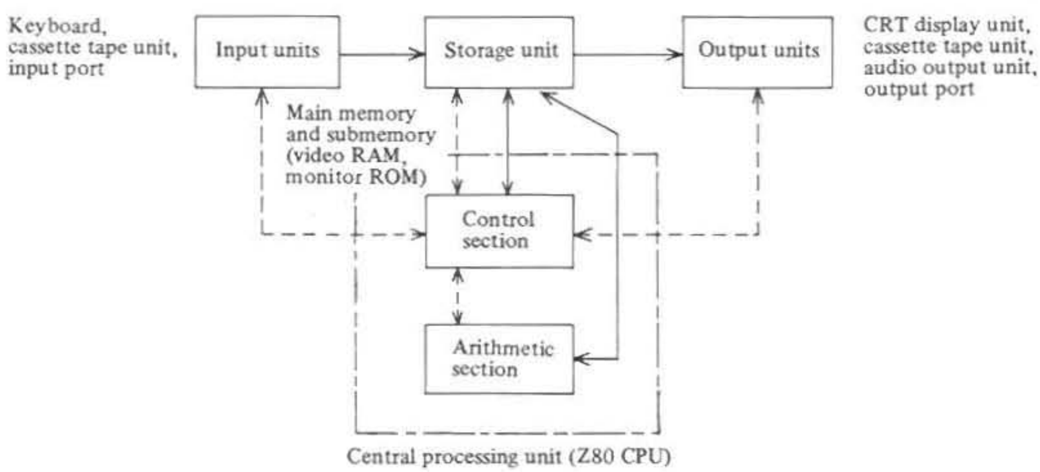


Figure 1.1 MZ-80A configuration

The central processing unit, which consists of control and arithmetic sections, performs active dynamically; it serves as the brain of the computer and controls it overall operation. Its operation, however, is made up of repetitions of the following simple operating sequence:

1. A data item containing an instruction is read from storage.
2. The instruction is executed.

In other words, logically speaking it is a collection of data items in the storage unit give instructions that cause the computer to operate in a dynamic manner. This collection of data items is called a program. It is, therefore, necessary to prepare a program to indicate the steps of a job and store it in the storage unit to cause the computer to perform the job.

Inside the computer, data and control signals are logically represented by binary numbers which are represented by the digits of 0 and 1. The number of digits of a binary number (i.e., a sequence of 0s and 1s) is counted in terms of bits. For example, the 8-bit binary number

0 0 1 1 0 1 0 1

is a data item which has a length of 8 bits (this is equivalent to 53 in decimal representation). Since bits are too small to be convenient for indicating the length of data, a unit called the "byte" is used to indicate a data item of 8 bits. One byte can represent up to 2^8 (= 256) different numbers.

The MZ-80A employs a Z80, a so-called 8-bit microprocessor (which process one byte of data at a time), as its central processing unit. Accordingly, programs which give instructions and data to be processed are all stored and transferred in byte units. Byte locations in the storage unit are designated by a 2-byte pointer in the central processing unit. With this 2-byte pointer, the Z80 can address up to 2^{16} (= 65536) locations. Since 2^{10} (= 1024) represents 1 K bytes, the Z80 is said to have an address space of 64 K bytes. As mentioned above, the MZ-80A main storage unit is made up of 48 K bytes, or 3/4 of the Z80 RAM (Random Access Memory) address space. RAM is a type of memory which can be freely read and written; on the other hand, ROM (Read Only Memory) can only be read.

The majority of special-purpose computers dedicated to automatic control systems and many personal computers have memories in which 1/3 to 1/2 or more of the memory space is composed of ROM for storage of control or system programs (e.g., BASIC interpreter programs). The use of RAM in the memory configuration of the MZ-80A is based on the premise that main memory should be freely available for a variety of uses. The MZ-80A stores all system programs in external files from which they are loaded into main memory by a monitor program.

The SA-5510 BASIC interpreter, one of the MZ-80A system programs, functions to translate BASIC source programs into machine code for execution.

The personal computer MZ-80A

1.2 Operating the MZ-80A

This section describes the constituent units of the MZ-80A and their functions.

■ Top view of the MZ-80A

Figure 1.2

■ Rear view of the MZ-80A

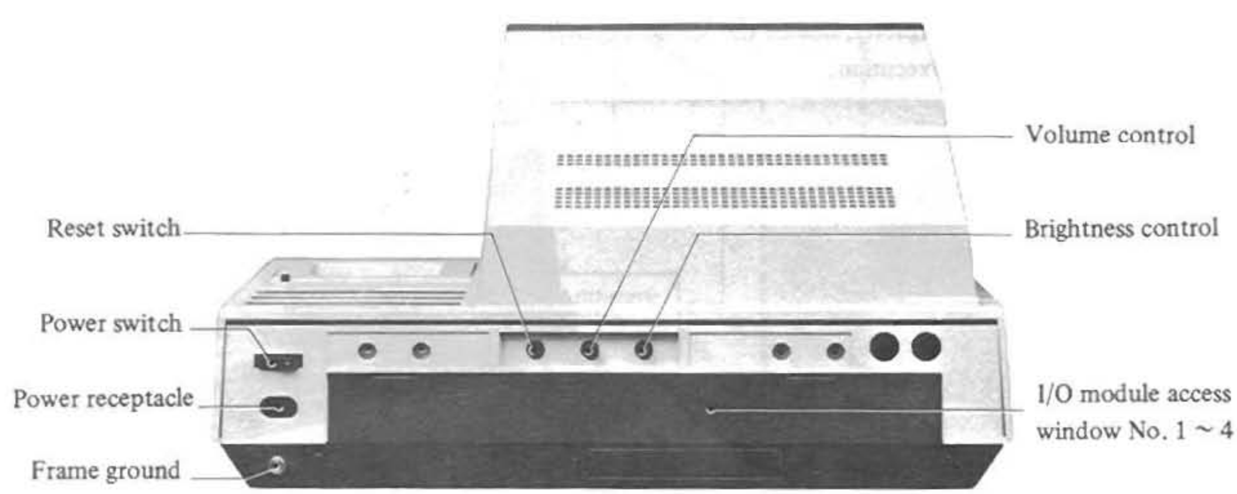


Figure 1.3

1.2.1 Activating system software

The MZ-80A personal computer is supported by system software which is filed in software packs.

BASIC SA-5510 is stored on a cassette tape file, and must undergo initial program loading whenever it is to be used. Loading is easily achieved.

First, turn on the power switch on the back of the MZ-80A. The Monitor program starts and the following message will be displayed on the CRT display.

```
＊ ＊ MONITOR SA-1510 ＊ ＊
＊ ▒
  ↑
  └─cursor flickers
```

Place the BASIC cassette file in the cassette tape deck and press the [L] key, then press the [CR] key. (L: Load)

The Monitor's program loader starts, and message "↓PLAY" is displayed. Press the [PLAY] button of the cassette tape deck.

The program loader loads the BASIC interpreter (photo at left of Figure 1.4), and upon completion of loading, the MZ-80A displays the message illustrated in the photo at right and the BASIC interpreter begins to operate.

The message "Ready" indicates that system control is at the BASIC command level and that the system is ready to accept any command.

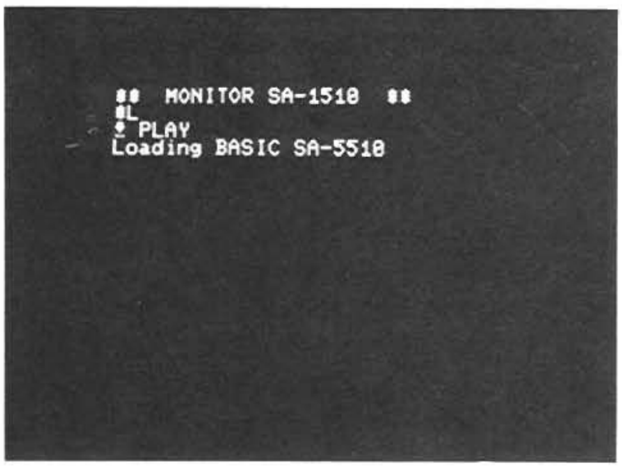
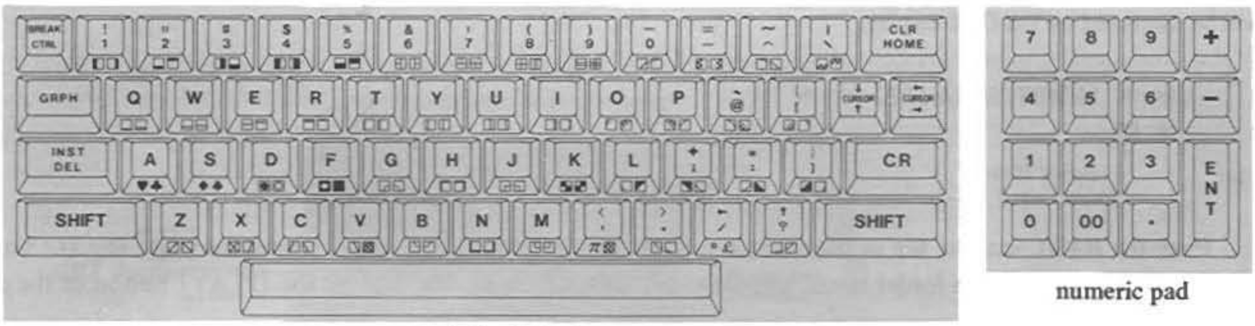


Figure 1.4

Please refer to the chapter 2 on activating system software from the diskette files and Monitor commands.

1.2.2 Keyboard

The keyboard of the MZ-80A is arranged as shown in Figure 1.5, and is divided into 2 areas; main keyboard and numeric pad.



main keyboard

numeric pad

Figure 1.5 The keyboard of the MZ-80A

The main keyboard (typewriter keyboard) conforms to ASCII standard and includes character keys and control keys (such as the carriage return key, the control key and the cursor control keys.)

The numeric pad is for entering numeric data and is similar to that of an ordinary electronic calculator.

The main keyboard has two operating modes;

[1] Normal mode
[2] Graphic mode

Keys provided on the main keyboard produce different characters according to operating mode, as shown in Figure 1.6.

normal mode	graphic mode
A	♥
a hold down SHIFT	♣ hold down SHIFT

Figure 1.6 Different characters of the A key

Note that the letter key normally produce capital letters. To enter lower case letters, hold down the SHIFT key then press the letter key—just opposite of an ordinary typewriter. The reason for this is that capital letters are generally easier to read on the screen, so most people prefer to write their programs in capital letters.

Figure 1.7 shows the control keys (the stippled keys).

Figure 1.7 Control keys

The functions of the control keys are explained below.

SHIFT : Similar to the shift key of an ordinary typewriter; when this key is depressed, the character keys and some control keys are shifted.

CR : Carriage return key. The ENT key has the same function as the CR key.

GRPH : If this key is pressed in the normal mode, the graphic mode is entered and the cursor pattern changes from " ▒ " to " ■ ", and vice versa.

INST DEL : DEL erases the character at the left of the cursor location, shifting all following characters of the string to the left one space. INST inserts a space where the cursor is located by shifting all following characters of the string to the right one space.

CLR HOME : HOME returns the cursor to the upper left hand corner of the display screen. CLR clears the display screen and also returns the cursor to the screen's upper left hand corner.
In the graphic mode, HOME produces the reverse character " H ", and CLR produces the reverse character " C ".

CURSOR CURSOR : Cursor control keys. Each key moves the cursor in the direction indicated by the arrow (normal position and shift position).
In the graphic mode, each key produces the reverse arrow; ↑ ↓ → ←.

BREAK CTRL : When this key is pressed with the SHIFT key depressed, a break code is generated, and halts execution of BASIC programs.

Figure 1.8 shows the CTRL key and some other keys (the stippled keys).

Figure 1.8 CTRL and some keys

The functions of these keys depressing the CTRL key are explained below.

CTRL + A : This locks the SHIFT key so that it does not need to be held down. Pressing these keys again or pressing the CR key releases the shift lock.

CTRL + E : This rolls down the listing of the CRT display.

CTRL + D : This rolls up the listing of the CRT display.

CTRL + Z : This generates the character "→". This character is used as a delimiter (PASCAL, FDOS, etc.)

CTRL + @ : This sets the character display mode to reverse mode. Pressing these keys again sets the character display mode to normal mode.

CTRL + { : This sets the V-RAM configuration to the MZ-80K mode.

CTRL + } : This sets the V-RAM configuration to the MZ-80A mode.

1.3 BASIC Operations for Programming

Now let's start our study of BASIC programming. Here, our purpose is to allow the beginner to gain familiarity with the basic elements of programs. In the first section, we will construct very short programs to illustrate fundamental concepts and learn about basic operations which are required during the course of BASIC programming. That is, we will learn:

1 How to construct a program.
2 How to run a program.
3 How to correct a program.
4 How to store a program (on cassette tape).
5 How to run a program stored in an external file.

1 Constructing a program

To have a computer do a job, it must be given sequence of instructions according to which it is to work. Determining the sequence of instructions, implementing them as a BASIC program, entering the program into the MZ-80A from the keyboard, and correcting the program afterwards are operations which are fundamental to program development. The problem is given below is a simple example of work to be done on a computer.

Example 0: Read two numeric data items from the keyboard, compute their sum, and display the result.

The sequence of instructions is, as indicated in the problem, "read two numeric data items from the keyboard," "compute their sum," and "display the result." These instructions are written in BASIC as follows:

```
10  INPUT A        }
20  INPUT B        } Read two numeric data items from the keyboard.
30  LET C = A + B ....... Compute their sum.
40  PRINT C ........... Display the result.
50  END ............. End.
```

On the first two lines, variables A and B are assigned two numeric values through the INPUT statement, which has the function of receiving data from the keyboard. On the next line, the sum of A and B is assigned to variable C. The content of C is shown on the display unit through the PRINT statement on the next line, which has the function of displaying data on the CRT display unit. Then the program ends. Although we explain these steps as if they were a matter of course, they are far from self-explanatory. Thus, it is here that we will begin our study.

There are two points to keep in mind in the above problem:

- A BASIC program is written using words such as INPUT, LET, PRINT, END, etc. Lines containing these words are called INPUT statements, PRINT statements, and so forth.
- Each line begins with a number such as 10.

In other words, a BASIC program is made up of statements beginning with a set of words (called reserved words) or their abbreviations, and numbers (called line numbers) which precede the statements. Although the above program has only five lines, it is a complete program. In fact, a single line can constitute a program if it contains a line number and a statement. Large programs have the same program elements as such a single line program.

The next step is to enter their program into the computer from the keyboard. This is not hard to do; you can enter it in the same way you type on a typewriter. You must take note, however, of the following:

- All variable names and words such as INPUT and PRINT must be entered in upper case letters. The MZ-80A keyboard prints upper case letters in the normal mode and lower case letters in the shift mode, so you need not press the SHIFT key (as with a typewriter) when keying upper case letters.
- Each line must be terminated by pressing the CR key (or ENT key on the numeric key pad). A line of data keyed in is not stored in memory as a program line until the CR key is pressed.

Now, key in the first line.

```
1 0   I N P U T   A  CR
```

The cursor on the screen will move to the beginning of the next line when the CR key is pressed. Enter the second and third lines in succession. The entire program is stored in memory when the END statement on line number 50 is entered, followed by pressing the CR key.

Now key in:

```
L I S T  CR
```

The listing of program input will appear on the screen. LIST is a command which displays the list of program lines stored in memory on the screen. It is called a command to distinguish it from statements (such as INPUT) which are used within the program.

2 Executing a program

To execute a program, give the RUN command to the computer. Key in:

```
R U N  CR
```

A " ? " mark will then appear on the next line and the cursor will flash. This means that the program execution has started and that the first INPUT statement is being executed. Key in, for example, the number 19 as the value of variable A. Entry of data during execution of the INPUT statement must also be terminated by pressing CR .

```
1 9  CR
```

It is convenient to use the ENT key, instead of the CR key, when entering numeric values from the numeric key pad.

The second INPUT statement is then executed and a " ? " mark again appears on the screen. Key in "81" as the value of variable B.

```
8 1  CR
```

The computer, on receiving the variable B value, performs computation and assignment operations as directed on line number 30, then displays the result

```
1 0 0
```

on the screen as directed by the PRINT statement on line number 40. Thus, we obtain the result of adding 19 + 81. The computer ends program execution when it encouters the END statement on line number 50, displays

```
Ready
```

on the screen and causes the cursor to start flashing again. The "Ready" message indicates that the computer is in a mode, called the command mode, in which no program is executed and commands are awaited. In the command mode, you can enter commands such as LIST and RUN or modify the program.

3 Correcting a program

The procedure for correcting or modifying a program is basically the same as the procedure for creating one. For example, to modify the above program so that the result of A – B is assigned to variable C and the content of C is displayed on the screen, it is necessary to key in

```
3 0   L E T   C = A – B  CR
```

When a line with the same line number as one of the old lines is entered, the old line is replaced by the new line.

A more convenient method, called screen editing, may be used when only portions of a line are to be changed, as in this example, where the plus sign is to be changed to a minus sign. With screen editing, all that is required is to move the cursor to the display position where a change is to be made using the cursor control keys and to overwrite the character(s). To terminate the editing session, press the [CR] key (The [CR] key may be pressed with the cursor in any position, as long as it is within a line). To insert or delete character(s), use the INSERT/DEL key. Run the program after modifying it.

[4] Storing a program

The programs stored in the computer main memory are lost when power to the computer is turned off. You must learn how to store programs in external files in order to execute or complete them later, or exchange them with friends who use the MZ-80A.

The cassette tape unit in the MZ-80A is an input/output device which is used not only for starting the BASIC interpreter, but for recording and reading programs and data. To record a program onto cassette tape, use the following procedure:

- Load a cassette tape in the unit. When recording at the beginning of the tape, rewind it by pressing the REW button before proceeding to the next step.
- Enter a SAVE command together with an appropriate program name.

The SAVE command causes the program in the computer to be saved on the cassette tape.

Now, let's record the above program (changed to a subtraction program through screen editing) on cassette tape. Name the program "Subtraction." After mounting a cassette tape on the MZ-80A, key in:

S A V E " S u b t r a c t i o n " [CR]

Then,

↓RECORD . PLAY

will appear on the screen. Press the RECORD and PLAY keys simultaneously, and

Writing "Subtraction"

will appear on the screen, indicating that the save operation is in progress.

The prompt "Ready" will again appear on the screen when the program has been saved.

[5] Reading a program from cassette tape

The LOAD command is used to read programs from cassette tape. To read the program "Subtraction," key in:

L O A D "S u b t r a c t i o n" [CR]

Figure 1.9 shows that, after the BASIC interpreter has been started, the program Subtraction has been read into the computer from the cassette tape by a LOAD command.

It also shows the messages Found and Loading, which are displayed in the course of program reading to indicate that the requested file has been found and that it is being read.

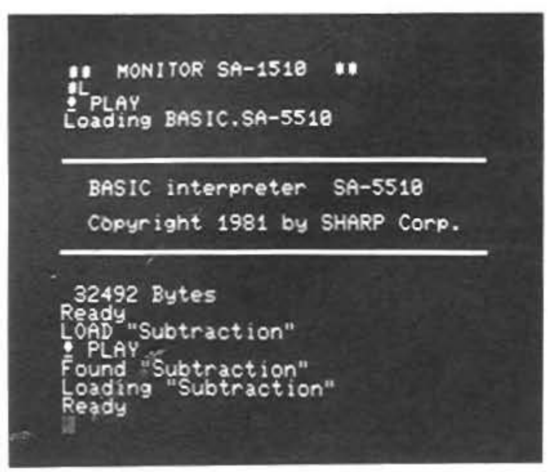


Figure 1.9 Loading a program

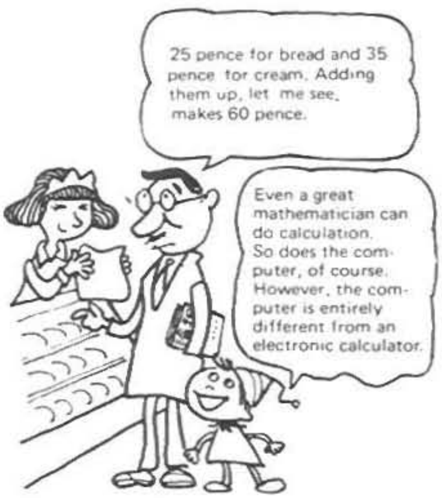
What is the Direct Mode?

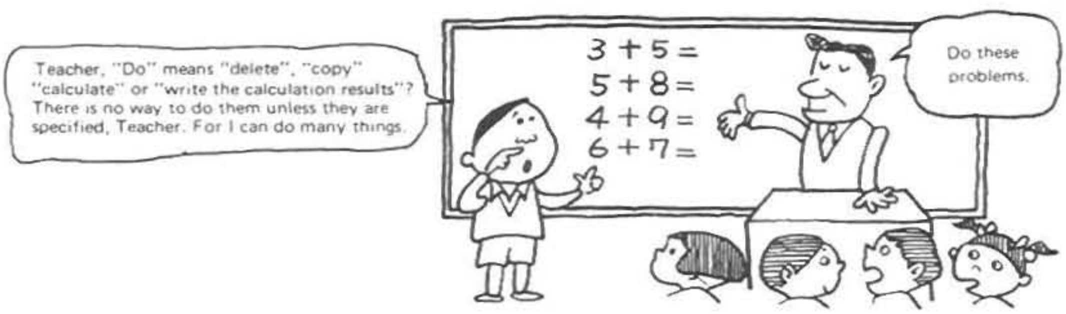
Using the computer like an electronic calculator is possible if required. This kind of operation is called "Direct Mode".

Like the electronic calculator, key-in 5 + 8 =.

To key-in the [+], press the key while holding down the [SHIFT] key.

In fact, however, the computer displays the characters on the CRT screen only as keyed-in, and of course, no calculation is executed even with the [CR] key depressed. Here lies the difference between your computer and the electronic calculator. Your computer requires an instruction of what should be done about 5 + 8 =.





PRINT

To use the computer in the same manner as the electronic calculator, the computation of 5 + 8 is required to be displayed on the CRT screen. For this, there is the PRINT command available as an instruction. Using this command, let's press the keys in the following order to transfer the instructions.

[P] [R] [I] [N] [T] [5] [+] [8] [CR]

As the keys are depressed, the characters below will be displayed on the CRT screen.

Ready .... Meaning "Go ahead with your work".
PRINT 5 + 8 .... Display the computation of 5 + 8, and with the [CR] key pressed indicating the end of a command.
13 .... This is the executed result of the command.
Ready .... What is to be done next?
▩ .... Cursor

The Four Arithmetic Operations

If you want to go on to multiplication and division, note that the computer uses signs slightly different from those of ordinary mathematics.

Multiplication sign	*
Division sign	/

Calculation with Parenthesis

The computer is capable of handling more complex calculations than an ordinary calculator. This is a calculation with parenthesis.

In case of ordinary mathematical operation, different signs of groupings are used to write as follows:

$3 \times 6\ [6 + 3\{\ 9 - 2\ (4 - 2) + 1\}\ ]$

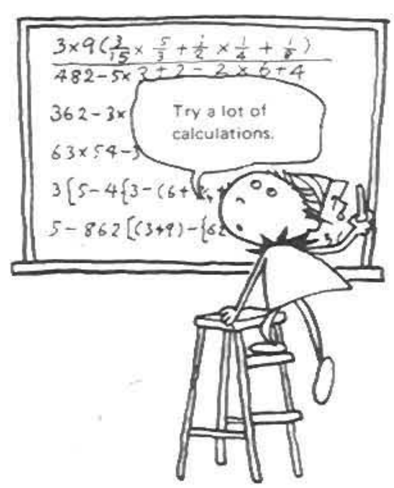
Whereas the parenthesis (   ) alone is used at all times with the computer.

3 * 6 * (6 + 3 * (9 – 2 * (4 – 2) + 1))

Even with the above, the computer never forgets the rule that computation in the inner signs of groupings be done first, and never makes any mistake.

Exercise

```
PRINT (6 + 4) / (6 – 4)
 5
PRINT 3 * (5 + 9 * (9 – 2) – 6 / (4 – 2) ) + 5
 200
PRINT (3 + 4) * (5 + 6)
 77
PRINT (10 + 20) / 6 * (2 + 3)
 25
PRINT (10 + 20) / (6 * (2 + 3) )
 1
```



String? Expression?

```
PRINT 3 + 5
```

With the above, pressing the CR key makes 8, doesn't it? Now, put the expression in quotation marks".

```
PRINT "3 + 5" and CR
3 + 5
```

Oh, the result is different. Try another one

```
PRINT "HELLO MY FRIEND" CR
HELLO MY FRIEND
```

As is clear from the above, the characters or symbols put between quotation marks " " are displayed as they are on the CRT screen.

The block of characters and/or symbols between the quotation marks is called a **string**.

PRINT "3 + 5"

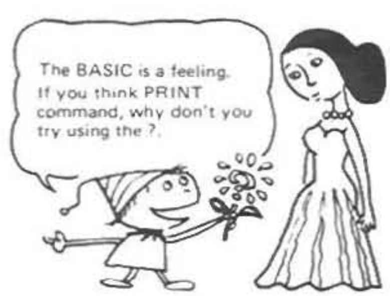
This is a string
put between quotation marks.

PRINT 3 + 5

This is an expression,
not a string.

It is necessary for you to know more about the strings. The free use of strings will double the pleasure in operating the computer.

PRINT is the command which you will have to get along with quite often. If you think it troublesome to key-in PRINT at every operation, **press the ? in place of PRINT**.



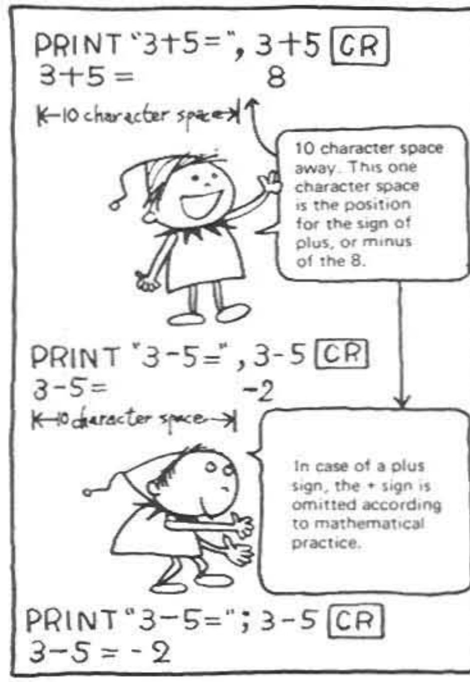
The computer automatically converts the ? to PRINT.

```
?3 * 5
 15
?(3 + 4) * 10
 70
```

What are the PRINT's 1st and 2nd Approaches?

It is possible to add a plurality of items, such as strings and expressions, to the PRINT command. In this case, individual items should be separated using **semicolons** and **commas**.

PRINT "3 + 5 = " ; 3 + 5 [CR]
3 + 5 = 8



The expression between the quotation marks is a string. The actual calculation is done according to the expression following the semicolon.

Try it.

What will happen when using a comma (,) in place of a semicolon (;) ?

PRINT "3 + 5 =" , 3 + 5 [CR]
3 + 5 =   8

Why! The result of 3 + 5 is displayed far away from the expression. This means the following difference lies between the semicolon and comma.

; .... Use of this separator results in display of output lists on successive lines.
, ... This separator causes output lists to be displayed in a tabulated format.

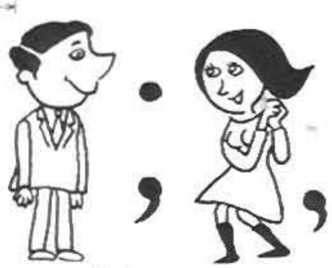
When a separation is made with a comma, the 6 character space is not away from the end position of a string, but 10 character space from the starting position of the string. This fact requires your special attention.

PRINT "12345", 3 + 5 [CR]
12345   8
10 character space

PRINT "123456789", 3 + 5 [CR]
123456789 8
10 character space

If the string is longer than 10 character space, the result 8 is automatically made a further 10 character space away.

PRINT "123456789012", 3 + 5 [CR]
123456789012   8
20 character space



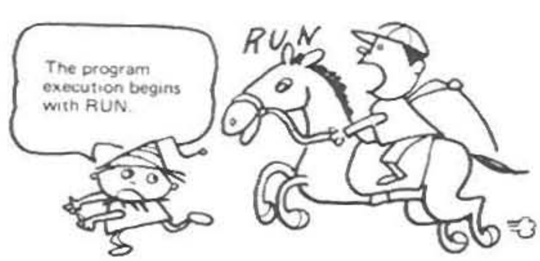
2nd approach

1st approach

Let the Computer Run!

Here is a program with statements covering several line.

```
10  A = 3
20  B = 5
30  C = A + B
40  PRINT A, B, C
50  END
```



This program is the one to instruct assignment of A = 3 and B = 5, and the display of A, B and C on the CRT screen in expression as follows:

C = A + B

The numeral at the head of each statement is called a line number. The computer is sure to execute the line numbers from small in value to large in the correct sequence. Therefore, this makes it possible to insert a new statement in the program afterwards. For example,

35 D = B – C

The computer executes a program in the sequence of the line numbers, and therefore, the line numbers are made in steps of 10, as illustrated in the above example, so that new statements can be inserted later whenever required. The line numbers can be selected at liberty from 1 to 65,535.

[Line No.] [Statement] [CR]

Displaying the character on the CRT screen alone is not sufficient for the keying-in of the statement. After the display of each statement, the [CR] key must be pressed each time, to commit that statement to the memory.

For example, presuming the following,

35 [CR]

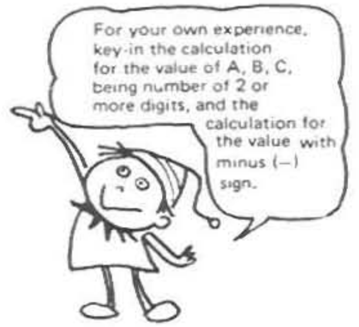
the statement in line number 35 is deleted from the computer.

Now, let's execute the program.
Press the keys as follows RUN [CR].

```
RUN
 3         5         8
```

←10 character space→←10 character space→

One character space for plus or minus sign. For minus value, minus (–) is inserted.



■ LIST for Quick Understanding

While continuing conversation with the computer by repeated trials and errors, the first keyed-in program may sometimes be gone from the CRT screen. Even so, don't worry. The computer never forgets any program once keyed-in. When you want to see the previous keyed-in program, key-in the following:

LIST [CR]

This is followed by the display all the stored programs on the CRT screen. If the program extends over tens and hundreds of lines beyond dispaly at a time, part of the stored programs can be displayed.

LIST − 30 [CR]	Displays a program up to line number 30.
LIST 30 − [CR]	Displays a program after line number 30.
LIST 30 − 50 [CR]	Displays a program between line numbers 30 and 50.
LIST 30 [CR]	Displays a program of line number 30.

The result of NEW ....

To store a new program, clear the previous program using the NEW command. Otherwise, two programs may overlap to cause confusion.

NEW [CR]

This will clear the previous program completely. To ensure this, key-in the LIST command to check that the program is cleared.

Error Puts the Computer in Confusion

```
10  A = 3
20  B = 5
30  C = A + B
40  PRINT A, B, C
50  END
```

This is the same program as used before. Did it run well? If there is an error in any statement, the computer tells you about it. For example,

```
50  EMD
```

If you make a mistake of M for N, the computer executes the program up to line number 40 as instructed, but it does not know what EMD is all about. The computer tells you about a syntax error, as follows:

```
* Error 1 in 50
```

Then, key-in correctly as 50 END. For two statements identical in statement number, if any, the computer takes up the one that was keyed-in later. With this, is your program complete?

If so, try to make a mistake in line number 20, for example.

```
20  5 = B
```

With this, the statement in line number 20 must be revised. Sure? .... Use the LIST command to check the revision.

```
10  A = 3
20  B = 5
30  C = A + B
40  PRINT A, B, C
50  END
205 = B
```

Oh, something funny occurs. Line number 20 is not revised. On top of that, a strange statement with line number 205 lists out. This results because the computer ignored a space (blank part) between 20 and 5 and arranged them as a line number. A space to the computer is entirely insignificant and ignored.

Note: The interpreter notifies the operator of occurence of an error during program execution or operation in the direct mode with the corresponding error number. Refer to the Error Message Table; page 120.

■ Collect the Statewent ¿¿¿

If you want to do the following about your program which has been stored in the computer;

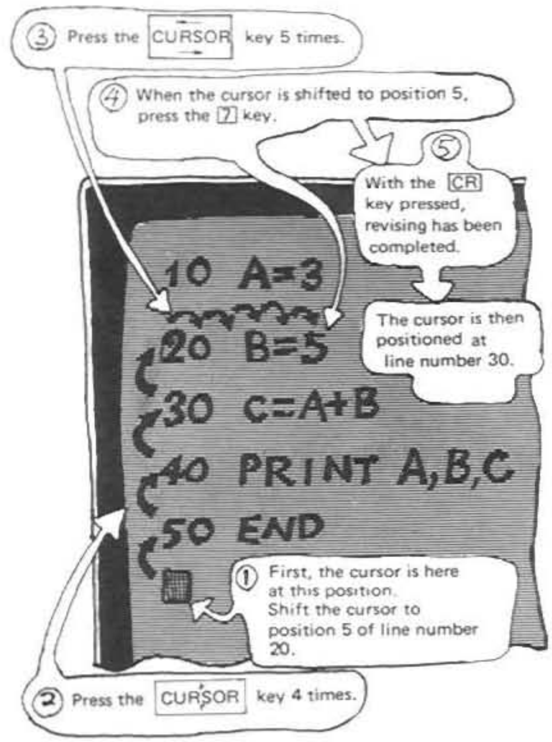
To correct errors,
To modify for a better statement,
To modify for a separate statement,
To modify part of a complete statement and to generate a new statement,
....

Let's study about statement modification, insertion and deletion when the above are required.

Cursor Shift

To revise the characters in a statement, the cursor must be shifted to the respective character positions. Now, let's revise 5 of the statement B = 5 in line number 20 to 7. Refer to the diagram at right for the shift procedure.

With this, the program displayed on the CRT screen has been modified. In fact, however, this has not yet modified the program stored in the computer. To modify the stored contents, the CR key must be pressed. What? Did you key-in 6 instead of 7? To modify the character to the left of the cursor, there are two methods available.



Method 1 Pressing the INST·DEL key.

With the INST·DEL key held down, the cursor shifts to the left by one character space, deleting the character next to it on the left. Press the 7 key again. Needless to say, the CR key must be pressed finally.

Method 2 Shifting to the left using the CURSOR key.

While pressing SHIFT key, depress the CURSOR key. The cursor shifts to the left by the number of times the key is pressed.

Then, press the 7 key again. The CR key must be pressed finally.

Correct the Statement!

Character Insertion

To modify the program on page 18 for the statement in line number 30, as follows,

30 D = 100 + A + B ⟸ (Do not press the CR key yet.)

shift the cursor to character A. Then press the keys as shown below.

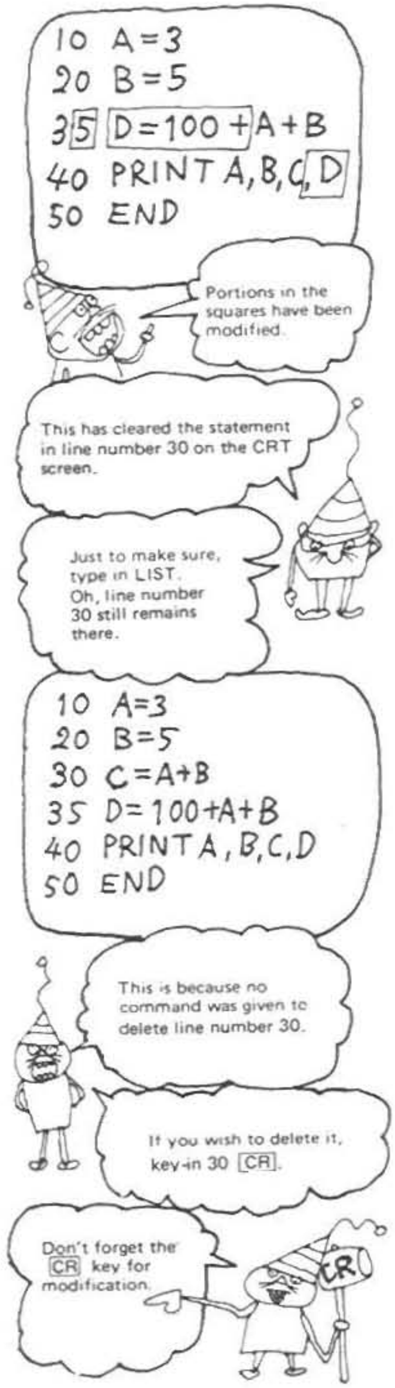
With the SHIFT key depressed, press the INST·DEL key 4 times.

There must be a space for just 4 characters to add 100 +. Key-in 100 + to this space. No more description is required for the revision of C to D. Since the statement has been modified so far, why not modify the line number from 30 to 35, and press the CR key. Modify line number 40 as shown below.

40 PRINT A, B, C, D

Then type RUN CR

```
RUN
 3        5        8           108
```



Character Deletion

35 D = 100 + A + B

Let's modify this statement. To modify it to the following,

35 D = A + B + C

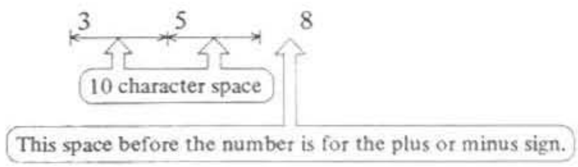
Shift the cursor to character A and press the INST·DEL key 4 times. This shifts the cursor until A + B portion comes right next to mark =

```
RUN
 3        5        8           16
```

Further Study of Comma and Semicolon

For review, the following example is taken again.

```
10  A = 3
20  B = 5
30  C = A + B
40  PRINT A, B, C
50  END
```



You remember this, don't you? In other words, when using commas between A, B and C, a numeral is displayed 10 character space away. Generate a program with new statements inserted, and run it. Statements to be inserted are the following:

```
32  D = B ↑ A
34  E = B * A
36  F = B / A
45  PRINT D, E, F
```

```
RUN
 3          5          8
 125        15         1.6666667
```

With the comma (,) revised to semicolon (;) for line numbers, 40 and 45, run the program once more. To modify the program, type in LIST and use the cursor in as smart a manner as possible.

```
RUN        (Space for plus/minus signs)
 3  5  8
 125   15  1.6666667
```

Semicolon (;) has a function that combines the characters or symbols on display together. Add semicolon (;) to the end of line number 40 then RUN, in order to make sure of this fact.

```
40  PTINT A; B; C;
RUN
 3  5  8  125  15  1.6666667
```

Colon and it's use

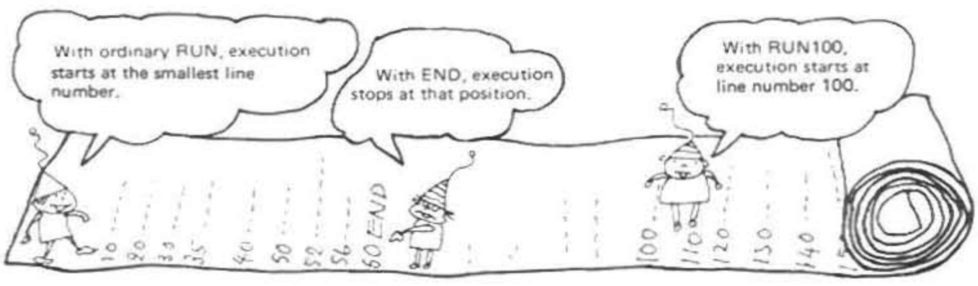
Use of Colon


```
10  A = 3
20  B = 5
30  C = A + B
32  D = B ↑ A
34  E = B ✱ A
36  F = B / A
40  PRINT A; B; C
45  PRINT D, E, F
50  END
```

This program consists of short statements. A program in this length can be processed under one line number, if requied.

```
100 A = 3 : B = 5 : C = A + B : D = B ↑ A :  E = B ✱ A : F = B / A : PRINT
A; B; C : PRINT D, E, F : END
RUN 100
```

Colon (:) is a symbol to be used when more than 2 statements are inserted in one line number. This kind of statement is called a "multi-statement". A statement with 2 lines can be described in one line number. 1 line consists of 40 characters, making it possible to use 76 characters including a line number.



How Much Left ? .... SIZE

It is natural for you to desire to know how much storage capacity is left at your disposal as programs are stored in the computer one after another.

For this, the following is done:

```
PRINT SIZE
```

In response to this, the computer tells you about the remaining storage capacity in bytes.

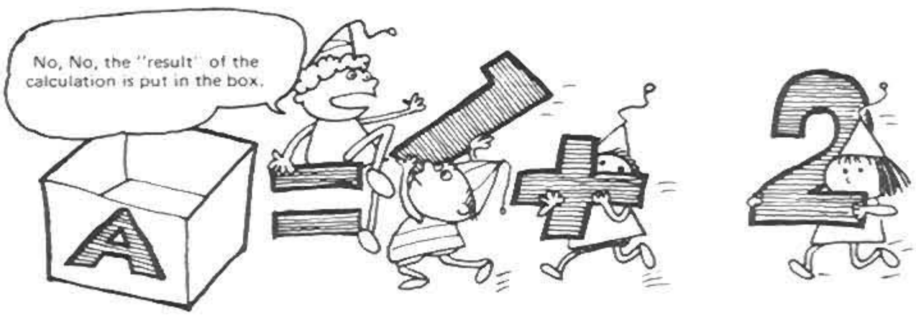
Does "A=B" Equal "B=A"?

Now let's give attention to the = sign we have often used so far. Try the following execution.

```
10  A = 1
20  PRINT A,
30  A = A + 2
40  PRINT A
50  END
RUN
 1           3
```

1 and 3 are on display. A = A + 2 is for line number 30. If this is an equation, A is subtracted from both expressions making 0 = 2, resulting in a contradiction. It is not an equation.

Sign = means that the result of the right expression is substituted by symbol A prepared on the left expression.



In line number 10, value 1 is substituted by symbol A, and at the right expression of line number 30, the value in symbol A and 2 are added and substituted by symbol A using symbol =.

At this time, value 1 previously put in A does not exist any more.

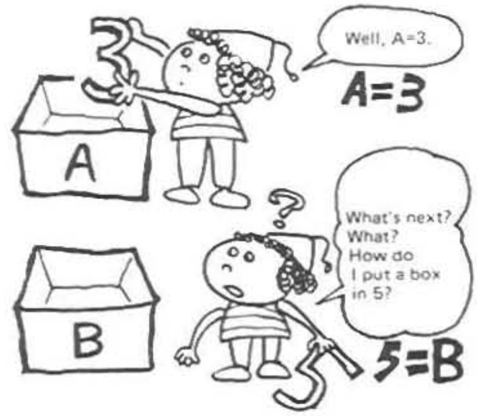
The following 2 programs produce different results which proves that "A = B" does not equal "B = A".

```
10  A = 5
20  B = 7
30  PRINT A, B
40  A = B
50  PRINT A, B
60  END
RUN
  5          7
  7          7
```


```
10  A = 5
20  B = 7
30  PRINT A, B
40  B = A
50  PRINT A, B
60  END
RUN
  5          7
  5          5
```

Variables the Computer is Very Fond of

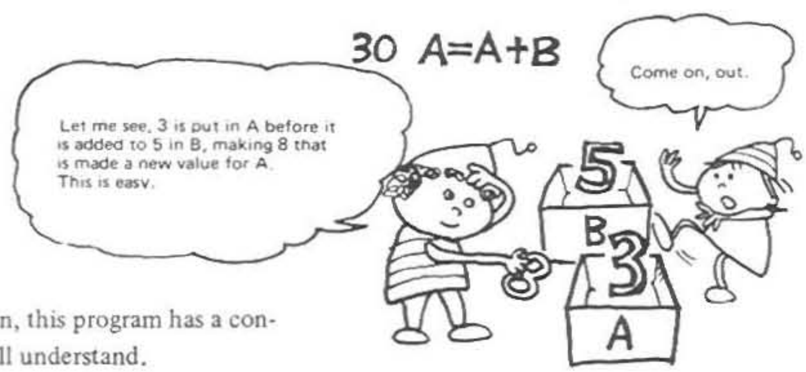
The variables used in the computer statements are different in usage from the mathematical variables. The statement-used variables are the names given to the boxes designed to accommodate values.



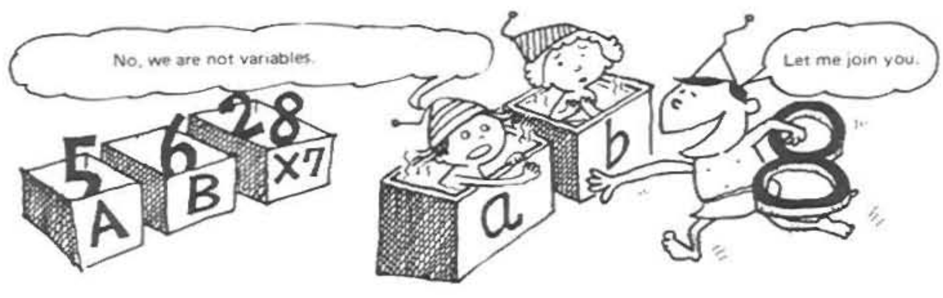
B = 5

This means that value 5 should be substituted by box B. Therefore, the use described under the "Error Puts Computer in Confusion" results in a difficult statement for the computer, though it cannot be mistaken as a line number. Because it says to put box B into 5.

```
10  A = 3
20  B = 5
30  A = A + B
40  PRINT A
50  END
RUN
  8
```



From the mathematical definition, this program has a contradiction, however, the computer will understand.

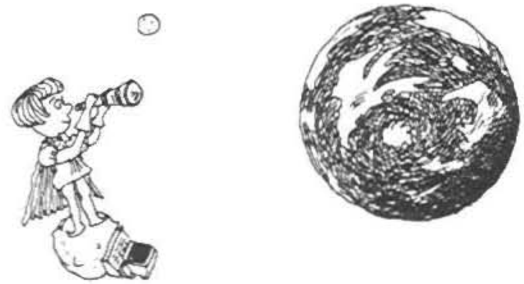


Characters subject to Variables

1. A variable should be a combination of two or less characters. Any variable over 2 characters can be stored, but the characters after the second are neglected in computer processing. For example, ABC and ABD can be displayed. In processing, however, they are regarded as the same variables as AB.
2. The following are the characters for use as variables:
   (1) A to Z. Alphabetical 26 ways.
   Example: A, M, Z
   (2) 260 characters with numeral of 1 figure (0 to 9) added to the alphabet.
   Example: A0, K5, Z9
   (3) Characters with two alphabetical characters combined.
   Example: AA, BK, XZ
   However, some variables, such as IF, ON, TO, etc in BASIC reserved words, should not be used.

Computing the Earth

The prince of a star takes accurate observation of the earth. "The earth is a blue planet over there in the Solar System. Though slightly distorted, the earth is approximately 13,000 kilometers in diameter. From orbit calculation, its mass is about 6×10^{18} thousand tons."



The prince went to his computer to generate the following program for calculations of volume VE, surface area SE and mean density ZE of the earth.

```
10  DE = 13000 ···················· Substitute the earth's diameter for variable DE.
20  WE = 6E + 18 ··················· Substitute the earth's mass for variable WE.
30  SE = 4 * π * (DE/2) ↑ 2 ········· This substitutes the surface area for variable SE.
40  VE = 4 * π * (DE/2) ↑ 3/3 ······· This substitutes the earth's volume for variable VE.
50  ZE = WE/VE * (1E – 2) ·········· This substitutes the mean density for variable ZE.
60  PRINT "EARTH DIAMETER" ; DE ; " KILOMETERS"
70  PRINT "EARTH SURFACE AREA" ; SE; "SQUARE KILOMETER"
80  PRINT "EARTH VOLUME" ; VE; "CUBIC KILOMETER"
90  PRINT "EARTH MASS" ; WE; "THOUSAND TONS"
100  PRINT "EARTH MEAN DENSITY" ; ZE; "KILOGRAM/CUBIC METER"
110  END
```

The prince of a star understands slightly the size of the earth. Pay much attention to the units used in the calculations. Further attention is focused on the sequence of calculations when the arithmetic expression contains, *, + or ↑. The operation priority is shown below:

1 ↑ (Power)
2 – (Minus sign)
3 *, / (Multiplication and Division)
4 +, – (Addition and Substraction)

The expressions below are complex in combination. Do you see any difference between the expressions?

```
2 + 3 ↑ 2 = 11
(2 + 3) ↑ 2 = 25

12/3 * 2 = 8
12/(3 * 2) = 2

2 * 2 ↑ 3 = 16
(2 * 2) ↑ 3 = 64.000001

12/3 ↑ 2 = 1.3333333
(12/3) ↑ 2 = 16
```

Archimedes and the Mysterious Soldier

The sum of the interior angles of a triangle is 180°. With a flash of inspiration, Archimedes sat on the road and drew a triangle. There came a mysterious soldier with his spear pointing at Archimedes.

Soldier: Archimedes, your life is finished. Be prepared to die!

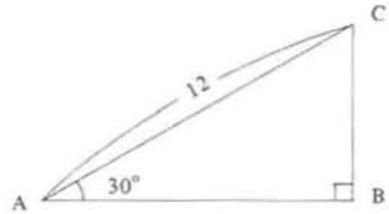
Archimedes: Wait a minute, I will finish this calculation.

Soldier: What? Angle A is 30° and angle B is a right angle.
It's easy to determine angle C. 60°. If side CA length is known, side AB and BC lengths or even the area of the triangle can be easily determined.

Archimedes: Don't be silly.

Soldier: All that needed is to generate a BASIC program. Let me see, Oh, Yes, it's good with CA = 12.

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



Using the inverse tangent ATN, let's determine the size of angle C from the side AB and BC lengths known. This requires the following to be keyed-in.

```
50 C = ATN (AB/BC) * 180/π
```

The result is in the unit of degree. The same result is obtained, isn't it?

The Function Family Members

Introduced here are more functions, such as SIN (X). Such functions are used with parentheses, in which constants, variables or arithmetic expressions can be placed.

Function	BASIC Symbol	Calculated Value	Example
Integer	INT (X)	Maximum integer within X	INT (3.14) = 3 INT (0.55) = 0 INT (−7.9) = −8
Absolute value	ABS (X)	Absolute value of X	ABS (2.9) = 2.9 ABS (−5.5) = 5.5
Sign	SGN (X)	1 if X is greater than 0. 0 if X is equal to 0. −1 if X is less than 0.	SGN (500) = 1 SGN (0) = 0 SGN (−3.3) = −1
Exponent function	EXP (X)	e^x (e = 2.7182818)	EXP (1) = 2.7182818 EXP (0) = 1
Common logarithm	LOG (X)	$\log_{10} X$ Provided X is greater than 0.	LOG (3) = 0.47712125
Natural logarithm	LN (X)	$\log_e X$ Provided X is greater than 0.	LN (3) = 1.0986123
Square root	SQR (X)	$\sqrt{X}$ Provided X is greater than or equal to 0.	SQR (9) = 3 SQR (0) = 0

Is PRINT 2 ∗ 2 Identical to PRINT 2 ↑ 2?

Well 934 ↑ 2 results in fractions of 872355.99, but 934 ∗ 934 results 872356. This is correct as an arithmetic expression, but calculations are done in a limited number of figures, involving unexpected errors. For example, 2 ↑ 2 is done using the formula called a progression expansion.

$$2\uparrow 2 = 1 + \frac{2\ln 2}{1!} + \frac{(2\ln 2)^2}{2!} + \cdots + \frac{(2\ln 2)^n}{n!} + \cdots$$

This calculation may cause the computer to scream. The computer will produce certain types of errors. These errors are, however of little concern.

Free Definition of Function.....DEF FN

Various functions have been described, and here is an explanation of DEF FN defined as a new function combining such various functions. Some definition examples are listed below:

DEF FNA (X) = 2 * X ↑ 2 + 3 * X + 1 .... $2X^2 + 3X + 1$ is defined as FNA (X).
DEF FNB (X) = SIN (X) ↑ 2 + COS (X) ↑ 2 .... $\sin^2 X + \cos^2 X$ is defined as FNB (X) this is always 1.
DEF FNE (V) = 1/2 * M * V ↑ 2 .... $1/2MV^2$ is defined as FNE (V).

DEF represents "define". New functions are named with FN suffixed. X or V in the parenthesis is called the argument. For example, the third function (seems to be motion energy) is used.

```
10 DEF FNE (V) = 1/2 * M * V ↑ 2
20 M = 5.5 : V = 3.5
30 PRINT FNE (V), FNE (V * 2), FNE (V * 3)
40 END
```

Motion energy at initial velocity V and motion energy with velocity doubled or tripled are displayed. DEF FN command is very convenient particulary when the same functions are often used in a long program.

Fall from an altitude of 10,000 meters!

How do you think the velocity and altitude of a fall from an altitude of 10,000 meters changes per second?

Fuction FNV (T) in the program is the fall velocity after a lapse of time T, and FNH (T) is the altitude at the same time.

Acceleration of gravity G, atomospheric resistance factor K and altitude H when a fall occurs are assigned by line number 20.

```
10 ? " ◙ " : T = 0
20 G = 9.8 : K = 0.15 : H = 10000
30 DEF FNV (T) = G/K * (1 − EXP ( − K * T))
40 DEF FNH (T) = H − FNV (T) * T
50 ? " ◘ "
60 PRINT "TIME        "; T : MUSIC " + A0" ...... Instruction with beep to be explained on page 76.
70 PRINT "VELOCITY" ; FNV (T)
80 PRINT "ALTITUDE" ; FNH (T)
90 T = T + 1 : GOTO 50 .................... This is a shift instruction for the program to be shifted
                                            to line No. 50.
```

◙ and ◘ are entered with the [CLR HOME] key pressing in the graphic mode.

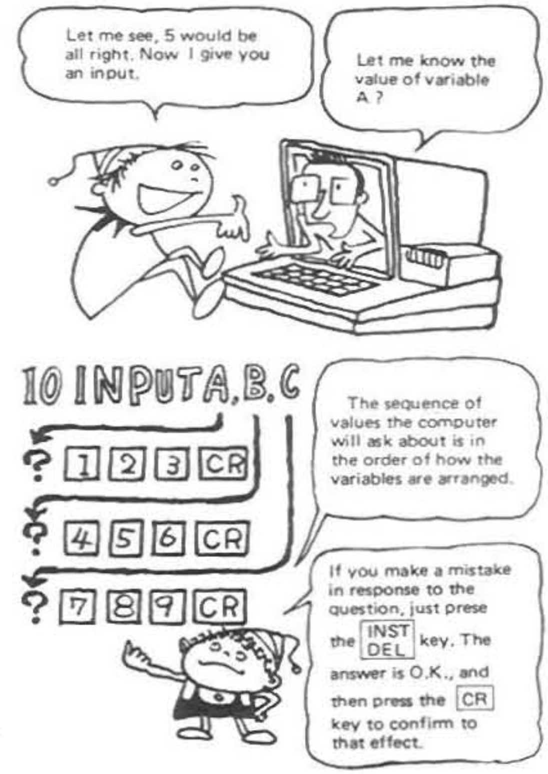
■ This is INPUT, Answer Please

To inform the computer of variables' values, we have so far taken the method where the value is first determined, as follows:

```
10  A = 3
20  B = 5
......................
```

There are several methods available for informing the computer of the values of variables. One of them uses a command called INPUT.

```
10  INPUT A, B, C
20  D = A + B + C
30  PRINT A, B, C, D
40  END
RUN
? ▩
```



This is new display, isn't it? ? ▩ is making an inquiry to you about the value of first variable A following the INPUT command. In response to this inquiry, key-in the value and press the CR key to inform the computer that everything is O.K.

Look! The same display is there. This is the inquiry about the value of the second variable B. If there are 3 variables, the computer asks question 3 times. If you reply using any key other than 0 to 9 by mistake, and press the CR key, the following is displayed.

* Error 4 in 10 .... Data type mismatch

The computer will then make inquiries about the values all over again.

```
10  INPUT A, B, C, D
20  INPUT E, F, G, H
30  PRINT H, G, F, E
40  PRINT D, C, B, A
50  END
```

```
10  INPUT "A = ?" ; A
20  INPUT "B = ?" ; B
30  INPUT "C = ?" ; C
40  S = A + B + C
50  M = S/3
60  PRINT "TOTAL" ; S, "MEAN" ; M
70  END
```

Yes or No in Reply to a Proposal?

On a sunny Sunday, a gentleman and a lady sit face to face in a nice coffee shop. He is 43 years old, and she is 22 years old.

Genteleman: I love you at first sight. Can you marry me?

Lady: Yes, if you love me so much. I don't care about the age difference. But not now. You have to wait until my age is half of yours.

Presume his age is A, hers is B and the number of years she asked him to wait is X. After X years, he is A + X years while she is B + X. Since her age is then half of his, the condition of A + X = 2 (B + X) is required. To solve the equation for X, the following is obtained.

$X = A - 2B.$

```
10 PRINT "WHAT IS HIS AGE ?"
20 INPUT A
30 PRINT "WHAT IS HER AGE ?"
40 INPUT B
50 X = A - 2 * B
60 PRINT "WAIT" ; X ; "YEARS!"
70 END
RUN
WHAT IS HIS AGE?
? 43 [CR]
WHAT IS HER AGE ?
? 22 [CR]
WAIT - 1 YEARS!
```

It is impossible to wait for − 1 year. In other words, they could have been married a year ago. Asked suddenly about a question, the computer may be confused at what variable you are talking about. In this program, a string indicating inquiry contents is inserted in line numbers 10 and 30. The string for an answer is also used in line number 60.

The INPUT method in this program can be simplified. Modify line numbers 10 and 30 as described below, deleting line numbers 20 and 40 from the program.

```
10 INPUT 'WHAT IS HIS AGE ?" ; A
30 INPUT "WHAT IS HER AGE ?" ; B
```

Those that follow line number 50 are identical to the above program.

```
RUN
WHAT IS HIS AGE? 43 [CR]
WHAT IS HER AGE? 22 [CR.]
WAIT - 1 YEARS!
```

DATA and READ go hand in hand

Another method to inform the computer of variables.

```
10  READ A, B, C, D
20  X = A + B + C + D
30  PRINT X
40  DATA 3, 5, 7, 9
RUN
 24
```

This program picks up values which are then used for calculation.

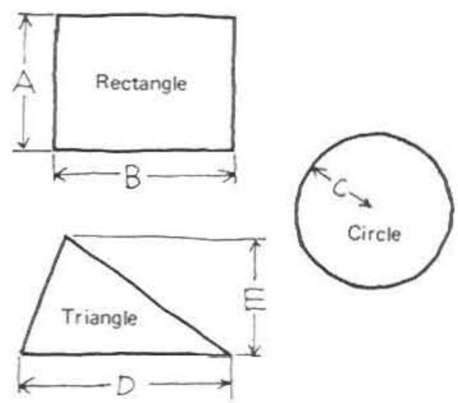
Two types of commands, READ and DATA, are used in this method.

```
READ  A, B, C, D  . . . . . . . . . . . . . . . . . . . . . . Some variables are arranged.
DATA  10, 11, 12, 13 . . . . . . . . . . . . . . . . . . . . . Number of values identical to that of variables that
                                                               follow READ.
```

Similar to the INPUT command, the arrangements of variables and values must be matched.

It is unexpectedly easy to generate programs to determine rectangular, circle and triangle areas using the READ and DATA commands.

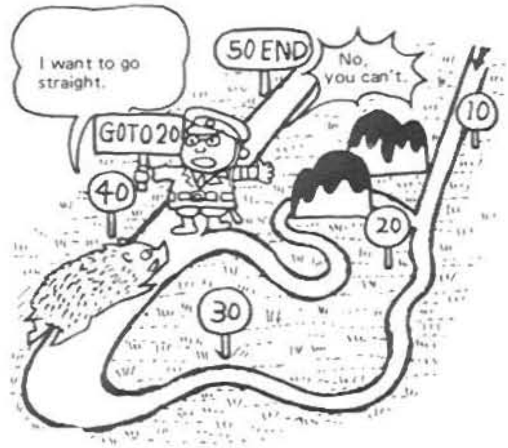
```
10  READ A, B
20  S1 = A * B
30  PRINT "RECTANGLE =" ; S1
40  READ C
50  S2 = π * C ↑ 2
60  PRINT "CIRCLE =" ; S2
70  READ D, E
80  S3 = D * E / 2
90  PRINT "TRIANGLE =" ; S3
100  DATA 2, 4, 6, 8, 10
110  END
```



There seems to be room for improvements in the program.
Try various ways yourself.

■ Don't Oppose GOTO

For programs described so far, the computer executes them in the correct sequence from small to large line numbers. In fact, however, execution requires the sequence to be changed on some occasions. On such occasions, GOTO statements are very effective. GOTO means that an unconditional branch is made to the line number specified.




```
10  N = 1
20  PRINT N
30  N = N + 1
40  GOTO 20
50  END
RUN
  1
  2
  3
```

Not stopped? .... Press the SHIFT key, then BREAK key to stop.

Once upon a time, the great Knight Sir Lancelot of the Lake did a great deed for King Arthur of Camelot. King Arthur was so greatful to Sir Lancelot he said, "I would like to give you any prize you care to ask for".

Sir Lancelot replied "Thank you my Lord, I would like to have 1 Ginea today, 2 Gineas tommorrow, 4 Gineas on the 3rd day, 8 Gineas on the 4th day and so on until the 30th day". King Arthur was so surprised by such a small request that he agreed immediatly.

Let us make the program below to find out how much King Arthur must pay.

```
10  D = 1 : F = 1 : S = 1
20  PRINT "DAYS", "GINEAS", "TOTAL"
30  PRINT D, F, S
40  D = D + 1 ........................... This is for adding oneday to each day.
50  F = 2 * F ........................... This is for multiplying oneday's total by two.
60  S = S + F ........................... This shows the total by adding to previous day total.
70  IF D = 31 THEN 90
80  GOTO 30
90  END
RUN
```

DAYS	GINEAS	TOTAL
1	1	1
2	2	3
⋮	⋮	⋮
10	512	1023
⋮	⋮	⋮
20	524288	1048575
⋮	⋮	⋮
30	.53687091E + 09	.10737418E + 10

On the 10th day he was given 1023 Gineas, on the 20th day he was given 1048575 on the 30th day he asked for about 1000000000 Gineas.

IF.....THEN

IF ~ THEN

```
10 IF △△△ THEN □□□
20 ■■■
```

If △△△ conditions are satisfied, then □□□ jobs can be executed. If not, omit □□□ and go to ■■■ of the next line number. This is the IF ~ THEN statement. If □□□ is a numberal, a jump is made to the line number of the numberal.

```
10 READ A
20 IF A >= 0 THEN PRINT "A =" ; A
30 GOTO 10
40 DATA -10, 20, 5, -9, 8, -6, 5
50 END
RUN
A = 20
A = 5
A = 8
A = 5
```

Positive numbers alone are displayed.
The general form of IF .... THEN statements is as follows:

IF conditions THEN statement or line number

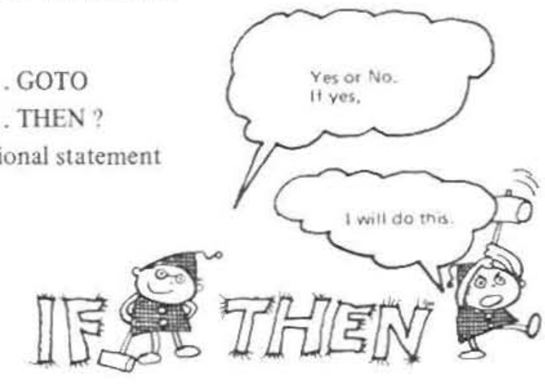
The conditions herein referred to are "greater than" or "less than" expressions using equal sign and unequal sign.

Sign Conditions	How to Use
=	A = B
<	A < B + C
>	A > B + C
<= or =<	A + B <= C
>= or =>	A >= B
< > or > <	A < > B

IF.....THEN and its Associates

If . . . conditions are true (Yes), statement after THEN is executed. If they are false (No), execution is advanced to the next line number. Here is an introduction to its associates.

IF . . . . THEN GOTO or IF . . . . GOTO
IF . . . . THEN PRINT or IF. . . . . . . THEN ?
IF . . . . THEN A = 5 ∗ 7 substitutional statement
IF . . . . THEN INPUT
IF . . . . THEN READ
IF . . . . THEN GOSUB
IF . . . . THEN RETURN
IF . . . . THEN STOP
IF . . . . THEN END




```
10 PRINT " 🅲 "
20 PRINT "INSERT OPTIONAL FIGURE FROM 1 TO 9."
25 PRINT "INSERT 0 WHEN YOU STOP."
30 L = 0 : M = 0 : N = 0
40 INPUT A
50 IF A = 0 THEN 90
60 IF A <= 3 THEN L = L + 1 : GOTO 40
70 IF A <= 6 THEN M = M + 1 : GOTO 40
80 N = N + 1 : GOTO 40
90 PRINT "YOU INSERTED FIGURES FROM 1 TO 3" ; L ; "TIMES" ;
100 PRINT "FROM 4 TO 6" ; M ; "TIMES";
110 PRINT "AND FROM 7 TO 9" ; N ; "TIMES"
120 END
```

A new symbol 🅲 is used in line number of 10. Display of 🅲 is possible when [CLR HOME] key is pressed, with the [SHIFT] key depressed in the graphic mode. This command will clear all the character on the CRT screen and shift the cursor to the top left corner of the CRT screen.

In addition, when the [CLR HOME] key alone is pressed in the graphic mode, symbol 🅷 appears.

This symbol functions only to shift the cursor to the top left corner.

If these are not clear, check with PRINT " 🅲 " or PRINT " 🅷 ".

