PC 8300 / SINCLAIR 1000

YOUR COMPUTER

INSTRUCTION MANUAL

TABLE OF CONTENT

Chapter 1

SETTING UP THE COMPUTER

Unpack package box of the computer that you just purchased from an authorized dealer, are the computer main unit together with a demonstration software tape, a AC adoptor, a TV cable, a pair of cassette cable, and this manual book. Fig. 1-1 shows a photo of the computer.

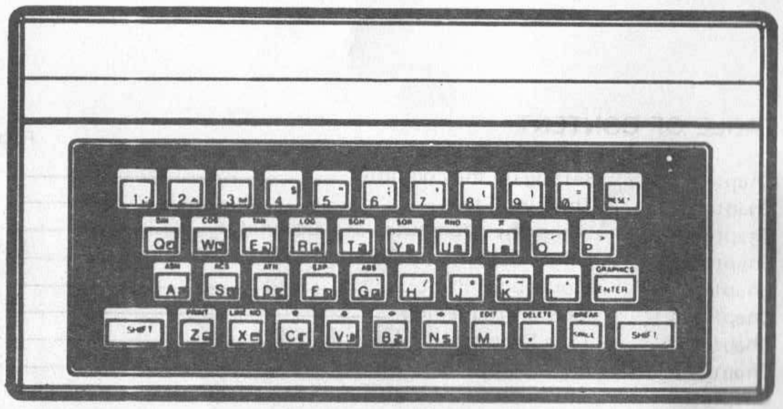


Fig. 1-1

HOOKING UP THE CRT DISPLAY

From the rear side of the computer (See Fig. 1-2) you can find a TV port and a monitor port. If a TV is being used as the computer's display, connect the TV port and the TV ANTENNA Coaxial input directly with the TV cable (NOTE: remove the external antenna from the TV port)

Due to the less distortion and more stabilize, it is recommended to use a monitor for display.

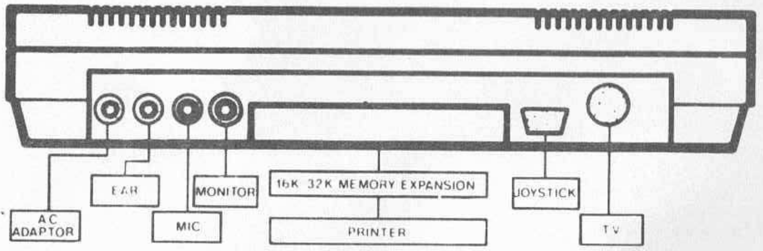


Fig. 1-2

NOTE: If your TV have a balancing antenna terminal (its resistance is 300 Ω) and not a coaxial antenna terminal (its resistance is 75 Ω), you must use a optional Resistance Converter to convert the resistance from 300 Ω to 75 Ω. Otherwise you cannot match the TV and the computer.

THE CASSETTE RECORDER

The Cassette recorder cable connects the computer to any ordinary cassette recorder via the two cassette interface ports labeled "EAR" and "MIC" on the rear side. The recorder is convenient to use for saving and loading programs on tape.
As you know, the cassette recorder can do two things TO SAVE (output and store) your programs from the computer with the SAVE command;
TO LOAD (playback and input) and the stored program from the tape into the computer.

SETTING UP A.C. ADOPTER

The last stage is to set up the AC adoptor, you can easily plug the adoptor into any AC power line, then plug the DC output of the adoptor into the computer DC input port (that you have seen from the rear side of the computer). Inserting the DC plug will simply turn ON the computer and the power lamp at the up right corner of the keyboard indicate whether the power is ON or OFF. You can also turn OFF the computer by unplugging the power cord from the computer jack. Be remember that any time you turn OFF of computer will erase the contents of the computer's memory. So be take care unless necessary, not to unplug the cord from the jack.
When you are not using the computer, it is best to unplug the adoptor from the AC power line. It is not a good idea to leave the live voltage around.

ADJUSTING THE SCREEN

TV Set

Now, turn down the TV volumn all the way, turn ON your TV, then, you can turn ON the computer by plug the power cord into the jack, and watch for the screen. If you have followed each step correctly, three things will happen when you turn ON the computer:
The power lamp should light, the computer's speaker should "beep!" and a blinking square called the "cursor" should appear on the bottom left corner of the screen.
If the power lamp do light and the specker do "Beep" but you don't see the blinding cursor on the screen, don't be worry. You just want to tune your TV. First, check whether your TV is set to channel 3 (or channel 2); if the channel are correct, tune the fine tuning knob of your TV set until you get a square cursor.
If after these steps, your screen still don't display correctly, you can press the ENTER key and hold it down while press the RESET key on your computer or you can simply turn OFF the computer then turn back ON again. That will probably do the trick.

Monitor

If you use a monitor instead of the TV set, the procedure will be easier and simplier. First, turn ON your monitor, then turn on your computer. Three things will happen when you turn ON the computer: The power indicator should light, the speaker should "beep" and a blinking cursor should appear on the monitor screen. No any adjustment are required in this step and the image appeared on the screen will be more clear and stable.

Chapter 2

THE FIRST STEP

INPUT & OUTPUT

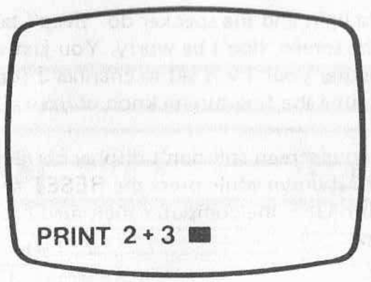
Turn the computer on (by pluging it in) just as describle in Chapter 1. The blinking cursor appears on the screen tell you that computer is ready to receive a command/instruction from you and do something you want it to do.
To try it, please use PRINT. PRINT is a Basic command that ask computer to display a number on the TV screen.

```
PRINT 2 + 3
```

tells the computer to work out the sum 2 + 3 and display the answer on the TV screen.
A message like this, telling the computer to do something straight away, is a COMMAND; this particular one is a PRINT command.
Notice that you didn't have to type in the whole word P-R-I-N-T letter by letter from keyboard. At the upper side of the Z key there is a keyboard "PRINT" appear and at the bottom of the keyboard, there are two SHIFT key located at the two end. To enter the PRINT command, you just want to press and holding down the SHIFT key while press the Z key. Try it. Now the word "PRINT" but not letter "Z" enter to the screen. There were some other keywords such as "SIN" "COS" "TAN" "LOG" "SGN" "SQR" "RND"..... appear on the keyboard.
After inter the PRINTER command, a word PRINT appear on the screen and the cursor moves to the right side of the word. Now type 2, again you should see 2 appear on the screen, and the cursor move along one space, then type + . To get "+" you should type letter L while pressing the shift key, some what like to type the PRINT command. Now type 3, you should get

```
PRINT 2 + 3 ■
```

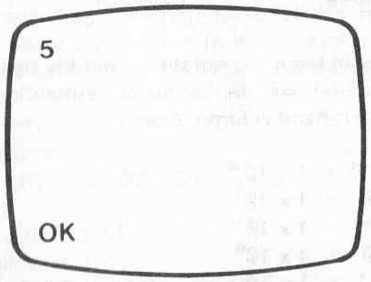
appear on the bottom of the screen (symbol ■ represents the blinking cursor)



```
PRINT 2+3 ■
```

It is important to realize that the bottom line of the screen shows only the current input to the computer. This line just echoes what you enter. The computer actually done nothing yet.

To tell the computer to read the message you just type in, you must press the ENTER key.
Pressing the ENTER key tells the computer the input line is complete and should be entered for processing. After the ENTER key is pressed, the output is printed in the upper left corner of the screen, as shown below.



```
5



OK
```

At the bottom left corner of the screen, some additional symbols appear. They make up a report message your computer displays after it processes the input. In this example the report message is OK which means the computer successfully complete the task you ask it to do. The report message will remain until a key is pressed.
After a command is executed or after a program execution is completed or interrupted, a report message will be displayed showing what has happened and where in the program it happened. The table in the Appendix B gives back report message with a general description and a list of the situations where it can occur.

Now that you've got the hang of it, try the following:

```
PRINT    1Ø
PRINT    1ØØØ
PRINT    1ØØØØØØ
PRINT    1ØØØØØØØØØØ
PRINT    1ØØØØØØØØØØØØ
```

Notice that each Zero is printed with a slash through it so that you don't confuse zero with the letter "O". It is a common practice in electronics and computer publications to print a zero as Ø.
Now try this example:

```
PRINT    1ØØØØØØØØØØØØØ
```

as you can see, something unexpected happen. At the upper side of the screen, you get

```
1E + 13
```

If you count the zeros after the 1, you'll find 13, which is also the number after E. In the field of computers, science and engineering, this way of expressing very large or very small numbers is common and is called Scientific or Power-of-ten notation.
The following examples are numbers expressed first in standard form, and then in scientific notation. Note that the sign "+" is optional for positive powers.

For example, 1E + 1Ø can be written as 1E1Ø.

Number	Power of Ten
1	1EØ
1Ø	1E1
1ØØ	1E2
1ØØØ	1E3
12345	1.2345E4

Try printing a number in scientific notation from the right-hand column above, and your computer will display the corresponding number in standard form in the left-hand column above.
In more general terms,

$$1E1\emptyset = 1 \times 1\emptyset^{1\emptyset}$$
$$1E3 = 1 \times 1\emptyset^{3}$$
$$1E1 = 1 \times 1\emptyset^{1}$$
$$1E\emptyset = 1 \times 1\emptyset^{\emptyset}$$
$$1E\text{-}1 = 1 \times 1\emptyset^{-1}$$

and so on. The number after the "E" is called the exponent, and the number before "E" called the mantissa.
Try

```
PRIN  1E – 1
```

the result is

```
Ø.1
```

Note that to get a minus sign, you must first press and hold down the SHIFT key, and press the "K" key while SHIFT key is pressing. likewise,

```
PRIN  1E – 2
```

will result

```
Ø.Ø1
```

the munus exponent means 1 divided by the number raised to the plus exponent.
Now try

```
PRINT  1/1ØØ
PRINT  1/1ØØØØ
```

will gives Ø.Ø1 & Ø.ØØØØ1 respectively. Note that the division sign is on the H-shift key.
Finally, enter

```
PRINT  1/1ØØØØØØ
```

which gives

```
1E – 6
```

And if you try even smaller numbers, such as 1/1ØØØØØØØ, you will continue get numbers in scientific notation.

