

SERVICE POLICY

TANDY

Pocket Computer PC-4 OWNER'S MANUAL

CAT. NO. 26-3650B



CUSTOM MANUFACTURED FOR RADIO SHACK
A Division of Tandy Corporation

U.S.A.: FORT WORTH, TEXAS 76102
CANADA: BARRIE, ONTARIO L4M 4W5

TANDY CORPORATION

AUSTRALIA	BELGIUM	FRANCE	U.K.
91 Kensington Avenue Mount Druitt, N.S.W. 2775	Rue du Plateau d'Assiette, 32 5140 Nellingen (Nantes)	Centre Commercial des 3 Fontaines S.P. 187 95023 Cergy Pontoise Cedex	Shannon Road Westborough West Midlands B510 7JN

Printed in Japan

CUSTOM MANUFACTURED FOR RADIO SHACK, A DIVISION OF TANDY CORPORATION

TERMS AND CONDITIONS OF SALE AND LICENSE OF TANDY COMPUTER EQUIPMENT
AND SOFTWARE PURCHASED FROM RADIO SHACK COMPANY-OWNED COMPUTER
CENTERS, RETAIL STORES AND RADIO SHACK FRANCHISEES OR DEALERS AT THEIR
AUTHORIZED LOCATIONS

LIMITED WARRANTY

- I. **CUSTOMER OBLIGATIONS**
- A. CUSTOMER assumes full responsibility that this computer hardware purchased (the "Equipment"), and any copies of software included with the Equipment or licensed separately (the "Software") meet the specifications, capacity, capabilities, versatility, and other requirements of CUSTOMER.
- B. CUSTOMER assumes full responsibility for the condition and effectiveness of the operating environment in which the Equipment and Software are to function, and for its installation.
- II. **LIMITED WARRANTIES AND CONDITIONS OF SALE**
- A. For a period of ninety (90) calendar days from the date of the Radio Shack sales document received upon purchase of the Equipment, RADIO SHACK warrants to the original CUSTOMER that the Equipment and the medium upon which the Software is stored is free from manufacturing defects. This warranty is only applicable to purchases of Tandy Equipment by the original customer from Radio Shack company-owned computer centers, retail stores, and Radio Shack franchisees and dealers at their authorized locations. The warranty is void if the Equipment's case or cabinet has been opened, or if the Equipment or Software has been subjected to improper or abnormal use. If a manufacturing defect is discovered during the stated warranty period, the defective Equipment must be returned to a Radio Shack Computer Center, a Radio Shack retail store, a participating Radio Shack franchisee or a participating Radio Shack dealer for repair along with a copy of the sales document or lease agreement. The original CUSTOMER'S sole and exclusive remedy in the event of a defect is limited to the correction of the defect by repair, replacement, or refund of the purchase price, at RADIO SHACK'S election and sole expense. RADIO SHACK has no obligation to replace or repair expendable items.
- B. RADIO SHACK makes no warranty as to the design, capability, capacity, or suitability for use of the Software, except as provided in this paragraph. Software is licensed on an "AS IS" basis, without warranty. The original CUSTOMER'S exclusive remedy, in the event of a Software manufacturing defect, is its repair or replacement within thirty (30) calendar days of the date of the Radio Shack sales document received upon license of the Software. The defective Software must be returned to a Radio Shack Computer Center, a Radio Shack retail store, a participating Radio Shack franchisee or Radio Shack dealer along with the sales document.
- C. Except as provided herein no employee, agent, franchisee, dealer or other person is authorized to give any warranties of any nature on behalf of RADIO SHACK.
- D. EXCEPT AS PROVIDED HEREIN, RADIO SHACK MAKES NO EXPRESS WARRANTIES, AND ANY IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE IS LIMITED IN ITS DURATION TO THE DURATION OF THE WRITTEN LIMITED WARRANTIES SET FORTH HEREIN.
- E. Some states do not allow limitations on how long an implied warranty lasts, so the above limitation(s) may not apply to CUSTOMER.
- III. **LIMITATION OF LIABILITY**