Leave Any Decision to IF

IF can select Even numbers

Let's consider a program for selecting even numbers only, out of many numerals, using IF . . . GOTO statement. IF has great ability to select numbers.

```
10  READ X : IF X = -9999 THEN STOP
20  IF X/2 <> INT (X/2) GOTO 10
30  PRINT X ; : GOTO 10
40  DATA 2, 13, 56, 55, 4, 78, 31
50  DATA 6, 22, 15, 19, 80, 11, -9999
RUN
 2 56 4 78 6 22 80
```

INT (X/2) in line number 20 is the statement for picking integers alone. Therefore, if X is even, X/2 <> INT (X/2) is impossible, with execution advancing to line number 30. If it is possible, it's regarded as odd, reading the next value.

To test your progress, let's try an exercise. How can you decide the multiple of 3 or 4? You've got it, haven't you? The answer is this.

Modification for the multiple of three ....

```
20  IF X/3 <> INT (X/3) GOTO 10
```

Modification for the multiple of four ....

```
20  IF X/4 <> INT (X/4) GOTO 10
```

IF can select Maximum and Minimum

```
10  S = 999 : L = -999
20  READ X : IF X = -9999 THEN 80
30  IF X > L THEN L = X
40  IF X > S THEN S = X
50  GOTO 20
60  DATA 2, -5, 91, 256, -43
70  DATA 87, 321, -76, -9999
80  PRINT "MAXIMUM VALUE =" ; L
90  PRINT "MIMINUM VALUE =" ; S
100  END
RUN
MAXIMUM VALUE = 321
MINIMUM VALUE = -76
```

Line number 10 is very important. Put as large a number as possible in variable S for substitution of the minimum value, and as small a number as possible in variable L for substitution of the maximum value. What about the execution results? Variable L and S come out as true maximum and minimum values. This is a good example of the use of IF . . . . THEN.

Password Found for Numbers

The greatest common divisor (GCD) is a passward for two integers. For example, presuming that two numbers are 10 and 20, divisible numbers for 10 and 20 are four numbers that are 1, 2, 5 and 10. Of these numbers, the maximum value, namely, 10 is the greatest common divisor for numbers 10 and 20.

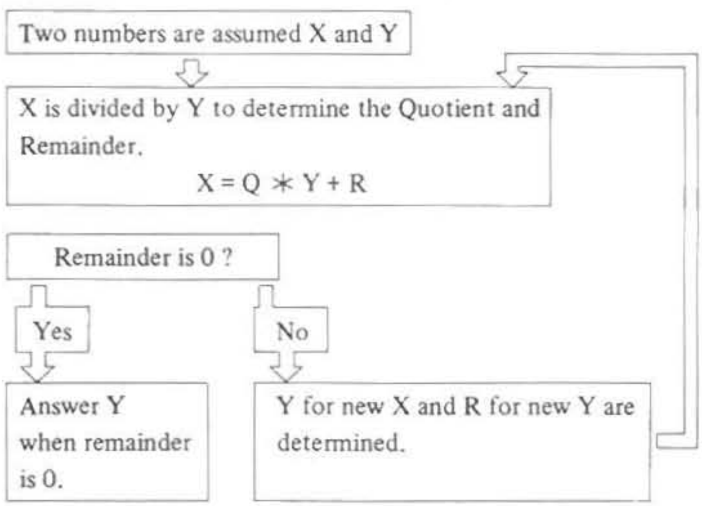
Now, let's generate a program.

```
10 PRINT "X", "Y", "PASSWORD"
20 READ X, Y
30 PRINT X, Y
40 Q = INT (X/Y)
50 R = X – Q * Y
60 X = Y : Y = R
70 IF R > 0 THEN 40
80 PRINT X : GOTO 20
90 DATA 63, 99, 1221, 121, 64, 658
100 DATA 12345678, 987654321
110 END
RUN
```

(Note on line 30:) Comma (,) following Y is very convenient for continuous display on the CRT screen.

Exposure of a Trick for this Program !

Long ago, a Greek mathematician, Euclid, developed this method of solution.



Using IF . . . . , try as many as possible.

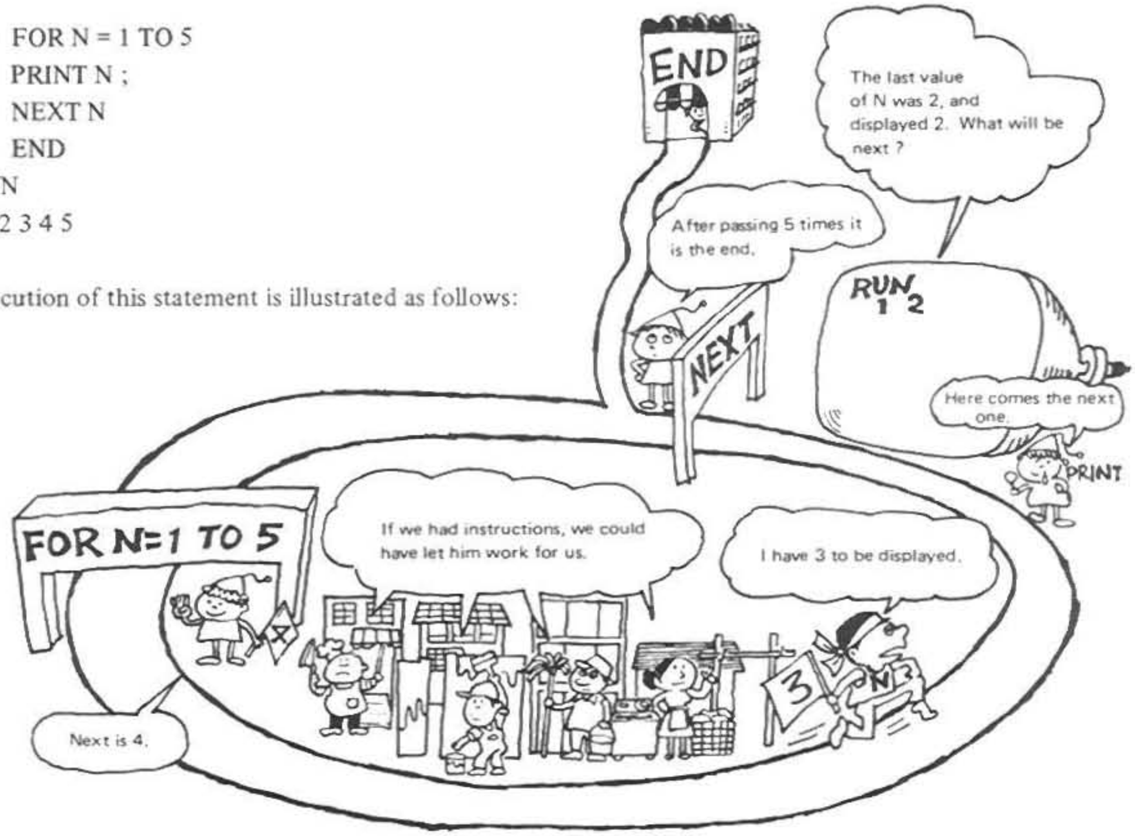
```
10 IF SGN (X) = –1 THEN ? "MINUS"
20 IF SGN (X) = 0  THEN ? "ZERO"
30 IF SGN (X) = 1  THEN ? "PLUS"
```

FOR.....NEXT is an Expert of Repetition

The FOR . . . . NEXT statement is an instruction used for repetition of a sequence of program statements. Let's have a look at a simple program, first.

```
10  FOR N = 1 TO 5
20  PRINT N ;
30  NEXT N
40  END
RUN
 1 2 3 4 5
```

The execution of this statement is illustrated as follows:



The variation of N is not only increased by 1, but can be increased, for example, by 0.5 or decreased by 2. The variation at this time is assigned by the word of STEP.

To increase in 0.5 increments:

```
10  FOR N = 1 TO 5 STEP 0.5
```

To decrease in 2 decrements:

```
10  FOR N = 5 TO 1 STEP –2
```

The general form of FOR . . . . NEXT statement is as follows:

FOR variable = Initial Value TO Last Value STEP Variation
Repeated Program
NEXT Variable

The initial value, final value and variation may be either a variable, constant or expression.

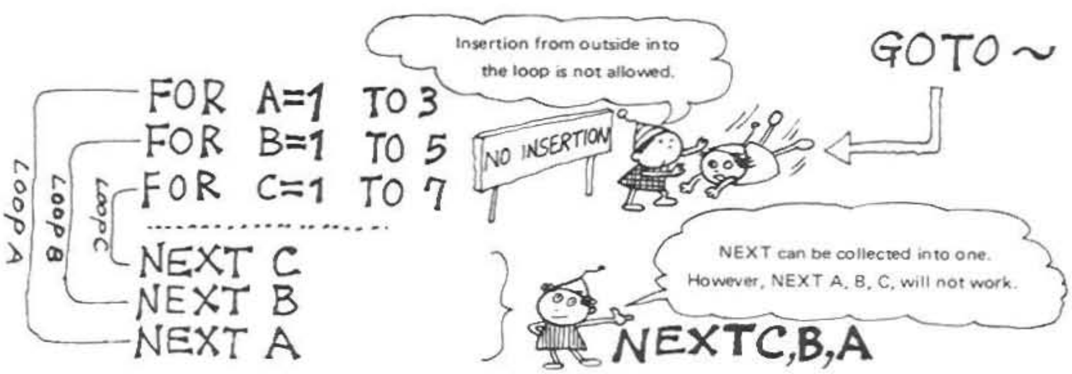
Loop in a loop

Alice is doing her homework. She is preparing a multiplication table using the computer, and a program which contains double FOR . . . . NEXT statements.


```
10  FOR X = 1 TO 9 ─────────────┐
20  FOR Y = 1 TO 9 ──┐          │
30  PRINT X * Y ;    │ Loop Y   │
40  NEXT Y ──────────┘          │ Loop X
50  PRINT                       │
60  NEXT X ─────────────────────┘
```

In the FOR . . . . NEXT loop for variable X, the FOR . . . . NEXT loop for variable Y is included. Variables X and Y vary from 1 to 9, respectively, and 1 is substituted for variable X to execute variable Y loop. In other words, with variable X remaining at 1, variable Y varies 1, 2, 3, ... to 9, and each time, the multiplication product with variable X is displayed at line number 30. When varialbe Y reaches 9, a line feed is executed at line number 50, and at line number 60, variable X is then 2.

The FOR . . . . NEXT loop can be used double, triple, etc., up to 15. What must be observed, however, is that loops are never crossed and no jump into the loop by means of GOTO is allowed.



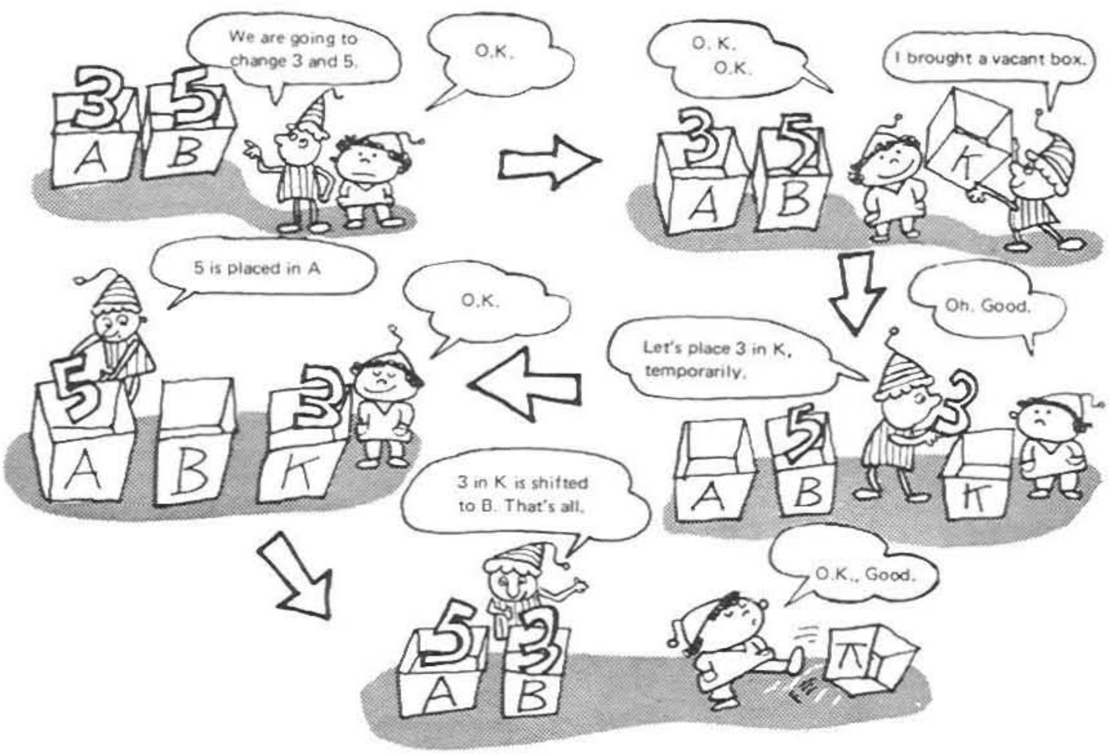
Thus, loop C is completely included in loop B, while loop B is completely included in loop A. As shown on the right, one word NEXT can be used for all three loops.

Line up in Numerical Order

With 4 numericals selected at random and keyed-in, the computer can arrange them in numerical order. This is a program for such a function. Use the INPUT command.

```
10 PRINT " [C] "
20 PRINT "TELL ME VALUES OF 4 NUMERALS" : PRINT
30 INPUT A, B, C, D
40 IF A <= B THEN K = A : A = B : B = K
50 IF B <= C THEN K = B : B = C : C = K
60 IF C <= D THEN K = C : C = D : D = K
70 IF A < B GOTO 40
80 IF B < C GOTO 40
90 IF A < C GOTO 40
100 PRINT A, B, C, D
110 PRINT : PRINT "ONCE MORE PLEASE" : PRINT
120 GOTO 30
```

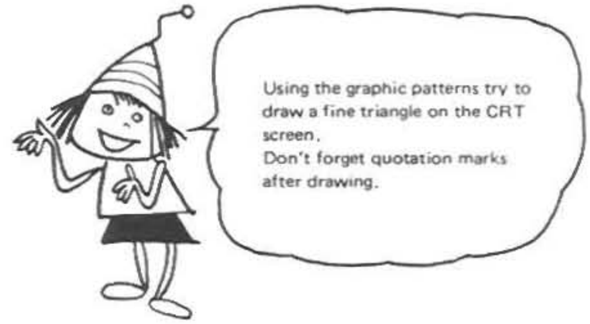
Give attention to line number 40. Using another variable K, after the THEN statement, the job is being done by changing the values of A and B. If A = 3 and B = 5 in the initial state;



By the above job, A = 5 and B = 3 are obtained. Similar processing is executed at line numbers 50 and 60. Line numbers 70 through 90 are prepared for the repetition of the changing job.

■ How Many Right Triangles are Possible?

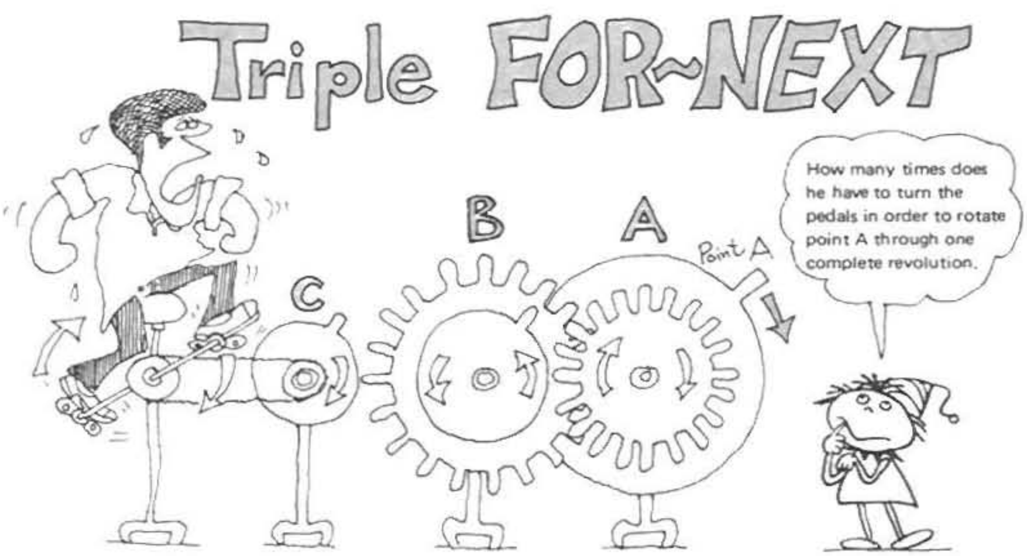
Now, let's generate a program that picks up positive integers from 1 to 20 to meet the Pythagorean theorem $A^2 = B^2 + C^2$.




```
10  PRINT " C "
20  PRINT "     |\ "
30  PRINT "     | \ "
40  PRINT "   B |  \ A"
50  PRINT "     |   \ "
60  PRINT "     L___\ "
70  PRINT "           "
80  PRINT "        C"
90  PRINT : PRINT "POSITIVE INTEGERS TO MEET PYTHAGOREAN THEOREM"
110 PRINT : PRINT " → A", " → B", " → C"
120 FOR A = 1 TO 20
130 FOR B = 1 TO 20
140 FOR C = 1 TO 20
150 IF A * A - B * B - C * C = 0 THEN PRINT A, B, C
160 NEXT C, B, A
180 END
```

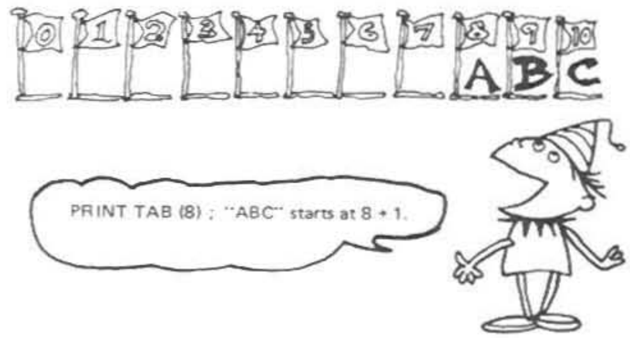
You already know the meaning of line number 10. Try to draw carefully so that a fine triangle is formed between line numbers 20 through 80. At line numbers 120 through 160, the FOR . . . . NEXT loop is triple. The equation shown at line number 150 is repeated 8000 times (20 × 20 × 20) with C from 1 to 20 at A = 1 and B = 1, and with C from 1 to 20 at A = 1 and B = 2, and so on.

This operation requires a considerable period of time for its completion.



■ TAB(   ) is Versatile

It is possible to assign where to start writing the characters or symbols of a string on the CRT screen. The TAB (   ) is used to do so.



Using PRINT TAB (8) ; "ABC", string ABC is displayed at the number in the parenthesis counted from the left hand side, namely, starting at the 9th position.

The numbers to be assigned for the parenthesis are from 0 to 78, and variables may be used if defined as numerals.

Let's operate an example of a simple program combined with the FOR . . . NEXT statements.

```
10  FOR X = 1 TO 20
20  PRINT TAB (X) ; " * "
30  NEXT X
```

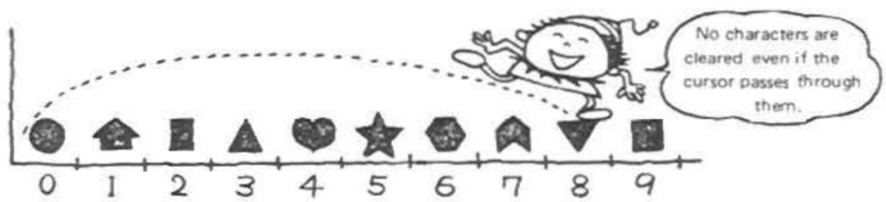
```
10  FOR Y = 1 TO 20
20  PRINT TAB (20 – Y) ; " * "
30  NEXT Y
```

Now, let's try a little more complex program.

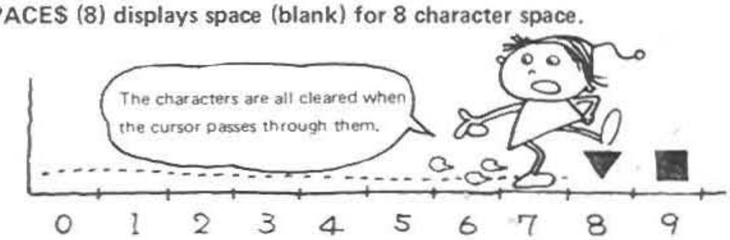
```
10  PRINT " ◘ " : PRINT SPACE$ (8) ;
20  FOR X = 1 TO 22 : PRINT " * " ; : NEXT X : PRINT
30  FOR Y = 1 TO 20
40  PRINT TAB (8) ; " * " ; TAB (29 – Y) ; " ■ " ; TAB (29) ; " * " : NEXT Y : PRINT SPACE$ (8) ;
50  FOR Z = 1 TO 22 : PRINT " * " ; : NEXT Z
```

A new statement is there at line numbers 10 and 40. When this SPACE$ (   ) is substituted for TAB, exactly same result is obtained. However, there is the difference shown below between the SPACE$ and TAB.

TAB (8) shifts the cursor by 8 character space from the left hand on CRT screen.



SPACE$ (8) displays space (blank) for 8 character space.

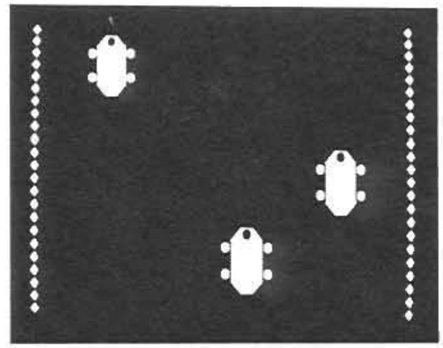


Grand Prix using RESTORE

Challenge of a Car Race ....

How about the simplest car race program ?

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



TAB (X) at line number 40 determines where to display automobiles on the road, particularly from the left side. What distance between the automobiles ? For this, random numbers from 1 to 9 are generated at line number 70, and at line numbers 80 through 100, automobiles are controlled so that they do not collide. By the way, RESTORE at line number 110 is not a familiar command, is it ?

RESTORE Returns to the Start of Data

No matter where it may be, or no matter how it may be scattered, DATA statement is read by READ statement.

```
    O.K.
10  DATA 27
20  READ A, B, C
30  DATA 10
40  ......
50  DATA 9, 13
60  READ D
70  END
```

```
    NO
10  READ A, B
20  READ C
30  DATA 27, 10, 9
40  READ D
50  END
```

Whey No ? Because variable D has no value of DATA to be read.
Then what about this ?

```
10  READ A, B
20  READ C
30  DATA 27, 10, 9
35  RESTORE
40  READ D
50  END
```

·········· **RESTORE statement enables the reading of the first data in the first DATA statement of the program.**

Talkative Strings

The computer should generate a language understandable to human beings and should talk to us .... To make such a desire come true, string variables are absolutely necessary.

```
10  A$ = "MIKE" : B$ = "PAUL"
20  C$ = "TONY" : D$ = "PETE"
30  E$ = "DENIS" : F$ = "MARTIN"
40  G$ = "PHILIP"
50  I$ = "JACK" : J$ = "HARRY"
60  K$ = "BILL" : L$ = "DAVID"
```

Ordinary variable symbols with $ (dollar sign) suffixed are called string variables. Processing, very similar to that of ordinary variables is possible. Let's look at some examples to see their characteristics.

```
70  PRINT B$
80  PRINT A$
RUN
PAUL
MIKE
```

Using " ; ", connects string variables.

```
100  PRINT B$ ; C$ ; A$ ; E$ ; L$ ; D$ ; K$
RUN 100
PAULTONYMIKEDENISDAVIDPETEBILL
```

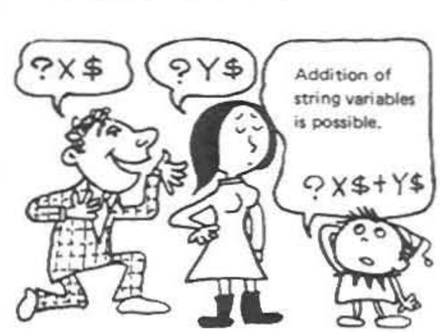
What will happen if " , " is used in place of " ; " ?

```
120  PRINT B$, A$, I$
RUN 120
PAUL          MIKE          JACK
```

(10 character space)

To combine string variables to generate a new string, add string variables together using "+".

```
140  X$ = B$ + C$ + D$ + J$ + G$
150  Y$ = A$ + C$ + B$ + F$ + I$ + G$
160  PRINT X$
170  PRINT Y$
```



With this, a new string variable is possible.

Another type of INPUT

Combine string variables and INPUT statement in a program to create a poem.

```
10  INPUT A$, B$, C$
20  PRINT A$ ; " " ; B$ ; " " ; C$
30  GOTO 10
RUN
?  A FROG JUMPS
?  INTO A POND
?  WITH A SPLASH OF WATER.
A FROG JUMPS INTO A POND WITH A SPLASH OF WATER.
```

Space for syllable separations.

Using INPUT statement, the input of a string can be keyed-in, requiring no quotation marks " ".

Another example of this is shown below.

```
10  PRINT "TYPE IN ANYTHING AT ALL"
20  INPUT AA$
30  PRINT "YOU HAVE JUST TYPED" ; AA$
40  GOTO 10
```

String variables, when combined with READ and DATA statements, can be generated into a program.

```
10  READ X1$, X2$
20  PRINT X1$ ; "LIKE CREAM" ; X2$
30  DATA DO YOU, CAKES ?
RUN
DO YOU LIKE CREAM CAKES ?
```

Note that quotation marks are not required when READ statement is used.

LEFT$, MID$, RIGHT$

LIFT $ (   ), MID $ (   ) and RIGHT $ (   ) are statements to generate new strings by taking out part of strings.

```
10  A$ = "AQUARIUS PISCES ARIES LEO"
20  B$ = LEFT$ (A$, 15)
30  PRINT B$
RUN
AQUARIUS PISCES
```

(Character up to the 15th from the left hand side.)

LEFT $ (A$, 15) selects the characters up to the 15th out of the string A$ in order to generate a new string. The string variables B$ has been defined for the new string.

To select some characters when counted from the right hand side of a string, RIGHT $ (   ) is used.

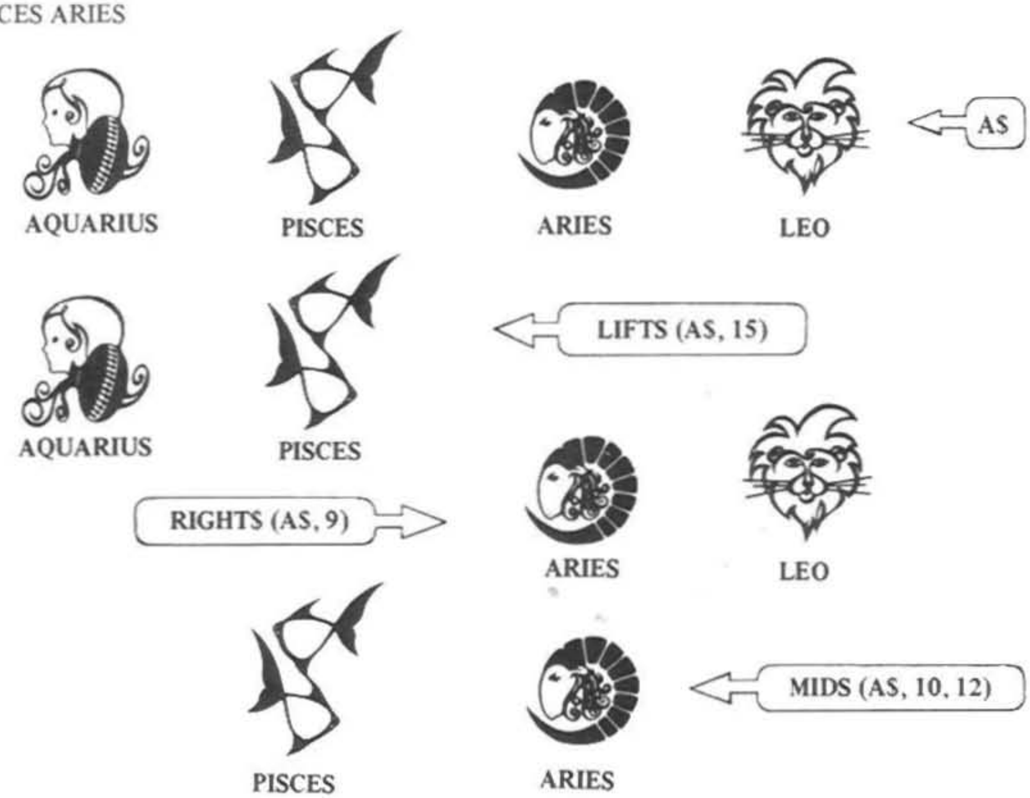
```
40  C$ = RIGHT $ (A$, 9)
50  PRINT C$
RUN
ARIES LEO
```

(Selects the last 9 characters from A$)

To select characters in the center of a string, MID$ (   ) is used.

```
60  D$ = MID$ (A$, 10, 12)
70  PRINT D$
RUN
PISCES ARIES
```

(Selects 12 characters starting at position 10 of A$)



■ LEN is a Measurement for Strings

LEN (   ) is used to discover the character count of a string. A simple example of this statement is as follows:


```
10  A$ = "ABCDEFG"
20  PRINT LEN (A$)
RUN
  7
```

The character count of a string variable A$, namely "7" is displayed.

Here is a program using LEN (   ) statement for drawing a square.

```
10  PRINT " C " : PRINT "TYPE HORIZONTAL SIDE USING * KEY"
20  INPUT A$
30  FOR J = 1 TO LEN (A$) – 2
40  PRINT TAB (2) ; " * " ; SPACE$ (LEN (A$) – 2) ; " * "
50  NEXT J
60  PRINT TAB (2) ; A$ : GOTO 20
```

Vary the values of * input. The computer performs square drawing by using LEN (   ). Then, drawing is made possible by characters or symbols other than " * ". Using LEFT $ (   ), line numbers 20 and 40 of the previous program are modified.

```
20  INPUT A$ : AA$ = LEFT $ (A$, 1)
40  PRINT TAB (2) ; AA$ ; SPACE $ (LEN (A$) – 2) ; AA$
```

The use of LEN makes a string parade possible.

```
10  S$ = "SHARP BASIC"
20  FOR M = 1 TO LEN (S$)
30  PRINT LEFT$ (S$, M)
40  NEXT M
RUN
S
SH
SHA
SHAR
SHARP
SHARP
SHARP B
SHARP BA
SHARP BAS
SHARP BASI
SHARP BASIC
```

```
10  S$ = "SHARP BASIC"
20  FOR M = 1 TO LEN (S$)
30  PRINT RIGHT$ (S$, M)
40  NEXT M
RUN
C
IC
SIC
ASIC
BASIC
 BASIC
P BASIC
RP BASIC
ARP BASIC
HARP BASIC
SHARP BASIC
```

ASC and CHR$ are Relatives

ASC


```
10  PRINT ASC ("A") ;
20  PRINT ASC ("ABC") ;
30  T$ = "Z" : PRINT ASC (T$)
40  END
RUN
  65 65 90
Ready
```

With strings in the parenthesis (   ) of ASC, when PRINT is keyed-in the result always shows numerals. Actually, this shows the ASCII code. All characters used with the computer are based on the ASCII code. For its table, refer to page 210. ASC (   ) picks up the ASCII code for the first character of string in the parenthnesis (   ). This gives a clear clue to the reason why the same result is obtained although the strings in the parentheses differ between line numbers 10 and 20. The ASCII code is for characters up to 255.

CHR$

If characters can be converted to the ASCII code, it is natural that there is a statement to reverse the conversion. That's right. CHR$ statement does that job.


```
PRINT CHR$ (65), CHR$ (ASC ("K"))
A          K
Ready
```

A cipher is generated using the numerals. Let CHR$ read it.

```
10  FOR J = 1 TO 24 : READ A
20  B$ = CHR$ (A)
30  PRINT B$ ; : NEXT : END
40  DATA 73, 32, 83, 84, 85, 68
50  DATA 89, 32, 66, 65, 83, 73
60  DATA 67, 32, 79, 70, 32, 77
70  DATA 90, 45, 56, 48, 65, 46
RUN
I  STUDY  BASIC  OF  MZ - 80A.
Ready
```

STR$ and VAL are Numeral Converters

STR $

```
10  A = 12 : B = 3 : C = A + B
20  C$ = STR$ (A) + STR$ (B)
30  PRINT C, C$
40  END
RUN
 15           123
Ready
```

The value of variable A is converted to a string of characters by STR$ (A) and string-processed. The reason why C$ contents are 123 is clear to you. In the following program, use STR$ to match the " . " of data.

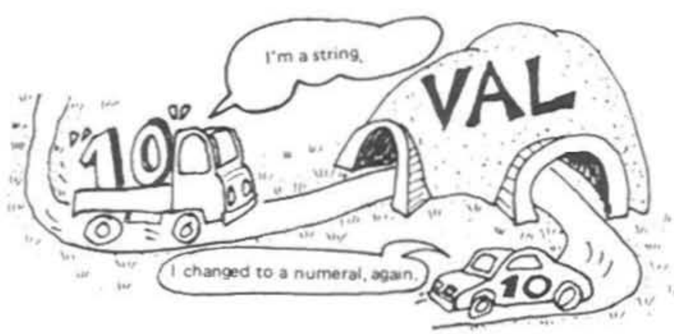
```
10  FOR X = 1 TO 5
20  READ A
30  L = 5 – LEN (STR$ (INT (A) ))
40  PRINT TAB (L) ; A
50  NEXT : END
60  DATA 1 . 2 3 4 5 6, 1 2 . 3 4 5 6
70  DATA 1 2 3 . 4 5 6, 1 2 3 4.5 6
80  DATA 1 2 3 4 5 . 6
```

The results of the program on the left are as follows.

```
    1. 2 3 4 5 6
   1 2. 3 4 5 6
  1 2 3. 4 5 6
 1 2 3 4. 5 6
1 2 3 4 5. 6
READY
```

VAL

VAL statement has a function opposite to STR $ statement. In other words, it converts a string of characters to a numeral.



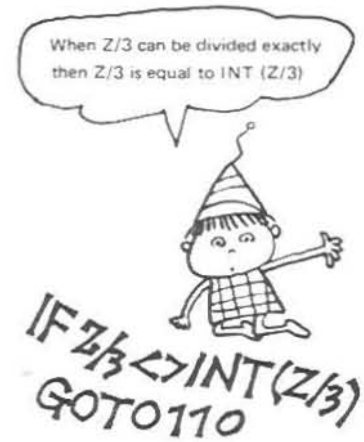

```
10  A$ = "1 2 3 4 5 6"
20  B = VAL (A$)
30  C = 6 5 4 3 2 1 + B
40  PRINT A$
50  PRINT B
60  PRINT C
80  END
RUN
1 2 3 4 5 6    ...............Not a numeral but a string.
 1 2 3 4 5 6   ...............Numeral, so there is a space for ± (plus/minus sign) to be placed
 7 7 7 7 7 7   ...............before the most significant digit of the numeral. For a netagive
READY                         numeral, a minus sign is placed in the space.
```

■ Print out as £123,456,789·········

This program reads an integer of an optional figure under the INPUT statement, and writes it adding commas (,) to every 3 figures from the right. Given 0 as an integer, the program terminates.

```
10  PRINT "INPUT INTEGER" ;
20  INPUT X$
30  IF X$ = "0" THEN END
40  PRINT "£" ;
50  FOR Y = 1 TO LEN (X$)
60  PRINT MID$ (X$, Y, 1) ;
70  Z = LEN (X$) – Y
80  IF Z/3 <> INT (Z/3) GOTO 110
90  IF Z = 0 GOTO 110
100  PRINT " , " ;
110  NEXT Y
120  PRINT : PRINT : GOTO 10
RUN
INPUT INTEGER ? 1 2 3 4 5 6 7 8 9
£ 123,456,789

INPUT INTEGER ? 1 2 3 4
£ 1,234
```



Line number 80 checks to see if Z (Character position counted from the right) is a multiple of 3. If so, a comma " , " is placed at line number 100. For example, presuming that the input integer is a number of 9 figures, the following is obtained.

Character count of INPUT integer
← LEN (X$) →

INPUT integer :	□	□	□	■	□	□	□	□	□
Y variable :	1	2	3	4	5	6	7	8	9
Z variable :	8	7	6	5	4	3	2	1	0

Z = LEN (X$) – Y

Take a number consisting of figures 1 to 4, and another number of the same figures but with a reverse arrangement to the former, then add up these two numbers. You will thus find that the sum is the same whether it is counted from the right or from the left.

```
10  PRINT " ■ ENTER SOME NUMBER COMPOSED OF FIGURES 1 TO 4 (WITHIN 8 DIGITS)"
20  Z$ = "" : INPUT X$
30  FOR K = LEN (X$) TO 1 STEP – 1
40  Y$ = MID$ (X$, K, 1)
50  Z$ = Z$ + Y$ : NEXT K : X = VAL (X$) + VAL (Z$)
60  PRINT X : PRINT : GOTO 20
```

Difference between the Simple and Compound Interests

£ 10,000 is deposited in a bank, and one year later, £ 10,600 is drawn. Interest rate, in this case, is found to be 600/1000 = 0.06 or 6%. Then, what is the interest when deposited for 2 years. There are two methods available for interest calculation. One is simple interest calculation based on the fact that the interest of £ 600 for the first year doubles for the second year, amounting to £ 1200. The other is compound interest calculation based on the idea that a deposit at the beginning of the second year is £ 10,600 with interest of £ 636 (£ 10,600 × 0.06) added to make £ 1236 for two years. Compound interest calculation is slightly better in interest rate. For a larger sum of money deposited for longer terms, the difference in interest rate between the two methods must be noticeable. The following is the equation for determining interest included for the n year in each calculation method.

Interest included by simple calculation (n year with rate R)

$$B = X\ (\text{principal}) + n \cdot X \cdot R$$

Interest included by compound calculation (n year with rate R)

$$C = X \cdot (1 + R)^n$$

Based on the above equations, the following program is generated to calculate interest included both in simple and compound interests.

```
10 PRINT "PRINCIPAL"
20 INPUT X
30 PRINT "INTEREST RATE %"
40 INPUT R
50 PRINT "NUMBER OF YEARS"
60 INPUT Y : PRINT : PRINT
70 PRINT "PRINCIPAL =" ; X
80 PRINT "INTEREST RATE =" ; R ; " % "
90 PRINT "YEARS" ; TAB (6) ; "SIMPLE" ;
100 PRINT TAB (17) ; "COMPOUND" ;
110 PRINT TAB (30) ; "DIFFERENCE"
120 FOR A = 1 TO Y
130 B = X + A * X * (R/100)
140 C = INT (10 * X * (1 + R/100) ↑ A)/10
150 D = C - B
160 PRINT A ; TAB (6) ; B ;
170 PRINT TAB (15) ; C ; TAB (30) ; D
180 NEXT A
190 PRINT : PRINT : GOTO 10
```

The following is an example of program execution:

```
PRINCIPAL = 10000
INTEREST RATE = 6%
YEARS SIMPLE  COMPOUND   DIFFERENCE
  1    10600   10600      0
  2    11200   11236      36
  3    11800   11910.1    110.1
  4    12400   12624.7    224.7
  5    13000   13382.2    382.2
  6    13600   14185.1    585.1
  7    14200   15036.3    836.29999
```

Annuity if Deposited for 5 years

In the previous example, we looked at the difference in interest between the simple and compound interest calculations for money deposited. Actually, however, monthly deposit, like fixed deposit, is more familiar to us. If a fixed amount of money X is deposited monthly, the interest included increases with $X(1 + R)$ for the first year, $X(1 + R)^2$ for the second and so on. In addition, when sum X is deposited yearly, the money to be deposited the year after, 2 years from now, will be $X(1 + R)$. Such an increase of deposits is shown below in equations:

Interest included a year after (Principal X and interest R)

$$M_1 = X(1 + R)$$

Interest included 2 years after

$$M_2 = X(1 + R)^2 + X(1 + R)$$

Interest included 3 years after

$$M_3 = X(1 + R)^3 + X(1 + R)^2 + X(1 + R)$$

Based on the above, the interest included is calculated in the following equation for n years.

$$M_n = X(1 + R)^n + X(1 + R)^{n-1} \ldots\ldots + X(1 + R)$$

This is simplified as follows:

$$M_n = X(\,(1 + R)^{n+1} - (1 + R)\,)/R$$

Here's the program generated to indicate what is the interest included for any desired year with the same amount of money deposited each year. Even though the same amount is deposited, this program is designed to allow inputs of minimum and maximum amounts.

```
10  PRINT "INTEREST RATE %" ;
20  INPUT R
30  PRINT "ENTER AMOUNTS"
40  PRINT "MINIMUM" ; : INPUT L
50  PRINT "MAXIMUM" ; : INPUT H
60  PRINT "NUMBER OF YEARS" ;
70  INPUT Y : PRINT : PRINT
80  PRINT "RATE" ; R ; "%"
90  PRINT "EACH YEAR" ; TAB (12) ; Y ; "YEARS"
100  R = R/100 : PRINT
110  FOR A = L TO H STEP 10000
120  B = INT (A * ((1 + R) ↑ (Y + 1) – (1 + R))/R)
130  PRINT A ; TAB (12) ; B
140  NEXT A
150  PRINT : PRINT : GOTO 10
```

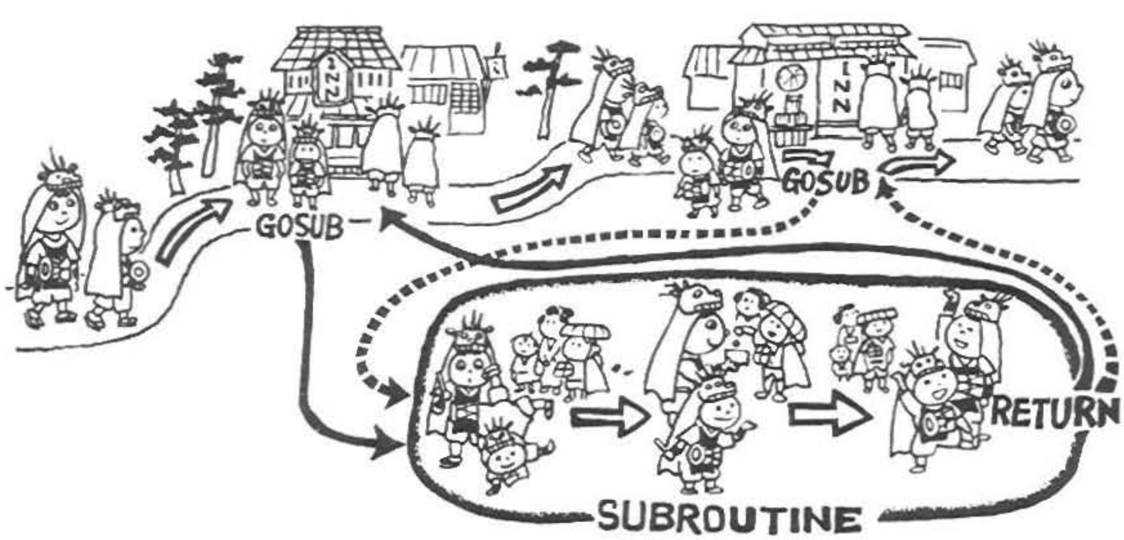
The Result of Program Execution

Subroutine is the Ace of Programs

In any program, jobs in the same procedure are repeated. Such jobs are summarized as a sub-program which can jump anytime. This sub-program in called a **subroutine**, for which the GOSUB statement is used. The following is an example of a program using the GOSUB statement.


```
10 PRINT " ◙ "
20 PRINT TAB (10) ; " ＊＊  TOTAL SALES ＊＊  " : PRINT
30 PRINT "            5         10        15        20        25        30        35"
40 PRINT " ┌────┬────┬────┬────┬────┬────┬────┐"
50 PRINT " ┐     FOUR" : GOSUB 200
60 PRINT " ┐     SUGER" : GOSUB 200
70 PRINT " ┐     WINE" :  GOSUB 200
80 PRINT " ┐     SAUCE" : GOSUB 200
110 PRINT " ┐ "
120 PRINT : END
200 PRINT " ┐" ; : READ A
210 FOR N = 1 TO A : PRINT " ▩ " ; : NEXT N
220 PRINT A
230 RETURN
240 DATA 20, 15, 21, 24
```

In the above program, subroutines are line numbers 200 through 230. By the RETURN statement at the end of subroutines, the program execution returns to the main program.



■ Stop, Check and Continue

The computer does not always work as desired when operated with a program generated. This requires a STOP statement to be inserted to check the contents of variables at the stop position. For example, in the following program, the STOP statement is inserted.

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

At the time, the display of variables is made in direct mode as follows:

PRINT A,B, X [CR]

This enables you to check the program. To re-start the program, give a command to the computer as follows:

CONT [CR]

The CONT command is used when ;

- Program execution is stopped with the [SHIFT] – [BREAK] keys.
- Program execution is stopped by the STOP or END statement.
- Inputs are stopped at the INPUT statement using the [SHIFT] – [BREAK] keys.

The CONT command cannot be used ;

- Before program has been executed using the RUN command.
- When program is edited after execution has been stopped.
- If an error occurs during execution. Program returns to the "Ready".
- To stop cassette tape operation, cassette tape operation is stopped with the [SHIFT] + [BREAK] keys.
- When the MUSIC command for music sound is stopped.

The computer restarts execution from the stop position. With the END statement at line number 50, the computer displays the "Ready" and stops again. Then, print in the direct mode, as follows:

X = 3 : Y = 5 [CR]

The computer will then continue program execution when the CONT command is given, displaying 3 and 5 for variables X and Y.

The following program continues to display a triangle of * marks, indefinitely.

```
10  A = 0 : B = 38 : C = 1
20  FOR X = A TO B STEP C
30  FOR Y = 0 TO X
40  PRINT "*";
50  NEXT Y : PRINT : NEXT X
60  K = A : A = B : B = K
70  C = –C : GOTO 20
```

With the [BREAK] key pressed while holding down the [SHIFT] key, program execution stops. Then, insert the following in the program and give the CONT command.

```
100  END
```

This is followed by the display below.

* Error 17 . . . . CONT command cannot be executed.

The CONT command cannot be used when a program is edited using a line number after program execution has been stopped with the STOP statement, END statement or [BREAK] key operation. This requires special attention.

■ Jump in masse Using the ON ..... GOTO Statement

You have learnt much about the GOTO statement. Description here is given of the On . . . . GOTO statement, an extended function of the GOTO statement.

ON [variable or expression] **GOTO** [Line No.] , [Line No.] , [Line No.] , ...

This value is 1. → first Line No.
This value is 2. → second Line No.
This value is 3. → third Line No.
. . . .

For example, when the value of a variable or expression after ON is 3, a jump is effected to the third line number that follows GOTO. In other words, it is possible to assign the branch of a program in accordance with the values of variables.

```
10  INPUT "NUMBER (1 - 3) ?" ;A
20  ON A GOTO 50, 60, 70
50  PRINT "X X X" : GOTO 10
60  PRINT "Y Y Y" : GOTO 10
70  PRINT "Z Z Z" : GOTO 10
RUN
NUMBER (1 - 3) ? 1
X X X
NUMBER (1 - 3) ? 2
Y Y Y
NUMBER (1 - 3) ? ▩
```

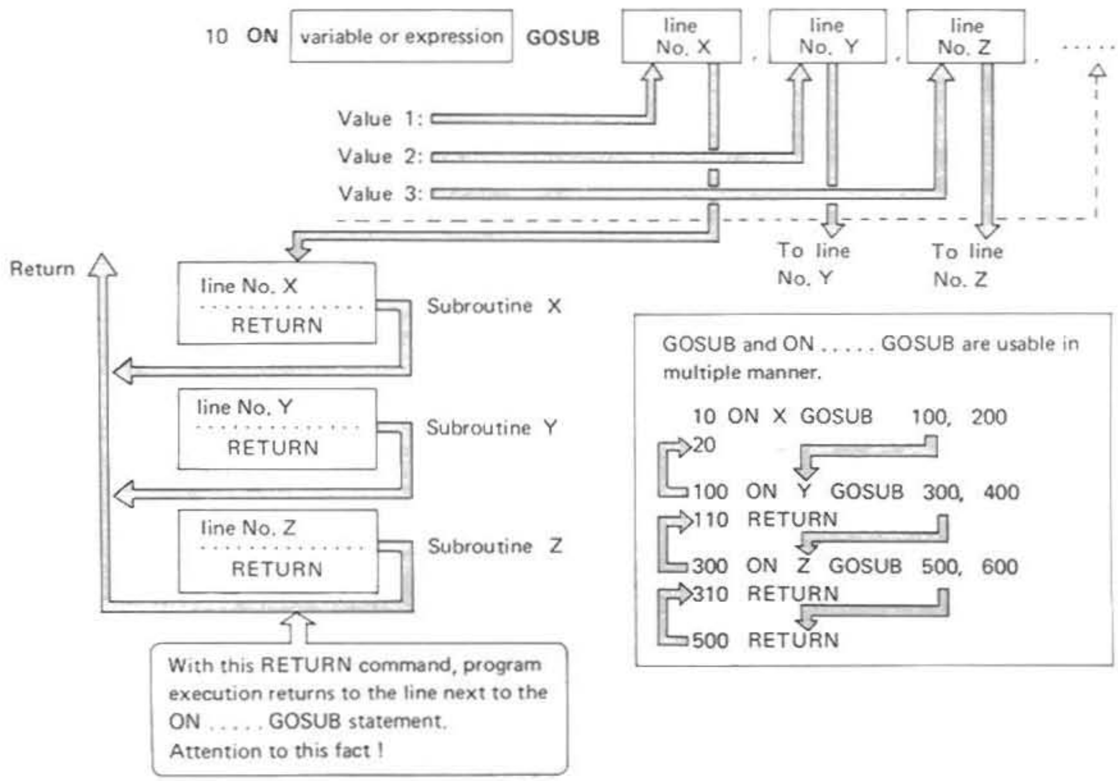
Given 1.2, for example, integer 1 is processed, cutting off any place of decimals.

Let's play a joker-pick game using the ON . . . . GOTO statement. A joker is included in 5 cards. The place of the joker is unknown, of course. Guess where the joker is to compete with the computer for a win. When asked "Do you pass?" in the program, key-in 1 for pass, 2 for not pass and 3 if the game is not to be played. Three passes are allowed.

```
10  R = INT (5 * RND (1)) + 1
20  N = N + 1 : IF N = 6 THEN 120
30  INPUT "DO YOU PASS?" ;X
40  ON X GOTO 60, 90, 50
50  PRINT "GAME NOT PLAYED !!!" : GOTO 120
60  NP = NP + 1
70  IF NP >= 4 THEN NP = NP - 1 : N = N - 1 : PRINT "NO PASS ALLOWED"
80  GOTO 20
90  NR = NR + 1
100  IF R = NR + NP THEN PRINT "UNLUCKY ! YOU HAVE SELECTED THE JOKER" : GOTO 120
110  PRINT "LUCKY ! YOU HAVE NOT SELECTED THE JOKER" : GOTO 20
120  END
```

ON.....GOSUB is the Use of a Subroutine Group

The ON . . . . . GOSUB statement is very similar in function to the ON . . . . . GOTO statement.



Now, let's consider the program for a time table to check your progress. Most important in the following program is that subroutines are called at line number 180, despite the jump made at line number 90 to line number 170 through 190 of subroutines.

Thus, the GOSUB and ON . . . . . GOSUB statement can be used in a convenient, multiple manner.

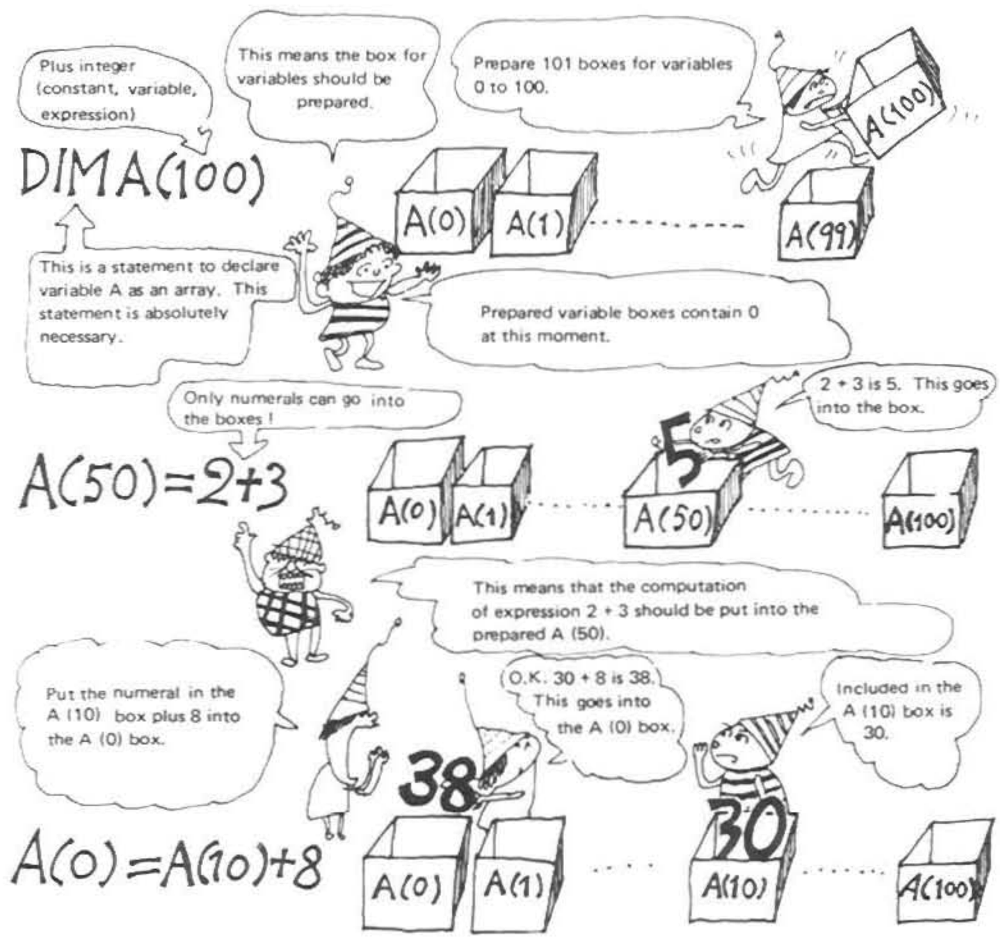
```
10  A$ = "FRENCH   " : B$ = "MATHEMATICS" : C$ = "ENGLISH"
20  D$ = "SCIENCE   " : E$ = "MUSIC       " : F$ = "ATHLETICS"
30  G$ = "SOCIAL STUDIES" : H$ = "ART           " : I$ = "TECHNOLOGY"
40  J$ = "RELIGION   " : K$ = "ECONOMICS"
50  PRINT "WHAT DAY OF THE WEEK ? "
55  PRINT "(1 - MON, 2 - TUE, 3 - WED, 4 - THU, 5 - FRI, 0 - ALL)"
60  INPUT X$ : X = ASC (X$) – 47
70  FOR Y = 0 TO 3 : PRINT TAB (3 + 8 * Y) ; Y + 1 ;
80  NEXT Y : PRINT
90  ON X GOSUB 170, 110, 120, 130, 140, 150
100  PRINT : GOTO 50
110  PRINT "MON : " ; "A$ : C$ ; D$ ; B$ : RETURN
120  PRINT "TUE : " ; H$ ; H$ ; E$ ; B$ : RETURN
130  PRINT "WED : " ; A$ ; C$ ; J$ ; K$ : RETURN
140  PRINT "THU : " ; D$; A$ ; E$ ; F$ : RETURN
150  PRINT "FRI : " ; A$ ; D$ ; I$ ; G$ : RETURN
170  FOR Y = 1 TO 5
180  ON Y GOSUB 110, 120, 130, 140, 150
190  PRINT : NEXT Y
200  RETURN
```

Primary Array has the Strength of 100 Men

Now, consider the substitution of variables for 100 items of data. The use of variables A1 and A2 makes the following possible.

```
10 A1 = 5
20 A2 = 30
30 A3 = 12
...........
```

Just a minute. This is terribly hard work for writing 100 statements! For this, the primary array is available as a new type of variable, which makes program generation very convenient. Now, let's look at what the primary array is all about.



Now, you have understood what the primary array is, haven't you?

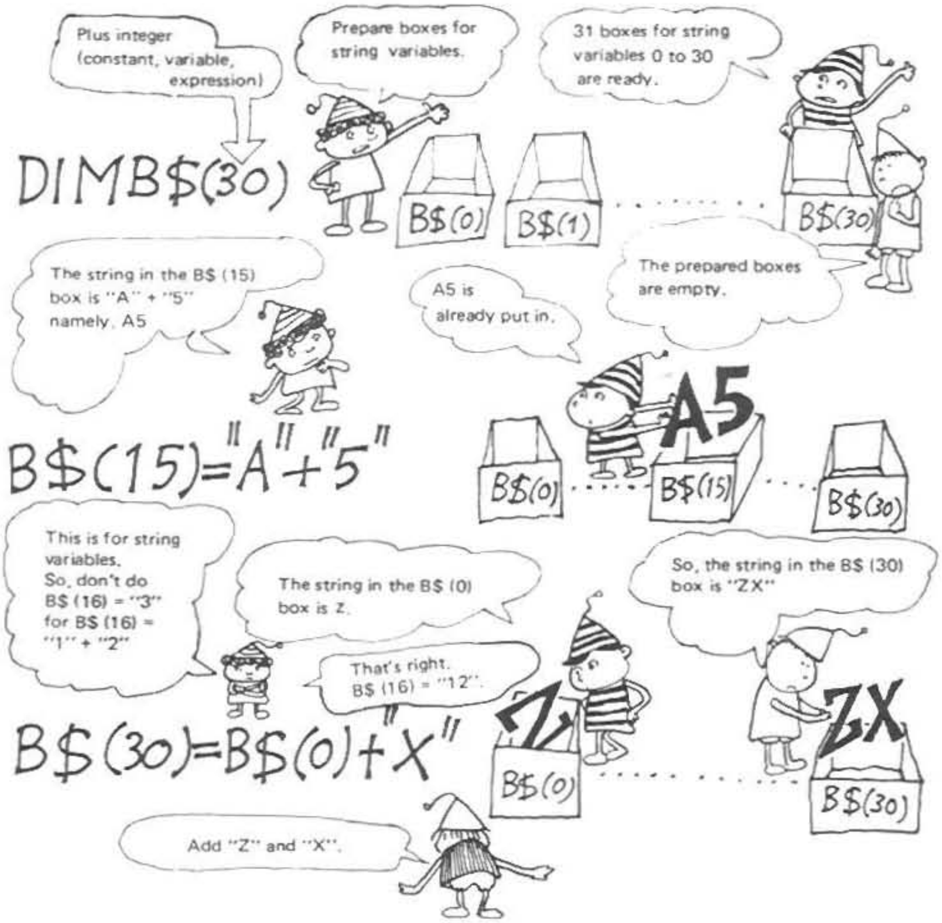
Using the primary array, the program has been generated as follows:

```
10 DIM A (100)
20 FOR J = 1 TO 100
30 READ A (J)
40 NEXT J
50 DATA 5, 30, 12, ........
```

See, the program is very short. As is clear from this example, variables in the form of an array can assign the parenthesis of subscribed variables, such as A (J), with variable J. This is the main feature of the primary array.

Array is also Available for String Variables

Since an array is available for numeral variables, there must be an array available even for string variables. Here's an introduction to what the primary array for string variables is all about.



Let's generate a simple program. Just a look at this. Keeping variable strings in the form of arrays eliminates the labour of writing whenever they are used. The program itself is neat and simple.

```
10  DIM A$ (2), B$ (2), C$ (2)
20  FOR J = 1 TO 2 : READ A$ (J), B$ (J)
30  C$ (J) = A$ (J) + " " + B$ (J)
40  PRINT A$ (J), B$ (J), C$ (J)
50  NEXT J
60  END
70  DATA YOUNG, GIRL, WHITE, ROSE

RUN
YOUNG       GIRL        YOUNG GIRL
WHITE       ROSE        WHITE ROSE
Ready
```

■ Array is the Master of File Generation

Some teachers say that testing is all right but putting test results in order is really hard. If so, some students insist testing should be stopped. A good method is available for teachers who are subject to giving tests to students.

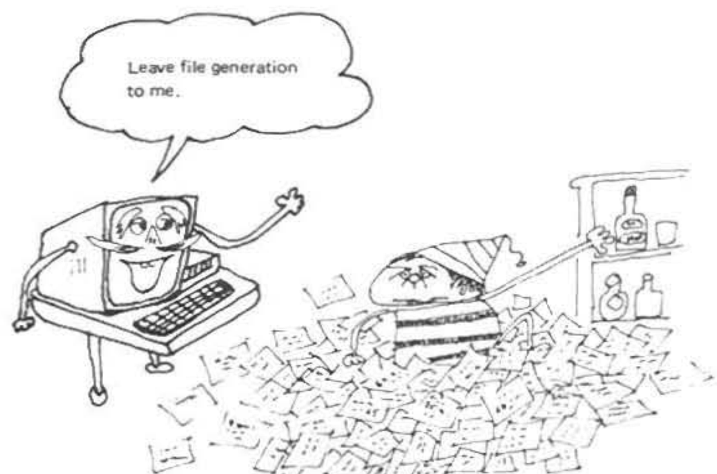
The use of an array helps them solve the problem! The following shows student identification and marks for mathematics.

Student No.	20	15	12	40	23	16	31	45	26	11
Marks	75	51	28	56	100	81	60	43	66	48

Generate a file program arranged in the order of merit.

```
10  DIM A (10), B (10)
20  FOR J = 1 TO 10
30  READ A (J), B (J) : NEXT
40  FOR K = 1 TO 9 : M = 0
50  FOR J = K TO 10
60  IF B (J) < = M THEN 80
70  M = B (J) : L = J
80  NEXT J
90  B (L) = B (K) : B (K) = M
100  A1 = A (L) : A (L) = A (K) : A (K) = A1
120  NEXT K
130  PRINT " [C] "
140  PRINT "ORDER OF MERIT (MATHEMATICS)"
150  PRINT
160  PRINT "STUDENT NO." ; TAB (14) ;
170  PRINT "MARKS"
180  FOR J = 1 TO 10
190  PRINT A (J) ; TAB (14) ; B (J) : NEXT J
200  END
210  DATA 20, 75, 15, 51, 12, 28, 40, 56, 23, 100
220  DATA 16, 82, 31, 60, 45, 43, 26, 66, 11, 48
RUN
ORDER OF MERIT (MATHEMATICS)

STUDENT No.        MARKS
    23              100
    16               81
    20               75
    26               66
    31               60
    40               56
    15               51
    11               48
    45               43
    12               28
Ready
```



■ Challenge of French Study

We used to study french words using word-notebooks. Smart and more simplified word-notebooks are available using the computer. French words and their meanings are contained in separate files. The computer gives two types of questions; one asking about the meanings of French words retrieved from the file and the other asking English to be translated to French. In the program, the primary string array is used as the files containing Frech words and their meanings. Executing the following program, try to test your French vocabulary, answering a variety of questions the computer will ask you.


```
10 DIM A$ (10), B$ (10), C$ (10)
20 FOR J = 1 TO 10
30 READ A$ (J), B$ (J)
40 C$ (J) = A$ (J) + B$ (J)
50 NEXT J
60 K = INT (10 * RND (1) ) + 1
70 PRINT " [C] WHAT IS MEANING OF THE WORD ? "
80 PRINT A$ (K),
90 INPUT X$
100 AX$ = A$ (K) + X$
110 IF C$ (K) = AX$ THEN PRINT "O.K. !! " : FOR M = 1 TO 3000 : NEXT : GOTO 150
120 PRINT "WRONG" : FOR M = 1 TO 1000 : NEXT M
130 PRINT " [↑] " ; SPACE$ (10) : PRINT " [↑] [↑] " ; TAB (12) ; SPACE$ (25)
140 PRINT " [H] " : GOTO 80
150 K = INT (10 * RND (1) ) + 1
160 PRINT " [C] TRANSLATE TO FRENCH"
170 PRINT B$ (K),
180 INPUT Y$
190 YB$ = Y$ + B$ (K)
200 IF C$ (K) = YB$ THEN PRINT "O.K.!!" : FOR M = 1 TO 3000 : NEXT M : GOTO 60
210 PRINT "WRONG" : FOR M = 1 TO 1000 : NEXT M
220 PRINT " [↑] " ; SPACE$ (10) : PRINT " [↑] [↑] " ; TAB (12); SPACE$ (25)
230 PRINT " [H] " : GOTO 170
240 END
250 DATA CHAT, CAT, PORTE, DOOR, MAISON, HOUSE, CHIEN
260 DATA DOG, CANARD, DUCK, POISSON, FISH, MAIN, HAND
270 DATA FENETRE, WINDOW, FILLETTE, GIRL, FEMME
280 DATA WIFE
RUN
WHAT IS MEANING OF THE WORD ?
POISSON        ?
```

In this case, the question about the meaning of poisson is answeres by keying-in that English. Display of O.K.!! is on the CRT screen to indicate you are correct. For any other answer, error display is made. Conversely, furthermore, there is the case when you answer "POISSON" when asked about translation.

Secondary Array is More Powerful

Let's look at this table (bottom right) which is an improvement on the test result table (bottom left) of mathematics, English and French for 3 students.