NUMBER LIMITS

The largest number the computer can handle is about 1.7Ø1411E38, and the smallest number greater than Ø is 2.93873588E – 39.
For example, try enter

```
PRINT  1.7Ø1411E38
```

it will be printed,
now try

```
PRINT  1.7Ø1412E38
```

and instead of the correct output, you'll see a blinking E appears following the 9, and while you will hear a soulless sound from the speaker to warning that something unexpected happen. The blinking E is a syntax error message from the computer that means you are trying to do something that doesn't fit the rules of BASIC language syntax. Just like a regular language. BASIC has a vocabulary consisting of PRINT, LIST, LOAD, RUN, etc., and a syntax that tells how the words can be used. In this example, the computer can't accept a number greater than 1.7Ø1411E38 and therefore flags 1-7Ø1412E38 as a syntax error.

```
PRINT  1.7Ø1412E38
```

Now, try a small number like

```
PRINT  2.93873588E – 39
```

and you will see

```
2.9387359E – 39
```

you will get the same result for

```
PRINT  2.938E – 39
PRINT  2.1E – 39
PRINT  1.9E – 39
PRINT  1.47E – 39
```

but if you try to

```
PRINT  1.46E – 39
```

you will get a Ø.

Therefore, be careful with very small or very large numbers, because the computer can't represent them accurately, you may get erronreous result.

Chapter 3

USING THE COMPUTER AS A CALCULATOR

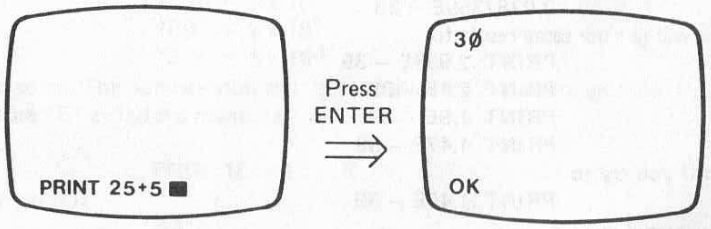
You have seen how easy it is to enter and print numbers. Now try using your computer as a arithematic calculator.

ADDITIONAL & SUBTRACTION

Suppose you want to add 25 + 5, you'll simply type

```
PRINT 25 + 5
```

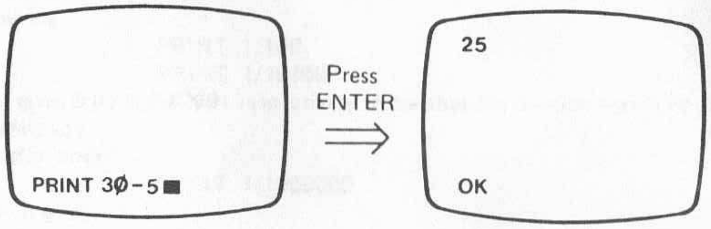
Note that the word PRINT is typed by shift-Z key and the Plus sign is typed by shift-L key. Then press ENTER key, the answer 3Ø appears on the upper side of the screen



Similarly, if you want to subtract 5 from 3Ø, type

```
PRINT 3Ø – 5
```

and then ENTER, the answer 25 will appears on the screen



The report message OK tells you that the addition/subtraction have been completed.

MULTIPLICATION & DIVISION

The computer uses an asterisk "*" for a multiplication sign, and uses the " / " symbol for a division sign. PRINT 4 * 8 will get 32, PRINT 32 / 8 will get 4. Try it, (NOTE the "*" is entered by press and hold down the SHIFT key while press J key and the " / " is entered by press and hold down the SHIFT key while press H key.)

WHO'S ON FIRST

In a simple calculation like

```
PRINT 4 + 8 / 2
```

you can't tell whether the answer should be 6 or 8, until you know in which order to carry out the arithmetic. If you add the 4 to 8 first then divided by 2, you get 6 however, if you divided the 8 by 2 first then plus to the 4, you will get another answer, that's 8.

Eight is the answer your computer will give. Here's how the computer choose the order in which to do arithmetic.

Try the following:

Expression	Answer
PRINT 3*8–5	19
PRINT 4*2+8	16
PRINT 9*5+2*6	57
PRINT 3/2–2*8+1.5	–13
PRINT 7+3*5–4/2	2Ø

The computer generally evaluates expressions from left to right. However, in the last example, the computer first multiplies 3 by 5, then stores the result internally as 15. Next, the computer divides 4 by 2 to get 2 and stores it internally. Finally, the computer adds 7 and 15 then subtracts 2 to yield 2Ø.

The BASIC in the computer is designed so that multiplication and division occur before addition and subtraction. Between multiplication and division, they are treated equally. For example:

```
PRINT 4*8/16
```

and

```
PRINT 8/16*4
```

yield the same answer.

Likewise addition and subtraction have no precedence between themselves. To illustrate, try the following;

```
PRINT 8+4–2
PRINT 4–2+8
```

both give 1Ø as the answer.

A SPECIAL CASE OF THE MINUS SIGN

As with most rules, there is at least one exception. What do you think

```
PRINT 3*–2
```

will give?

The computer will yield –6, because 3 times –2 is –6. In fact we don't want to subtract 2 from 3, and the computer wouldn't even accept this as subtraction because of the * after the 3. We want the minus sign in front of the 2 to be interpreted as the number –2.

When the minus sign is used to negate (make negative) the following, it's called a Unary Minus. For example:

```
PRINT –2
```

print a –2 on the screen, because we negate a 2. And

```
PRINT ––2
```

yields a 2 on the screen because we negated a minus 2. The term "Unary", which comes from Latin and means "one", is fitting because it applies to the one thing that follows it.

However, in the case of 4–2, the minus is called Binary Minus after the LATIN WORD "binary", which means "two". In this illustration, the binary minus relates two numbers, the 4 and 2. Therefore, a unary operation applies to one number, but a binary operation applies to two.
Try these examples:

```
PRINT -2*3
PRINT 8*-3+3
PRINT -1Ø/-2*-5-2
```

the first yield –6, the second yield –21 and the last yield –27. Notice how this last example is calculated. The computer divides –1Ø by –2 to get 5 first, multiplies this result by –5 to get –25, then subtracts 2 to get –27.

PARENTHESIS

If you are in doubt about how something will be evaluated, you can always use parenthesis to force the evaluation the way you want. Notice that the left parenthesis is on the shift –8 key and the right parenthesis on the shift –9 key. For example, try

```
PRINT -1Ø/(-2*-5-5)
```

It gives –1.25 because the expression in parenthesis is evaluated first.

PI (π)

Your computer has many math functions not found in a single calculation. These functions are convenient to have when you write computer programs or just use the computer as a super calculator for school work, or personal use.

PI (π) = 3.141592......

is one of these functions.
Notice that at the upper side of the I key, there is a symbol π. You can simply use SHIFT-I key (holding down the SHIFT key while press the I key) to get entering of PI, also you can type P-I by the P key and I key to enter PI. (Notice that PI appears instead of the symbol π. To try it, please type

```
PRINT PI
```

then press ENTER key, and the answer 3.1415927 will appear on the upper of the screen.
As an example, you can evaluate the circumference of a circle 10 cm in diameter by the formula π * DIAMETER. To show it, type

```
PRINT PI*1Ø
```

you get 31.415927 as an answer on the screen. Notice that PI is stored in the computer with greater accuracy that the 8 digits printed by the PRINT command. To show the extra digits, try

```
PRINT PI-3.141
```

the answer is Ø.ØØØ59265364. It show that PI is stored as about 3.14159265364.

Powers (**)

If you want to evaluate 1Ø*1Ø*1Ø = $1Ø^3$, you didn't have to enter

```
PRINT 1Ø*1Ø*1Ø
```

to get the result. You can use the power function of your computer. The

power function is labeled as "**" by two time press the shift-J key. (Hold down SHIFT key, then press J key two times)
Now try

```
PRINT 1Ø**3
```

you get 1ØØØ on the screen.
The power function is convenient because it's so general. For example, to evaluate the 4/5 power of 13, you type

```
PRINT 13**(4/5)
```

and get the answer 7.7831371. Notice that the parenthesis are necessary, or else you will evaluate the 13^4 then divided by the 5. Which is not the same. It is because the Power operation have higher precedence than multiplication and division.
You can also use decimal powers. For example:

```
PRINT 16**.5
```

will yield 4. (Because $16^{0.5} = 16^{½} = \sqrt{16} = 4$)

SQUARE ROOT - SQR

The square root it also gives as a math function, which appear on upper side of the Y key as SQR. You can hold down the shift key then press Y to get entering of SQR. Please try the following

```
PRINT SQR 16
```

and

```
PRINT SQR 625
```

the answers will be 4 and 25 respectively.

APPROXIMATIONS AND PRECISION

The calculators are done by the computer with powers and other math functions are generally correct. However, small errors may creep into the calculations because the computer calculates and stores numbers to only a certain degree of precision.
For example

```
PRINT 36**.5
```

will get 6. If we subtract 6. We should get Ø. Try it by typing

```
PRINT 36**.5-6
```

when you press ENTER, strangely, instead of Ø, you get –1.8626452E–9 on the screen, which is not zero.
Similarly, PRINT SQR 36 computer will answer as 6, but

```
PRINT SQR 36-6
```

then press ENTER, the computer will answer –1.8626452E–9.

Why? it is because the square root calculated by the computer is slightly less than 4 by 1.8626452E–9. In case of

```
PRINT 36**.5 or PRINT SQR 36
```

the PRINT command rounds off the result to 6. However, if the true value of 6 is subtracted from the square root, the difference shows up. You must keep in minds that the numbers are held only to a certain degree of precision in any computer.

INTEGER FUNCTION - INT

The integer function INT rounds down a number to the next smallest integer. To try it please type

```
                PRINT  INT  2.1
                PRINT  INT  1Ø.9
                PRINT  INT  (2.5 + 5)
```

you will get 2, 1Ø & 3 respectively. Notice that by adding Ø.5 to 2.5 you have rounded the number 2.5 up to 3. In general, you can round up by adding Ø.5 to any number.

The important application of the INT function is with numbers that have decimal parts.