- A. EXCEPT AS PROVIDED HEREIN, RADIO SHACK SHALL HAVE NO LIABILITY OR RESPONSIBILITY TO CUSTOMER OR ANY OTHER PERSON OR ENTITY WITH RESPECT TO ANY LIABILITY, LOSS OR DAMAGE CAUSED OR ALLEGED TO BE CAUSED DIRECTLY OR INDIRECTLY BY "EQUIPMENT" OR "SOFTWARE" SOLD, LEASED, LICENSED OR FURNISHED BY RADIO SHACK, INCLUDING, BUT NOT LIMITED TO, ANY INTERRUPTION OF SERVICE, LOSS OF BUSINESS OR ANTICIPATORY PROFITS OR CONSEQUENTIAL DAMAGES RESULTING FROM THE USE OR OPERATION OF THE "EQUIPMENT" OR "SOFTWARE" IN NO EVENT SHALL RADIO SHACK BE LIABLE FOR LOSS OF PROFITS, OR ANY INDIRECT, SPECIAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF ANY BREACH OF THIS WARRANTY OR IN ANY MANNER ARISING OUT OF OR CONNECTED WITH THE SALE, LEASE, LICENSE, USE OR ANTICIPATED USE OF THE "EQUIPMENT" OR "SOFTWARE". NOTWITHSTANDING THE ABOVE LIMITATIONS AND WARRANTIES, RADIO SHACK'S LIABILITY HEREUNDER FOR DAMAGES INCURRED BY CUSTOMER OR OTHERS SHALL NOT EXCEED THE AMOUNT PAID BY CUSTOMER FOR THE PARTICULAR "EQUIPMENT" OR "SOFTWARE" INVOLVED.
- B. RADIO SHACK shall not be liable for any damages caused by delay in delivering or furnishing Equipment and/or Software.
- C. No action arising out of any claimed breach of this Warranty or transactions under this Warranty may be brought more than two (2) years after the cause of action has accrued or more than four (4) years after the date of the Radio Shack sales document for the Equipment or Software, whichever first occurs.
- D. Some states do not allow the limitation or exclusion of incidental or consequential damages, so the above limitation(s) or exclusion(s) may not apply to CUSTOMER.
- IV. **SOFTWARE LICENSE**
- RADIO SHACK grants to CUSTOMER a non-exclusive, paid-up license to use the TANDY Software on one computer, subject to the following provisions:
- A. Except as otherwise provided in this Software License, applicable copyright laws shall apply to the Software.
- B. Title to the medium on which the Software is recorded (cassette and/or diskette) or stored (ROM) is transferred to CUSTOMER, but not title to the Software.
- C. CUSTOMER may use Software on a multiterminal or network system only if either, the Software is expressly licensed to be for use on a multiterminal or network system, or one copy of this software is purchased for each node or terminal on which Software is to be used simultaneously.
- D. CUSTOMER shall not use, make, manufacture, or reproduce copies of Software except for use on one computer and as is specifically provided in this Software License. Customer is expressly prohibited from disassembling the Software.
- E. CUSTOMER is permitted to make additional copies of the Software only for backup or archival purposes or if additional copies are required in the operation of one computer with the Software, but only to the extent the Software allows a backup copy to be made. However, for TRS800 Software, CUSTOMER is permitted to make a limited number of additional copies for CUSTOMER'S own use.
- F. CUSTOMER may rent or distribute unmodified copies of the Software provided CUSTOMER has purchased one copy of the Software for each one sold or distributed. The provisions of this Software License shall also be applicable to third parties receiving copies of the Software from CUSTOMER.
- G. All copyright notices shall be retained on all copies of the Software.
- V. **APPLICABILITY OF WARRANTY**
- A. The terms and conditions of this Warranty are applicable as between RADIO SHACK and CUSTOMER to either a sale of the Equipment and/or Software License to CUSTOMER or to a transaction whereby Radio Shack sells or conveys such Equipment to a third party for lease to CUSTOMER.
- B. The limitations of liability and Warranty provisions herein shall inure to the benefit of RADIO SHACK, the author, owner and/or licensor of the Software and any manufacturer of the Equipment sold by Radio Shack.
- VI. **STATE LAW RIGHTS**
- The warranties granted herein give the original CUSTOMER specific legal rights, and the original CUSTOMER may have other rights which vary from state to state.

This manual explains fundamental computer operation and gives you an outline of BASIC language. Before using the computer, read this manual thoroughly and master each function fully. Be sure to observe the Use Precautions to ensure the longevity of the instrument.

Pocket Computer PC-4 Operation Manual

© 1985 Tandy Corporation
All Rights Reserved.

Reproduction or use, without express written permission from Tandy Corporation and/or its licensor, of any portion of this manual is prohibited. While reasonable efforts have been taken in the preparation of this manual to assure its accuracy, Tandy Corporation assumes no liability resulting from any errors or omissions in this manual, or from the use of the information contained herein.

