

**PERSONAL COMPUTER**

**CASIO PB-100**

---

**INSTRUCTION MANUAL**

---

Thank you very much for purchasing the Casio PB-100. This unit is a "handy type" personal computer which is excellent for those who are beginning to learn about computers.

By using this handy personal computer, you can enter the world of computers and can freely perform programming using BASIC language.

---

## CONTENTS

Prior to Operation .....	3
Use Precautions .....	3
Power Supply and Battery Replacement .....	4
Chapter 1 Name and Operation of Each Section .....	5
1-1 Name of Each Section .....	5
1-2 How to Read the Display .....	10
Chapter 2 Prior to Calculating .....	11
2-1 Contrast Adjustment .....	11
2-2 RAM Pack for Expansion .....	11
2-3 Memory Expansion .....	12
2-4 Auto Power Off .....	13
Chapter 3 How to Calculate .....	14
3-1 Calculation Priority Sequence .....	14
3-2 Input/Output Number of Positions and Operation Number of Positions .....	14
3-3 How to Perform Fundamental Calculations .....	15
3-4 Callout of Previous Calculation Result .....	16
3-5 Error Messages .....	17
3-6 Key Operation .....	17
Chapter 4 Manual Calculation .....	20
4-1 What Is Manual Calculation? .....	20
4-2 Operation Method for Manual Calculation .....	20
4-3 Manual Calculation Examples .....	21
4-3-1 How to Perform Fundamental Calculation .....	21
4-3-2 How to Perform Function Calculation .....	23
4-4 Arrays .....	26

This manual explains fundamental computer operation and gives you an outline of BASIC language.

Prior to using the computer, read this manual thoroughly and fully master each function. Be sure to pay attention to the use precautions to insure long life of the unit.

<b>Chapter 5 Program Calculation</b> .....	<b>27</b>
<b>5-1 Program Outline</b> .....	<b>27</b>
<b>5-2 Program Fundamentals</b> .....	<b>28</b>
5-2-1 Constants and Variables .....	28
5-2-2 Substitution Statements .....	29
<b>5-3 Program Writing and Execution</b> .....	<b>30</b>
5-3-1 Program Writing .....	30
5-3-2 Program Execution .....	32
<b>5-4 Program Editing</b> .....	<b>34</b>
<b>5-5 Program Debug</b> .....	<b>40</b>
<b>5-6 Program Commands</b> .....	<b>44</b>
5-6-1 Input Command (INPUT, KEY) .....	44
5-6-2 Output Command (PRINT, CSR) .....	45
5-6-3 Jump Command (GOTO) .....	47
5-6-4 Judgement Command (IF ~ THEN) .....	48
5-6-5 Loop Command (FOR-NEXT) .....	49
5-6-6 Subroutine Command (GOSUB, RETURN) .....	51
5-6-7 Multistatement .....	54
5-6-8 Stop Command (STOP) .....	54
5-6-9 End Command (END) .....	54
5-6-10 Execute Command (RUN) .....	54
5-6-11 List Command (LIST) .....	55
5-6-12 Mode Designation (MODE) .....	55
5-6-13 Output Format (SET) .....	56
5-6-14 Character Functions (LEN, MID, VAL) .....	56
5-6-15 Memory Clear (VAC) .....	57
5-6-16 Program Clear (CLEAR, CLEAR A) .....	57
5-6-17 Option Specifications .....	58
Cassette Magnetic Tape (SAVE, LOAD, SAVE A, LOAD A, PUT, GET, VER) .....	58
Printer .....	60
<b>Error Message List</b> .....	<b>61</b>
<b>Program Command List</b> .....	<b>62</b>
<b>Function Digit Capacity</b> .....	<b>64</b>
<b>Specifications</b> .....	<b>65</b>

## Prior to Operation

This computer is delivered to you through our strict testing process, high-level electronics technology and rigid quality control.

In order to insure long life of the computer, please be sure to note the following precautions.

### ■ Use Precautions

- Since the computer is constructed using precision electronics parts, never attempt to take it apart. Also, do not subject the computer to shock such as throwing or dropping or to extreme temperature variations. Be especially careful to avoid locations where there is high temperature, high humidity or a lot of dust. Furthermore, if the ambient temperature is low, the display response speed may be slow or there may be no display. However, it will return to normal when normal temperature conditions are resumed.
- Do not attempt to connect any equipment to the adaptor socket other than our exclusive optional equipment.
- While the computer is operating, a "--" will be displayed. At this time, key operation will be ineffective except for one section. Therefore, always be sure to press the keys while confirming the display.
- Be sure to replace the batteries every 2 years regardless of the amount of use. If worn out batteries are used, they will leak and may cause a malfunction so never leave them inside the computer.
- For care of the computer, avoid using thinner or benzine. Wipe off with a soft, dry cloth or use a cloth which has been dampened with a neutral detergent.
- In case of malfunction, contact the store where it was purchased or a nearby dealer.
- Prior to seeking maintenance, please read this manual again and also check the power supply as well as program or operational error.

## ■ Power Supply and Battery Replacement

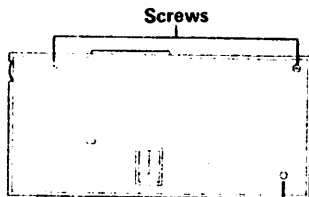
This unit uses two lithium batteries (CR2032) for a power supply.

If the contrast is weak even when the contrast control is adjusted for maximum contrast (refer to page 11), this means that the batteries are worn out. Therefore, please replace the batteries at the earliest opportunity using the following procedure.

Furthermore, even though the unit is functioning normally, be sure to replace the batteries every 2 years.

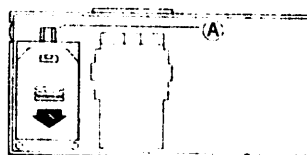
### ● How to Replace the Batteries

(1) After turning the power switch off, loosen the two screws on the rear panel and remove the rear panel.



**ALL RESET button**  
(After replacing the batteries,  
press with a pointed object.)

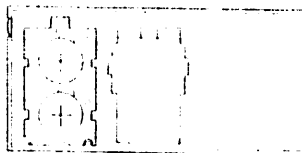
(2) While pressing (A), slide the battery compartment lid in the direction of the arrow and remove it.



(3) Remove the old batteries.

(This will be easier if you tap the unit lightly with the battery compartment facing down.)

(4) Using a dry cloth, wipe off the new batteries and insert them with the ⊕ side facing up.



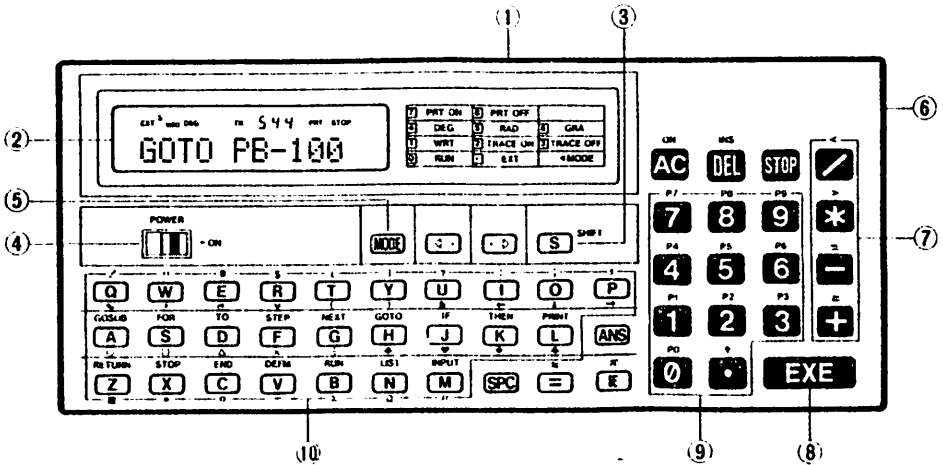
(5) While pressing the batteries down with the battery compartment lid, slide it closed.

(6) Replace the rear panel and tighten the screws and after turning the power switch on, press the ALL RESET button.

- \* Be sure to replace both batteries.
- \* Never throw the old batteries into a fire. This is very dangerous as they might explode.
- \* Be sure to position the ⊕ and ⊖ terminals correctly.

# Chapter 1

## Name and Operation of Each Section



- |                     |  |
|---------------------|--|
| ① Adaptor connector | ⑥ Display contrast control             |
| ② Display window    | ⑦ Calculation instruction keys         |
| ③ Shift key         | ⑧ Execute key                          |
| ④ Power switch      | ⑨ Numerical keys and decimal point key |
| ⑤ Mode key          | ⑩ Alphabet keys                        |

### 1-1 Name of Each Section



Each key has 1 or 2 operations. The operations can be divided using the shift out mode, whereby the keys are pressed directly, and the shift in mode, whereby a key is pressed after pressing the **SHIFT** (SHIFT) Key.

**Example:**




GOSUB ----- Shift in mode  
**A** ----- Shift out mode

























## SHIFT **Shift Key (Symbolized by hereafter)**

If this key is pressed, the shift in mode is selected ("S" is displayed) and the shift in functions on the keyboard can be used. Do not confuse  (red letter) with the regular .

## **Mode Key**

This is pressed in conjunction with the  and  through  Keys to designate the computer's condition or angular unit in advance.

-   ..... "EXT" is displayed and the extension mode is designated. Small English letters and special symbols can be used. To release the extension mode press   again.
-   ..... "RUN" is displayed and manual calculation and program execution can be performed.
-   ..... "WRT" is displayed and program write-in and checking/editing can be performed.
-   ..... "TR" is displayed and execution trace can be performed. (See page 43 for details.)
-   ..... When "TR" is displayed, it will be extinguished and the execution trace function will be released.
-   ..... "DEG" is displayed and the angular unit will be designated as "degree".
-   ..... "RAD" is displayed and the angular unit will be designated as "radian".
-   ..... "GRA" is displayed and the angular unit will be designated as "gradient".
-   ..... "PRT" is displayed and if a printer is connected, printout can be performed.
-   ..... When "PRT" is displayed, it will be extinguished and the printout function will be released.

## **Cursor Keys**

Press to move the cursor left or right. If pressed once, it moves one character. If you keep pressing, it will continue to move automatically.

## **All Clear Key**

- Press to clear the entire display.
- If pressed during program execution, program execution will stop.
- When an error message is displayed, press to clear the error message display.
- When auto power off (automatic energy saving function, refer to page 13) is in operation and the display is off, press to turn power back on.

## **Delete/Insert Key**

- Deletes one character at the position of the blinking cursor.
- In the shift in mode, press to open up one character space for character insertion.



## Stop Key

pressed during program execution, "STOP" will be displayed and program execution will stop at the end of the line.

During execution trace with "STOP" on the display, this key will display the program number and the line number.

## Execute Key

When the result of a manual calculation is required, press instead of "=".

In the "WRT" mode, when writing in a program, press to write (store) each line in the computer. If this key is not pressed, nothing will be written in.


In the "RUN" mode, press for data input during program execution or press to continue program execution while "STOP" is displayed.

## Answer Key

For manual calculation, press to call out the calculation result (answer) of the previous calculation.

## Exponent/Pi Key

When inputting exponential value, press after inputting the mantissa portion.

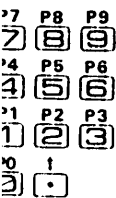
Example:  $2.56 \times 10^{34} \rightarrow$  

The exponential portion may be a maximum of  $\pm 99$ . If this is exceeded, an error will occur.





## Equal Key/Comparison Key

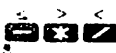
Press when using a substitution statement or for comparison when using an IF statement (equal sign).

In the shift in mode, press for comparison when using an IF statement.




## Numerical Keys/Program Number Keys


- Press when inputting numerical values into the computer. Press  at the location of the decimal point.
- In the shift in mode,  through  become the program number designation keys and when a program has been written in, the program will start.
- The  Key is pressed in the shift in mode when a power ( $x^y$ ) is required.



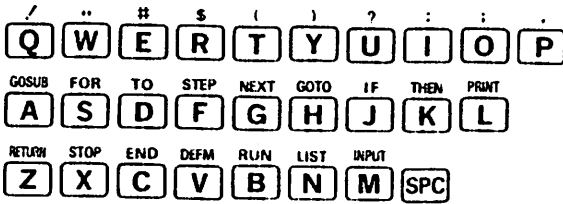
## Calculation Instruction Keys/Comparison Keys

When performing addition, subtraction, multiplication and division, press at the respective locations.

 is used for multiplication (corresponds to "x").

 is used for division (corresponds to  $\div$ ).