Try the following

```
                PRINT  54.2–54
```

you get the correct answer Ø.2 on the screen. Now try

```
                PRINT  54.1–54
```

you get .Ø99999994. This answer is in error. It is because, as indicated earlier, numbers in computer are held only to a certain degree of precision. This problem may appear when you use numbers with decimal parts, called floating Point Numbers. Some flaoting point numbers cause more trouble than others. Those that do are usually numbers with a decimal part of Ø.1. It's a particular problem if you want to write a program for financial applications. However, there is an easy solution. The way is just to round off the answer.

For example, to round off any number N to two decimal places, use

```
                IN (1ØØ*N+Ø.5)/1ØØ
```

Try the following and check the results.

```
                PRINT  INT (1ØØ*Ø.5+Ø.5)/1ØØ
                PRINT  INT (1ØØ*1.5493+Ø.5)/1ØØ
                PRINT  INT (1ØØ*26.8935+Ø.5)/1ØØ
```

If you want to round off to three decimal places, use 1ØØØ instead of 1ØØ, and so on.

Chapter 4

OTHER MATHEMATIC FUNCTIONS

Your computer has some other math functions that you can find at the keyboard, they are:

SIN	(Sine)
COS	(Cosine)
TAN	(Tangent)
LOG	(Natural Logarithm (to base e))
SGN	(Signum)
RND	(Random)
ASN	(Arcsine)
ACS	(Arccosine)
ATN	(Arctangent)
EXP	(e^x)
ABS	(Absolute magnitude)

You can get these function by held down the SHIFT key then press the correspondent key, respectively.

EXPONENTIAL - EXP

The exponential function give the powers relative to the base of natural eogarithms; e = 2.7182818.

Like PI, base e cannot be written exactly with a finite number of digits. In general, EXP N = e^N , where N is any number.

```
            PRINT EXP 2.5
```

give the answer $e^{2.5}$ = 12.182494

NATURAL LOGARITHM - LOG

The natural log is inverse function of the EXP. In general,

```
            X = LOG (EXP X)
```

And that is why LOG and EXP are called inverse functions. You can get back the number you started with, X, by using the inverse function.

```
            X = EXP (LOG X)
```

Try the following

```
            PRINT LOG 1
            PRINT LOG 2.7182818
```

Notice that in the second one you can't get back to 1 exactly. That is because the number 2.7182818 is not the exactly value of e. Now try

```
            PRINT LOG EXP 1
```

the answer is correct.

ABSOLUTE VALUE FUNCTION - ABS

The absolute value function, ABS, always returns the positive value of a number.

```
Try                                          Answer
                 PRINT ABS 5                    5
                 PRINT ABS –5                   5
                 PRINT ABS (2*5+1)             11
```

SIGN FUNCTION - SGN

The sign function, SGN, return a +1 if a number is positive, Ø if the number is Ø, and −1 if the number is negative.

Try

```
PRINT SGN 5
PRINT SGN Ø
PRINT SGN −5
PRINT (2*5+1)
```

RANDOM FUNCTION - RND

The random function' RND, gives a number greater or equal to Ø and less than 1, RND is not a true random number generator, but instead of taking a sequence of 65.536 numbers that are almost random, and is called a pseudo-random generator.

Try

```
PRINT RND
```

to see whether the results is in random.

TRIGONOMETRICAL FUNCTIONS

The SIN, COS, TAN, ASN, ACS, ATN functions in your computer are all to work in radians, not degrees.

Try

```
PRINT SIN 3
```

where the 3 is in radians but not degrees. If you want to use degrees instead of radians, you must use the format

```
N = M*PI/18Ø
```

where M is the degrees value. To evaluate SIN 3Ø, you must type

```
PRINT SIN (3Ø*PI/18Ø)
```

VARIABLES

'Some calculator',you will ask, 'can store a number away and remember it later. Can my computer do that?

Of course, in fact your computer can store away literally hundreds, using the LET statement. Suppose that pencil cost 62 cents a dozen, and you want to remember this, type

```
LET PENCILS = 62
```

Now the computer has reserved a place inside it's memory where you can store a number and then, it has given this place the name 'pencil' so you can refer to it later. This combination of storage space and name is called a variable, it has stored the number 62 in this space. In this way, we say that the computer has assigned the value 62 to the variable (It's name is pencil).

Now, if you want to know how much a dozen of pencil cost? Simply type

```
PRINT PENCILS
```

If you want to know how much two dozen of pencil cost, type

```
PRINT 2*PENCILS
```

Suppose you want to change the cost of pencil from 62 cents to 59 cents, type

```
LET PENCILS = 59
```

this does not reserve any extra storage space, but replaces the old value of 62 with 59, now you type

```
PRINT PENCILS
```

confident in the expectation of getting the most up-to-date price available.

Now type

```
PRINT BILL
```

you will get a UV report message appears on the screen. The message means that you have used a Undefined Variable, − the computer has no idea that how much the BILL costs, because you haven't told it, Now type

```
LET BILL = 18.5
```

& everything will be OK. Try it by type

```
PRINT BILL
```

again.

Type

```
PRINT PENCILS + BILL
```

you get the answer as 77.5 on the screen.

The CLEAR statement can release all the storage space that had been reserved for variables − then every variable is as though it had never been defined. Press and release the RESET key or turn the computer OFF & ON will also do the clear statement. But that's not a good idea to release the variable by RESET or OFF − ON the computer.

A variable can have almost any name that you like, so long as it starts with a letter. For example, these are allowed to be the names of variable.

```
TWO LITRE OF MILK
RADIO 54
X39
K9 BD
```

But the following are illegal

2 BEAR 3	(begins with a digit)
TABLET?	(? is not a letter or a digit)
FORTH − AB	(− is not a letter of a digit)

inverse video characters are not allowed for variable.

Chapter 5

MORE ABOUT PRINT

Beside printing numbers, your computer can also print individual characters or strings of characters. For example, type

```
        PRINT "JOHN"
```

In fact, you can print anything between the quotation marks, even numerals. Try this

```
        PRINT "ROOM 4Ø7, COMPUTER CENTRE"
```

For a space, use the SPACE key at the bottom right of the keyboard. All the characters between quotation marks are a string of characters, which is commonly called a string. Characters can be any printable symbol, even , . *, +, etc.

Another interesting feature is adding characters. What do you get when you add APPLES and ORANGE ? To try this type

```
      PRINT "APPLES" + "ORANGE"
```

The result is that APPLESORANGES appears on the screen, since the + operation combines strings in the order they are given. This combination of strings is calld concatenation and is useful in some programming. Only + is allowed in such operation.

If you are printing numbers out, you can use these strings to explain what the numbers mean. For example, type

```
        LET PENCIL = 62
```

& then

```
        PRINT 'THE PRICE OF PENCILS IS ? ; PENCILS :" CENTS
                PER DOZEN."
```

Now, you will get

```
        THE PRICE OF PENCILS IS 62 CENTS PER DOZEN.
```

appears on the screen. In this example, behind "IS" and before "CENTS" there is a space typed in to separate "62" from "IS" and "CENTS". It is because, inside the PRINT statement, a semicolon tells the computer to print characters/numbers one after one.

For example,

```
        LET N = 32.5
```

and

```
        PRINT "THE ANSWER IS" ; N; "DOLLARS"
```

will get

```
        THE ANSWER IS 32.5 DOLLARS
```

PRINT GRAPHICS

The keyboard of your computer is a real marvel of efficient design. Besides the standard alphabetic characters and numerals you can also print graphic symbols and inverse video characters, the graphic symbols or characters are the small squares on some keys. On the key 1, 2 and 3, there are three special animation symbols. These graphic symbols, together with the music feature of your computer, permit you to perform some joyful program as your games or other application.

To print the graphics symbols on a key, first, press and hold down the SHIFT key, second, press the ENTER key, you will see the cursor change to a blinking (inverse G), to indicate that the computer is in graphics mode. While you still holding down the SHIFT key, press the 1 key – or any key with animation symbol or small square graphics symbol in the lower right corner. That symbol will appears on the screen. Notice that you have to hold down the SHIFT key because the graphics symbol is a shift key.

To get an inverse video symbol, follow first and second steps above, then release the SHIFT key. Now, press any key, and the inverse video version will appear on the screen. For example, press the H key and you will see a black H on a white ground.

PRINTING WITH AT

You can control printing position with the BASIC command AT. Try

```
        PRINT AT Ø.Ø; "E"        (print at top left)
        PRINT AT Ø, 21; "E"      (print at top right)
        PRINT AT 21, Ø; "E"      (print at bottom left)
        PRINT AT 21, 31; "E"     (print at bottom right)
```

For use with the PRINT command, the screen is divided into 22 rows and 32 columns. The rows are numbered from Ø to 21, from the top to bottom; and the columns from Ø to 31 from left to right. As you can see from the above examples, the first number after AT is the row number, and the second one is the column number. That statement form can be as following

```
    PRINT at row, column; item
```

there are item can be either a number or a string. For example, type

```
        PRINT AT 5,5; " THE COMPUTER "
```

If you input a row and column number over the range limited, you will get a message on the screen. To try this, you type

```
        PRINT AT 23.5; "HELLO"
```

you will get a IR message appears on the screen. IR means "Integer out of range". The reason is that 23 in the PRINT AT statement is out of the allowed range. (Ø to 21.)

PRINTING WITH TAB

Just as you can control printing with the TAB key on a typewriter, you can use the TAB command in BASIC to control printing with the computer. Also, like AT command, the numbers after TAB are allowed from Ø to 31. Type the following

```
        PRINT TAB 1Ø, "HELLO"
```

you get the "HELLO" appears near the middle of the top line of the screen. If you type

```
        PRINT TAB 31, "HELLO"
```

you will see that the H is pointed in column 31, but ELLO appears on the next line.

Chapter 6

BASIC PROGRAMING

The real difference between a computer and a calculator is that the computer can store a series of instruction in its memory, and, when you enter some information, the computer execute these instructions step by step, unnecessary to enter them manually each time.

When you type

```
PRINT 3+5*2
```

then ENTER, the computer answer with 13 on its screen, immediately. This circumstance is called immediate execution.