Name / Subject	John	Peter	Paul
Mathematics.	92	75	72
English	70	94	78
French	65	60	95

	Student	John	Peter	Paul
Subject	N / M	1	2	3
Mathematics.	1	A (1, 1)	A (1, 2)	A (1, 3)
English	2	A (2, 1)	A (2, 2)	A (2, 3)
French	3	A (3, 1)	A (3, 2)	A (3, 3)

M = 1 . . . Mathematics
M = 2 . . . English
M = 3 . . . French

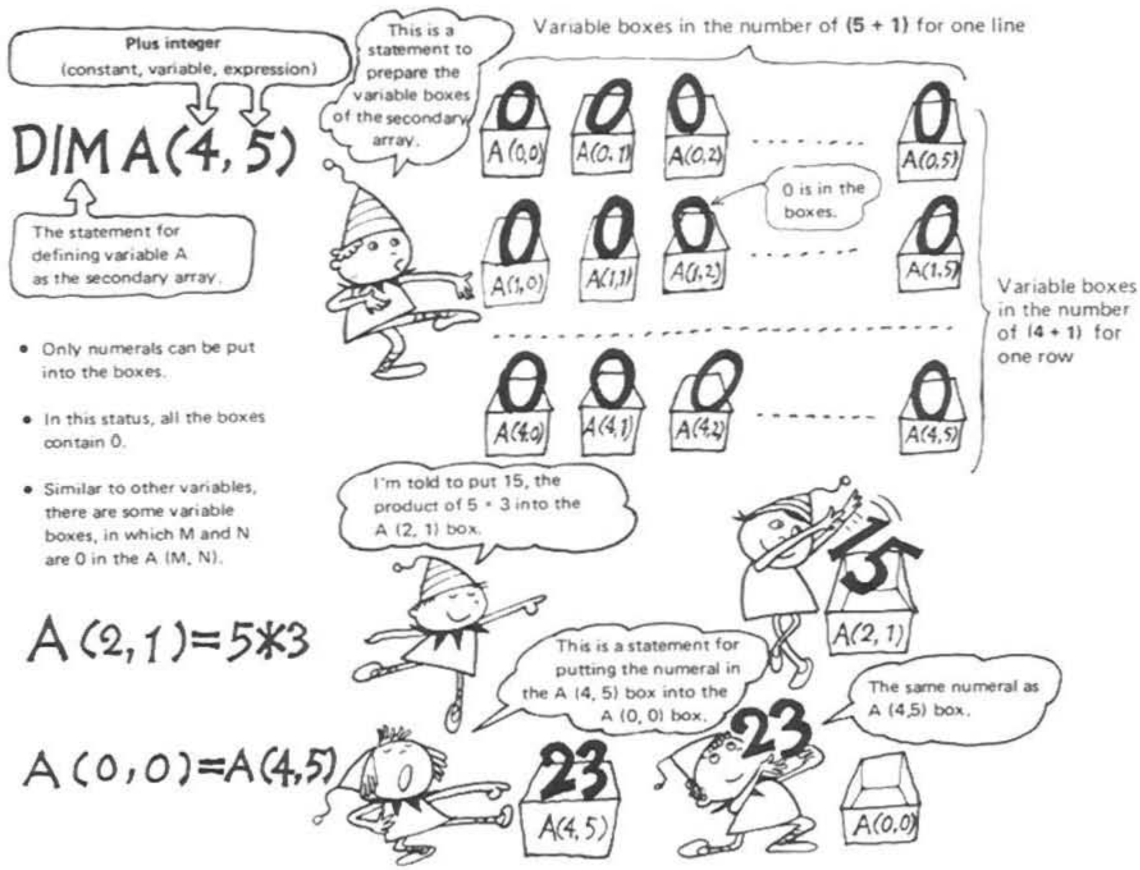
N = 1 . . . John
N = 2 . . . Peter
N = 3 . . . Paul

In the table at right, the subject, student and marks are expressed as M (1 - 3), N (1 - 3) and A (M, N), respectively. This is very convenient, for example, as is evident in the following:

A (2, 3) · · · · M = 2 means English, N = 3 means Paul · · · · · English marks of Paul

Simple! Writing A (2, 3) alone gives a clear description of the English mark of Paul. M and N in the A (M, N) represent separate items. Writing A (M, N) using two items is called the secondary array. Two items used mean secondary array. The primary array previously described has one item.

Now, look at how this secondary array can be used in the program for the computer.



Two Exercises

Determine the value and curve of SIN when an angle varies from 0 degree to 180 degrees with 10 degree increments.

```
10 PRINT " ◙➡ " ; TAB (5) ; "SIN"
20 PRINT
30 FOR K = 0 TO 180 STEP 10
40 X = K * π/180 · · · · · Value in degree unit is
50 S = SIN (X)               changed to radian.
60 A = INT (10 * S)
70 PRINT K ; TAB (4) ;
80 PRINT S ; TAB (18 + A) ;
90 FOR J = 0 TO A
100 PRINT " * " ;
110 NEXT J
120 PRINT
130 NEXT K
140 END
```

Execute the above program. This graph is rather rough. As for drawing graphs, a lot of examples will be described later so that you can learn more about them.

Now, let's generate a program to determine the prime numbers. A prime number is the one that cannot be divided by any integer smaller than itself, except for 1. Since the first prime number is 2, the multiples of 2, namely, even number larger than 2 are not prime numbers. To use variables with subscripts effectively, even numbers are excluded from the start.

```
10 DIM P (255)
20 FOR J = 0 TO 255
30 P (J) = J * 2 + 3 : NEXT J
```

Substitute 256 odd numbers from 3 to 513 for the parenthesis of variable P with a subscript.

```
40 FOR K = 0 TO SQR (255)
50 IF P (K) = 0 THEN 90
60 KK = K + P (K)
70 FOR L = KK TO 255 STEP P (K)
80 P (L) = 0 : NEXT L
90 NEXT K
```

Find prime numbers from the small in value and substitute 0 for the values of the multiples in the parenthesis of P.

```
100 PRINT 2 ;
110 FOR M = 0 TO 255
120 IF P (M) = 0 THEN 140
130 PRINT P (M) ;
140 NEXT M
150 END
```

The only even number "2" is first displayed, and then the values of P (   ) which are not 0, namely, prime numbers are on display.

This program is a bit complex, isn't it? Note that the multiples of the prime number are excluded from the start. Details on structured programming of prime numbers will be described later.

Here's Advice on how Lists can be made

Names are sorted out when making a list of members. The use of a convenient program, if any, facilitates listing of any kind.

Here you learn how to sort strings for address books, telephone numbers or housekeeping account books.

```
10 PRINT "HOW MANY PERSONS ARE SORTED?"
20 INPUT X
30 DIM N$ (X)
40 PRINT "KEY – IN NAMES ONE BY ONE"
50 PRINT "BUT IF 0, JOB DISCONTINUED!"
60 FOR A = 1 TO X : A$ = STR$ (A)
70 PRINT "NAME PLEASE " ; " ( " ; A$ ; " ) "
80 INPUT N$ (A) ———— Name is keyed-in.
90 IF N$ (A) = "0" THEN 110
100 NEXT A
110 A = A – 1
120 FOR B = 1 TO A – 1
130 FOR C = 1 TO A – B
140 D = LEN (N$ (C)) : E = LEN (N$ (C + 1)) : F = 1 : IF D < E THEN E = D
142 X = ASC (MID$ (N$ (C), F, 1))
143 Y = ASC (MID$ (N$ (C + 1), F, 1)) : IF X > Y THEN 150
144 IF X < Y THEN 180
145 IF (E = F) * (D = E) THEN 180
146 IF (E = F) * (D > E) THEN 150
148 F = F + 1 : GOTO 142
150 K$ = N$ (C)          }
160 N$ (C) = N$ (C + 1)  }  The order is substituted.
170 N$ (C + 1) = K$      }
180 NEXT C, B
190 PRINT
200 FOR B = 1 TO A
210 PRINT N$ (B) ———— Result is displayed.
220 NEXT B
230 PRINT : END
```

Original List (Keyed-in)		Sorted List
TOM BROWN	⇨	ANNE MILLER
HAROLD GREEN		ELICE THOMAS
JIM JONES		HAROLD GREEN
ANNE MILLER		JIM JONES
TOM CARTER		TOM BROWN
ELICE THOMAS		TOM CARTER

■ Cards if Dealt by a Poker Player

The computer deals cards for you. It shuffles them correctly using random numbers, causing no trickery to occur.

```
10  DIM X (4, 13)
20  C = 0
30  PRINT : FOR A = 1 TO 5
40  GOSUB 90 : PRINT : NEXT A : PRINT
50  PRINT "IS YOUR HAND ALRIGHT WITH THESE CARDS?"
60  INPUT "ALL RIGHT (1), GIVE ME NEXT (2) ?" ; A
70  ON A GOTO 400, 30
80  GOTO 50
90  C = C + 1 : IF C = 51 THEN 500
100  M = INT (4 * RND (1)) + 1
110  N = INT (13 * RND (1)) + 1
120  IF X (M, N) = -1 THEN 100
130  X (M, N) = -1
140  IF N = 1 THEN PRINT "ACE : " ; : GOTO 180
150  IF N = 10 THEN PRINT N ; TAB (5) ; " : " ; : GOTO 180
160  IF N < 10 THEN PRINT N ; TAB (5) ; " : " ; : GOTO 180
170  ON N - 10 GOTO 200, 210, 220
180  ON M GOTO 300, 310, 320, 330
200  PRINT "JACK : " ; : GOTO 180
210  PRINT "QUEEN : " ; : GOTO 180
220  PRINT "KING : " ; : GOTO 180
300  A$ = " ♠ " : GOTO 340
310  A$ = " ♥ " : GOTO 340
320  A$ = " ♦ " : GOTO 340
330  A$ = " ♣ " : GOTO 340
340  FOR B = 1 TO N
350  PRINT A$ ;
360  NEXT B
370  RETURN
400  PRINT
410  PRINT "THEN I RESHUFFLE."
420  FOR M = 1 TO 4 : FOR N = 1 TO 13
430  X (M, N) = 0
440  NEXT N, M : GOTO 20
500  PRINT
510  PRINT "TWO CARDS REMAIN . . . DO YOU CONTINUE ?"
520  INPUT "YES (1), NO (2) ? " ; B
530  ON B GOTO 400, 550
540  GOTO 510
550  END
```

Points of the program:

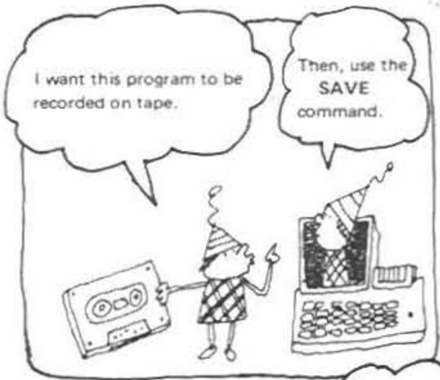
Line number 100 and 110 . . . . Turning up a new card.

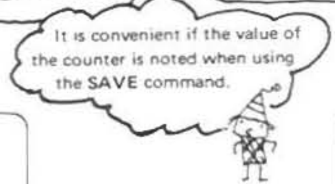
Line number 120 . . . . If a newly turned up card has been previously turned up, another card is to be turned up.

Line number 130 . . . . Mark dealt cards with "-1".

Line numbers from 420 to 440 . . . . All cards are collected and their marks are returned to "0".

■ Program Recording (SAVE)





SAVE [CR] key operation

SAVE "NAME" [CR] key operation

The name should be no more than 16 characters.

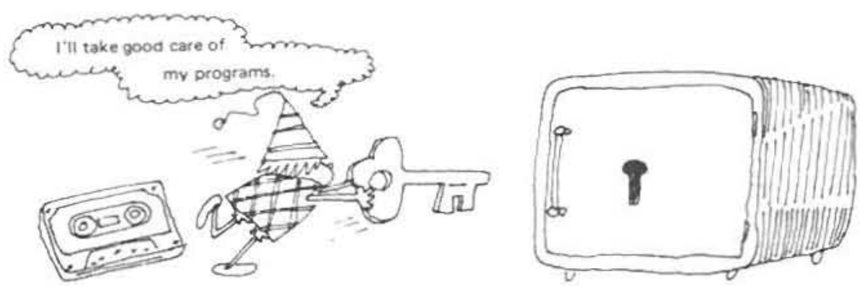
CRT screen display — ↓ RECORD · PLAY — Preparations for recording are O.K. !!

Cassette tape recorder operation — Press the [RECORD] and [PLAY] buttons — Recording start!!

CRT screen display — Writing "NAME" — Recording in progress

No name is displayed with no file name given to a program.

CRT screen display — Ready — Recording completed



Use of VERIFY and LOAD Commands

Verify

(1) Rewinding the tape. ⇨ (2) VERIFY "NAME" [CR]
"Name" is not required if a file name is not given.

(3) ↓ PLAY. . . CRT screen display. ⇨ (4) [PLAY] button pressed. Cassette tape recorder operation.

(5) Found "＊＊＊＊" . . . . . This is on display if another program is found. Its file name, if given, is displayed as ••••.
•••••(file name)

⇩

(6) Found "NAME" . . . . Display indicating that the program to be verified was found.

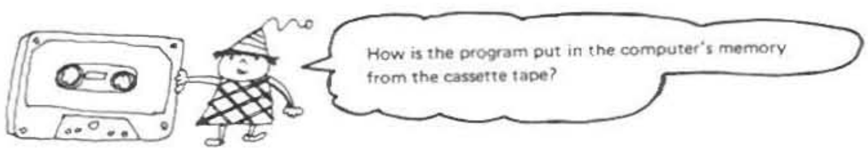
⇩

(7) Verifying "NAME"
Checking that recording is correctly done is in progress.

⇦ (8) CHECK SUM ERROR
＊ Error 70
Ready
meaning an error in recording. When this is on display, do **SAVE** once again.

⇨ (8) OK
Ready
With this display, recording is verified.

Load



(1) LOAD "NAME" [CR]

(2) ↓ PLAY

(3) Press the [PLAY] button.

(4) Found "＊＊＊＊"

(5) Found "NAME"

(2)–(5): Display and operation are same as the steps (3) to (6) of the VERIFY command.

(6) Loading "NAME" — Transfer to the computer in progress (loading).

(7) Ready — Loading completed.

Data can also be Stored on Cassette Tape

Data storage is also required if programs can be stored .....
To do so, 5 more statements must be leaned. Then, a cassette tape can be used as a storage of data.

WOPEN/T .... This prepares for data writings. It also serves to name a group of data.
PRINT/T .... Identical in use to the PRINT statement, this writes data on a cassette tape.
ROPEN/T .... This statement prepares for data readouts. It serves to find a data group with the name given.
INPUT/T .... Identical in use to the INPUT statement, this reads data out of the cassette tape.
CLOSE/T .... This statement must be executed before ROPEN if WOPEN is executed or before WOPEN if ROPEN is executed.

To store data, numerals from 1 to 99 are first written on a cassette tape. The "DATA" at line number 10 is the name given to a group of data to be written. A maximum of 16 characters can be used to name a group of data. Of course, it is unnecessary to have a name if so desired.

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

Now, it is time to read the data which has just been written. First, rewind the cassette tape, then execute the following:

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

Whey not execute the above program again with 100 substituted for 99 at line number 20 ? An error will occur this time. Because the 100th data was not originally written. It is impossible to memorize the written data counts. For this, a numeral, for example, –99999999 unrelated to that use for data is written as a mark at the end of written data.

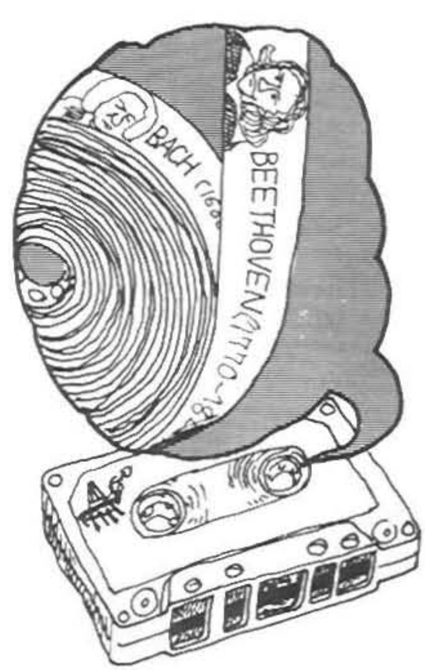
```
10 WOPEN/T "DATA"
20 FOR X = 1 TO 99
30 PRINT/T X
40 NEXT X
50 PRINT/T -99999999
60 CLOSE/T
70 END
```

```
10 ROPEN/T "DATA"
20 FOR X = 1 TO 200
30 INPUT/T A
40 IF A = -99999999 THEN 70
50 PRINT A
60 NEXT X
70 CLOSE/T
80 END
```

Technique to Memorize a Music History

Statements for data storage and readouts can also be used for strings.
The five composer's names are written on the cassette tape and read out of it.

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



This is identical to numeric data writing. Then, readouts are done as follows:

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

With this, writing and readout are completed. As you may have noticed, the name of string variable N$ used for writing is different from that of string variable M$ used for readouts. Since the value itself is written in the cassette tape as data, it has nothing to do with the name of the substituted variable. This makes it possible to change the variable name in the program as long as the string data is read by the string variable and the numeral data by the numeral variable.

Now, from what you have learnt so far, let's generate a data file with mixed numeric and string data. To also write the years when the previous composers died, for example, the following statements should be modified from the previous program.

```
15 DIM D (5)
65 D (1) = 1750 : D (2) = 1791 : D (3) = 1827
67 D (4) = 1849 : D (5) = 1897
90 PRINT/T N$ (J), D (J)
```

It is clear from the above that the generated file stores string and numeric data in alternate sequence. Accordingly, the readouts of the file must match the alternate sequence, for which line numbers 200, 230 and 240 should be modified as follows:

```
200 DIM M$ (5), T (5)
230 INPUT/T M$ (K), T (K)
240 PRINT M$ (K), T (K)
```

With those statements remaining unmodified, the numeric data is transferred to the string variable M$ ( ), causing an error to occur.

List of School Work Results

This is a program for recording the results of French, English and science for a certain class.

```
10 INPUT "HOW MANY STUDENTS IN THE CLASS? " ; N
20 DIM N$ (N), K (N), E (N)
30 DIM R (N)
40 A$ = " (MARKS) "
50 FOR X = 1 TO N
60 PRINT : PRINT X
70 INPUT "NAME PLEASE ? " ; N$ (X)
80 PRINT "FRENCH " ; A$ ; : INPUT K (X)
90 PRINT "ENGLISH" ; A$ ; : INPUT E (X)
100 PRINT "SCIENCE" ; A$ ; : INPUT R (X)
120 NEXT X
130 WOPEN/T " RESULT "                   <---- For writing a data group named "RESULT".
140 PRINT/T N                            <---- Writing the number of students in the class.
150 FOR X = 1 TO N
160 PRINT/T N$ (X), K (X) , E (X), R (X) <---- Writing the marks for students.
170 NEXT X
180 CLOSE/T                              <---- Writing completed.
```

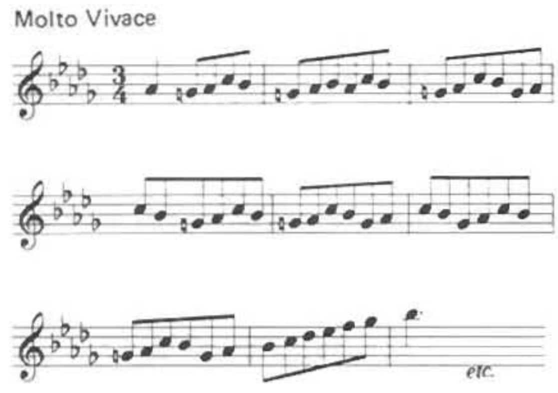
Now, let's read the written data of results, and calculate the mean of individual students' points and the mean of each subject.

```
10 ROPEN/T " RESULT "            <---- For finding the data group named " RESULT ".
20 INPUT/T N                     <---- Readouts of the number of students in the class.
30 DIM N$ (N), K (N), E (N)
40 DIM R (N)
50 FOR X = 1 TO N
60 INPUT/T N$ (X), K (X)         <---- Readouts of the name and marks for French.
70 INPUT/T E (X), R (X)          <---- Readouts of marks for English and science.
80 NEXT X
90 CLOSE/T                       <---- Readouts completed.
100 PRINT TAB (12) ; " FRENCH " ;
110 PRINT TAB (19) ; "ENGLISH" ;
120 PRINT TAB (27) ; "SCIENCE" ;
130 PRINT TAB (34) ; "MEAN"
140 FOR X = 1 TO N
150 PRINT N$ (X) ; TAB (11) ; K (X) ;
160 PRINT TAB (18) ; E (X) ;
170 PRINT TAB (26) ; R (X) ;
190 PRINT TAB (33) ; INT (10/3 * (K (X) + E (X) + R (X))) / 10
200 K (0) = K (0) + K (X) : E (0) = E (0) + E (X)
210 R (0) = R (0) + R (X)
220 NEXT X : PRINT " MEAN" ;
230 PRINT TAB (11) ; INT (10 * (K (0) / N)) / 10 ;
240 PRINT TAB (18) ; INT (10 * (E (0) / N)) / 10 ;
250 PRINT TAb (26) ; INT (10 * (R (0) / N)) / 10
260 END
```

Music Library Kept on Tapes

This data file is indispensable to generate a "Music Library" as discussed in the paragraph "MUSIC Statement".

Data for tunes is string data consisting of various symbol groups. If a data group is named per tune, any tune can be picked out of those recorded on the tape when its name is designated.

For example, a tune can be picked up from this music library for use in the music box of your timer, with some modifications. The tunes in the music library can also be used for programs of games and graphics, providing a number of applications.

F. Chopin " Puppy Waltz "

To write the etude of F. Kroepsch used on page 79 into a data file, the following changes must be made:

```
300  WOPEN/T "ETUDE"
310  PRINT/T J1$, J2$, J3$, N1$, N2$, N3$
320  CLOSE/T
```

Attention is required to the fact that the character count for data writing should be within 255 characters. If written as follow;

```
305 MA$ = J1$ + J2$ + J3$ : MB$ = N1$ + N2$ + N3$
310  PRINT/T MA$, MB$
```

the contents of string variables MA$ and MB$ exceed 255 characters, which make data writing incomplete.

The length of string data may vary with each tune, therefore it is neccessary to write data indicating the end of each tune so that a data error does not occur in data readouts. End mark of tune " ◧■ ", for example makes each tune consist of string data within 100 characters for execution give below:

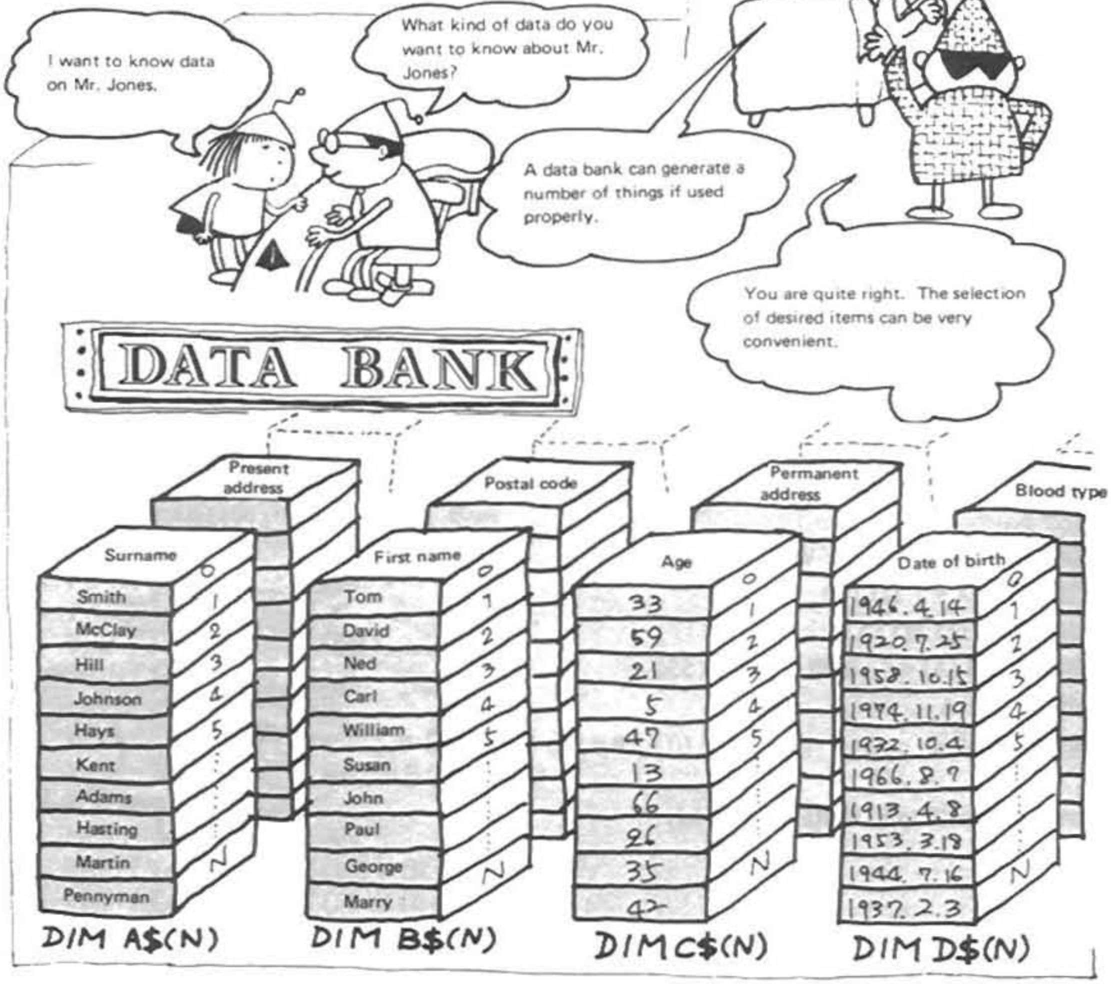
```
500  ROPEN/T "PUPPY WALTZ"
510  FOR A = 1 TO 100
520  INPUT/T M$ (A)
530  IF M$ (A) = " ◧■ " THEN 550
540  NEXT A
550  CLOSE/T
```

This can read the "Puppy Waltz" completely.

Data Dank is a Computer's Speciality

The computer's splendid memory makes it easy to compile a list with a number of items. The sequence of items as data, for example, name, age, date of birth, present address and permanent address, is brought into memory. This is called a data bank.

This data bank makes it simple to retrieve and sort out items depending on the type of program. Retrieving is made for all the data related to any required items assigned. For example, a blood type is assigned to retrieve all people of the same blood type, and this is called "retrieving by blood type". In addition, with an eye to a certain item, names are arranged in ABC order, for example, or ages are arranged in the order of the young to the old. This is called "sorting".



Telephone Number List is also a Data Bank

With the above understood, a summary is made of the program in which string data is put into the memory of the DATA statement. Based on this program, modifications are possible so that the address and postal code are also available.

```
10  N = 12
20  DIM M$ (N)  . . . . . . . . . . . Surname
30  DIM N$ (N)  . . . . . . . . . . . First name
40  DIM A$ (N) . . . . . . . . . . . . Home dialling code
50  DIM B$ (N) . . . . . . . . . . . . Home telephone number
60  DIM C$ (N) . . . . . . . . . . . . Work dialling code
70  DIM D$ (N) . . . . . . . . . . . . Work telephone number
80  DIM F (N)
90  FOR K = 1 TO N
100  READ M$ (K), N$ (K)
110  READ A$ (K), B$ (K)
120  READ C$ (K), D$ (K)
130  NEXT K
140  PRINT : PRINT : X = 0
150  PRINT "WHAT IS THE SURNAME" ;
160  INPUT X$ . . . . . . . . . . . . . . . . . . . . . . . . Key-in the name to be retrieved.
170  FOR K = 1 TO N
180  IF M$ (K) = X$ THEN X = X + 1 : F (X) = K . . Retireving by use of the surname.
190  NEXT K
200  IF X <> 0 THEN 240
210  PRINT "NO RELEVANT PERSON FOUND !"
220  PRINT "PLEASE RE – ENTER"
230  GOTO 140
240  PRINT : PRINT
250  FOR K = 1 TO X
260  L = F (K)  . . . . . . . . . . . . . . . . . . . . . . . . For display of persons with the same surname.
270  PRINT "NAME" ;TAB (11) ;" : " ;N$ (L) ;"    " ;M$ (L)
280  PRINT "HOME NUMBER : " ;
290  PRINT "(" ; A$ (L) ; ") " ; B$ (L)
300  PRINT "WORK NUMBER : " ;
310  PRINT " (" ;C$ (L) ;") " ;D$ (L) : PRINT
320  NEXT K
330  GOTO 140
340  DATA JONES, JOHN, 01, 364, 9617, 01, 969, 3678
350  DATA DAVIS, PETER, 021, 396, 2137, 01, 323, 6146
360  DATA SMITH, PAUL, 0449, 73246, 0449, 71277
370  DATA JONES, DAVID, 061, 631, 1235, 061, 312, 1975
380  DATA RICHARDS, ROBIN, 0273, 61976, 0903, 47216
390  DATA SMITH, HARRY, 01, 638, 2174, 29, 147636
400  DATA LAKE, COLIN, 4967, 13642, 4967, 32132
410  DATA WATSON, JOHN, 01, 961, 2431, 0427, 21369
420  DATA CARTER, DAVID, 6317, 21974, 01, 316, 2638
430  DATA HOMLES, FRANK, 2238, 76194, 2238, 78352
440  DATA JONES, FRED, 9743, 61665, 01, 424, 6913
450  DATA WILSON, JAMES, 01, 692, 5687, 0374, 68421
460  END
```

■ SOS in Morse Code

The Morse code was invented by Samuel F. Breese Morse, an American artist, in 1838, and is one of the most important communications media even today.

The principle is simple. It sets up the ratio of times when a specified wave of frequency is produced and not produced.

Prolonged sound ——	Transmission of sound 3 times as long as short sound.
Short sound —	
Pause	No sound for the same period of time as short sound.

The Morse code is based on the combination of these three sounds to represent the necessary symbols. Shown below is part of the Morse code, according to which try to strike SOS. Very difficult? The Morse code requires practice until your fingers move naturally and quickly without thinking of where to press.

To make this easy, the program for the Morse code is generated in the following section. Line numbers 20 to 270 are strings to generate signals from A to Z, and line numbers 290 to 380 for those from 0 to 9. Brief description is given of the program.

+A5 ——→ Sound A (la) in the high frequency range with its tonal length of 5 (equivalent to the prolonged signal of the Morse code).

+A2 ——→ Sound A (la) in the high frequency range with its tonal length of 2 (equivalent to the short signal of the Morse code).

R2 ——→ Pause with no sound with its length of 2.

A	·—	G	——·	N	—·	T	—	Z	——··	6	—····
B	—···	H	····	O	———	U	··—	1	·————	7	——···
C	—·—·	J	·———	P	·——·	V	···—	2	··———	8	———··
D	—··	K	—·—	Q	——·—	W	·——	3	···——	9	————·
E	·	L	·—··	R	·—·	X	—··—	4	····—	0	—————
F	··—·	M	——	S	···	Y	—·——	5	·····	I	··

Signals in Dots and Dashes

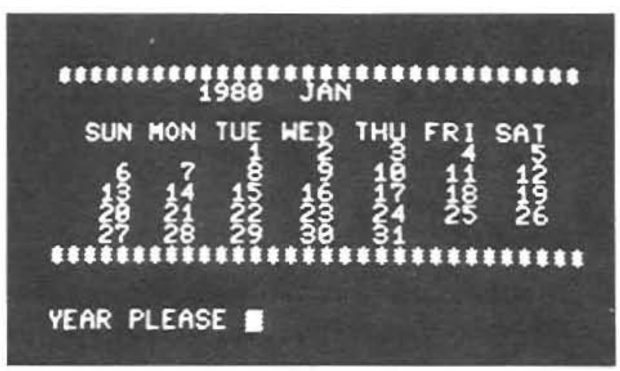
```
10  DIM A1 (100), M$ (127)
20  M$ (65) = " +A2R2+A5"
30  M$ (66) = " +A5R2+A2R2+A2R2+A2"
40  M$ (67) = " +A5R2+A2R2+A5R2+A2"
50  M$ (68) = " +A5R2+A2R2+A2"
60  M$ (69) = " +A2"
70  M$ (70) = " +A2R2+A2R2+A5R2+A2"
80  M$ (71) = " +A5R2+A5R2+A2"
90  M$ (72) = " +A2R2+A2R2+A2R2+A2"
100  M$ (73) = " +A2R2+A2"
110  M$ (74) = " +A2R2+A5R2+A5R2+A5"
120  M$ (75) = " +A5R2+A2R2+A5"
130  M$ (76) = " +A2R2+A5R2+A2R2+A2"
140  M$ (77) = " +A5R2+A5"
150  M$ (78) = " +A5R2+A2"
160  M$ (79) = " +A5R2+A5R2+A5"
170  M$ (80) = " +A2R2+A5R2+A5R2+A2"
180  M$ (81) = " +A5R2+A5R2+A2R2+A5"
190  M$ (82) = " +A2R2+A5R2+A2"
200  M$ (83) = " +A2R2+A2R2+A2"
210  M$ (84) = " +A5"
220  M$ (85) = " +A2R2+A2R2+A5"
230  M$ (86) = " +A2R2+A2R2+A2R2+A5"
240  M$ (87) = " +A2R2+A5R2+A5"
250  M$ (88) = " +A5R2+A2R2+A2R2+A5"
260  M$ (89) = " +A5R2+A2R2+A5R2+A5"
270  M$ (90) = " +A5R2+A5R2+A2R2+A2"
280  REM NO.
290  M4 (48) = " +A5R2+A5R2+A5R2+A5R2+A5"
300  M$ (49) = " +A2R2+A5R2+A5R2+A5R2+A5"
310  M$ (50) = " +A2R2+A2R2+A5R2+A5R2+A5"
320  M$ (51) = " +A2R2+ A2R2+A2R2+A5R2+A5"
330  M$ (52) = " +A2R2+A2R2+A2R2+A2R2+A5"
340  M$ (53) = " +A2R2+A2R2+A2R2+A2R2+A2"
350  M$ (54) = " +A5R2+A2R2+A2R2+A2R2+A2"
360  M$ (55) = " +A5R2+A5R2+A2R2+A2R2+A2"
370  M$ (56) = " +A5R2+A5R2+A5R2+A2R2+A2"
380  M$ (57) = " +A5R2+A5R2+A5R2+A5R2+A2"
390  REM "SPACE"
400  M$ (32) = "R5"
1000  INPUT "TYPE IN A MASSAGE " ; A$
1010  FOR J = 1 TO LEN (A$)
1020  A1 (J) = ASC (MID$ (A$, J, 1))
1030  NEXT J
1040  FOR J = 1 TO LEN (A$)
1050  MUSIC M$ (A1 (J)), "R5"
1060  NEXT J
1070  GOTO 1000
```

Key in alphabet from A to Z and munerals from 0 to 9. For example, when you key-in "I LOVE YOU", the Morse code will be generated accordingly. Using the Morse code, you can declare your love to your sweetheart!

■ Unending "Time".....

At the end of this introduction to the BASIC Language, the program for the "Perpetual Calendar" is introduced. It requires no detailed explanation. Our "time" continues eternally.

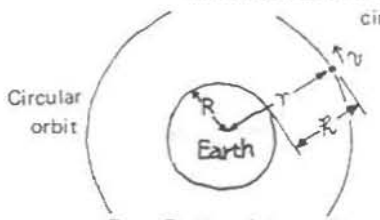
```
5   DIM M$ (12), W$ (7)
10  FOR K = 1 TO 12 : READ M$ (K) : NEXT K
20  FOR K = 1 TO 7 : READ W$ (K) : NEXT K
30  INPUT "YEAR PLEASE ? " ; Y : INPUT "MONTH PLEASE ? " ; MT
40  H = MT : GOSUB 400 : K2 = YB + 1
50  H = MT + 1 : GOSUB 400 : K1 = YB + 1
60  N = K1 - K2 : IF N >= 0 THEN L = 28 + N : GOTO 70
65  L = 35 + N
70  IF MT = 12 THEN L = 31
75  PRINT " ■ " : GOSUB 190
80  PRINT TAB (8) ; Y ; "    " ; M$ (MT) : PRINT : T = 4
90  FOR N = 1 TO 7 : PRINT TAB (T) : W$ (N) ; : T = T + 4 : NEXT N : PRINT
100 T = 0 : IF K2 = 0 THEN 120
110 FOR N = 1 TO K2 : PRINT TAB (T) ; : T = T + 4 : NEXT N : T = T - 4
120 FOR N = 1 TO L : N$ = STR$ (N) : J = LEN (N$)
130 PRINT TAB (T + 5 - J) ; N$ ; : T = T + 4
140 IF T = 28 THEN T = 0 : PRINT
150 NEXT N
160 IF T <> 0 THEN PRINT
170 GOSUB 190
180 PRINT " ■ " : GOTO 30
190 FOR Z = 1 TO 31 : PRINT " * " ; : NEXT Z : PRINT : RETURN
200 DATA JAN, FEB, MAR, APR, MAY, JUN
210 DATA JUL, AUG, SEP, OCT, NOV, DEC
220 DATA SUN, MON, TUE, WED, THU, FRI, SAT
230 END
400 X = Y
410 N = H - 3 : J = 12 : GOSUB 600 : MM = Z
420 IF MM > 9 THEN X = X - 1
430 N = X : J = 400 : GOSUB 600 : X = Z
440 X4 = INT (X/4) : X1 = INT (X/100)
450 KY = X + X4 - X1
460 N = MM : J = 5 : GOSUB 600 : MZ = Z
470 M5 = INT (MM/5) : M2 = INT (MZ/2)
480 N = MZ : J = 2 : GOSUB 600 : P = Z
490 KM = 13 * M5 + 5 * M2 + 3 * P
500 N = KY + KM + 3 : J = 7 : GOSUB 600 : YB = Z
510 RETURN
600 REM Z = N, J
610 K = INT (N/J)
620 Z = N - K * J
630 IF Z < 0 THEN Z = Z + J
640 RETURN
```



Miniature Space Dictionary

If you are interested in space, including astronomy and man-made satellites, you might like to try calculations and graphic drawings by using the computer. Shown below are equations and values required for such attempts.

Unlike the earth, the movement of objects in space should mathematical calculations without any complexity caused by atmospheric resistance. For more accurate values, however, consideration must be given to the effects by the planets, the perturbation caused by strains in the form of the earth and gas pressure in space, even though rarefied. There is air of 10^{-9} mmHg at an altitude of 800 km in space, for example. In addition, a man-made satellite stationed at an altitude of approx. 36,000 km tilts approximately 1 degree per year in its orbit being affected by other heavenly bodies.



R: Radius of the earth
h: Height from the earth surface

Motion velocity of a satellite in circular orbit : $v = \sqrt{\frac{G \cdot M}{r}}$ (m/s)

Period : $T = 2\pi\sqrt{\frac{r^3}{G \cdot M}}$ (s)

$G = 6.67 \times 10^{-11}$ $(N \cdot m^2/kg^2)$

Universal gravitation constant

M : Mass of the earth (kg)

r : Distance from the earth's center to satellite (m)

Motion velocity of a satellite in oval orbit : $v = \sqrt{G \cdot M\left(\frac{2}{r} - \frac{1}{a}\right)}$ (m/s)

Period : $T = 2\pi\sqrt{\frac{a^3}{G \cdot M}}$ (s)

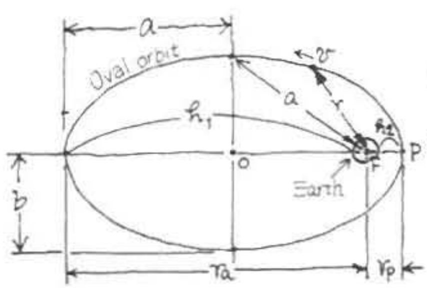
Long radius : $a = \frac{h_1 + h_2 + 2R}{2}$ (m)

Short radius : $b = \sqrt{a^2 - (a \cdot e)^2}$ (m)

Eccentricity : $e = OF/OP = \frac{\sqrt{a^2 - b^2}}{a}$

$r_p = a(1 - e)$ (m)

$r_a = a(1 + e)$ (m)



	Mass (1 for the Sun)	Equatorial radius	Eccentricity	Averaged distance from the Sun (a)
Sun	1.000	696 000 km	–	–
Mercury	0.166×10^{-6}	2 440	0.20563	0.57910×10^{8} km
Venus	2.448×10^{-6}	6 056	0.00678	1.08210 〃
Earth	30.034×10^{-7}	6 378	0.01672	1.49600 〃
Mars	3.227×10^{-7}	3 390	0.09338	2.27944 〃
Jupiter	95.479×10^{-5}	71 400	0.04829	7.7834 〃
Saturn	2.856×10^{-4}	60 400	0.05604	14.2700 〃
Uranus	4.373×10^{-5}	23 700	0.04613	28.7103 〃
Neptune	5.178×10^{-5}	25 110	0.01004	44.971 〃
Pluto	0.552×10^{-6}	3 400	0.24842	59.136 〃
Moon	3.694×10^{-8}	1 738	0.0549*	384 400 km*

Mass of the Sun = 1.991×10^{30} kg   * Value to the earth   Source: Chronological Table of Science

A solution of simultaneous equation

Introduction to methodological study of programming 【1】

Solution of systems of simultaneous linear equations is a basic data processing problem associated with all science and engineering problems. In this section, we will try to construct a basic subroutine which is used to solve systems of simultaneous linear equations in n unknowns.

Although there are a number of algorithms for solving simultaneous linear equations, we here employ the elimination method which is familiar to you from your school days.

■ Approach to the problem

Consider the problem of solving the system of simultaneous linear equations shown in (1) below.

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots\cdots + a_{1n}x_n = b_1 \, . \\ a_{21}x_1 + a_{22}x_2 + \cdots\cdots + a_{2n}x_n = b_2 \, . \\ \cdots\cdots\cdots\cdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots\cdots + a_{nn}x_n = b_n \end{cases} \quad (1)$$

Multiplying both sides of the second equation in (1) by a_{11}/a_{21} and subtracting the result from the first equation, we obtain

$$\left(a_{12} - a_{22}\frac{a_{11}}{a_{21}}\right) x_2 + \left(a_{13} - a_{23}\frac{a_{11}}{a_{21}}\right) x_3 + \cdots\cdots + \left(a_{1n} - a_{2n}\frac{a_{11}}{a_{21}}\right) x_n = b_1 - b_2\frac{a_{11}}{a_{21}} \quad (2)$$

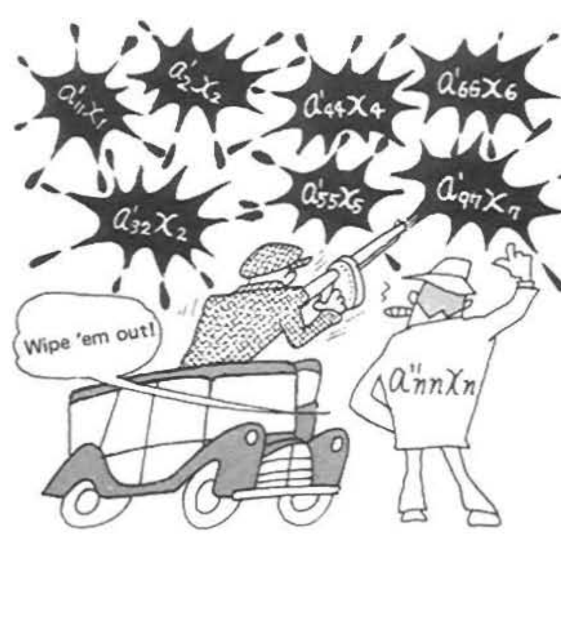
This means that we obtain an equation in which x_1 is eliminated. Performing this process up through the nth equation, we obtain a set of simultaneous linear equations in which x_1 is not included, that is, one unknown is eliminated. Rewriting the coefficients of all equations as a'_{22}, a'_{2n}, and so on, we obtain

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots\cdots + a_{1n}x_n = b_1 \, . \\ \qquad a'_{22}x_2 + \cdots\cdots + a'_{2n}x_n = b'_2 \, . \\ \qquad \cdots\cdots\cdots\cdots \\ \qquad a'_{n2}x_2 + \cdots\cdots a'_{nn}x_n = b'_n \end{cases} \quad (3)$$

The process of creating the set of equations in (3) from the set of equations in (1) is called elimination using the first row and column as a pivot.

Performing the elimination process on the set of equations in (3) using the second row and column as a pivot, we find a system of simultaneous linear equations whose third to nth equations do not contain the term x_2. Repeating the elimination process by pivoting $n-1$ times, we obtain

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots\cdots + a_{1n}x_n = a_{1n+1} \, . \\ \qquad a'_{22}x_2 + \cdots\cdots \quad a'_{2n}x_n = a'_{2n+1} \, . \\ \qquad\qquad \cdots\cdots\cdots\cdots \\ \qquad\qquad\qquad a''_{nn}x_n = a''_{nn+1} \end{cases} \quad (4)$$



The value of x_n is obtained by the nth equation in (4), then the value of x_{n-1} is obtained from the $(n-1)$th equation, and so on.

Although this method seems simple, if you solve by hand, it will take a lot of labor and scratch paper if 5 or 6 unknowns are involved. You will lose interest in solving the problem by hand if the number of unknowns is 10 or 20.

It is best to use a computer to perform repetitions of such simple operations. The computer can eliminate obstacles in a lump and give us the correct answer immediately.

■ Programming

Let's formulate an algorithm for solving systems of simultaneous linear equations.

1	Assign the number of unknowns to variable N. Prepare a 1-dimensional array X(N) to store the value of the unknowns. Prepare a 2-dimensional array A(N, N+1) and to it assign the coefficients of the equations in (1).
2	Call the subroutine for solving systems of simultaneous linear equations.
3	Print the values of the unknowns (that is, the contents of X(1) to X(N)) on the CRT display unit.

Subroutine (for solving systems of simultaneous linear equations)

4	Perform the elimination process N−1 times to change the contents of array A to the coefficients of the equations in (4).
5	Find the values of the unknowns in sequence and assign them to X(N) through X(1).

In this manner, we can clearly identify the subroutine (for solving systems of simultaneous linear equations) as a basic module which takes the value of variable N as the number of unknowns and the contents of 2-dimensional array A(N, N+1) as the coefficients of the equations in (1), finds values of the unknowns, and assign them to array X(N) from X(N) to X(1).

First, let's consider step 4 which is the most important elimination step. As seen from the equations in (2), the kth (k = 1 to N−1) elimination process is carried out basically by repetitions of assignment

$$a_{ij} \longleftarrow a_{kj} - a_{ij}\frac{a_{kk}}{a_{ik}} \qquad (\leftarrow \text{denotes assignment.}) \qquad (5)$$

for coefficient a_{ij}. This can be accomplished by executing the 2-level loop

FOR $i = k+1$ TO N

  FOR $j = k+1$ TO N+1

    $a_{ij} \longleftarrow a_{kj} - a_{ij}\frac{a_{kk}}{a_{ik}}$

  NEXT j

NEXT i

(FOR j … NEXT j: Loop for one row; FOR i … NEXT i: Loop for rows k through N)

Step 4 can be programmed in this way using variables K, I, and J as follows:

```
FOR K=1 TO N-1
   FOR I =K+1 TO N
      FOR J=K+1 TO N+1
         A(I , J)=A(K , J)-A(I , J)*A(K , K)/A(I , K)
      NEXT J
   NEXT I
NEXT K
```

(NEXT J, NEXT I, NEXT K: These statements can be combined to one as NEXT J, I, K.)

Now proceed to step 5, where the values of the unknowns are found in sequence. To find the value of unknown x_i, all that is required is to assign x_{i+1} through x_n to the set of equations in (5). Consequently, we obtain

FOR $j = i+1$ TO N

  $a_{iN+1} \longleftarrow a_{iN+1} - a_{ij}\ x_j$

NEXT j

$x_i \longleftarrow a_{iN+1} / a_{ii}$

Although the program code

```
FOR I=N TO 1 STEP-1
   FOR J=I+1 TO N
      A(I . N+1)=A(I . N+1)-A(I . J)*X(J)
   NEXT J
   X(I)=A(I . N+1)/A(I . I)
NEXT I
```

seems satisfactory, in this program, when I = N, J has a value of N + 1 and the computer executes the statement within the loop controlled by J. As it stands, a dimensional overflow would occur at X(J) or an incorrect answer would result (even when X(N+1) is defined) if its content is nonzero. We can avoid such errors by placing the step for finding the value of unknown x_n outside of the loop; that is, by changing the I-controlled loop as follows:

```
X(N)=A(N . N+1)/A(N . N)
FOR I=N-1 TO 1 STEP -1
..............................................

NEXT I
```

We can complete the subroutine for solving systems of simultaneous linear equations by adding a RETURN statement to the end of the above program code.

The essential problem in step [1] is in how to assign the unknowns and the coefficients of the simultaneous linear equations in question to variable N and 2-dimensional array A(N, N+1). Many methods are possible, such as using the READ and DATA statements; however, we have decided here to enter them one at a time from the keyboard.

In step [3], we decided to display the values of the unknowns on the CRT display unit in the format:

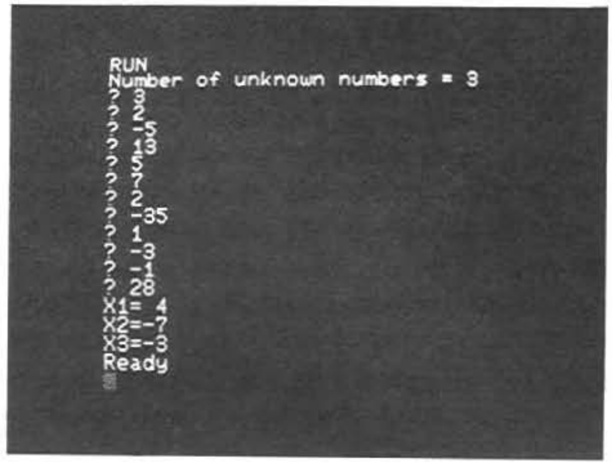
```
X1=.........
X2=.........
...............
```

These steps can be coded without difficulty. Finally, we obtain a complete program as follows:

```
5 REM ................................................................ Read data
10 INPUT "Number of unknown numbers =" ; N
20 DIM A(N, N+1), X(N)
30 FOR S1=1 TO N: FOR S2=1 TO N+1
40 INPUT A(S1, S2)
50 NEXT S2, S1
100 GOSUB 1000
195 REM .............................................................. Print Xi
200 FOR I=1 TO N
210 PRINT "X": STR$(I); "=" ; X(I)
220 NEXT I
230 END
995 REM ........................................................... Elimination
1000 FOR K=1 TO N-1
1010 FOR I=K+1 TO N
1020 FOR J=K+1 TO N+1
1030 A(I, J)=A(K, J) -A(I, J) * A(K, K) /A(I, K)
1040 NEXT J, I, K
1095 REM ............................................................. Find Xi
2000 X(N)=A(N, N+1) /A(N, N)
2010 FOR I=N-1 TO 1 STEP -1
2020 FOR J=I+1 TO N
2030 A(I, N+1)=A(I, N+1) -A(I, J) * X(J)
2040 NEXT J
2050 X(I)=A(I, N+1) /A(I, I)
2060 NEXT I
2070 RETURN
```

Run the program and solve the following system of simultaneous linear equations in three unknowns.

$$\begin{cases} 3x_1 + 2x_2 - 5x_3 = 13 \\ 5x_1 + 7x_2 + 2x_3 = -35 \\ x_1 - 3x_2 - x_3 = 28 \end{cases} \quad (6)$$



When the program is run and the coefficient values and the values which appear on the right-hand side of the equations are entered as shown in figure, the following solution is obtained:

$$x_1 = 4$$
$$x_2 = -7$$
$$x_3 = -3$$

- **Exercises**

1. Solve the following systems of simultaneous linear equations:

$$\begin{cases} x_1 + 2x_2 + 3x_3 - x_4 = 5 \\ x_1 - 3x_2 + 5x_3 + 2x_4 = 3 \\ -x_1 - 4x_2 + x_3 + 7x_4 = -7 \\ -x_1 + x_2 + 11x_3 - 3x_4 = -22 \end{cases}$$

$$\begin{cases} x_1 + 2x_2 + 4x_3 + 8x_4 + 16x_5 = 32 \\ x_1 + 3x_2 + 9x_3 + 27x_4 + 81x_5 = 243 \\ x_1 + 4x_2 + 16x_3 + 64x_4 + 256x_5 = 1024 \\ x_1 + 5x_2 + 25x_3 + 125x_4 + 625x_5 = 3125 \\ x_1 + 6x_2 + 36x_3 + 216x_4 + 1296x_5 = 7776 \end{cases}$$

The program constructed above does not specify the number of unknowns explicitly. In practice, however, the number of unknowns is restricted by the space available for defining the necessary arrays (or the size of the BASIC text area). If you use it on actual problems, however, you will find that the program can solve systems of simultaneous linear equations with more than 80 unknowns when it is used on the MZ-80A. Try solving various systems of simultaneous linear equations in which there are a large number of unknowns.

2. After running program several times, you may encounter the error message
   * Error 2 in 1030.

which indicates that an overflow error occurred during execution of line number 1030. This condition will occur, for example, if an attempt is made to solve the system of simultaneous linear equations:

$$\begin{cases} x_1 + 2x_2 + 3x_3 = -26 \\ 3x_1 + 5x_2 + 2x_3 = -39 \\ 2x_1 + 4x_2 + x_3 = -27 \end{cases} \quad (7)$$

The overflow condition is caused because x_1 and x_2 are eliminated simultaneously from the third equation during the first elimination operation and the denominator of the division on line number 1030 is set to 0 during the second elimination operation. This type of error can be avoided if we exchange the second and third rows before entering data.

Since the pivot position moves diagonally as program execution proceeds, undesirable conditions will occur if a diagonal element happens to be zero. Try developing of countermeasures for this problem.

Find 1000 prime numbers

Introduction to methodological study of programming [2]

It is essential to always keep the objective and the procedure for accomplishing it in mind when formulating a program. Most programmers, however, are seldom conscious of this problem and create complicated, inscrutable programs in their own style. Frequently, such programs can hardly be understood even by those who wrote them, much less by others.

Such problems arise because the programmer does not clarify the relationship beteen the structure of the problem and the algorithm for solving it when the program is written. Consideration of this problem has led to active methodological study of programming itself. E.W. Dijkstra is one of the leaders in this field, and has written many outstanding books on this subject. In one of these books†, he introduced his own idea about programming (called structured programming), using the problem of finding prime numbers as an example. In this section, we briefly explain his concepts using the same problem.

Problem:

Print 1000 prime numbers 2, 3, 5, 7, 11, ... in increasing order of magnitude.

■ **Approach to the problem**

Many different, programs could be used to solve this problem. The first major proposition, however, is that the program constructed must be a practical one. An extreme approach might be as follows.

"Find and print prime numbers starting at the smallest one? 2 is the smallest prime number, so, let's print 2 first. Next is 3, next is 5, next is 7. All we have to do is to print them using the PRINT statement. . . . "

This is not, however, an efficient method of using the BASIC interpreter, and cannot truly be regarded as programming.

According to Dijkstra, many decisions must be made until a program is completed. We should make such decisions only when they are actually required, rather than making them without discipline. In other words, programming should be conducted in stages.

The first decision to be made in our problem is whether the 1000 prime numbers are to be found first, then printed all at once, or whether they are to be printed as they are found. By deciding on the former method, we place the program in perspective as follows.

```
GOSUB "Find 1000 prime numbers."  )
GOSUB "Print all numbers found."  }  Main program
END                               )

Find 1000 prime numbers.          )
                                  }  Subroutines
Print all numbers found.          )
```

†) Dijkstra, Hoar, Dahr: Structured Programming, 1969 ALGOL is used as the programming language in this publication.

Notice that the problem has been divided neatly into a main program and two subroutines. This is one of the basic principles of structured programming suggested by Dijsktra.

Now, proceed to the next step. Since we are to find 1000 prime numbers before printing them, we need to store them somehow in a storage location. The next step is to determine the method of storing the prime numbers.

It would be unwise to declare a numeric array of 1000 elements simply because 1000 prime numbers must be memorized. It is possible to define a string array and place T's (true) in locations in the string designated by subscripts which happen to be prime numbers and F's (false) in other locations. Another possible method is to record the prime numbers in a data file on cassette tape. It is easy to identify prime numbers if they are stored in a string array and identified by the characters "T" and "F"; and it is possible to store prime numbers for a long time if they are recorded on cassette tape. What method should be used?

For an algorithm in which unprocessed numbers are divided by the prime numbers already found to identify prime numbers, the first method is most appropriate; that is, to prepare a numeric array of 1000 elements. Accordingly, we declare a numeric array of 1000 elements at the beginning of the main program. DIM "Numeric array of 1000 elements".

```
DIM "Numeric array of 1000 elements"
```

Next, the structure of the array must be determined. We can use PRIM as the array name (though the array name is identified only by the first two characters) and use a 2-dimensional numeric array. This is because, since the maximum subscript value for 1-dimensional arrays is 255, it is not possible to identify all array elements with a 1-dimensional array. Since all array elements must be identified within the subscript range of 255, we use an array structure such that the 1000 elements are grouped into 10 subarrays of 100 elements each. The DIM statement for the required array is as follows:

```
DIM PRIM (9 . 99)
```

With this decided, we can go on to determine the format for display of the 1000 prime numbers.

There are many ways of outputting the results, such as displaying them on the CRT display unit or printing them on the printer. As for format, one prime number may be printed on one line or they may be printed in a tabular form; and so on. We will use the simplest format; that is, sequentially printing the numbers on the CRT screen without formatting them. The following subroutine ("Print all numbers found") will be adequate for this purpose:

```
FOR M=0 TO 9 : FOR N=0 TO 99
   PRINT PRIM (M . N) ;
NEXT N . M
```

Now we have finished that main program and the subroutine "Print all numbers found." The remaining task is to formulate the subroutine "Find 1000 prime numbers." Since we are going to use the array described above, it is natural to find the prime numbers sequentially, starting with the smallest one, and to place them into the array in increasing order of its subscripts.

Assuming that we have a subroutine "Find the next prime number" which, given parameter I, examines I + 1, I + 2, ... insequence, places the first prime number found into I, and returns control to the calling program, we can form the subroutine "Find 1000 prime numbers" as follows:

```
I←1
FOR M=0 TO 9 : FOR N=0 TO 99
   GOSUB  "Find the next prime number"
   PRIM (M . N)←I
NEXT N . M
RETURN
```

Now we are approaching the nucleus of the program for finding prime numbers. The subroutine "Find the next prime number" must find and assign to I the smallest prime number which is greater than I. A simple algorithm which you will think of immediately is to divide I by 2, 3, 4, 5, ..., I-1, and identify I as a prime number when I is indivisible by any of them. With this algorithm, you must perform 99 divisions using divisors from 2 to 100 to recognize 101 as a prime number. You will soon recognize that this algorithm wastes a great amount of time. It is apparent that numbers which are indivisible by 2 are also indivisible by multiples of 2 (e.g., 4, 6, 8, ...), numbers which are indivisible by 3 are also indivisible by multiples of 3, and so on. Since our goal is to find prime numbers sequentially starting with the smallest one, to determine whether a number is a prime or not we need only to determine whether it is divisible by any of the prime numbers which have been found so far. For example, to determine whether 101 is a prime number or not, we need only divide it by a total of 25 prime numbers, i.e., 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, and 97.

Even this algorithm wastes a considerable amount of time. For example, we need not divide 101 by 11. When a number p can be divided by a number other than itself, it can be expressed as $p = f \cdot g$. Assuming that f is not greater than g ($f \leqq g$), we find $f^2 \leqq f \cdot g = p$. Accordingly, we need only to examine integers which are not greater than $\sqrt{p}$. For 101, we need only divide it by four prime numbers (2, 3, 5, and 7), if it is not indivisible by these numbers, we can regard 101 as a prime number. This fact affects programming efficiency greatly. In fact, to determine whether 10100 (which is 100 times greater than 101), is a prime number or not, we only need to divide it by 25 prime numbers from 2 to 97. Otherwise, as you will see later, you would have to divide it by far more than 1000 prime numbers.

Taking the above into consideration, we can formulate the algorithm for the subroutine "Find the next prime number" as follows:

```
l1  I ← I+1 : X ← 0 : Y ← 0
l2  While (PRIM (X, Y))² < I
    ┌─ Divide I by PRIM (X, Y)
    │ ┌─ If divisible GOTO l1
    │ │    Otherwise
    │ │      ┌─ Determine the subscripts X and Y of the array element containing the next prime number
    └─└─     └─ GOTO l2
    Return
```

The above algorithm is coded as follows:

```
1500    REM --- Find the next prime number ---
1510    I=I+1 : X=0 : Y=0
1520    IF PRIM(X . Y)*PRIM(X . Y)>I THEN RETURN
1530    L=I/ PRIM(X . Y)
1540    IF L- INT(L)=0 THEN 1510
1550    Y=Y+1
1560    IF Y<100 THEN 1520
1570    X=X+1 : Y=0 : GOTO 1520
```

The statement on line 1520 determines whether the square of the prime number by which the parameter I is to be divided exceeds I.

The reason the power operator is not used in this statement is that an expression containing the power operator is inappropriate as a condition clause for the IF statement because evaluation of expressions containing the power operator are internally conducted by approximation.†

Now let's finish the program. Do not forget, however, to assign the first prime number (i.e., 2) to PRIM (0, 0). This is because the subroutine "Find the next prime number" assumes that there is a preceding prime.

†) If the power operator is to be used on line 1520, the line must be coded as follows:

```
1520 IF INT (PRIM (X, Y) ↑ 2 + 0.00000001) > I THEN RETURN
```

```
10  REM .......................................... 1000 prime numbers
20  DIM PRIM (9,99)
30  PRIM (0,0) = 10
40  GOSUB 1000
50  GOSUB 3000
60  END
1000  REM .................................... Find 1000 prime numbers
1010  I = 1
1020  FOR M = 0 TO 9 : FOR N = 0 TO 99
1030  GOSUB 2000
1040  PRIM (M, N) = I
1050  NEXT N, M
1060  RETURN
2000  REM .................................... Find the next prime number
2010  I = I + 1 : X = 0 : Y = 0
2020  IF PRIM (X, Y) * PRIM (X, Y) > I THEN RETURN
2030  L = I/PRIM (X, Y)
2040  IF L - INT (L) = 0 THEN 2000
2050  Y = Y + 1
2060  IF Y < 100 THEN 2020
2070  X = X + 1 : Y = 0 : GOTO 2020
3000  REM .................................... Print 1000 prime numbers
3010  FOR M = 0 TO 9 : FOR N = 0 TO 99
3020  PRINT/P PRIM (M, N),
3030  NEXT N, M
3040  RETURN
```

Print listing of 1000 prime numbers on the line printer MZ-80P5.

2	3	5	7	11	13	17	19
23	29	31	37	41	43	47	53
59	61	67	71	73	79	83	89
97	101	103	107	109	113	127	131
137	139	149	151	157	163	167	173
179	181	191	193	197	199	211	223
227	229	233	239	241	251	257	263
269	271	277	281	283	293	307	311
313	317	331	337	347	349	353	359
367	373	379	383	389	397	401	409
419	421	431	433	439	443	449	457
461	463	467	479	487	491	499	503
509	521	523	541	547	557	563	569
571	577	587	593	599	601	607	613
617	619	631	641	643	647	653	659
661	673	677	683	691	701	709	719
727	733	739	743	751	757	761	769
773	787	797	809	811	821	823	827
829	839	853	857	859	863	877	881
883	887	907	911	919	929	937	941
947	953	967	971	977	983	991	997
1009	1013	1019	1021	1031	1033	1039	1049
1051	1061	1063	1069	1087	1091	1093	1097
1103	1109	1117	1123	1129	1151	1153	1163
1171	1181	1187	1193	1201	1213	1217	1223

....

omitted

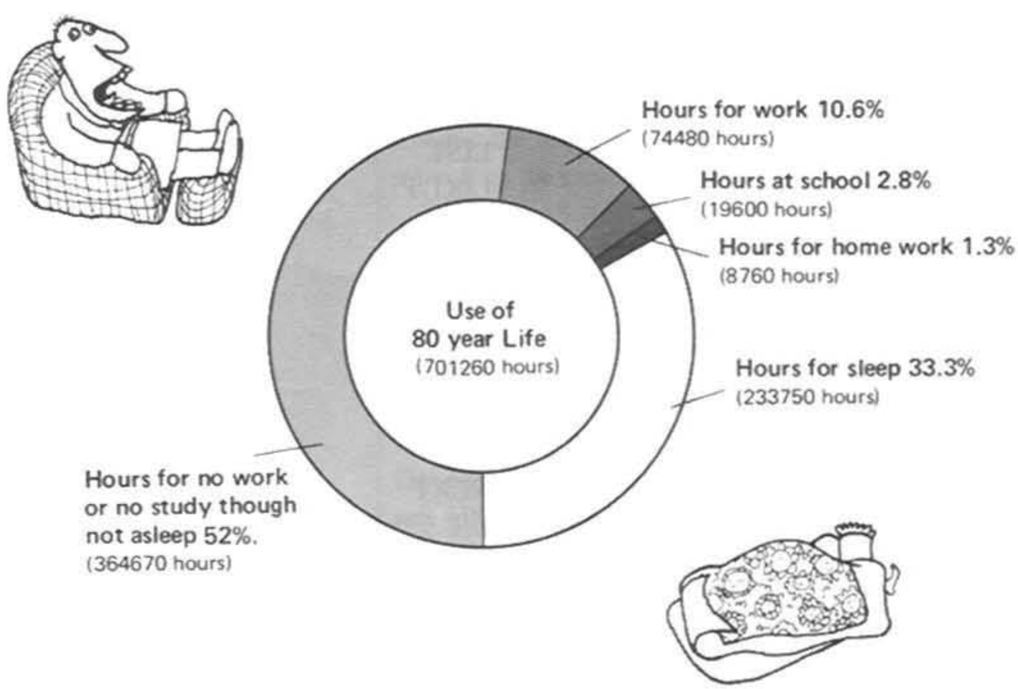
....