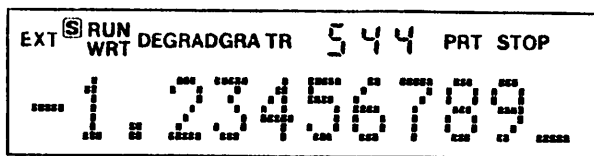
In the shift in mode, press for comparison of a judgement in an IF statement.



### Alphabet Keys/One-Key Command Keys/Character Keys

- When writing in a program or writing a command/function command, if these keys are pressed, letters of the alphabet will be displayed. Press the **SPC** Key when a space is required.
- **Q** ~ **P** Keys: In the shift in mode, the characters which are written on the panel above the keys will be displayed.
- **A** ~ **M** Keys: In the shift in mode, the one-key commands which are written on the panel above the keys will be displayed.

## 1-2 How to Read the Display



Displays the calculation value or result. The respective display positions are composed of 5 horizontal and 7 vertical dots. Up to a maximum of 12 positions are available for display of numbers or characters. (Zero is displayed as 0.) If a formula or statement exceeds 12 positions, the numbers or characters will move to the left and up to a maximum of 62 characters can be input.

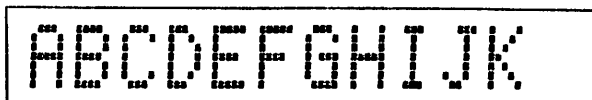
The blinking cursor is displayed until 55 characters have been input. From the 56th character on, a blinking "█" will be displayed instead.

A 4-position numerical display is available on the upper portion of the display to indicate the number of remaining steps.

Furthermore, during operation, a "--" will be displayed in the rightmost position of the 4-position display on the upper portion of the display.

Also, various symbols such as "DEG", "RAD" and "GRA" for angular units, "[S]" (when the [S] Key is pressed), "RUN" (RUN mode), "WRT" (WRT mode), "PRT" (PRT mode) and "STOP" will be displayed to indicate the respective situation.

### • Alphabet display example



### • Symbol display example

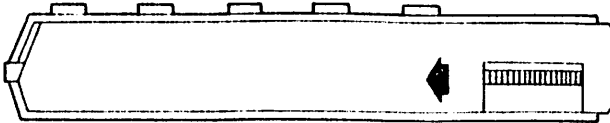


## Chapter 2

# Prior to Calculating

### 2-1 Contrast Adjustment

Adjustment of display contrast can be performed using the adjustment control located on the right side of the computer.



Turn in the direction of the arrow to increase contrast. Turn in the opposite direction to reduce contrast. This is used to compensate contrast of the display in accordance with battery capacity or to adjust to compensate for the viewing angle.

### 2-2 RAM Pack for Expansion (option)

The standard RAM area (memory) of this unit is 544 steps/26 memories. However, this can be increased to a maximum of 1,568 steps/222 memories by using the OR-1 optional RAM pack. This expanded RAM area can be used the same as the standard area and permits step number increase and memory expansion (refer to page 12).

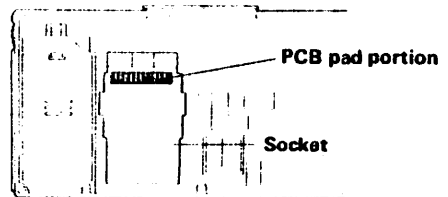
#### ● How to install the RAM pack

##### (preparation)

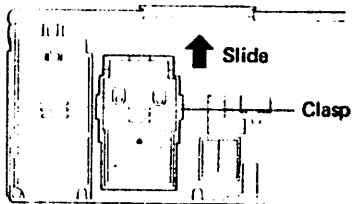
Since the internal circuitry may be damaged by static electricity, prior to handling the pack, be sure to ground yourself by touching some metallic object such as a doorknob so as to discharge any static electricity.

##### (procedure)

- (1) Turn the power switch off.
- (2) Loosen the two screws on the rear panel and remove the rear panel.



(3) Insert the pack into the socket on the computer body and slide the clasp into a locked position.



\*Never touch the connector portion of the RAM pack or the PCB pad portion of the computer body.

(4) Replace the rear panel and tighten the screws.

- After installing or removing the RAM pack, be sure to press the ALL RESET button with a pointed object after turning the power switch on. If the ALL RESET button is not pressed, the memory contents may be changed or a meaningless display may be shown.
- Use care not to allow the connector portion of the pack or the PCB pad portion of the computer body to become dusty or dirty, and avoid getting fingerprints on them as this will cause poor contact.
- Be sure to place the removed pack in its case and store in a location where it is not subject to dust or dirt.

## 2-3 Memory Expansion

There are normally 26 memories (variables). The number of steps at this time is 544. The maximum number of standard memories is 94. Using a RAM pack, this can be expanded to 222. For memory expansion, program steps are converted to memory using 8 steps per memory.

Number of Memories	Number of Program Steps	
	Standard	Expanded
26	544	1568
27	536	1560
28	528	1552
⋮	⋮	⋮
46	384	1408
⋮	⋮	⋮
94	0	1024
⋮	⋮	⋮
200	—	176
⋮	⋮	⋮
222	—	0

Memory expansion is performed in units of 1 using a DEFM command.

**Example:**

Expand by 30 and make 56.

**Operation:**

Select the RUN mode (press **RUN** **2**) or the WRT mode (press **WRT** **1**).

DEFM 30 **EXE**

\*\*\*VAR: 56

\* DEFM can be input by pressing **D E F M** or by pressing **DEFM** **V**.

A DEFM command is also used to confirm the number of memories which are currently designated.

**Example:**

A total of 56 memories are designated.

DEFM

**EXE**

\*\*\*VAR: 56

- When a large number of program steps are already in use, in order to protect the existing program, if a designation is attempted which would cause an insufficient number of steps, an error will occur. (ERR 1 ..... insufficient number of steps)
- The exclusive character variable (\$) is not counted when designating since it is a special memory.

## 2-4 .Auto Power Off

This is an automatic energy-saving function which prevents wasted power consumption when you forget to turn off the power switch. Approximately 7 minutes after the last key operation (except during program execution), power will go off automatically. In this case, power can be resumed by pressing the **AC** Key or turning the power switch off and then on again.

\* Even if power is turned off, memory contents and program contents will not be erased. However, angular unit designation and the respective mode designation ("WRT", "TR", "PRT", etc.) are all released.

# Chapter 3

## How to Calculate

Manual calculation and program calculation are performed in the "RUN" mode. (Press  $\square$   $\square$  and RUN will be displayed.)

Furthermore, with respect to "DEG", "RAD" and "GRA", since these only apply to angular unit, the display of these has no effect for a calculation which has nothing to do with angular unit.

### 3-1 Calculation Priority Sequence (True Algebraic Logic)

This unit determines the calculation priority sequence internally and will perform calculations based on that sequence.

The calculation priority sequence is determined as follows.

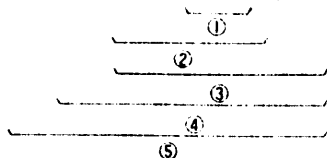
- ① Functions (SIN, COS, TAN, etc.)
- ② Power
- ③ Multiplication and division ( $\times$  and  $\div$ )
- ④ Addition and subtraction (+ and -)

When the priority is the same, calculation will begin from the left.

Furthermore, when parentheses are used, these have the highest priority.

Example:

$$2 + 3 * \text{SIN} (17 + 13) \uparrow 2 = 2.75$$



### 3-2 Input/Output Number of Positions and Operation Number of Positions

The number of input positions for this unit are 12 positions for the mantissa portion and 2 positions for the exponential portion. Internal operations are also performed using 12 positions for the mantissa portion and 2 positions for the exponential portion. The range is  $1 \times 10^{-99} - \pm 9.9999999999 \times 10^{+99}$  and 0.

The number of output positions is 10 positions for the mantissa portion and 2 positions for the exponential portion. However, if an exponential portion is attached, the mantissa portion will be 8 positions.

\* For function results, etc., when the number of display positions (12 positions) is exceeded, up to 12 positions will be displayed, including the 0 and the decimal point.

**Example:**

$$(1 \times 10^5) \div 7 = 14285.71429$$

1 [E] 5 [7] [EXE]

14285.71429

$$(1 \times 10^5) \div 7 - 14285 = 0.7142857$$

1 [E] 5 [7] [=] 14285 [EXE]

0.7142857

When the calculation result exceeds  $10^{10}$  (10,000,000,000) or is below  $10^{-3}$  (0.001), it is automatically displayed using an exponential display.

**Example:**

$$1234567890 \times 10 = 12345678900$$

1234567890 [x] 10 [EXE]

1.2345678E10

$$(= 1.23456789 \times 10^{10})$$

Exponential sign

\* The exponential portion is displayed along with an exponential sign following the mantissa portion.

$$1.234 \div 10000 = 0.0001234$$

1.234 [7] 10000 [EXE]

1.234E-04

$$(= 1.234 \times 10^{-4})$$

### 3-3 How to Perform Fundamental Calculations

#### (1) Calculation symbols and function commands

For calculation symbols used in BASIC, the "+" and "-" are used for addition and subtraction. However, for multiplication and division, "\*" and "/" are used instead of "x" and "÷".

**Example:**

$$2 + 3 - 4 \times 5 \div 6 \text{ becomes } 2 + 3 - 4 * 5 / 6$$

The functions usable with this unit are as follows.

Function Name		Form
Trigonometric function	$\sin x$	SIN $x$
	$\cos x$	COS $x$
	$\tan x$	TAN $x$
Inverse trigonometric function	$\sin^{-1} x$	ASN $x$
	$\cos^{-1} x$	ACS $x$
	$\tan^{-1} x$	ATN $x$
Square root	$\sqrt{x}$	SQR $x$
Exponential function	$e$	EXP 1*
Natural logarithm	$\ln x$	LN $x$
Common logarithm	$\log x$	LOG $x$
Change to integer	INT $x$	INT $x$



Delete fraction from the integer portion	FRAC $x$	FRAC $x$
Change to absolute value	$x$	ABS $x$
Symbolize	positive number $\rightarrow 1$ 0 $\rightarrow 0$ negative number $\rightarrow -1$	SGN $x$
Round off	(round off $x$ at $10^y$ )	RND ( $x, y$ )*
Random number		RAN #

\* In the case of the RND function, the argument must be enclosed in parentheses.

★ EXP is a command to call out the numerical value of the exponential table.

### 3-4 Callout of Previous Calculation Result

The result obtained by executing a manual calculation or program calculation is stored until the next calculation is executed. This result can be displayed by pressing the **ANS** Key.

**Example:**

$$741 + 852 = 1593$$

$$2431 - 1593 = 838$$

**Operation:**

7 4 1 + 8 5 2

**EXE**

2 4 3 1 - ANS

**EXE**

7 4 1 + 8 5 2

1 5 9 3

2 4 3 1 - 1 5 9 3

8 3 8

Also, the numerical value which is displayed following a calculation can be used in the next calculation just as it is.

**Example:**

$$25.3 + 13.9 = 39.2$$

$$39.2 \times 7.6 = 297.92$$

2 5 . 3 + 1 3 . 9 **EXE**

**x** 7 . 6

**EXE**

3 9 . 2

3 9 . 2 \* 7 . 6

2 9 7 . 9 2

### 3-5 Error Messages

If the formula or substitution statement do not conform to BASIC grammar or if the calculation range of the computer is exceeded, an error will occur during execution and an error message will be displayed. Concerning power ( $x \uparrow y$ ) however, when  $y$  is a natural number, an error message will not be displayed even if  $x$  is smaller than 0 (zero). The following error messages are displayed for manual calculation.

ERR 2	(Syntax error)
-------	----------------

ERR 3	(Mathematical error)
-------	----------------------

The following error messages are displayed for program calculation.

ERR 2 P0-10	(A syntax error has occurred on line 10 of program P0.)
-------------	---

ERR 3 P2-30	(A mathematical error has occurred on line 30 of program P2.)
-------------	---

(Refer to page 61 for an explanation of error messages.)

- \* If the calculation result exceeds  $\pm 9.9999999999 \times 10^{99}$ , an overflow will occur and an ERR 3 error message will be displayed. Also, if the result is less than  $1.0 \times 10^{-99}$ , an underflow will occur and the calculation result will become 0.

### 3-6 Key Operation

For manual calculation as well as for program calculation and program write-in, key operation is performed as follows.

#### (1) Alphabetical

Example: Input ABC

Operation: 

A	B	C
---	---	---

ABC
-----

Example: Input SIN

Operation: 

S	I	N
---	---	---

SIN
-----

#### • Numerical input

Example: Input 123

Operation: 

1	2	3
---	---	---

123
-----

Example: Input 96.3

Operation: 

9	6	.	3
---	---	---	---

96.3
------

● **Symbol input**

**Example:** Input \$#?

**Operation:**

\$#?