Now type

```
10 PRINT 3+5*2
```

the 1Ø and 2Ø is refered as line number. You can use any line numbers as long as they are integers from 1 to 9999, and increase in the ascending order. Now you type ENTER, nothing happen but the line you just type enter to the top line of the screen. (see Fig. 6.1)

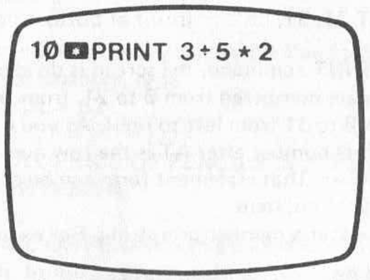


Fig. 6.1

To ask the computer to execute the statement, you must type RUN then press ENTER, and the answer 13 appears on the top line of the screen. (see Fig. 6.2). The remark on the bottom line "OK IN 1Ø" means line 1Ø's statement has been finished. You can RUN the same statement as many times as you like. A group of these statements is called a program. Differ from immediate execution, this circumstance is called deferred execution.

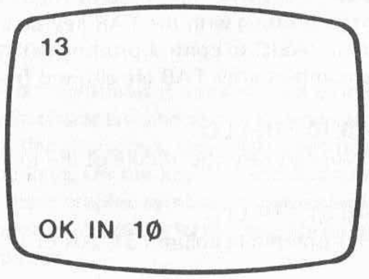


Fig. 6.2.

USING THE LINE NO

One of the unique features of the computer is the automatic line number function. The key "LINE NO." is located above the X key. When this key is pressed (By holding down the SHIFT key then press X key), will clear the bottom part of the screen and enter automatically a line number which is equal to ten plus the line number of the current line. It will be limited to 9999. Of course you can enter a line number manually by the numerical keys above the keyboard. If you are a programer or you are going to develop your own programs, you will be found that this feature of automatic line number is very helpful.

For our further discussion, please type

```
NEW
```

then ENTER, you will erase all the statement/data in the memory you have entered to the computer before.

Now type

```
LIST
```

the LIST command is usually used to list out your program on the screen. But now you can see nothing because all the programs inside have been erased by the NEW command (of course it must follow by a ENTER).

Now, enter your own program by using the automatic line number key.

(1) Press LINE NO. a 1Ø appears on the bottom part of the screen. (without ENTER).

(2) Type PRINT AT 1Ø, 4; " MICRO COMPUTER" and press ENTER, a complete line.

```
1Ø* PRINT AT 1Ø;4; " MICROCOMPUTER"
```

appears above the screen. The symbol * indicates that this line is currently entered.

(3) Type LINE NO. (without ENTER) again, you will automatically get a new line number 2Ø appears on the bottom part of the screen.

(4) Type PRINT AT 12,6; "YOUR GOOD COMPANION" follow to the 2Ø, then ENTER, you will get

```
1Ø PRINT at 1Ø,4; "MICROCOMPUTER"
2Ø * PRINT AT 12,6; "YOUR GOOD COMPANION"
```

on the screen. As you can see, the symbol * moves to the current line 2Ø. You can use the LINE NO. to get lines 3Ø, 4Ø ....and so on

```
3Ø PRINT AT 18,14; "PROGRAM BY JACK"
4Ø PRINT AT 19,14; "ON 3/2/1983, N.Y."
5Ø STOP
```

Now you get the completed program as below:

```
1Ø PRINT AT 1Ø,4; " MICROCOMPUTER"
2Ø PRINT AT 12,6; "YOUR GOOD COMPANION"
3Ø PRINT AT 18,14; "PROGRAM BY JACK"
4Ø PRINT AT 19,14; "ON 3/2/1983, N.Y."
5Ø * STOP.
```

Try to RUN it yourself.

EDITING PROGRAMS

You see that a symbol ■ by the line 5Ø? This is called the Program Cursor, and the line it points is the current line. You can move this cursor up to line 4Ø by press the key $\boxed{\uparrow}$ and down to line 5Ø again by press the key $\boxed{\downarrow}$.

Now suppose you want to change some thing in line 3Ø. Use the key to move the program cursor to line 3Ø, then press EDIT key (shifted M), and a carry of line 3Ø will be displayed at the bottom part of the screen.

```
        3Ø ■ PRINT AT 18, 14; "PROGRAM BY JACK"
```

Now press the key $\boxed{\Rightarrow}$ until the blinking cursor moves to the right of JACK.

```
        3Ø PRINT AT 18, 14; "PROGRAM BY JACK C."
```

Now press ENTER, then the New line 3Ø will replace the old one in your program. (You can use key $\boxed{\Leftarrow}$ to move the cursor to left position such as you use the $\boxed{\Rightarrow}$ key.)

Now, RUN the program to see what have been changed.

You can also change a line by re-type it with correction. For example, type in

```
        2Ø PRINT AT 12,7; "YOUR BEST COMPANION."
```

and press the ENTER key. This new line will replace the old line 2Ø. Try it.

To delete any line in your program, just type this line number and press ENTER. For example, type 5Ø then ENTER, the last line in your program will disappear.

LISTING PROGRAMS

Anytime the computer is not executing, you can check your program by using the LIST command. Type LIST and press ENTER, and you will see the program stored in your computer. (But notice that the program cursor disappears from the lines).

Now type

```
        LIST 3Ø
```

Notice that only line 3Ø, 4Ø & 5Ø appear on the screen, and the program cursor now are at line 3Ø.

The LIST command starts listing your program from whatever line is given to the end of the program, if possible. If you use the LIST command without a line number, the listing will show in sequence the line numbers greater than Ø. That is why when you LIST the program, the program cursor do disappear, – it goes to the "lineØ".

Note that the screen can only show 22 lines at a time. If your program is longer than 22 lines, you will have to list it in stages.

Chapter 7

TAPE STORAGE – SAVE & LOAD

When you turn off your computer, all the program & data inside the computer will be lost. The only way to save is to record them onto a cassette tape; you can load them back into the computer again at any time you need.

SAVING PROGRAMS

One of the best features of the microcomputer is its ability to save and load programs. Using an ordinary cassette recorder, you can save on cassette tape any programs you have written.

To try it, use the program you have completed in Chapter 6. Now let's save this program on tape:

(1) Connect the MIC jacks on the recorder and computer. It is better to connect only MIC jack and do not connect EAR jack because some tape recorders distort the computer's signals if the computer's MIC and EAR leads are connected to the recorder at the same time.

(2) Load a blank cassette tape to your recorder, it is advisable to buy high quality tapes for your computer programs, since any imperfections may prevent your program from being reloaded.

(3) Type in SAVE "(program name)". You can use any name of any length. For example, type in (without ENTER)

```
        SAVE "HELLO"
```

Notice that your program name must be within the quotation marks.

(4) Put your cassette recorder on RECORD and wait for a few seconds to allow the tape to staleilize its speed.

(5) Now, you can press the ENTER to start the saving process. After the ENTER has been pressed the screen will go blank for about 5 seconds with some thin white lines dancing around the bottom of the screen. This pattern is followed by horizontal black and white stripes jumping around, indicating that output is going to the tape recorder. (see Fig. 7.1)

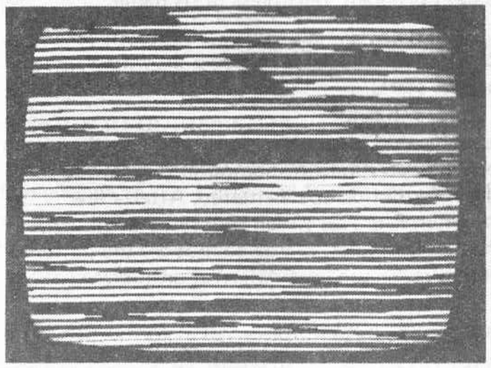


Fig. 7.1
Photo: Saving a program.

When the recording is over, the screen will go blank, and a OK report will appear on the screen.

Now, you can stop your recorder, rewind the tape to the beginning, your program in the computer has not been affected in any way by SAVING it. To verify this, type LIST you can check your program on the screen.

You should verify that the program was recorded by listening to the tape. If a lead was not connected, or if you have a bad tape, then you will not get a record. The pattern on the screen indicates only that information is being sent to the recorder. It does not indicate that the information was recorded. If the tape sounds normal, and there not a lot of background noise, you probably made a good copy.

LOADING FROM TAPE

For testing you can reload the program from the tape to your computer.

(1) Connect the EAR jacks on the recorder and your computer.

(2) Rewind the tape before loading the program into the computer. For a prerecorder tape with only one program, such as the one you just saved, just rewind the tape all the way.

(3) Adjust the volume on the tape recorder to about 3/4 of maximum. If your recorder has base and treble controls, set the treble high and the base low, because computer information is high pitched.

(4) Turn ON your computer and type a NEW (following ENTER) to ensure no any program inside the computer's memory. Then type (without ENTER).

```
        LOAD "(program name)"
```

There you can type the program name as

```
        "HELLO"
```

Notice that your program name must be within quotations mark. You can also load by entering (without ENTER)

```
        LOAD "  "
```

This command loads the next program, regardless of its name.

(5) Press the play key on recorder then press ENTER key on your computer, the loading process starts.

During loading, you will see many thin black lines slanting across the screen as shown in fig. 7.2. This pattern indicates that the computer is receiving the commands and is waiting for the information to be entered. After a few seconds, a different pattern (shown in Fig. 7.3) which looks like thick black bars dancing up and down with the black lines. If you cannot get the pattern, your recorder volume may be too high or too low. Adjust it until black stripes appear.

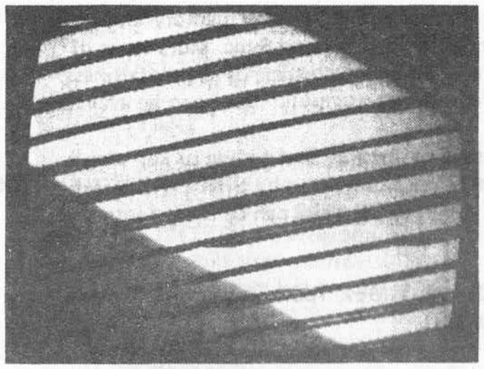


Fig. 7.2
Read for loading

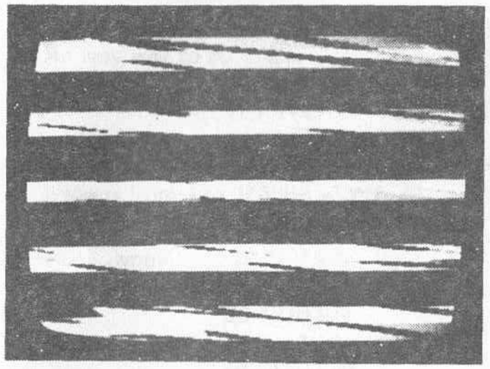


Fig. 7.3
Program is loading

After about 15 seconds, the bars will be disappeared and you will see a OK message in the bottom left corner of the screen. The "beep" and the message indicate the program has successfully been loaded. (If you cannot get the OK indication your recorder volume may be in incorrect position yet. Adjust it and try again.)