6143	6151	6163	6173	6197	6199	6203	6211
6217	6221	6229	6247	6257	6263	6269	6271
6277	6287	6299	6301	6311	6317	6323	6329
6337	6343	6353	6359	6361	6367	6373	6379
6389	6397	6421	6427	6449	6451	6469	6473
6481	6491	6521	6529	6547	6551	6553	6563
6569	6571	6577	6581	6599	6607	6619	6637
6653	6659	6661	6673	6679	6689	6691	6701
6703	6709	6719	6733	6737	6761	6763	6779
6781	6791	6793	6803	6823	6827	6829	6833
6841	6857	6863	6869	6871	6883	6899	6907
6911	6917	6947	6949	6959	6961	6967	6971
6977	6983	6991	6997	7001	7013	7019	7027
7039	7043	7057	7069	7079	7103	7109	7121
7127	7129	7151	7159	7177	7187	7193	7207
7211	7213	7219	7229	7237	7243	7247	7253
7283	7297	7307	7309	7321	7331	7333	7349
7351	7369	7393	7411	7417	7433	7451	7457
7459	7477	7481	7487	7489	7499	7507	7517
7523	7529	7537	7541	7547	7549	7559	7561
7573	7577	7583	7589	7591	7603	7607	7621
7639	7643	7649	7669	7673	7681	7687	7691
7699	7703	7717	7723	7727	7741	7753	7757
7759	7789	7793	7817	7823	7829	7841	7853
7867	7873	7877	7879	7883	7901	7907	7919

■ 701,260 Hours

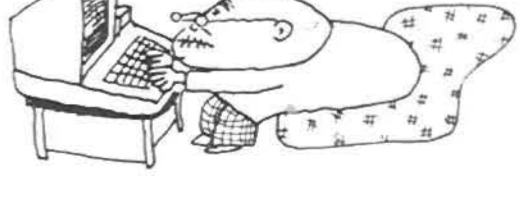
You have mastered your computer to use it as if it were part of your brains, haven't you ? What did you say ? You are too busy to have time for that. Indeed, we are living in this busy world, aren't we ?

By the way, let's predict, using the computer, what a busy life you are leading. Calculations are made for a life of 80 years that is a little longer than the English average. For education 16 years are spent at infant school, primary school, high school and university or college. 245 days a year are for going to school and 5 hours a day for lessons. After school, 1.5 hours are for home work every day throughout the year. After graduation from university, 8 hours a day for 245 days a years a year are for work as a salaried man with 8 hours a day for sleep. 365.24 days a year are assumed. Based on the above, calculations are made, with results as follows:



How did you enjoy the calculations ? During a life of 80 years, studying accounts for 4.1% and working 10.6%. Whey not analyze and predict the use of your own time for you future reference and consideration ?

Don't forget to add your time working on your computer.



1.4 Reserved word

A BASIC sentence is composed of reserved words–also called key words–which include statements, built-in functions and special signs (and also commands), and other elements, such as constants, variables, arrays and expressions. Table 1.1 shows all reserved words of the BASIC interpreter SA-5510.

A	ABS		INT		RETURN
	ASC	L	LEFT$		RIGHT$
	ATN		LEN		RND
	AUTO		LET		ROPEN/T
C	CHARACTER$		LIMIT		RUN
	CHR$		LIST	S	SAVE
	CLOSE/T		LIST/P		SET
	CLR		LN		SGN
	CONT		LOAD		SIN
	COPY/P		LOG		SIZE
	COS	M	MID$		SPACE$
	CSRH		MON		SQR
	CSRV		MUSIC		STEP
	CURSOR	N	NEW		STOP
D	DATA		NEXT		STR$
	DEF FN	O	ON		STRING$
	DIM		OUT	T	TAB
E	END	P	PAGE/P		TAN
	EXP		PEEK		TEMPO
F	FOR		POKE		THEN
G	GET		PRINT		TI$
	GOSUB		PRINT/P		TO
	GOTO		PRINT/T	U	USR
I	IF	R	READ	V	VAL
	INP		REM		VERIFY
	INPUT		RESET	W	WOPEN/T
	INPUT/T		RESTORE		

TABLE 1.1 All reserved words of the BASIC interpreter SA-5510

1.5 List of BASIC interpreter SA-5510 commands, statements and functions

1.5.1 Commands

LOAD	LOAD "A"	Loads the BASIC text assigned the file name "A" from the cassette tape into the text area.
	LIMIT $A000: LOAD "B"	To load a machine language program file to be linked with a BASIC text, the BASIC area of memory must be partitioned from the machine language area by the LIMIT statement. Note: When a LOAD command is executed for a BASIC text file, the text area is cleared of any programs previously stored.
SAVE	SAVE "C"	Assignes the file name "C" to the BASIC text in the text area and stores it on the cassette tape. File name is valid up to 16 characters.
RUN	RUN	Executes the BASIC text in the text area from the top. Note: The RUN command clears all variables (fills them with 0 or null string) before running text.
	RUN 1000	Executes the BASIC text starting at line number 1000.
VERIFY	VERIFY "C"	This command compares the program contained in the BASIC text area with its equivalent text assigned the file name "C" in the cassette tape file.
AUTO	AUTO	Automatically generates and assigns line numbers 10, 20, 30 ...... during creation.
	AUTO 200, 20	Automatically generates line numbers at intervals 20 starting at line 200. 200, 220, 240 .... An AUTO command is terminated by pressing the BREAK key.
LIST	LIST	Displays all lines of BASIC text currently contained in the text area.
	LIST -500	Displays all lines of BASIC text up through line 500.
LIST/P	LIST/P	Prints out all lines contained in the BASIC text area on the line printer.
NEW	NEW	Clears the text area and variable area. Further, disestablishes the machine language program area set by a LIMIT statement by removing the partition.

CONT	CONT	Continues program execution which was halted by a STOP statement or the BREAK key, starting at the statement following the STOP statement or the statement halted by the BREAK key.
MON	MON	Transfers system control from the BASIC interpreter to the MONITOR. (To transfer system control from the MONITOR to the BASIC interpreter, execute monitor command J.)

1.5.2 Assignment statement

LET	⟨LET⟩ A = X + 3	Substitutes X + 3 into numeric variable A. LET may be omitted.

1.5.3 Input/output statements

PRINT	10 PRINT A	Displays the numeric value of A on the CRT screen.
	? A$	Displays the character string of variable A$ on the CRT screen.
	100 PRINT A; A$, B; B$	Combinations of numeric variables and string variables can be specified in a PRINT statement. When a semicolon is used as the separator, no space is displayed between the data strings. When a colon is used, variable data to the right of the colon is displayed from the next tab set position. (A tab is set every 10 character positions.)
	110 PRINT "COST ="; C$	Displays the string between double quotation marks as is, and C$.
	120 PRINT	Performs a new line operation (i.e., advances the cursor one line).
INPUT	10 INPUT A	Obtains numeric data for variable A from the keyboard.
	20 INPUT A$	Obtains string data for string variable A$ from the keyboard.
	30 INPUT "VALUE?"; D	Displays "VALUE?" on the screen before obtaining data from the keyboard. A semicolon separates the string from the variable.
	40 INPUT X, X$, Y, Y$	Numeric variables and string variables can be used in combination by separating them from each other with a comma. The types of data entered from the keyboard must be the same as those of the corresponding variables.
GET	10 GET N	Obtains a numeral for variable N from the keyboard. When no key is pressed, zero is substituted into N.
	20 GET K$	Obtains a character for variable K$ from the keyboard. When no key is pressed, a null is substituted into K$.

READ~DATA		Substitutes constants specified in the DATA statement into the corresponding variables specified in the READ statement. The corresponding constant and variable must be of the same data type.
	10 READ A, B, C 1010 DATA 25, −0.5, 500	In READ and DATA statements at left, values of 25, −0.5 and 500 are substitutes for variables A, B and C, respectively.
	10 READ H$, H, S$, S 30 DATA HEART, 3 35 DATA SPADE, 11	In the example at left, the first string constant of the DATA statement on line number 10 is substituted into the first variable of the READ statement; that is; "HEART" is substituted into H$. Then, numeric constant 3 is substituted into numeric variable H, and so on.
RESTORE		With a RESTORE statement, data in the following DATA statement which has already been read by preceding READ statements can be re-read from the beginning by the following READ statements.
	10 READ A, B, C 20 RESTORE 30 READ D, E 100 DATA 3, 6, 9, 12, 15	The READ statement on line number 10 substitutes 3, 6 and 9 into variables A, B and C, respectively. Because of the RESTORE statement, the READ statement on line number 30 substitutes not 12 and 15, but 3 and 6 again into D and E, respectively.

1.5.4 Loop statement

FOR ~ TO NEXT	10 FOR A=1 TO 10 20 PRINT A 30 NEXT A	The statement on line number 10 specifies that the value of variable A is varied from 1 to 10 in increments of one. The initial value of A is 1. The statement on line number 20 displays the value of A. The statement on line number 30 increments the value of A by one and returns program execution to the statement on line number 10. Thus, the loop is repeated until the value of A becomes 10. (After the specified number of loops has been completed, the value of A is 11.)
	10 FOR B=2 TO 8 STEP 3 20 PRINT B ↑ 2 30 NEXT	The statement on line number 10 specifies that the value of variable B is varied from 2 to 8 in increments of 3. The value of STEP may be made negative to decrement the value of B.
	10 FOR A=1 TO 3 20 FOR B=10 TO 30 30 PRINT A, B 40 NEXT B 50 NEXT A	The FOR-NEXT loop for variable A includes the FOR-NEXT loop for variable B. As is shown in this example, FOR-NEXT loops can be enclosed in other FOR-NEXT loops at different levels. Lower level loops must be completed within higher level loops. The maximum number of levels of FOR-NEXT loops is 16.
	60 NEXT B, A 70 NEXT A, B	In substitution for NEXT statement at line numbers 40 and 50, a statement at line number 60 shown at left can be used. However, statement at line number 70 cannot be used, causing an error to occur.

1.5.5 Branch statements

GOTO	100 GOTO 200	Jumps to the statement on line number 200.
GOSUB ~ RETURN	100 GOSUB 700 .... 800 RETURN	Calls the subroutine starting on line number 700. At the end of subroutine, program execution returns to the statement following the corresponding GOSUB statement.
IF ~ THEN	10 IF A>20 THEN 200	Jumps to the statement on line number 200 when the value of variable A is more than 20; otherwise the next line is executed.
	50 IF B<3 THEN B=B+3	Substitutes B+3 into variable B when the value of B is less than 3; otherwise the next line is executed.
IF ~ GOTO	100 IF A>=B THEN 10	Jumps to the statement on line number 10 when the value of variable A is equal to or greater than the value of B; otherwise the next line is executed.
IF ~ GOSUB	30 IF A=B*2 GOSUB 90	Jumps to the subroutine starting on line number 700 when the value of variable A is twice the value of B; otherwise the next statement is executed. (When other statements follow a conditional statement on the same line and the conditions are not satisfied, those following an ON statement are executed sequentially, but those following an IF statement are ignored and the statement on the next line is executed.)
ON ~ GOTO	50 ON A GOTO 70, 80, 90	Jumps to the statement on line number 70 when the value of variable A is 1, to the statement on line number 80 when it is 2 and to the statement on line number 90 when it is 3. When the value of A is 0 or more than 3, the next statement is executed. This statement has the same function as the INT function, so that when the value of A is 2.7, program execution jumps to the statement on line number 80.
ON ~ GOSUB	90 ON A GOSUB 700, 800	Jumps to the subroutine on line number 700 when the value of variable A is 1 and jumps to the subroutine on line number 800 when it is 2.

1.5.6 Definition statements

DIM		When an array is used, the number of array elements must be declared with a DIM statement. The number of elements ranges from 0 to 255.
	10 DIM A(20)	Declares that 21 array elements, A(0) through A(20), are used for one-dimensional numeric array A(n).

	20 DIM B(79, 79)	Declares that 6400 array elements, B(0, 0) through B(79, 79), are used for two-dimensional numeric array B(m, n).
	30 DIM C1$(10)	Declares that 11 array elements, C1$(0) through C1$(10), are used for one-dimensional string array C1$(n).
	40 DIM K$(7, 5)	Declares that 48 array elements, K$(0, 0) through K$(7, 5), are used for two-dimensional string array K$ (m, n).
DEF FN	100 DEF FNA (X)=X↑2−X 110 DEF FNB(X)=LOG(X) +1 120 DEF FNZ(Y)=LN(Y)	A DEF FN statement defines a function. The statement on line number 100 defines FNA(X) as X^2-X. The statement on line number 110 defines FNB(X) as $\log_{10}X + 1$ and the statement on line number 120 defines FNZ(Y) as $\log_e Y$. The number of variables included in the function must be 1.

1.5.7 Comment and control statements

REM	200 REM JOB-1	Comment statement (not executed).
STOP	850 STOP	Stops program execution and awaits a command entry. When a CONT command is entered, program execution is continued.
END	1999 END	Declares the end of a program. Although the program is stopped, the following program is executed if a CONT command is entered.
CLR	300 CLR	Clears all variables and arrays, that is, fills all numeric variables and arrays with zeros and all string variables and arrays with nulls.
CURSOR	50 CURSOR 25, 15 60 PRINT "ABC"	The CURSOR command moves the cursor to any position on the screen. The first operand represents the horizontal location of the destination, and must be between 0 and 39. The second operand repressents the vertical location of the destination and must be between 0 and 24. The left example displays "ABC" starting at location (25, 15) (the 26th position from the left side and the 16th position from the top).
CSRH		System variable indicating the X-coordinate (horizontal location) of the cursor.
CSRV		System variable indicating the Y-coordinate (vertical location) of the cursor.
SIZE	? SIZE	Displays the amount of unused memory area in bytes.
TI$	100 TI$ = "102030"	Sets the built-in clock to 10:20:30 AM. Data between the double quotation marks must be numerals.

1.5.8 Music control statements

MUSIC TEMPO		The MUSIC statement generates a melody from the speaker according to the melody string data enclosed in quotation marks or string variables at the tempo specified by the TEMPO statement.
	300 TEMPO 7 310 MUSIC "DE#FGA"	The TEMPO statement on line number 300 specifies tempo 7. The MUSIC statement on line number 310 generates a melody consisting of D, E, F sharp, G and A. Each note is a quarter note. When the TEMPO statement is omitted, default tempo is set.
	300 M1$ = "C3EG + C" 310 M2$ = "+E+C+E+G" 320 M3$ = "+#B8R5" 330 MUSIC M1$,M2$,M3$	In this example, the melody is divided into 3 parts and substituted in 3 string variables. The following melody is generated from the speaker at tempo 4. 

1.5.9 Graphic control statements

SET		Sets a dot in the specified position on the CRT screen. The first operand specifies the X-coordinates (0-79) and the second operand specifies the Y-coordinates (0-49).
	300 SET 40, 25	Displays a dot in the center of the screen.
RESET		Resets a dot in the specified position on the CRT screen.
	310 RESET 40, 25	Resets a dot from the center of the screen.

1.5.10 Cassette data file input/output statements

WOPEN/T	10 WOPEN/T "DATA-1"	Defines the file name of a cassette data file to be created as "DATA-1" and opens.
PRINT/T	20 PRINT/T A, A$	Writes the contents of variable A and string variable A$ in order in the cassette data file which was opened by a WOPEN/T statement.
CLOSE/T	30 CLOSE/T	Closes the cassette data file which was opened by a WOPEN/T statement.
ROPEN/T	110 ROPEN/T "DATA-2"	Opens the cassette data file specified with file name "DATA-2".
INPUT/T	120 INPUT/T B, B$	Reads data sequentially from the beginning of the cassette data file which was opened by the ROPEN/T statement and substitutes numerical data into variable B and string data into string variable B$ respectively.
CLOSE/T	130 CLOSE/T	Closes the cassette data file which was opened by a ROPEN/T statement.

1.5.11 Machine language control statements

LIMIT	100 LIMIT 49151	Limits the area in which BASIC programs can be loaded to the area up to address 49151 ($BFFF in hexadecimal).
	100 LIMIT A	Limits the area in which BASIC programs can be loaded to the area up to the address indicated by variable A.
	100 LIMIT $BFFF	Limits the area in which BASIC programs can be loaded to the area up to $BFFF (hexadecimal). Hexadecimal numbers are indicated by a dollar sign as shown at left.
	300 LIMIT MAX	Set the maximum address of the area in which BASIC programs can be loaded to the maximum address of the memory installed.
	200 LIMIT $BFFF 210 LOAD "S-R1"	Loads machine language program (object program) "S-R1" in the machine language link area from the cassette tape when the loading address of the program is $C000 or higher.
POKE	120 POKE 49450, 175	Stores 175 ($AF in hexadecimal) in address 49450.
	130 POKE AD, DA	Stores data (between 0 and 255) specified by variable DA into the address indicated by variable AD.
PEEK	150 A=PEEK (49450)	Substitutes data stored in address 49450 into variable A.
	160 B=PEEK (C)	Substitutes the contents of the address indicated by variable C into variable B.
USR	500 USR (49152)	Transfers program control to address 49152. This function is the same as that performed by the CALL instruction, which calls a machine language program. When a RET command is encountered in the machine language program, program control is returned to the BASIC program.
	550 USR (AD)	Calls the program starting at the address specified by variable AD.
	570 USR ($C000)	Calls the program starting at address $C000.
	770 USR (AD, DA$)	When string data is given together with address data, this USR function places the first address of the memory area containing string data DA$ in the CPU's DE register and the length of DA$ in the BC register prior to execution of a CALL instruction.

1.5.12 Printer control statements

PRINT/P		Performs the nearly same operation as the PRINT statement on the optional printer (MZ-80P4, P5 or P6).
	10 PRINT/P A, A$	Prints the numeric value of A and the character string of variable A$ on the line printer.
	20 PRINT/P CHR$ (5)	Executes paper home feed. (CHR$(5) is a control code.)
	30 PRINT/P CHR$(18)	Sets the enlarged character print mode. (CHR$(18) is also a control code.)
COPY/P	10 COPY/P 1	Causes the printer to copy the character display.
PAGE/P	100 PAGE/P 20	Specifies 20 lines to be contained in one page of the line printer.

1.5.13 I/O input/output statements

INP		Reads data on the specified I/O port.
	10 INP @12, A 20 PRINT A	The statement on line number 10 reads data on I/O port 12.
OUT		Outputs data to the specified I/O port.
	30 B = ASC ("A") 40 OUT @13, B	The statement on line 40 outputs the ASCII code of the character "A" to I/O port 13.

1.5.14 Arithmetic functions

ABS	100 A = ABS (X)	Substitutes the absolute value of variable X into variable A. X may be either a constant or an expression. Ex) ABS (−3) = 3 ABS (12) = 12
INT	100 A = INT (X)	Substitutes the greatest integer which is less than X into variable A. X may be either a numeric constant or an expression. Ex) INT (3.87) = 3 INT (0.6) = 0 INT (−3.87) = −4
SGN	100 A = SGN (X)	Substitutes one of the following values into variable A: −1 when X<0, 0 when X=0 and 1 when X>0. X may be either a constant or an expression. Ex) SGN (0.4) = 1 SGN (0) = 0 SGN (−400) = −1

SQR	100 A = SQR (X)	Substitutes the square root of variable X into variable A. X may either a numeric constant or an expression; however, it must be greater than or equal to 0.
SIN	100 A = SIN (X)	Substitutes the sine of variable X in radians into variable A. X may be either a numeric constant or an expression. The relationship between degrees and radians is as follows. $1 \text{ degree} = \frac{\pi}{180} \text{ radians}$
	110 A = SIN (30∗π/180)	Therefore, when substituting the sine of 30° into A, the statement is written as shown on line number 110 at left.
COS	200 A = COS (X) 210 A = COS (200∗π/180)	Substitutes the cosine of variable X in radians into variable A. X may be either a numeric constant or an expression. The same relationship as shown in the explanation of the SIN function is used to convert degrees into radians. The statement shown on line number 210 substitutes the cosine of 200° into variable A.
TAN	300 A = TAN (X) 310 A = TAN (Y∗π/180)	Substitutes the tangent of variable X in radians into variable A. X may be either a numeric constant or an expression. The statement on line number 310 is used to substitute the tangent of numeric variable Y in degrees into variable A.
ATN	400 X = ATN (A) 410 Y = 180/π∗ATN (A)	Substitutes the arctangent of variable A into variable X in radians. A may be either a numeric constant or an expression. Only the result between $-\pi/2$ and $\pi/2$ is obtained. The statement on line number 410 is used to substitute the arctangent in degrees.
EXP	100 A = EXP (X)	Substitutes the value of exponential function e^x into variable A. X may either a numeric constant or an expression.
LOG	100 A = LOG (X)	Substitutes the value of the common logarithm of variable X into variable A. X may be either a numeric constant or an expression; however, it must be positive.
LN	100 A = LN (X)	Substitutes the natural logarithm of variable X into variable A. X may be either a numeric constant or an expression; however, it must be positive.
	110 A = LOG (X)/LOG (Y) 120 A = LN (X)/LN (Y)	To obtain the logarithm of X with the base Y, the statement on line number 110 or line number 120 is used.
RND		This function generates random numbers which take any value between 0.00000001 and 0.99999999, and works in two manners depending upon the value in parentheses.

	100 A = RND (1) 110 B = RND (10)	When the value in parentheses is positive, the function gives the random number following the one previously given in the random number group generated. The value obtained is independent of the value in parentheses.
	100 A = RND (0) 110 B = RND (–3)	When the value in parentheses is less than or equal to 0, the function gives the initial value of the random number group generated. Therefore, statements on line numbers 100 and 110 both give the same value to variables A and B.

1.5.15 String control functions

LEFT $	10 A$ = LEFT$ (X$, N)	Substitutes the first N characters of string variable X$ into string variable A$. N may be either a constant, a variable or an expression.
MID $	20 B$ = MID$ (X$, M, N)	Substitutes the N characters following the Mth character from the beginning of string variable X$ into string variable B$.
RIGHT $	30 C$ = RIGHT $ (X$, N)	Substitutes the last N characters of string variable X$ into string variable C$.
SPACE $	40 D$ = SPACE $ (N)	Substitutes the N spaces into string variable D$.
STRING $	50 E$ = STRING $ ("*", 10)	Substitutes the ten repetitions of "*" into string varialble E$.
CHR $	60 F$ = CHR $ (A)	Substitutes the character corresponding to the ASCII code in numeric variable A into string variable F$. A may be either a constant, a variable or an expression.
ASC	70 A = ASC (X$)	Substitutes the ASCII code (in decimal) corresponding to the first character of string variable X$ into numeric variable A.
STR$	80 N$ = STR$ (I)	Converts the numeric value of numeric variable I into string of numerals and substitutes it into string variable N$.
VAL	90 I = VAL (N$)	Converts string of numerals contained in string variable N$ into the numeric data as is and substitutes it into numeric variable I.
LEN	100 LX = LEN (X$)	Substitutes the length (number of bytes) of string variable X$ into numeric variable LX.
	110 LS = LEN (X$ + Y$)	Substitutes the length (number of bytes) of string variable X$ and Y$ into numeric variable LX.

1.5.16 Tabulation function

TAB	10 PRINT TAB (X); A	Displays the value of variable A at the Xth position from the left side.

1.5.17 Arithmetic operators

The number to the left of each operator indicates its operational priority. Any group of operations enclosed in parentheses has first priority.

❶ ↑	10 A = X ↑ Y (power)	Substitutes X^Y into variable A. (If X is negative and Y is not an integer, an error results.)
❷ −	10 A = −B (negative sign)	Note that "−" in −B is the negative sign and "−" in 0-B represents subtraction.
❸ *	10 A = X * Y (multiplication)	Multiplies X by Y and substitutes the result into variable A.
❸ /	10 A = X/Y (division)	Divides X by Y and substitutes the result into variable A.
❹ +	10 A = X + Y (addition)	Adds X and Y and substitutes the result into variable A.
❹ −	10 A = X − Y (subtraction)	Subtracts X from Y and substitutes the result into variable A.

1.5.18 Logical operators

=	10 IF A=X THEN . . .	If the value of variable A is equal to X, the statement following THEN is executed.
	20 IF A$ = "XYZ" THEN . . .	If the content of variable A$ is "XYZ", the statement following THEN is executed.
>	10 IF A>X THEN . . .	If the value of variable A is greater than X, the statement following THEN is executed.
<	10 IF A<X THEN . . .	If the value of variable A is less than X, the statement following THEN is executed.
<> or ><	10 IF A<>X THEN . . .	If the value of variable A is not equal to X, the statement following THEN is executed.
>= or =>	10 IF A>=X THEN . . .	If the value of variable A is greater than or equal to X, the statement following THEN is executed.
<= or =<	10 IF A<=X THEN . . .	If the value of variable A is less than or equal to X, the statement following THEN is executed.
*	40 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, the statement following THEN is executed.
+	50 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 the value of Y, the statement following to THEN is executed.

1.5.19 Other symbols

?	200 ? "A + B ="; A + B 210 PRINT "A + B ="; A + B	Can be used instead of PRINT. Therefore, the statement on line number 200 is identical in function to that on line number 210.
:	220 A=X : B=X↑2 : ?A, B	Separates two statements from each other. This separator is used when multiple statements are written on the same line. Three statements are written on line number 220.
;	230 PRINT "AB";"CD"; "EF"	Displays characters to the right of separators following characters on the left. The statement on line 230 displays "ABCDEF" on the screen with no spaces between characters.
	240 INPUT "X =" ; X$	Displays "X=" on the screen and awaits entry of data for X$ from the keyboard.
,	250 PRINT "AB", "CD", "E"	Displays character strings in a tabulated format; i.e. AB first appears, then CD appears in the position corresponding to the starting position of A plus 10 spaces and E appears in the position corresponding to the starting position of C plus 10 spaces.
	300 DIM A(20), B$(3, 6)	A comma is used to separate two variables.
" "	320 A$ = "SHARP BASIC" 330 B$ = "MZ-80A"	Indicates that characters between double quotation marks form a string constant.
$	340 C$ = "ABC"+CHR$(3)	Indicates that the variable followed by a dollar sign is a string variable.
	500 LIMIT $BFFF	Indicates that numeric data following a dollar sign is represented in hexadecimal notation.
π	550 S = SIN (X ∗ π/180)	π represents 3.1415927 (ratio of the circumference of a circle to its diameter).

1.5.20 Error Message Table

Error No.	Meaning
1	Syntax error
2	Operation result overflow
3	Illegal data
4	Data type mismatch
5	String length exceeded 255 characters
6	Insufficient memory capacity
7	The size of an array defined was larger than that defined previously.
8	The length of a BASIC text line was too long.
9	
10	The number of levels of GOSUB nests exceeded 16.
11	The number of levels of FOR-NEXT nests exceeded 16.
12	The number of levels of functions exceeded 6.
13	NEXT was used without a corresponding FOR.
14	RETURN was used without a corresponding GOSUB.
15	Undefined function was used.
16	Unused reference line number was specified in a statement.
17	CONT command cannot be executed.
18	A writing statement was issued to the BASIC control area.
19	Direct mode commands and statements are mixed together.
20	
21	
22	
23	
24	A READ statement was used without a corresponding DATA statement.
25	
26	
27	
28	
29	
30	
31	
32	
33	
34	
35	

Error No.	Meaning
36	
37	
38	
39	
40	
41	
42	
43	OPEN statement (ROPEN or WOPEN) was issued to a file which is already open.
44	
45	
46	
47	
48	
49	
50	
51	
52	
53	
54	
55	
56	
57	
58	
59	
60	
61	
62	
63	Out of file
64	
65	The printer is not ready.
66	Printer hardware error
67	Out of paper
68	
69	
70	Check sum error

1.6 How to obtain copied BASIC tapes

The BASIC tape will not be exchanged for new ones after purchase. It is recommended that the original BASIC tape be copied using the following procedure to generate a copied BASIC tape, and that the copied BASIC tape be used ordinarily. Be sure to keep the original BASIC tape in a safe place.

Activate BASIC interpreter by the original BASIC tape. Execute the following command.

USR ($11FD)

the message "↓RECORD. PLAY" is displayed. Set the new cassette tape into the deck and press the RECORD and PLAY buttons.

Upon completion of writing, a copied BASIC tape will be obtained. Any number of copied BASIC tapes can be made using this procedure. However, copied BASIC tape cannot be made by copying another copied BASIC tape.

Monitor Program of the MZ-80A

Chapter 2

2.1 Monitor SA-1510 Commands and subroutines

A monitor program generally monitors system programs such as the BASIC interpreter. The MZ-80A uses a Monitor program called MONITOR SA-1510 in 4K bytes ROM. It includes various functional subroutines which control the keyboard, display, sound circuit, cassette tape deck, etc. These subroutines are called by the BASIC interpreter when it executes INPUT statement, SAVE command, MUSIC statement or other commands or statements. Monitor subroutines may also be called by the user at will.

MONITOR SA-1510 occupies 4K bytes of memory and is stored in Monitor ROM addresses $0000 through $0FFF. Its required work area is included within memory addresses $1000 through $11FF.

2.1.1 Using monitor commands

Following shows the message when the power switch of the MZ-80A is turned on.

```
* * MONITOR SA-1510 * *
* ▩
```

The cursor flickers to inform the operator that system control is in Monitor command level. Monitor commands are as follows:

L [CR] .... Loads the cassette tape file into memory.

J xxxx [CR] ..... Transfers system control to the specified address, that is, loads the specified address in the program counter of the CPU.
xxxx : 4-digit hexadecimal number
The start addresses of the BASIC SA-5510 are as follows:
Warm start address = $1250
Cold start address = $1200

F [CR] .... Transfers system control to the floppy disk drive control routine which is stored on floppy disk drive interface card.

B [CR] .... Sets or resets the key-entry-bell alternately.

2.1.2 Monitor subroutines

MONITOR SA-1510 subroutines are listed in Table 2.1. The subroutine names indicated are the same as the labels shown in the monitor program assembly listing in 2.2. Each name is a mnemonic representing the subroutine's function.

Table 2.1 **Monitor Subroutine List**

Subroutine name (hexadecimal address)	Function	Registers preserved
CALL LETNL ($0006)	To change the line and set the cursor to the beginning of the next line.	All registers except AF
CALL NL ($0009)	Changes the line and sets cursor to its beginning if the cursor is not already located at the beginning of a line.	All registers except AF
CALL PRNTS ($000C)	Displays one space only at the cursor position on the display screen.	All registers except AF
CALL PRNT ($0012)	Handles data in A register as ASCII code and displays it on the screen, starting at the cursor position. However, a carriage return is performed for 0DH and the various cursor control operations are performed for 11H-16H when these are included.	All registers except AF
CALL MSG ($0015)	Displays a message, starting at the cursor position on the screen. The starting address of the message must be specified in the DE register in advance. The message is written in ASCII code and must end in 0DH. A carriage return is not executed, however, cursor control codes (11H-16H) are.	All registers
CALL MSGX ($0018)	Almost the same as MSG, except that cursor control codes are for reverse character display.	All registers
CALL BELL ($003E)	Sounds a tone (approximately 880 Hz) momentarily.	All registers except AF
CALL MELDY ($0030)	Plays music according to music data. The starting address of the music data must be specified in advance in the DE register. As with BASIC, the musical interval and the duration of notes of the musical data are expressed in that order in ASCII code. The end mark must be either 0DH or C8H " ■ ". The melody is over if C flag is 0 when a return is made; if C flag is 1 it indicates that SHIFT + BREAK were pressed.	All registers except AF
CALL XTEMP ($0041)	Sets the musical tempo. The tempo data (01 to 07) is set in and called from A register. 01 : Slowest 04 : Medium speed 07 : Fastest Care must be taken here to ensure that the tempo data is entered in A register in binary code, and not in the ASCII code corresponding to the numbers 1 to 7 (31H to 37H).	All registers
CALL MSTA ($0044)	Continuously sounds a note according to a specified division factor. The division factor *nn'* consists of two bytes of data ; *n'* is stored at address 11A1H and *n* is stored at address 11A2H. The relationship between the division factor and the frequency produced is 2 MHz/*nn'*.	BC and DE only

Subroutine name (hexadecimal address)	Function	Registers preserved
CALL MSTP ($0047)	Discontinues a tone being sounded.	All registers except AF
CALL TIMST ($0033)	Sets the built-in clock. (The clock is activated by this call.) The call conditions are: A register ← 0 (AM), A register ← 1 (PM) DE register ← the time in seconds (2 bytes)	All registers except AF
CALL TIMRD ($003B)	Reads the value of the built-in clock. The conditions upon return are: A register ← 0 (AM), A register ← 1 (PM) DE register ← the time in seconds (2 bytes)	All registers except AF and DE
CALL BRKEY ($001E)	Checks whether SHIFT + BREAK were pressed. Z flag is set if they were pressed, and Z flag is reset if they were not.	All registers except AF
CALL GETL ($0003)	Inputs one line entered from the keyboard. The starting address where the data input is to be stored must be specified in advance in the DE register. CR functions as the end mark. 80 is the maximum number of characters which can be input (including the end mark 0DH). Key input is displayed on the screen and cursor control is also accepted. The BREAK code (1BH) followed by a carriage return code (0DH) is set at the beginning of the address specified in the DE register when SHIFT + BREAK are pressed.	All registers
CALL GETKY ($001B)	Takes one character only into A register from the keyboard in ASCII code. A return is made after 00 is set in A register if no key is pressed when the subroutine is executed. However, key input is not displayed on the screen. Codes which are taken into A register when these special keys are pressed are shown below.	All registers except AF

Special key	Code taken into A register
DEL	60 H
INST	61 H
GRPH graphic mode	62 H
GRPH normal mode	63 H
BREAK	64 H
CR or ENT	66 H
CTRL + A ~ Z	01 H ~ 1 AH
CTRL + [	1 BH
CTRL + \	1 CH
CTRL +]	1 DH
CTRL + ^	1 EH
CTRL + –	1 FH

<table>
<tr><th>Subroutine name (hexadecimal address)</th><th>Function</th><th>Registers preserved</th></tr>
<tr><td>CALL PRTHL
($03BA)</td><td>Displays the contents of the HL register on the display screen as a 4-digit hexadecimal number.</td><td>All registers except AF</td></tr>
<tr><td>CALL PRTHX
($03C3)</td><td>Displays the contents of the A register on the display screen as a 2-digit hexadecimal number.</td><td>All registers except AF</td></tr>
<tr><td>CALL ASC
($03DA)</td><td>Converts the contents of the lower 4 bits of A register from hexadecimal to ASCII code and returns after setting the converted data in A register.</td><td>All registers except AF</td></tr>
<tr><td>CALL HEX
($03F9)</td><td>Converts the 8 bits of A register from ASCII code to hexadecimal and returns after setting the converted data in the lower 4 bits of A register.
When C flag = 0 upon return A register ← hexadecimal
When C flag = 1 upon return A register is not assured</td><td>All registers except AF</td></tr>
<tr><td>CALL HLHEX
($0410)</td><td>Handles a consecutive string of 4 characters in ASCII code as hexadecimal string data and returns after setting the data in the HL register. The call and return conditions are as follows.
DE ← starting address of the ASCII string (string "3" "1" "A" "5")
CALL HLHEX (↑ DE)
C flag = 0 HL ← hexadecimal number (e.g., HL = 31A5H)
C flag = 1 HL is not assured.</td><td>All registers except AF and HL</td></tr>
<tr><td>CALL 2HEX
($041F)</td><td>Handles 2 consecutive ASCII strings as hexadecimal strings and returns after setting the data in A register. The call and return conditions are as follows.
DE ← starting address of the ASCII string (e.g., "3" "A")
CALL 2HEX (↑ DE)
C flag = 0 A register ← hexadecimal number (e.g., A register = 3AH)
C flag = 1 A register is not assured.</td><td>All registers except AF and DE</td></tr>
<tr><td>CALL ??KEY
($09B3)</td><td>Awaits key input while causing the cursor to flash. When a key entry is made it is converted to display code and set in A register, then a return is made.</td><td>All registers except AF</td></tr>
<tr><td>CALL ?ADCN
($0BB9)</td><td>Converts an ASCII value to display code. Call and return conditions are as follows.
A register ← ASCII value
CALL ?ADCN
A register ← display code</td><td>All registers except AF</td></tr>
<tr><td>CALL ?DACN
($0BCE)</td><td>Converts a display code to an ASCII value. Call and return conditions are as follows.
A register ← display code
CALL ?DACN
A register ← ASCII value</td><td>All registers except AF</td></tr>
<tr><td>CALL ?DPCT
($0DDC)</td><td>Controls the display on the display screen. The relationship between A register at the time of the call and control is as follows.
<table>
<tr><th>A reg.</th><th>Same function</th><th>A reg.</th><th>Same function</th></tr>
<tr><td>C0H</td><td>Scrolling</td><td>C8H</td><td>INST</td></tr>
<tr><td>C1H</td><td>↓</td><td>C9H</td><td>GRPH (graphic → normal)</td></tr>
<tr><td>C2H</td><td>↑</td><td>CAH</td><td>GRPH (normal → graphic)</td></tr>
<tr><td>C3H</td><td>→</td><td>CCH</td><td>CTRL + @ (rev. ↔ norm.)</td></tr>
<tr><td>C4H</td><td>←</td><td>CDH</td><td>CR or ENT</td></tr>
<tr><td>C5H</td><td>HOME</td><td>CEH</td><td>CTRL + D (roll up)</td></tr>
<tr><td>C6H</td><td>CLR</td><td>CFH</td><td>CTRL + E (roll down)</td></tr>
<tr><td>C7H</td><td>DEL</td><td></td><td></td></tr>
</table></td><td>All registers</td></tr>
</table>

Subroutine name (hexadecimal address)	Function	Registers preserved
CALL ?BLNK ($0DA6)	Checks vertical blanking of the display screen. Waits until the vertical blanking interval starts and then returns when blanking takes place.	All registers

CALL ?PONT ($0FB1)	Sets the current position of the cursor on the display screen in register HL. The return conditions are as follows.
	CALL ?PONT
	HL ← cursor position on the display screen (V-RAM address)
	(Note) The X-Y coordinates of the cursor are contained in DSPXY (1171 H). The current position of the cursor is loaded as follows.
	LD HL, (DSPXY) ; H ← Y coordinate on the screen
	L ← X coordinate on the screen
	The cursor position is set as follows.
	LD (DSPXY), HL
		All registers except AF and HL

CALL WRINF ($0021)	Writes the current contents of a certain part of the header buffer (described later) onto the tape, starting at the current tape position. Return conditions C flag = 0 No error occurred. C flag = 1 The BREAK key was pressed.	All registers except AF
CALL WRDAT ($0024)	Writes the contents of the specified memory area onto the tape as a CMT data block in accordance with the contents of a certain part of the header buffer. Return conditions C flag = 0 No error occurred. C flag = 1 The BREAK key was pressed.	All registers except AF
CALL RDINF ($0027)	Reads the first CMT header found starting at the current tape position into a certain part of the header buffer. Return conditions C flag = 0 No error occurred. C flag = 1, A register = 1 A check sum error occurred. C flag = 1, A register = 2 The BREAK key was pressed.	All registers except AF
CALL RDDAT ($002A)	Reads in the CMT data block according to the current contents of a certain part of the header buffer. Return conditions C flag = 0 No error occurred. C flag = 1, A register = 1 A check sum error occurred. C flag = 1, A register = 2 The BREAK key was pressed.	All registers except AF
CALL VERFY ($002D)	Compares the first CMT file found following the current tape position with the contents of the memory area indicated by its header. Return conditions C flag = 0 No error occurred. C flag = 1, A register = 1 A match was not obtained. C flag = 1, A register = 2 The BREAK key was pressed.	All registers except AF

(Note) The contents of the header buffer at the specific addresses are as follows. The buffer starts at address $10F0 and consists of 116 bytes.

Address	Contents
IBUFE ($10F0)	This byte indicates one of the following file modes. 01. Object file (machine language program) 02. BASIC text file 03. BASIC data file 04. Source file (ASCII file) 05. Relocatable file (relocatable binary file) A0. PASCAL interpreter text file A1. PASCAL interpreter data file
IBU1 ($10F1~$1101)	These 17 bytes indicate the file name. However, since 0DH is used as the end mark, in actuality the file name is limited to 16 bytes. Example: S A M P L E 0D
IBU18 ($1102~$1103)	These two bytes indicate the byte size of the data block which is to follow.
IBU20 ($1104~$1105)	These two bytes indicate the data address of the data block which is to follow. The loading address of the data block which is to follow is indicated by "CALL RDDAT". The starting address of the memory area which is to be output as the data block is indicated by "CALL WRDAT".
IBU22 ($1106~$1107)	These two bytes indicate the execution address of the data block which is to follow.
IBU24 ($1108~$1163)	These bytes are used for supplemental information, such as comments.

Example

Address	Content	
10F0	01	; indicates an object file (machine language program).
10F1	'S'	; the file name is "SAMPLE".
10F2	'A'	
10F3	'M'	
10F4	'P'	
10F5	'L'	
10F6	'E'	
10F7	0D	
10F8	} Variable	
1101		
1102	00	; the size of the file is 2000H bytes.
1103	20	
1104	00	; the data address of the file is 1200H.
1105	12	
1106	50	; the execution address of the file is 1250H.
1107	12	

2.2 MONITOR SA-1510 Assembly Listing

The MONITOR SA-1510 assembly listing is shown in following pages.

This assembly listing was obtained with the Z80-Assembler of the MZ-80A Floppy Disk Operating System. The meaning of each column is as follows.

Relative address — Relocatable OBJ code — Label — Mnemonic (Op Code) — Operand — Comment

```
0000                MONIT:  ENT
0000 C34A00                 JP      START
0003                GETL:   ENT
0003 C3A807                 JP      ?GETL
0006                LETNL:  ENT
0006 C38009                 JP      ?LTNL
0009                NL:     ENT
0009 C37B09                 JP      ?NL
000C                PRNTS:  ENT
000C C39309                 JP      ?PRTS
000F                PRNTT:  ENT
000F C38409                 JP      ?PRTT
0012                PRNT:   ENT
0012 C39509                 JP      ?PRNT
0015                MSG:    ENT
0015 C39308                 JP      ?MSG
0018                MSGX:   ENT
0018 C3A108                 JP      ?MSGX           ; RST 3
001B                GETKY:  ENT
001B C3B308                 JP      ?GET
001E                BRKEY:  ENT
001E C3110D                 JP      ?BRK
```

Figure 2.1

Since the first address of MONITOR SA-1510 is $0000, relative addresses and relocatable OBJ codes may be regarded as absolute addresses and OBJ codes without interpretation.