**Example:** Input @ ¥ Ω

**Operation:** .... (extension mode designation)

.... (extension mode release)

Extension mode  
EXT  
EXT  
@ ¥ Ω  
@ ¥ Ω

● **Input of numerical value with exponent**

**Example:** Input  $7.896 \times 10^{15}$

**Operation:**

7.896E15

**Example:** Input  $-2.369 \times 10^{-45}$

**Operation:**

-2.369E-45

**(2) Changing input contents (correction, deletion and insertion)**

● **Correction**

For correction, move the cursor to the location to be corrected (using and ) and at that position, press the correct character, number or symbol.

**Example:** Correct "A\$" to "B\$".

A\$ \_

**Operation:** Move the cursor 2 character positions to the left.

Press the Key.

A\$  
B\$

**Example:** Correct "LIST" to "RUN".

LIST \_

**Operation:** Move the cursor 4 character positions to the left.

Press or .

LIST  
RUN \_

● **Deletion**

For deletion, move the cursor to the position to be deleted and press the **[DEL]** Key. Each time the key is pressed, one character is deleted and the characters to the right of that will move one position to the left.

**Example:** Delete one of the "I" characters from "SIIN".

SIIN\_

**Operation:** Move the cursor 2 character positions to the left.

**[←][←]**

Press **[DEL]**.

SIIN  
SIN

**Example:** Delete "X," from "INPUT X, Y".

INPUT X,Y\_

**Operation:** Move the cursor 3 positions to the left.

**[←][←][←]**

Press **[DEL][DEL]**.

INPUT X,Y  
INPUT Y

● **Insertion**

For insertion, move the cursor to a position located just to the right of the character after which you want to make an insertion. At that position, press **[SHIFT][INS]** and one character space will be opened up. Then press the desired character, number or symbol key.

**Example:** Change "T=A\$" to "T\$=A\$".

T=A\$\_

**Operation:** Move the cursor 3 character positions to the left.

**[←][←][←]**

Press **[SHIFT][INS]** and open up one character space.

Press **[SHIFT][=]**.

T=A\$  
T\_ =A\$  
T\$=A\$

**Example:** Change "PRINT X" to "PRINT SIN X".

PRINT X\_

**Operation:** Move the cursor 1 character position to the left.

**[←]**

Press **[SHIFT][INS][SHIFT][INS][SHIFT][INS]**.

Press **[S][I][N]**.

PRINT X  
PRINT \_ X  
PRINT SINX

The above are methods for changing input contents.

# Chapter 4

## Manual Calculation

### 4-1 What Is Manual Calculation?

In this case, calculations are not made automatically by storing calculation formulas as a program.

Calculations are performed manually by substituting the calculation on the right side of the numerical formula for the left side or by calling out the contents of the variable. This is called "manual calculation".

### 4-2 Operation Method for Manual Calculation

- Addition, subtraction, multiplication and division are performed by true algebraic logic operation.  $+$ ,  $-$ ,  $\times$  ( $\times$ ),  $\div$  ( $\div$ ) and  $\text{EXE}$  ( $=$ ) are used.

The  $\text{EXE}$  Key is used to obtain the calculation result.

**Example:**  $12 + 36 - 9 \times 5 \div 4 = 36.75$

**Operation:**

1 2 + 3 6 - 9 \* 5 / 4  
 $\text{EXE}$

12+36-9\*5/4  
36.75

- Function calculation is performed the same as a normal formula, including addition, subtraction, multiplication and division, and data is written following the function command.

**Example:**  $\log 1.23 = 0.0899051114$

**Operation:**

LOG 1.23  
 $\text{EXE}$

LOG1.23  
0.0899051114

- \* In this manual, the frames around the letters of the alphabet and the numbers will be omitted.

**Example:**  $\text{SIN}(15) + 8$   
S I N ( 1 5 ) + 8  $\text{EXE}$

- For memory calculation, etc., when storing the numerical value or calculation result or when totalling, variables are used. Letters of the alphabet (A through Z) or a combination of letters and numbers (when used as an array) are used for variables and they operate as memories.

A substitution formula is used to input a numerical value or a calculation result into a variable.

**Example:** Store 1234 in variable A.

**Operation:** A  $\equiv$  1 2 3 4

EXE

A = 1234

**Example:** Add the result of  $23 \times 56$  to variable K.

**Operation:** K  $\equiv$  K  $\div$  23  $\times$  56

EXE

K=K+23\*56

This method is performed manually similar to a substitution statement in a program.

\* Prior to pressing the EXE Key, corrections can be made by moving the cursor to the position to be corrected and pressing the desired key.

(Refer to page 18.)

\* To clear the entire display, press AC.

## 4-3 Manual Calculation Examples

### 4-3-1 How to Perform Fundamental Calculation

• Addition, subtraction, multiplication and division calculation

**Example:**  $23 + 4.5 - 53 = -25.5$

**Operation:** 23  $\div$  4.5  $\div$  53 EXE

-25.5

**Example:**  $56 \times (-12) \div (-2.5) = 268.8$

**Operation:** 56  $\times$  12  $\div$  2.5 EXE

268.8

**Example:**  $12369 \times 7532 \times 74103 = 6.9036806 \times 10^{12}$  (=6903680600000)

**Operation:** 12369  $\times$  7532  $\times$  74103 EXE

6.9036806 E 12

**Example:**  $1.23 \div 90 \div 45.6 = 2.9970760 \times 10^{-4}$  (=0.00029970760)

**Operation:** 1.23  $\div$  90  $\div$  45.6 EXE

2.9970760 E -04

\* When the result exceeds  $10^{10}$  (10,000,000,000) or is less than  $10^{-3}$  (0.001), it will be displayed exponentially.

**Example:**  $7 \times 8 + 4 \times 5 = 76$

**Operation:** 7  $\times$  8  $\div$  4  $\times$  5 EXE

76

**Example:**  $12 + (2.4 \times 10^5) \div 42.6 - 78 \times 36.9 = 2767.602817$

**Operation:** 12  $\div$  2.4  $\div$  5  $\div$  42.6  $\div$  78  $\times$  36.9 EXE

2767.602817

• Memory calculation

Example:  $12 \times 45 = 540$

$12 \times 31 = 372$

$75 \div 12 = 6.25$

Operation:  $A \text{ [ ] } 12 \text{ [ ] EXE}$

$A \text{ [ ] } * 45 \text{ [ ] EXE}$

$A \text{ [ ] } * 31 \text{ [ ] EXE}$

$75 \text{ [ ] } \div A \text{ [ ] EXE}$

—
540
372
6.25

Example  $23 + 9 = 32$

$53 - 6 = 47$

$45 \times 2 = 90$

$99 \div 3 = 33$

Total 22

Operation:  $M \text{ [ ] } 23 \text{ [ ] } + 9 \text{ [ ] EXE}$

$M \text{ [ ] } 53 \text{ [ ] } - 6 \text{ [ ] EXE}$

$M \text{ [ ] } 45 \text{ [ ] } * 2 \text{ [ ] EXE}$

$M \text{ [ ] } 99 \text{ [ ] } \div 3 \text{ [ ] EXE}$

$M \text{ [ ] EXE}$

22
----

\* For this calculation method, since the result of the respective calculations are not known, when you want to see the calculation results, use the following method.

$23 \text{ [ ] } + 9 \text{ [ ] EXE}$

$M \text{ [ ] } \text{ [ ] } \text{ [ ] } \text{ [ ] EXE$

$53 \text{ [ ] } - 6 \text{ [ ] EXE}$

$M \text{ [ ] } \text{ [ ] } \text{ [ ] } \text{ [ ] EXE$

$45 \text{ [ ] } * 2 \text{ [ ] EXE}$

$M \text{ [ ] } \text{ [ ] } \text{ [ ] } \text{ [ ] EXE$

$99 \text{ [ ] } \div 3 \text{ [ ] EXE}$

$M \text{ [ ] } \text{ [ ] } \text{ [ ] } \text{ [ ] EXE$

$M \text{ [ ] EXE}$

32
47
90
33
22

### 4-3-2 How to Perform Function Calculation

- Trigonometric functions (sin, cos, tan) and inverse trigonometric functions ( $\sin^{-1}$ ,  $\cos^{-1}$ ,  $\tan^{-1}$ )

When using trigonometric/inverse trigonometric functions, be sure to designate the angular unit.

**Example:**  $\sin 12.3456^\circ = 0.2138079201$

**Operation:**  $\boxed{\text{MODE}} \boxed{4} \rightarrow \text{°DEG}$

SIN 12.3456  $\boxed{\text{EXE}}$

0.2138079201

**Example:**  $2 \cdot \sin 45^\circ \times \cos 65.1^\circ = 0.5954345575$

**Operation:**  $2 \boxed{\times} \boxed{\text{SIN}} \boxed{45} \boxed{\times} \boxed{\text{COS}} \boxed{65.1} \boxed{\text{EXE}}$

0.5954345575

**Example:**  $\sin^{-1} 0.5 = 30^\circ$

**Operation:** ASN 0.5  $\boxed{\text{EXE}}$

30

**Example:**  $\cos\left(\frac{\pi}{3} \text{rad}\right) = 0.5$

**Operation:**  $\boxed{\text{MODE}} \boxed{5} \rightarrow \text{°RAD}$

COS  $\boxed{\text{PI}} \boxed{\div} \boxed{3} \boxed{\text{EXE}}$

0.5

**Example:**  $\cos^{-1} \sqrt{\frac{2}{3}} = 0.7853981634 \text{rad}$

**Operation:** ACS  $\boxed{\text{SQRT}} \boxed{2} \boxed{\div} \boxed{3} \boxed{\text{EXE}}$

0.7853981634

**Example:**  $\tan(-35 \text{gra}) = -0.612800788$

**Operation:**  $\boxed{\text{MODE}} \boxed{6} \rightarrow \text{°GRA}$

TAN  $\boxed{-} \boxed{35} \boxed{\text{EXE}}$

-0.612800788

- Logarithmic functions (log, ln) and exponential functions ( $e$ ,  $x^y$ )

EXP ( $e$ ) cannot be used in a continuous calculation.  
Also, it cannot be used in a multistatement.

**Example:**  $\log 1.23 (= \log_{10} 1.23) = 0.0899051114$

**Operation:** LOG 1.23  $\boxed{\text{EXE}}$

0.0899051114

**Example:**  $\ln 90 (= \log_e 90) = 4.49980967$

**Operation:** LN 90  $\boxed{\text{EXE}}$

4.49980967



**Example:**  $e = 2.718281828$

(This function is a command to call out the numerical value of the exponential table.)

**Operation:** EXP 1 **EXE**

2.718281828

**Example:**  $10^{1.23} = 16.98243652$

(To get the antilogarithm of common logarithm 1.23)

**Operation:** 10 **1/x** 1.23 **EXE**

16.98243652

**Example:**  $5.6^{2.3} = 52.58143837$

**Operation:** 5.6 **1/x** 2.3 **EXE**

52.58143837

**Example:**  $123^{1/3} (= \sqrt[3]{123}) = 1.988647795$

**Operation:** 123 **1/x** 1/3 **EXE**

1.988647795

**Example:**  $\log \sin 40^\circ + \log \cos 35^\circ = 0.278567983$

The antilogarithm is 0.5265407845 (logarithmic calculation of  $\sin 40^\circ \times \cos 35^\circ$ )

**Operation:** **MODE** **DEG**

LOG SIN 40 **+** LOG COS 35 **EXE**

-0.278567983

10 **1/x** **ANS** **EXE**

0.5265407845

\* The input range of power ( $x \uparrow y$ ) is  $x > 0$ .

● **Other functions** ( $\sqrt{\quad}$ , SGN, RAN #, RND, ABS, INT, FRAC)

**Example:**  $\sqrt{2} + \sqrt{5} = 3.65028154$

**Operation:** SQR 2 **+** SQR 5 **EXE**

3.65028154

**Example:** Give "1" to a positive number, "-1" to a negative number, and "0" to a zero.

**Operation:** SGN 6 **EXE**

1

SGN 0 **EXE**

0

SGN -2 **EXE**

-1

**Example:** Random number generation (pseudo random number of  $0 < \text{RAN}\# < 1$ )

**Operation:** RAN **EXE**

0.790373907

**Example:** The result of  $12.3 \times 4.56$  is rounded off at  $10^{-2}$ .

$12.3 \times 4.56 = 56.088$

**Operation:** RND **1/x** 12.3 **\*** 4.56 **1/x** 2 **EXE**

\* For RND ( $x, y$ ),  $y$  is  $|y| < 100$

56.1

**Example:**  $|-78.9 \div 5.6| = 14.08928571$

**Operation:** ABS  $\left[ \frac{\text{SWT}}{\text{SWT}} \right] \left[ \frac{\text{L}}{\text{L}} \right] 78.9 \left[ \frac{\text{M}}{\text{M}} \right] 5.6 \left[ \frac{\text{SWT}}{\text{SWT}} \right] \left[ \frac{\text{L}}{\text{L}} \right] \text{EXE}$

14.08928571

**Example:** The integer portion of  $\frac{7800}{96}$  is 81.

**Operation:** INT  $\left[ \frac{\text{SWT}}{\text{SWT}} \right] \left[ \frac{\text{L}}{\text{L}} \right] 7800 \left[ \frac{\text{M}}{\text{M}} \right] 96 \left[ \frac{\text{SWT}}{\text{SWT}} \right] \left[ \frac{\text{L}}{\text{L}} \right] \text{EXE}$

81

\* This function obtains the maximum integer which does not exceed the original numerical value.