To ensure that the program is already loaded into your computer, type LIST (follow by ENTER) and you will see your program on the screen again. RUN it and see what will happen: The result is just same as it saved before!

MEMORY REQUIREMENTS

Your computer has a standard 2K bytes of memory built-in. It is more than enough to run a program as short as your HELLO program. But some programs need more than 2K bytes of memory to run it. If you run a program more than 2K bytes but less than 16K bytes, you must have to buy a 16K RAM (random access memory) Module.

It is important to point out that your computer can run all the pre-recorded tape for SINCLAIR/TIMEX 1000 microcomputer written by BASIC language. Most of these tape are required 16K bytes of memory and are commonly advertised in magazines like Syntax and Sync. More tapes of program written by both BASIC & assemble language of games, business, educational and personal program for the computer will soon be available.

However, due to different usage of memory and coding of key words, programs saved by the computer cannot be read by SINCLAIR/TIMEX 1000, even programs saved by SINCLAIR/TIMEX 1000 can be loaded into the computer and run properly.

(Note that when loads SINCLAIR/TIMEX 1000 program the variables of its program cannot be preserved.)

Chapter 8

MORE BASIC PROGRAMING

A NEW START

List the program on the screen. Now type in the key word

```
NEW
```

then, press ENTER, the programs on the screen are erased by this command immediately, while you will hear a "beep!" sound and the blinking cursor appears on the bottom left corner of the screen. The NEW command also clears the display file and the variable storage area. It resets the computer to the place where you started originally – with nothing in memory. It is better to use the NEW rather than use the RESET key when you want to start over.

IF & GO STATEMENTS

Suppose you want to print out integers from 1 to 1Ø, one number to a line. An easy way to do this is

```
1ØPRINT 1
2Ø PRINT 2
3Ø PRINT 3
```

and so on. But this will require 1Ø statement, and if you want to print the integers from 1 to 2ØØ this way, it will require 2ØØ statement. But this way obviously is not good presentation. A better way to do is what we called loop. For Example:

```
1Ø N = 1
2Ø PRINT N
3Ø N = N+1
4Ø GOTO 2Ø
```

The GOTO statement in line 4Ø perform a loop between line 2Ø and line 4Ø. Try it. The program will print forward until you get a "SF IN 2Ø" on the screen. (The SF means the screen full.)

There is a way to control how long a loop runs. It is to use the IF statement as a condition, if this condition is met, the computer will execute the GOTO statement otherwise skip it and go to the next line. For example, suppose you want to print integers from 1 to 2Ø, change the line 4Ø by:

```
4Ø IF N < 2Ø THEN GOTO 2Ø.
```

In general, the IF statement works like this

```
IF  arithmetic expression THEN any statement.
```

First, the arithmetic expression is evaluated. If it evaluates to zero (false) all the rest of that program line is ignored, and the computer goes on to the next line.

The IF statement is a very powerful one, and it will appear in almost every program you write.

REM, INPUT & CONT.

Type NEW then carefully type in this program

```
        10 REM THIS PROGRAM EXTRACTS SQUARE ROOTS
        20 INPUT A
        30 PRINT A, SQR A
        40 GOTO 20
```

Now run it. Apparently the screen goes blank, & nothing happens but the blinking cursor in the bottom left corner: the machine has gone into input mode. This is the effect of the INPUT statement in line 1Ø. The machine now is waiting for you to type in a number (or even an expression).

Just a minute though, what happened to line 1Ø? REM in line 1Ø stands for remark, or reminder, and is there solely to remind you of what the program does. A REM statement consists of REM followed by anything you like, and the computer will ignore it.

Now type in some number, 4. For example, then ENTER. the 4 and its square root appear on the screen, and again the computer do want another number. This is because of line 4Ø, GOTO 2Ø. The computer jumps back to line 2Ø & starts again.

To stop it, Type

```
        STOP
```

then ENTER, the computer reports back with report ST IN 2Ø. ST stands for STOP.. After ST, if you want more square root number, then type:

```
        CONT
```

(stands for CONTINUE) and the computer will clear the screen and ask for another number.

Note that the screen cleared is not because CONT statement, but because it is a command. All commands clear the screen first.

SCROLLING

When you write a program to input data, the program will stop when the screen is full. For example, the program shown above:

```
        20 PRINT SIN A
        30 GOTO 10
```

Keep entering numbers until the screen is full and you get the report SF IN 3Ø appears on the screen. If you want to continue inputing, you must type CONT. But, your computer has a command that can prevent a full screen from occuring, that is SCROLL. Add

```
        15 SCROLL
```

to your program. Now, when you run this program, the output will appear on the bottom line of the screen and scroll upward.

SCROLL moves the top line off the screen and sets the printing position to the bottom line 21. If you enter SCROLL as a command, it will only clear the screen.

ANOTHER INPUT LOOP

Type in these line

```
        100 REM THIS PROGRAM MEASURES STRINGS
        110 INPUT A$
        120 PRINT A$, LEN A$
        130 GOTO 110
```

This is a separate program from the last one, but you can keep them both in at the same time. To run the new one, type

```
        RUN 100
```

This program inputs a string instead of number, and prints out it and its length. Because the computer is expecting a string, it prints out two string quotes.

Now look back at the RUN 1ØØ which just jumps to line 1ØØ, that is some similiar to GOTO command. In fact, you can use

```
        GOTO 100
```

to run this program.

What is the difference between RUN and GOTO? One big difference between RUN and GOTO is that GOTO does not clear the value assigned to a variable. For example, if you run the program below

```
        10 LET A = 5
        20 PRINT A.
```

Now enter a LET command

```
        LET A = 10
```

then type

```
        GOTO 20
```

and an answer 1Ø print for A. But if you tried:

```
        RUN 20
```

an error report will be appeared, because RUN clears the variable memory area.

Remember that a CLEAR command also clears the variable memory area. To try it, type

```
        CLEAR
```

and then type

```
        PRINT A
```

A error report UV will appear, (UV is short for Undefined Variable).

READING THE KEYBOARD

The input command is useful for calculations or other data processing in which you want the computer to stop, wait for input, then continue execution. However, in designing games, you may want the computer to response for input but without stopping the program. A familiar example is the joystick of a computer game. The game machine continues to response as you move the joystick, instead of halting the action to ask for input.

Your computer has a special statement called INKEY$, which can be used to tell automatically what key is pressed. If a key is pressed, the INKEY$

function returns either its character or the null string " " – means no character. Now run the program below:

```
10 LET K$ = INKEY$
20 PRINT K$
30 GOTO 10
```

An ever shorter version is

```
10 PRINT INKEY$
20 GOTO 10
```

and a shorter version still is

```
10 PRINT INKEY$
20 RUN
```

Notice how quickly INKEY$ scans the keyboard.

DRAWING PICTURES BY JOYSTICK

You can easily draw pictures on the screen by using the commands we have discussed so far. Try this program to draw a vertical line of asterisk.

```
10 LET X = 11
20 LET Y = 16
30.LET K$ = INKEY $
40 IF K$ = "F" THEN LET X = X+1
50 IF K$ = "4" THEN LET X = X–1
60 IF K$ = "U" THEN LET Y = Y+1
70 IF K$ = "7" THEN LET Y = Y–1
80 PRINT AT X, Y; "*"
90 GOTO 30
```

You can move the asterisk all over the screen and draw all kinds of patterns by your joystick. Of course, you can substitute any character for the asterisk.

RUN it and you will find that you can move the asterisk all over the screen and draw all kinds of patterns.

FAST MODE AND SLOW MODE

Your computer can be speed up by 4 times if input the FAST mode. Type this program

```
10 LET X = INT (21 * RND +0.5)
20 LET Y = INT (31 * RND + 0.5)
30 PRINT AT X, Y; "*"
4 GOTO 10
```

Run it in normal mode. Now type FAST (then ENTER) then run the preceeding program. Notice that the screen goes blank while the program is running. This happens because your computer is spending all of its time on calculation instead of on refreshing the TV screen.

The term refreshing refers to the rewriting of the picture on the screen, which must be done at 60 frames a second. In your computer a device called a Z80 microprocessor controls all the computer's activities. In normal SLOW mode, the microprocessor must continually interrupt its calculation on refresh the screen. In FAST mode, the microprocessor concentrates on the calculations alone.

To return to normal SLOW mode, type SLOW then ENTER. SLOW and FAST can also be included as statement in your program. When you need the speed and do not care about the display, use FAST. When you need to see the display, switch to SLOW.

THE PAUSE

The PAUSE command stops calculation and shows the display. PAUSE halts execution in units of one TV frame. Two frames make up a complete picture on your screen. The image for a standard U.S. TV-transmission is formed at the rate of 1/60 frames per second; therefore, the smallest unit for a PAUSE is 1/60 of a second. The maximum number of pause allowed is 65,535. Pause from 32.767 to 65.535 are not timed. Instead, you can press a key to terminate the PAUSE. In fact, you can always terminate a PAUSE by pressing any key.

Let's try timing a PAUSE. Since one frame = 1/60 second, 20 seconds should equal to 1,200 frames. ENTER

```
PAUSE 1200
```

Note that everything disappears from the screen. Keep your PAUSE time from when you press the ENTER key until a OK report appears.

The problem is that if using a PAUSE in your program that other statements will be collected the time. For example, ENTER

```
10 LET N = 0
20 PAUSE 60
30 PRINT AT 0,0; N
40 LET N = N+1
50 GOTO 20
```

This program acts as a digital timer. About every second, it prints the elapsed time in seconds since the program started. Although a PAUSE lasts one second, the computer takes some time to execute the other statements. To compensate for the difference, you have to adjust the PAUSE 60 downward. Unfortunately, the smallest increment you can adjust is 1/60 seconds, this is not accurate enough. Another problem is the annoying screen flicker that may occurs when it reactivates the screen.