This assembly listing is for reference only. The Sharp corporation is not obliged to answer any questions about the contents of this program.

```
**  Z80 ASSEMBLER SB-7201  <MZ-80A.MONITOR>  PAGE 01            09/04/81

0000                   ;
0000                   ;
0000                   ;      MONITOR PROGRAM
0000                   ;         (MZ-80A)
0000                   ;
0000                   ;                SA-1510
0000                   ;             REV. '81.8.26
0000                   MONIT:  ENT
0000 C34A00                    JP     START
0003                   GETL:   ENT
0003 C3A807                    JP     ?GETL
0006                   LETNL:  ENT
0006 C3B009                    JP     ?LTNL
0009                   NL:     ENT
0009 C37B09                    JP     ?NL
000C                   PRNTS:  ENT
000C C39309                    JP     ?PRTS
000F                   PRNTT:  ENT
000F C38409                    JP     ?PRTT
0012                   PRNT:   ENT
0012 C39509                    JP     ?PRNT
0015                   MSG:    ENT
0015 C39308                    JP     ?MSG
0018                   MSGX:   ENT
0018 C3A108                    JP     ?MSGX          ; RST 3
001B                   GETKY:  ENT
001B C3B308                    JP     ?GET
001E                   BRKEY:  ENT
001E C3110D                    JP     ?BRK
0021                   WRINF:  ENT
0021 C33604                    JP     ?WRI
0024                   WRDAT:  ENT
0024 C37004                    JP     ?WRD
0027                   RDINF:  ENT
0027 C3CF04                    JP     ?RDI
002A                   RDDAT:  ENT
002A C3EF04                    JP     ?RDD
002D                   VERFY:  ENT
002D C37505                    JP     ?VRFY
0030                   MELDY:  ENT
0030 C38801                    JP     ?MLDY          ; RST 6
0033                   TIMST:  ENT
0033 C3FA02                    JP     ?TMST
0036 00                        NOP
0037 00                        NOP
0038 C33810                    JP     1038H          ; INTERRUPT ROUTINE
003B                   TIMRD:  ENT
003B C34403                    JP     ?TMRD
003E                   BELL:   ENT
003E C3E502                    JP     ?BEL
0041                   XTEMP:  ENT
0041 C3EC02                    JP     ?TEMP
0044                   MSTA:   ENT
0044 C3AB02                    JP     MLDST
0047                   MSTP:   ENT
0047 C3BE02                    JP     MLDSP
004A                   ;
004A                   ;;;;;
004A                   ;
004A 31F010            START:  LD     SP,SP
```

```
**  Z80 ASSEMBLER SB-7201  <MZ-80A.MONITOR>  PAGE 02            09/04/81

004D ED56                      IM     1              ; IM 1 SET
004F CD4D07                    CALL   ?MODE
0052 06FF                      LD     B,FFH          ; BUFFER CLEAR
0054 21F110                    LD     HL,NAME
0057 CDD80F                    CALL   ?CLER
005A 3E16                      LD     A,16H          ; LASTER CLR.
005C CD1200                    CALL   PRNT
005F 3ECF                      LD     A,CFH
0061 2100D8                    LD     HL,D800H
0064 1803                      JR     +5
0066 C33510                    JP     1035H          ; JUMP NMI
0069 CDE309                    CALL   #CLR8
006C 217903                    LD     HL,TIMIN       ; INTERUPT
006F 3EC3                      LD     A,C3H
0071 323810                    LD     (1038H),A
0074 223910                    LD     (1039H),HL
0077 3E04                      LD     A,04
0079 329E11                    LD     (TEMPW),A
007C CDBE02                    CALL   MLDSP
007F CD0900                    CALL   NL
0082 110001                    LD     DE,MSG?3
0085 DF                        RST    3
0086 CDE502                    CALL   ?BEL
0089 3EFF              SS:     LD     A,FFH
008B 329D11                    LD     (SWRK),A       ; KEY IN SILENT
008E 2100E8                    LD     HL,E800H       ; USR ROM ?
0091 3655                      LD     (HL),55H
0093 1835                      JR     FD2
0095 CD0900            ST1:    CALL   NL
0098 3E2A                      LD     A,2AH          ; '*' PRINT
009A CD1200                    CALL   PRNT
009D 11A311                    LD     DE,BUFER
00A0 CD0300                    CALL   GETL
00A3 1A                ST2:    LD     A,(DE)
00A4 13                        INC    DE
00A5 FE0D                      CP     0DH
00A7 28EC                      JR     Z,ST1
00A9 FE4A                      CP     'J'            ; JUMP
00AB 280E                      JR     Z,GOTO
00AD FE4C                      CP     'L'            ; LOAD PROGRAM
00AF 2828                      JR     Z,LOAD
00B1 FE46                      CP     'F'            ; FLOPPY ACCESS
00B3 2812                      JR     Z,FD
00B5 FE42                      CP     'B'            ; KEY IN BELL
00B7 2808                      JR     Z,SG
00B9 18E8                      JR     ST2
00BB                   ;
00BB                   ;    JUMP COMMAND
00BB                   ;
00BB CD1004            GOTO:   CALL   HLHEX
00BE 38D5                      JR     C,ST1
00C0 E9                        JP     (HL)
00C1                   ;
00C1                   ;  KEY SOUND ON OFF
00C1                   ;
00C1 3A9D11            SG:     LD     A,(SWRK)
00C4 2F                        CPL
00C5 18C4                      JR     SS+2
00C7                   ;
00C7                   ;  FLOPPY
```

```
**  Z80 ASSEMBLER SB-7201  <MZ-80A.MONITOR>  PAGE 03          09/04/81

00C7                 ;
00C7 2100F0          FD:     LD     HL,F000H
00CA 7E              FD2:    LD     A,(HL)
00CB B7                      OR     A
00CC 20C7                    JR     NZ,ST1
00CE E9              FD1:    JP     (HL)
00CF                 ;
00CF                 ;
00CF                 ;   ERROR (LOADING)
00CF                 ;
00CF FE02            ?ER:    CP     02H
00D1 28C2                    JR     Z,ST1
00D3 111801                  LD     DE,MSGE1
00D6 DF                      RST    3               ; CALL MSGX
00D7 18BC                    JR     ST1
00D9                 ;
00D9                 ;
00D9                 ;  LOAD COMMAND
00D9                 ;
00D9 CDCF04          LOAD:   CALL   ?RDI
00DC 38F1                    JR     C,?ER
00DE CD0900          LOA0:   CALL   NL
00E1 11F700                  LD     DE,MSG?2
00E4 DF                      RST    3               ; CALL MSGX
00E5 11F110                  LD     DE,NAME
00E8 DF                      RST    3               ; CALL MSGX
00E9 CDEF04                  CALL   ?RDD
00EC 38E1                    JR     C,?ER
00EE 2A0611                  LD     HL,(EXADR)
00F1 7C                      LD     A,H
00F2 FE12                    CP     12H
00F4 389F                    JR     C,ST1
00F6 E9                      JP     (HL)
00F7 4C              MSG?2:  DEFM   'L'             ; LOADING
00F8 B7A1                    DEFW   A1B7H
00FA 9CA6                    DEFW   A69CH
00FC B097                    DEFW   97B0H
00FE 200D                    DEFW   0D20H
0100 2A2A2020        MSG?3:  DEFM   '**  MONITOR SA-1510  **'
0104 4D4F4E49
0108 544F5220
010C 53412D31
0110 35313020
0114 202A2A
0117 0D                      DEFB   0DH
0118                 ;
0118                 ;
0118 43              MSGE1:  DEFM   'C'             ; CHECK SUM ERROR
0119 9892                    DEFW   9298H
011B 9FA9                    DEFW   A99FH
011D 20A4                    DEFW   A420H
011F A5B3                    DEFW   B3A5H
0121 2092                    DEFW   9220H
0123 9D9D                    DEFW   9D9DH
0125 B79D                    DEFW   9DB7H
0127 0D                      DEFB   0DH
0128                 ;
0128                 ;  CR PAGE MODE1
0128                 ;
0128 CD2B0A          .CR:    CALL   .MANG
```

```
**  Z80 ASSEMBLER SB-7201  <MZ-80A.MONITOR>  PAGE 04          09/04/81

012B 0F                      RRCA
012C D2860E                  JP     NC,CURS2
012F 2E00                    LD     L,0
0131 24                      INC    H
0132 FE18                    CP     +24
0134 2804                    JR     Z,.CR1
0136 24                      INC    H
0137 C3660E                  JP     CURS1
013A 227111          .CR1:   LD     (DSPXY),HL
013D                 ;
013D                 ;  SCROL PAGE MODE1
013D                 ;
013D                 .SCROL: ENT
013D 01C003                  LD     BC,03C0H
0140 1100D0                  LD     DE,SCRN
0143 2128D0                  LD     HL,SCRN+40
0146 EDB0                    LDIR
0148 EB                      EX     DE,HL
0149 0628                    LD     B,+40
014B CDD80F                  CALL   ?CLER
014E 011A00                  LD     BC,26
0151 117311                  LD     DE,MANG
0154 217411                  LD     HL,MANG+1
0157 EDB0                    LDIR
0159 3600                    LD     (HL),0
015B 3A7311                  LD     A,(MANG)
015E B7                      OR     A
015F CAE50E                  JP     Z,?RSTR
0162 217211                  LD     HL,DSPXY+1
0165 35                      DEC    (HL)
0166 18D5                    JR     .SCROL
0168                 ;
0168                 ;
0168                 ;
0168                 ;  CTBL PAGE MODE1
0168                 ;
0168 3D01            .CTBL:  DEFW   .SCROL
016A 5D0E                    DEFW   CURSD
016C 6E0E                    DEFW   CURSU
016E 7B0E                    DEFW   CURSR
0170 950E                    DEFW   CURSL
0172 0904                    DEFW   HOMO
0174 B30E                    DEFW   CLRS
0176 F20E                    DEFW   DEL
0178 2D0F                    DEFW   INST
017A E10E                    DEFW   ALPHA
017C EE0E                    DEFW   KANA
017E E50E                    DEFW   ?RSTR
0180 170A                    DEFW   REV
0182 2801                    DEFW   .CR
0184 E50E                    DEFW   ?RSTR
0186 E50E                    DEFW   ?RSTR
0188                 ;
0188                 ;
0188                 ;
0188                 ;  MELODY
0188                 ;
0188                 ;   DE=DATA LOW ADR.
0188                 ;  EXIT   CF=1 BREAK
0188                 ;         CF=0 OK
```

```
** Z80 ASSEMBLER SB-7201  <MZ-80A.MONITOR>  PAGE 05            09/04/81

0188                 ;
0188                 ?MLDY:  ENT
0188 C5                      PUSH    BC
0189 D5                      PUSH    DE
018A E5                      PUSH    HL
018B 3E02                    LD      A,02H
018D 32A011                  LD      (OCTV),A
0190 0601                    LD      B,01
0192 1A              MLD1:   LD      A,(DE)
0193 FE0D                    CP      0DH             ; CR
0195 283B                    JR      Z,MLD4
0197 FEC8                    CP      C8H             ; END MARK
0199 2837                    JR      Z,MLD4
019B FECF                    CP      CFH             ; UNDER OCTAVE
019D 2827                    JR      Z,MLD2
019F FE2D                    CP      2DH             ; '-'
01A1 2823                    JR      Z,MLD2
01A3 FE2B                    CP      2BH             ; '+'
01A5 2827                    JR      Z,MLD3
01A7 FED7                    CP      D7H             ; UPPER OCTAVE
01A9 2823                    JR      Z,MLD3
01AB FE23                    CP      23H             ; "#"
01AD 212902                  LD      HL,MTBL
01B0 2004                    JR      NZ,+6
01B2 214102                  LD      HL,M#TBL
01B5 13                      INC     DE
01B6 CDDD01                  CALL    ONPU
01B9 38D7                    JR      C,MLD1
01BB CDC802                  CALL    RYTHM
01BE 3815                    JR      C,MLD5
01C0 CDAB02                  CALL    MLDST
01C3 41                      LD      B,C
01C4 18CC                    JR      MLD1
01C6 3E03            MLD2:   LD      A,+3
01C8 32A011                  LD      (OCTV),A
01CB 13                      INC     DE
01CC 18C4                    JR      MLD1
01CE 3E01            MLD3:   LD      A,1
01D0 18F6                    JR      MLD2+2
01D2 CDC802          MLD4:   CALL    RYTHM
01D5 F5              MLD5:   PUSH    AF
01D6 CDBE02                  CALL    MLDSP
01D9 F1                      POP     AF
01DA C39F06                  JP      RET3
01DD                 ;
01DD                 ;  ONPU TO RATIO CONV
01DD                 ;
01DD                 ;   EXIT (RATIO)=RATIO VALUE
01DD                 ;        C=ONTYO**TEMPO
01DD                 ;
01DD C5              ONPU:   PUSH    BC
01DE 0608                    LD      B,8
01E0 1A              ONP1:   LD      A,(DE)
01E1 BE                      CP      (HL)
01E2 2809                    JR      Z,ONP2
01E4 23                      INC     HL
01E5 23                      INC     HL
01E6 23                      INC     HL
01E7 10F8                    DJNZ    -6
01E9 37                      SCF
```

```
** Z80 ASSEMBLER SB-7201  <MZ-80A.MONITOR>  PAGE 06            09/04/81

01EA 13                      INC     DE
01EB C1                      POP     BC
01EC C9                      RET
01ED 23              ONP2:   INC     HL
01EE D5                      PUSH    DE
01EF 5E                      LD      E,(HL)
01F0 23                      INC     HL
01F1 56                      LD      D,(HL)
01F2 EB                      EX      DE,HL
01F3 7C                      LD      A,H
01F4 B7                      OR      A
01F5 2809                    JR      Z,+11
01F7 3AA011                  LD      A,(OCTV)
01FA 3D                      DEC     A
01FB 2803                    JR      Z,+5
01FD 29                      ADD     HL,HL
01FE 18FA                    JR      -4
0200 22A111                  LD      (RATIO),HL
0203 21A011                  LD      HL,OCTV
0206 3602                    LD      (HL),2
0208 2B                      DEC     HL
0209 D1                      POP     DE
020A 13                      INC     DE
020B 1A                      LD      A,(DE)
020C 47                      LD      B,A
020D E6F0                    AND     F0H             ; ONTYO ?
020F FE30                    CP      30H
0211 2803                    JR      Z,+5
0213 7E                      LD      A,(HL)          ; HL=ONTYO
0214 1805                    JR      +7
0216 13                      INC     DE
0217 78                      LD      A,B
0218 E60F                    AND     0FH
021A 77                      LD      (HL),A          ; HL=ONTYO
021B 215902                  LD      HL,OPTBL
021E 85                      ADD     A,L
021F 6F                      LD      L,A
0220 4E                      LD      C,(HL)
0221 3A9E11                  LD      A,(TEMPW)
0224 47                      LD      B,A
0225 AF                      XOR     A
0226 C3AB09                  JP      ONP3
0229                 ;
0229 43              MTBL:   DEFB    43H             ; C
022A 7707                    DEFW    0777H
022C 44                      DEFB    44H             ; D
022D A706                    DEFW    06A7H
022F 45                      DEFB    45H             ; E
0230 ED05                    DEFW    05EDH
0232 46                      DEFB    46H             ; F
0233 9805                    DEFW    0598H
0235 47                      DEFB    47H             ; G
0236 FC04                    DEFW    04FCH
0238 41                      DEFB    41H             ; A
0239 7104                    DEFW    0471H
023B 42                      DEFB    42H             ; B
023C F503                    DEFW    03F5H
023E 52                      DEFB    52H             ; R
023F 0000                    DEFW    0
0241 43              M#TBL:  DEFB    43H             ; #C
```

```
**  Z80 ASSEMBLER SB-7201  <MZ-80A.MONITOR>  PAGE 07          09/04/81

0242 0C07              DEFW  070CH
0244 44                DEFB  44H           ; *D
0245 4706              DEFW  0647H
0247 45                DEFB  45H           ; *E
0248 9805              DEFW  0598H
024A 46                DEFB  46H           ; *F
024B 4805              DEFW  0548H
024D 47                DEFB  47H           ; *G
024E B404              DEFW  04B4H
0250 41                DEFB  41H           ; *A
0251 3104              DEFW  0431H
0253 42                DEFB  42H           ; *B
0254 BB03              DEFW  03BBH
0256 52                DEFB  52H           ; *R
0257          ;
0257 0000              DEFW  0
0259 01       OPTBL:   DEFB  1
025A 02                DEFB  2
025B 03                DEFB  3
025C 04                DEFB  4
025D 06                DEFB  6
025E 08                DEFB  8
025F 0C                DEFB  0CH
0260 10                DEFB  10H
0261 18                DEFB  18H
0262 20                DEFB  20H
0263          ;
0263          ;
0263          ;   FLASHING DATA SAVE
0263          ;
0263 219211   ?SAVE:   LD    HL,FLSDT
0266 36EF              LD    (HL),EFH
0268 3A7011            LD    A,(KANAF)
026B B7                OR    A
026C 2802              JR    Z,KSLO
026E 36FF              LD    (HL),FFH
0270 7E       KSLO:    LD    A,(HL)
0271 F5                PUSH  AF
0272 CDB10F            CALL  ?PONT
0275 7E                LD    A,(HL)
0276 328E11            LD    (FLASH),A
0279 F1                POP   AF
027A 77                LD    (HL),A
027B AF                XOR   A
027C 2100E0            LD    HL,KEYPA
027F 77                LD    (HL),A
0280 2F                CPL
0281 77                LD    (HL),A
0282 C9                RET
0283          ;
0283          ;
0283          ;    COLOUMN MANAGEMENT POINTER
0283          ;        MGP.D=DECREMENT POINTER
0283          ;        MGP.I=INCREMENT POINTER
0283          ;
0283 F5       MGP.I:   PUSH  AF
0284 E5                PUSH  HL
0285 217C11            LD    HL,MGPNT
0288 7E                LD    A,(HL)
0289 3C                INC   A
```

```
**  Z80 ASSEMBLER SB-7201  <MZ-80A.MONITOR>  PAGE 08          09/04/81

028A FE33              CP    51            ; COLUMN=51 THEN 0
028C 2001              JR    NZ,MGPO
028E AF                XOR   A
028F E5       MGPO:    PUSH  HL
0290 6F                LD    L,A
0291 3A9111            LD    A,(SPAGE)
0294 B7                OR    A
0295 7D                LD    A,L
0296 E1                POP   HL
0297 2001              JR    NZ,+3
0299 77                LD    (HL),A
029A E1                POP   HL
029B F1                POP   AF
029C C9                RET
029D          ;
029D          ;
029D          ;
029D F5       MGP.D:   PUSH  AF
029E E5                PUSH  HL
029F 217C11            LD    HL,MGPNT
02A2 7E                LD    A,(HL)
02A3 3D                DEC   A
02A4 F28F02            JP    P,MGPO
02A7 3E32              LD    A,50
02A9 18E4              JR    MGPO
02AB          ;
02AB                   ORG   02ABH         ; MLDST
02AB          ;
02AB          ;   MELODY START & STOP
02AB          ;
02AB 2AA111   MLDST:   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 3E34     MLDSP:   LD    A,34H         ; MODE SET
02C0 3207E0            LD    (CONTF),A
02C3 AF                XOR   A
02C4 3208E0   MLDS1:   LD    (SUNDG),A
02C7 C9                RET
02C8          ;
02C8          ;   RHYTHM
02C8          ;
02C8          ;     B=COUNT
02C8          ;    EXIT CF=1 BREAK
02C8          ;         CF=0 OK
02C8          ;
02C8 2100E0   RYTHM:   LD    HL,KEYPA
02CB 36F0              LD    (HL),F0H
02CD 23                INC   HL
02CE 7E                LD    A,(HL)
02CF E681              AND   81H
```

```
02D1 2002                  JR     NZ,+4
02D3 37                    SCF
02D4 C9                    RET
02D5 3A08E0                LD     A,(TEMP)
02D8 0F                    RRCA
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              ;   BELL
02E5              ;
02E5              ?BEL:    ENT
02E5 D5                    PUSH   DE
02E6 11B10D                LD     DE,?BELD
02E9 F7                    RST    6              ; CALL MELDY
02EA D1                    POP    DE
02EB C9                    RET
02EC              ;
02EC              ;
02EC              ;   TEMPO
02EC              ;
02EC              ;      ACC=VALUE
02EC              ?TEMP:   ENT
02EC F5                    PUSH   AF
02ED C5                    PUSH   BC
02EE E60F                  AND    0FH
02F0 47                    LD     B,A
02F1 3E08                  LD     A,8
02F3 90                    SUB    B
02F4 329E11                LD     (TEMPW),A
02F7 C1                    POP    BC
02F8 F1                    POP    AF
02F9 C9                    RET
02FA              ;
02FA              ;   TIME SET
02FA              ;
02FA              ;      ACC=0 : AM
02FA              ;         =1 : PM
02FA              ;      DE=SEC, BINARY
02FA              ;
02FA              ?TMST:   ENT
02FA F3                    DI
02FB C5                    PUSH   BC
02FC D5                    PUSH   DE
02FD E5                    PUSH   HL
02FE 329B11                LD     (AMPM),A
0301 3EF0                  LD     A,F0H
0303 329C11                LD     (TIMFG),A
0306 21C0A8                LD     HL,A8C0H       ; 12H
0309 AF                    XOR    A
030A ED52                  SBC    HL,DE
030C E5                    PUSH   HL
030D 23                    INC    HL
030E EB                    EX     DE,HL
030F 2107E0                LD     HL,CONTF
0312 3674                  LD     (HL),74H
```



```
0314 3680                  LD     (HL),80H
0316 2B                    DEC    HL             ; CONT2
0317 73                    LD     (HL),E
0318 72                    LD     (HL),D
0319 2B                    DEC    HL             ; CONT1
031A 360A                  LD     (HL),0AH
031C 3600                  LD     (HL),0
031E 23                    INC    HL
031F 23                    INC    HL             ; CONTF
0320 3680                  LD     (HL),80H
0322 2B                    DEC    HL             ; CONT2
0323 4E           ?TMS1:   LD     C,(HL)
0324 7E                    LD     A,(HL)
0325 BA                    CP     D
0326 20FB                  JR     NZ,?TMS1
0328 79                    LD     A,C
0329 BB                    CP     E
032A 20F7                  JR     NZ,?TMS1
032C 2B                    DEC    HL
032D 00                    NOP
032E 00                    NOP
032F 00                    NOP
0330 360C                  LD     (HL),0CH
0332 367B                  LD     (HL),7BH
0334 23                    INC    HL
0335 D1                    POP    DE
0336 4E           ?TMS2:   LD     C,(HL)
0337 7E                    LD     A,(HL)
0338 BA                    CP     D
0339 20FB                  JR     NZ,?TMS2
033B 79                    LD     A,C
033C BB                    CP     E
033D 20F7                  JR     NZ,?TMS2
033F E1                    POP    HL
0340 D1                    POP    DE
0341 C1                    POP    BC
0342 FB                    EI
0343 C9                    RET
0344              ;
0344              ;;
0344              ;   TIME READ
0344              ;
0344              ;      EXIT  ACC=0 ;AM
0344              ;               =1 ;PM
0344              ;            DE=SEC. BINARY
0344              ;
0344              ?TMRD:   ENT
0344 E5                    PUSH   HL
0345 2107E0                LD     HL,CONTF
0348 3680                  LD     (HL),80H
034A 2B                    DEC    HL             ; CONT2
034B F3                    DI
034C 5E                    LD     E,(HL)
034D 56                    LD     D,(HL)
034E FB                    EI
034F 7B                    LD     A,E
0350 B2                    OR     D
0351 280E                  JR     Z,?TMR1
0353 AF                    XOR    A
0354 21C0A8                LD     HL,A8C0H
```

```
**  Z80 ASSEMBLER SB-7201  <MZ-80A.MONITOR>  PAGE 11            09/04/81

0357 ED52                 SBC    HL,DE
0359 3810                 JR     C,?TMR2
035B EB                   EX     DE,HL
035C 3A9B11               LD     A,(AMPM)
035F E1                   POP    HL
0360 C9                   RET
0361 11C0A8       ?TMR1:  LD     DE,A8C0H
0364 3A9B11               LD     A,(AMPM)
0367 EE01                 XOR    1
0369 E1                   POP    HL
036A C9                   RET
036B F3           ?TMR2:  DI
036C 2106E0               LD     HL,CONT2
036F 7E                   LD     A,(HL)
0370 2F                   CPL
0371 5F                   LD     E,A
0372 7E                   LD     A,(HL)
0373 2F                   CPL
0374 57                   LD     D,A
0375 FB                   EI
0376 13                   INC    DE
0377 18EB                 JR     ?TMR1+3
0379              ;
0379              ;
0379              ;    TIME INTERRUPT
0379              ;
0379              TIMIN:  ENT
0379 F5                   PUSH   AF
037A C5                   PUSH   BC
037B D5                   PUSH   DE
037C E5                   PUSH   HL
037D 219B11               LD     HL,AMPM
0380 7E                   LD     A,(HL)
0381 EE01                 XOR    1
0383 77                   LD     (HL),A
0384 2107E0               LD     HL,CONTF
0387 3680                 LD     (HL),80H
0389 2B                   DEC    HL                 ; CONT2
038A E5                   PUSH   HL
038B 5E                   LD     E,(HL)
038C 56                   LD     D,(HL)
038D 21C0A8               LD     HL,A8C0H
0390 19                   ADD    HL,DE
0391 2B                   DEC    HL
0392 2B                   DEC    HL
0393 EB                   EX     DE,HL
0394 E1                   POP    HL
0395 73                   LD     (HL),E
0396 72                   LD     (HL),D
0397 E1                   POP    HL
0398 D1                   POP    DE
0399 C1                   POP    BC
039A F1                   POP    AF
039B FB                   EI
039C C9                   RET
039D              ;
039D              ;    DSP03 PAGE MODE1
039D              ;
039D EB           .DSP03: EX     DE,HL
039E 3601                 LD     (HL),+1
```

```
**  Z80 ASSEMBLER SB-7201  <MZ-80A.MONITOR>  PAGE 12            09/04/81

03A0 23                   INC    HL
03A1 3600                 LD     (HL),0
03A3 C37B0E               JP     CURSR
03A6              ;
03A6              ;
03A6              ;
03A6              ;  MANAGMENT PAGE MODE1
03A6              ;
03A6              ;
03A6 3A7211       .MANG2: LD     A,(DSPXY+1)
03A9 85                   ADD    A,L
03AA 6F                   LD     L,A
03AB 7E                   LD     A,(HL)
03AC 23                   INC    HL
03AD CB16                 RL     (HL)
03AF B6                   OR     (HL)
03B0 CB1E                 RR     (HL)
03B2 0F                   RRCA
03B3 EB                   EX     DE,HL
03B4 2A7111               LD     HL,(DSPXY)
03B7 C9                   RET
03B8              ;
03B8              ;
03BA                      ORG    03BAH          ;PRNTHL   9
03BA              PRTHL:  ENT
03BA 7C                   LD     A,H
03BB CDC303               CALL   PRTHX
03BE 7D                   LD     A,L
03BF 1802                 JR     PRTHX
03C1              ;
03C3                      ORG    03C3H          ;PRTHX    23
03C3              ;
03C3              ;   (ACC) PRINT
03C3              ;
03C3              PRTHX:  ENT
03C3 F5                   PUSH   AF
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              ;    GETL RETURN
03D5              ;
03D5 D1           GETLL:  POP    DE
03D6 E1                   POP    HL
03D7 C1                   POP    BC
03D8 F1                   POP    AF
03D9 C9                   RET
03DA              ;
03DA              ;
03DA              ;
03DA                      ORG    03DAH          ;ASC      31
03DA              ;
03DA              ;  HEXA TO ASCII
```

```
**  Z80 ASSEMBLER SB-7201  <MZ-80A,MONITOR>  PAGE 13           09/04/81

03DA                 ;
03DA                 ASC:    ENT
03DA E60F                    AND   0FH
03DC FE0A                    CP    0AH
03DE 3802                    JR    C,NOADD
03E0 C607                    ADD   A,7
03E2 C630            NOADD:  ADD   A,30H
03E4 C9                      RET
03E5                 ;
03E5                 ;  ASCII TO HEXA
03E5                 ;
03E5 FE30            HEXJ:   CP    30H
03E7 D8                      RET   C
03E8 FE3A                    CP    3AH
03EA 3806                    JR    C,HEX1
03EC D607                    SUB   7
03EE FE40                    CP    40H
03F0 3003                    JR    NC,HEX2
03F2 E60F            HEX1:   AND   0FH
03F4 C9                      RET
03F5 37              HEX2:   SCF
03F6 C9                      RET
03F7                 ;
03F7                 ;
03F9                         ORG   03F9H           ;HEX    23
03F9                 HEX:    ENT
03F9 18EA                    JR    HEXJ
03FB                 ;
03FB                 ;;
03FB                 ;  @DPCT PUCH
03FB                 ;
03FB 2A7111          HOME:   LD    HL,(DSPXY)
03FE 3A7C11                  LD    A,(MGPNT)
0401 94                      SUB   H
0402 3002                    JR    NC,HOM1
0404 C632                    ADD   A,50
0406 327C11          HOM1:   LD    (MGPNT),A
0409 210000          HOM0:   LD    HL,0
040C C3690E                  JP    CURS3
040F                 ;
040F                 ;
0410                         ORG   0410H           ;HLHEX   15
0410                 ;
0410                 ;
0410                 ;  4 ASCII TO (HL)
0410                 ;
0410                 ;  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
```

```
**  Z80 ASSEMBLER SB-7201  <MZ-80A,MONITOR>  PAGE 14           09/04/81

041F                 ;
041F                         ORG   041FH           ;2HEX    482
041F                 ;
041F                 ;
041F                 ;  2 ASCII TO (ACC)
041F                 ;
041F                 ;  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
0441 018000                  LD    BC,80H
0444 CD1A07          WRI1:   CALL  CKSUM
0447 CDA306                  CALL  MOTOR
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
0458 DF                      RST   3               ; CALL MSGX
0459 11F110                  LD    DE,NAME
045C DF                      RST   3               ; CALL MSGX
045D D1                      POP   DE
045E CD7A07          WRI2:   CALL  GAP
0461 CD8504                  CALL  WTAPE
0464 C35205          WRI3:   JP    RET2
0467                 ;
0467 57              MSG#7:  DEFM  'W'             ; WRITING
```

```
**  Z80 ASSEMBLER SB-7201  <MZ-80A.MONITOR>  PAGE 15            09/04/81

0468 9DA6                   DEFW  A69DH
046A 96A6                   DEFW  A696H
046C B097                   DEFW  97B0H
046E 200D                   DEFW  0D20H
0470               ;;
0470               ;   WRITE DATA
0470               ;
0470               ;    EXIT CF=0 : OK
0470               ;          =1 : BREAK
0470               ;
0470               ?WRD:    ENT
0470 F3                     DI
0471 D5                     PUSH  DE
0472 C5                     PUSH  BC
0473 E5                     PUSH  HL
0474 16D7                   LD    D,D7H          ;'W'
0476 1E53                   LD    E,53H          ; 'S'
0478 ED4B0211               LD    BC,(SIZE)
047C 2A0411                 LD    HL,(DTADR)
047F 78                     LD    A,B
0480 B1                     OR    C
0481 2848                   JR    Z,RET1
0483 18BF                   JR    WRI1
0485               ;
0485               ;;
0485               ;   TAPE WRITE
0485               ;
0485               ;     BC=BYTE SIZE
0485               ;     HL=DATA LOW ADR.
0485               ;
0485               ;    EXIT  CF=0 : OK
0485               ;           =1 : BREAK
0485               ;
0485 D5            WTAPE:   PUSH  DE
0486 C5                     PUSH  BC
0487 E5                     PUSH  HL
0488 1602                   LD    D,2
048A 3EF0                   LD    A,F0H
048C 3200E0                 LD    (KEYPA),A
048F 7E            WTAP1:   LD    A,(HL)
0490 CD6707                 CALL  WBYTE
0493 3A01E0                 LD    A,(KEYPB)
0496 E681                   AND   81H            ; SHIFT & BREAK
0498 C29E04                 JP    NZ,WTAP2
049B 37                     SCF
049C 182D                   JR    WTAP3
049E 23            WTAP2:   INC   HL
049F 0B                     DEC   BC
04A0 78                     LD    A,B
04A1 B1                     OR    C
04A2 C28F04                 JP    NZ,WTAP1
04A5 2A9711                 LD    HL,(SUMDT)
04A8 7C                     LD    A,H
04A9 CD6707                 CALL  WBYTE
04AC 7D                     LD    A,L
04AD CD6707                 CALL  WBYTE
04B0 CD570D                 CALL  LONG
04B3 15                     DEC   D
04B4 C2BB04                 JP    NZ,+7
04B7 B7                     OR    A
```

```
**  Z80 ASSEMBLER SB-7201  <MZ-80A.MONITOR>  PAGE 16            09/04/81

04B8 C3CB04                 JP    WTAP3
04BB 0600                   LD    B,0
04BD CD3E0D                 CALL  SHORT
04C0 05                     DEC   B
04C1 C2BD04                 JP    NZ,-4
04C4 E1                     POP   HL
04C5 C1                     POP   BC
04C6 C5                     PUSH  BC
04C7 E5                     PUSH  HL
04C8 C38F04                 JP    WTAP1
04CB               WTAP3:
04CB E1            RET1:    POP   HL
04CC C1                     POP   BC
04CD D1                     POP   DE
04CE C9                     RET
04CF               ;
04CF               ;;
04CF               ;   READ INFORMATION
04CF               ;
04CF               ;    EXIT ACC=0 : OK CF=0
04CF               ;            =1 : ER CF=1
04CF               ;            =2 : BREAK CF=1
04CF               ;
04CF               ?RDI:    ENT
04CF F3                     DI
04D0 D5                     PUSH  DE
04D1 C5                     PUSH  BC
04D2 E5                     PUSH  HL
04D3 16D2                   LD    D,D2H          ; 'R'
04D5 1ECC                   LD    E,CCH          ; 'L'
04D7 018000                 LD    BC,80H
04DA 21F010                 LD    HL,IBUFE
04DD CDA306        RD1:     CALL  MOTOR
04E0 DA7005                 JP    C,RTP6
04E3 CD5806                 CALL  TMARK
04E6 DA7005                 JP    C,RTP6
04E9 CD0505                 CALL  RTAPE
04EC C35205                 JP    RTP4
04EF               ;;
04EF               ;   READ DATA
04EF               ;
04EF               ;    EXIT SAME UP
04EF               ;
04EF               ?RDD:    ENT
04EF F3                     DI
04F0 D5                     PUSH  DE
04F1 C5                     PUSH  BC
04F2 E5                     PUSH  HL
04F3 16D2                   LD    D,D2H          ; 'R'
04F5 1E53                   LD    E,53H          ; 'S'
04F7 ED4B0211               LD    BC,(SIZE)
04FB 2A0411                 LD    HL,(DTADR)
04FE 78                     LD    A,B
04FF B1                     OR    C
0500 CA5205                 JP    Z,RTP4
0503 18D8                   JR    RD1
0505               ;
0505               ;;
0505               ;   READ TAPE
0505               ;
```

```
**  Z80 ASSEMBLER SB-7201  <MZ-80A.MONITOR>  PAGE 17          09/04/81

0505                  ;    BC=SIZE
0505                  ;    DE=LOAD ADR.
0505                  ;
0505                  ;   EXIT  ACC=0 ; OK CF=0
0505                  ;            =1 ; ER   =1
0505                  ;            =2 ; BREAK=1
0505                  ;
0505 D5               RTAPE:  PUSH  DE
0506 C5                       PUSH  BC
0507 E5                       PUSH  HL
0508 2602                     LD    H,2
050A 0101E0           RTP1:   LD    BC,KEYPB
050D 1102E0                   LD    DE,CSTR
0510 CD0106           RTP2:   CALL  EDGE
0513 DA7005                   JP    C,RTP6
0516 CDA209                   CALL  DLY3              ; CALL DLY2*3
0519 1A                       LD    A,(DE)
051A E620                     AND   20H
051C CA1005                   JP    Z,RTP2
051F 54                       LD    D,H
0520 210000                   LD    HL,0
0523 229711                   LD    (SUMDT),HL
0526 E1                       POP   HL
0527 C1                       POP   BC
0528 C5                       PUSH  BC
0529 E5                       PUSH  HL
052A CD2406           RTP3:   CALL  RBYTE
052D DA7005                   JP    C,RTP6
0530 77                       LD    (HL),A
0531 23                       INC   HL
0532 0B                       DEC   BC
0533 78                       LD    A,B
0534 B1                       OR    C
0535 C22A05                   JP    NZ,RTP3
0538 2A9711                   LD    HL,(SUMDT)
053B CD2406                   CALL  RBYTE
053E DA7005                   JP    C,RTP6
0541 5F                       LD    E,A
0542 CD2406                   CALL  RBYTE
0545 DA7005                   JP    C,RTP6
0548 BD                       CP    L
0549 C26305                   JP    NZ,RTP5
054C 7B                       LD    A,E
054D BC                       CP    H
054E C26305                   JP    NZ,RTP5
0551 AF               RTP8:   XOR   A
0552                  RTP4:
0552 E1               RET2:   POP   HL
0553 C1                       POP   BC
0554 D1                       POP   DE
0555 CD0007                   CALL  MSTOP
0558 F5                       PUSH  AF
0559 3A9C11                   LD    A,(TIMFG)
055C FEF0                     CP    F0H
055E 2001                     JR    NZ,+3
0560 FB                       EI
0561 F1                       POP   AF
0562 C9                       RET
0563                  ;
0563 15               RTP5:   DEC   D
```

```
**  Z80 ASSEMBLER SB-7201  <MZ-80A.MONITOR>  PAGE 18          09/04/81

0564 2806                     JR    Z,RTP7
0566 62                       LD    H,D
0567 CDE20F                   CALL  GAPCK
056A 189E                     JR    RTP1
056C 3E01             RTP7:   LD    A,1
056E 1802                     JR    RTP9
0570 3E02             RTP6:   LD    A,2
0572 37               RTP9:   SCF
0573 18DD                     JR    RTP4
0575                  ;
0575                  ;;
0575                  ;   VERIFY
0575                  ;
0575                  ;    EXIT ACC =0 ; OK CF=0
0575                  ;             =1 ; ER CF=1
0575                  ;             =2 ; BREAK CF=1
0575                  ;
0575                  ?VRFY:  ENT
0575 F3                       DI
0576 D5                       PUSH  DE
0577 C5                       PUSH  BC
0578 E5                       PUSH  HL
0579 ED4B0211                 LD    BC,(SIZE)
057D 2A0411                   LD    HL,(DTADR)
0580 16D2                     LD    D,D2H             ; 'R'
0582 1E53                     LD    E,53H             ; 'S'
0584 78                       LD    A,B
0585 B1                       OR    C
0586 28CA                     JR    Z,RTP4
0588 CD1A07                   CALL  CKSUM
058B CDA306                   CALL  MOTOR
058E 38E0                     JR    C,RTP6
0590 CD5806                   CALL  TMARK
0593 DA7005                   JP    C,RTP6
0596 CD9B05                   CALL  TVRFY
0599 18B7                     JR    RTP4
059B                  ;
059B                  ;;
059B                  ;   DATA VERIFY
059B                  ;
059B                  ;     BC=SIZE
059B                  ;     HL=DATA LOW ADR
059B                  ;     CSMDT=CHECK SUM
059B                  ;    EXIT ACC=0 ; OK CF=0
059B                  ;            =1 ; ER   =1
059B                  ;            =2 ; BREAK=1
059B                  ;
059B D5               TVRFY:  PUSH  DE
059C C5                       PUSH  BC
059D E5                       PUSH  HL
059E 2602                     LD    H,2
05A0 0101E0           TVF1:   LD    BC,KEYPB
05A3 1102E0                   LD    DE,CSTR
05A6 CD0106           TVF2:   CALL  EDGE
05A9 DA7005                   JP    C,RTP6
05AC CDA209                   CALL  DLY3              ; CALL DLY2*3
05AF 1A                       LD    A,(DE)
05B0 E620                     AND   20H
05B2 CAA605                   JP    Z,TVF2
```

```
05B5 54                     LD    D,H
05B6 E1                     POP   HL
05B7 C1                     POP   BC
05B8 C5                     PUSH  BC
05B9 E5                     PUSH  HL
05BA CD2406        TVF3:    CALL  RBYTE
05BD DA7005                 JP    C,RTP6
05C0 BE                     CP    (HL)
05C1 C26C05                 JP    NZ,RTP7
05C4 23                     INC   HL
05C5 0B                     DEC   BC
05C6 78                     LD    A,B
05C7 B1                     OR    C
05C8 C2BA05                 JP    NZ,TVF3
05CB 2A9911                 LD    HL,(CSMDT)
05CE CD2406                 CALL  RBYTE
05D1 BC                     CP    H
05D2 2098                   JR    NZ,RTP7
05D4 CD2406                 CALL  RBYTE
05D7 BD                     CP    L
05D8 2092                   JR    NZ,RTP7
05DA 15                     DEC   D
05DB CA5105                 JP    Z,RTP8
05DE 62                     LD    H,D
05DF 18BF                   JR    TVF1
05E1               :
05E1               :    PRINT '00'
05E1               :
05E1 11FC09        OETLD:   LD    DE,OOMSG
05E4 DF                     RST   3
05E5 C30708                 JP    AUTO2
05E8               :
05E8               :
05E8               :
05E8               :    ROLL UP
05E8               :
05E8 217A11        ROLUP:   LD    HL,PBIAS
05EB 3A7F11                 LD    A,(ROLEND)
05EE BE                     CP    (HL)
05EF CAE50E                 JP    Z,?RSTR
05F2 C3A90F                 JP    ROLU1
05F5               :
05F5               :
05F5               :  FLASHING DATA LOAD
05F5               :
05F5 F5            ?LOAD:   PUSH  AF
05F6 3A8E11                 LD    A,(FLASH)
05F9 CDB10F                 CALL  ?PONT
05FC 77                     LD    (HL),A
05FD F1                     POP   AF
05FE C9                     RET
05FF               :
05FF               :
05FF               :
0601                        ORG   0601H           :EDGE   35
0601               :
0601               ::
0601               :  EDGE
0601               :
0601               :  BC=KEYPB
```



```
0601               :   DE=CSTR
0601               :  EXIT CF=0 : EDGE
0601               :          =1 : BREAK
0601               :
0601 3EF0          EDGE:    LD    A,F0H
0603 3200E0                 LD    (KEYPA),A
0606 00                     NOP
0607 0A            EDG1:    LD    A,(BC)
0608 E681                   AND   81H             : SHIFT & BREAK
060A C20F06                 JP    NZ,+5
060D 37                     SCF
060E C9                     RET
060F 1A                     LD    A,(DE)
0610 E620                   AND   20H
0612 C20706                 JP    NZ,EDG1
0615 0A            EDG2:    LD    A,(BC)
0616 E681                   AND   81H
0618 C21D06                 JP    NZ,+5
061B 37                     SCF
061C C9                     RET
061D 1A                     LD    A,(DE)
061E E620                   AND   20H
0620 CA1506                 JP    Z,EDG2
0623 C9                     RET
0624               :
0624               :
0624                        ORG   0624H           :RBYTE  220
0624               :
0624               ::
0624               :   1 BYTE READ
0624               :
0624               :     EXIT  SUMDT=STORE
0624               :      CF=1 : BREAK
0624               :      CF=0 : DATA=ACC
0624               :
0624 C5            RBYTE:   PUSH  BC
0625 D5                     PUSH  DE
0626 E5                     PUSH  HL
0627 210008                 LD    HL,0800H
062A 0101E0                 LD    BC,KEYPB
062D 1102E0                 LD    DE,CSTR
0630 CD0106        RBY1:    CALL  EDGE
0633 DA5406                 JP    C,RBY3
0636 CDA209                 CALL  DLY3            : CALL DLY2*3
0639 1A                     LD    A,(DE)
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 7D            RBY2:    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 E1              RBY3:   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 CDE20F          TMARK:  CALL    GAPCK
065B C5                      PUSH    BC
065C D5                      PUSH    DE
065D E5                      PUSH    HL
065E 212828                  LD      HL,2828H
0661 7B                      LD      A,E
0662 FECC                    CP      CCH             ;'L'
0664 CA6A06                  JP      Z,+6
0667 211414                  LD      HL,1414H
066A 229511                  LD      (TMCNT),HL
066D 0101E0                  LD      BC,KEYPB
0670 1102E0                  LD      DE,CSTR
0673 2A9511          TM1:    LD      HL,(TMCNT)
0676 CD0106          TM2:    CALL    EDGE
0679 DA9F06                  JP      C,TM4
067C CDA209                  CALL    DLY3            ; CALL DLY2*3
067F 1A                      LD      A,(DE)
0680 E620                    AND     20H
0682 CA7306                  JP      Z,TM1
0685 25                      DEC     H
0686 C27606                  JP      NZ,TM2
0689 CD0106          TM3:    CALL    EDGE
068C DA9F06                  JP      C,TM4
068F CDA209                  CALL    DLY3            ; CALL DLY2*3
0692 1A                      LD      A,(DE)
0693 E620                    AND     20H
0695 C27306                  JP      NZ,TM1
0698 2D                      DEC     L
0699 C28906                  JP      NZ,TM3
069C CD0106                  CALL    EDGE
069F                 RET3:
069F E1              TM4:    POP     HL
06A0 D1                      POP     DE
06A1 C1                      POP     BC
06A2 C9                      RET
06A3                 ;
06A3                 ;;
06A3                 ;   MOTOR ON
06A3                 ;
06A3                 ;     D=@W@ :WRITE
06A3                 ;      =@R@ : READ
06A3                 ;   EXIT CF=0 :OK
06A3                 ;           =1 :BREAK
06A3 C5              MOTOR:  PUSH    BC
06A4 D5                      PUSH    DE
06A5 E5                      PUSH    HL
06A6 060A                    LD      B,10
```



```
06A8 3A02E0          MOT1:   LD      A,(CSTR)
06AB E610                    AND     10H
06AD 280A                    JR      Z,MOT4
06AF 06A6            MOT2:   LD      B,A6H           ; 2 SEC DLAY
06B1 CDA70D                  CALL    DLY12
06B4 10FB                    DJNZ    -3
06B6 AF                      XOR     A
06B7 18E6            MOT7:   JR      RET3
06B9 3E06            MOT4:   LD      A,06H
06BB 2103E0                  LD      HL,CSTPT
06BE 77                      LD      (HL),A
06BF 3C                      INC     A
06C0 77                      LD      (HL),A
06C1 10E5                    DJNZ    MOT1
06C3 CD0900                  CALL    NL
06C6 7A                      LD      A,D
06C7 FED7                    CP      D7H             ; 'W'
06C9 2805                    JR      Z,MOT8
06CB 119E0D                  LD      DE,MSG#1
06CE 1807                    JR      MOT9
06D0 11F406          MOT8:   LD      DE,MSG#3
06D3 DF                      RST     3               ; CALL MSGX
06D4 11A00D                  LD      DE,MSG#2
06D7 DF              MOT9:   RST     3               ; CALL MSGX
06D8 3A02E0          MOT5:   LD      A,(CSTR)
06DB E610                    AND     10H
06DD 20D0                    JR      NZ,MOT2
06DF CD110D                  CALL    ?BRK
06E2 20F4                    JR      NZ,MOT5
06E4 37                      SCF
06E5 18D0                    JR      MOT7
06E7                 ;
06E7                 ;
06E7                 ;
06E7                 ;  ?KEY  GRAPH KEY INPUT
06E7                 ;   ;
06E7 06C9            #GRP:   LD      B,C9H
06E9 3A7011                  LD      A,(KANAF)
06EC B7                      OR      A
06ED 2001                    JR      NZ,+3
06EF 04                      INC     B
06F0 78                      LD      A,B
06F1 C3D608                  JP      ?KY1
06F4                 ;
06F4                 ;
06F4 7F20            MSG#3:  DEFW    207FH           ; PRESS RECORD
06F6 5245434F                DEFM    'RECORD.'
06FA 52442E
06FD 0D                      DEFB    0DH
06FE                 ;
06FE                 ;
0700                         ORG     0700H           ;MSTOP   89
0700                 ;
0700                 ;;
0700                 ;  MOTOR STOP
0700                 ;
0700 F5              MSTOP:  PUSH    AF
0701 C5                      PUSH    BC
0702 D5                      PUSH    DE
0703 060A                    LD      B,10
```

```
** Z80 ASSEMBLER SB-7201 <MZ-80A.MONITOR> PAGE 23          09/04/81

0705 3A02E0    MST1:  LD    A,(CSTR)
0708 E610             AND   10H
070A 280B             JR    Z,MST3
070C 3E06      MST2:  LD    A,06H
070E 3203E0           LD    (CSTPT),A
0711 3C               INC   A
0712 3203E0           LD    (CSTPT),A
0715 10EE             DJNZ  MST1
0717 C3E60E    MST3:  JP    ?RSTR1
071A                  ;
071A                  ;
071A                  ;
071A                  ;
071A                  ;   CHECK SUM
071A                  ;
071A                  ;     BC=SIZE
071A                  ;     HL=DATA ADR.
071A                  ;    EXIT SUMDT=STORE
071A                  ;         CSMDT=STORE
071A                  ;
071A C5        CKSUM: PUSH  BC
071B D5               PUSH  DE
071C E5               PUSH  HL
071D 110000           LD    DE,0
0720 78        CKS1:  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 7E        CKS2:  LD    A,(HL)
0730 C5               PUSH  BC
0731 0608             LD    B,+8
0733 07        CKS3:  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                  ;   SWEP PART2
073E                  ;
073E 07        SWEP4: RLCA
073F 07               RLCA
0740 07               RLCA
0741 4F               LD    C,A
0742 7B               LD    A,E
0743 25               DEC   H
0744 0F               RRCA
0745 30FC             JR    NC,-2
0747 7C               LD    A,H
0748 81               ADD   A,C
0749 4F               LD    C,A
074A C3740A           JP    SWEP01
```

```
** Z80 ASSEMBLER SB-7201 <MZ-80A.MONITOR> PAGE 24          09/04/81

074D                  ;
074D                  ;   MODE SET OF KEYPORT
074D                  ;
074D           ?MODE: ENT
074D 2103E0           LD    HL,KEYPF
0750 368A             LD    (HL),8AH           ; 10001010
0752 3607             LD    (HL),07H           ; PC3=1
0754 3605             LD    (HL),05H           ; PC2=1
0756 3601      VGOFF: LD    (HL),01H           ; PC0 SET
0758 C9               RET
0759                  ;
0759                  ;
0759                  ;
0759                  ;
0759                  ORG   0759H              ;DLY1  7
0759                  ;;
0759                  ;  107 MICRO SEC DELY
0759                  ;
0759 3E0E      DLY1:  LD    A,14
075B 3D               DEC   A
075C C25B07           JP    NZ,-1
075F C9               RET
0760                  ;
0760                  ORG   0760H              ;DLY2  240
0760                  ;
0760 3E0D      DLY2:  LD    A,13
0762 3D               DEC   A
0763 C26207           JP    NZ,-1
0766 C9               RET
0767                  ;
0767                  ;
0767                  ;;
0767                  ;  1 BYTE WRITE
0767                  ;
0767 C5        WBYTE: PUSH  BC
0768 0608             LD    B,+8
076A CD570D           CALL  LONG
076D 07        WBY1:  RLCA
076E DC570D           CALL  C,LONG
0771 D43E0D           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 C5        GAP:   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
```

```
** Z80 ASSEMBLER SB-7201  <MZ-80A.MONITOR>  PAGE 25          09/04/81

078E CD3E0D        GAP1:   CALL  SHORT
0791 0B                    DEC   BC
0792 78                    LD    A,B
0793 B1                    OR    C
0794 20F8                  JR    NZ,-6
0796 CD570D        GAP2:   CALL  LONG
0799 15                    DEC   D
079A 20FA                  JR    NZ,-4
079C CD3E0D        GAP3:   CALL  SHORT
079F 1D                    DEC   E
07A0 20FA                  JR    NZ,-4
07A2 CD570D                CALL  LONG
07A5 D1                    POP   DE
07A6 C1                    POP   BC
07A7 C9                    RET
07A8               ;
07A8   P           KEYPA:  EQU   E000H
07A8   P           KEYPB:  EQU   E001H
07A8   P           KEYPC:  EQU   E002H
07A8   P           KEYPF:  EQU   E003H
07A8   P           CSTR:   EQU   E002H
07A8   P           CSTPT:  EQU   E003H
07A8   P           CONT0:  EQU   E004H
07A8   P           CONT1:  EQU   E005H
07A8   P           CONT2:  EQU   E006H
07A8   P           CONTF:  EQU   E007H
07A8   P           SUNDG:  EQU   E008H
07A8   P           TEMP:   EQU   E008H
07A8                       SKP   H
```

```
** Z80 ASSEMBLER SB-7201  <MZ-80A.MONITOR>  PAGE 26          09/04/81

07A8               ;
07A8               ;
07A8               ;   GET 1 LINE STATEMENT  *
07A8               ;
07A8               ;         DE = DATA STORE LOW ADR.
07A8               ;              (END =CR )
07A8               ;           ORG 07E6H
07A8               ;
07A8               ?GETL:  ENT
07A8 F5                    PUSH  AF
07A9 C5                    PUSH  BC
07AA E5                    PUSH  HL
07AB D5                    PUSH  DE
07AC CD6302        GETL0:  CALL  ?SAVE
07AF CDCA08                CALL  ?KEY
07B2 FECB                  CP    CBH
07B4 28F9                  JR    Z,-5
07B6 CDCA08        GETL1:  CALL  ?KEY
07B9 CDFF09                CALL  ?FLAS
07BC 28F8                  JR    Z,-6
07BE F5                    PUSH  AF
07BF AF                    XOR   A
07C0 329311                LD    (STRGF),A
07C3 F1                    POP   AF
07C4 47            AUTO3:  LD    B,A
07C5 CDF505                CALL  ?LOAD
07C8 3A9D11                LD    A,(SWRK)        ; BELL WORK
07CB B7                    OR    A
07CC CCE502                CALL  Z,?BEL
07CF 78                    LD    A,B
07D0 FEE7                  CP    E7H             ; 00
07D2 CAE105                JP    Z,GETLD
07D5 FEE6                  CP    E6H             ; ↑[ PAGE MODE K
07D7 2868                  JR    Z,CHGPK
07D9 FEEE                  CP    EEH             ; ↑] PAGE MODE A
07DB 2861                  JR    Z,CHGPA
07DD FEE5                  CP    E5H             ; ↑Z CODE=5AH
07DF 2876                  JR    Z,DMT
07E1 FEE0                  CP    E0H
07E3 CA8B08                JP    Z,LOCK
07E6 30CE                  JR    NC,GETL1
07E8 E6F0                  AND   F0H
07EA FEC0                  CP    C0H
07EC 2015                  JR    NZ,GETL2
07EE 78                    LD    A,B
07EF FECD                  CP    CDH             ; CR
07F1 2868                  JR    Z,GETL3
07F3 FECB                  CP    CBH             ; BREAK
07F5 CA4E08                JP    Z,GETLC
07F8 FEC7                  CP    C7H             ; CRT EDITION
07FA 303D                  JR    NC,GETL5
07FC 3A7011                LD    A,(KANAF)
07FF B7                    OR    A
0800 78                    LD    A,B
0801 2836                  JR    Z,GETL5
0803 78            GETL2:  LD    A,B
0804 CDB50D                CALL  ?DSP
0807 CD6302        AUTO2:  CALL  ?SAVE
080A 3A9311                LD    A,(STRGF)
080D B7                    OR    A
```

```
**  Z80 ASSEMBLER SB-7201  <MZ-80A,MONITOR>  PAGE 27          09/04/81

080E 2014                    JR     NZ,AUTO5
0810 1E14           AUTOL:   LD     E,20
0812 CDCA08                  CALL   ?KEY
0815 20AD                    JR     NZ,AUTO3
0817 CDF109                  CALL   AUTCK
081A 389A           GETL11:  JR     C,GETL1
081C 1D                      DEC    E
081D 20F3                    JR     NZ,AUTOL+2
081F 3E01                    LD     A,1
0821 329311                  LD     (STRGF),A
0824 CDA70D         AUTO5:   CALL   DLY12
0827 CDA70D                  CALL   DLY12
082A CDCA08                  CALL   ?KEY
082D CDFF09                  CALL   ?FLAS
0830 208C                    JR     NZ,AUTO3-6        ; INPUT NEWKEY
0832 CDF109                  CALL   AUTCK
0835 38E3                    JR     C,GETL11
0837 188C                    JR     AUTO3+1
0839 CDDC0D         GETL5:   CALL   ?DPCT
083C 18C9                    JR     AUTO2
083E                ;
083E                ;
083E                ;  CHANGE PAGE MODE
083E                ;
083E AF             CHGPA:   XOR    A
083F 1802                    JR     +4
0841 3EFF           CHGPK:   LD     A,FFH
0843 329111                  LD     (SPAGE),A
0846 3EC6                    LD     A,C6H             ; CLRS CODE
0848 CDDC0D                  CALL   ?DPCT
084B C3AC07         CHOP1:   JP     GETL0
084E                ;
084E                ;    BREAK IN
084E                ;
084E E1             GETLC:   POP    HL
084F E5                      PUSH   HL
0850 361B                    LD     (HL),1BH
0852 23                      INC    HL
0853 360D                    LD     (HL),0DH
0855 1829                    JR     GETLR
0857                ;
0857                ;
0857                ;
0857 065A           DMT:     LD     B,5AH             ; *
0859 18A8                    JR     GETL2
085B                ;
085B                         ORG    085BH             ;GETL3     122
085B                ;
085B CD2B0A         GETL3:   CALL   .MANG             ; CR
085E 0628                    LD     B,40              ; 1LINE
0860 3024                    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                      LD     A,(HL)
086D CDCE0B                  CALL   ?DACN
0870 12                      LD     (DE),A
```

```
**  Z80 ASSEMBLER SB-7201  <MZ-80A,MONITOR>  PAGE 28          09/04/81

0871 23                      INC    HL
0872 CB9C                    RES    3,H
0874 13                      INC    DE
0875 10F5                    DJNZ   -9
0877 EB                      EX     DE,HL
0878 360D                    LD     (HL),0DH
087A 2B                      DEC    HL
087B 7E                      LD     A,(HL)
087C FE20                    CP     20H               ; SPACE
087E 28F8                    JR     Z,-6
0880 CD8009         GETLR:   CALL   ?LTNL
0883 C3D503                  JP     GETLL
0886                ;
0886 0F             GETLA:   RRCA
0887 30DC                    JR     NC,GETL6
0889 18D8                    JR     GETLB
088B                ;
088B                ;
088B                ;
088B                ;  SHIFT LOCK
088B                ;
088B 218F11         LOCK:    LD     HL,SFTLK
088E 7E                      LD     A,(HL)
088F 2F                      CPL
0890 77                      LD     (HL),A
0891 18B8                    JR     CHOP1
0893                ;
0893                ;
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)
0897 FE0D                    CP     0DH               ; CR
0899 280C                    JR     Z,MSGX2
089B CD9509                  CALL   ?PRNT
089E 13                      INC    DE
089F 18F5                    JR     MSG1
08A1                ;
08A1                ?MSGX:   ENT                      ; ALL PRINT MSG
08A1 F5                      PUSH   AF
08A2 C5                      PUSH   BC
08A3 D5                      PUSH   DE
08A4 1A             MSGX1:   LD     A,(DE)
08A5 FE0D                    CP     0DH
08A7 CAE60E         MSGX2:   JP     Z,?RSTR1
08AA CDB90B                  CALL   ?ADCN
08AD CD6C09                  CALL   PRNT3
08B0 13                      INC    DE
08B1 18F1                    JR     MSGX1
08B3                ;
08B3                ;
08B3                ;  GET 1KEY
08B3                ;  EXIT  ACC=ASCII
```

```
** Z80 ASSEMBLER SB-7201 <MZ-80A,MONITOR> PAGE 29          09/04/81

0883                    ;   IF NO KEY   ACC=00
0883                    ;
0883                    ?GET:   ENT
0883 C5                         PUSH  BC
0884 E5                         PUSH  HL
0885 0609                       LD    B,9
0887 216511                     LD    HL,SWPW+1
088A CDDB0F                     CALL  ?CLRFF
088D E1                         POP   HL
088E C1                         POP   BC
088F CDCA08                     CALL  ?KEY
08C2 D6F0                       SUB   F0H
08C4 C8                         RET   Z
08C5 C6F0                       ADD   A,F0H
08C7 C3CE0B                     JP    ?DACN
08CA                    ;
08CA                    ;
08CA                            ORG   08CAH          ;?KEY   124
08CA                    ;
08CA                    ;   1KEY INPUT
08CA                    ;   IN       B = KEY MODE(SHIFT,CTRL,GRAPH)
08CA                    ;            C=KEY DATA (COLUMN & ROW)
08CA                    ;   EXIT   ACC=DISPLAY CODE
08CA                    ;          IF NO KEY   ACC=F0H
08CA C5                 ?KEY:   PUSH  BC
08CB D5                         PUSH  DE
08CC E5                         PUSH  HL
08CD CD500A             ?KY8:   CALL  ?SWEP
08D0 78                         LD    A,B
08D1 07                         RLCA
08D2 381E                       JR    C,?KY2
08D4 3EF0                       LD    A,F0H
08D6 5F                 ?KY1:   LD    E,A
08D7 CDF109                     CALL  AUTCK
08DA 3A6E11                     LD    A,(KDATW)
08DD B7                         OR    A
08DE 280A                       JR    Z,?KY11
08E0 CDA70D                     CALL  DLY12          ; 12 M SEC DELAY
08E3 CD500A                     CALL  ?SWEP
08E6 78                         LD    A,B
08E7 07                         RLCA
08E8 3808                       JR    C,?KY2
08EA 7B                 ?KY11:  LD    A,E
08EB FEF0                       CP    F0H
08ED 2050                       JR    NZ,?KY9
08EF C39F06             ?KY10:  JP    RET3
08F2 07                 ?KY2:   RLCA
08F3 07                         RLCA
08F4 07                         RLCA
08F5 DAE706                     JP    C,#GRP
08F8 07                         RLCA
08F9 DAC50B                     JP    C,#BRK
08FC 2600                       LD    H,0
08FE 69                         LD    L,C            ; HL=BC
08FF 79                         LD    A,C            ; TENKEY CHECK
0900 FE38                       CP    38H
0902 3026                       JR    NC,?KY6        ; TEN KEY THEN JUMP
0904 3A7011                     LD    A,(KANAF)
0907 B7                         OR    A
0908 78                         LD    A,B
```

```
** Z80 ASSEMBLER SB-7201 <MZ-80A,MONITOR> PAGE 30          09/04/81

0909 07                         RLCA
090A 2023                       JR    NZ,?KY4
090C 47                         LD    B,A
090D 3A8F11                     LD    A,(SFTLK)
0910 B7                         OR    A
0911 78                         LD    A,B
0912 2803                       JR    Z,+5
0914 17                         RLA
0915 3F                         CCF
0916 1F                         RRA
0917 17                         RLA
0918 17                         RLA
0919 3007                       JR    NC,?KY3
091B 11DA0C                     LD    DE,KTBLC       ; CONTROL ON
091E 19                 ?KY5:   ADD   HL,DE
091F 7E                         LD    A,(HL)         ; DISPLAY CODE
0920 18B4                       JR    ?KY1
0922 1F                 ?KY3:   RRA                  ; CTRL OFF
0923 3005                       JR    NC,?KY6
0925 11320C                     LD    DE,KTBLS       ; SHIFT ON
0928 18F4                       JR    ?KY5
092A 11EA0B             ?KY6:   LD    DE,KTBL        ; NORMAL KEY
092D 18EF                       JR    ?KY5
092F 07                 ?KY4:   RLCA
0930 3808                       JR    C,?KY7
0932 07                         RLCA
0933 38E6                       JR    C,?KY5-3       ; JUMP CTRL
0935 116A0C                     LD    DE,KTBLG       ; GRAPH MODE
0938 18E4                       JR    ?KY5
093A 11A20C             ?KY7:   LD    DE,KTBLGS      ;GRAPH MODE & SHIFT ON
093D 18DF                       JR    ?KY5
093F                    ;
093F CDF109             ?KY9:   CALL  AUTCK
0942 3C                         INC   A              ; Z FLG CLR
0943 7B                         LD    A,E
0944 18A9                       JR    ?KY10
0946                    ;
0946                    ;
0946                    ;
0946                            ORG   0946H          ;?PRT   109
0946                    ;
0946                    ;
0946                    ;     PRINT ROUTINE
0946                    ;          1 CHA.
0946                    ;   C=ASCII DATA
0946                    ;
0946 79                 ?PRT:   LD    A,C
0947 CDB90B                     CALL  ?ADCN
094A 4F                         LD    C,A
094B E6F0                       AND   F0H
094D FEF0                       CP    F0H            ; ILLEGAL DATA
094F C8                         RET   Z
0950 FEC0                       CP    C0H
0952 79                         LD    A,C
0953 2017                       JR    NZ,PRNT3
0955 CDDC0D             PRNT5:  CALL  ?DPCT
0958 FEC3                       CP    C3H
095A 2813                       JR    Z,PRNT4
095C FEC5                       CP    C5H            ; HOME
095E 2807                       JR    Z,PRNT2
```

```
**  Z80 ASSEMBLER SB-7201  <MZ-80A,MONITOR>  PAGE 31            09/04/81

0960 FECD              CP     CDH           ; CR
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 CDB50D   PRNT3:   CALL   ?DSP
096F 3A9411   PRNT4:   LD     A,(DPRNT)     ; TAB POINT +1
0972 3C                INC    A
0973 FE50              CP     +80
0975 3802              JR     C,+4
0977 D650              SUB    +80
0979 18ED              JR     PRNT2+1
097B          ;
097B          ?NL:     ENT
097B 3A9411            LD     A,(DPRNT)
097E B7                OR     A
097F C8                RET    Z
0980          ;
0980          ;
0980          ;  NEW LINE
0980          ;
0980          ?LTNL:   ENT
0980 3ECD              LD     A,CDH         ; CR
0982 18D1              JR     PRNT5
0984          ;
0984          ;
0984          ;  PRINT TAB  ;
0984          ;
0984          ?PRTT:   ENT
0984 CD0C00            CALL   PRNTS
0987 3A9411            LD     A,(DPRNT)
098A B7                OR     A
098B C8                RET    Z
098C D60A              SUB    +10
098E 38F4              JR     C,-10
0990 20FA              JR     NZ,-4
0992 C9                RET
0993          ;
0993          ;   PRINT SPACE
0993          ;
0993          ?PRTS:   ENT
0993 3E20              LD     A,20H
0995          ;
0995          ;  PRINT ROUTINE  ;
0995          ;
0995          ?PRNT:   ENT
0995 FEOD              CP     ODH
0997 28E7              JR     Z,?LTNL
0999 C5                PUSH   BC
099A 4F                LD     C,A
099B 47                LD     B,A
099C CD4609            CALL   ?PRT
099F 78                LD     A,B
09A0 C1                POP    BC
09A1 C9                RET
09A2          ;
09A2          ;
09A2          ;   DLY3  324MICRO SEC DELAY
```

```
**  Z80 ASSEMBLER SB-7201  <MZ-80A,MONITOR>  PAGE 32            09/04/81

09A2          ;
09A2          ;      DLY2*3
09A2          ;
09A2 ED44     DLY3:    NEG
09A4 ED44              NEG
09A6 3E2A              LD     A,42
09A8 C36207            JP     DLY2+2
09AB          ;
09AB          ;  ONPU 2
09AB          ;
09AB 81       ONP3:    ADD    A,C
09AC 10FD              DJNZ   -1
09AE C1                POP    BC
09AF 4F                LD     C,A
09B0 AF                XOR    A
09B1 C9                RET
09B2          ;
09B2          ;
09B2          ;
09B3                   ORG    09B3H              ;??KEY   76
09B3          ;
09B3          ;  KEY BOAD SEARCH  ;
09B3          ;    & DISPLAY CODE CONV.  ;
09B3          ;
09B3          ;  EXIT A = DISPLAY CODE
09B3          ;  WITH CURSOR DISPLAY
09B3          ;
09B3 C5       ??KEY:   PUSH   BC
09B4 D5                PUSH   DE
09B5 E5                PUSH   HL
09B6 CD6302            CALL   ?SAVE
09B9 CDCA08   KSL2:    CALL   ?KEY               ; KEY INPUT
09BC CDFF09            CALL   ?FLAS
09BF 28F8              JR     Z,KSL2
09C1 CDF505            CALL   ?LOAD
09C4 C39F06            JP     RET3
09C7          ;
09C7          ;
09C7          ;    PAGE TOP CALCULATION;
09C7          ;     (PAGETP)=(PBIAS)*8
09C7          ;
09C7 D5       PAGE:    PUSH   DE
09C8 E5                PUSH   HL
09C9 217A11            LD     HL,PBIAS
09CC AF                XOR    A
09CD ED6F              RLD
09CF 57                LD     D,A
09D0 5E                LD     E,(HL)
09D1 ED67              RRD
09D3 AF                XOR    A
09D4 CB1A              RR     D
09D6 CB1B              RR     E
09D8 2100D0            LD     HL,SCRN
09DB 19                ADD    HL,DE
09DC 227D11            LD     (PAGETP),HL
09DF E1                POP    HL
09E0 D1                POP    DE
09E1 C9                RET
09E2          ;
09E2          ;
```

```
** Z80 ASSEMBLER SB-7201  <MZ-80A.MONITOR>  PAGE 33          09/04/81

09E2                  ;     CLEAR 2
09E2                  ;
09E2 AF               #CLR08: XOR    A
09E3 010008           #CLR8:  LD     BC,0800H
09E6 D5               CLEAR:  PUSH   DE
09E7 57                       LD     D,A
09E8 72               CLEAR1: LD     (HL),D
09E9 23                       INC    HL
09EA 0B                       DEC    BC
09EB 78                       LD     A,B
09EC B1                       OR     C
09ED 20F9                     JR     NZ,CLEAR1
09EF D1                       POP    DE
09F0 C9                       RET
09F1                  ;
09F1                  ;
09F1                  ;     AUTO REPEAT CHECK
09F1                  ;
09F1 216E11           AUTCK:  LD     HL,KDATW
09F4 7E                       LD     A,(HL)
09F5 23                       INC    HL
09F6 56                       LD     D,(HL)
09F7 77                       LD     (HL),A
09F8 92                       SUB    D
09F9 D0                       RET    NC
09FA 34                       INC    (HL)
09FB C9                       RET
09FC                  ;
09FC                  ;      00 MESSAGE
09FC                  ;
09FC 3030             00MSG:  DEFW   3030H
09FE 0D                       DEFB   0DH
09FF                  ;
09FF                  ;
09FF                  ;
09FF                          ORG    09FFH              ;?FLAS    81
09FF                  ;
09FF                  ;
09FF                  ;
09FF                  ;  FLASHING  1
09FF                  ;
09FF F5               ?FLAS:  PUSH   AF
0A00 E5                       PUSH   HL
0A01 3A02E0                   LD     A,(KEYPC)
0A04 07                       RLCA
0A05 07                       RLCA
0A06 380A                     JR     C,FLAS1
0A08 3A9211                   LD     A,(FLSDT)
0A0B CDB10F           FLAS2:  CALL   ?PONT
0A0E 77                       LD     (HL),A
0A0F E1               FLAS3:  POP    HL
0A10 F1                       POP    AF
0A11 C9                       RET
0A12 3A8E11           FLAS1:  LD     A,(FLASH)
0A15 18F4                     JR     FLAS2
0A17                  ;
0A17                  ;   REVERSE CRT
0A17                  ;
0A17 219011           REV:    LD     HL,REVFLG
0A1A 7E                       LD     A,(HL)
```

```
** Z80 ASSEMBLER SB-7201  <MZ-80A.MONITOR>  PAGE 34          09/04/81

0A1B B7                       OR     A
0A1C 2F                       CPL
0A1D 77                       LD     (HL),A
0A1E 2805                     JR     Z,REV1
0A20 3A14E0                   LD     A,(E014H)          ; RESET
0A23 1803                     JR     +5
0A25 3A15E0           REV1:   LD     A,(E015H)          ; SET
0A28 C3E50E                   JP     ?RSTR
0A2B                  ;
0A2B                  ;
0A2B                  ;   CRT MANAGEMENT
0A2B                  ;      EXIT  HL:DSPXY
0A2B                  ;            DE:MANG ADR. (ON DSPXY)
0A2B                  ;            B :MANG BIT POSITION
0A2B                  ;            A :MANG DATA
0A2B                  ;            CY:MANG=1
0A2B                  ;
0A2B 217311           .MANG:  LD     HL,MANG
0A2E 3A9111                   LD     A,(SPAGE)
0A31 B7                       OR     A
0A32 C2A603                   JP     NZ,.MANG2          ; PAGE MODE1
0A35 3A7C11                   LD     A,(MGPNT)
0A38 D608                     SUB    08H
0A3A 23                       INC    HL
0A3B 30FB                     JR     NC,-3
0A3D C608                     ADD    A,08H
0A3F 4E                       LD     C,(HL)
0A40 2B                       DEC    HL
0A41 47                       LD     B,A
0A42 04                       INC    B
0A43 C5                       PUSH   BC
0A44 7E                       LD     A,(HL)
0A45 CB19                     RR     C
0A47 1F                       RRA
0A48 10FB                     DJNZ   -3
0A4A C1                       POP    BC
0A4B EB                       EX     DE,HL
0A4C 2A7111           .MANG1: LD     HL,(DSPXY)
0A4F C9                       RET
0A50                  ;
0A50                  ;
0A50                  ;
0A50                  ;;
0A50                          ORG    0A50H              ; ?SWEP #106#111
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                  ;        C    = ROW & COLOUMN
0A50                  ;
0A50 D5               ?SWEP:  PUSH   DE
0A51 E5                       PUSH   HL
0A52 AF                       XOR    A
0A53 326E11                   LD     (KDATW),A          ; RESET KEY COUNTER
0A56 06FA                     LD     B,FAH
0A58 57                       LD     D,A
```

```
**  Z80 ASSEMBLER SB-7201  <MZ-80A,MONITOR>  PAGE 35          09/04/81

0A59 CD110D          CALL  ?BRK
0A5C 2004            JR    NZ,SWEP6
0A5E 1688            LD    D,88H           ; BREAK ON
0A60 1828            JR    SWEP9
0A62 216411  SWEP6:  LD    HL,SWPW
0A65 E5              PUSH  HL
0A66 3026            JR    NC,SWEP11
0A68 57              LD    D,A
0A69 E660            AND   60H             ; SHIFT & CTRL =NO DATA
0A6B 2021            JR    NZ,SWEP11
0A6D 7A              LD    A,D             ; GRAPH CHECK
0A6E AE              XOR   (HL)
0A6F CB67            BIT   4,A
0A71 72              LD    (HL),D
0A72 2802            JR    Z,SWEP0
0A74 CBFA    SWEP01: SET   7,D
0A76 05      SWEP0:  DEC   B
0A77 E1              POP   HL              ; SWEP COLOUMN WORK
0A78 23              INC   HL
0A79 78              LD    A,B
0A7A 3200E0          LD    (KEYPA),A
0A7D FEF0            CP    F0H
0A7F 2011            JR    NZ,SWEP3
0A81 7E              LD    A,(HL)          ; KDATW
0A82 FE03            CP    03H
0A84 3804            JR    C,SWEP9
0A86 3600            LD    (HL),0
0A88 CBBA            RES   7,D             ; RESET DATA IN
0A8A 42      SWEP9:  LD    B,D
0A8B E1              POP   HL
0A8C D1              POP   DE
0A8D C9              RET
0A8E                 ;
0A8E 3600    SWEP11: LD    (HL),0
0A90 18E4            JR    SWEP0
0A92 3A01E0  SWEP3:  LD    A,(KEYPB)
0A95 5F              LD    E,A
0A96 2F              CPL
0A97 A6              AND   (HL)
0A98 73              LD    (HL),E
0A99 E5              PUSH  HL
0A9A 216E11          LD    HL,KDATW
0A9D C5              PUSH  BC
0A9E 0608            LD    B,8
0AA0 CB03    SWEP8:  RLC   E
0AA2 3801            JR    C,SWEP7
0AA4 34              INC   (HL)
0AA5 10F9    SWEP7:  DJNZ  SWEP8
0AA7 C1              POP   BC
0AA8 B7              OR    A
0AA9 28CB            JR    Z,SWEP0
0AAB 5F              LD    E,A
0AAC 2608    SWEP2:  LD    H,8
0AAE 78              LD    A,B
0AAF 3D              DEC   A               ; TBL ADJUST
0AB0 E60F            AND   0FH
0AB2 C33E07          JP    SWEP4
0AB5                 ;
0AB5                 ;  ASCII TO DISPLAY CODE TABL  ;
0AB5                 ;
```

```
**  Z80 ASSEMBLER SB-7201  <MZ-80A,MONITOR>  PAGE 36          09/04/81

0AB5         ATBL:
0AB5         ;  00 - 0F  ;
0AB5 CC              DEFB  CCH       ; ↑@
0AB6 E0              DEFB  E0H       ; ↑A SHIFT LOCK
0AB7 F2              DEFB  F2H       ; ↑B
0AB8 F3              DEFB  F3H       ; ↑C
0AB9 CE              DEFB  CEH       ; ↑D ROLL UP
0ABA CF              DEFB  CFH       ; ↑E ROLL DOWN
0ABB F6              DEFB  F6H       ; ↑F
0ABC F7              DEFB  F7H       ; ↑G
0ABD F8              DEFB  F8H       ; ↑H
0ABE F9              DEFB  F9H       ; ↑I
0ABF FA              DEFB  FAH       ; ↑J
0AC0 FB              DEFB  FBH       ; ↑K
0AC1 FC              DEFB  FCH       ; ↑L
0AC2 FD              DEFB  FDH       ; ↑M
0AC3 FE              DEFB  FEH       ; ↑N
0AC4 FF              DEFB  FFH       ; ↑O
0AC5         ;  10 - 1F
0AC5 E1              DEFB  E1H       ; ↑P
0AC6 C1              DEFB  C1H       ; ↑Q CUR. DOWN
0AC7 C2              DEFB  C2H       ; ↑R CUR. UP
0AC8 C3              DEFB  C3H       ; ↑S CUR. RIGHT
0AC9 C4              DEFB  C4H       ; ↑T CUR. LEFT
0ACA C5              DEFB  C5H       ; ↑U HOME
0ACB C6              DEFB  C6H       ; ↑V CLEAR
0ACC E2              DEFB  E2H       ; ↑W
0ACD E3              DEFB  E3H       ; ↑X
0ACE E4              DEFB  E4H       ; ↑Y
0ACF E5              DEFB  E5H       ; ↑Z SEP.
0AD0 E6              DEFB  E6H       ; ↑[
0AD1 EB              DEFB  EBH       ; ↑\
0AD2 EE              DEFB  EEH       ; ↑]
0AD3 EF              DEFB  EFH       ; ↑^
0AD4 F4              DEFB  F4H       ; ↑-
0AD5         ;  20 - 2F  ;
0AD5 00              DEFB  00H       ; SPACE
0AD6 61              DEFB  61H       ; !
0AD7 62              DEFB  62H       ; "
0AD8 63              DEFB  63H       ; #
0AD9 64              DEFB  64H       ; $
0ADA 65              DEFB  65H       ; %
0ADB 66              DEFB  66H       ; &
0ADC 67              DEFB  67H       ; '
0ADD 68              DEFB  68H       ; (
0ADE 69              DEFB  69H       ; )
0ADF 6B              DEFB  6BH       ; *
0AE0 6A              DEFB  6AH       ; +
0AE1 2F              DEFB  2FH       ; ,
0AE2 2A              DEFB  2AH       ; -
0AE3 2E              DEFB  2EH       ; .
0AE4 2D              DEFB  2DH
0AE5         ;  30 -3F  ;
0AE5 20              DEFB  20H       ; 0
0AE6 21              DEFB  21H       ; 1
0AE7 22              DEFB  22H       ; 2
0AE8 23              DEFB  23H       ; 3
0AE9 24              DEFB  24H       ; 4
0AEA 25              DEFB  25H       ; 5
0AEB 26              DEFB  26H       ; 6
```

```
**  Z80 ASSEMBLER SB-7201  <MZ-80A.MONITOR>  PAGE 37          09/04/81

01 0AEC 27                DEFB  27H        ; 7
02 0AED 28                DEFB  28H        ; 8
03 0AEE 29                DEFB  29H        ; 9
04 0AEF 4F                DEFB  4FH        ; :
05 0AF0 2C                DEFB  2CH        ; ;
06 0AF1 51                DEFB  51H        ; <
07 0AF2 2B                DEFB  2BH        ; =
08 0AF3 57                DEFB  57H        ; >
09 0AF4 49                DEFB  49H        ; ?
10 0AF5           ;  40 - 4F  ;
11 0AF5 55                DEFB  55H        ; @
12 0AF6 01                DEFB  01H        ; A
13 0AF7 02                DEFB  02H        ; B
14 0AF8 03                DEFB  03H        ; C
15 0AF9 04                DEFB  04H        ; D
16 0AFA 05                DEFB  05H        ; E
17 0AFB 06                DEFB  06H        ; F
18 0AFC 07                DEFB  07H        ; G
19 0AFD 08                DEFB  08H        ; H
20 0AFE 09                DEFB  09H        ; I
21 0AFF 0A                DEFB  0AH        ; J
22 0B00 0B                DEFB  0BH        ; K
23 0B01 0C                DEFB  0CH        ; L
24 0B02 0D                DEFB  0DH        ; M
25 0B03 0E                DEFB  0EH        ; N
26 0B04 0F                DEFB  0FH        ; O
27 0B05           ;  50 - 5F  ;
28 0B05 10                DEFB  10H        ; P
29 0B06 11                DEFB  11H        ; Q
30 0B07 12                DEFB  12H        ; R
31 0B08 13                DEFB  13H        ; S
32 0B09 14                DEFB  14H        ; T
33 0B0A 15                DEFB  15H        ; U
34 0B0B 16                DEFB  16H        ; V
35 0B0C 17                DEFB  17H        ; W
36 0B0D 18                DEFB  18H        ; X
37 0B0E 19                DEFB  19H        ; Y
38 0B0F 1A                DEFB  1AH        ; Z
39 0B10 52                DEFB  52H        ; [
40 0B11 59                DEFB  59H        ; \
41 0B12 54                DEFB  54H        ; ]
42 0B13 50                DEFB  50H        ; ^
43 0B14 45                DEFB  45H        ; -
44 0B15           ;  60 - 6F  ;
45 0B15 C7                DEFB  C7H        ; UFO
46 0B16 C8                DEFB  C8H
47 0B17 C9                DEFB  C9H
48 0B18 CA                DEFB  CAH
49 0B19 CB                DEFB  CBH
50 0B1A CC                DEFB  CCH
51 0B1B CD                DEFB  CDH
52 0B1C CE                DEFB  CEH
53 0B1D CF                DEFB  CFH
54 0B1E DF                DEFB  DFH
55 0B1F E7                DEFB  E7H
56 0B20 E8                DEFB  E8H
57 0B21 E9                DEFB  E9H
58 0B22 EA                DEFB  EAH
59 0B23 EC                DEFB  ECH
60 0B24 ED                DEFB  EDH
```

```
**  Z80 ASSEMBLER SB-7201  <MZ-80A.MONITOR>  PAGE 38          09/04/81

01 0B25           ;  70 - 7F  ;
02 0B25 D0                DEFB  D0H
03 0B26 D1                DEFB  D1H
04 0B27 D2                DEFB  D2H
05 0B28 D3                DEFB  D3H
06 0B29 D4                DEFB  D4H
07 0B2A D5                DEFB  D5H
08 0B2B D6                DEFB  D6H
09 0B2C D7                DEFB  D7H
10 0B2D D8                DEFB  D8H
11 0B2E D9                DEFB  D9H
12 0B2F DA                DEFB  DAH
13 0B30 DB                DEFB  DBH
14 0B31 DC                DEFB  DCH
15 0B32 DD                DEFB  DDH
16 0B33 DE                DEFB  DEH
17 0B34 C0                DEFB  C0H
18 0B35           ;  80 - 8F  ;
19 0B35 40                DEFB  40H        ; )
20 0B36 BD                DEFB  BDH        ;
21 0B37 9D                DEFB  9DH
22 0B38 B1                DEFB  B1H
23 0B39 B5                DEFB  B5H
24 0B3A B9                DEFB  B9H
25 0B3B B4                DEFB  B4H        ; ''O
26 0B3C 9E                DEFB  9EH        ; ''A
27 0B3D B2                DEFB  B2H        ; ''I
28 0B3E B6                DEFB  B6H        ; ''U
29 0B3F BA                DEFB  BAH        ; ''E
30 0B40 BE                DEFB  BEH        ; ^
31 0B41 9F                DEFB  9FH        ; ''YA
32 0B42 B3                DEFB  B3H        ; ''YU
33 0B43 B7                DEFB  B7H        ; ''YO
34 0B44 BB                DEFB  BBH        ; ''TSU
35 0B45           ;  90 - 9F  ;
36 0B45 BF                DEFB  BFH        ; -
37 0B46 A3                DEFB  A3H        ; 'A
38 0B47 85                DEFB  85H        ; 'I
39 0B48 A4                DEFB  A4H        ; '
40 0B49 A5                DEFB  A5H        ; -
41 0B4A A6                DEFB  A6H        ; 'O
42 0B4B 94                DEFB  94H        ; 'KA
43 0B4C 87                DEFB  87H        ; 'KI
44 0B4D 88                DEFB  88H        ; 'KU
45 0B4E 9C                DEFB  9CH        ; 'KE
46 0B4F 82                DEFB  82H        ; 'KO
47 0B50 98                DEFB  98H        ; 'SA
48 0B51 84                DEFB  84H        ; 'SHI
49 0B52 92                DEFB  92H        ; 'SU
50 0B53 90                DEFB  90H        ; 'SE
51 0B54 83                DEFB  83H        ; 'SO
52 0B55           ;  A0 - AF  ;
53 0B55 91                DEFB  91H        ; 'TA
54 0B56 81                DEFB  81H        ; 'CHI
55 0B57 9A                DEFB  9AH        ; 'TSU
56 0B58 97                DEFB  97H        ; 'TE
57 0B59 93                DEFB  93H        ; 'TO
58 0B5A 95                DEFB  95H        ; 'NA
59 0B5B 89                DEFB  89H        ; 'NI
60 0B5C A1                DEFB  A1H        ; 'NU
```

```
**  Z80 ASSEMBLER SB-7201  <MZ-80A.MONITOR>  PAGE 39          09/04/81

0B5D AF                  DEFB  AFH      ; 'NE
0B5E 8B                  DEFB  8BH      ; 'NO
0B5F 86                  DEFB  86H      ; 'HA
0B60 96                  DEFB  96H      ; 'HI
0B61 A2                  DEFB  A2H      ; 'FU
0B62 AB                  DEFB  ABH      ; 'HE
0B63 AA                  DEFB  AAH      ; 'HO
0B64 8A                  DEFB  8AH      ; 'MA
0B65              ; B0 - BF ;
0B65 8E                  DEFB  8EH      ; 'MI
0B66 B0                  DEFB  B0H      ; 'MU
0B67 AD                  DEFB  ADH      ; 'ME
0B68 8D                  DEFB  8DH      ; 'MO
0B69 A7                  DEFB  A7H      ; 'YA
0B6A A8                  DEFB  A8H      ; 'YU
0B6B A9                  DEFB  A9H      ; 'YO
0B6C 8F                  DEFB  8FH      ; 'RA
0B6D 8C                  DEFB  8CH      ; 'RI
0B6E AE                  DEFB  AEH      ; 'RU
0B6F AC                  DEFB  ACH      ; 'RE
0B70 9B                  DEFB  9BH      ; 'RO
0B71 A0                  DEFB  A0H      ; 'WA
0B72 99                  DEFB  99H      ; 'N
0B73 BC                  DEFB  BCH      ; (
0B74 B8                  DEFB  B8H
0B75              ; C0 - CF ;
0B75 80                  DEFB  80H      ; )
0B76 3B                  DEFB  3BH
0B77 3A                  DEFB  3AH
0B78 70                  DEFB  70H
0B79 3C                  DEFB  3CH
0B7A 71                  DEFB  71H
0B7B 5A                  DEFB  5AH
0B7C 3D                  DEFB  3DH
0B7D 43                  DEFB  43H
0B7E 56                  DEFB  56H
0B7F 3F                  DEFB  3FH
0B80 1E                  DEFB  1EH
0B81 4A                  DEFB  4AH
0B82 1C                  DEFB  1CH
0B83 5D                  DEFB  5DH
0B84 3E                  DEFB  3EH
0B85              ; D0 - DF ;
0B85 5C                  DEFB  5CH
0B86 1F                  DEFB  1FH
0B87 5F                  DEFB  5FH
0B88 5E                  DEFB  5EH
0B89 37                  DEFB  37H
0B8A 7B                  DEFB  7BH
0B8B 7F                  DEFB  7FH
0B8C 36                  DEFB  36H
0B8D 7A                  DEFB  7AH
0B8E 7E                  DEFB  7EH
0B8F 33                  DEFB  33H
0B90 4B                  DEFB  4BH
0B91 4C                  DEFB  4CH
0B92 1D                  DEFB  1DH
0B93 6C                  DEFB  6CH
0B94 5B                  DEFB  5BH
0B95              ; E0 - EF ;
```

```
**  Z80 ASSEMBLER SB-7201  <MZ-80A.MONITOR>  PAGE 40          09/04/81

0B95 78                  DEFB  78H
0B96 41                  DEFB  41H
0B97 35                  DEFB  35H
0B98 34                  DEFB  34H
0B99 74                  DEFB  74H
0B9A 30                  DEFB  30H
0B9B 38                  DEFB  38H
0B9C 75                  DEFB  75H
0B9D 39                  DEFB  39H
0B9E 4D                  DEFB  4DH
0B9F 6F                  DEFB  6FH
0BA0 6E                  DEFB  6EH
0BA1 32                  DEFB  32H
0BA2 77                  DEFB  77H
0BA3 76                  DEFB  76H
0BA4 72                  DEFB  72H
0BA5              ; F0 - FF ;
0BA5 73                  DEFB  73H
0BA6 47                  DEFB  47H
0BA7 7C                  DEFB  7CH
0BA8 53                  DEFB  53H
0BA9 31                  DEFB  31H
0BAA 4E                  DEFB  4EH
0BAB 6D                  DEFB  6DH
0BAC 48                  DEFB  48H
0BAD 46                  DEFB  46H
0BAE 7D                  DEFB  7DH
0BAF 44                  DEFB  44H
0BB0 1B                  DEFB  1BH
0BB1 58                  DEFB  58H
0BB2 79                  DEFB  79H
0BB3 42                  DEFB  42H
0BB4 60                  DEFB  60H
0BB5              ;
0BB9                     ORG   0BB9H           ;?ADCN   21
0BB9              ;
0BB9              ;
0BB9              ;     ASCII TO DISPLAY CODE CONVERTE
0BB9              ;
0BB9              ;          IN   ACC:ASCII
0BB9              ;          EXIT ACC:DISPLAY CODE
0BB9              ;
0BB9 C5           ?ADCN: PUSH  BC
0BBA E5                  PUSH  HL
0BBB 21B50A              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              ;
0BC5              ;   ?KEY BREAK KEY INPUT
0BC5              ;
0BC5 3ECB         #BRK:  LD    A,CBH           ; BREAK CODE
0BC7 B7                  OR    A
0BC8 C3EF08              JP    ?KY10
0BCB              ;
0BCB              ;
0BCB              ;
```

```
0BCE               ORG   0BCEH         ;?DACN   472
0BCE         ;
0BCE         ;  DISPLAY CODE TO ASCII CONV.  ;
0BCE         ;
0BCE         ;     ACC = DISPLAY CODE
0BCE         ;   EXIT ACC = ASCII
0BCE         ;
0BCE C5      ?DACN:  PUSH  BC
0BCF E5              PUSH  HL
0BD0 D5              PUSH  DE
0BD1 21B50A          LD    HL,ATBL
0BD4 54              LD    D,H
0BD5 5D              LD    E,L
0BD6 010001          LD    BC,0100H
0BD9 EDB1            CPIR
0BDB 2806            JR    Z,DACN1
0BDD 3EF0            LD    A,F0H
0BDF D1      DACN2:  POP   DE
0BE0 E1      DACN3:  POP   HL
0BE1 C1              POP   BC
0BE2 C9              RET
0BE3         ;
0BE3 B7      DACN1:  OR    A
0BE4 2B              DEC   HL
0BE5 ED52            SBC   HL,DE
0BE7 7D              LD    A,L
0BE8 18F5            JR    DACN2
0BEA         ;
0BEA         ;
0BEA         ;
0BEA         ;  KEY MATRIX TO DISPLAY CODE TABL
0BEA         ;
0BEA         KTBL:
0BEA         ;S0   00 - 07  ;
0BEA 22              DEFB  22H         ; 2
0BEB 21              DEFB  21H         ; 1
0BEC 17              DEFB  17H         ; W
0BED 11              DEFB  11H         ; Q
0BEE 01              DEFB  01H         ; A
0BEF C7              DEFB  C7H         ; DEL
0BF0 00              DEFB  00H         ; NULL
0BF1 1A              DEFB  1AH         ; Z
0BF2         ;S1   08 - 0F  ;
0BF2 24              DEFB  24H         ; 4
0BF3 23              DEFB  23H         ; 3
0BF4 12              DEFB  12H         ; R
0BF5 05              DEFB  05H         ; E
0BF6 04              DEFB  04H         ; D
0BF7 13              DEFB  13H         ; S
0BF8 18              DEFB  18H         ; X
0BF9 03              DEFB  03H         ; C
0BFA         ;S2    0 - 17  ;
0BFA 26              DEFB  26H         ; 6
0BFB 25              DEFB  25H         ; 5
0BFC 19              DEFB  19H         ; Y
0BFD 14              DEFB  14H         ; T
0BFE 07              DEFB  07H         ; G
0BFF 06              DEFB  06H         ; F
0C00 16              DEFB  16H         ; V
0C01 02              DEFB  02H         ; B
```



```
0C02         ;S3   18 - 1F  ;
0C02 28              DEFB  28H         ; 8
0C03 27              DEFB  27H         ; 7
0C04 09              DEFB  09H         ; I
0C05 15              DEFB  15H         ; U
0C06 0A              DEFB  0AH         ; J
0C07 08              DEFB  08H         ; H
0C08 0E              DEFB  0EH         ; N
0C09 00              DEFB  00H         ; SPACE
0C0A         ;S4   20 - 27  ;
0C0A 20              DEFB  20H         ; 0
0C0B 29              DEFB  29H         ; 9
0C0C 10              DEFB  10H         ; P
0C0D 0F              DEFB  0FH         ; O
0C0E 0C              DEFB  0CH         ; L
0C0F 0B              DEFB  0BH         ; K
0C10 2F              DEFB  2FH         ; ,
0C11 0D              DEFB  0DH         ; M
0C12         ;S5   28 - 2F  ;
0C12 BE              DEFB  BEH         ; ^
0C13 2A              DEFB  2AH         ; -
0C14 52              DEFB  52H         ; [
0C15 55              DEFB  55H         ; @
0C16 4F              DEFB  4FH         ; :
0C17 2C              DEFB  2CH         ; ;
0C18 2D              DEFB  2DH         ; /
0C19 2E              DEFB  2EH         ; .
0C1A         ;S6   30 - 37  ;
0C1A C5              DEFB  C5H         ; HOME
0C1B 59              DEFB  59H         ; \
0C1C C3              DEFB  C3H         ; CURSOL *RIGHT
0C1D C2              DEFB  C2H         ; CURSOL *UP
0C1E CD              DEFB  CDH         ; CR
0C1F 54              DEFB  54H         ; ]
0C20 00              DEFB  00H         ;NULL
0C21 49              DEFB  49H         ; ?
0C22         ;S7   38 - 3F  ;
0C22 28              DEFB  28H         ; 8
0C23 27              DEFB  27H         ; 7
0C24 25              DEFB  25H         ; 5
0C25 24              DEFB  24H         ; 4
0C26 22              DEFB  22H         ; 2
0C27 21              DEFB  21H         ; 1
0C28 E7              DEFB  E7H         ; 00   ***
0C29 20              DEFB  20H         ; 0
0C2A         ;S8   40 - 47  ;
0C2A 6A              DEFB  6AH         ; *
0C2B 29              DEFB  29H         ; 9
0C2C 2A              DEFB  2AH         ; -
0C2D 26              DEFB  26H         ; 6
0C2E 00              DEFB  00H         ; NULL
0C2F 23              DEFB  23H         ; 3
0C30 00              DEFB  00H         ;NULL
0C31 2E              DEFB  2EH         ;.
0C32         ;   KTBL SHIFT ON
0C32         KTBLS:
0C32         ;S0
0C32 62              DEFB  62H         ; "
0C33 61              DEFB  61H         ; !
0C34 97              DEFB  97H         ; w
```

```
**  Z80 ASSEMBLER SB-7201  <MZ-80A.MONITOR>  PAGE 43          09/04/81

0C35 91                 DEFB  91H        ; q
0C36 81                 DEFB  81H        ; a
0C37 C8                 DEFB  C8H        ; INSERT
0C38 00                 DEFB  00H        ; NULL
0C39 9A                 DEFB  9AH        ; z
0C3A          :S1
0C3A 64                 DEFB  64H        ; $
0C3B 63                 DEFB  63H        ; ■
0C3C 92                 DEFB  92H        ; r
0C3D 85                 DEFB  85H        ; e
0C3E 84                 DEFB  84H        ; d
0C3F 93                 DEFB  93H        ; s
0C40 98                 DEFB  98H        ; x
0C41 83                 DEFB  83H        ; c
0C42          :S2
0C42 66                 DEFB  66H        ; &
0C43 65                 DEFB  65H        ; %
0C44 99                 DEFB  99H        ; y
0C45 94                 DEFB  94H        ; t
0C46 87                 DEFB  87H        ; g
0C47 86                 DEFB  86H        ; f
0C48 96                 DEFB  96H        ; v
0C49 82                 DEFB  82H        ; b
0C4A          :S3
0C4A 68                 DEFB  68H        ; (
0C4B 67                 DEFB  67H        ; '
0C4C 89                 DEFB  89H        ; i
0C4D 95                 DEFB  95H        ; u
0C4E 8A                 DEFB  8AH        ; j
0C4F 88                 DEFB  88H        ; h
0C50 8E                 DEFB  8EH        ; n
0C51 00                 DEFB  00H        ; SPACE
0C52          :S4
0C52 BF                 DEFB  BFH        ; ‾
0C53 69                 DEFB  69H        ; )
0C54 90                 DEFB  90H        ; p
0C55 8F                 DEFB  8FH        ; o
0C56 8C                 DEFB  8CH        ; l
0C57 8B                 DEFB  8BH        ; k
0C58 51                 DEFB  51H        ; <
0C59 8D                 DEFB  8DH        ; m
0C5A          :S5
0C5A A5                 DEFB  A5H        ; ~
0C5B 2B                 DEFB  2BH        ; =
0C5C BC                 DEFB  BCH        ; {
0C5D A4                 DEFB  A4H        ; '
0C5E 6B                 DEFB  6BH        ; *
0C5F 6A                 DEFB  6AH        ; +
0C60 45                 DEFB  45H        ; +
0C61 57                 DEFB  57H        ; >
0C62          :S6
0C62 C6                 DEFB  C6H        ; CLR
0C63 80                 DEFB  80H        ; |
0C64 C4                 DEFB  C4H        ; CURSOL ←LEFT
0C65 C1                 DEFB  C1H        ; CURSOL ↓DOWN
0C66 CD                 DEFB  CDH        ; CR
0C67 40                 DEFB  40H        ; }
0C68 00                 DEFB  00H        ; NULL
0C69 50                 DEFB  50H        ; ↑
0C6A          KTBLG:
```

```
**  Z80 ASSEMBLER SB-7201  <MZ-80A.MONITOR>  PAGE 44          09/04/81

0C6A          :S0        GRAPHIC MODE
0C6A 3E                 DEFB  3EH        ; ■2
0C6B 37                 DEFB  37H        ; ■1
0C6C 38                 DEFB  38H        ; ■W
0C6D 3C                 DEFB  3CH        ; ■Q
0C6E 53                 DEFB  53H        ; ■A
0C6F C7                 DEFB  C7H        ; DELETE
0C70 00                 DEFB  00H        ; NULL
0C71 76                 DEFB  76H        ; ■Z
0C72          :S1
0C72 7B                 DEFB  7BH        ; ■4
0C73 7F                 DEFB  7FH        ; ■3
0C74 30                 DEFB  30H        ; ■R
0C75 34                 DEFB  34H        ; ■E
0C76 47                 DEFB  47H        ; ■D
0C77 44                 DEFB  44H        ; ■S
0C78 6D                 DEFB  6DH        ; ■X
0C79 DE                 DEFB  DEH        ; ■C
0C7A          :S2
0C7A 5E                 DEFB  5EH        ; ■6
0C7B 3A                 DEFB  3AH        ; ■5
0C7C 75                 DEFB  75H        ; ■Y
0C7D 71                 DEFB  71H        ; ■T
0C7E 4B                 DEFB  4BH        ; ■G
0C7F 4A                 DEFB  4AH        ; ■F
0C80 DA                 DEFB  DAH        ; ■V
0C81 6F                 DEFB  6FH        ; ■B
0C82          :S3
0C82 BD                 DEFB  BDH        ; ■8
0C83 1F                 DEFB  1FH        ; ■7
0C84 7D                 DEFB  7DH        ; ■I
0C85 79                 DEFB  79H        ; ■U
0C86 5C                 DEFB  5CH        ; ■J
0C87 72                 DEFB  72H        ; ■H
0C88 32                 DEFB  32H        ; ■N
0C89 00                 DEFB  00H        ; SPACE
0C8A          :S4
0C8A 9C                 DEFB  9CH        ; ■0
0C8B A1                 DEFB  A1H        ; ■9
0C8C D6                 DEFB  D6H        ; ■P
0C8D B0                 DEFB  B0H        ; ■O
0C8E B4                 DEFB  B4H        ; ■L
0C8F 5B                 DEFB  5BH        ; ■K
0C90 60                 DEFB  60H        ; ←PAI,■,
0C91 1C                 DEFB  1CH        ; ■M
0C92          :S5
0C92 9E                 DEFB  9EH        ; ■^
0C93 D2                 DEFB  D2H        ; ■-
0C94 D8                 DEFB  D8H        ; ■[
0C95 B2                 DEFB  B2H        ; ■@
0C96 B6                 DEFB  B6H        ; ■;
0C97 42                 DEFB  42H        ; ■:
0C98 DB                 DEFB  DBH        ; ■/
0C99 B8                 DEFB  B8H        ; ■.
0C9A          :S6
0C9A C5                 DEFB  C5H        ; HOME
0C9B D4                 DEFB  D4H        ; ■\
0C9C C3                 DEFB  C3H        ; CURSOL →RIGHT
0C9D C2                 DEFB  C2H        ; CURSOL ↑UP
0C9E CD                 DEFB  CDH        ; CR
```

```
**  Z80 ASSEMBLER SB-7201  <MZ-80A.MONITOR>  PAGE 45          09/04/81

0C9F 4E                      DEFB  4EH           ; ■]
0CA0 00                      DEFB  00H           ; NULL
0CA1 BA                      DEFB  BAH           ; ■?
0CA2                  KTBLGS:
0CA2                  ;S0     GRAPHIC MODE & SHIFT ON
0CA2 36                      DEFB  36H           ; ?2
0CA3 3F                      DEFB  3FH           ; ?1
0CA4 78                      DEFB  78H           ; ?W
0CA5 7C                      DEFB  7CH           ; ?Q
0CA6 46                      DEFB  46H           ; ?A
0CA7 C8                      DEFB  C8H           ; INST
0CA8 00                      DEFB  00H           ; NULL
0CA9 77                      DEFB  77H           ; ?Z
0CAA                  ;S1
0CAA 3B                      DEFB  3BH           ; ?4
0CAB 7E                      DEFB  7EH           ; ?3
0CAC 70                      DEFB  70H           ; ?R
0CAD 74                      DEFB  74H           ; ?E
0CAE 48                      DEFB  48H           ; ?D
0CAF 41                      DEFB  41H           ; ?S
0CB0 DD                      DEFB  DDH           ; ?X
0CB1 D9                      DEFB  D9H           ; ?C
0CB2                  ;S2
0CB2 1E                      DEFB  1EH           ; ?6
0CB3 7A                      DEFB  7AH           ; ?5
0CB4 35                      DEFB  35H           ; ?Y
0CB5 31                      DEFB  31H           ; ?T
0CB6 4C                      DEFB  4CH           ; ?G
0CB7 43                      DEFB  43H           ; ?F
0CB8 A6                      DEFB  A6H           ; ?V
0CB9 6E                      DEFB  6EH           ; ?B
0CBA                  ;S3
0CBA A2                      DEFB  A2H           ; ?8
0CBB 5F                      DEFB  5FH           ; ?7
0CBC 3D                      DEFB  3DH           ; ?I
0CBD 39                      DEFB  39H           ; ?U
0CBE 5D                      DEFB  5DH           ; ?J
0CBF 73                      DEFB  73H           ; ?H
0CC0 33                      DEFB  33H           ; ?N
0CC1 00                      DEFB  00H           ; SPACE
0CC2                  ;S4
0CC2 9D                      DEFB  9DH           ; ?0
0CC3 A3                      DEFB  A3H           ; ?9
0CC4 B1                      DEFB  B1H           ; ?P
0CC5 D5                      DEFB  D5H           ; ?O
0CC6 56                      DEFB  56H           ; ?L
0CC7 6C                      DEFB  6CH           ; ?K
0CC8 D0                      DEFB  D0H           ; ?,
0CC9 1D                      DEFB  1DH           ; ?M
0CCA                  ;S5B
0CCA 9F                      DEFB  9FH           ; ?^
0CCB D1                      DEFB  D1H           ; ?-
0CCC B3                      DEFB  B3H           ; ?[
0CCD D7                      DEFB  D7H           ; ?@
0CCE 4D                      DEFB  4DH           ; ?:
0CCF B5                      DEFB  B5H           ; ?;
0CD0 1B                      DEFB  1BH           ; ?/
0CD1 B9                      DEFB  B9H           ; ?.
0CD2                  ;S6
0CD2 C6                      DEFB  C6H           ; CLR
```

```
**  Z80 ASSEMBLER SB-7201  <MZ-80A.MONITOR>  PAGE 46          09/04/81

0CD3 D3                      DEFB  D3H           ; ?\
0CD4 C4                      DEFB  C4H           ; CURSOL →LEFT
0CD5 C1                      DEFB  C1H           ; CURSOL ↓DOWN
0CD6 CD                      DEFB  CDH           ; CR
0CD7 B7                      DEFB  B7H           ; ?]
0CD8 00                      DEFB  00H           ; NULL
0CD9 BB                      DEFB  BBH           ; ??
0CDA                  ;
0CDA                  KTBLC:
0CDA                  ;S0     CTRL ON
0CDA F0                      DEFB  F0H           ; CODE 80H=NOT KEY
0CDB F0                      DEFB  F0H
0CDC E2                      DEFB  E2H           ; ↑W
0CDD C1                      DEFB  C1H           ; ↑Q
0CDE E0                      DEFB  E0H           ; ↑A SHIFT LOCK
0CDF F0                      DEFB  F0H
0CE0 00                      DEFB  00H
0CE1 E5                      DEFB  E5H           ; ↑Z
0CE2                  ;S1
0CE2 F0                      DEFB  F0H
0CE3 F0                      DEFB  F0H
0CE4 C2                      DEFB  C2H           ; ↑R
0CE5 CF                      DEFB  CFH           ; ↑E ROLL DOWN
0CE6 CE                      DEFB  CEH           ; ↑D ROLL UP
0CE7 C3                      DEFB  C3H           ; ↑S
0CE8 E3                      DEFB  E3H           ; ↑X
0CE9 F3                      DEFB  F3H           ; ↑C
0CEA                  ;S2
0CEA F0                      DEFB  F0H
0CEB F0                      DEFB  F0H
0CEC E4                      DEFB  E4H           ; ↑Y
0CED C4                      DEFB  C4H           ; ↑T
0CEE F7                      DEFB  F7H           ; ↑G
0CEF F6                      DEFB  F6H           ; ↑F
0CF0 C6                      DEFB  C6H           ; ↑V CLR
0CF1 F2                      DEFB  F2H           ; ↑B
0CF2                  ;S3
0CF2 F0                      DEFB  F0H
0CF3 F0                      DEFB  F0H
0CF4 F9                      DEFB  F9H           ; ↑I
0CF5 C5                      DEFB  C5H           ; ↑U HOME
0CF6 FA                      DEFB  FAH           ; ↑J
0CF7 F8                      DEFB  F8H           ; ↑H
0CF8 FE                      DEFB  FEH           ; ↑N
0CF9 F0                      DEFB  F0H
0CFA                  ;S4
0CFA F0                      DEFB  F0H
0CFB F0                      DEFB  F0H
0CFC E1                      DEFB  E1H           ; ↑P
0CFD FF                      DEFB  FFH           ; ↑O
0CFE FC                      DEFB  FCH           ; ↑L
0CFF FB                      DEFB  FBH           ; ↑K
0D00 F0                      DEFB  F0H
0D01 FD                      DEFB  FDH           ; ↑M
0D02                  ;S5
0D02 EF                      DEFB  EFH           ; ↑^
0D03 F4                      DEFB  F4H           ; ↑-
0D04 E6                      DEFB  E6H           ; ↑[
0D05 CC                      DEFB  CCH           ; ↑@ REVERSE
0D06 F0                      DEFB  F0H
```

```
0D07 F0                  DEFB  F0H
0D08 F0                  DEFB  F0H
0D09 F0                  DEFB  F0H
0D0A              ;S6
0D0A F0                  DEFB  F0H
0D0B EB                  DEFB  EBH            ; ↑\
0D0C F0                  DEFB  F0H
0D0D F0                  DEFB  F0H
0D0E F0                  DEFB  F0H
0D0F EE                  DEFB  EEH            ; ↑]
0D10 F0                  DEFB  F0H
0D11              ;
0D11              ;
0D11              ;
0D11              ;  BREAK KEY CHECK  ;
0D11              ;     AND SHIFT,GRPH,CTRL KEY CHECK
0D11              ;
0D11              ;  EXIT  BREAK ON  : ZERO = 1
0D11              ;          BREAK OFF : ZERO = 0
0D11              ;        BREAK OFF: ZERO = 0
0D11              ;        NOT KEY  :  CY=0
0D11              ;        KEY IN   :  CY=1
0D11              ;         A D6=1  :  SHIFT ON
0D11              ;              =0 :        OFF
0D11              ;            D5=1 :  CTRL  ON
0D11              ;              =0 :        OFF
0D11              ;            D4=1 :  GRPH  ON
0D11              ;              =0 :        OFF
0D11              ;
0D11              ?BRK:  ENT
0D11 3EF0                LD    A,F0           ; LINE 0 SWEEP
0D13 3200E0              LD    (KEYPA),A
0D16 00                  NOP
0D17 3A01E0              LD    A,(KEYPB)
0D1A B7                  OR    A
0D1B 17                  RLA
0D1C 3019                JR    NC,?BRK1       ;CTRL?
0D1E 1F                  RRA
0D1F 1F                  RRA
0D20 3005                JR    NC,?BRK2       ; SHIFT?
0D22 1F                  RRA
0D23 3006                JR    NC,?BRK3       ; GRPH ?
0D25 3F                  CCF                  ; NOT KEY
0D26 C9                  RET
0D27 3E40         ?BRK2: LD    A,40H          ; A D6=1 SHIFT ON
0D29 37                  SCF
0D2A C9                  RET
0D2B 3A6E11       ?BRK3: LD    A,(KDATW)
0D2E 3E01                LD    A,1
0D30 326E11              LD    (KDATW),A
0D33 3E10                LD    A,10H          ; A:D4=1 GRPH
0D35 37                  SCF
0D36 C9                  RET
0D37 E602         ?BRK1: AND   02H            ; SHIFT ?
0D39 C8                  RET   Z
0D3A 3E20                LD    A,20H          ; A:D5=1  CTRL
0D3C 37                  SCF
0D3D C9                  RET
0D3E              ;;
0D3E              ;  1 BIT WRITE
```



```
0D3E              ;
0D3E F5           SHORT: PUSH  AF
0D3F 3E03                LD    A,03H
0D41 3203E0              LD    (CSTPT),A
0D44 CD5907              CALL  DLY1
0D47 CD5907              CALL  DLY1
0D4A 3E02                LD    A,02H
0D4C 3203E0              LD    (CSTPT),A
0D4F CD5907              CALL  DLY1
0D52 CD5907              CALL  DLY1
0D55 F1                  POP   AF
0D56 C9                  RET
0D57              ;
0D57 F5           LONG:  PUSH  AF
0D58 3E03                LD    A,03H
0D5A 3203E0              LD    (CSTPT),A
0D5D CD5907              CALL  DLY1
0D60 CD5907              CALL  DLY1
0D63 CD5907              CALL  DLY1
0D66 CD5907              CALL  DLY1
0D69 3E02                LD    A,02H
0D6B 3203E0              LD    (CSTPT),A
0D6E CD5907              CALL  DLY1
0D71 CD5907              CALL  DLY1
0D74 CD5907              CALL  DLY1
0D77 CD6007              CALL  DLY2
0D7A F1                  POP   AF
0D7B C9                  RET
0D7C              ;
0D7C              ;;
0D7C              ;    @DSP PUCH
0D7C              ;
0D7C FE08         ?DSPA: CP    08H
0D7E 2810                JR    Z,DSP03
0D80 CB0E                RRC   (HL)
0D82 10FC                DJNZ  -2
0D84 CBC6                SET   0,(HL)
0D86 CB8E                RES   1,(HL)
0D88 47                  LD    B,A
0D89 CB06                RLC   (HL)
0D8B 10FC                DJNZ  -2
0D8D C37B0E       DSP04: JP    CURSR
0D90              ;
0D90 23           DSP03: INC   HL
0D91 CBC6                SET   0,(HL)
0D93 CB8E                RES   1,(HL)
0D95 18F6                JR    DSP04
0D97              ;
0D97 CBFE         DSP02: SET   7,(HL)
0D99 23                  INC   HL
0D9A CB86                RES   0,(HL)
0D9C 18EF                JR    DSP04
0D9E              ;
0D9E              ;   PRESS PLAY MESSAGE
0D9E              ;
0D9E 7F20         MSG#1: DEFW  207FH
0DA0 504C4159     MSG#2: DEFM  'PLAY'
0DA4 0D                  DEFB  0DH
0DA5              ;
0DA5              ;
```

```
**  Z80 ASSEMBLER SB-7201  <MZ-80A,MONITOR>  PAGE 49           09/04/81

0DA5                 ;
0DA6                          ORG   0DA6H            ;?BLNK    15
0DA6                 ;
0DA6                 ;
0DA6                 ;  V-BLANK CHECK  ;
0DA6                 ;
0DA6 C9              ?BLNK:   RET
0DA7                 ;
0DA7 C5              DLY12:   PUSH  BC
0DA8 0623                     LD    B,35             ; 12 M SEC DELAY
0DAA CDA209                   CALL  DLY3
0DAD 10FB                     DJNZ  -3
0DAF C1                       POP   BC
0DB0 C9                       RET
0DB1                 ;
0DB1                 ;   BELL DATA
0DB1 D7              ?BELD:   DEFB  D7H
0DB2 4130                     DEFM  'A0'
0DB4 0D                       DEFB  0DH
0DB5                 ;
0DB5                          ORG   0DB5H            ;?DSP    39
0DB5                 ;
0DB5                 ;
0DB5                 ;
0DB5                 ;  DISPLAY ON POINTER  ;
0DB5                 ;
0DB5                 ;    ACC = DISPLAY CODE
0DB5                 ;  EXCEPT F0
0DB5 F5              ?DSP:    PUSH  AF
0DB6 C5                       PUSH  BC
0DB7 D5                       PUSH  DE
0DB8 E5                       PUSH  HL
0DB9 47                       LD    B,A
0DBA CDB10F          DSP01:   CALL  ?PONT
0DBD 70                       LD    (HL),B
0DBE 2A7111                   LD    HL,(DSPXY)
0DC1 7D                       LD    A,L
0DC2 FE27                     CP    +39
0DC4 20C7                     JR    NZ,DSP04
0DC6 CD2B0A                   CALL  .MANG
0DC9 38C2                     JR    C,DSP04
0DCB 3A9111                   LD    A,(SPAGE)
0DCE B7                       OR    A
0DCF C29D03                   JP    NZ,.DSP03
0DD2 EB                       EX    DE,HL
0DD3 78                       LD    A,B
0DD4 FE07                     CP    07H
0DD6 28BF                     JR    Z,DSP02
0DD8 18A2                     JR    ?DSPA
0DDA                 ;
0DDA                 ;
0DDA                 ;
0DDC                          ORG   0DDCH            ;?DPCT
0DDC                 ;
0DDC                 ;
0DDC                 ;  DISPLAY CONTROL   ;
0DDC                 ;
0DDC                 ;  ACC = CONTROL CODE
0DDC                 ;
0DDC F5              ?DPCT:   PUSH  AF
```

```
**  Z80 ASSEMBLER SB-7201  <MZ-80A,MONITOR>  PAGE 50           09/04/81

0DDD C5                       PUSH  BC
0DDE D5                       PUSH  DE
0DDF E5                       PUSH  HL
0DE0 47                       LD    B,A
0DE1 E6F0                     AND   F0H
0DE3 FEC0                     CP    C0H
0DE5 C2E50E                   JP    NZ,?RSTR
0DE8 A8                       XOR   B
0DE9 07                       RLCA
0DEA 4F                       LD    C,A
0DEB 0600                     LD    B,+0
0DED 21FF0D                   LD    HL,CTBL
0DF0 3A9111                   LD    A,(SPAGE)
0DF3 B7                       OR    A
0DF4 2803                     JR    Z,+5
0DF6 216801                   LD    HL,.CTBL         ; PAGE MODE1
0DF9 09                       ADD   HL,BC
0DFA 5E                       LD    E,(HL)
0DFB 23                       INC   HL
0DFC 56                       LD    D,(HL)
0DFD EB                       EX    DE,HL
0DFE E9                       JP    (HL)
0DFF                 ;
0DFF 1F0E            CTBL:    DEFW  SCROL
0E01 5D0E                     DEFW  CURSD
0E03 6E0E                     DEFW  CURSU
0E05 7B0E                     DEFW  CURSR
0E07 950E                     DEFW  CURSL
0E09 FB03                     DEFW  HOME
0E0B B30E                     DEFW  CLRS
0E0D F20E                     DEFW  DEL
0E0F 2D0F                     DEFW  INST
0E11 E10E                     DEFW  ALPHA
0E13 EE0E                     DEFW  KANA
0E15 E50E                     DEFW  ?RSTR
0E17 170A                     DEFW  REV
0E19 730F                     DEFW  CR
0E1B E805                     DEFW  ROLUP
0E1D 590F                     DEFW  ROLD
0E1F                 ;
0E1F                 ;
0E1F                 ;
0E1F 217A11          SCROL:   LD    HL,PBIAS
0E22 0E05                     LD    C,5
0E24 3A7F11                   LD    A,(ROLEND)
0E27 81                       ADD   A,C
0E28 327F11                   LD    (ROLEND),A
0E2B 3A7B11                   LD    A,(ROLTOP)
0E2E 81                       ADD   A,C
0E2F 327B11                   LD    (ROLTOP),A
0E32 79              SCROL1:  LD    A,C
0E33 86                       ADD   A,(HL)
0E34 77                       LD    (HL),A
0E35 CDC709                   CALL  PAGE
0E38 2A7D11                   LD    HL,(PAGETP)
0E3B 11E803                   LD    DE,1000
0E3E 19                       ADD   HL,DE            ; HL=PAGETOP+1000
0E3F 0628                     LD    B,40
0E41 AF                       XOR   A
0E42 CB9C            SCROL2:  RES   3,H
```

```
**  Z80 ASSEMBLER SB-7201  <MZ-80A.MONITOR>  PAGE 51          09/04/81

0E44 77                   LD    (HL),A
0E45 23                   INC   HL
0E46 10FA                 DJNZ  SCROL2
0E48 3A7A11               LD    A,(PBIAS)       ; SCROL
0E4B 6F                   LD    L,A
0E4C 26E2                 LD    H,E2H
0E4E 7E                   LD    A,(HL)
0E4F 217911               LD    HL,MANGE        ; COLOUMN MANAG. END
0E52 B7                   OR    A
0E53 0607                 LD    B,7
0E55 CB1E                 RR    (HL)
0E57 2B                   DEC   HL
0E58 10FB                 DJNZ  -3
0E5A C3E50E               JP    ?RSTR
0E5D              ;
0E5D              ;
0E5D 2A7111       CURSD:  LD    HL,(DSPXY)
0E60 7C                   LD    A,H
0E61 FE18                 CP    +24
0E63 282E                 JR    Z,CURS4
0E65 24                   INC   H
0E66 CD8302       CURS1:  CALL  MGP.I
0E69 227111       CURS3:  LD    (DSPXY),HL
0E6C 1877                 JR    ?RSTR
0E6E              ;
0E6E 2A7111       CURSU:  LD    HL,(DSPXY)
0E71 7C                   LD    A,H
0E72 B7                   OR    A
0E73 2835                 JR    Z,CURS5
0E75 25                   DEC   H
0E76 CD9D02       CURSU1: CALL  MGP.D
0E79 18EE                 JR    CURS3
0E7B              ;
0E7B 2A7111       CURSR:  LD    HL,(DSPXY)
0E7E 7D                   LD    A,L
0E7F FE27                 CP    +39
0E81 3003                 JR    NC,CURS2
0E83 2C                   INC   L
0E84 18E3                 JR    CURS3
0E86 2E00         CURS2:  LD    L,+0
0E88 24                   INC   H
0E89 7C                   LD    A,H
0E8A FE19                 CP    +25
0E8C 38D8                 JR    C,CURS1
0E8E 2618                 LD    H,+24
0E90 227111               LD    (DSPXY),HL
0E93 1842         CURS4:  JR    CURS6
0E95              ;
0E95 2A7111       CURSL:  LD    HL,(DSPXY)
0E98 7D                   LD    A,L
0E99 B7                   OR    A
0E9A 2803                 JR    Z,+5
0E9C 2D                   DEC   L
0E9D 18CA                 JR    CURS3
0E9F 2E27                 LD    L,+39
0EA1 25                   DEC   H
0EA2 F2760E               JP    P,CURSU1
0EA5 2600                 LD    H,0
0EA7 227111               LD    (DSPXY),HL
0EAA 3A9111       CURS5:  LD    A,(SPAGE)
```

```
**  Z80 ASSEMBLER SB-7201  <MZ-80A.MONITOR>  PAGE 52          09/04/81

0EAD B7                   OR    A
0EAE 2035                 JR    NZ,?RSTR
0EB0 C3590F               JP    ROLD
0EB3              ;
0EB3 217311       CLRS:   LD    HL,MANG
0EB6 061B                 LD    B,27
0EB8 CDD80F               CALL  ?CLER
0EBB 2100D0               LD    HL,D000H        ; SCRN TOP
0EBE E5                   PUSH  HL
0EBF CDE209               CALL  #CLR08
0EC2 E1                   POP   HL
0EC3 3A9111               LD    A,(SPAGE)
0EC6 B7                   OR    A
0EC7 2008                 JR    NZ,CLRS1
0EC9 227D11               LD    (PAGETP),HL
0ECC 3E7D                 LD    A,7DH
0ECE 327F11               LD    (ROLEND),A
0ED1 3A00E2       CLRS1:  LD    A,(E200H)
0ED4 C30904       HOM00:  JP    HOM0
0ED7              ;
0ED7              ;  CURS PART2 (CURS6)
0ED7              ;
0ED7 3A9111       CURS6:  LD    A,(SPAGE)
0EDA B7                   OR    A
0EDB C23D01               JP    NZ,.SCROL
0EDE C39F0F               JP    ROLU
0EE1              ;
0EE1              ;
0EE1              ;
0EE1                      ORG   0EE1H           ;ALPHA  13
0EE1              ;
0EE1 AF           ALPHA:  XOR   A
0EE2 327011       ALPH1:  LD    (KANAF),A
0EE5              ;
0EE5              ;
0EE5              ;  RESTORE  ;
0EE5              ;
0EE5              ?RSTR:  ENT
0EE5 E1                   POP   HL
0EE6 D1           ?RSTR1: POP   DE
0EE7 C1                   POP   BC
0EE8 F1                   POP   AF
0EE9 C9                   RET
0EEA              ;
0EEA              ;  MONITOR WORK AREA  ;
0EEA              ;
0EEA  P           SCRN:   EQU   D000H
0EEA  P           KANST:  EQU   E003H           ; KANA STATUS PORT
0EEA              ;
0EEA              ;
0EEA              ;
0EEE                      ORG   0EEEH           ;KANA   195
0EEE              ;
0EEE 3E01         KANA:   LD    A,+1
0EF0 18F0                 JR    ALPH1
0EF2              ;
0EF2              ;
0EF2 2A7111       DEL:    LD    HL,(DSPXY)
0EF5 7C                   LD    A,H             ; HOME ?
0EF6 B5                   OR    L
```

```
**  Z80 ASSEMBLER SB-7201  <MZ-80A.MONITOR>  PAGE 53          09/04/81

0EF7 28EC                  JR     Z,?RSTR
0EF9 7D                    LD     A,L
0EFA B7                    OR     A
0EFB 200D                  JR     NZ,DEL1          ; LEFT SIDE ?
0EFD CD2B0A                CALL   .MANG
0F00 3808                  JR     C,DEL1
0F02 CDB10F                CALL   ?PONT
0F05 2B                    DEC    HL
0F06 3600                  LD     (HL),+0
0F08 188B                  JR     CURSL
0F0A CD2B0A       DEL1:    CALL   .MANG
0F0D 0F                    RRCA
0F0E 3E28                  LD     A,40
0F10 3001                  JR     NC,+3
0F12 07                    RLCA                    ; ACC=80
0F13 95                    SUB    L
0F14 47                    LD     B,A
0F15 CDB10F                CALL   ?PONT
0F18 E5                    PUSH   HL
0F19 D1                    POP    DE
0F1A 1B                    DEC    DE
0F1B CBE2                  SET    4,D
0F1D CB9C         DEL2:    RES    3,H
0F1F CB9A                  RES    3,D
0F21 7E                    LD     A,(HL)
0F22 12                    LD     (DE),A
0F23 23                    INC    HL
0F24 13                    INC    DE
0F25 10F6                  DJNZ   DEL2
0F27 2B                    DEC    HL
0F28 3600                  LD     (HL),+0
0F2A C3950E                JP     CURSL
0F2D              ;
0F2D CD2B0A       INST:    CALL   .MANG
0F30 0F                    RRCA
0F31 2E27                  LD     L,+39
0F33 7D                    LD     A,L
0F34 3001                  JR     NC,+3
0F36 24                    INC    H
0F37 CDB40F                CALL   ?PNT1
0F3A E5                    PUSH   HL
0F3B 2A7111                LD     HL,(DSPXY)
0F3E 3002                  JR     NC,+4
0F40 3E4F                  LD     A,+79
0F42 95                    SUB    L
0F43 47                    LD     B,A
0F44 D1                    POP    DE
0F45 1A                    LD     A,(DE)
0F46 B7                    OR     A
0F47 209C                  JR     NZ,?RSTR
0F49 CDB10F                CALL   ?PONT
0F4C 7E                    LD     A,(HL)
0F4D 3600                  LD     (HL),0
0F4F 23           INST1:   INC    HL
0F50 CB9C                  RES    3,H
0F52 5E                    LD     E,(HL)
0F53 77                    LD     (HL),A
0F54 7B                    LD     A,E
0F55 10F8                  DJNZ   INST1
0F57 188C                  JR     ?RSTR
```

```
**  Z80 ASSEMBLER SB-7201  <MZ-80A.MONITOR>  PAGE 54          09/04/81

0F59              ;
0F59 217A11       ROLD:    LD     HL,PBIAS
0F5C 3A7B11                LD     A,(ROLTOP)
0F5F BE                    CP     (HL)
0F60 2883                  JR     Z,?RSTR
0F62 CD9D02                CALL   MGP.D
0F65 7E                    LD     A,(HL)
0F66 D605                  SUB    5
0F68 77           ROL2:    LD     (HL),A
0F69 6F                    LD     L,A
0F6A 26E2                  LD     H,E2H
0F6C 7E                    LD     A,(HL)
0F6D CDC709                CALL   PAGE
0F70 C3E50E                JP     ?RSTR
0F73
0F73              ;
0F73 CD2B0A       CR:      CALL   .MANG
0F76 0F                    RRCA
0F77 D2860E                JP     NC,CURS2
0F7A 2E00                  LD     L,0
0F7C 24                    INC    H
0F7D 7C                    LD     A,H
0F7E FE18                  CP     +24
0F80 2817                  JR     Z,CR3
0F82 3007                  JR     NC,CR2
0F84 CD8302                CALL   MGP.I
0F87 24                    INC    H
0F88 C3660E                JP     CURS1
0F8B 25           CR2:     DEC    H
0F8C 227111                LD     (DSPXY),HL
0F8F 219F0F                LD     HL,ROLU
0F92 E5                    PUSH   HL
0F93 F5                    PUSH   AF
0F94 C5                    PUSH   BC
0F95 D5                    PUSH   DE
0F96 CD9F0F                CALL   ROLU
0F99 227111       CR3:     LD     (DSPXY),HL
0F9C CD8302                CALL   MGP.I
0F9F              ;
0F9F 217A11       ROLU:    LD     HL,PBIAS
0FA2 3A7F11                LD     A,(ROLEND)
0FA5 BE                    CP     (HL)
0FA6 CA1F0E                JP     Z,SCROL
0FA9 CD8302       ROLU1:   CALL   MGP.I
0FAC 7E                    LD     A,(HL)
0FAD C605                  ADD    A,5
0FAF 18B7                  JR     ROL2
0FB1              ;
0FB1                       ORG    0FB1H            ;?PONT    3
0FB1              ;
0FB1              ;
0FB1              ;  COMPUTE POINT ADR .  ;
0FB1              ;
0FB1              ;    HL = SCREEN CORDINATE
0FB1              ;    EXIT
0FB1              ;    HL = POINT ADR. ON SCREEN
0FB1              ;
0FB1 2A7111       ?PONT:   LD     HL,(DSPXY)
0FB4              ;
0FB4                       ORG    0FB4H            ;?PNT1
```

```
** Z80 ASSEMBLER SB-7201  <MZ-80A,MONITOR>  PAGE 55          09/04/81

0FB4                ;
0FB4 F5             ?PNT1:  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 3A9111                 LD    A,(SPAGE)
0FC2 B7                     OR    A
0FC3 2005                   JR    NZ,?PNT2
0FC5 2A7D11                 LD    HL,(PAGETP)
0FC8 ED52                   SBC   HL,DE
0FCA 19             ?PNT2:  ADD   HL,DE
0FCB 05                     DEC   B
0FCC F2CA0F                 JP    P,-2
0FCF 0600                   LD    B,+0
0FD1 09                     ADD   HL,BC
0FD2 CB9C                   RES   3,H
0FD4 D1                     POP   DE
0FD5 C1                     POP   BC
0FD6 F1                     POP   AF
0FD7 C9                     RET
0FD8                ;
0FD8                ;
0FD8                        ORG   0FD8H           ;?CLER    40
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 C5             GAPCK:  PUSH  BC
0FE3 D5                     PUSH  DE
0FE4 E5                     PUSH  HL
0FE5 0101E0                 LD    BC,KEYPB
0FE8 1102E0                 LD    DE,CSTR
0FEB 2664           GAPCK1: LD    H,100
0FED CD0106         GAPCK2: CALL  EDGE
0FF0 380B                   JR    C,GAPCK3
0FF2 CDA209                 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
```

```
** Z80 ASSEMBLER SB-7201  <MZ-80A,MONITOR>  PAGE 56          09/04/81

0FFD C39F06         GAPCK3: JP    RET3
1000                        SKP   H
```

```
**  Z80 ASSEMBLER SB-7201  <MZ-80A,MONITOR>  PAGE 57            09/04/81

1000            ;
1000            ;
1000            ;    MONITOR WORK AREA  ;
1000            ;      (MZ-80A)         ;
1000            ;
1000            ;
1000            ;
10F0                    ORG   10F0H
10F0            SP:     ENT
10F0            IBUFE:  ENT                  ; TAPE BUFFER(128B)
10F0            ATRB:   ENT                  ; ATRIBUTE
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  +92
1164            SWPW:   ENT                  ; SWEEP WORK
1164                    DEFS  +10
116E            KDATW:  ENT                  ; KEY WORK
116E                    DEFS  +2
1170            KANAF:  ENT                  ; KANA FLAG
1170                    DEFS  +1
1171            DSPXY:  ENT                  ; DISPLAY CO-ORDINATES
1171                    DEFS  +2
1173            MANG:   ENT                  ; COLOUMN MANAGEMENT
1173                    DEFS  +6
1179            MANGE:  ENT
1179                    DEFS  +1             ; COLOUMN MANAG. END
117A            PBIAS:  ENT
117A                    DEFS  +1             ; PAGE BIAS
117B            ROLTOP: ENT
117B                    DEFS  +1             ; ROLL TOP BIAS
117C            MGPNT:  ENT
117C                    DEFS  +1             ; COLOUMN MANAG. POINTER
117D            PAGETP: ENT
117D                    DEFS  +2             ; PAGE TOP
117F            ROLEND: ENT
117F                    DEFS  +1             ; ROLL END
1180                    DEFS  +14            ; BIAS
118E            FLASH:  ENT                  ; FLASHING DATA
118E                    DEFS  +1
118F            SFTLK:  ENT                  ; SHIFT LOCK
118F                    DEFS  +1
1190            REVFLG: ENT                  ; REVERSE FLAG
1190                    DEFS  +1
1191            SPAGE:  ENT                  ; PAGE CHENGE
1191                    DEFS  +1
1192            FLSDT:  ENT                  ; CURSOL 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
```

```
**  Z80 ASSEMBLER SB-7201  <MZ-80A,MONITOR>  PAGE 58            09/04/81

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                  ; ONPÜ RATIO
11A1                    DEFS  +2
11A3            BUFER:  ENT                  ; GET LINE BUFFER
11A3                    DEFS  +81
11F4                    END
```

```
**  Z80 ASSEMBLER SB-7201  <MZ-80A,MONITOR>  PAGE 59                09/04/81

#BRK    0BC5  #CLR08  09E2  #CLR8   09E3  #GRP    06E7  .CR     0128
.CR1    013A  .CTBL   0168  .DSP03  039D  .MANG   0A2B  .MANG1  0A4C
.MANG2  03A6  .SCROL  013D  00MSG   09FC  2HE1    0434  2HEX    041F
??KEY   09B3  ?ADCN   0BB9  ?BEL    02E5  ?BELD   0DB1  ?BLNK   0DA6
?BRK    0D11  ?BRK1   0D37  ?BRK2   0D27  ?BRK3   0D2B  ?CLER   0FD8
?CLRFF  0FDB  ?DACN   0BCE  ?DINT   0FDD  ?DPCT   0DDC  ?DSP    0DB5
?DSPA   0D7C  ?ER     00CF  ?FLAS   09FF  ?GET    08B3  ?GETL   07A8
?KEY    08CA  ?KY1    08D6  ?KY10   08EF  ?KY11   08EA  ?KY2    08F2
?KY3    0922  ?KY4    092F  ?KY5    091E  ?KY6    092A  ?KY7    093A
?KY8    08CD  ?KY9    093F  ?LOAD   05F5  ?LTNL   0980  ?MLDY   0188
?MODE   074D  ?MSG    0893  ?MSGX   08A1  ?NL     097B  ?PNT1   0FB4
?PNT2   0FCA  ?PONT   0FB1  ?PRNT   0995  ?PRT    0946  ?PRTS   0993
?PRTT   0984  ?RDD    04EF  ?RDI    04CF  ?RSTR   0EE5  ?RSTR1  0EE6
?SAVE   0263  ?SWEP   0A50  ?TEMP   02EC  ?TMR1   0361  ?TMR2   036B
?TMRD   0344  ?TMS1   0323  ?TMS2   0336  ?TMST   02FA  ?VRFY   0575
?WRD    0470  ?WRI    0436  ALPH1   0EE2  ALPHA   0EE1  AMPM    119B
ASC     03DA  ATBL    0AB5  ATRB    10F0  AUTCK   09F1  AUTO2   0807
AUTO3   07C4  AUTO5   0824  AUTOL   0810  BELL    003E  BRKEY   001E
BUFER   11A3  CHGP1   084B  CHGPA   083E  CHGPK   0841  CKS1    0720
CKS2    072F  CKS3    0733  CKSUM   071A  CLEAR   09E6  CLEAR1  09E8
CLRS    0EB3  CLRS1   0ED1  COMNT   1108  CONT0   E004  CONT1   E005
CONT2   E006  CONTF   E007  CR      0F73  CR2     0F8B  CR3     0F99
CSMDT   1199  CSTPT   E003  CSTR    E002  CTBL    0DFF  CURS1   0E66
CURS2   0E86  CURS3   0E69  CURS4   0E93  CURS5   0EAA  CURS6   0ED7
CURSD   0E5D  CURSL   0E95  CURSR   0E7B  CURSU   0E6E  CURSU1  0E76
DACN1   0BE3  DACN2   0BDF  DACN3   0BE0  DEL     0EF2  DEL1    0F0A
DEL2    0F1D  DLY1    0759  DLY12   0DA7  DLY2    0760  DLY3    09A2
DMT     0857  DPRNT   1194  DSP01   0DBA  DSP02   0D97  DSP03   0D90
DSP04   0D8D  DSPXY   1171  DTADR   1104  EDG1    0607  EDG2    0615
EDGE    0601  EXADR   1106  FD      00C7  FD1     00CE  FD2     00CA
FLAS1   0A12  FLAS2   0A0B  FLAS3   0A0F  FLASH   118E  FLSDT   1192
GAP     077A  GAP1    078E  GAP2    0796  GAP3    079C  GAPCK   0FE2
GAPCK1  0FEB  GAPCK2  0FED  GAPCK3  0FFD  GETKY   001B  GETL    0003
GETL0   07AC  GETL1   07B6  GETL11  081A  GETL2   0803  GETL3   085B
GETL5   0839  GETL6   0865  GETLA   0886  GETLB   0863  GETLC   084E
GETLD   05E1  GETLL   03D5  GETLR   0880  GOTO    00BB  HEX     03F9
HEX1    03F2  HEX2    03F5  HEXJ    03E5  HL1     041D  HLHEX   0410
HOM0    0409  HOM00   0ED4  HOM1    0406  HOME    03FB  IBUFE   10F0
INST    0F2D  INST1   0F4F  KANA    0EEE  KANAF   1170  KANST   E003
KDATW   116E  KEYPA   E000  KEYPB   E001  KEYPC   E002  KEYPF   E003
KSL0    0270  KSL2    09B9  KTBL    0BEA  KTBLC   0CDA  KTBLG   0C6A
KTBLGS  0CA2  KTBLS   0C32  LETNL   0006  LOA0    00DE  LOAD    00D9
LOCK    088B  LONG    0D57  M#TBL   0241  MANG    1173  MANGE   1179
MELDY   0030  MGP.D   029D  MGP.I   0283  MGP0    028F  MGPNT   117C
MLD1    0192  MLD2    01C6  MLD3    01CE  MLD4    01D2  MLD5    01D5
MLDS1   02C4  MLDSP   02BE  MLDST   02AB  MONIT   0000  MOT1    06A8
MOT2    06AF  MOT4    06B9  MOT5    06D8  MOT7    06B7  MOT8    06D0
MOT9    06D7  MOTOR   06A3  MSG     0015  MSG#1   0D9E  MSG#2   0DA0
MSG#3   06F4  MSG#7   0467  MSG1    0896  MSG?2   00F7  MSG?3   0100
MSGE1   0118  MSGX    0018  MSGX1   08A4  MSGX2   08A7  MST1    0705
MST2    070C  MST3    0717  MSTA    0044  MSTOP   0700  MSTP    0047
MTBL    0229  NAME    10F1  NL      0009  NOADD   03E2  OCTV    11A0
ONP1    01E0  ONP2    01ED  ONP3    09AB  ONPU    01DD  ONTYO   119F
OPTBL   0259  PAGE    09C7  PAGETP  117D  PBIAS   117A  PRNT    0012
PRNT2   0967  PRNT3   096C  PRNT4   096F  PRNT5   0955  PRNTS   000C
PRNTT   000F  PRTHL   03BA  PRTHX   03C3  RATIO   11A1  RBY1    0630
RBY2    0649  RBY3    0654  RBYTE   0624  RD1     04DD  RDDAT   002A
RDINF   0027  RET1    04CB  RET2    0552  RET3    069F  REV     0A17
REV1    0A25  REVFLG  1190  ROL2    0F68  ROLD    0F59  ROLEND  117F
ROLTOP  117B  ROLU    0F9F  ROLU1   0FA9  ROLUP   05E8  RTAPE   0505
```

```
**  Z80 ASSEMBLER SB-7201  <MZ-80A,MONITOR>  PAGE 60                09/04/81

RTP1    050A  RTP2    0510  RTP3    052A  RTP4    0552  RTP5    0563
RTP6    0570  RTP7    056C  RTP8    0551  RTP9    0572  RYTHM   02C8
SCRN    D000  SCROL   0E1F  SCROL1  0E32  SCROL2  0E42  SFTLK   118F
SG      00C1  SHORT   0D3E  SIZE    1102  SP      10F0  SPAGE   1191
SS      0089  ST1     0095  ST2     00A3  START   004A  STRGF   1193
SUMDT   1197  SUNDG   E008  SWEP0   0A76  SWEP01  0A74  SWEP11  0A8E
SWEP2   0AAC  SWEP3   0A92  SWEP4   073E  SWEP6   0A62  SWEP7   0AA5
SWEP8   0AA0  SWEP9   0A8A  SWPW    1164  SWRK    119D  TEMP    E008
TEMPW   119E  TIMF0   119C  TIMIN   0379  TIMRD   003B  TIMST   0033
TM1     0673  TM2     0676  TM3     0689  TM4     069F  TMARK   0658
TMCNT   1195  TVF1    05A0  TVF2    05A6  TVF3    05BA  TVRFY   059B
VERFY   002D  VGOFF   0756  WBY1    076D  WBYTE   0767  WRDAT   0024
WRI1    0444  WRI2    045E  WRI3    0464  WRINF   0021  WTAP1   048F
WTAP2   049E  WTAP3   04CB  WTAPE   0485  XTEMP   0041
```