**Example:** The decimal portion of  $\frac{7800}{96}$  is 0.25.

**Operation:** FRAC  $\left[ \frac{\text{SWT}}{\text{SWT}} \right] \left[ \frac{\text{L}}{\text{L}} \right] 7800 \left[ \frac{\text{M}}{\text{M}} \right] 96 \left[ \frac{\text{SWT}}{\text{SWT}} \right] \left[ \frac{\text{L}}{\text{L}} \right] \text{EXE}$

0.25

### • Designation of number of effective positions and designation of number of decimal positions

Designation of number of effective positions and number of decimal positions is performed using a "SET" command.

Designation of number of effective positions ..... SET E  $n$  ( $n = 0$  through 9)

Designation of number of decimal positions ..... SET F  $n$

Designation release ..... SET N

\* When the designation of the number of effective positions is "SET E 0", the number of positions is 8.

\* The last designated position will be displayed rounded off.

Furthermore, the original numerical values will remain inside the computer and in the memory.

**Example:**  $100 \div 6 = 16.66666666\cdots$

**Operation:** SET E 4  $\left[ \frac{\text{EXE}}{\text{EXE}} \right]$  (designates 4 effective positions)

100  $\left[ \frac{\text{M}}{\text{M}} \right] 6 \left[ \frac{\text{EXE}}{\text{EXE}} \right]$

1.667E01

**Example:**  $123 \div 7 = 17.57142857\cdots$

**Operation:** SET F 2  $\left[ \frac{\text{EXE}}{\text{EXE}} \right]$  (designates 2 decimal positions)

123  $\left[ \frac{\text{M}}{\text{M}} \right] 7 \left[ \frac{\text{EXE}}{\text{EXE}} \right]$

17.57

**Example:**  $1 \div 3 = 0.33333333\cdots$

**Operation:** SET N  $\left[ \frac{\text{EXE}}{\text{EXE}} \right]$  (releases the designation)

1  $\left[ \frac{\text{M}}{\text{M}} \right] 3 \left[ \frac{\text{EXE}}{\text{EXE}} \right]$

0.3333333333

## 4-4 Arrays

For arrays, one-dimensional arrays are used with letters attached such as  $A(i)$ ,  $B(j)$ , etc. Since these arrays are used both with the normal 26 memories and with expanded memories, pay attention to the following array arrangement.

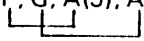
$$\begin{aligned}
 &A=A(0) \\
 &B=A(1)=B(0) \\
 &C=A(2)=B(1)=C(0) \\
 &D=A(3)=B(2)=C(1)=D(0) \\
 &E=A(4)=B(3)=C(2)=D(1)=E(0) \\
 &\vdots \\
 &Y=A(24)=B(23)=C(22)\cdots\cdots\cdots=Y(0) \\
 &Z=A(25)=B(24)=\cdots\cdots\cdots=Y(1)=Z(0) \\
 &\left\{ \begin{array}{l} A(26)=B(25)=\cdots\cdots\cdots=Y(2)=Z(1) \\ A(27)=B(26)=\cdots\cdots\cdots=Y(3)=Z(2) \\ \vdots \\ A(93)=B(92)=\cdots\cdots\cdots=Y(69)=Z(68) \end{array} \right.
 \end{aligned}$$

Expanded memories

When arrays are used in this manner, since the same memory may be used depending on the array argument, avoid using the same memory in the same program.

**Example:**

- Can be used at the same time ..... A, B, C, F(0), F(9)
- Cannot be used at the same time ..... F, G, A(5), A(6)



Perform memory expansion correctly according to the size of the array.

# Chapter 5

## Program Calculation

### 5-1 Program Outline

Program calculation is a method for:

- ① Programming the calculation contents to be executed.
- ② Storing the program in the computer.
- ③ Using that program, obtaining the result automatically by simply inputting data.

Let's examine the programming concept and programming procedure required to process a given problem using the computer.

#### ● Programs and programming

When computer users process a problem using a computer, they compose instructions which are written in a language that the computer can understand. These instructions are called a "program" and composing these instructions is known as "programming".

#### ● What is a program?

In order to make a program, there are various rules or grammar. However, this will be explained later in detail. At this time, let's take a look at an example of a simple, fundamental program to see what it looks like.

	Command	Operand	
10	INPUT	A,B	----- Input statement
20	C=A+B		----- Operation statement
30	PRINT	C	----- Output statement

The above is a fundamental program which consists of an input statement, an operation statement, an output statement and line numbers. An input statement is used to input the data. An operation statement is used to process that data. An output statement is used to output the execution result. Line numbers are used at the beginning of each line. In the case of the operation statement, not only one but many can be used or judgement statements can be added to make a long and complicated program. However, fundamentally speaking, they are all the same. Also, on one line, following the line number, there is a word which consists of letters of the alphabet which has a special meaning and which tells the computer what to do next. This word is called a "command". Following this command is a character string which contains information required to process the command. This is called the "operand".



Furthermore, a character string is a group of characters which is enclosed in quotation marks such as "123". This is not a numerical value. In other words, "123" just happens to be 1 and 2 and 3 in sequence and is considered the same as "ABC". A character variable is made by attaching a "\$" to a regular variable (A, B, X, Y, etc.). These can be freely selected from within this range.

**Example:** A\$, B\$, C\$, X\$, Y\$

Comparison or addition of each character variable is possible but other operations such as subtraction, multiplication and division cannot be performed.

**Example:** If A\$ = "123" and B\$ = "456"

As a result of C\$ = A\$ + B\$, C\$ becomes "123456".  
 (For C\$ = B\$ + A\$, C\$ becomes "456123".)

A character variable can contain up to 7 characters.  
 Also, in addition to these character variables, there is also an exclusive character variable.  
 This exclusive character variable is "\$" and can contain up to 30 characters.

**Example:** \$= "1234567890ABCDEFGHI"

Since this exclusive character variable can use a character function (MID function) which will be explained later, it is much more convenient than other character variables.

\* Numerical variables and character variables which contain the same letter cannot be used at the same time.

Numerical variable A }  
 Character variable A\$ } Cannot be used at the same time.

Since numerical variables and character variables use the same memory, they cannot be used at the same time.

### 5-2-2 Substitution Statements

BASIC substitution statements are in the following format.

Variable = numerical expression

In a BASIC substitution statement, the right side which contains addition, subtraction, multiplication or division is called a "numerical expression".

**Example:** Y=2\*X+3

The "2\*X+3" on the right side is a numerical expression.  
 The "=" does not mean "equal", it means "substitute".

In Y = 2\*X+3, the left side is the variable and the right side is the numerical expression. In other words, the meaning is different from normal mathematics where "the left side (Y) and the right side (2\*X+3) are equal".  
 It means "input the operation result of the right side (2\*X+3) into the left side (Y)".  
 It may be easier to understand by thinking of Y = 2\*X+3 as Y ← 2\*X+3.

## 5-3 Program Writing and Execution

### 5-3-1 Program Writing



Storing a program in the memory device of the computer is called "program writing".

This operation is performed by key input on the keyboard as follows.




1. Designate the WRT mode.
2. Designate the program area.
3. Input the program in line units (write-in).



There are 10 program areas, namely, P0 through P9. Programs can be written in any of these program areas.



#### (1) WRT mode designation



Since program writing is performed in the WRT mode, press   and "WRT" will be displayed.



#### (2) Program area designation



For program area designation, press the  Key then press a numerical key from  through .



  → P0



  → P5



  → P1



  → P6



  → P2

  → P7


  → P3

  → P8





  → P4

  → P9

#### (3) Program input (write-in)

Program writing is performed in line number units. Up to 62 characters can be written in, including the line number. Press  at the end of the line.

##### ● The role of the Key

The  Key is pressed for program writing, data input and to obtain the result of a manual calculation. Since the  Key must be pressed in order to store the key input on the line in the computer, program writing and changes, additions or deletions of the stored program are all followed by pressing the  Key as the final step. Even though the contents are changed on the display, the stored contents will not be changed without pressing the  Key.

**Example:** Write the following program in P0.

10 INPUT A,B

20 V=A+B

30 W=A-B

40 PRINT V,W

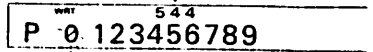
50 END

**Operation:**

① Designate the WRT mode.



Number of remaining steps

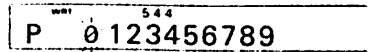


The currently designated program areas with no programs written in program area will blink.

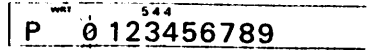
\* This display varies depending on the number of memories or the size of the written program.

\* The area numbers will not be displayed for those areas where programs have already been written.

② Designate program area P0.



③ When a previous program remains, clear it. (Not required if nothing is written.)



\* To clear all the program areas (P0 through P9), press **C L E A R A EXE**.

④ Write line 10.

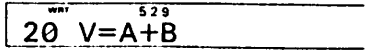


Be sure to press at the end of the line.

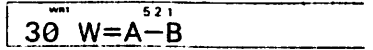
Means one character space (May be omitted)



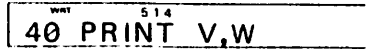
⑤ Write line 20.



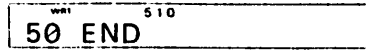
⑥ Write line 30.



⑦ Write line 40.



⑧ Write line 50.



• When the program is complete, write an "END" command. This is not required in the above program but when a GOTO statement or GOSUB statement is used, be



sure to use it to clearly designate the end location.

- The spacing between the line numbers and commands and between commands and operands is to make the display easy to read. In BASIC language, it has no special meaning except for a PRINT statement message, etc. so it may normally be omitted.
- In this program, line numbers have been divided into increments of 10 but they may be freely used within a range of 1 through 9999. However, it is more convenient for subsequent addition/insertion if they are divided into increments of 10. Since program execution is performed in sequence from lower numbers to higher numbers, use line numbers in the desired execution sequence.
- To clear the program in one program area, a CLEAR command was used. However, to clear all of the programs in areas P0 through P9, a CLEAR A command is used.

### 5-3-2 Program Execution

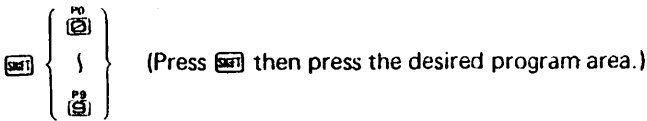
Program execution is performed in the RUN mode. (Press **MODE** and "RUN" will be displayed.)

There are 2 methods for executing a program which has been written.

#### 1. Program execution method

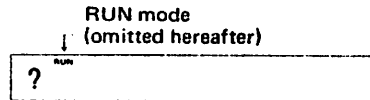
##### ① Execution using program area designation

For this method, execution begins as soon as the program area is designated.



**Example:** To start the program in the previous example

**Operation:** **SHIFT** **P0**



\* This "?" is because an INPUT statement is written in the program as the first step.

##### ② Execution using a RUN command

RUN **EXE** ("RUN" may be input by pressing either

**R** **U** **N** **EXE** or **SHIFT** **R** **EXE** .)



- \* When performing following the previous example, a "?" is displayed. When the program is in an input await condition, "?" will not be released even if **AC** is pressed. Therefore, press **MODE** then perform operation ② to re-input data. Also, to execute from along the way, input the line number after the RUN command then press the **EXE** Key.

**Example:** To start from line 20.

**Operation:** RUN 20 **EXE**

- \* For method (1), it is not necessary to designate the program area to be executed. However, for method (2), it is necessary to designate the program area to be executed. (If the program area is different, the program written in that program area will be executed.)

## 2. Key input during program execution

Key input may be performed during program execution using an INPUT statement and KEY function. Key input using the KEY function is only 1 key input but even if there is no key input, execution will continue. For key input using an INPUT statement, a " ? " will be displayed and the program will stop in an input await condition. Therefore, execution will start by pressing the **EXE** Key after data input.

**Example:** Execute the program written in P0 in the previous example.

**Operation:**

- To execute the program

**STOP** P0 **EXE**

?