Chapter 9

THE COMPUTER AND MUSIC

One of it's excellent features of your computer is it's programable sound feature.

Up to now, you have heard so many different sound from your computer When you turn it on, it sounds a "beep!"; each time you press a key, you get a sound feedback of difference frequency from difference key. At the time SAVE/LOAD a program to/from a tape, you heard a "beep!" sound to indicate the finish of the process. When you type in a error command/ statement, it gives a buzzer sound. But the most important thing is that you can control the computer's sound output by your program, even you can ask your computer to perform any melody!

BEEP MODE & NO BEEP MODE

"So many sound!" you may think. "May I play my computer more quietly?" The answer is sure. You can simply type

```
NOBEEP
```

then ENTER. Now the computer is in No BEEP mode, and no any sound will be responded when you press the keys. Later, if you like to have beep sound again, you type

```
BEEP
```

then ENTER. Now the computer already in BEEP mode, and the keys will respond it's lively sound if you activated any key.

Turn ON the computer or a RESET cycle will automatically set the computer into BEEP mode.

USING MUSIC STATEMENT

Most of the modern computer magazine are recently talking about some interesting title: Computer & Music. Computer & poem, Computer & Literature, etc. The excellent design of your computer will bring you to enter one of these interesting area : Computer & Music.

The MUSIC statement in your computer's BASIC will help you to perform some music tones, even a melody, via the speaker inside the computer. To try it, you enter MUSIC followed by some strings specified. For example

```
MUSIC "C1ØD1ØE1ØF1ØG1Ø"
```

perform a "1-2-3-4-5-" (Pronounce as do-re-mi-fa-so) of 1Ø units of time. The string (remember that a string must be inside the quotation) represents a series of notes, each can be 2 to 4 characters long. The first one or two characters represent the note and the octave, sharps are represented by the inverse video characters. Following the note are one or two digits specifying how many units of time the note is to last, and can be from 1 to 99 (a Ø is equivalent to 256). All the possible notes and their frequencies are listed in the Appendix A.

A rest (pause) is just like an ordinary note but with a "." replacing the note

characters. Spaces are ignored in interpreting the string and empty string is allowed.

If you have entered an incorrect string format for MUSIC statement, you will get a MF error report on the screen.

TEMPO IN YOUR MUSIC

The TEMPO statement is used to control the note duration unit used in the MUSIC statement. For example, if you type in

```
TEMPO 2Ø
```

you have got the note duration unit equal to 2Ø x 3.94 ms. Normally the TEMPO format is as

```
TEMPO n, Ø ≤ n ≤ 255
```

and n=Ø is equivalent to n=256. If you have typed a number outside this limit you will get an IR error.

For example type and run this program:

```
NEW
1Ø MUSIC "C1ØD1ØE1ØF1ØG1Ø"
2Ø RUN
```

Your short melody will going forward without stop. It is because the RUN statement in line 2Ø cause a loop between line 1Ø & line 2Ø, and it is an undefinite loop. To stop this program you must use the BREAK command, it is a shift key up-on the space key on the keyboard. (You can also use the RESET key to stop the program, but it's not a good idea to break a program by RESET, because in some circumstance, RESET can clear some data in your program).

Now, BREAK it then type (followed by an ENTER)

```
TEMPO 1Ø
```

and run the short program again to see what happens. As you can hear, the music goes faster. You can BREAK it again and type TEMPO 1ØØ, or TEMPO 4Ø, TEMPO 5 to see how the tempo statement effect. Try it yourself.

You can also use the TEMPO statement inside your program to perform a more animating music.

PROGRAMING YOUR OWN MELODY

Use the character – Note table listed in Appendix A, you can easily program out any melody you like and use the TEMPO statement to perform it lively. For example, type

```
NEW
1Ø REM ...OLD SUSANNA
2Ø LET A$= "C6D2E3G3G6A2G3E3C6D2E3E3D3C3D12"
3Ø LET N = 1
35 TEMPO 3Ø
4Ø MUSIC A$
45 LET N=N+1
5Ø IF N<3 THEN GOTO 4Ø
```

```
60 IF N>3 AND N<6 THEN GOSUB 200
70 IF N=6 THEN GOSUB 250
80 IF N<6 THEN GOTO 40
100 STOP
200 TEMPO 10
210 MUSIC A$
220 RETURN
250 FOR I=1 TO 3
260 TEMPO 25/I
270 MUSIC A$
280 NEXT I
290 RETURN
```

Now, wait a minute, what is the GOSUB? It's the short for GOTO subroutine. A subroutine is a rather short program that do some special jobs and can be called by a main program where or whenever you need. In the example shown above, from line 10 to line 100 is a main program, line 200 & line 250 are two subroutines. Note that a subroutine must be followed by a RETURN. A GOSUB n statement is some what like GOTO n statement except that the computer remembers the line number of the GOSUB statement so that it can come back again after doing the subroutine.

SOUND STATEMENT

The SOUND statement can use for special sound output for your games and even in your music, you can use the SOUND statement to perform a special effection. The normal format of SOUND statement is

SOUND m , n

where m, n is a integer

$0 \leq m \leq 255$

$0 \leq n \leq 65535$

(m=0 is equivalent to m=256; and n=0 is equivalent to n=65536) When m,n is specified, the sound statement produces a sound of frequency equal to 32500/m Hz and its duration equal to n/65 ms in the FAST mode. In computer and display mode (called SLOW mode), the duration is approximately four times longer and the sound is modulated by 50 Hz. For example, type this program and run it.

```
FAST
10 FOR M=100 TO 200, step 10
20 FOR N=100 TO 1000, step 10
30 SOUND M,N
40 NEXT N
50 NEXT M
60 STOP
```

Note that there are two loops inside this program; M loop and N loop, N loop occurs entirely within the M loop, that is what we called Nested Loops. The nested loops is very useful in your programing.

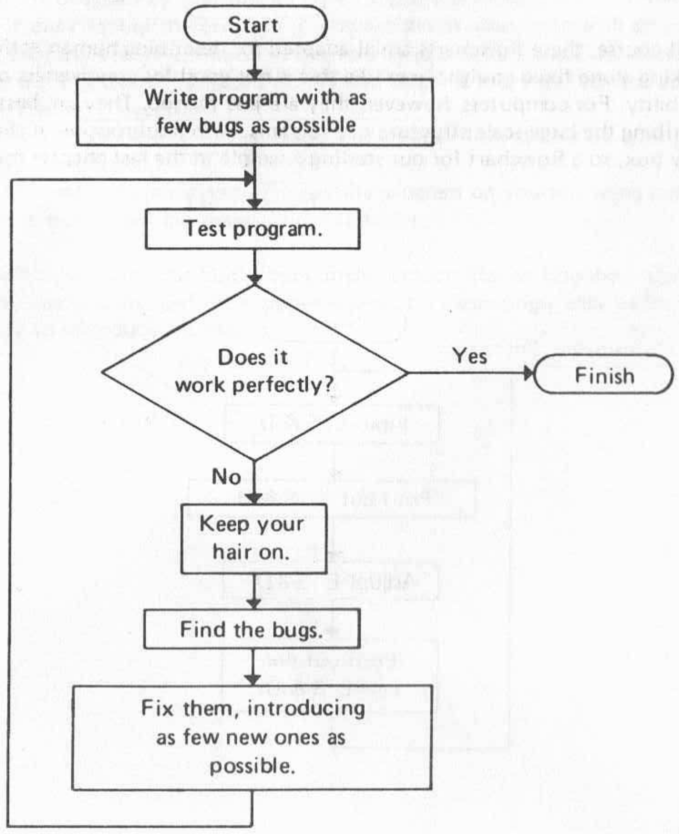
You must be careful if you are using the nested loops, the two loops, must be one inside the other, otherwise your program didn't run.

Chapter 10
MAKING YOUR PROGRAMS WORK

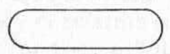
There is more to the art of programming computers than just knowing which statement does what. You will probably already have found that most of your programs have what are technically known as bugs when you first type them in: maybe just typing errors, or maybe mistakes in your own ideas of what the program should do. You might put this down to inexperience, but you would be deluding yourself.

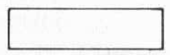
EVERY PROGRAM STARTS OFF WITH BUGS.

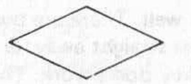
Many programs finish up with bugs as well. There are two corollaries to this; first, you must test all your programs straight away; & second, there's no point in losing your temper every time they don't work. The general plan can be illustrated with a flowchart:


Start
Write program with as few bugs as possible
Test program.
Does it work perfectly?
Yes
Finish
No
Keep your hair on.
Find the bugs.
Fix them, introducing as few new ones as possible.

The idea is that you follow the arrows from box to box, doing what it says at each one. We have used different sorts of boxes for different sorts of instructions:

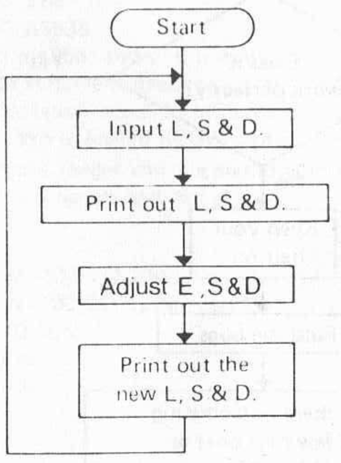
A rounded box  is start or finish.

A rectangular box  is a straightforward instruction.

A diamond  asks you to make some kind of decision before carrying on.

(These shapes are fairly widely used, but nothing earth-shattering depends on them.)

Of course, these flowcharts are ill-adapted for describing human activities; thinking along fixed straight lines like this is not good for creativeness or flexibility. For computers, however, they are just the job. They are best at describing the large-scale structure of programs, with a subroutine in almost every box, so a flowchart for our sterling example in the last chapter might be.



Conceptually, then, rectangular boxes correspond to GOSUBs; although in practice some boxes – like the one above that inputs L, S & D – are translated directly into BASIC statements, without a subroutine.

Anything – like flowcharts, subroutines, & also REM statements – that makes the program clearer gives you a better understanding of it; & then you are bound to write fewer bugs. But subroutines also help you get out the bugs you've written anyway, by making the program easier to test. You will find it much easier to test the subroutines individually & make sure that they fit together properly, than to test a whole unstructed program.