Hardware Configuration of the MZ-80A

Chapter

3

3.1 The MZ-80A system configuration

Figure 3.1 shows the standard system configuration of the MZ-80A personal computer.

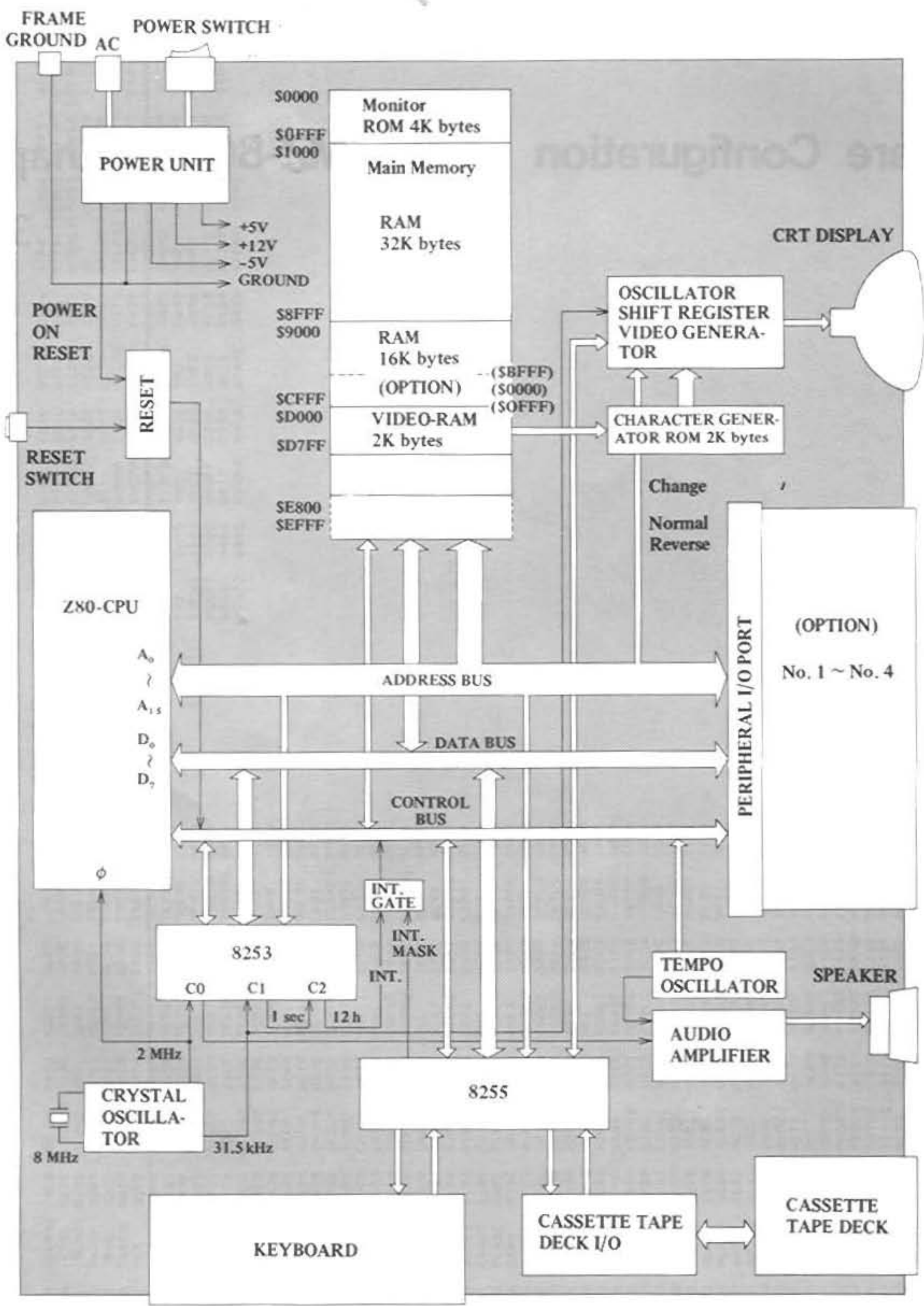


Figure 3.1 MZ-80A System Diagram

As is shown in the figure, a Z80 (Sharp LH0080) is used as the CPU, and is operated with a clock of 2 MHz. The CPU is reset when the power is turned on or when the reset switch is manually operated. The memory configuration corresponding to address buses \$0000-\$FFFF is as follows.

\$0000-\$0FFF is used for the monitor program (ROM); the large 48 K byte space from \$1000-\$CFFF is used as main memory (memory from \$9000-\$CFFF is optional); addresses from \$D000 on are used for video RAM, floppy control, and memory mapped I/O.

The keyboard and cassette tape deck are controlled by means of programmable peripheral interface 8255. Further, a rectangular audio wave generated by the output port of counter 1 of programmable interval timer 8253 is input to the sound generator, which outputs sound to the speaker. The two counters other than this IC serve as internal clocks for the MZ-80A.

Table 3.1 shows the configuration of MZ-80A memory mapped I/O.

Table 3.1 Assignment of memory mapped I/O.

Address	Memory Read	Memory Write	Device
\$E000		D_7 : Resets cursor timer $D_3 \sim D_0$: Key strobe	8255
\$E001	$D_7 \sim D_0$: Key data		
\$E002	D_7 : $\overline{\text{V-Blank}}$ D_6 : Status of cursor timer D_5 : Read data (cassette) D_4 : READ/WRITE status (cassette)	D_3 : Motor ON/OFF (cassette) D_2 : Masking of timer interrupt D_1 : Write data (cassette) D_0 : $\overline{\text{V-Gate}}$	
\$E003		Mode control	
\$E004		Setting of counter #0	8253
\$E005	Reading of counter #1	Setting of counter #1	
\$E006	Reading of counter #2	Setting of counter #2	
\$E007		Mode control	
\$E008	D_7 : Status of tempo timer D_0 : H-Blank	D_0 : Sound ON/OFF	
\$E00C	Memory swap		
\$E010	Resets memory swap		
\$E014	Normal (CRT display)		
\$E015	Reverse (CRT display)		
\$E200 ~ \$E2FF	Roll up/roll down		

3.1.1 Memory configurations

The memory map for the MZ-80A is shown in Figure 3.2. The screened parts of the figure indicate user area, and the 32 K bytes of main memory RAM are the standard package. The remaining 16 K bytes of main memory RAM area optional, and can be installed in the RAM socket provided on the CPU board.

The 4 K bytes of main memory area which are indicated by the dark screening can be used for swapping the address spaces used by the MONITOR ROM. The left side of the figure shows the memory map under normal conditions, while the right side shows the memory map when the MONITOR ROM has been swapped. As is shown in Table 3.1, memory swaps are performed under control of memory mapped I/O by executing memory read instructions such as the following.

To place the memory in the state shown on the right . . . . LD A, (E00CH)

To place the memory in the state shown on the left . . . . LD A, (E010H)

The memory configuration shown on the right is especailly effective when the system programs used start at address $0000 and when the system programs utilized make active use of interrupt processing.

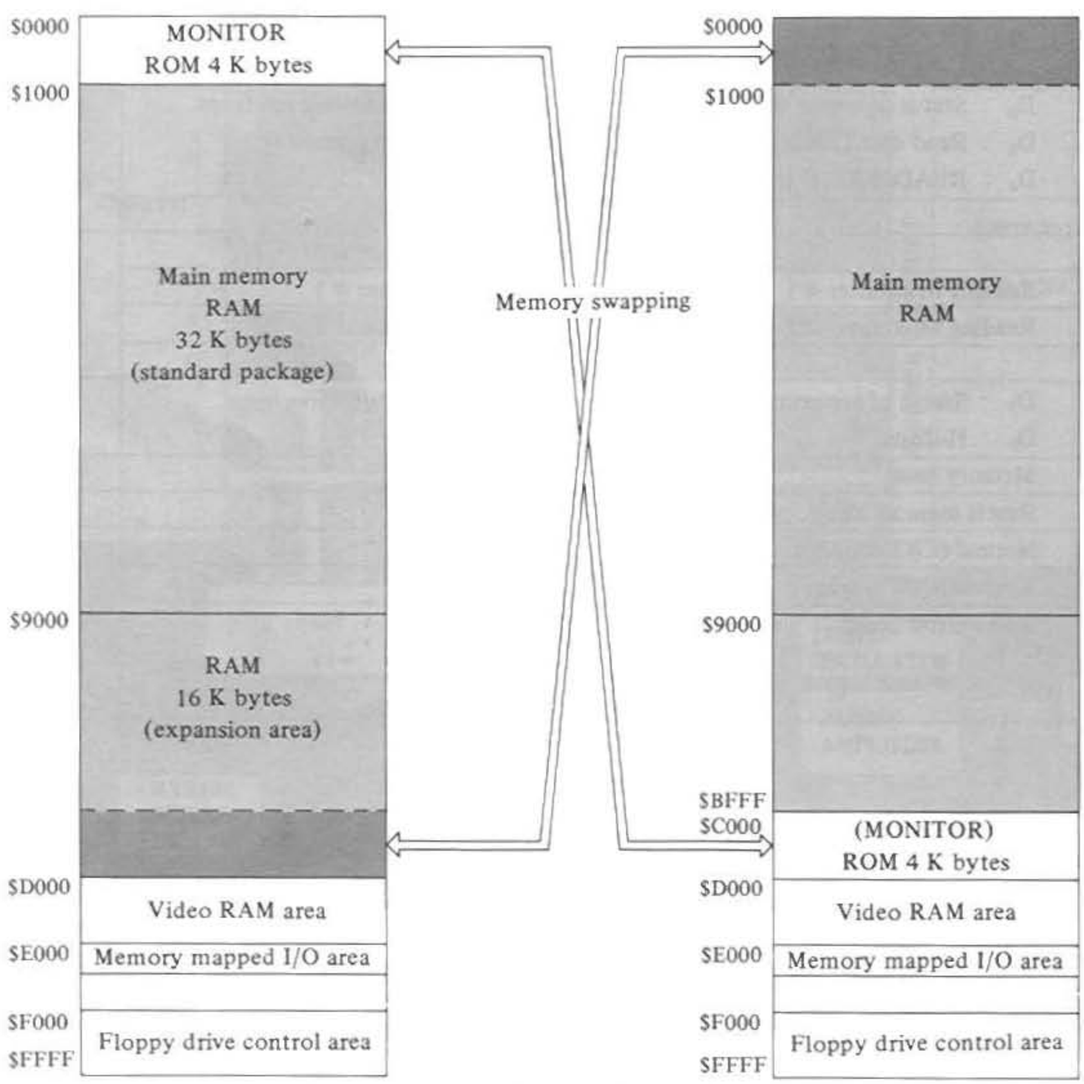


Figure 3.2 Memory maps for normal state and memory swap state.

When the memory is swapped, the 4 K ROM occupies the address space from $C000-$CFFF; however, the monitor program is ineffective in this condition. If necessary, it is possible to remove the monitor ROM from the socket on the CPU board and replace it with another user ROM which has been programmed to allow operation in the space from $C000-$CFFF. In such cases, use ROM 2732, which is the same as the monitor ROM. Also, if it is necessary to alter part of the monitor program for use, it can be modified by block-transferring it from $C000-$CFFF to $0000-$0FFF and making the changes in RAM.

The 2 K byte area from $D000-$D7FF is assigned to video RAM. This area is the standard package which is used for the MZ-80A main unit CRT display.