- In this program, since 2 variables are input, first, input the value of variable A.

47 **EXE**

?

- Next, input the value of variable B.

69 **EXE**

**EXE**

116

-22

In this manner, data is input during execution using the input statement as follows.

data **EXE**

Also, when in an input await condition as a result of an INPUT statement, other operations such as manual calculation can be performed.

Furthermore, when you want to stop program execution while in an input await condition, press **STOP** **EXE**.

## 5-4 Program Editing

MicroVAX 3100

- Program editing is the performance of changes/additions/deletions in line number units or changing of line numbers in order to permit programs to be logically executed.
- Program editing is performed by calling out each line using a LIST command.
- The LIST command can be used in both the RUN mode and the WRT mode. When used in the RUN mode, the program contents will be displayed and when used in the WRT mode, it will permit program editing.

### 1. Program list display in the RUN mode

Operation:

LIST **EXE**

(LIST may be input by pressing  
[L][I][S][T][**EXE**] or [STOP][LIST][**EXE**].)

10 INPUT A,B	Displayed for approximately 2 seconds (same for the following)
20 V=A+B	
30 W=A-B	
40 PRINT V,W	
50 END	
READY P0	

If the list is not required from the beginning, designate the line number.  
To list from line 30

Operation:

LIST 30 **EXE**

30 W=A-B
40 PRINT V,W
50 END
READY P0

- \* During LIST command execution, each line will be sequentially displayed, so if you want to stop, press the **STOP** Key.  
To resume the stopped LIST command, press the **EXE** Key.

### 2. Program change/addition/deletion in the WRT mode

Designate the WRT mode by pressing **STOP** [1].

#### ① Change

Each time the **EXE** Key is pressed, one line will be displayed starting from the line which was designated using a LIST command.

If the line number designation is omitted, the display will begin sequentially from the first line.

**a. Partial change**

**Example:** Change the "+" on line 20 of the previous example to "\*".

**Operation:**

- If the P0 program area is not designated, designate P0.



```
P 510
  123456789
```

Blinking means that a program is written and this is the currently designated program area.

- Call out line 20 using a LIST command.

LIST 20 EXE

```
20 510
  V=A+B
```

- Move the cursor below the "+".



```
20 510
  V=A±B
```

- \* If a cursor movement keys (← and →) remain pressed for more than 1 second, the cursor will move quickly and continuously.

- Make the change.



```
30 510
  W=A-B
```

- \* Be sure to press the EXE Key. If it is not pressed, only the display will change and the written program will not be changed.

- If left as is, line 30 can be changed, so press AC and the change is complete.



```
510
```

- \* Since any other key operation will result in an unnecessary change being made, avoid pressing any other keys besides the EXE and AC Keys.

- Let's list the program and check the change.



LIST EXE

```
READY P0
10 INPUT A,B
20 V=A*B
30 W=A-B
40 PRINT V,W
50 END
READY P0
```

**b. Complete change of one line**

**Example:** Change "W = A - B" on line 30 to "W = V/2".

**Operation:**

P <sup>510</sup> 123456789

- Write the new line 30.

30

30 W=V/2 <sup>510</sup>

- Confirm the program list.

LIST

```
READY P0
10 INPUT A,B
20 V=A*B
30 W=V/2
40 PRINT V,W
50 END
READY P0
```

**② Addition**

Addition may be made in line units by writing new lines between existing lines.

**Example:** Add "U = V\*2" between line numbers 30 and 40 of the previous example and change line 40 to "PRINT V, W, U".

**Operation:**

P <sup>510</sup> 123456789

- Input line number 35 to make input between line numbers 30 and 40.

35

35 U=V\*2 <sup>502</sup>

\* For inputting between line numbers 30 and 40, line numbers may be freely selected in the range from 31 through 39.

- To change line 40, call it out using a LIST command and add "U".

LIST 40

```
40 PRINT V,W 502
50 END 500
```

- Let's list the program to confirm the additions.

**LIST** **EXE**

**LIST** **EXE**

```

READY P0
10 INPUT A,B
20 V=A*B
30 W=V/2
35 U=V*2
40 PRINT V,W
PRINT V,W,U
50 END
READY P0

```

③ Deletion

a. Partial deletion

**Example:** Delete "V," from line 40 of the previous example.

**LIST** **1**

```

P = 123456789

```

- Call out line 40 using a **LIST** command and move the cursor below the "V".

**LIST 40** **EXE**

```

40 PRINT 500V,W

```

**←** **→**

```

40 PRINT 500V,W

```

- Delete "V," using the **DEL** Key.

**DEL** **DEL**

**EXE**

```

40 PRINT 500W,U
50 END 502

```

- \* If the **EXE** Key is not pressed, the program contents will not be changed.

**AC**

```

502

```

- \* Be sure to press **AC** to release the change condition for line 50.

- List the program to confirm the deletion.

☐☐☐☐

LIST ☐☐

READY P0
10 INPUT A,B
20 V=A*B
30 W=V/2
35 U=V*2
40 PRINT W,U
50 END
READY P0

**b. Complete deletion of one line**

If you input the line number for the line to be cleared, the entire line will be deleted.

**Example:** Delete line 30.

Operation: ☐☐☐☐ 1

P 123456789
-------------

- Input line number 30.

30 ☐☐

510
-

- Confirm the deletion.

☐☐☐☐

LIST ☐☐

READY P0
10 INPUT A,B
20 V=A*B
35 U=V*2
40 PRINT W,U
50 END
READY P0

#### ④ Line renumbering

**Example:** Write the following program in P2.

```
10 INPUT N
20 M=N*N
30 L=SQR N
40 PRINT M,L
50 END
```

Move line 20 between lines 30 and 40.

**Operation:** **MOD** **1**

P \_ 1 \_ 3456789

- Call out line 20 using a LIST command.

LIST 20 **EXE**

20 M=N\*N

- Move the cursor below the "2" of line number "20".

⏪ ⏩ ⏴ ⏵ ⏴ ⏵ ⏴ ⏵

20 M=N\*N

- Change 20 to 35 and input.

35 **EXE**

30 L=SQR N

- To complete the change, press **AC** and release the change condition.

**AC**

-

- List the program to see how the contents have been changed.

**MOD** **2**

LIST **EXE**

READY P2
10 INPUT N
20 M=N*N
30 L=SQR N
35 M=N*N
40 PRINT M,L
50 END
READY P2



- In this condition, the contents on line 20 were moved between line 30 and line 40 but line 20 still remains, so delete it.

MOD (1)

20 EXE

471
P 1 3456789
479

- This completes line renumbering. Confirm by listing the program.

MOD (2)

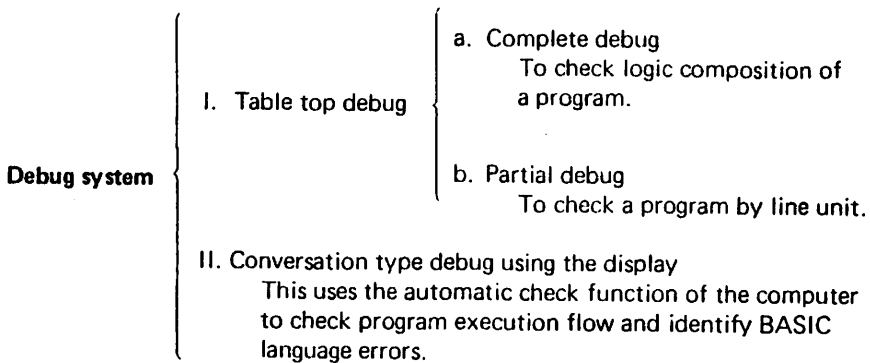
LIST EXE

READY P2
10 INPUT N
30 L=SQR N
35 M=N*N
40 PRINT M,L
50 END
READY P2

## 5-5 Program Debug

### (1) Program debug system

The debug system of this unit is divided into table top debug and conversation type debug using the display.



Since table top debug is performed during programming, we will explain conversation type debug using the display here.

## (2) Conversation type debug

If an error occurs during program execution, an error message will be shown on the display. These errors will be shown in line units and will indicate the kind of BASIC language error. Based on the error message which is shown on the display, debugging is then manually performed while conversing with the display. For the meaning of the error messages, refer to the Error Message List on page 61.

**Example:**

```
10 INPUT X
20 IF X ≤ 0;PRINT "X ≤ 0":GOTO 10
30 Y=X↑2+3*X+15
40 PRINT Y
50 END
```

Line 20 of the above program is a judgement of the input range of power ( $x \uparrow y$ ). When  $x < 0$ , program execution will return to the INPUT statement on line 10.

Suppose line 30 of the above program is mistakenly input as follows.

```
30 Y=X↑2+3X+15
```

**Operation:**

- If this program is executed, a "?" will be displayed as a result of the INPUT statement on line 10.

  RUN 

?

- Suppose "45" is input at this time. The display would show.

45 

ERR2 P0-30

- This means that "a syntax error occurred on line 30". Confirm the program contents. program contents.

  (1)

LIST 30 

P \_ 123456789

30 Y=X↑2+3X±

- The "\*" was omitted between "3" and "X" on line 30. Therefore, correct it by following the procedure for program editing.

30 Y=X↑2+3\_X

40 PRINT Y

### (3) Debug while executing the program

Conversation type debug is performed by obtaining information from the computer using an error message. However, when an error message is not displayed but the calculation result is not as it should be, repeat program execution and confirm the calculation results along the way.

There are two ways to do this: (1) the execution process is stopped using a STOP command; (2) execution is performed in one line unit using the TR (trace) mode.

#### ■ Debug using a STOP command

**Example:** Write the following program.

```
10 Y=0
20 INPUT N,X
30 FOR I=1 TO N
40 Y=Y+X*X
50 NEXT I
60 PRINT Y
70 END
```

In order to see the value of Y in this FOR-NEXT loop, the result of each loop is viewed using a STOP statement.

#### Operation:

- The STOP statement should be placed right after the calculation formula. Therefore, write a STOP statement between line 40 and line 50.

 F1

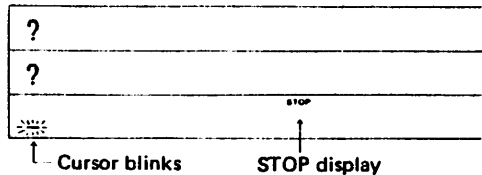
45 STOP 

- As a result of this, after the calculation on line 40 is completed, execution processing will stop and a check can be made.

 F3  F4 RUN  F5


4 

87 



- What is the value of Y at this point?

Y 

7569 

- If the program is resumed, it will stop at the next STOP statement and the value of Y can be obtained again.

EXE  
Y EXE

-	STOP
15138	STOP

● **By repeating this operation, the calculation process can be seen.**

This example used a simple program but when actually assembling a complicated program, checking the process using table top debug is very difficult. So, if the variables are checked using this kind of STOP statement, programming mistakes can be found and corrected more easily.

■ **Debug using the TR (trace) mode**

If program execution is performed using the TR mode (press **TR** (2) ), execution will stop at each line and execution debug can be performed easily.

Let's use the TR mode to debug the example which was previously debugged using a STOP command.

**Operation:**

Designate the RUN mode. .... **TR** (2)

Designate the TR mode. .... **TR** (2)

**RUN** **EXE**

**EXE**

Check the execution process. .... **STOP**

**STOP**

Continue program execution. .... **EXE**

**4** **EXE**

**87** **EXE**

**STOP**

**EXE**

**EXE**

**EXE**

The value of Y ..... **Y** **EXE**

**EXE**

Repeated hereafter

READY P0		
READY P0	TR	
P0-10	TR	STOP
?	TR	
?	TR	STOP
P0-20	TR	STOP
?		"TR" and "STOP" will be omitted hereafter.
?		
-		
P0-20		
P0-30		
P0-40		
7569		
P0-45		

Debug using the TR mode is ideal for checking the entire flow and is convenient for checking to see where mistakes have been made.

## 5-6 Program Commands

### 5-6-1 Input Command

#### • Input statement

An input command is used to input the data during program calculation.

An input statement is used to input data into a variable using the keys during program execution and program execution stops after displaying a "?".

**Format:** INPUT ["character string",] variable [, "character string", variable]  
(Items enclosed in brackets may be omitted.)

The "character string" may be omitted. However, when it is written, since the characters enclosed in quotation marks will be displayed preceding the question mark, this can be used as a message during input.

The variables following the INPUT statement can be numerical variables (A, B, etc.), character variables (X\$, Y\$, etc.) and the exclusive character variable (\$). These can be written consecutively using a ",".

**Example:**

INPUT A

?

INPUT "DATA=" , A

DATA=?

As a result of an INPUT statement, a "?" will be displayed and an input await condition will occur. At this time, if data is input and the **EXE** Key is pressed, program execution will proceed to the next process.