Subroutines, then, help with the box 'find the bugs' & this is the box where you need all the help you can get, for it is often the most exasperating. Other hints for finding bugs are

(i) Check that there are no typing errors. Always do this.

(ii) Try to work out what all the variables should be at each stage – & if possible explain this in REM statements. You can check the variables at a given point in the program by inserting a PRINT statement there.

(iii) If the effect of the program is to make the program stop with an error report, then use this information as thoroughly as you can. Look up the report code, & work out why it stopped on the line where it did. Print out the values of the variables, if necessary.

(iv) You might be able to step through a program line by line by typing in its lines as commands.

(v) Pretend to be the computer: run the program on yourself using pencil & paper to note down the values of the variables.

Once you've found the bugs, fixing them is much like writing the original program, but you just test the program again. It is surprisingly easy to fix one bug, only to introduce another.

APPENDIX A

BASIC FUNCTIONS AND STATEMENTS

Function	Type of argument (X)	Result
ABS	number	Absolute magnitude
ACS	number	Arc cosine in radians AG error if X not in the range −1 to +1
ASN	number	Arcsine in radians AG error if X not in the range −1 to +1
ATN	number	Arctangent in radians
CHR$	number	The character whose code is X, rounded to the nearest integer. IR error if X not in the range Ø to 255.
CODE	string	The code of the first character in X (or Ø if X is the empty string).
COS	number (in radians)	Cosine
EXP	number	e^X
INKEY$	none	Reads the keyboard. The results is the character representing (in normal mode) the key pressed if there is exactly one, else the empty string).
INT	number	Integer part (always rounds down)
LEN	string	Length of the string (Ø if empty string).
LOG	number	Natural logarithm (to base e) AG error if X≤Ø
PEEK	number	The value of the byte in memory whose address is X (round to the nearest integer). IR error if X not in the range Ø to 65535.
PI	none	π (3.14159265)
RND	none	The next pseudo-random number Y in a sequence generated by taking the powers of 75 modulo 65537, subtracting 1 and dividing by 65536. Ø≤Y<1.
SGN	number	Signum, the sign (−1, Ø or +1) of X.
SIN	number (in radians)	Sine
SQR	number	Square root AG error if X<Ø
STR$	number	The string of characters that would be displayed if X were printed.
TAN	number (in radians)	Tangent
USR	number	Calls the machine code subroutine whose starting address is X (rounded to the nearest integer). On return, the result is the contents of the BC register pair. IR error if X is not in the range Ø to 65535.
VAL	string	Evaluates X (without its bounding quotes) as a numerical expression. IE error if X contains a syntax error, or gives a string value. Other errors possible, depending on the expression.

STATEMENTS

in this liast,

a	represents a single letter
v	represents a variable
x, y, z	represents numerical expressions
m,n	represents numerical expressions that are rounded to the nearest integer
e	represents an expression
f	represents a string valued expression
s	represents a statement

Note that arbitrary expression are allowed everywhere (except for the line number at the beginning of a statement).

All statements except INPUT can be used either as commands or in programs (although they may be more sensible in one than the other).

BEEP	Enables the keyboard audio feedback. A beep of different frequencies will be produced for different keys pressed while entering program lines, commands or INPUT data.
CLEAR	Deletes all variables, freeing the space they occupied.
CLS	(Clear Screen) Clears the display file. If RAM size is greater than 1½K then the display file is expanded to its full size.
CONT	Suppose "XX IN YY" was the last report. Then CONT has the effect GOTO YY if XX = ST GOTO YY+1 if XX = ST (STOP statement)
COPY	Sends a copy of the display to the printer, if attached; otherwise does nothing. Unlike all other commands, a COPY command does not clear the screen first. There must be no spaces before COPY. Report BK if BREAK pressed.

DIM a (n1, ... , n_k)	Delete any array with the name a, and sets up an array a of numbers with k dimensions, n1, ... , n_k. Initializes all the values to Ø. OM error occurs if there is no room to fit the array in. An array is undefined until it is dimensioned in a DIM statement.
DIM a+ (n1, ... n_k)	Deletes any array or string with the name a$, and sets up an array a$ of characters with k dimensions n1, ... n_k. Initializes all the values to " ". This can be considered as an array of strings of fixed length n_k, with k-1 dimensions n_1, ... n_{k-1}. OM error occurs if there is no room to fit the array in. An array is undefined until it is dimensioned in a DIM statement.
FAST	Starts fast mode, in which the display file is displayed only at the end of the program, while INPUT data is being typed in, or during a pause.
FOR a=x, TO y	FOR a=x to y STEP 1
FOR a=x TO y STEP z	Deletes any simple variable a, and sets up a control variable with value x, limit y, step z, and looping address 1 more than the line number of the FOR statement (−1 if it is a command). See NEXT a. OM error occurs if there is no room for the control variable.
GOSUB n	Pushes the line number of the GOSUB statement onto a stack; then as GOTO n. OM error can occur if there are not enough RETURNs.
GOTO n	Jumps to n (or, if there is none, the first line after that).
If x then s	If x is true (non-zero) then s is executed. The form 'IF x THEN line number' is not allowed.
INPUT v	Stops (with no special prompt) and waits for the user to type in an expression; the value of this is assigned to v. In fast mode, the display file is displayed. INPUT cannot be used as a command; II error occurs if you try. If BREAK is pressed, the program stops with report BK.

LET v=e	Assigns the value of e to the variable v. "LET" can be omitted.
	A simple variable is undefined until it is assigned to in a LET or INPUT statement.
	If v is a subscripted string variable, or a sliced string variable (substring), then the assignment is Procrustean: the string value of e is either truncated or filled out with spaces on the right, to make it the same length as the variable v.
LIST	LIST O
LIST n	Lists the program to the screen, starting at line n, and makes n the current line.
	OM or SF error if the listing is too long to fit on the screen; CONT will do exactly the same again.
LLIST	LLIST Ø
LLIST n	Like LIST, but using the printer instead of the screen.
	Should do nothing if the printer is not attached.
	Stops with report BK if BREAK pressed.
LOAD f	Looks for a program called f on tape, and loads it and its variables.
	If f=" ", then loads the first program available. If BREAK is pressed or a tape error is detected, then (i) if no program has yet been read in from tape, stops with report BK and old program; (ii) if part of a program has been read in, then executes NEW.
LPRINT ...	Like PRINT, but using the printer instead of the screen. A line of text is sent to the printer. (i) when printing spills over from one line to the next, (ii) after an LPRINT statement that does not end in a comma or a semicolon, (iii) when a comma or TAB item requires a new line, or (iv) at the end of the program, if there is anything left unprinted.

In an AT item, only the column number has any effect; the line number is ignored (except that the same error conditions arise as for PRINT if it is out of range). An AT item never sends a line of text to the printer.

MUSIC f	Produces a melody from the speaker according to the content of the string f and the preselected tempo. The string represents a series of notes, each 2 to 4 characters long. The first one or two characters represent the note and the octave. Sharps are represented by the inverse video characters. The following table gives all the possible notes and their frequencies.

Characters	Note	Frequency (Hz)
C <	$\underset{\cdot}{1}$	131.Ø
$\boxed{C}$ <	# $\underset{\cdot}{1}$	138.3
D <	$\underset{\cdot}{2}$	147.1
$\boxed{D}$ <	# $\underset{\cdot}{2}$	155.5
E <	$\underset{\cdot}{3}$	165.Ø
$\boxed{E}$ <	$\underset{\cdot}{4}$	174.7
F <	$\underset{\cdot}{4}$	174.7
$\boxed{F}$ <	# $\underset{\cdot}{4}$	184.7
G <	$\underset{\cdot}{5}$	195.8
$\boxed{G}$ <	# $\underset{\cdot}{5}$	2Ø7.Ø
A <	$\underset{\cdot}{6}$	219.6
$\boxed{A}$ <	# $\underset{\cdot}{6}$	233.8
B <	$\underset{\cdot}{7}$	246.2
$\boxed{B}$ <	1	262.1
C	1	262.1
$\boxed{C}$	# 1	277.8
D	2	295.5
$\boxed{D}$	# 2	312.5
E	3	331.6
$\boxed{E}$	4	349.5
F	4	349.5
$\boxed{F}$	# 4	369.3
G	5	391.6
$\boxed{G}$	# 5	416.7
A	6	439.2
$\boxed{A}$	# 6	471.Ø
B	7	492.4
$\boxed{B}$	$\dot{1}$	524.2
C >	$\dot{1}$	524.2
$\boxed{C}$ >	# $\dot{1}$	56Ø.3
D >	$\dot{2}$	59Ø.9
$\boxed{D}$ >	# $\dot{2}$	625.Ø
E >	$\dot{3}$	663.3
$\boxed{E}$ >	$\dot{4}$	7Ø6.5
F >	$\dot{4}$	7Ø6.5
$\boxed{F}$ >	# $\dot{4}$	738.6
G >	$\dot{5}$	792.7
$\boxed{G}$ >	# $\dot{5}$	833.3
A >	$\dot{6}$	878.4
$\boxed{A}$ >	# $\dot{6}$	955.9
B >	$\dot{7}$	984.8
$\boxed{B}$ >	$\dot{1}$	1Ø48.4

Following the note are one or two digits specifying how many units of time the note is to last. This note duration unit can be changed by the TEMPO statement, and is initialized to approximately 98.5mS after reset (TEMPO 25). The one or two digits can be from 1 to 99 (a Ø is equivalent to 256). A rest (pause) is just like an ordinary note but with a "." replacing the note characters. Spaces are ignored in interpreting the string and empty string is allowed.

MF error results if the format of the string is incorrect.

NEW	Starts the BASIC system again, deleting program and variables, using the memory up to but not including the byte whose address is in the system variable RAMTOP (bytes 16388 and 16389).
NEXT a	(i) Finds the control variable a. (ii) Adds its step to its value. (iii) If the step ≥ Ø and the value > the limit; or if the step < Ø and the value < the limit, then jumps to the looping line. NF error if there is a simple variable a. UV error if there is no simple or control variable a.
NOBEEP	Turns off the keyboard audio feedback feature. No beep will be produced on pressing the keys.
PAUSE n	Stops computing and displays the display file for n frames (at 5Ø frames per second or 6Ø for USA system) or until a key is pressed. Ø ≤ n ≤ 65535, else IR error. If n ≥ 32768 then the pause is not timed, but lasts until a key is pressed.
PLOT m,n	Turns on the pixel (m.n); moves the PRINT position to just after that pixel. (Ø,Ø) is lower left hand corner. Ø≤ m ≤ 63, Ø ≤ n ≤ 43 else IR error
POKE m,n	Writes the value n to the byte in memory with address m. Ø ≤ m ≤ 65535, Ø ≤ n ≤ 255 else IR error.