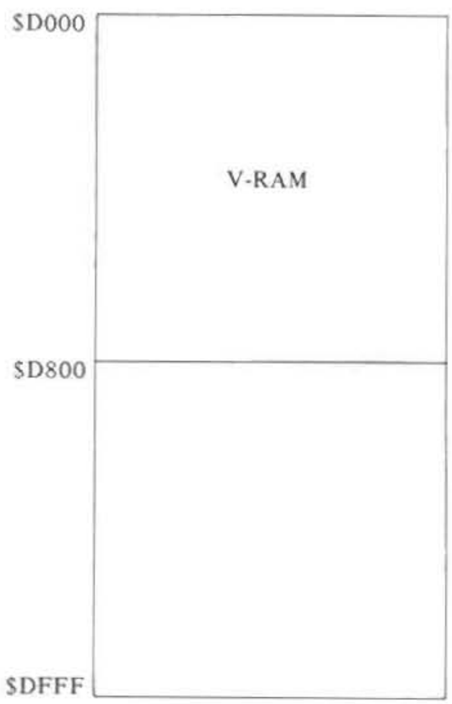


Figure 3.3 MZ-80A video RAM area.

Although up to 1,000 characters (40 characters × 25 lines) can be displayed on the MZ-80A main unit CRT screen, twice this amount of memory area is provided in video RAM. This makes it possible to roll the screen displayed up or down.

Upon system reset, data is written into video RAM starting at address $D000, and when more than 50 lines of data are written – that is, when data has been written into the area from $D000 through $D7CF – $D028 through $D7F7 become the actual video RAM area. When more data is written and one line scrolled, the area from $D050-$D7FF becomes the video RAM area, followed by the area from $D000-$D020. Thus, the video RAM is used in an anchored configuration. Figure 3.4 shows the relationship between video RAM and the display for the 2 K byte video RAM area when its first K byte, its middle K byte, or its last K byte is displayed on the CRT screen. However, in this example, the actual video RAM area capable of displaying data on the CRT screen is the 2000 bytes starting at $D1E0.

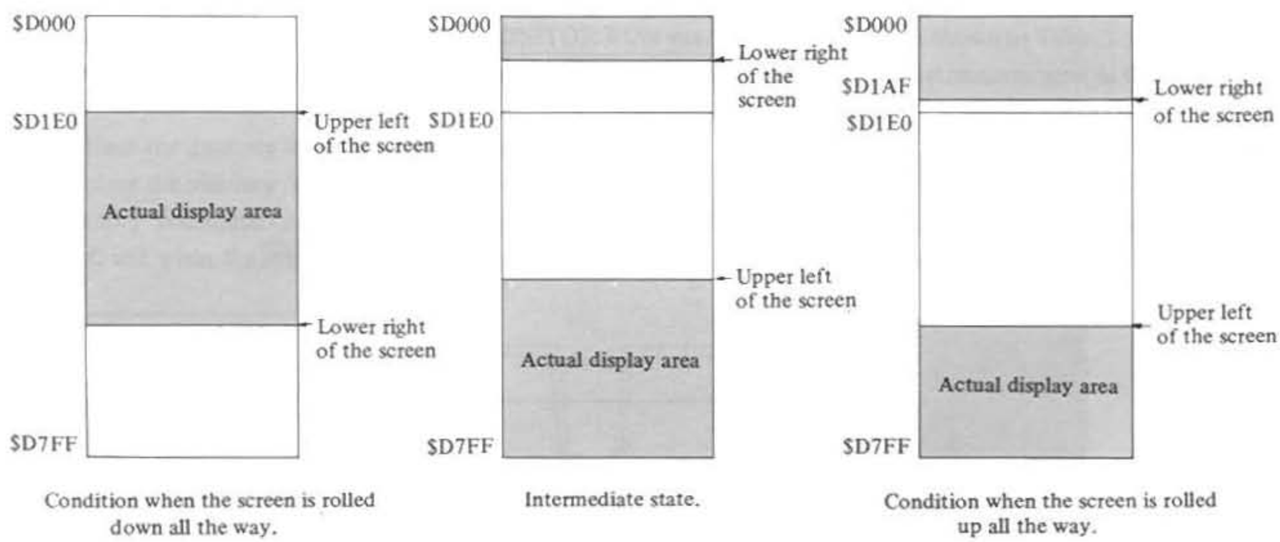


Figure 3.4 Relationship between video RAM and the area actually displayed.

Memory mapped I/O address $E200-$E2FF are used for rolling the display up and down. When a memory read instruction is executed against address $E200 (such as LD A, (E200H)), the CRT display is rolled all the way down. When such an instruction is executed against address $E2FF, the CRT display is rolled all the way up. The lower bytes of these addresses from 00H-FFH can be freely used to roll the screen up or down in 8-character units.

3.1.2 Key scanning system

The relationship between strobe signals and bit data during keyboard scanning is shown in Figure 3.6.

Strobe signals are delivered to four terminals (PA_3, PA_2, PA_1, PA_0) of the 8255, fed into BCD-to-decimal decoder/driver 74145, then delivered to 10 keyboard strobe input terminals. Keys are discriminated by strobe signals and key data.

For instance when the strobe is '2H' and the key data is 'FBH', it indicates that the [S] key is being pressed.

10 keyboard strobe input terminals

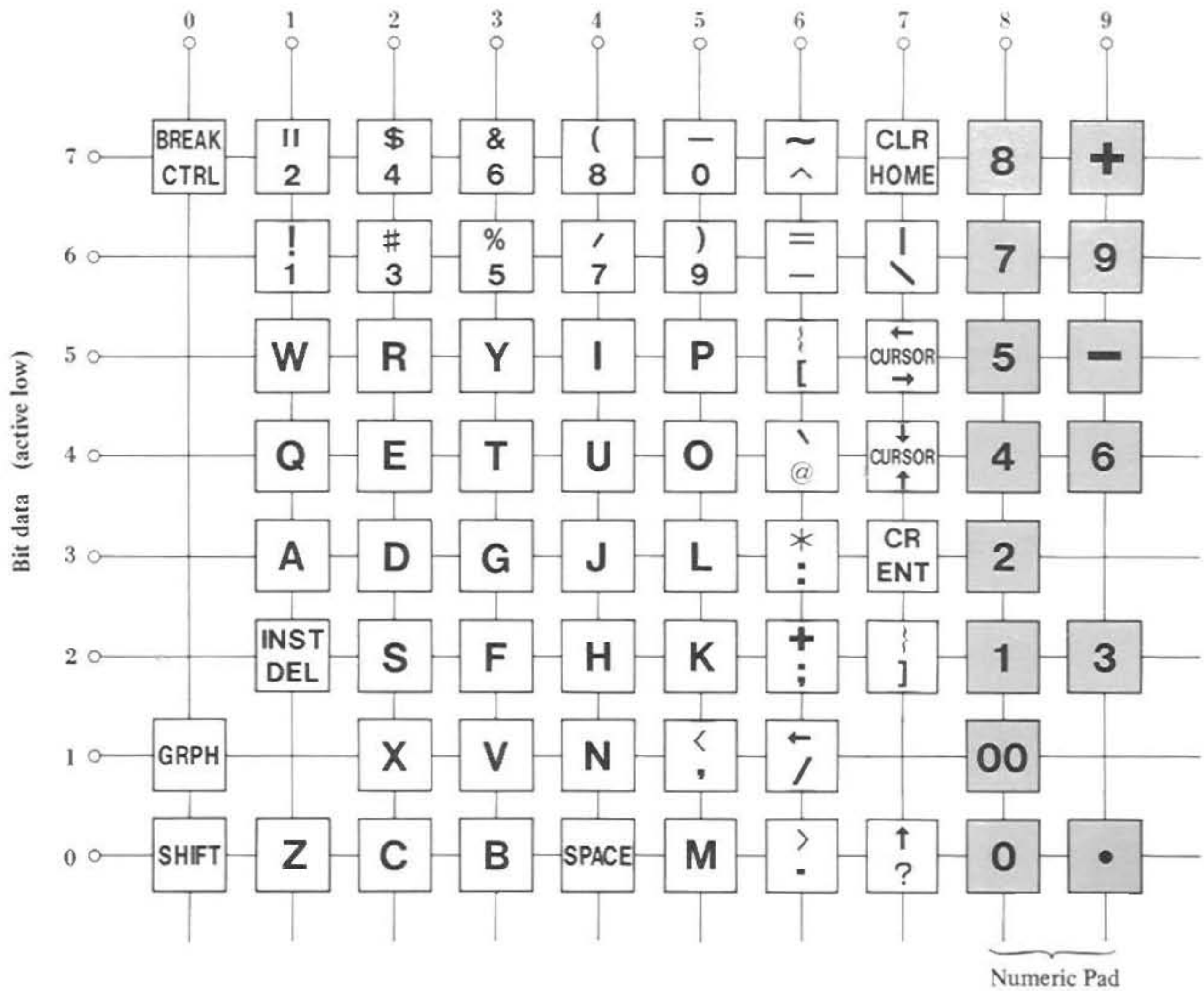


Figure 3.6 Key scanning strobe signal and bit data.

3.2 The MZ-80A circuit diagram

This section includes MZ-80A CPU board circuit diagrams for reference. These diagrams are arranged as follows:

(1) CPU board, block 1 : CPU signal system
(2) CPU board, block 2
(3) CPU board, block 3 : RAM signal system
(4) CPU board, block 4 : 8255 and 8253 signal system
(5) CPU board, block 5 : Peripheral I/O port and power terminal

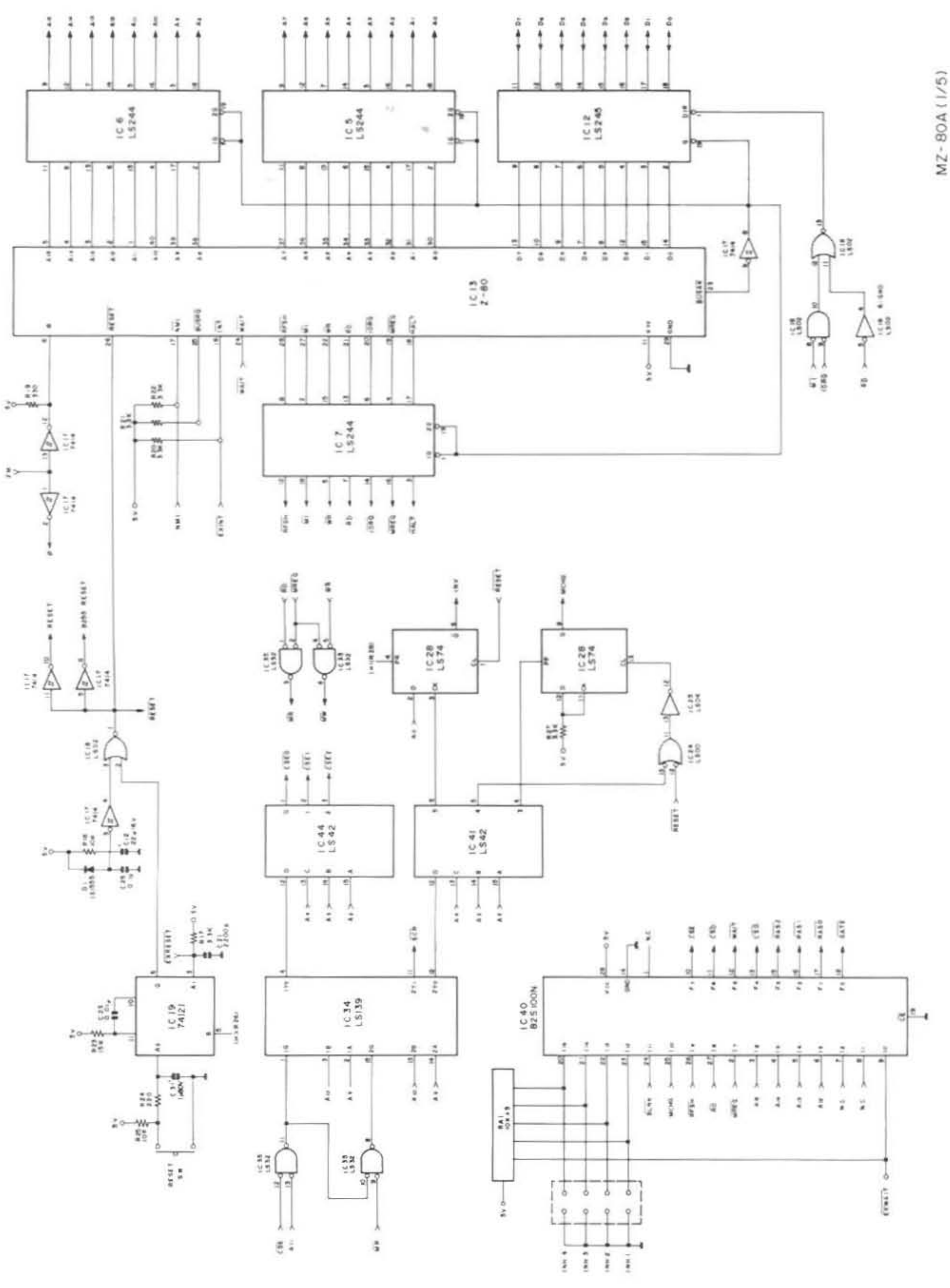


Figure 3.7 CPU board, block 1 : CPU signal system

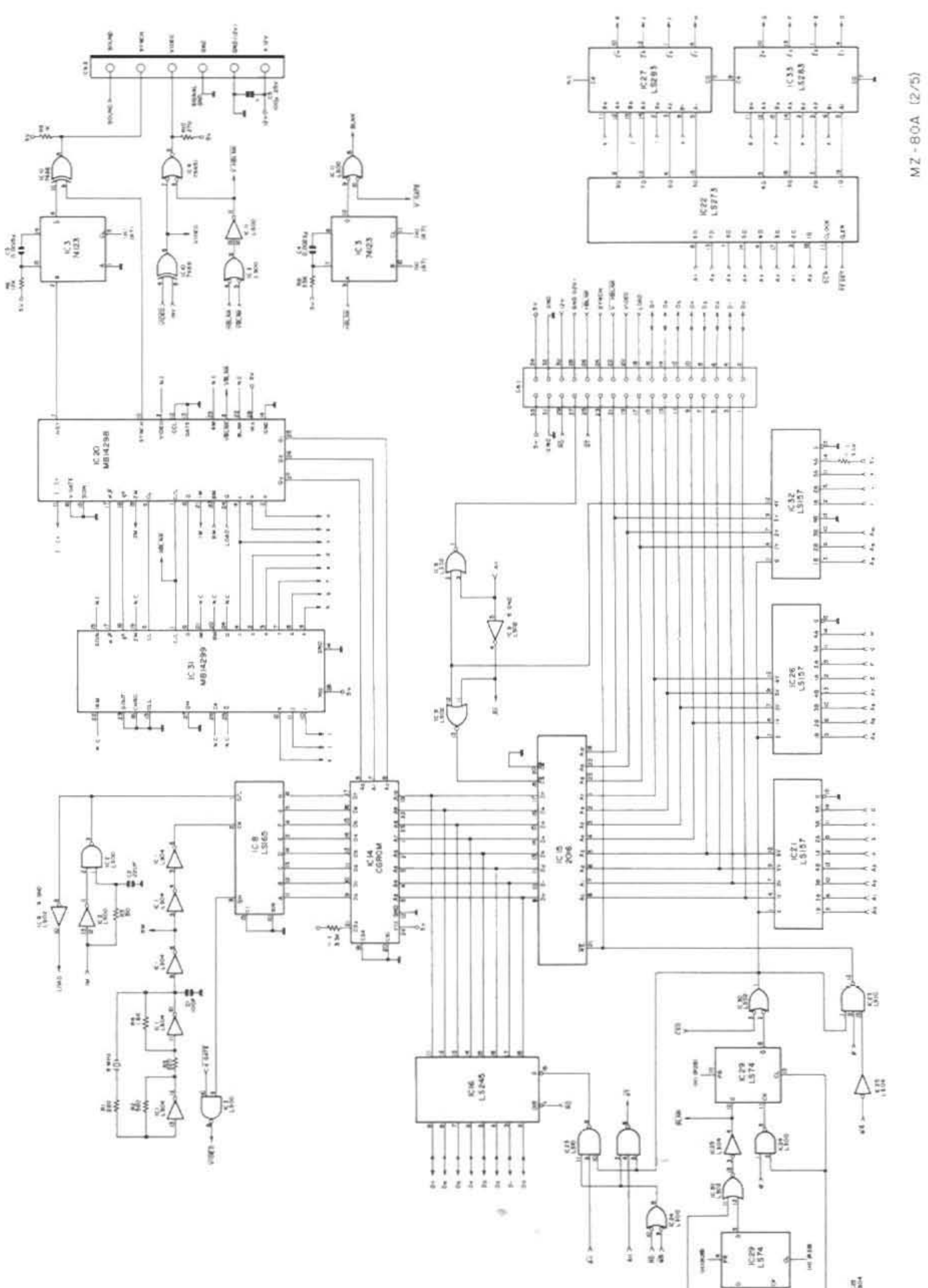


Figure 3.8 CPU board, block 2

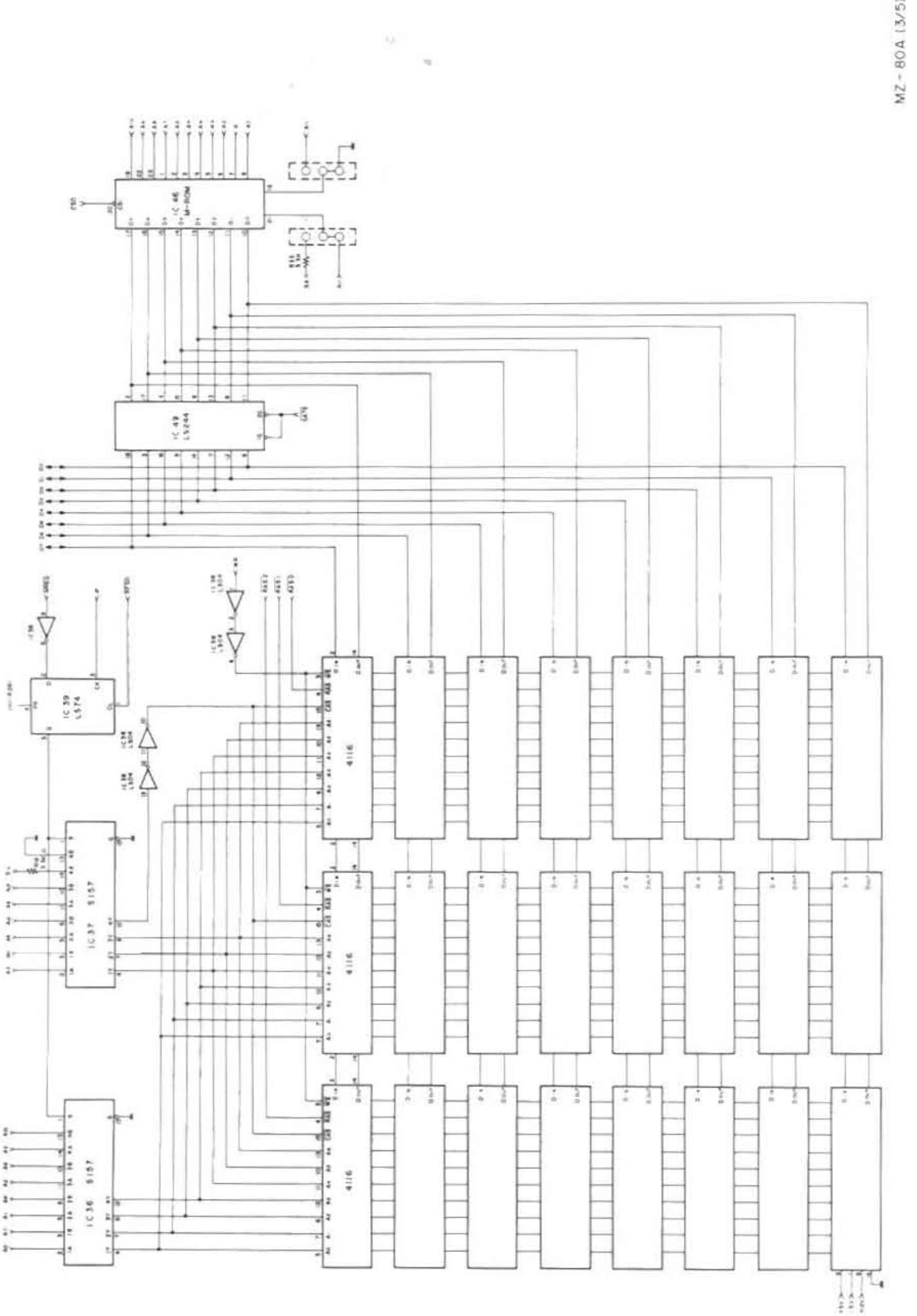


Figure 3.9   CPU board, block 3: RAM signal system

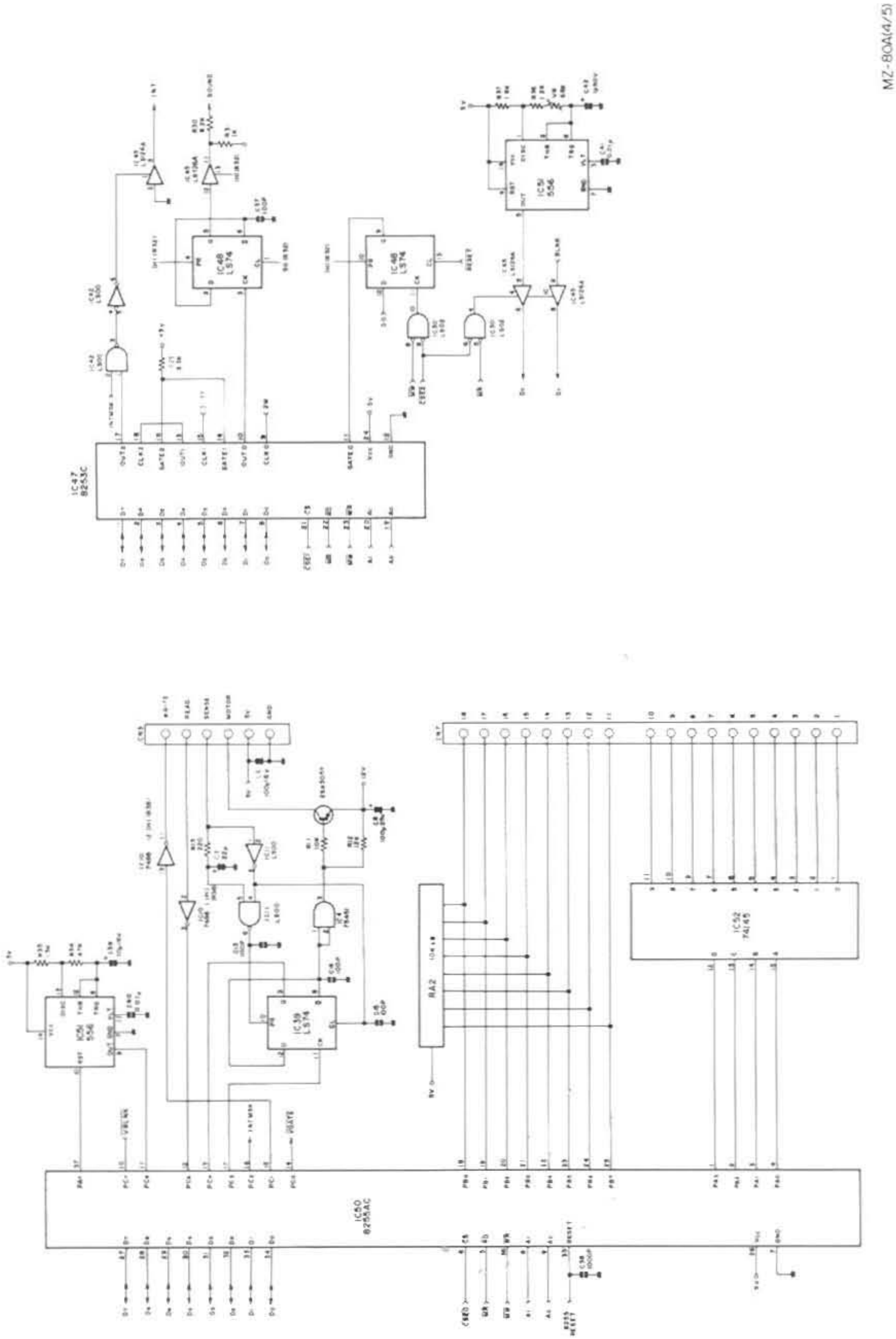
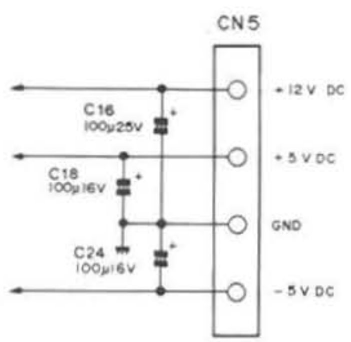


Figure 3.10 CPU board, block 4 : 8255 and 8253 signal system



CN 4

	A	B
1	D_1	D_0
2	D_3	D_2
3	GND	GND
4	D_5	D_4
5	D_7	D_6
6	GND	A_0
7	RESET	A_1
8	GND	A_2
9	$\overline{HALT}$	A_3
10	$\overline{M_1}$	A_4
11	GND	A_5
12	$\overline{WR}$	A_6
13	$\overline{RD}$	A_7
14	GND	A_8
15	$\overline{IOREQ}$	A_9
16	$\overline{MREQ}$	A_{10}
17	GND	A_{11}
18	$\overline{EXINT}$	A_{12}
19	GND	A_{13}
20	$\overline{NMI}$	A_{14}
21	$\overline{EX\ WAIT}$	A_{15}
22	$\overline{EX\ RESET}$	Φ

A : PARTS SIDE

MZ-80A (5/5)

Figure 3.11 CPU board, block 5 : Peripheral I/O port and power terminal

3.3 Expansion equipments

A variety of peripheral devices is available for expanding the MZ-80A personal computer system. Figure 3.12 shows a typical expanded system configuration. With the floppy disk drive, numerous data and program files can be stored and accessed at high speed. With the printer, hard copies of listings and printed graphic patterns can be obtained. This improved processing efficiency, resulting in a wider range of applications.

The MZ-80FB dual floppy disk drive uses a double density mini-floppy diskette (286K bytes/diskette) with a diameter of 5.25 inches, both sides of which are used for recording. It enables use of the DISK BASIC interpreters, which is suitable for practical business applications of the double precision DISK BASIC interpreter, which performs 16 digit BCD operations. Thus, the expanded system exhibits an ability which is comparable with that of larger computers with the aid of a variety of the floppy disk operating system software.

Figure 3.12 Typical expansion system

Figure 3.13 shows peripheral devices which can be connected to the MZ-80A. Devices which are enclosed in a thick solid line are interface cards or RAM blocks and they are connected to the expansion I/O port via interface terminals or connected to the specified connectors in the main cabinet.

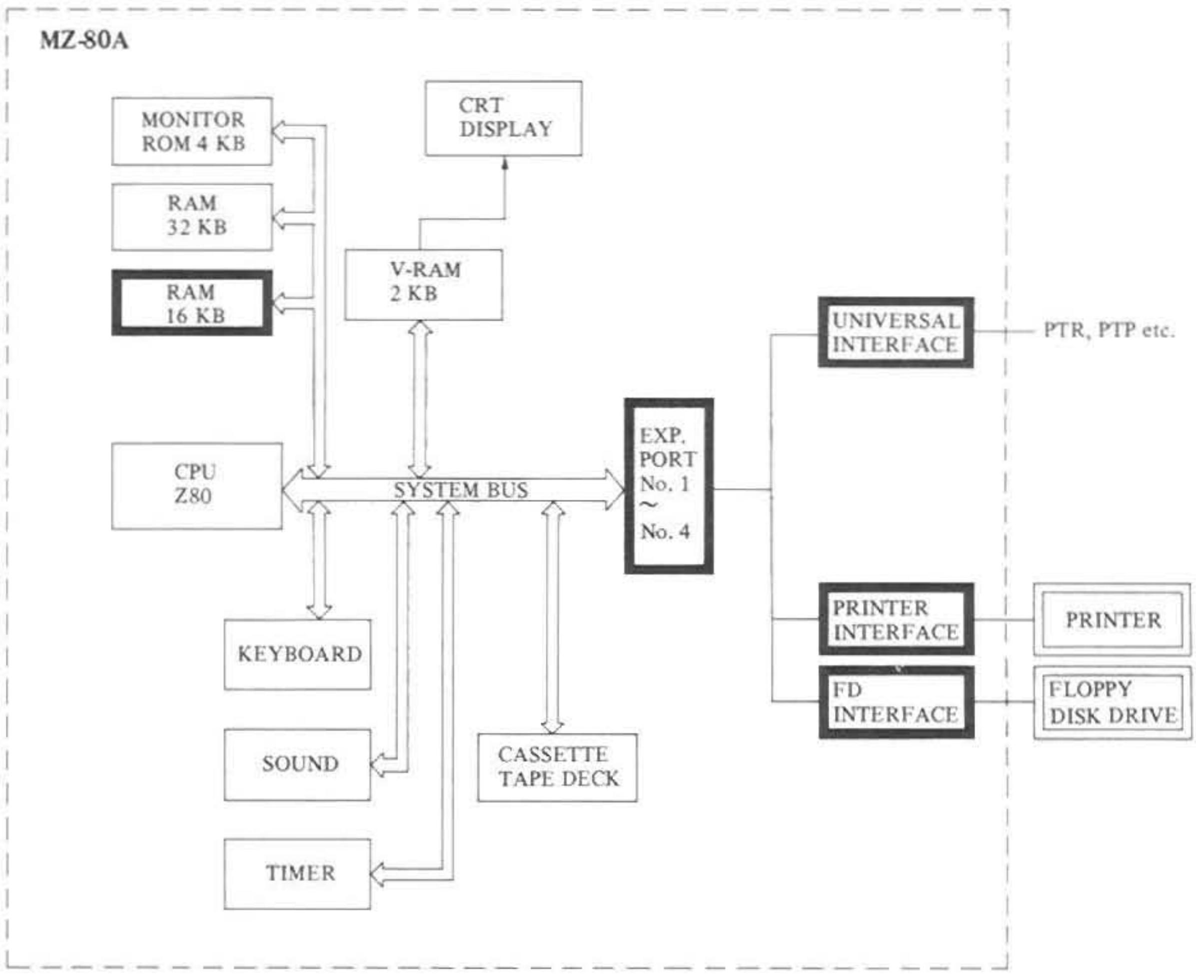


Figure 3.13 MZ-80A system expansion.

WARNING AND CAUTION

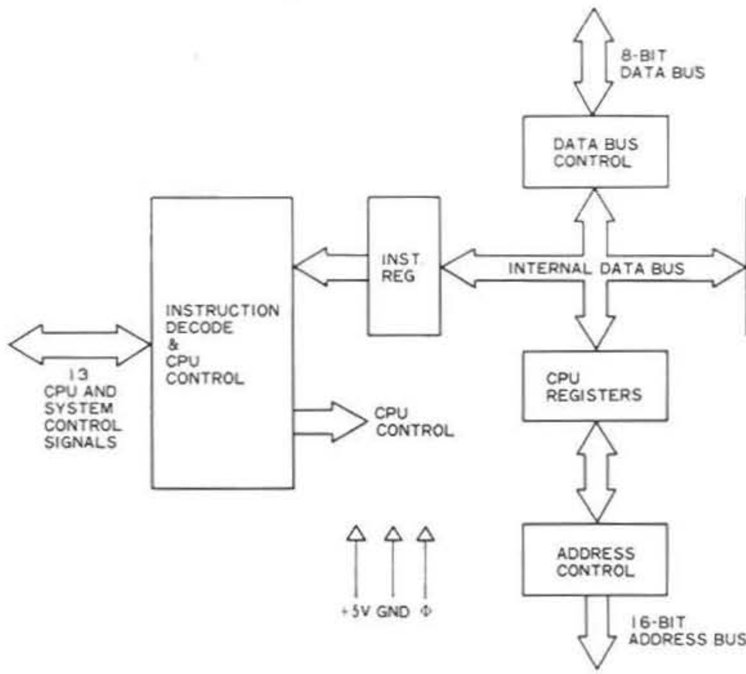
Warning : Dangerous voltage is inside of the main cabinet. Leave the installation of optional devices which can be connected in the main cabinet to your dealer.

Caution : If the power is turned on with the upper part of the main cabinet lifted, electrical parts may be damaged.
Metal articles remaining in the cabinet can cause serious trouble.
Ensure that no paper clips or other metallic articles fall into cabinet.

3.4 Technical data of Z-80 CPU

1.0 ARCHITECTURE

A block diagram of the internal architecture of the Z-80 CPU is shown in Figure 1.0-1. The diagram shows all of the major elements in the CPU and it should be referred to throughout the following description.



Z-80 CPU BLOCK DIAGRAM
FIGURE 1.0-1

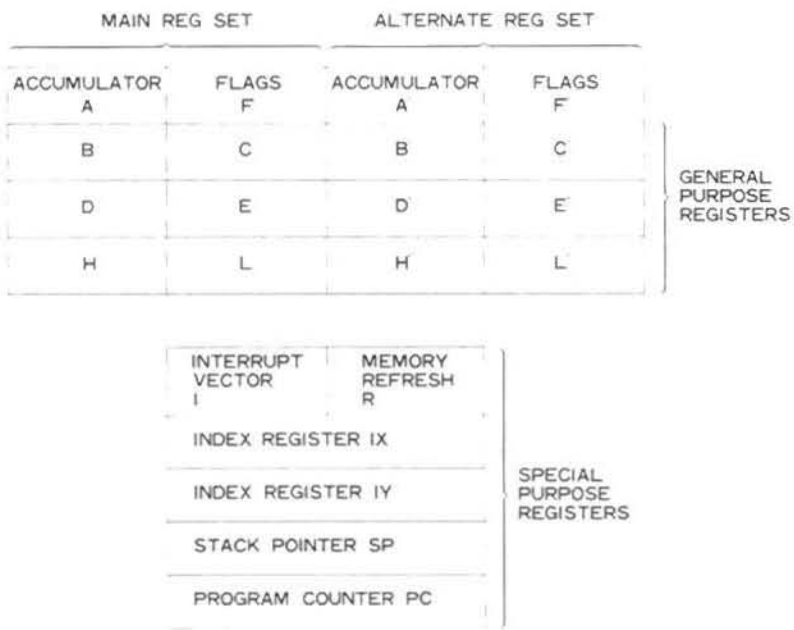
1.1 CPU REGISTERS

The Z-80 CPU contains 208 bits of R/W memory that are accessible to the programmer. Figure 1.0-2 illustrates how this memory is configured into eighteen 8-bit registers and four 16-bit registers. All Z-80 registers are implemented using static RAM. The registers include two sets of six general purpose registers that may be used individually as 8-bit registers or in pairs as 16-bit registers. There are also two sets of accumulator and flag registers.

Special Purpose Registers

1. **Program counter (PC).** The program counter holds the 16-bit address of the current instruction being fetched from memory. The PC is automatically incremented after its contents have been transferred to the address lines. When a program jump occurs the new value is automatically placed in the PC, overriding the incrementer.

2. **Stack Pointer (SP).** The stack pointer holds the 16-bit address of the current top of a stack located anywhere in external system RAM memory. The external stack memory is organized as a last-in first-out (LIFO) file. Data can be pushed onto the stack from specific CPU registers or popped off of the stack into specific CPU registers through the execution of PUSH and POP instructions. The data popped from the stack is always the last data pushed onto it. The stack allows simple implementation of multiple level interrupts, unlimited subroutine nesting and simplification of many types of data manipulation.



Z-80 CPU REGISTER CONFIGURATION
FIGURE 1.0-2

3. **Two Index Registers (IX & IY).** The two independent index registers hold a 16-bit base address that is used in indexed addressing modes. In this mode, an index register is used as a base to point to a region in memory from which data is to be stored or retrieved. An additional byte is included in indexed instructions to specify a displacement from this base. This displacement is specified as a two's complement signed integer. This mode of addressing greatly simplifies many types of programs, especially where tables of data are used.

4. **Interrupt Page Address Register (I).** The Z-80 CPU can be operated in a mode where an indirect call to any memory location can be achieved in response to an interrupt. The I Register is used for this purpose to store the high order 8-bits of the indirect address while the interrupting device provides the lower 8-bits of the address. This feature allows interrupt routines to be dynamically located anywhere in memory with absolute minimal access time to the routine.

5. **Memory Refresh Register (R).** The Z-80 CPU contains a memory refresh counter to enable dynamic memories to be used with the same ease as static memories. Seven bits of this 8-bit register are automatically incremented after each instruction fetch. The eighth bit will remain as programmed as the result of an LD R, A instruction. The data in the refresh counter is sent out on the lower portion of the address but along with a refresh control signal while the CPU is decoding and executing the fetched instruction. This mode of refresh is totally transparent to the programmer and does not slow down the CPU operation. The programmer can load the R register for testing purposes, but this register is normally not used by the programmer. During refresh, the contents of the I register are placed on the upper 8 bits of the address bus.

Accumulator and Flag Registers

The CPU includes two independent 8-bit accumulators and associated 8-bit flag registers. The accumulator holds the results of 8-bit arithmetic or logical operations while the flag register indicates specific conditions for 8 or 16-bit operations, such as indicating whether or not the result of an operation is equal to zero. The programmer selects the accumulator and flag pair that he wishes to work with a single exchange instruction so that he may easily work with either pair.

General Purpose Registers

There are two matched sets of general purpose registers, each set containing six 8-bit registers that may be used individually as 8-bit registers or as 16-bit register pairs by the programmer. One set is called BC, DE and HL while the complementary set is called BC', DE' and HL'. At any one time the programmer can select either set of registers to work with through a single exchange command for the entire set. In systems where fast interrupt response is required, one set of general purpose registers and an accumulator/flag register may be reserved for handling this very fast routine. Only a simple exchange commands need be executed to go between the routines. This greatly reduces interrupt service time by eliminating the requirement for saving and retrieving register contents in the external stack during interrupt or subroutine processing. These general purpose registers are used for a wide range of applications by the programmer. They also simplify programming, especially in ROM based systems where little external read/write memory is available.

1.2 ARITHMETIC AND LOGIC UNIT (ALU)

The 8-bit arithmetic and logical instructions of the CPU are executed in the ALU. Internally the ALU communicates with the registers and the external data bus on the internal data bus. The type of functions performed by the ALU include:

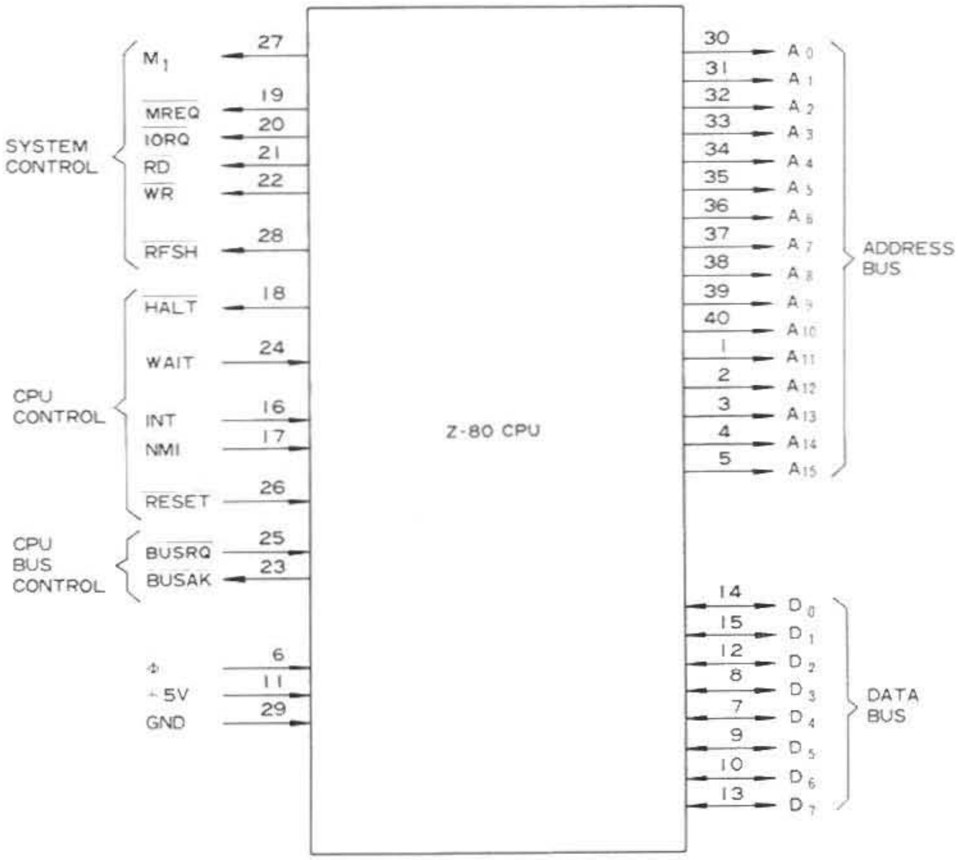
Add
Subtract
Logical AND
Logical OR
Logical Exclusive OR
Compare
Left or right shifts or rotates (arithmetic and logical)
Increment
Decrement
Set bit
Reset bit
Test bit

1.3 INSTRUCTION REGISTER AND CPU CONTROL

As each instruction is fetched from memory, it is placed in the instruction register and decoded. The control section performs this function and then generates and supplies all of the control signals necessary to read or write data from or to the registers, control the ALU and provide all required external control signals.

2.0 PIN DESCRIPTION

The Z-80 CPU is packaged in an industry standard 40 pin Dual In-Line Package. The I/O pins are shown in Figure 2.0-1 and the function of each is described below.



Z-80 PIN CONFIGURATION
FIGURE 2.0-1

A_0-A_{15} (Address Bus) — Tri-state output, active high. A_0-A_{15} constitute a 16-bit address bus. The address bus provides the address for memory (up to 64K bytes) data exchanger and for I/O device data exchanges. I/O addressing uses the 8 lower address bits to allow the user to directly select up to 256 input or 256 output ports. A_0 is the least significant address bit. During refresh time, the lower 7 bits contain a valid refresh address.

D_0-D_7 (Data Bus) — Tri-state input/output, active high. D_0-D_7 constitute an 8-bit bidirectional data bus. The data bus is used for data exchanges with memory and I/O devices.

$\overline{M_1}$ (Machine Cycle one) — Output, active low. $\overline{M_1}$ indicates that the current machine cycle is the OP code fetch cycle of an instruction execution. Note that during execution of 2-byte op-codes, $\overline{M_1}$ is generated as each op code byte is fetched. These two byte op-codes always begin with CBH, DDH, EDH or FDH. $\overline{M_1}$ also occurs with $\overline{IORQ}$ to indicate an interrupt acknowledge cycle.

$\overline{MREQ}$ (Memory Request) — Tri-state output, active low. The memory request signal indicates that the address bus holds a valid address for a memory read or memory write operation.

$\overline{\text{IORQ}}$
(Input/Output Request)

Tri-state output, active low. The $\overline{\text{IORQ}}$ signal indicates that the lower half of the address bus holds a valid I/O address for a I/O read or write operation. An IORQ signal is also generated with an $\overline{\text{M}_1}$ signal when an interrupt is being acknowledged to indicate that an interrupt response vector can be placed on the data bus. Interrupt Acknowledge operations occur during M_1 time while I/O operations never occur during M_1 time.

$\overline{\text{RD}}$
(Memory Read)

Tri-state output, active low. $\overline{\text{RD}}$ indicates that the CPU wants to read data from memory or an I/O device. The addressed I/O device or memory should use this signal to gate data onto the CPU data bus.

$\overline{\text{WR}}$
(Memory Write)

Tri-state output, active low. $\overline{\text{WR}}$ indicates that the CPU data bus holds valid data to be stored in the addressed memory or I/O device.

$\overline{\text{RFSH}}$
(Refresh)

Output, active low. $\overline{\text{RFSH}}$ indicates that the lower 7 bits of the address bus contain a refresh address for dynamic memories and the current $\overline{\text{MREQ}}$ signal should be used to do a refresh read to all dynamic memories.

$\overline{\text{HALT}}$
(Halt state)

Output, active low. $\overline{\text{HALT}}$ indicates that the CPU has executed a HALT software instruction and is awaiting either a non maskable or a maskable interrupt (with the mask enabled) before operation can resume. While halted, the CPU executes NOP's to maintain memory refresh activity.

$\overline{\text{WAIT}}$
(Wait)

Input, active low. $\overline{\text{WAIT}}$ indicates to the Z-80 CPU that the addressed memory or I/O devices are not ready for a data transfer. The CPU continues to enter wait states for as long as this signal is active. This signal allows memory or I/O devices of any speed to be synchronized to the CPU.

$\overline{\text{INT}}$
(Interrupt Request)

Input, active low. The Interrupt Request signal is generated by I/O devices. A request will be honored at the end of the current instruction if the internal software controlled interrupt enable flip-flop (IFF) is enabled and if the $\overline{\text{BUSRQ}}$ signal is not active. When the CPU accepts the interrupt, an acknowledge signal ($\overline{\text{IORQ}}$ during M_1 time) is sent out at the beginning of the next instruction cycle. The CPU can respond to an interrupt in three different modes.

$\overline{\text{NMI}}$
(Non Maskable Interrupt)

Input, negative edge triggered. The non maskable interrupt request line has a higher priority than $\overline{\text{INT}}$ and is always recognized at the end of the current instruction, independent of the status of the interrupt enable flip-flop. $\overline{\text{NMI}}$ automatically forces the Z-80 CPU to restart to location 0066_H. The program counter is automatically saved in the external stack so that the user can return to the program that was interrupted. Note that continuous WAIT cycles can prevent the current instruction from ending, and that a $\overline{\text{BUSRQ}}$ will override a $\overline{\text{NMI}}$.

$\overline{\text{RESET}}$

Input, active low, $\overline{\text{RESET}}$ forces the program counter to zero and initializes the CPU. The CPU initialization includes:

1) Disable the interrupt enable flip-flop
2) Set Register I = 00_H
3) Set Register R = 00_H
4) Set Interrupt Mode 0

During reset time, the address bus and data bus go to a high impedance state and all control output signals go to the inactive state.

$\overline{\text{BUSRQ}}$ (Bus Request)

Input, active low. The bus request signal is used to request the CPU address bus, data bus and tri-state output control signals to go to a high impedance state so that other devices can control these buses. When $\overline{\text{BUSRQ}}$ is activated, the CPU will set these buses to a high impedance state as soon as the current CPU machine cycle is terminated.

$\overline{\text{BUSAK}}$ (Bus Acknowledge)

Output, active low. Bus acknowledge is used to indicate to the requesting device that the CPU address bus, data bus and tri-state control bus signals have been set to their high impedance state and the external device can now control these signals.

Φ

Single phase TTL level clock which requires only a 330 ohm pull-up resistor to +5 volts to meet all clock requirements. (2 MHz)

3.0 INSTRUCTION SET

The Z-80 CPU can execute 158 different instruction types including all 78 of the 8080A CPU. The instructions can be broken down into the following major groups:

- Load and Exchange
- Block Transfer and Search
- Arithmetic and Logical
- Rotate and Shift
- Bit Manipulation (set, reset, test)
- Jump, Call and Return
- Input/Output
- Basic CPU Control

3.1 INTRODUCTION TO INSTRUCTION TYPES

The load instructions move data internally between CPU registers or between CPU registers and external memory. All of these instructions must specify a source location from which the data is to be moved and a destination location. The source location is not altered by a load instruction. Examples of load group instructions include moves between any of the general purpose registers such as move the data to Register B from Register C. This group also includes load immediate to any CPU register or to any to Register B from Register C. This group also includes load immediate to any CPU register or to any external memory location. Other types of load instructions allow transfer between CPU registers and memory locations. The exchange instructions can trade the contents of two registers.

A unique set of block transfer instructions is provided in the Z-80. With a single instruction a block of memory of any size can be moved to any other location in memory. This set of block moves is extremely valuable when large strings of data must be processed. The Z-80 block search instructions are also valuable for this type of processing. With a single instruction, a block of external memory of any desired length can be searched for any 8-bit character. Once the character is found or the end of the block is reached, the instruction automatically terminates. Both the block transfer and the block search instructions can be interrupted during their execution so as to not occupy the CPU for long periods of time.

The arithmetic and logical instructions operate on data stored in the accumulator and other general purpose CPU registers or external memory locations. The results of the operations are placed in the accumulator and the appropriate flags are set according to the result of the operation. An example of an arithmetic operation is adding the accumulator to the contents of an external memory location. The results of the addition are placed in the accumulator. This group also includes 16-bit addition and subtraction between 16-bit CPU registers.

The rotate and shift group allows any register or any memory location to be rotated right or left with or without carry either arithmetic or logical. Also, a digit in the accumulator can be rotated right or left with two digits in any memory location.

The bit manipulation instructions allow any bit in the accumulator, any general purpose register or any external memory location to be set, reset or tested with a single instruction. For example, the most significant bit of register H can be reset. This group is especially useful in control applications and for controlling software flags in general purpose programming.

The jump, call and return instructions are used to transfer between various locations in the user's program. This group uses several different techniques for obtaining the new program counter address from specific external memory locations. A unique type of call is the restart instruction. This instruction actually contains the new address as a part of the 8-bit OP code. This is possible since only 8 separate addresses located in page zero of the external memory may be specified. Program jumps may also be achieved by loading register HL, IX or IY directly into the PC, thus allowing the jump address to be a complex function of the routine being executed.

The input/output group of instructions in the Z-80 allow for a wide range of transfers between external memory locations or the general purpose CPU registers, and the external I/O devices. In each case, the port number is provided on the lower 8 bits of the address bus during any I/O transaction. One instruction allows this port number to be specified by the second byte of the instruction while other Z-80 instructions allow it to be specified as the content of the C register. One major advantage of using the C register as a pointer to the I/O device is that it allows different I/O ports to share common software driver routines. This is not possible when the address is part of the OP code if the routines are stored in ROM. Another feature of these input instructions is that they set the flag register automatically so that additional operations are not required to determine the state of the input data (for example its parity). The Z-80 CPU includes single instructions that can move blocks of data (up to 256 bytes) automatically to or from any I/O port directly to any memory location. In conjunction with the dual set of general purpose registers, these instructions provide for fast I/O block transfer rates. The value of this I/O instruction set is demonstrated by the fact that the Z-80 CPU can provide all required floppy disk formatting (i.e., the CPU provides the preamble, address, data and enables the CRC codes) on double density floppy disk drives on an interrupt driven basis.

Finally, the basic CPU control instructions allow various options and modes. This group includes instructions such as setting or resetting the interrupt enable flip flop or setting the mode of interrupt response.

3.2 ADDRESSING MODES

Most of the Z-80 instructions operate on data stored in internal CPU registers, external memory or in the I/O ports. Addressing refers to how the address of this data is generated in each instruction. This section gives a brief summary of the types of addressing used in the Z-80 while subsequent sections detail the type of addressing available for each instruction group.

Immediate. In this mode of addressing the byte following the OP code in memory contains the actual operand.

OP Code	} one or 2 bytes
Operand	

d_7 d_0

Examples of this type of instruction would be to load the accumulator with a constant, where the constant is the byte immediately following the OP code.

Immediate Extended. This mode is merely an extension of immediate addressing in that the two bytes following the OP codes are the operand.

OP code	one or 2 bytes
Operand	low order
Operand	high order

Examples of this type of instruction would be to load the HL register pair (16-bit register) with 16 bits (2 bytes) of data.

Modified Page Zero Addressing. The Z-80 has a special single byte CALL instruction to any of 8 locations in page zero of memory. This instruction (which is referred to as a restart) sets the PC to an effective address in page zero. The value of this instruction is that it allows a single byte to specify a complete 16-bit address where commonly called subroutines are located, thus saving memory space.

OP Code	one byte

b_7 b_0

Effective address is $(b_5\ b_4\ b_3\ 000)_2$

Relative Addressing. Relative addressing uses one byte of data following the OP code to specify a displacement from the existing program to which a program jump can occur. This displacement is a signed two's complement number that is added to the address of the OP code of the following instruction.

OP Code	Jump relative (one byte OP code)
Operand	8-bit two's complement displacement added to Address (A+2)

The value of relative addressing is that it allows jumps to nearby locations while only requiring two bytes of memory space. For most programs, relative jumps are by far the most prevalent type of jump due to the proximity of related program segments. Thus, these instructions can significantly reduce memory space requirements. The signal displacement can range between +127 and −128 from A+2. This allows for a total displacement of +129 to −126 from the jump relative OP code address. Another major advantage is that it allows for relocatable code.

Extended Addressing. Extended Addressing provides for two bytes (16 bits) of address to be included in the instruction. This data can be an address to which a program can jump or it can be an address where an operand is located.

OP Code	one or two bytes
Low Order Address or Low order operand	
High Order Address or High order operand	

Extended addressing is required for a program to jump from any location in memory to any other location, or load and store data in any memory location.

When extended addressing is used to specify the source or destination address of an operand, the notation (nn) will be used to indicate the content of memory at nn, where nn is the 16-bit address specified in the instruction. This means that the two bytes of address nn are used as a pointer to a memory location. The use of the parentheses always means that the value enclosed within them is used as a pointer to a memory location. For example, (1200) refers to the contents of memory at location 1200.

Indexed Addressing. In this type of addressing, the byte of data following the OP code contains a displacement which is added to one of the two index registers (the OP code specifies which index register is used) to form a pointer to memory. The contents of the index register are not altered by this operation.

OP Code	two byte OP code
OP Code	
Displacement	Operand added to index register to form a pointer to memory

An example of an indexed instruction would be to load the contents of the memory location (Index Register + Displacement) into the accumulator. The displacement is a signed two's complement number. Indexed addressing greatly simplifies programs using tables of data since the index register can point to the start of any table. Two index registers are provided since very often operations require two or more tables. Indexed addressing also allows for relocatable code.

The two index registers in the Z-80 are referred to as IX and IY. To indicate indexed addressing the notation:

(IX + d) or (IY + d)

is used. Here d is the displacement specified after the OP code. The parentheses indicate that this value is used as a pointer to external memory.

Register Addressing. Many of the Z-80 OP codes contain bits of information that specify which CPU register is to be used for an operation. An example of register addressing would be to load the data in register B into register C.

Implied Addressing. Implied addressing refers to operations where the OP code automatically implies one or more CPU registers as containing the operands. An example is the set of arithmetic operations where the accumulator is always implied to be the destination of the results.

Register Indirect Addressing. This type of addressing specifies a 16-bit CPU register pair (such as HL) to be used as a pointer to any location in memory. This type of instruction is very powerful and it is used in a wide range of applications.

OP Code	} one or two bytes

An example of this type of instruction would be to load the accumulator with the data in the memory location pointed to by the HL register contents. Indexed addressing is actually a form of register indirect addressing except that a displacement is added with indexed addressing. Register indirect addressing allows for very powerful but simple to implement memory accesses. The block move and search commands in the Z-80 are extensions of this type of addressing where automatic register incrementing, decrementing and comparing has been added. The notation for indicating register indirect addressing is to put parentheses around the name of the register that is to be used as the pointer. For example, the symbol

(HL)

specifies that the contents of the HL register are to be used as a pointer to a memory location. Often register indirect addressing is used to specify 16-bit operands. In this case, the register contents point to the lower order portion of the operand while the register contents are automatically incremented to obtain the upper portion of the operand.

Bit Addressing. The Z-80 contains a large number of bit set, reset and test instructions. These instructions allow any memory location or CPU register to be specified for a bit operation through one of three previous addressing modes (register, register indirect and indexed) while three bits in the OP code specify which of the eight bits is to be manipulated.

ADDRESSING MODE COMBINATIONS

Many instructions include more than one operand (such as arithmetic instructions or loads). In these cases, two types of addressing may be employed. For example, load can use immediate addressing to specify the source and register indirect or indexed addressing to specify the destination.

3.3 INSTRUCTION OF OP CODES AND EXECUTION TIMES

The following section gives a summary of the Z-80 instructions set. The instructions are logically arranged into groups as shown on tables 3.3-1 through 3.3-11. Each table shows the assembly language mnemonic OP code, the actual OP code, the symbolic operation, the content of the flag register following the execution of each instruction, the number of bytes required for each instruction as well as the number of memory cycles and the total number of T states (external clock periods) required for the fetching and execution of each instruction.

All instruction OP codes are listed in binary notation. Single byte OP codes require two hex characters while double byte OP codes require four hex characters. The conversion from binary to hex is repeated here for convenience.

Binary	Hex	Decimal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	OP-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
LD r,r'	r←r'	•	•	•	•	•	•	01 r r'	1	1	4	
LD r,n	r←n	•	•	•	•	•	•	00 r 110 ← n →	2	2	7	
LD r,(HL)	r←(HL)	•	•	•	•	•	•	01 r 110	1	2	7	
LD r,(IX+d)	r←(IX+d)	•	•	•	•	•	•	11 011 101 01 r 110 ← d →	3	5	19	
LD r,(IY+d)	r←(IY+d)	•	•	•	•	•	•	11 111 101 01 r 110 ← d →	3	5	19	
LD (HL),r	(HL)←r	•	•	•	•	•	•	01 110 r	1	2	7	
LD (IX+d),r	(IX+d)←r	•	•	•	•	•	•	11 011 101 01 110 r ← d →	3	5	19	
LD (IY+d),r	(IY+d)←r	•	•	•	•	•	•	11 111 101 01 110 r ← d →	3	5	19	
LD (HL),n	(HL)←n	•	•	•	•	•	•	00 110 110 ← n →	2	3	10	
LD (IX+d),n	(IX+d)←n	•	•	•	•	•	•	11 011 101 00 110 110 ← d → ← n →	4	5	19	
LD (IY+d),n	(IY+d)←n	•	•	•	•	•	•	11 111 101 00 110 110 ← d → ← n →	4	5	19	
LD A,(BC)	A←(BC)	•	•	•	•	•	•	00 001 010	1	2	7	
LD A,(DE)	A←(DE)	•	•	•	•	•	•	00 011 010	1	2	7	
LD A,(nn)	A←(nn)	•	•	•	•	•	•	00 111 010 ← n → ← n →	3	4	13	
LD (BC),A	(BC)←A	•	•	•	•	•	•	00 000 010	1	2	7	
LD (DE),A	(DE)←A	•	•	•	•	•	•	00 010 010	1	2	7	
LD (nn),A	(nn)←A	•	•	•	•	•	•	00 110 010 ← n → ← n →	3	4	13	
LD A,I	A←I	•	↕	IFF2	↕	0	0	11 101 101 01 010 111	2	2	9	
LD A,R	A←R	•	↕	IFF2	↕	0	0	11 101 101 01 011 111	2	2	9	
LD I,A	I←A	•	•	•	•	•	•	11 101 101 01 000 111	2	2	9	
LD R,A	R←A	•	•	•	•	•	•	11 101 101 01 001 111	2	2	9	

r, r'	Reg.
000	B
001	C
010	D
011	E
100	H
101	L
111	A

Notes: r, r' means any of the registers A, B, C, D, E, H, L
IFF the content of the interrupt enable flip-flop (IFF) is copied into the P/V flag

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ = flag is affected according to the result of the operation.

8-BIT LOAD GROUP
TABLE 3.3-1

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	OP-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments	
LD dd,nn	dd←nn	●	●	●	●	●	●	00 dd0 001	3	3	10	dd	Pair
								← n →				00	BC
								← n →				01	DE
LD IX,nn	IX←nn	●	●	●	●	●	●	11 011 101	4	4	14	10	HL
								00 100 001				11	SP
								← n →					
								← n →					
LD IY,nn	IY←nn	●	●	●	●	●	●	11 111 101	4	4	14		
								00 100 001					
								← n →					
								← n →					
LD HL,(nn)	H←(nn+1)	●	●	●	●	●	●	00 101 010	3	5	16		
	L←(nn)							← n →					
								← n →					
LD dd,(nn)	dd_H←(nn+1)	●	●	●	●	●	●	11 101 101	4	6	20		
	dd_L←(nn)							01 dd1 011					
								← n →					
								← n →					
LD IX,(nn)	IX_H←(nn+1)	●	●	●	●	●	●	11 011 101	4	6	20		
	IX_L←(nn)							00 101 010					
								← n →					
								← n →					
LD IY,(nn)	IY_H←(nn+1)	●	●	●	●	●	●	11 111 101	4	6	20		
	IY_L←(nn)							00 101 010					
								← n →					
								← n →					
LD (nn),HL	(nn+1)←H	●	●	●	●	●	●	00 100 010	3	5	16		
	(nn)←L							← n →					
								← n →					
LD (nn),dd	(nn+1)←dd_H	●	●	●	●	●	●	11 101 101	4	6	20		
	(nn)←dd_L							01 dd0 011					
								← n →					
								← n →					
LD (nn),IX	(nn+1)←IX_H	●	●	●	●	●	●	11 011 101	4	6	20		
	(nn)←IX_L							00 100 010					
								← n →					
								← n →					
LD (nn),IY	(nn+1)←IY_H	●	●	●	●	●	●	11 111 101	4	6	20		
	(nn)←IY_L							00 100 010					
								← n →					
								← n →					
LD SP,HL	SP←HL	●	●	●	●	●	●	11 111 001	1	1	6		
LD SP,IX	SP←IX	●	●	●	●	●	●	11 011 101	2	2	10		
								11 111 001					
LD SP,IY	SP←IY	●	●	●	●	●	●	11 111 101	2	2	10		
								11 111 001					

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	OP-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
PUSH qq	$(SP-2) \leftarrow qq_L$ $(SP-1) \leftarrow qq_H$	•	•	•	•	•	•	11 qq0 101	1	3	11	qq — Pair 00 — BC
PUSH IX	$(SP-2) \leftarrow IX_L$ $(SP-1) \leftarrow IX_H$	•	•	•	•	•	•	11 011 101 11 100 101	2	4	15	01 — DE 10 — HL
PUSH IY	$(SP-2) \leftarrow IY_L$ $(SP-1) \leftarrow IY_H$	•	•	•	•	•	•	11 111 101 11 100 101	2	4	15	11 — AF
POP qq	$qq_H \leftarrow (SP+1)$ $qq_L \leftarrow (SP)$	•	•	•	•	•	•	11 qq0 001	1	3	10	
POP IX	$IX_H \leftarrow (SP+1)$ $IX_L \leftarrow (SP)$	•	•	•	•	•	•	11 011 101 11 100 001	2	4	14	
POP IY	$IY_H \leftarrow (SP+1)$ $IY_L \leftarrow (SP)$	•	•	•	•	•	•	11 111 101 11 100 001	2	4	14	

Notes: dd is any of the register pairs BC, DE, HL, SP
qq is any of the register pairs AF, BC, DE, HL
$(PAIR)_H$, $(PAIR)_L$ refer to high order and low order eight bits of the register pair respectively.
E.g. $BC_L = C$, $AF_H = A$

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ flag is affected according to the result of the operation.

16-BIT LOAD GROUP
TABLE 3.3-2

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	OP-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
EX DE, HL	DE↔HL	•	•	•	•	•	•	11 101 011	1	1	4	
EX AF, AF′	AF↔AF′	•	•	•	•	•	•	00 001 000	1	1	4	
EXX	(BC, DE, HL) ↔ (BC′, DE′, HL′)	•	•	•	•	•	•	11 011 001	1	1	4	Register bank and auxiliary register bank exchange
EX (SP), HL	H↔(SP+1) L↔(SP)	•	•	•	•	•	•	11 100 011	1	5	19	
EX (SP), IX	IX_H↔(SP+1) IX_L↔(SP)	•	•	•	•	•	•	11 011 101 11 100 011	2	6	23	
EX (SP), IY	IY_H↔(SP+1) IY_L↔(SP)	•	•	•	•	•	•	11 111 101 11 100 011	2	6	23	
LDI	(DE)←(HL) DE←DE+1 HL←HL+1 BC←BC−1	•	•	① ↕	•	0	0	11 101 101 10 100 000	2	4	16	Load (HL) into (DE), increment the pointers and decrement the byte counter (BC)
LDIR	(DE)←(HL) DE←DE+1 HL←HL+1 BC←BC−1 Repeat until BC=0	•	•	0	•	0	0	11 101 101 10 110 000	2 2	5 4	21 16	If BC≠0 If BC=0
LDD	(DE)←(HL) DE←DE−1 HL←HL−1 BC←BC−1	•	•	① ↕	•	0	0	11 101 101 10 101 000	2	4	16	
LDDR	(DE)←(HL) DE←DE-1 HL←HL-1 BC←BC-1 Repeat until BC=0	•	•	0	•	0	0	11 101 101 10 111 000	2 2	5 4	21 16	If BC≠0 If BC=0
CPI	A−(HL) HL←HL+1 BC←BC−1	•	② ↕	① ↕	↕	1	↕	11 101 101 10 100 001	2	4	16	

Mnemonic	Symbolic Operation	Flags						OP-Code	No. of Bytes	No. of M Cycles	No. of T States	Comments
		C	Z	P/V	S	N	H	76 543 210				
CPIR	A − (HL) HL←HL+1 BC←BC−1 Repeat until A = (HL) or BC = 0	•	② ↕	① ↕	↕	1	↕	11 101 101 10 110 001	2 2	5 4	21 16	If BC ≠ 0 and A ≠ (HL) If BC = 0 or A = (HL)
CPD	A − (HL) HL←HL−1 BC←BC−1	•	② ↕	① ↕	↕	1	↕	11 101 101 10 101 001	2	4	16	
CPDR	A − (HL) HL←HL−1 BC←BC−1 Repeat until A = (HL) or BC = 0	•	② ↕	① ↕	↕	1	↕	11 101 101 10 111 001	2 2	5 4	21 16	If BC ≠ 0 and A ≠ (HL) If BC = 0 or A = (HL)

Notes: ① P/V flag is 0 if the result of BC-1 = 0, otherwise P/V = 1
② Z flag is 1 if A = (HL), otherwise Z = 0.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ = flag is affected according to the result of the operation.