Furthermore, when in an input await condition, even if the **AC** Key is pressed, the condition will not be released. Therefore, when you want to stop the program along the way, press **END** (⏏).

- \* Data which can be input using an INPUT statement includes numerical values or the results (answers) of numerical expression (for numerical variables) and character strings (for character variables).

In the case of INPUT A

Numerical value ..... 123 **EXE** → A=123

Result of a numerical expression ..... 14 \* 25 **EXE** → A=350

In the case of INPUT B\$

Character string ..... ABC **EXE** → B\$=ABC

789 **EXE** → B\$=789

Furthermore, other numerical variables can also be used as input for numerical variables.

In the case of INPUT A (make X = 987654)

Variable ..... X **EXE** → A=X

=987654

### ● KEY function

This function is used to read one character into the character variable by pressing one key during program execution. This function is different from an INPUT statement and does not stop in an input await condition ("?" display). Even when there is no key input, program execution will proceed sequentially.

**Format:** character variable = KEY

A\$, \$, etc. are used for the character variable.

**Example:**

```
10 A$=KEY
20 IF A$= 'A' THEN 100
30 IF A$= 'B' THEN 200
40 IF A$= 'C' THEN 300
50 GOTO 10
.
.
.
```

This program shows the data input using the KEY function and a portion of the distribution. However, a determination will be made using the IF statements whether the character data read by the KEY function on line 10 was input or not. Using the KEY function, even if the **EXE** Key is not pressed, the first key input only will be read. However, since program execution will not stop as it does when using an INPUT statement, an input await condition is achieved by incorporating the following IF statements.

The IF statements on lines 20 through 40 are judgement commands and perform distribution using character variables which were input using the KEY function. For details concerning IF statements, refer to page 48.

## 5-6-2 Output Command

### ● PRINT statement

A PRINT statement is used to display the calculation result or data. It displays the character string, contents of the variable and calculation result following the command.

**Format:** PRINT [ CSR ] [ { { numerical expression } } ] [ { { character expression } } ] [ { { ; } } ..... ]

Either one of the items enclosed in {} can be used.

Items enclosed in [ ] can be omitted.

The output control function following the PRINT statement is a CSR function, which designates the location where the following data is to be displayed. For the numerical expression, a variable or calculation formula is written. In the case of a variable, the contents of the variable will be displayed. In the case of a calculation formula, the calculation result will be displayed.

**Example:**

```
PRINT A      (make A = 12345)
PRINT 789
PRINT A*2    (make A = 147)
PRINT B$     (make B$ = "PB-100")
```

12345
789
294
PB-100

In the case of a character expression, the characters enclosed in quotation marks will be displayed.

**Example:**

```
PRINT 'ABC'
PRINT 'XYZ' + '123'
```

ABC
XYZ 123

The numerical expressions or character expressions may be written consecutively by using a ";" or ", ". However, the number of characters which can be written on one line must be 62 or less, including the line number. The number of characters in the character string enclosed in quotation marks must be 30 or less.

The difference between the ";" and the ", " is that with the ";", the numerical expression or character expression will be displayed following the previous expression and with the ", ", the display will go off once and then the next display will be made. When a ";" is not written after the data, "STOP" will be displayed after the data is displayed, and program execution will stop. Therefore, when you want to display the following data or continue program execution, press the **EXE** Key.

● **CSR function**

The CSR function is an output control function and designates the location where the data is to be displayed.

**Format:** PRINT CSR numerical expression\*  $\left\{ \begin{matrix} * \\ ; \end{matrix} \right\} \left\{ \begin{matrix} \text{numerical expression} \\ \text{character expression} \end{matrix} \right\} \dots\dots \left. \right\}$

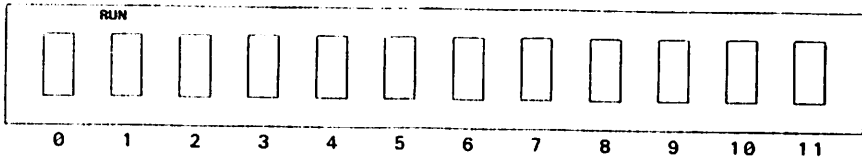
Either one of the items enclosed in { } can be used.

Items enclosed in [ ] can be omitted.

\* The value of the numerical expression is 0 to 11 by disregarding decimal portion.

Using the value of this numerical expression, designation is made as to at which position from the left of the display output data display will begin.

The method for counting the positions on the display is shown below.



**Example:**

```
PRINT A      (make A=12345)
PRINT CSR 1;A
PRINT CSR 5;A
PRINT B$     (make B$ = ABCDE)
PRINT CSR 2;B$
PRINT CSR 10;B$
```

12345
12345
12345
ABCDE
ABCDE
AB
ABCDE

\* If a “,” is used instead of a “;” following the CSR function, the display will be cleared once and then display will be made from the left using the **EXE** Key.

### 5-6-3 Jump Command

● **GOTO statement**

A GOTO statement is also called an “unconditional jump” and is a command which causes program execution to go to a designated location (line number) unconditionally.

**Format:** GOTO { numerical expression ..... line number (1 through 9999)  
 # numerical expression ..... program area number  
 (0 through 9)

When a numerical expression is written immediately following the GOTO statement, program execution jumps to a line number. When a “#” is written immediately following the GOTO statement, program execution jumps to a program area.

The numerical expression may be a numerical value, a variable or a calculation formula.

**Example:**

```
GOTO 10      ..... jump to line 10
GOTO N      ..... jump to the line number which is the value of variable N
              (jump to line 30 if N is 30)
```



- GOTO A\*100 ..... jump to the line number which is the result of A\*100 (jump to line 200 if A is 2)
- GOTO #2 ..... jump to the P2 program area
- GOTO #X ..... jump to the program area which is the value of variable X (jump to the P8 program area if X is 8)
- GOTO #P+1 ..... jump to the program area which is the result of P+1 (jump to the P5 program area if P is 4)

A GOTO statement is used to repeat program execution from the beginning or to jump to another program to perform a particular calculation.

### 5-6-4 Judgement Command

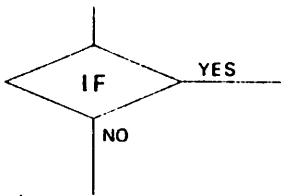
- IF statement

An IF statement is also called a "conditional jump". This command is used to perform some operation or to jump to a designated location only when a certain condition is satisfied.

**Format:** IF comparison expression { THEN line number or #n (n=0 through 9) ; command or substitution statement }

The comparison expression following the "IF" compares the right side and the left side of "=" or "<=" and if YES, program execution proceeds after the "THEN" or ";". If NO, program execution proceeds to the next line.

This operation is shown in the flowchart below.



This means that if the IF statement condition is fulfilled, the process goes in the "YES" direction and if the IF statement is not fulfilled, the process goes in the "NO" direction. In other words, an IF statement indicates a branch and selects the next operation as a result of judgement. An IF statement can be used to terminate a loop (repetition) when the number of data is unknown or to select the next operation based on a calculation result, etc.

Constants/variables/numerical expressions/character constants/character variables can be used for this comparison.

$A > 10$	variable and constant (if A is greater than 10 → YES)
$X \geq Y$	variable and variable (if X is equal to or greater than Y → YES)
$N = L + 3$	variable and numerical expression (if N is equal to the sum of L and 3 → YES)
$A\$ = "XYZ"$	character variable and character constant (if the character string contained in A\$ is equal to "XYZ" → YES)
$P\$ = Q\$$	character variable and character variable (if the character string in P\$ and the character string in Q\$ are equal → YES)

\* Comparison of variables and character variables cannot be made.

\* Character string comparison is based on the ASCII value of individual characters of

"THEN" or ";" are used separately depending upon what follows.

```

THEN 150 (line number)
THEN #9 (program area)
;PRINT A
;Z=X+Y

```

### 5-6-5 Loop Command

#### • FOR-NEXT statement

A FOR-NEXT statement is used when you want to perform similar operations repeatedly and the number of repetitions (loops) is known.

**Format:** FOR variable =  $\frac{n}{\text{initial value}}$  TO  $\frac{m}{\text{final value}}$  [STEP  $\frac{l}{\text{increment}}$ ]  
NEXT variable

(Item enclosed in brackets may be omitted.)  
( $n$ ,  $m$  and  $l$  are numerical expressions.)

In other words, this is a command to repeatedly execute the command between "FOR" and "NEXT" while a variable changes from  $n$  to  $m$  in increments of  $l$ .  
When execution reaches  $m$ , it proceeds to the command following "NEXT".

**Example:**

To increase variable I in increments of 2 between 1 and 10.

```
FOR I=1 TO 10 STEP 2
  )
NEXT I
```

To reduce variable A in increments of 0.5 between 50 and 1.

```
FOR A=50 TO 1 STEP -0.5
  )
NEXT A
```

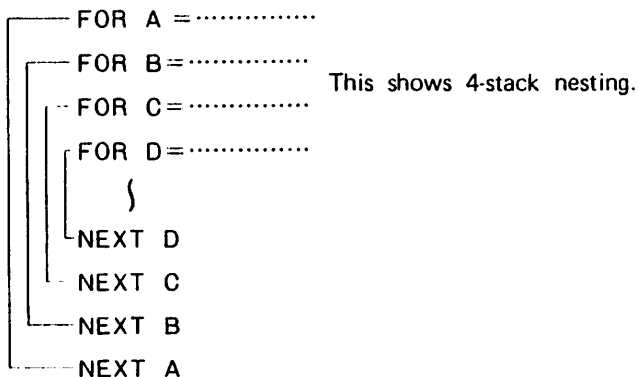
To increase variable P in increments of 1 between Q and R.

```
FOR P =Q TO R
  )
NEXT P
```

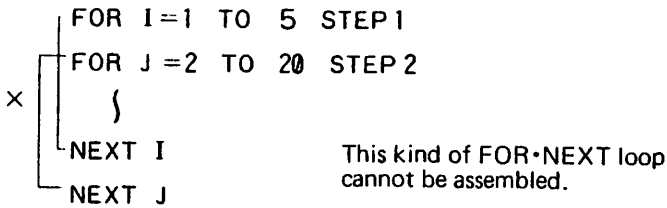
\* When increase is performed in increments of 1, "STEP" may be omitted.

**\* Nesting**

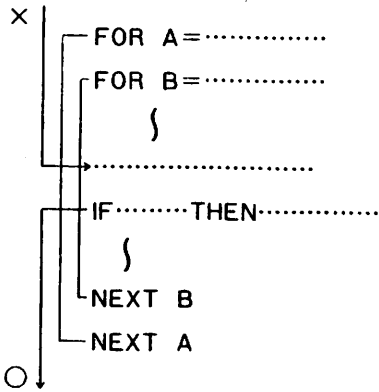
Up to 4 FOR-NEXT loops can be stacked. This stacking is called "nesting".



When nesting is performed in this manner, attention must be paid to the NEXT statement and its variable which correspond to the FOR statement.



Furthermore, exit from FOR-NEXT loop is permitted but entry to FOR-NEXT loop is not permitted.



## 5-6-6 Subroutine Command

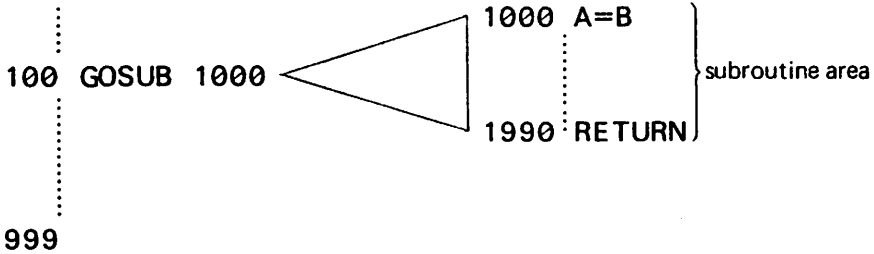
### ● GOSUB statement

A subroutine is also called a "subprogram". It is different from programs shown so far (called "main routines"). It is a separate program to be called out from a main routine. The command to call out this subroutine is a GOSUB statement. Using this command, program execution jumps from the main routine to the subroutine and after executing the program in the subroutine, jumps back to the original location in the main routine using the RETURN statement in the subroutine.