PRINT ...	The '...' is a sequence of PRINT items, separated by commas or semicolons, and they are written to the display file for display on the television. The position (line and column) where the next character is to be printed is called the PRINT position.

A PRINT item can be

(i) empty, i.e. nothing

(ii) a numerical expression

First, a minus sign is printed if the value is negative. Now let x be the modulus of the value.

If x $<1Ø^{-5}$ or x $\geq 1Ø^{13}$, then it is printed using scientific notation. The mantissa part has up to eight digits (with no trailing zeros), and the decimal point (absent if only one digit) is after the first. The exponent part is E, followed by + or –, followed by one or two digts.

Otherwise x is printed in ordinary decimal notation with up to eight significant digits, and no trailing zeros after the decimal point. A decimal point right at the beginning is always followed by a zero, so for instance Ø.Ø3 and Ø.3 are printed as such.

Ø is printed as a single digit Ø.

(iii) a string expression.
The tokens in the string are expanded, possibly with a space before or after.

Unused characters and control characters are printed as "?".

(iv) AT m,n
The PRINT position is changed to line m (mounting from the top), column n (counting from the left).
Ø ≤ m ≤ 21, else SF error if m=22 or 23, IR error otherwise.
Ø ≤ n ≤ 31, else IR error.

(v) TAB n
n is reduced modulo 32. Then, the PRINT position is moved to column n, staying on the same line unless this would involve backspacing, in which case it moves on to the next line. Ø ≤ n ≤ 255, else IR error.
A semicolon between two items le- ves the PRINT position unchanged, so that the second item follows on immediately after the first. A comma, on the other hand, moves the PRINT position on at least one

	place, and after that, however many as are nec necessary to leave it in column Ø or 16, throwing a new line if necessary. At the end of the PRINT statement, if it does not end in a semicolon or comma, a new line is thrown. OM error may occur with 1½K or less of memory. SF error means that the screen is filled. In both cases, the cure is CONT, which will clear the screen and carry on.
RND	RAND Ø
RAND n	Sets the system variable (called SEED) used to generate the next value of RND. If n=Ø, then SEED is given the value n; if n=Ø then it is given the value of another system variable (called FRAMES) that counts the frames so for displayed on the television, and so should be fairly random. IR error occurs if n is not in the range Ø to 65535.
REM ...	No effect. '...' can be any sequence of characters except 'ENTER'.
RETURN	Pops a line number from the GOSUB stack, and jumps to the line after it. RG error occurs when there is no line number on the stack.
RUN	RUN Ø.
RUN n	CLEAR, and then GOTO n.
SAVE f	Records the program and variables on tape, and calls it f. SAVE should not be used inside a GOSUB routine. NA error occurs if f is the empty string, which is not allowed.
SCROLL	Scrolls the display file up one line, losing the top line and making an empty line at the bottom. PRINT position is moved to the start of that empty line.

SLOW	**Puts the computer into compute and display mode, in which the display file is displayed continuously, and computing is done during the spaces at the top and bottom of the picture.**
SOUND m,n	**Produces a sound of frequency equal to 325ØØ/m Hz and duration equal to n/65mS in the fast mode. In compute and display mode, the duration is approximately four times longer and the sound is modulated by 5Ø Hz.** **Ø ≤m ≤255, Ø≤n ≤65535, else IR error.** **m=Ø is equivalent to m=256 and n=Ø is equivalent to n=65536.**
STOP	**Stops the program with an ST report. CONT will resume with the following line.**
TEMPO n	**Sets the note duration unit used in the MUSIC statement to n x 3.94mS. After reset a "TEMPO 25" statement is executed automatically.** **Ø ≤n ≤ 255, else IR error.** **n=Ø is equivalent to n=256.**
UNPLOT m,n	**Like PLOT, but blanks out a pixel instead of turning it on.**

APPENDIX B

REPORT MESSAGES

After a command is executed or after a program execution is completed or interrupted, a report message will be displayed showing what has happened and where in the program it happened. If a command is executed then only the message without the line number is displayed. The message remains until a key is pressed.

The following table gives each report message with a general description and a list of the situations where it can occur.

Message	Meaning	Situation
OK	Successful completion, or jump to line number bigger than any existing. A report with "OK" does not change the line number used by CONT.	Any
NF	NEXT without FOR. The control variable does not exist (has not been set up by a FOR statement) but there is an ordinary variable with the same name.	NEXT
UV	Undefined variable. For a simple variable this will happen if the variable is used before it has been assigned to in a LET statement. For a subscripted variable it will happen if the variable is used before it has been dimensioned in a DIM statement. For a control variable this will happen if the variable is used before it has been set up as a control variable in a FOR statement, when there is no ordinary simple variable with the same name.	Any
BS	Bad subscript. If the subscript is hopelessly out of range (negative, or bigger than 65535) then IR error will result.	Subscripted variables
OM	Out of memory. Note that the report message may be incomplete on the screen.	LET, INPUT, DIM, PRINT, LIST, PLOT, UNPLOT, FOR, GOSUB. Sometimes during function evaluation.
SF	Screen full. CONT will make room by clearing the screen.	PRINT, LIST
OV	Overflow. Calculations have led to a number greater than about $1Ø^{38}$.	Any arithmetic
RG	RETURN without GOSUB. No corresponding GOSUB for a RETURN statement.	RETURN
II	Illegal INPUT. Attempt to execute INPUT as a command.	INPUT
ST	STOP statement executed. CONT will execute the next statement.	STOP
AG	Invalid argument to certain functions.	SQR, LOG, ASN, ACS, **
IR	Integer out of range. When an integer is required, the floating point argument is rounded to the nearest integer. If this is outside a suitable range then IR error results.	RUN, RAND, POKE, DIM, GOTO, GOSUB, LIST, LLIST, PAUSE, PLOT, UNPLOT, CHR$, PEEK, USR, SOUND, TEMPO
	For array access, see also "BS".	Array access
IE	Invalid expression The text of the (string) argument of VAL does not form a valid numerical expression.	VAL
BK	Program interrupted by "BREAK" key. In certain circumstances, the "SPACE" key acts as "BREAK". This is recognized: (a) at the end of a statement while a program is running. (b) while the computer is looking for a program on tape, or (c) while the computer is using the printer.	At end of any statement, or in LOAD, SAVE, LPRINT, LLIST, COPY or INPUT
NA	Program name provided is the null string which is not allowed.	SAVE
MF	Music string format incorrect.	MUSIC

APPENDIX C

ORDER OF PRIORITY OF OPERATORS

OPERATORS

The following are operators and their priorities:

Operator	Operation	Priority
**	Raising to a power AG error if the left operand is negative	10
–	Unary minus	9
*	Multiplication	8
/	Division	8
+	Addition (on numbers), or concatensation (on strings)	6
–	Subtraction	6
=	Equal to	5
>	Greater than	5
<	Less than	5
<=	Less than or equal to	5
>=	Greater than or equal to	5
<>	Not equal to Both operands must be of the same type. The result is a number, 1 if the comparison holds, else Ø.	5
NOT	Operand always a number. Gives 1 if operand Ø, else Ø	4
AND	Right operand always a number. Numeric left operand: A AND B = A if B ≠ Ø Ø if B = Ø String left operand: A$ and B = A$ if B ≠ Ø " " if B = Ø	3
OR	Both operands numbers A OR B = 1 if B ≠ Ø A if B = Ø	

Should you need to return your computer for the reason that it cannot work properly, please read this Additional Note and check each item carefully first.

ADDITIONAL NOTE : FOR TROUBLE SHOOTING

Item	Symton	Cause	Remedy
1	The power indicator LED not light, no image on the screen and no "beep"	1. Power is not supplied 2. Connections are not completed	1. Check the power source. 2. Connect the cords properly and insert the power plug into AC power properly
2	An image is not displayed & a "beep" is not heard	The computer is not properly initialized	1. Press RESET key. 2. Hold down the ENTER key while press and release the RESET key. 3. Turn OFF the computer then turn On again.
3	A "beep" is heard but no image (a cursor) is displayed on your TV	1. A proper channel is not set 2. Antenna input is not match (Refer to Chapter 1, page 2) 3. The TV cord has been plugged into MONITOR part.	1. Adjust the TV channel to that of the RF modulator inside the computer 2. Used a Resistance Converter to match the Antenna input properly. (Refer to Chapter 1 of the Instruction Manual) 3. Plug the TV cord correctly into TV port.
4	An image on TV screen is disturbed	The TV receiver is not adjusted properly	Adjust tuning, vertical sync. and horizontal sync.
5	Cannot input character from the keyboard	1. A key is held down 2. A erroneous program has been RUN	1. Check if any key on the keyboard not release to normal position, release it properly. 2. See Item 6
6	After your program has been RUN, the computer cannot be controlled (The BREAK function is not initiated)	An erroneous program is entered/Run	1. Press RESET then list out your program to check the program. 2. If RESET key cannot initiated the computer. Use the Cold Reset (held down the ENTER key while press and release the Reset key)
7	After LOAD a program from tape recorder, the computer cannot be controlled.	An erroneous program is loaded.	Same as Item 6

8	A program cannot be loaded	1. The cables connect the computer and the cassette not plug-in properly and correctly. 2. The memory size is not enough to store the program.	1. Check the recorder's cable and plug it properly and correctly 2. Use an expansion RAM pack to expand the memory size
9	When plug the RAM pack in the computer cannot be initiated	1. The RAM pack is not connect to the computer properly. 2. The RAM pack is damaged.	Check if the edge connector of the RAM pack have been plug properly into the computer. Warning: Each time you plug/unplug the RAM pack to/from the computer, you must turn OFF the computer first, otherwise you will damage the computer and the RAM pack!

LAMDA 8300 1983

Nec D780C-1 cpu (Z80A)

3.25 Mhz

2 KB RAM

VIDEO 32 x24

ATARI Joystick port