EXCHANGE GROUP AND BLOCK TRANSFER AND SEARCH GROUP
TABLE 3.3-3

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	OP-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
ADD A,r	A←A+r	↕	↕	V	↕	0	↕	10 [000] r	1	1	4	r — Reg.
ADD A,n	A←A+n	↕	↕	V	↕	0	↕	11 [000] 110	2	2	7	000 — B
								← n →				001 — C
ADD A,(HL)	A←A+(HL)	↕	↕	V	↕	0	↕	10 [000] 110	1	2	7	010 — D
ADD A,(IX+d)	A←A+(IX+d)	↕	↕	V	↕	0	↕	11 011 101	3	5	19	011 — E
								10 [000] 110				100 — H
								← d →				101 — L
ADD A,(IY+d)	A←A+(IY+d)	↕	↕	V	↕	0	↕	11 111 101	3	5	19	111 — A
								10 [000] 110				
								← d →				
ADC A,s	A←A+s+CY	↕	↕	V	↕	0	↕	[001]				s is any of r, n, (HL), (IX+d), (IY+d) as shown for ADD instruction The indicated bits replace the [000] in the ADD set above.
SUB s	A←A−s	↕	↕	V	↕	1	↕	[010]				
SBC A,s	A←A−s−CY	↕	↕	V	↕	1	↕	[011]				
AND s	A←A∧s	0	↕	P	↕	0	1	[100]				
OR s	A←A∨s	0	↕	P	↕	0	0	[110]				
XOR s	A←A⊻s	0	↕	P	↕	0	0	[101]				
CP s	A−s	↕	↕	V	↕	1	↕	[111]				
INC r	r←r+1	•	↕	V	↕	0	↕	00 r [100]	1	1	4	
INC (HL)	(HL)←(HL)+1	•	↕	V	↕	0	↕	00 110 [100]	1	3	11	
INC (IX+d)	(IX+d)←(IX+d)+1	•	↕	V	↕	0	↕	11 011 101	3	6	23	
								00 110 [100]				
								← d →				
INC (IY+d)	(IY+d)←(IY+d)+1	•	↕	V	↕	0	↕	11 111 101	3	6	23	
								00 110 [100]				
								← d →				
DEC m	m←m−1	•	↕	V	↕	1	↕	[101]				m is any of r, (HL), (IX+d), (IY+d) as shown for INC. Same format and states as INC. Replace [100] with [101] in OP-code.

Notes: The V symbol in the P/V flag column indicates that the P/V flag contains the overflow of the result of the operation Similarly the P symbol indicates parity. V = 1 means overflow, V = 0 means not overflow, P = 1 means parity of the result is even, P = 0 means panty of the result is odd.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown, ↕ = flag is affected according to the result of the operation.

8-BIT ARITHMETIC AND LOGICAL GROUP
TABLE 3.3-4

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	OP-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
DAA	Converts acc content into packed BCD following add or subtract with packed BCD operands	↕	↕	P	↕	•	↕	00 100 111	1	1	4	Decimal adjust accumulator
CPL	$A \leftarrow \bar{A}$	•	•	•	•	1	1	00 101 111	1	1	4	Complement accumulator (one's complement)
NEG	$A \leftarrow \bar{A}+1$	↕	↕	V	↕	1	↕	11 101 101 01 000 100	2	2	8	Negate acc. (two's complement)
CCF	$CY \leftarrow \overline{CY}$	↕	•	•	•	0	X	00 111 111	1	1	4	Complement carry flag
SCF	$CY \leftarrow 1$	1	•	•	•	0	0	00 110 111	1	1	4	Set carry flag
NOP	No operation $PC \leftarrow PC+1$	•	•	•	•	•	•	00 000 000	1	1	4	
HALT	CPU halted	•	•	•	•	•	•	01 110 110	1	1	4	
DI	$IFF \leftarrow 0$	•	•	•	•	•	•	11 110 011	1	1	4	
EI	$IFF \leftarrow 1$	•	•	•	•	•	•	11 111 011	1	1	4	
IM 0	Sst interrupt mode 0	•	•	•	•	•	•	11 101 101 01 000 110	2	2	8	
IM 1	Set interrupt mode 1	•	•	•	•	•	•	11 101 101 01 010 110	2	2	8	
IM 2	Set interrupt mode 2	•	•	•	•	•	•	11 101 101 01 011 110	2	2	8	

Notes: IFF indicates the interrupt enable flip-flop
CY indicates the carry flip-flop.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ = flag is affected according to the result of the operation.

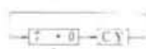
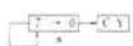
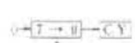
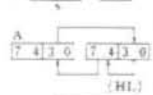
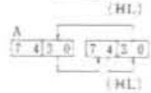
GENERAL PURPOSE ARITHMETIC AND CPU CONTROL GROUPS
TABLE 3.3-5

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	OP-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments	
ADD HL,ss	HL←HL+ss	↕	•	•	•	0	X	00 ss1 001	1	3	11	ss	Reg.
ADC HL,ss	HL←HL+ss+CY	↕	↕	V	↕	0	X	11 101 101	2	4	15	00	BC
								01 ss1 010				01	DE
SBC HL,ss	HL←HL−ss−CY	↕	↕	V	↕	1	X	11 101 101	2	4	15	10	HL
								01 ss0 010				11	SP
ADD IX,pp	IX←IX+pp	↕	•	•	•	0	X	11 011 101	2	4	15	pp	Reg.
								00 pp1 001				00	BC
												01	DE
												10	IX
												11	SP
ADD IY,rr	IY←IY+rr	↕	•	•	•	0	X	11 111 101	2	4	15	rr	Reg.
								00 rr1 001				00	BC
												01	DE
												10	IY
												11	SP
INC ss	ss←ss+1	•	•	•	•	•	•	00 ss0 011	1	1	6		
INC IX	IX←IX+1	•	•	•	•	•	•	11 011 101	2	2	10		
								00 100 011					
INC IY	IY←IY+1	•	•	•	•	•	•	11 111 101	2	2	10		
								00 100 011					
DEC ss	ss←ss−1	•	•	•	•	•	•	00 ss1 011	1	1	6		
DEC IX	IX←IX−1	•	•	•	•	•	•	11 011 101	2	2	10		
								00 101 011					
DEC IY	IY←IY−1	•	•	•	•	•	•	11 111 101	2	2	10		
								00 101 011					

Notes: ss is any of the register pairs BC, DE, HL, SP
pp is any of the register pairs BC, DE, IX, SP
rr is any of the register pairs BC, DE, IY, SP.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ = flag is affected according to the result of the operation.

16-BIT ARITHMETIC GROUP
TABLE 3.3-6

Mnemonic	Symbolic Operation	C	Z	P/V	S	N	H	OP-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
RLC A		↕	•	•	•	0	0	00 000 111	1	1	4	Rotate left circular accumulator
RL A		↕	•	•	•	0	0	00 010 111	1	1	4	Rotate left accumulator
RRC A		↕	•	•	•	0	0	00 001 111	1	1	4	Rotate right circular accumulator
RR A		↕	•	•	•	0	0	00 011 111	1	1	4	Rotate right accumulator
RLC r		↕	↕	P	↕	0	0	11 001 011 00 000 r	2	2	8	Rotate left circular register r
RLC (HL)		↕	↕	P	↕	0	0	11 001 011 00 000 110	2	4	15	r Reg. 000 B 001 C 010 D 011 E 100 H 101 L 111 A
RLC (IX+d)		↕	↕	P	↕	0	0	11 011 101 11 001 011 ← d → 00 000 110	4	6	23	
RLC (IY+d)		↕	↕	P	↕	0	0	11 111 101 11 001 011 ← d → 00 000 110	4	6	23	
RL s		↕	↕	P	↕	0	0	010				Instruction format and states are as shown for RLC, m. To form new OP-code replace 000 of RLC, m with shown code.
RRC s		↕	↕	P	↕	0	0	001				
RR s		↕	↕	P	↕	0	0	011				
SLA s		↕	↕	P	↕	0	0	100				
SRA s		↕	↕	P	↕	0	0	101				
SRL s		↕	↕	P	↕	0	0	111				
RLD		•	↕	P	↕	0	0	11 101 101 01 101 111	2	5	18	Rotate digit left and right between the accumulator and location (HL).
RRD		•	↕	P	↕	0	0	11 101 101 01 100 111	2	5	18	The content of the upper half of the accumulator is unaffected.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ = flag is affected according to the result of the operation.

ROTATE AND SHIFT GROUP
TABLE 3.3-7

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	OP-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments	
BIT b, r	$Z \leftarrow \overline{r_b}$	•	↕	X	X	0	1	11 001 011	2	2	8	r	Reg.
								01 b r				000	B
BIT b, (HL)	$Z \leftarrow \overline{(HL)_b}$	•	↕	X	X	0	1	11 001 011	2	3	12	001	C
								01 b 110				010	D
BIT b, (IX + d)	$Z \leftarrow \overline{(IX+d)_b}$	•	↕	X	X	0	1	11 011 101	4	5	20	011	E
								11 001 011				100	H
								← d →				101	L
								01 b 110				111	A
BIT b, (IY + d)	$Z \leftarrow \overline{(IY+d)_b}$	•	↕	X	X	0	1	11 111 101	4	5	20	b	Bit Tested
								11 001 011				000	0
								← d →				001	1
								01 b 110				010	2
SET b, r	$r_b \leftarrow 1$	•	•	•	•	•	•	11 001 011	2	2	8	011	3
								[11] b r				100	4
SET b, (HL)	$(HL)_b \leftarrow 1$	•	•	•	•	•	•	11 001 011	2	4	15	101	5
												110	6
								[11] b 110				111	7
SET b, (IX + d)	$(IX+d)_b \leftarrow 1$	•	•	•	•	•	•	11 011 101	4	6	23		
								11 001 011					
								← d →					
								[11] b 110					
SET b, (IY + d)	$(IY+d)_b \leftarrow 1$	•	•	•	•	•	•	11 111 101	4	6	23		
								11 001 011					
								← d →					
								[11] b 110					
RES b, s	$s_b \leftarrow 0$ s ≡ r, (HL), (IX + d), (IY + d)							[10]				To form new OP-code replace [11] of SET b, m with [10]. Flags and time states for SET instruction.	

Notes: The notation s_b indicates bit b (0 to 7) or location s.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ = flag is affected according to the result of the operation.

BIT SET, RESET AND TEST GROUP
TABLE 3.3-8

Mnemonic	Symbolic Operation	Flags						OP-Code	No. of Bytes	No. of M Cycles	No. of T States	Comments
		C	Z	P/V	S	N	H	76 543 210				
JP nn	PC←nn	•	•	•	•	•	•	11 000 011 ← n → ← n →	3	3	10	
JP cc,nn	If condition cc is true PC←nn, otherwise continue	•	•	•	•	•	•	11 cc 010 ← n → ← n →	3	3	10	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
JR e	PC←PC+e	•	•	•	•	•	•	00 011 000 ← e-2 →	2	3	12	
JR C,e	If C=0 continue	•	•	•	•	•	•	00 111 000 ← e-2 →	2	2	7	If condition not met
	If C=1 PC←PC+e								2	3	12	If condition is met
JR NC,e	If C=1 continue	•	•	•	•	•	•	00 110 000 ← e-2 →	2	2	7	If condition not met
	If C=0 PC←PC+e								2	3	12	If condition is met
JR Z,e	If Z=0 continue	•	•	•	•	•	•	00 101 000 ← e-2 →	2	2	7	If condition not met
	If Z=1 PC←PC+e								2	3	12	If condition is met
JR NZ,e	If Z=1 continue	•	•	•	•	•	•	00 100 000 ← e-2 →	2	2	7	If condition not met
	If Z=0 PC←PC+e								2	3	12	If condition is met
JP (HL)	PC←HL	•	•	•	•	•	•	11 101 001	1	1	4	
JP (IX)	PC←IX	•	•	•	•	•	•	11 011 101 11 101 001	2	2	8	
JP (IY)	PC←IY	•	•	•	•	•	•	11 111 101 11 101 001	2	2	8	
DJNZ,e	B←B−1 If B=0 continue	•	•	•	•	•	•	00 010 000 ← e-2 →	2	2	8	If B=0
	If B≠0 PC←PC+e								2	3	13	If B≠0

Notes: e represents the extension in the relative addressing mode.

e is a signed two's complement number in the range <−126, 129>

e−2 in the op-code provides an effective address of pc +e as PC is incremented by 2 prior to the addition of e.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
‡ = flag is affected according to the result of the operation.

JUMP GROUP
TABLE 3.3-9

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	OP-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
CALL nn	$(SP-1) \leftarrow PC_H$	•	•	•	•	•	•	11 001 101	3	5	17	
	$(SP-2) \leftarrow PC_L$							← n →				
	$PC \leftarrow nn$							← n →				
CALL cc, nn	If condition cc is false	•	•	•	•	•	•	11 cc 100	3	3	10	If cc is false
	continue, otherwise							← n →				
	same as CALL nn							← n →	3	5	17	If cc is true
RET	$PC_L \leftarrow (SP)$	•	•	•	•	•	•	11 001 001	1	3	10	
	$PC_H \leftarrow (SP+1)$											
RET cc	If condition cc is false	•	•	•	•	•	•	11 cc 000	1	1	5	If cc is false
	continue, otherwise								1	3	11	If cc is true
	same as RET											
RETI	Return from	•	•	•	•	•	•	11 101 101	2	4	14	
	interrupt							01 001 101				
RETN	Return from non	•	•	•	•	•	•	11 101 101	2	4	14	
	maskable interrupt							01 000 101				
RST p	$(SP-1) \leftarrow PC_H$	•	•	•	•	•	•	11 t 111	1	3	11	
	$(SP-2) \leftarrow PC_L$											
	$PC_H \leftarrow 0$											
	$PC_L \leftarrow P$											

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

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown
↕ = flag is affected according to the result of the operation.

CALL AND RETURN GROUP
TABLE 3.3-10

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	OP-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
IN A,(n)	A←(n)	•	•	•	•	•	•	11 011 011 ← n →	2	3	11	n to $A_0 - A_7$ Acc to $A_8 - A_{15}$
IN r,(C)	r←(C) If r=110 only the flags will be affected	•	↕ ①	P	↕	0	0	11 101 101 01 r 000	2	3	12	C to $A_0 - A_7$ B to $A_8 - A_{15}$
INI	(HL)←(C) B←B−1 HL←HL+1	•	↕	X	X	1	X	11 101 101 10 100 010	2	4	16	C to $A_0 - A_7$ B to $A_8 - A_{15}$
INIR	(HL)←(C) B←B−1 HL←HL+1 Repeat until B=0	•	1 ①	X	X	1	X	11 101 101 10 110 010	2 2	5 If B≠0 4 If B=0	21 16	C to $A_0 - A_7$ B to $A_8 - A_{15}$
IND	(HL)←(C) B←B−1 HL←HL−1	•	↕	X	X	1	X	11 101 101 10 101 010	2	4	16	C to $A_0 - A_7$ B to $A_8 - A_{15}$
INDR	(HL)←(C) B←B−1 HL←HL−1 Repeat until B=0	•	1	X	X	1	X	11 101 101 10 111 010	2 2	5 If B≠0 4 If B=0	21 16	C to $A_0 - A_7$ B to $A_8 - A_{15}$
OUT (n),A	(n)←A	•	•	•	•	•	•	11 010 011	2	3	11	n to $A_0 - A_7$ Acc to $A_8 - A_{15}$
OUT (C),r	(C)←r	•	• ①	•	•	•	•	11 101 101 01 r 001	2	3	12	C to $A_0 - A_7$ B to $A_8 - A_{15}$
OUTI	(C)←(HL) B←B−1 HL←HL+1	•	↕	X	X	1	X	11 101 101 10 100 011	2	4	16	C to $A_0 - A_7$ B to $A_8 - A_{15}$
OTIR	(C)←(HL) B←B−1 HL←HL+1 Repeat until B=0	•	1 ①	X	X	1	X	11 101 101 10 110 011	2 2	5 If B≠0 4 If B=0	21 16	C to $A_0 - A_7$ B to $A_8 - A_{15}$
OUTD	(C)←(HL) B←B−1 HL←HL−1	•	↕	X	X	1	X	11 101 101 10 101 011	2	4	16	C to $A_0 - A_7$ B to $A_8 - A_{15}$
OTDR	(C)←(HL) B←B−1 HL←HL−1 Repeat until B=0	•	1	X	X	1	X	11 101 101 10 111 011	2 2	5 If B≠0 4 If B=0	21 16	C to $A_0 - A_7$ B to $A_8 - A_{15}$

Notes: ① If the result of B-1 is zero the Z flag is set, otherwise it is reset.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ = flag is affected according to the result of the operation.

INPUT AND OUTPUT GROUP
TABLE 3.3-11

3.4 FLAGS

Each of the two Z-80 CPU Flag registers contains six bits of information which are set or reset by various CPU operations. Four of these bits are testable; that is, they are used as conditions for jump, call or return instructions. For example a jump may be desired only if a specific bit in the flag register is set. The four testable flag bits are:

1) **Carry Flag (C)** – This flag is the carry from the highest order bit of the accumulator. For example, the carry flag will be set during an add instruction where a carry from the highest bit of the accumulator is generated. This flag is also set if a borrow is generated during a subtraction instruction. The shift and rotate instructions also affect this bit.

2) **Zero Flag (Z)** – This flag is set if the result of the operation loaded a zero into the accumulator. Otherwise it is reset.

3) **Sign Flag (S)** – This flag is intended to be used with signed numbers and it is set if the result of the operation was negative. Since bit 7 (MSB) represents the sign of the number (A negative number has a 1 in bit 7), this flag stores the state of bit 7 in the accumulator.

4) **Parity/Overflow Flag (P/V)** – This dual purpose flag indicates the parity of the result in the accumulator when logical operations are performed (such as AND A, B) and it represents overflow when signed two's complement arithmetic operations are performed. The Z-80 overflow flag indicates that the two's complement number in the accumulator is in error since it has exceeded the maximum possible (+127) or is less than the minimum possible (−128) number than can be represented in two's complement notation. For example consider adding:

```
    +120 =    0111 1000
    +105 =    0110 1001
    -----     ---------
   C =  0     1110 0001 = −31 (wrong) Overflow has occurred
```

Here the result is incorrect. Overflow has occurred and yet there is no carry to indicate an error. For this case the overflow flag would be set. Also consider the addition of two negative numbers:

```
      −5 =    1111 1011
     −16 =    1111 0000
     ----     ---------
   C =  1     1110 1011 = −21 correct
```

Notice that the answer is correct but the carry is set so that this flag can not be used as an overflow indicator. In this case the overflow would not be set.

For logical operations (AND, OR, XOR) this flag is set if the parity of the result is even and it is reset if it is odd.

There are also two non-testable bits in the flag register. Both of these are used for BCD arithmetic. They are:

1) **Half carry (H)** – This is the BCD carry or borrow result from the least significant four bits of operation. When using the DAA (Decimal Adjust Instruction) this flag is used to correct the result of a previous packed decimal add or subtract.

2) **Add/Subtract Flag (N)** – Since the algorithm for correcting BCD operations is different for addition or subtraction, this flag is used to specify what type of instruction was executed last so that the DAA operation will be correct for either addition or subtraction.

The Flag register can be accessed by the programmer and its format is as follows:

S	Z	X	H	X	P/V	N	C

X means flag is indeterminate.

Table 3.4-1 lists how each flag bit is affected by various CPU instructions. In this table a '•' indicates that the instruction does not change the flag, an 'X' means that the flag goes to an indeterminate state, a '0' means that it is reset, a '1' means that it is set and the symbol '↕' indicates that it is set or reset according to the previous discussion. Note that any instruction not appearing in this table does not affect any of the flags.

Table 3.4-1 includes a few special cases that must be described for clarity. Notice that the block search instruction sets the Z flag if the last compare operation indicated a match between the source and the accumulator data. Also, the parity flag is set if the byte counter (register pair BC) is not equal to zero. This same use of the parity flag is made with the block move instructions. Another special case is during block input or output instructions, here the Z flag is used to indicate the state of register B which is used as a byte counter. Notice that when the I/O block transfer is complete, the zero flag will be reset to a zero (i.e. B = 0) while in the case of a block move command the parity flag is reset when the operation is complete. A final case is when the refresh or I register is loaded into the accumulator, the interrupt enable flip flop is loaded into the parity flag so that the complete state of the CPU can be saved at any time.

Instruction	C	Z	P/V	S	N	H	Comments
ADD A, s; ADC A, s	↕	↕	V	↕	0	↕	8-bit add or add with carry
SUB s; SBC A, s; CP s; NEG	↕	↕	V	↕	1	↕	8-bit subtract, subtract with carry, compare and negate accumulator
AND s	0	↕	P	↕	0	1	} Logical operations
OR s; XOR s	0	↕	P	↕	0	0	} And set's different flags
INC s	•	↕	V	↕	0	↕	8-bit increment
DEC m	•	↕	V	↕	1	↕	8-bit decrement
ADD DD, ss	↕	•	•	•	0	X	16-bit add
ADC HL, ss	↕	↕	V	↕	0	X	16-bit add with carry
SBC HL, ss	↕	↕	V	↕	1	X	16-bit subtract with carry
RLA; RLCA; RRA; RRCA;	↕	•	•	•	0	0	Rotate accumulator
RL m; RLC m; RR m; RRC m SLA m; SRA m; SRL m	↕	↕	P	↕	0	0	Rotate and shift location s
RLD; RRD	•	↕	P	↕	0	0	Rotate digit left and right
DAA	↕	↕	P	↕	•	↕	Decimal adjust accumulator
CPL	•	•	•	•	1	1	Complement accumulator
SCF	1	•	•	•	0	0	Set carry
CCF	↕	•	•	•	0	X	Complement carry
IN r, (C)	•	↕	P	↕	0	0	Input register indirect
INI; IND; OUTI; OUTD	•	↕	X	X	1	X	} Block input and output
INIR; INDR; OTIR; OTDR	•	1	X	X	1	X	} Z = 0 if B ≠ 0 otherwise Z = 1
LDI; LDD	•	X	↕	X	0	0	} Block transfer instructions
LDIR; LDDR	•	X	0	X	0	0	} P/V = 1 if BC ≠ 0, otherwise P/V = 0
CPI; CPIR; CPD; CPDR	•	↕	↕	X	1	X	Block search instructions Z = 1 if A = (HL), otherwise Z = 0 P/V = 1 if BC ≠ 0, otherwise P/V = 0
LD A, I; LD A, R	•	↕	IFF	↕	0	0	The content of the interrupt enable flip-flop (IFF) is copied into the P/V flag
BIT b, s	•	↕	X	X	0	1	The state of bit b of location s is copied into the Z flag
NEG	↕	↕	V	↕	1	↕	Negative accumulator

The following notation is used in this table:

Symbol	Operation
C	Carry/link flag. C = 1 if the operation produced a carry from the MSB of the operand or result.
Z	Zero flag. Z = 1 if the result of the operation is zero.
S	Sign flag. S = 1 if the MSB of the result is one.
P/V	Parity or overflow flag. Parity (P) and overflow (V) share the same flag. Logical operations affect this flag with the parity of the result while arithmetic operations affect this flag with the overflow of the result. If P/V holds parity, P/V = 1 if the result of the operation is even, P/V = 0 if result is odd. If P/V holds overflow, P/V = 1 if the result of the operation produced an overflow.
H	Half-carry flag. H = 1 if the add or subtract operation produced a carry into or borrow from into bit 4 of the accumulator.
N	Add/Subtract flag. N = 1 if the previous operation was a subtract.
	H and N flags are used in conjunction with the decimal adjust instruction (DAA) to properly correct the result into packed BCD format following addition or subtraction using operands with packed BCD format.
↕	The flag is affected according to the result of the operation.
•	The flag is unchanged by the operation.
0	The flag is reset by the operation.
1	The flag is set by the operation.
X	The flag is a "don't care."
V	P/V flag affected according to the overflow result of the operation.
P	P/V flag affected according to the parity result of the operation.
r	Any one of the CPU registers A, B, C, D, E, H, L.
s	Any 8-bit location for all the addressing modes allowed for the particular instruction.
ss	Any 16-bit location for all the addressing modes allowed for that instruction.
ii	Any one of the two index registers IX or IY.
R	Refresh counter.
n	8-bit value in range <0, 255>
nn	16-bit value in range <0, 65535>
m	Any 8-bit location for all the addressing modes allowed for the particular instruction.

SUMMARY OF FLAG OPERATION

TABLE 3.4-1

4.0 INTERRUPT RESPONSE

The purpose of an interrupt is to allow peripheral devices to suspend CPU operation in an orderly manner and force the CPU to start a peripheral service routine. Usually this service routine is involved with the exchange of data, or status and control information, between the CPU and the peripheral. Once the service routine is completed, the CPU returns to the operation from which it was interrupted.

INTERRUPT ENABLE – DISABLE

The Z-80 CPU has two interrupt inputs, a software maskable interrupt and a non maskable interrupt. The non maskable interrupt (NMI) can *not* be disabled by the programmer and it will be accepted whenever a peripheral device requests it. This interrupt is generally reserved for very important functions that must be serviced whenever they occur, such as an impending power failure. The maskable interrupt (INT) can be selectively enable or disabled by the programmer. This allows the programmer to disable the interrupt during periods where his program has timing constraints that do not allow it to be interrupted. In the Z-80 CPU there is an enable flip flop (called IFF) that is set or reset by the programmer using the Enable Interrupt (EI) and Disable Interrupt (DI) instructions. When the IFF is reset, an interrupt can not be accepted by the CPU.

Actually, for purposes that will be subsequently explained, there are two enable flip flops, called IFF_1 and IFF_2.

IFF_1	IFF_2
Actually disables interrupts from being accepted.	Temporary storage location for IFF_1.

The state of IFF_1 is used to actually inhibit interrupts while IFF_2 is used as a temporary storage location for IFF_1. The purpose of storing the IFF_1 will be subsequently explained.

A reset to the CPU will force both IFF_1 and IFF_2 to the reset state so that interrupts are disabled. They can then be enabled by an EI instruction at any time by the programmer. When an EI instruction is executed, any pending interrupt request will not be accepted until after the instruction following EI has been executed. This single instruction delay is necessary for cases when the following instruction is a return instruction and interrupts must not be allowed until the return has been completed. The EI instruction sets both IFF_1 and IFF_2 to the enable state. When an interrupt is accepted by the CPU, both IFF_1 and IFF_2 are automatically reset, inhibiting further interrupts until the programmer wishes to issue a new EI instruction. Note that for all of the previous cases, IFF_1 and IFF_2 are always equal.

The purpose of IFF_2 is to save the status of IFF_1 when a non maskable interrupt occurs. When a non maskable interrupt is accepted, IFF_1 is reset to prevent further interrupts until reenabled by the programmer. Thus, after a non maskable interrupt has been accepted, maskable interrupts are disabled but the previous state of IFF_1 has been saved so that the complete state of the CPU just prior to the non maskable interrupt can be restored at any time. When a Load Register A with Register I (LD A, I) instruction or a Load Register A with Register R (LD A, R) instruction is executed, the state of IFF_2 is copied into the parity flag where it can be tested or stored.

A second method of restoring the status of IFF_1 is thru the execution of a Return From Non Maskable Interrupt (RETN) instruction. Since this instruction indicates that the non maskable interrupt service routine is complete, the contents of IFF_2 are now copied back into IFF_1, so that the status of IFF_1 just prior to the acceptance of the non maskable interrupt will be restored automatically.

Figure 4.0-1 is a summary of the effect of different instructions on the two enable flip flops.

Action	IFF_1	IFF_2	
CPU Reset	0	0	
DI	0	0	
EI	1	1	
LD A, I	•	•	$IFF_2 \rightarrow$ Parity flag
LD A, R	•	•	$IFF_2 \rightarrow$ Parity flag
Accept NMI	0	•	
RETN	IFF_2	•	$IFF_2 \rightarrow IFF_1$

"•" indicates no change

FIGURE 4.0-1
INTERRUPT ENABLE/DISABLE FLIP FLOPS

CPU RESPONSE

Non Maskable

A nonmaskable interrupt will be accepted at all times by the CPU. When this occurs, the CPU ignores the next instruction that it fetches and instead does a restart to location 0066H. Thus, it behaves exactly as if it had received a restart instruction but, it is to a location that is not one of the 8 software restart locations. A restart is merely a call to a specific address in page 0 of memory.

Maskable

The CPU can be programmed to respond to the maskable interrupt in any one of three possible modes.

Mode 0

This mode is identical to the 8080A interrupt response mode. With this mode, the interrupting device can place any instruction on the data bus and the CPU will execute it. Thus, the interrupting device provides the next instruction to be executed instead of the memory. Often this will be a restart instruction since the interrupting device only need supply a single byte instruction. Alternatively, any other instruction such as a 3 byte call to any location in memory could be executed.

The number of clock cycles necessary to execute this instruction is 2 more than the normal number for the instruction. This occurs since the CPU automatically adds 2 wait states to an interrupt response cycle to allow sufficient time to implement an external daisy chain for priority control.

After the application of RESET the CPU will automatically enter interrupt Mode 0.

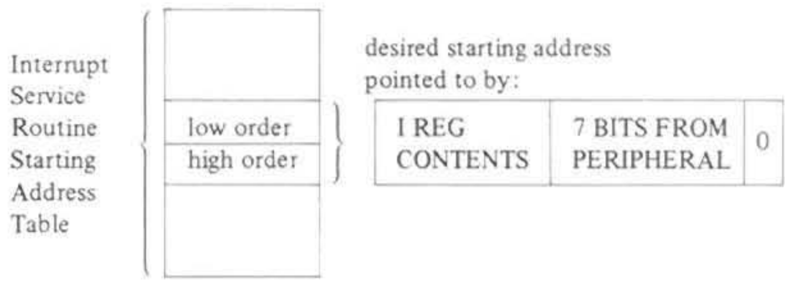
Mode 1

When this mode has been selected by the programmer, the CPU will respond to an interrupt by executing a restart to location 0038H. Thus the response is identical to that for a non maskable interrupt except that the call location is 0038H instead of 0066H. Another difference is that the number of cycles required to complete the restart instruction is 2 more than normal due to the two added wait states.

Mode 2

This mode is the most powerful interrupt response mode. With a single 8 bit byte from the user an indirect call can be made to any memory location.

With this mode the programmer maintains a table of 16 bit starting addresses for every interrupt service routine. This table may be located anywhere in memory. When an interrupt is accepted, a 16 bit pointer must be formed to obtain the desired interrupt service routine starting address from the table. The upper 8 bits of this pointer is formed from the contents of the I register. The I register must have been previously loaded with the desired value by the programmer, i.e. LD I, A. Note that a CPU reset clears the I register so that it is initialized to zero. The lower eight bits of the pointer must be supplied by the interrupting device. Actually, only 7 bits are required from the interrupting device as the least significant bit must be a zero. This is required since the pointer is used to get two adjacent bytes to form a complete 16 bit service routine starting address and the addresses must always start in even locations.



The first byte in the table is the least significant (low order) portion of the address. The programmer must obviously fill this table in with the desired addresses before any interrupts are to be accepted.

Note that this table can be changed at any time by the programmer (if it is stored in Read/Write Memory) to allow different peripherals to be serviced by different service routines.

Once the interrupting devices supplies the lower portion of the pointer, the CPU automatically pushes the program counter onto the stack, obtains the starting address from the table and does a jump to this address. This mode of response requires 19 clock periods to complete (7 to fetch the lower 8 bits from the interrupting device, 6 to save the program counter, and 6 to obtain the jump address.)

Appendix

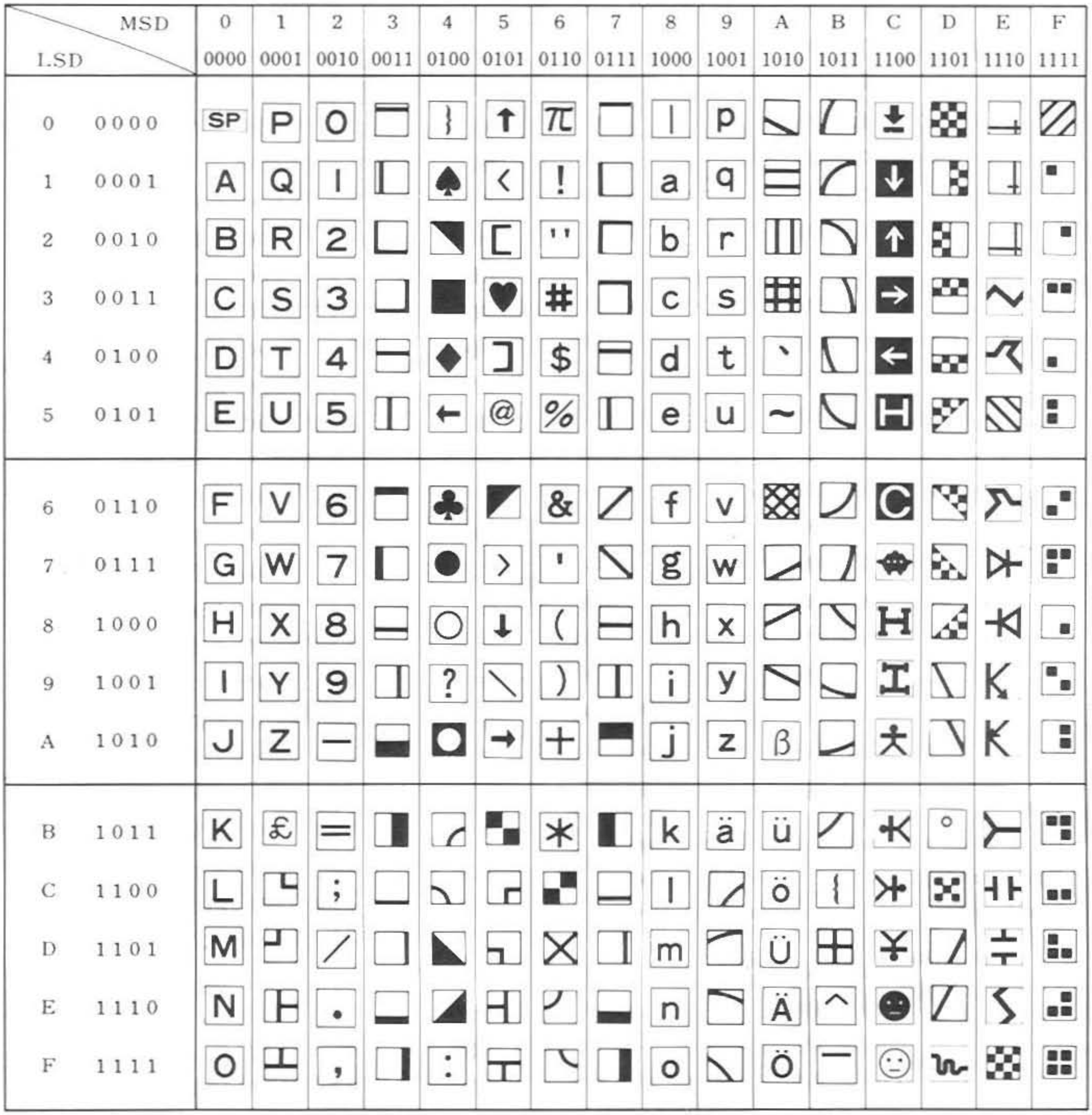
A.1 ASCII Code Table

The following are the ASCII codes for characters:

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	[graphic]	[graphic]	}	[graphic]	q	n	[graphic]	[graphic]	[graphic]	[graphic]
1	0001		↓	!	1	A	Q	[graphic]	[graphic]	[graphic]	[graphic]	a	[graphic]	[graphic]	[graphic]	♠	●
2	0010		↑	"	2	B	R	[graphic]	[graphic]	[graphic]	e	z	Ü	[graphic]	[graphic]	[graphic]	[graphic]
3	0011		→	#	3	C	S	[graphic]	[graphic]	[graphic]	`	w	m	[graphic]	[graphic]	[graphic]	♥
4	0100		←	$	4	D	T	[graphic]	[graphic]	[graphic]	~	s	[graphic]	[graphic]	[graphic]	[graphic]	[graphic]
5	0101		[graphic]	%	5	E	U	[graphic]	[graphic]	[graphic]	[graphic]	u	[graphic]	[graphic]	[graphic]	[graphic]	[graphic]
6	0110		[graphic]	&	6	F	V	[graphic]	[graphic]	[graphic]	t	i	[graphic]	→	[graphic]	[graphic]	[graphic]
7	0111			'	7	G	W	[graphic]	[graphic]	[graphic]	g	[graphic]	o	[graphic]	[graphic]	[graphic]	[graphic]
8	1000			(	8	H	X	☺	[graphic]	[graphic]	h	ö	[graphic]	[graphic]	[graphic]	[graphic]	♣
9	1001			)	9	I	Y	[graphic]	[graphic]	[graphic]	[graphic]	k	Ä	[graphic]	[graphic]	[graphic]	[graphic]
A	1010			*	:	J	Z	[graphic]	[graphic]	[graphic]	b	f	ö	[graphic]	[graphic]	[graphic]	◆
B	1011			+	;	K	[	[graphic]	°	^	x	v	ä	[graphic]	[graphic]	[graphic]	£
C	1100			,	<	L	\	[graphic]	[graphic]	[graphic]	d	[graphic]	[graphic]	[graphic]	[graphic]	[graphic]	↓
D	1101	CR		−	=	M	]	[graphic]	[graphic]	[graphic]	r	ü	y	[graphic]	[graphic]	[graphic]	[graphic]
E	1110			.	>	N	↑	[graphic]	[graphic]	[graphic]	p	β	[graphic]	[graphic]	[graphic]	[graphic]	[graphic]
F	1111			/	?	O	←	÷		[graphic]	c	j	[graphic]	[graphic]	[graphic]	[graphic]	π

A.2 Display Code Table

The following are the display codes of the MZ-80A.



A. 3 Mnemonic Codes and Corresponding Object Codes

(Mnemonic codes are arranged in alphabetic order.)

OP-Code	Mnemonic
8E	ADC A, (HL)
DD8E*05*	ADC A, (IX+d)
FD8E*05*	ADC A, (IY+d)
8F	ADC A, A
88	ADC A, B
89	ADC A, C
8A	ADC A, D
8B	ADC A, E
8C	ADC A, H
8D	ADC A, L
CE*20*	ADC A, n
ED4A	ADC HL, BC
ED5A	ADC HL, DE
ED6A	ADC HL, HL
ED7A	ADC HL, SP
86	ADD A, (HL)
DD86*05*	ADD A, (IX+d)
FD86*05*	ADD A, (IY+d)
87	ADD A, A
80	ADD A, B
81	ADD A, C
82	ADD A, D
83	ADD A, E
84	ADD A, H
85	ADD A, L
C6*20*	ADD A, n
09	ADD HL, BC
19	ADD HL, DE
29	ADD HL, HL
39	ADD HL, SP
DD09	ADD IX, BC
DD19	ADD IX, DE
DD29	ADD IX, IX
DD39	ADD IX, SP
FD09	ADD IY, BC
FD19	ADD IY, DE
FD29	ADD IY, IY
FD39	ADD IY, SP

Note

nn, n, d and e in the operands of each mnemonic code represent constant data. The example values set forth below are used for these constants in this table.

nn = 584H
n = 20H
d = 5
e = 30H

Data codes represented by example values are shown in italic and underlined.

OP-Code	Mnemonic
A6	AND (HL)
DDA6*05*	AND (IX+d)
FDA6*05*	AND (IY+d)
A7	AND A
A0	AND B
A1	AND C
A2	AND D
A3	AND E
A4	AND H
A5	AND L
E6*20*	AND n
CB46	BIT 0, (HL)
DDCB*05*46	BIT 0, (IX+d)
FDCB*05*46	BIT 0, (IY+d)
CB47	BIT 0, A
CB40	BIT 0, B
CB41	BIT 0, C
CB42	BIT 0, D
CB43	BIT 0, E
CB44	BIT 0, H
CB45	BIT 0, L
CB4E	BIT 1, (HL)
DDCB*05*4E	BIT 1, (IX+d)
FDCB*05*4E	BIT 1, (IY+d)
CB4F	BIT 1, A
CB48	BIT 1, B
CB49	BIT 1, C
CB4A	BIT 1, D
CB4B	BIT 1, E
CB4C	BIT 1, H
CB4D	BIT 1, L
CB56	BIT 2, (HL)
DDCB*05*56	BIT 2, (IX+d)
FDCB*05*56	BIT 2, (IY+d)
CB57	BIT 2, A
CB50	BIT 2, B
CB51	BIT 2, C
CB52	BIT 2, D
CB53	BIT 2, E

OP-Code	Mnemonic
CB54	BIT 2, H
CB55	BIT 2, L
CB5E	BIT 3, (HL)
DDCB*05*5E	BIT 3, (IX+d)
FDCB*05*5E	BIT 3, (IY+d)
CB5F	BIT 3, A
CB58	BIT 3, B
CB59	BIT 3, C
CB5A	BIT 3, D
CB5B	BIT 3, E
CB5C	BIT 3, H
CB5D	BIT 3, L
CB66	BIT 4, (HL)
DDCB*05*66	BIT 4, (IX+d)
FDCB*05*66	BIT 4, (IY+d)
CB67	BIT 4, A
CB60	BIT 4, B
CB61	BIT 4, C
CB62	BIT 4, D
CB63	BIT 4, E
CB64	BIT 4, H
CB65	BIT 4, L
CB6E	BIT 5, (HL)
DDCB*05*6E	BIT 5, (IX+d)
FDCB*05*6E	BIT 5, (IY+d)
CB6F	BIT 5, A
CB68	BIT 5, B
CB69	BIT 5, C
CB6A	BIT 5, D
CB6B	BIT 5, E
CB6C	BIT 5, H
CB6D	BIT 5, L
CB76	BIT 6, (HL)
DDCB*05*76	BIT 6, (IX+d)
FDCB*05*76	BIT 6, (IY+d)
CB77	BIT 6, A
CB70	BIT 6, B
CB71	BIT 6, C
CB72	BIT 6, D
CB73	BIT 6, E

OP-Code	Mnemonic
CB74	BIT 6,H
CB75	BIT 6,L
CB7E	BIT 7,(HL)
DDCB057E	BIT 7,(IX+d)
FDCB057E	BIT 7,(IY+d)
CB7F	BIT 7,A
CB78	BIT 7,B
CB79	BIT 7,C
CB7A	BIT 7,D
CB7B	BIT 7,E
CB7C	BIT 7,H
CB7D	BIT 7,L
DC8405	CALL C,nn
FC8405	CALL M,nn
D48405	CALL NC,nn
CD8405	CALL nn
C48405	CALL NZ,nn
F48405	CALL P,nn
EC8405	CALL PE,nn
E48405	CALL PO,nn
CC8405	CALL Z,nn
3F	CCF
BE	CP (HL)
DDBE05	CP (IX+d)
FDBE05	CP (IY+d)
BF	CP A
B8	CP B
B9	CP C
BA	CP D
BB	CP E
BC	CP H
BD	CP L
FE20	CP n
EDA9	CPD
EDB9	CPDR
EDA1	CPI

OP-Code	Mnemonic
EDB1	CPIR
2F	CPL
27	DAA
35	DEC (HL)
DD3505	DEC (IX+d)
FD3505	DEC (IY+d)
3D	DEC A
05	DEC B
0B	DEC BC
0D	DEC C
15	DEC D
1B	DEC DE
1D	DEC E
25	DEC H
2B	DEC HL
DD2B	DEC IX
FD2B	DEC IY
2D	DEC L
3B	DEC SP
F3	DI
102E	DJNZ e
FB	EI
E3	EX (SP),HL
DDE3	EX (SP),IX
FDE3	EX (SP),IY
08	EX AF,AF'
EB	EX DE,HL
D9	EXX
76	HALT
ED46	IM 0
ED56	IM 1

OP-Code	Mnemonic
ED5E	IM 2
ED78	IN A,(C)
DB*20*	IN A,(n)
ED40	IN B,(C)
ED48	IN C,(C)
ED50	IN D,(C)
ED58	IN E,(C)
ED60	IN H,(C)
ED68	IN L,(C)
34	INC (HL)
DD34*05*	INC (IX+d)
FD34*05*	INC (IY+d)
3C	INC A
04	INC B
03	INC BC
0C	INC C
14	INC D
13	INC DE
1C	INC E
24	INC H
23	INC HL
DD23	INC IX
FD23	INC IY
2C	INC L
33	INC SP
EDAA	IND
EDBA	INDR
EDA2	INI
EDB2	INIR
E9	JP (HL)
DDE9	JP (IX)
FDE9	JP (IY)
DA*8405*	JP C,nn
FA*8405*	JP M,nn
D2*8405*	JP NC,nn
C3*8405*	JP nn

OP-Code	Mnemonic
C2*8405*	JP NZ,nn
F2*8405*	JP P,nn
EA*8405*	JP PE,nn
E2*8405*	JP PO,nn
CA*8405*	JP Z,nn
38*2E*	JR C,e
18*2E*	JR e
30*2E*	JR NC,e
20*2E*	JR NZ,e
28*2E*	JR Z,e
02	LD (BC),A
12	LD (DE),A
77	LD (HL),A
70	LD (HL),B
71	LD (HL),C
72	LD (HL),D
73	LD (HL),E
74	LD (HL),H
75	LD (HL),L
36*20*	LD (HL),n
DD77*05*	LD (IX+d),A
DD70*05*	LD (IX+d),B
DD71*05*	LD (IX+d),C
DD72*05*	LD (IX+d),D
DD73*05*	LD (IX+d),E
DD74*05*	LD (IX+d),H
DD75*05*	LD (IX+d),L
DD36*0520*	LD (IX+d),n
FD77*05*	LD (IY+d),A
FD70*05*	LD (IY+d),B
FD71*05*	LD (IY+d),C
FD72*05*	LD (IY+d),D
FD73*05*	LD (IY+d),E
FD74*05*	LD (IY+d),H
FD75*05*	LD (IY+d),L
FD36*0520*	LD (IY+d),n
32*8405*	LD (nn),A
ED43*8405*	LD (nn),BC

OP-Code	Mnemonic
ED538405	LD (nn),DE
228405	LD (nn),HL
DD228405	LD (nn),IX
FD228405	LD (nn),IY
ED738405	LD (nn),SP
0A	LD A,(BC)
1A	LD A,(DE)
7E	LD A,(HL)
DD7E05	LD A,(IX+d)
FD7E05	LD A,(IY+d)
3A 8405	LD A,(nn)
7F	LD A,A
78	LD A,B
79	LD A,C
7A	LD A,D
7B	LD A,E
7C	LD A,H
ED57	LD A,I
7D	LD A,L
3E20	LD A,n
46	LD B,(HL)
DD4605	LD B,(IX+d)
FD4605	LD B,(IY+d)
47	LD B,A
40	LD B,B
41	LD B,C
42	LD B,D
43	LD B,E
44	LD B,H
45	LD B,L
0620	LD B,n
ED4B8405	LD BC,(nn)
018405	LD BC,nn
4E	LD C,(HL)
DD4E05	LD C,(IX+d)
FD4E05	LD C,(IY+d)
4F	LD C,A
48	LD C,B
49	LD C,C
4A	LD C,D

OP-Code	Mnemonic
4B	LD C,E
4C	LD C,H
4D	LD C,L
0E20	LD C,n
56	LD D,(HL)
DD5605	LD D,(IX+d)
FD5605	LD D,(IY+d)
57	LD D,A
50	LD D,B
51	LD D,C
52	LD D,D
53	LD D,E
54	LD D,H
55	LD D,L
1620	LD D,n
ED5B8405	LD DE,(nn)
118405	LD DE,nn
5E	LD E,(HL)
DD5E05	LD E,(IX+d)
FD5E05	LD E,(IY+d)
5F	LD E,A
58	LD E,B
59	LD E,C
5A	LD E,D
5B	LD E,E
5C	LD E,H
5D	LD E,L
1E20	LD E,n
66	LD H,(HL)
DD6605	LD H,(IX+d)
FD6605	LD H,(IY+d)
67	LD H,A
60	LD H,B
61	LD H,C
62	LD H,D
63	LD H,E
64	LD H,H
65	LD H,L
2620	LD H,n
2A8405	LD H,(nn)

OP-Code	Mnemonic
218405	LD HL,nn
ED47	LD I,A
DD2A8405	LD IX,(nn)
DD218405	LD IX,nn
FD2A8405	LD IY,(nn)
FD218405	LD IY,nn
6E	LD L,(HL)
DD6E05	LD L,(IX+d)
FD6E05	LD L,(IY+d)
6F	LD L,A
68	LD L,B
69	LD L,C
6A	LD L,D
6B	LD L,E
6C	LD L,H
6D	LD L,L
2E20	LD L,n
ED7B8405	LD SP,(nn)
F9	LD SP,HL
DDF9	LD SP,IX
FDF9	LD SP,IY
318405	LD SP,nn
EDA8	LDD
EDB8	LDDR
EDA0	LDI
EDB0	LDIR
ED44	NEG
00	NOP
B6	OR (HL)
DDB605	OR (IX+d)
FDB605	OR (IY+d)
B7	OR A
B0	OR B
B1	OR C
B2	OR D
B3	OR E

OP-Code	Mnemonic
B4	OR H
B5	OR L
F620	OR n
EDBB	OTDR
EDB3	OTIR
ED79	OUT (C),A
ED41	OUT (C),B
ED49	OUT (C),C
ED51	OUT (C),D
ED59	OUT (C),E
ED61	OUT (C),H
ED69	OUT (C),L
D320	OUT (n),A
EDAB	OUTD
EDA3	OUTI
F1	POP AF
C1	POP BC
D1	POP DE
E1	POP HL
DDE1	POP IX
FDE1	POP IY
F5	PUSH AF
C5	PUSH BC
D5	PUSH DE
E5	PUSH HL
DDE5	PUSH IX
FDE5	PUSH IY
CB86	RES 0,(HL)
DDCB0586	RES 0,(IX+d)
FDCB0586	RES 0,(IY+d)
CB87	RES 0,A
CB80	RES 0,B
CB81	RES 0,C
CB82	RES 0,D
CB83	RES 0,E
CB84	RES 0,H

OP-Code	Mnemonic
CB85	RES 0,L
CB8E	RES 1,(HL)
DDCB*05*8E	RES 1,(IX+d)
FDCB*05*8E	RES 1,(IY+d)
CB8F	RES 1,A
CB88	RES 1,B
CB89	RES 1,C
CB8A	RES 1,D
CB8B	RES 1,E
CB8C	RES 1,H
CB8D	RES 1,L
CB96	RES 2,(HL)
DDCB*05*96	RES 2,(IX+d)
FDCB*05*96	RES 2,(IY+d)
CB97	RES 2,A
CB90	RES 2,B
CB91	RES 2,C
CB92	RES 2,D
CB93	RES 2,E
CB94	RES 2,H
CB95	RES 2,L
CB9E	RES 3,(HL)
DDCB*05*9E	RES 3,(IX+d)
FDCB*05*9E	RES 3,(IY+d)
CB9F	RES 3,A
CB98	RES 3,B
CB99	RES 3,C
CB9A	RES 3,D
CB9B	RES 3,E
CB9C	RES 3,H
CB9D	RES 3,L
CBA6	RES 4,(HL)
DDCB*05*A6	RES 4,(IX+d)
FDCB*05*A6	RES 4,(IY+d)
CBA7	RES 4,A
CBA0	RES 4,B
CBA1	RES 4,C
CBA2	RES 4,D
CBA3	RES 4,E
CBA4	RES 4,H

OP-Code	Mnemonic
CBA5	RES 4,L
CBAE	RES 5,(HL)
DDCB*05*AE	RES 5,(IX+d)
FDCB*05*AE	RES 5,(IY+d)
CBAF	RES 5,A
CBA8	RES 5,B
CBA9	RES 5,C
CBAA	RES 5,D
CBAB	RES 5,E
CBAC	RES 5,H
CBAD	RES 5,L
CBB6	RES 6,(HL)
DDCB*05*B6	RES 6,(IX+d)
FDCB*05*B6	RES 6,(IY+d)
CBB7	RES 6,A
CBB0	RES 6,B
CBB1	RES 6,C
CBB2	RES 6,D
CBB3	RES 6,E
CBB4	RES 6,H
CBB5	RES 6,L
CBBE	RES 7,(HL)
DDCB*05*BE	RES 7,(IX+d)
FDCB*05*BE	RES 7,(IY+d)
CBBF	RES 7,A
CBB8	RES 7,B
CBB9	RES 7,C
CBBA	RES 7,D
CBBB	RES 7,E
CBBC	RES 7,H
CBBD	RES 7,L
C9	RET
D8	RET C
F8	RET M
D0	RET NC
C0	RET NZ
F0	RET P
E8	RET PE
E0	RET PO

OP-Code	Mnemonic
C8	RET Z
ED4D	RETI
ED45	RETN
CB16	RL (HL)
DDCB0516	RL (IX+d)
FDCB0516	RL (IY+d)
CB17	RL A
CB10	RL B
CB11	RL C
CB12	RL D
CB13	RL E
CB14	RL H
CB15	RL L
17	RLA
CB06	RLC (HL)
DDCB0506	RLC (IX+d)
FDCB0506	RLC (IY+d)
CB07	RLC A
CB00	RLC B
CB01	RLC C
CB02	RLC D
CB03	RLC E
CB04	RLC H
CB05	RLC L
07	RLCA
ED6F	RLD
CB1E	RR (HL)
DDCB051E	RR (IX+d)
FDCB051E	RR (IY+d)
CB1F	RR A
CB18	RR B
CB19	RR C
CB1A	RR D
CB1B	RR E
CB1C	RR H
CB1D	RR L
1F	RRA

OP-Code	Mnemonic
CB0E	RRC (HL)
DDCB050E	RRC (IX+d)
FDCB050E	RRC (IY+d)
CB0F	RRC A
CB08	RRC B
CB09	RRC C
CB0A	RRC D
CB0B	RRC E
CB0C	RRC H
CB0D	RRC L
0F	RRCA
ED67	RRD
C7	RST 00H
CF	RST 08H
D7	RST 10H
DF	RST 18H
E7	RST 20H
EF	RST 28H
F7	RST 30H
FF	RST 38H
9E	SBC A,(HL)
DD9E05	SBC A,(IX+d)
FD9E05	SBC A,(IY+d)
9F	SBC A,A
98	SBC A,B
99	SBC A,C
9A	SBC A,D
9B	SBC A,E
9C	SBC A,H
9D	SBC A,L
DE20	SBC A,n
ED42	SBC HL,BC
ED52	SBC HL,DE
ED62	SBC HL,HL
ED72	SBC HL,SP
37	SCF

OP-Code	Mnemonic	OP-Code	Mnemonic
CBC6	SET 0,(HL)	CBE6	SET 4,(HL)
DDCB05C6	SET 0,(IX+d)	DDCB05E6	SET 4,(IX+d)
FDCB05C6	SET 0,(IY+d)	FDCB05E6	SET 4,(IY+d)
CBC7	SET 0,A	CBE7	SET 4,A
CBC0	SET 0,B	CBE0	SET 4,B
CBC1	SET 0,C	CBE1	SET 4,C
CBC2	SET 0,D	CBE2	SET 4,D
CBC3	SET 0,E	CBE3	SET 4,E
CBC4	SET 0,H	CBE4	SET 4,H
CBC5	SET 0,L	CBE5	SET 4,L
CBCE	SET 1,(HL)	CBEE	SET 5,(HL)
DDCB05CE	SET 1,(IX+d)	DDCB05EE	SET 5,(IX+d)
FDCB05CE	SET 1,(IY+d)	FDCB05EE	SET 5,(IY+d)
CBCF	SET 1,A	CBEF	SET 5,A
CBC8	SET 1,B	CBE8	SET 5,B
CBC9	SET 1,C	CBE9	SET 5,C
CBCA	SET 1,D	CBEA	SET 5,D
CBCB	SET 1,E	CBEB	SET 5,E
CBCC	SET 1,H	CBEC	SET 5,H
CBCD	SET 1,L	CBED	SET 5,L
CBD6	SET 2,(HL)	CBF6	SET 6,(HL)
DDCB05D6	SET 2,(IX+d)	DDCB05F6	SET 6,(IX+d)
FDCB05D6	SET 2,(IY+d)	FDCB05F6	SET 6,(IY+d)
CBD7	SET 2,A	CBF7	SET 6,A
CBD0	SET 2,B	CBF0	SET 6,B
CBD1	SET 2,C	CBF1	SET 6,C
CBD2	SET 2,D	CBF2	SET 6,D
CBD3	SET 2,E	CBF3	SET 6,E
CBD4	SET 2,H	CBF4	SET 6,H
CBD5	SET 2,L	CBF5	SET 6,L
CBD8	SET 3,B	CBFE	SET 7,(HL)
CBDE	SET 3,(HL)	DDCB05FE	SET 7,(IX+d)
DDCB05DE	SET 3,(IX+d)	FDCB05FE	SET 7,(IY+d)
FDCB05DE	SET 3,(IY+d)	CBFF	SET 7,A
CBDF	SET 3,A	CBF8	SET 7,B
CBD9	SET 3,C	CBF9	SET 7,C
CBDA	SET 3,D	CBFA	SET 7,D
CBDB	SET 3,E	CBFB	SET 7,E
CBDC	SET 3,H	CBFC	SET 7,H
CBDD	SET 3,L	CBFD	SET 7,L

OP-Code	Mnemonic
CB26	SLA (HL)
DDCB*05*26	SLA (IX+d)
FDCB*05*26	SLA (IY+d)
CB27	SLA A
CB20	SLA B
CB21	SLA C
CB22	SLA D
CB23	SLA E
CB24	SLA H
CB25	SLA L
CB2E	SRA (HL)
DDCB*05*2E	SRA (IX+d)
FDCB*05*2E	SRA (IY+d)
CB2F	SRA A
CB28	SRA B
CB29	SRA C
CB2A	SRA D
CB2B	SRA E
CB2C	SRA H
CB2D	SRA L
CB3E	SRL (HL)
DDCB*05*3E	SRL (IX+d)
FDCB*05*3E	SRL (IY+d)
CB3F	SRL A
CB38	SRL B
CB39	SRL C
CB3A	SRL D
CB3B	SRL E
CB3C	SRL H
CB3D	SRL L
96	SUB (HL)
DD96*05*	SUB (IX+d)
FD96*05*	SUB (IY+d)
97	SUB A
90	SUB B
91	SUB C
92	SUB D

OP-Code	Mnemonic
93	SUB E
94	SUB H
95	SUB L
D6*20*	SUB n
AE	XOR (HL)
DDAE*05*	XOR (IX+d)
FDAE*05*	XOR (IY+d)
AF	XOR A
A8	XOR B
A9	XOR C
AA	XOR D
AB	XOR E
AC	XOR H
AD	XOR L
EE*20*	XOR n

A. 4 Specifications

1. MZ-80A GENERAL SPECIFICATIONS

CPU	SHARP LH0080 (Z80-CPU)	**Key layout**	73 keys ASCII standard main keyboard Numeric pad
Clock	2 MHz	**Editing function**	Cursor control; up, down, left, right, home, clear Deletion, insertion
Memory	ROM 4K bytes (monitor program) ROM 2K bytes (character generator) RAM 32K bytes (dynamic RAM) Can be expanded to 48K bytes. (option; 16K bytes)	**Clock function**	Built-in
Display	9" CRT (green display) Character display 8×8 dot matrix Characters; 1000 (40 characters x 25 lines) Pseudo-graphic display 80 x 50 dots	**Power supply**	Local supply rating voltage
		Temperature	Operating temp; 0° to 35°C Storage temp; −15° to 60°C
		Humidity	Lower than 80%
		Weight	Approx. 10 kg
Cassette	Standard audio cassette tape Data transfer speed; 1200 bits/sec. Data transfer system; SHARP PWM	**Dimensions**	Width; 440 mm Depth; 480 mm Height; 260 mm
Sound output	Max. 400 mW (440 Hz)		

2. CPU BOARD SECTION SPECIFICATIONS

CPU	SHARP LH0080 (Z80-CPU) 1 pc.		
ROM	Monitor ROM (4K bytes) 1 pc. Character generator ROM (2K bytes) 1 pc.	**Programmable peripheral interface**	8255 1 pc.
RAM	Standard; 16K bits dynamic RAM 16 pcs. Optional; 16K bits dynamic RAM 8 pcs. Video RAM (2K bytes) 1 pc.	**Programmable counter**	8253 1 pc.

3. POWER SUPPLY SECTION SPECIFICATIONS

INPUT	Use a power source with the voltage shown on rating name plate.	**OUTPUT**	5V, −5V, 12V (stabilizing), 12V (non-stabilizing)

4. DISPLAY SECTION SPECIFICATIONS

Size	9″
Vertical horizontal frequency	60Hz (vertical), 15.75kHz (horizontal)
Power source	DC 12V, 1.1A ±10%
Picture tube	E2728B3; 9″90° deflection explosion proof type Heater; 12V, 75mA
ICs	2 pcs.
Transistors	7 pcs.
Diodes	13 pcs.
Sound output	400mW max. (440 Hz) Speaker 8cm, round dynamic type (32Ω)
Control knobs	Volume, V-Hold, Contrast, H-Hold, Brightness, Focus
Working temperature	−10°C to 50°C
Video output	40Vp-p standard (35Vp-p limit)

Resolution	Horizontal  *The pattern of the left in the center of the picture must be clear.
Non-linearity distortion	Horizontal; ±8% (±14% max.) Vertical; ±8% (±12% max.)
Geometrical distortion	Pincushion dist.; 1% (2% max.) Barrel dist.; 1% (2% max.) Trapezoidal dist.; 1% (2% max.) Parallelogram dist.; 1° (2.5° max.)
High voltage	Zero beam; 11.0kV (10.0kV, min., 12.0kV, max.)
Power supply	DC 12.0V, 1.05A (1.2A max.)
Working range	12V ±10%
Scan size	Horizontal; 10% (15% max.) Vertical; 10% (15% max.)
Horizontal lock-in range	±300Hz (±100Hz limit)
Vertical lock-in range	−12Hz (−6Hz limit)
Audio frequency characteristic	440Hz (0dB) −10dB ±4dB at 100Hz −12dB ±4dB at 10kHz

5. CASSETTE TAPE DECK SECTION SPECIFICATIONS

System	PWM recording
Power source	5V ±0.25V (rated)
Rated amperage	Wait; 2mA Record; 70mA (TEAC test tape) Playback; 7mA (TEAC test tape)
Semiconductors	4 transistors 1 IC 4 diodes
Applied tape	From C30 to C60
Tape speed	4.75 cm/sec.
Track	2-track monaural type
Motor	Electronic governor motor (12V)
Biasing	DC system

Erasing	DC system
Playback sensitivity	1m sec. to 500μ sec. (standard)
Input level	Below 0.4V ("L") Over 2.0V ("H")
Input impedance	Over 10kΩ (record jack)
Output level	Below 0.4V ("L") Over 2.0V ("H")
Working temperature	−10°C to 50°C
Storage temperature	−25°C to 70°C

Specifications and design subject to change without prior notice for product improvement. In such cases, items mentioned may be partially different from the product.

A. 5 Caring for the system

- **Power cable**
  Don't place heavy objects such as desks or chairs on the power cable and do not damage the covering of the power cable or a severe accident may occur. Be sure to pull the plug (not the cable) when disconnecting the unit from the AC outlet.

- **Line voltage**
  The correct line voltage is shown on rating plate. Extremely high or low line voltages may cause trouble or result in incorrect operation. Contact your dealer if such trouble occurs.

- **Ventilation**
  Ventilation holes are provided in the cabinet. Never place the unit on a carpet or the like because the holes on the bottom plate of the cabinet will be covered. Place the set in a well ventilated location.

- **Moisture and dust**
  Place the unit in a location which is free from moisture and dust.

- **Temperature**
  Do not expose the unit to direct sunlight and do not place it near heaters to prevent its temperature from rising.

- **Water and other foreign substances**
  Operating the unit when it is wet or contains foreign articles such as clips, pins or other metallic items is dangerous. If water or other liquid enters the unit, immediately pull the power plug and contact your dealer.

- **Shock**
  If the unit is subjected to shock the sensitive electronic parts may be damaged.

- **Trouble**
  If any trouble occurs, stop operating the unit immediately and contact your dealer.

- **Long periods of disuse**
  When the unit is not operated for a long time, be sure to pull the power plug from the AC outlet.

- **Connection of peripheral devices**
  When connecting peripheral devices, use only parts and devices designated by the Sharp Corporation. Use of parts and devices other than those designated (or modification of the set) may result in trouble.

- **Stains**
  Remove stains from the cabinet with a soft cloth moistened with water or detergent. Never use solvents such as benzine, or discoloration will result.

- **Noise**
  When the unit is used in locations where there are high electrical noise levels induced in the AC line, use a line filter to remove the noise. Keep the signal cables away from power cables and other electric equipment.

- **Use and storage**
  Do not use or store the unit with the upper cabinet open, or trouble may occur.

- **Radio wave interference**
  When a radio or TV set is used near the MZ-80A, noise may interfere with broadcast reception. Equipment causing a strong magnetic field may interfere with operation of the MZ-80A.
  Keep such equipment at least 2 to 3 meters away from the MZ-80A.

- **Power switch operation**
  Once the power switch is turned off, wait at least 10 seconds before turning it on again. This ensures correct operation of the microprocessor. Never insert the power plug into an AC outlet with the power switch set to ON.

- **Cassette deck maintenance**
  Dirty cassette deck recording and reproducing heads may result in incorrect data recording or reproduction. Be sure to clean the heads every month. Commercially available cleaning tape is convenient.

- **Discoloration of CRT screen**
  If a certain spot of the CRT screen is lit an external period of time the spot may become discolored. (If it is necessary for certain spot to be lit for an extended time, turn down the brightness control on the display control unit.)

SHARP CORPORATION

MODEL: MZ8AM01E

TINSE0038PAZZ | MZ-80A
080311-250182 | E2