**Format:** GOSUB { numerical expression  
                  # numerical expression ..... subroutine callout (jump) command  
                  RETURN ..... command to return to main routine

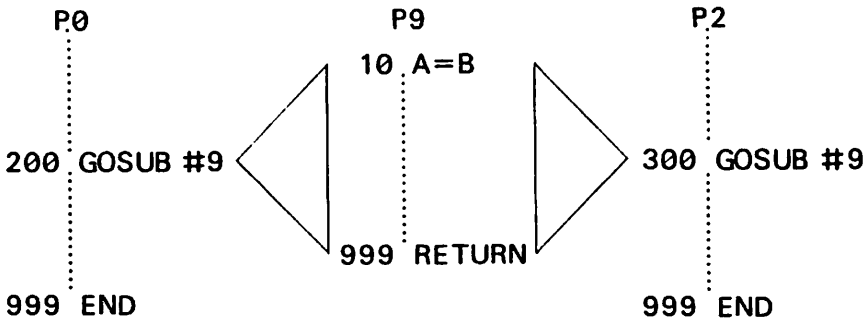
The numerical expression following the GOSUB statement indicates the initial line number of the subroutine area. Without a RETURN statement at the end of the subroutine area, program execution cannot return to the main routine.

Example:



The numerical expression following a GOSUB statement may be either a numerical variable or a calculation formula. In the case of a variable or a numerical expression, the subroutine which is called out will be different depending on the numerical value contained in the variable or the result of the numerical expression. When a '#' is attached prior to the numerical expression, another program area (P0 through P9) will be used as the subroutine. This method is very convenient because even different programs can use the same routine.

Example:

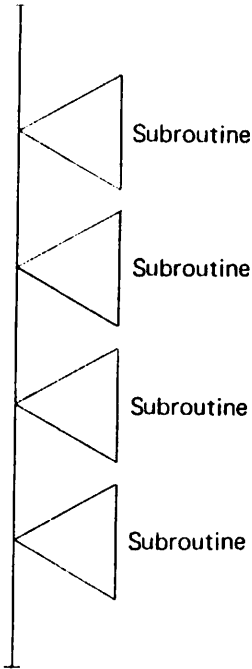


● **Nesting**

Similar to the FOR-NEXT statement, GOSUB statements can also be stacked. The number of times the subroutine is called out is fixed.

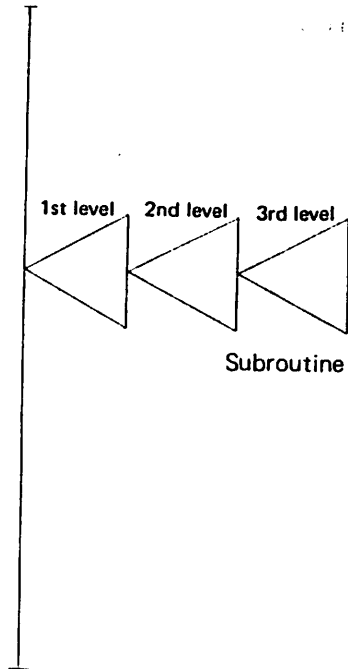
This nesting can be performed up to 8 levels. Therefore, a subroutine can be called out from a subroutine.

**Main routine**



In this example, the subroutines are not stacked so you can use as many as you like.

**Main routine**



This example shows three levels of nesting. Up to 8 levels can be stacked.

A subroutine is convenient for assembling common portions of a main routine in order to save the number of steps or for assembling portions separately as subroutines when assembling a complicated program.

### 5-6-7 Multistatement

A multistatement is used to connect two or more commands using a " : " .

Example:

```

10 A=2
20 B=10 } 10 A=2:B=10:C=50
30 C=50

10 PRINT 'NO.' ;N;
20 INPUT A } 10 PRINT 'NO.' ;N: INPUT A

```

Note: A VAC (memory clear) command cannot be used in a multistatement.

### 5-6-8 Stop Command

• STOP statement

A STOP statement is a command to stop program execution temporarily. When a program is stopped using this command, press the **EXE** Key to resume program execution.

Any number of STOP commands can be written in a program.

### 5-6-9 End Command

• END statement

An END statement is a command to terminate program execution and program execution cannot be resumed as it can with a STOP statement.

This END statement is written at the end of a program. When a subroutine follows the main program, be sure to write an END statement at the end of the main routine.

### 5-6-10 Execute Command

• RUN command

A RUN command is used to execute a program. It cannot be used by writing it in a program.

**Format:** RUN [line number] (Item in brackets may be omitted)

When followed by a line number, the program will start from that line number. If it is omitted, the program will start from the initial line.

Example:

```

RUN EXE ..... start from the beginning
RUN 20 EXE ..... start from line 20
RUN 55 EXE ..... start from line 55

```

When the designated line number does not exist, execution will start from the line with the next nearest line number.

## 5-6-11 List Command

- **LIST command**

A LIST command is used to display the program contents. It can be used in both the "RUN" mode and the "WRT" mode.

**Format:** LIST [line number] (Item in brackets may be omitted)  
LIST A

When followed by a line number, display will be made in sequence starting from the designated line number of the currently designated program area. When no line number is designated, display will be made from the beginning of the program.

In the case of "LIST A", this is a command to display the programs in all program areas. It displays program contents sequentially from P0 through P9. This command cannot be used by writing it in a program.

When performed in the "RUN" mode, the program contents will be displayed sequentially from the designated line. However, when performed in the "WRT" mode, one line is displayed each time the **EXE** Key is pressed.

**Example:**

(RUN mode)		(WRT mode)								
LIST <b>EXE</b>	<table border="1"><tr><td>10 A=0</td></tr><tr><td>20 INPUT B</td></tr><tr><td>30 A=A+B</td></tr><tr><td>40 GOTO 20</td></tr></table>	10 A=0	20 INPUT B	30 A=A+B	40 GOTO 20	LIST 20 <b>EXE</b>	<table border="1"><tr><td>20 INPUT B_</td></tr><tr><td>30 A=A+B_</td></tr><tr><td>40 GOTO 20_</td></tr></table>	20 INPUT B_	30 A=A+B_	40 GOTO 20_
10 A=0										
20 INPUT B										
30 A=A+B										
40 GOTO 20										
20 INPUT B_										
30 A=A+B_										
40 GOTO 20_										

If performed in the "WRT" mode, program editing (refer to page 34) is possible.

## 5-6-12 Mode Designation

- **MODE command**

The MODE command is used to designate the angular unit of printer output condition in a program.

**Format:** MODE *n* (*n* = 4 through 8)

MODE 4	"DEG" designation	} angular unit designation
MODE 5	"RAD" designation	
MODE 6	"GRA" designation	
MODE 7	PRINT mode designation	
MODE 8	PRINT mode release	

This MODE command is the same as the designation which is performed by pressing the **EXE** Key during manual operation.



## 5-6-13 Output Format

### ● SET command

A SET command is used to designate the display output format. It designates the number of effective positions and the number of decimal positions.

**Format:** SET E  $n$  ..... designation of number of effective positions

SET F  $n$  ..... designation of number of decimal positions

SET N ..... designation release  
( $n = 0$  through 9)

\* If SET E 0 is used when designating the number of effective positions, 8 positions will be designated. This command can be performed manually or by writing it in a program. Refer to page 25 for display contents.

## 5-6-14 Character Functions

### ● LEN

The LEN function is used to count the number of characters in a character variable. It permits the size of the character variable to be known.

**Format:** LEN (character variable)

#### Example:

If A\$ = "ABCDE", LEN(A\$) = 5.

### ● MID

The MID function is only used with the exclusive character variable (\$). It extracts a certain number of characters from the character string in the \$ variable.

**Format:** MID ( $m$  [,  $n$ ])  $m$  and  $n$  are numerical expressions and must be between 1 and 30.  
(Items in [ ] may be omitted.)

This means to extract  $n$  characters from the  $m$ th character of the character string stored in the exclusive character variable (\$).

Numerical expression  $m$  should not exceed the number of characters stored. Also,  $m + n$  should not exceed the number of stored characters + 1.

Furthermore, when numerical expression  $n$  is omitted, all of the characters from  $m$  on will be extracted.

#### Example:

If \$ = "PB-100"

MID (2,3) = "B-1" and MID (4) = "100"

- **VAL**

The VAL function changes the numbers in a character variable into a numerical value.

**Format:** VAL (character variable)

Since this function changes the numbers in the character variable into a numerical value, when there are no numbers in the character variable (for example, "ABC"), an error will occur.

**Example:** If Z\$ = \*78963\*, VAL (Z\$) = 78963

**Note:** When this function is used in a program and an error occurs as a result of the data in the variable being other than numbers, "ERR 2" will be displayed and the program area and line number will not be displayed.

### 5-6-15 Memory Clear

- **VAC**

The VAC command clears the data in all variables. It makes numerical variables "0" and makes character variables "null".

This command can be used by writing it in a program or manually. Therefore, when you want to clear all data prior to executing a program, make input at the beginning of the program.

**Example:** Writing in the program

```
10 VAC
   .
   .
   .
```

Manual execution

```
VAC EXT
```

**Note:** This VAC command cannot be used in a multistatement (refer to page 54). Therefore, write it independently on one line.

### 5-6-16 Program Clear

- **Clear command**





A CLEAR command is used to clear a program which has been written. It is executed manually in the "WRT" mode.

**Format:** CLEAR

CLEAR A

A "CLEAR" command only clears the program which is in the currently designated program area (P0, P1, etc.) A "CLEAR A" command clears all programs in all program areas from P0 through P9.

**Example:**

 **1** CLEAR      .....clears a single program  
 **1** CLEAR A      .....clears all programs

### 5-6-17 Option Specifications

#### ■ Cassette Magnetic Tape

In order to record programs or data stored in this unit on a cassette tape, use the FA-3 cassette interface and an ordinary tape recorder. If the tape recorder has a remote terminal, remote control can be conveniently performed from the PB-100 through the FA-3.

For tape recorder connection procedures and detailed operating procedures, refer to the FA-3 operation manual.

#### ● Program recording

**Format:** SAVE ["filename"]     (Item in brackets may be omitted.)

The filename may be composed using 8 characters or less enclosed by quotation marks and may contain alphabetical letters/numbers/symbols.

**Example:**

**^ABC^**  
**^NO.1^**

This command starts the tape recorder in the RECORD position.

**Operation:** SAVE ["filename"] 

A SAVE command can only be used manually.

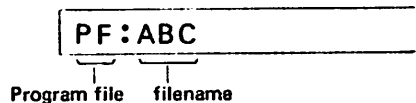
#### ● Program callout

**Format:** LOAD ["filename"]     (Item in brackets may be omitted.)

This command starts the tape recorder in the PLAYBACK position.

**Operation:** LOAD ["filename"] 

Display during program load



Even if a program has already been written in the designated program area prior to callout, the former program will be erased starting with the initial line number of the program to be loaded and the new load will be made correctly.

● **Recording of all programs**

**Format:** SAVE A ["filename"] (Item in brackets may be omitted.)

This command simultaneously records all of the programs which are written in all program areas from P0 through P9. The operation method is similar to the SAVE command and the tape recorder is started in the RECORD position.

**Operation:**

SAVE A ["filename"] **EXE**

● **Callout of all programs**

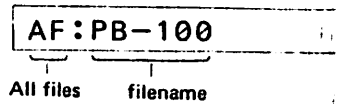
**Format:** LOAD A ["filename"] (Item in brackets may be omitted.)

This command simultaneously calls out the programs of all the program areas which were recorded using the SAVE A command. The operation is similar to the LOAD command and the tape recorder is started in the PLAYBACK position.

**Operation:**

LOAD A ["filename"] **EXE**

Display during program load



Even if programs are already written in program areas prior to callout, the former programs will be cleared and then the new programs will be called out. Furthermore, both the SAVE A command and the LOAD A command can only be used manually.

● **Data recording**

**Format:** PUT ["filename"] variable 1 [, variable 2]  
(Items in brackets may be omitted.)

The data which is recorded on the tape is the data in the variables from variable 1 through variable 2.

**Example:** PUT **"PB"** A .....data of variable A  
 PUT **"1-2"** A,Z .....data of variables A through Z  
 PUT **"DT"** \$,A,Z(10)..... data of character variable \$ and variables A through Z(10)

When recording the data in the exclusive character variable \$, write \$ first. This command can be used either manually or by writing it in a program. For manual operation, start the tape recorder in the RECORD position.

**Operation:**

PUT ["filename"] variable 1 [, variable 2] **EXE**

When performing by writing it in the program, write the PUT command along with the line number and start the written program.

- **Data callout**

**Format:** GET ["filename"] variable 1 [, variable 2]  
(Items in brackets may be omitted.)

This command can be used both manually and by writing it in a program.

For manual use, start the tape recorder in the PLAYBACK position and operate as follows.

GET ["filename"] variable 1 [, variable 2] **EXE**

For use in a program, write it with a line number attached and start the program.

- **Checking of the file which has been recorded on the tape**

A VER command is used to check whether the programs or data have been recorded properly.

**Format:** VER ["filename"] (Item in brackets may be omitted.)

The operation sequence is similar to program load.

- **Printer**

An exclusive mini printer can be connected to the PB-100. By connecting this printer, program lists or data can be extracted. Also, the output of calculation results during execution can be printed out.

For printer connection procedures and operating procedures, refer to the mini printer operation manual.

To print a program list, press **MODE** **2** and designate the PRT mode.

**Program list**

**MODE** **2** **MODE** **7**

**LIST** **EXE** or **LIST A** **EXE**

**MODE** **8** (PRT mode release)

After printout is complete, be sure to press **MODE** **8** and release the PRT mode.

Also, to print calculation results or operation contents, printout can be performed automatically by writing "MODE 7" and "MODE 8" in the program.

**Example:**

```
    )  
100 MODE 7  
110 PRINT A  
120 MODE 8
```

When "MODE 7" is written in the program, be sure to write "MODE 8" prior to program termination and release the PRT mode.

## Error Message List

Error code	Meaning	Cause	Corrective measure
1	Memory overflow or system stack overflow	<ul style="list-style-type: none"> <li>• Program cannot be written due to insufficient number of steps or memory cannot be expanded.</li> <li>• Stack overflow due to a complicated calculation formula.</li> </ul>	<ul style="list-style-type: none"> <li>• Clear unnecessary programs or reduce the number of memories.</li> <li>• Divide and simplify the numerical expression.</li> </ul>
2	Syntax error	<ul style="list-style-type: none"> <li>• A mistake has been made in writing the program, etc.</li> <li>• The left side format is different from the right side format in a substitution statement, etc.</li> </ul>	<ul style="list-style-type: none"> <li>• Correct the error in the input program, etc.</li> </ul>
3	Mathematical error	<ul style="list-style-type: none"> <li>• The calculation result of a numerical expression is <math>10^{100}</math> or greater.</li> <li>• Outside the input range of a numerical function.</li> <li>• The result is indefinite or impossible.</li> </ul>	<ul style="list-style-type: none"> <li>• Correct the calculation formula or data.</li> <li>• Verify the data.</li> </ul>
4	Undefined line number error	<ul style="list-style-type: none"> <li>• No designated line number for a GOTO statement or a GOSUB statement</li> </ul>	<ul style="list-style-type: none"> <li>• Correct the designated line number.</li> </ul>
5	Argument error	<ul style="list-style-type: none"> <li>• For a command or function that requires an argument, the argument is outside the input range.</li> </ul>	<ul style="list-style-type: none"> <li>• Correct the argument error.</li> </ul>
6	Variable error	<ul style="list-style-type: none"> <li>• Attempt was made to use a memory which has not been expanded.</li> <li>• Attempt was made to use the same memory for a numerical variable and a character variable at the same time.</li> </ul>	<ul style="list-style-type: none"> <li>• Expand the memory properly.</li> <li>• Do not use the same memory for a numerical variable and a character variable at the same time.</li> </ul>
7	Nesting error	<ul style="list-style-type: none"> <li>• A RETURN statement appears other than during subroutine execution.</li> <li>• A NEXT statement appears other than during a FOR loop or the variable of the NEXT statement is different from that of the FOR statement.</li> <li>• Subroutine nesting exceeds 8 levels.</li> <li>• FOR loop nesting exceeds 4 levels.</li> </ul>	<ul style="list-style-type: none"> <li>• Remove the unnecessary RETURN statement or NEXT statement.</li> <li>• Reduce the subroutines or FOR-NEXT loops to within the maximum levels.</li> </ul>
9	Option error	<ul style="list-style-type: none"> <li>• Execution is performed in the PRT mode or option command such as SAVE is executed when no printer or tape recorder is connected.</li> </ul>	<ul style="list-style-type: none"> <li>• Connect a printer or tape recorder.</li> <li>• Release the PRT mode.</li> </ul>

## Program Command List

Classification	Command name	Format	Function
Input statement	INPUT	INPUT variable string	Causes data to be entered from the keyboard during execution of a program. The program execution is stopped until after the end of input. <b>P.44</b>
	KEY	Character variable = KEY	Reads a character entered during execution of a program and assigns it to a character variable. Since the program is not stopped by this command, nothing is assigned to the character variable if no key entry is made. <b>P.45</b>
Output statement	PRINT	PRINT output control function { ; } output element { ( ; ) . . . }	Outputs the designated output element. <b>P.45</b>
	CSR	CSR $n \left\{ \begin{matrix} ; \\ , \end{matrix} \right\}$ ( $0 \leq n \leq 11$ )	Displays from the designated $n$ th position. <b>P.46</b>
Branching	GOTO	GOTO { line number } variable	Causes control to jump to the designated line number. <b>P.47</b>
	IF { THEN } ;	IF comparison expression { THEN line number } ; command	Causes control to jump to the line number following THEN, or executes the command following " ; ", if the result of the comparison is true. Causes control to proceed to the next line number if the result of the comparison is false. <b>P.48</b>
	GOSUB	GOSUB { line number } variable	Calls out the subroutine of the designated line number for execution. After the subroutine is executed, control returns to the GOSUB statement by the RETURN statement to proceed to the command following that statement. <b>P.51</b>
	RETURN	RETURN	Signifies the end of a subroutine; returns control to a line number next to the GOSUB statement. <b>P.51</b>
Looping	FOR	FOR $v = e_1$ TO $e_2$ [STEP $e_3$ ]  * $v$ denotes a numerical variable, and $e_1$ , $e_2$ and $e_3$ represent a numerical expression respectively.	Declares the beginning of a loop in which numerical value $v$ changes from initial value $e_1$ to terminal value $e_2$ in increments of $e_3$ . The loop is repeated " $\left[ \frac{e_2 - e_1}{e_3} \right] + 1$ " times between the FOR and NEXT statements. If the increment $e_3$ is omitted, $e_3$ is regarded as " 1 ". <b>P.49</b>

Classification	Command name	Format	Function
Looping	NEXT	NEXT $\nu$	Signifies the end of a FOR loop. If the result of $\nu$ plus $e_3$ is equal to or smaller than $e_2$ , the loop is repeated again. If it is greater than $e_2$ , control proceeds to the line next to the NEXT statement. <b>P.49</b>
Execution stop	STOP	STOP	Stops the execution of a program temporarily to bring the system into a key-in wait state. The execution can be continued by pressing the <b>EXE</b> Key. <b>P.54</b>
Execution end	END	END	Signifies the end of a program, the system returning to its pre-execution state. The execution of a program, once ended, cannot be continued even if the <b>EXE</b> Key is pressed. <b>P.54</b>
Data clearing	VAC	VAC	Clears all variable data for a program. <b>P.57</b>
Program listing	LIST	LIST [line number]	Displays a listing of all the statements in a program from the designated line number downward. <b>P.55</b>
All program listing	LIST A	LIST A	Displays a listing of the statements in all programs. <b>P.55</b>
Program execution	RUN	RUN [line number]	Causes a program to start from the designated line number. <b>P.54</b>
Program erasing	CLEAR	CLEAR	Clears the currently designated program area of a program. <b>P.57</b>
	CLEAR A	CLEAR A	Clears all the programs. <b>P.57</b>
Angular unit designation	MODE	MODE $\left\{ \begin{array}{l} 4 \\ 5 \\ 6 \end{array} \right\}$	Designates trigonometric angular units as degree (4), radian (5) or gradient (6). <b>P.55</b>
Format designation	SET	SET $\left\{ \begin{array}{l} En \\ Fn \\ N \end{array} \right\}$ ( $0 \leq n \leq 9$ )	Designates the number of effective positions or number of decimal positions for the displayed numerical value. <b>P.56</b>
Character function	LEN	LEN (character variable)	Calculates the size of the character variable. <b>P.56</b>
	MID	MID ( $m[, n]$ )	Extracts $n$ characters from the $m$ th character in the exclusive character variable (\$). <b>P.56</b>
	VAL	VAL (character variable)	Converts the numbers in a character variable to a numerical value. <b>P.57</b>



Classification	Command name	Format	Function
Option use	SAVE	SAVE ["filename"]	Records only the program in the currently designated program area on tape. <b>P.58</b>
	LOAD	LOAD ["filename"]	Calls out the program from the tape and loads it to the currently designated program area. <b>P.58</b>
	SAVE A	SAVE A ["filename"]	Records the programs in all program areas on tape at the same time. <b>P.59</b>
	LOAD A	LOAD A ["filename"]	Calls out all programs from the tape and loads them to the respective program areas. <b>P.59</b>
	PUT	PUT ["filename"] variable	Records the data in the variable on tape. <b>P.59</b>
	GET	GET ["filename"] variable	Calls out the data from the tape and loads it in the variable. <b>P.60</b>
	VER	VER ["filename"]	Checks to confirm that the programs or data have been recorded on the tape properly. <b>P.60</b>

\* Items enclosed in [ ] may be omitted.

Either one of the items enclosed in { } may be used.

Function	Digit Capacity	Input range	Result accuracy
$\sin x, \cos x, \tan x$		$ x  < 1440^\circ$ (8 $\pi$ rad, 1600 gra)	10th digit $\pm 1$
$\sin^{-1} x, \cos^{-1} x$		$ x  \leq 1$	"
$\tan^{-1} x$			"
$\log x, \ln x$		$x > 0$	"
$e^x$		$x = 1$	"
$\sqrt{x}$		$x \geq 0$	"
$x^y (x \div y)$		$x > 0$	"

# Specifications

## ■ Type

PB-100

## ■ Fundamental calculation functions

Negative numbers, exponentials, parenthetical addition, subtraction, multiplication and division (with priority sequence judgement function (true algebraic logic))

## ■ Built-in function

Trigonometric/inverse trigonometric functions (angular units – degree/radian/gradient), logarithmic/exponential functions, square root, powers, conversion to integer, deletion of integer portion, absolute value, symbolization, designation of number of effective positions, designation of number of decimal positions, random numbers,  $\pi$

## ■ Commands

INPUT, PRINT, GOTO, FOR-NEXT, IF-THEN, GOSUB, RETURN, STOP, END, RUN, LIST, LIST A, MODE, SET, VAC, CLEAR, CLEAR A, DEFM, SAVE, SAVE A, LOAD, LOAD A, PUT, GET, VER

## ■ Program functions

KEY, CSR, LEN, MID, VAL

## ■ Calculation range

$\pm 1 \times 10^{-99}$  to  $\pm 9.999999999 \times 10^{99}$  and 0 (internal calculation uses 12 mantissa positions)

## ■ Program system

Stored system

## ■ Program language

BASIC

## ■ Number of steps

Maximum 544 steps (maximum 1,568 steps when optional RAM pack is loaded)

## ■ Program capacity

Maximum 10 programs (P0 through P9)

## ■ Number of memories

Standard 26, expandable to 94 (maximum 222 memories when optional RAM pack is loaded) and exclusive character variable (\$)

## ■ Number of stacks

Subroutine – 8 levels  
FOR-NEXT loop – 4 levels  
Numerical value – 6 levels  
Calculation elements – 12 levels

## ■ Display system and contents

10 mantissa positions (including minus sign) or 8 mantissa positions (7 positions for negative number) and 2 exponential positions. Also, display of respective conditions such as EXT,  $\square$ , RUN, WRT, DEG, RAD, GRA, TR, PRT, STOP

■ **Display elements**

12-position dot matrix display (liquid crystal)

■ **Main components**

C-MOS VLSI and others

■ **Power supply**

2 lithium batteries (CR2032)

■ **Power consumption**

Maximum 0.02 W

■ **Battery life**

Mainframe only – approximately 360 hours (Continuous use)

■ **Auto power-off**

Power is turned off automatically approximately 7 minutes after last operation.

■ **Ambient temperature range**

0°C to 40°C (32°F to 104°F)

■ **Dimensions**

9.8H x 165W x 71mmD (3/8"H x 6-1/2"W x 2-3/4"D)

■ **Weight**

116 g (4.1 oz) including batteries

**GUIDELINES LAID DOWN BY FCC RULES FOR USE OF THE UNIT IN THE U.S.A.  
(not applicable to other areas).**

This equipment generates and uses radio frequency energy and if not installed and used properly, that is, in strict accordance with the manufacturer's instructions, may cause interference to radio and television reception. It has been type tested and found to comply with the limits for a Class B computing device in accordance with the specifications in Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference in a residential installation. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- ... reorient the receiving antenna
- ... relocate the computer with respect to the receiver
- ... move the computer away from the receiver
- ... plug the computer into a different outlet so that computer and receiver are on different branch circuits.

If necessary, the user should consult the dealer or an experienced radio/television technician for additional suggestions. The user may find the following booklet prepared by the Federal Communications Commission helpful: "How to Identify and Resolve Radio-TV Interference Problems". This booklet is available from the US Government Printing Office, Washington, D.C., 20402, Stock No. 004-000-00345-4.