CONTENTS

Prior to Operation	1
Use Precautions	1
Power Supply and Battery Replacement	2
Chapter 1 Name and Operation of Each Section	3
1-1 Name of Each Section	3
1-2 How to Read the Display	8
Chapter 2 Prior to Calculating	9
2-1 Contrast Adjustment	9
2-2 RAM Expansion Pack	9
2-3 Memory Expansion	10
2-4 Auto Power Off	11
Chapter 3 How to Calculate	13
3-1 Calculation Priority Sequence	13
3-2 Input/Output and Operation Number of Positions	13
3-3 How to Perform Fundamental Calculations	14
3-4 Callout of Previous Calculation Result	15
3-5 Error Messages	16
3-6 Key Operation	16
Chapter 4 Manual Calculation	19
4-1 What Is Manual Calculation?	19
4-2 Operation Method for Manual Calculation	19
4-3 Manual Calculation Examples	20
4-3-1 How to Perform Fundamental Calculation	20
4-3-2 How to Perform Function Calculation	22
4-4 Arrays	25
Chapter 5 Program Calculation	27
5-1 Program Outline	27
5-2 Program Fundamentals	28
5-2-1 Constants and Variables	28
5-2-2 Substitution Statements	29
5-3 Program Writing and Execution	30
5-3-1 Program Writing	30
5-3-2 Program Execution	32
5-4 Program Editing	34
5-5 Program Debug	40

5-6	Program Commands	44
5-6-1	Input Command (INPUT, KEYS)	44
5-6-2	Output Command (PRINT, CSR)	45
5-6-3	Jump Command (GOTO, ON ~ GOTO)	47
5-6-4	Judgment Command (IF ~ THEN)	49
5-6-5	Loop Command (FOR ~ NEXT)	50
5-6-6	Subroutine Command (GOSUB, RETURN, ON ~ GOSUB)	52
5-6-7	Data Processing Command (READ, DATA, RESTORE)	55
5-6-8	Multistatement	56
5-6-9	Stop Command (STOP)	57
5-6-10	End Command (END)	57
5-6-11	Comment Statement (REM)	57
5-6-12	Program Protection Command (PASS)	58
5-6-13	Execute Command (RUN)	59
5-6-14	List Command (LIST)	59
5-6-15	Mode Designation (MODE)	60
5-6-16	Output Format (SET)	60
5-6-17	Character Functions (LEN, MIDS, VAL, STRS)	60
5-6-18	Memory Clear (CLEAR)	62
5-6-19	Program Clear (NEW, NEW ALL)	62
5-6-20	Option Specifications	62
	Cassette Magnetic Tape (SAVE, LOAD, SAVE ALL, LOAD ALL, PUT, GET, VERIFY)	62
	Printer	65
5-6-21	General Functions	65
	Error Message List	69
	Program Command List	70
	Function Digit Capacity	73
	Using a PC-4 (26-3650) program	73
	Specifications	77

Prior to Operation

The PC-4 is a handy personal computer, excellent for those who are beginning to learn about computers.

With the PC-4, you can enter the world of computers and start programming using BASIC language.

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

In order to ensure the longevity of the computer, please be sure to note the following precautions.

■ Use Precautions

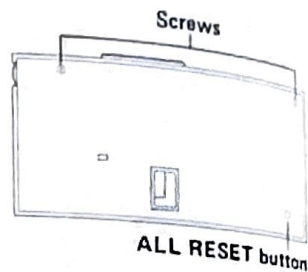
- Since the computer is constructed using precision electronics parts, never attempt to take it apart. Also, do not subject the computer to shock such as throwing or dropping it and avoid extreme temperature variations. Be especially careful to avoid high temperature locations where there is also high humidity or a lot of dust. However, if the ambient temperature is too low, the display response speed may be slow or there may be no display. When normal temperature conditions are resumed, the computer will operate normally.
- Do not attempt to connect any equipment to the adapter socket other than our exclusive optional equipment.
- While the computer is operating, a "--" (dash) 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. Worn out batteries may leak and cause a malfunction. Therefore, never leave old batteries inside the computer.
- To keep the computer clean, wipe off the surface with a soft, dry cloth or one which has been moistened with a neutral detergent.
- In case of a malfunction, contact the Radio Shack store where it was purchased.
- Prior to seeking service, please read this manual again and check the power supply as well as the program. Also, an operational error may be the cause of an apparent malfunction.

■ Power Supply and Battery Replacement

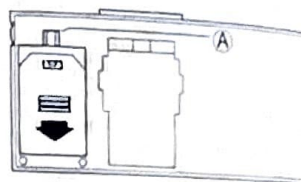
This instrument uses two lithium batteries (CR2032 Cat. No. 23-162) for a power supply. If the display contrast is weak, even when the contrast control is adjusted for maximum (refer to page 9), the batteries should be replaced at the earliest opportunity. Be sure to replace the batteries every 2 years.

● How to Replace the Batteries

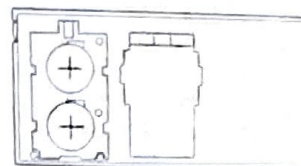
(1) After turning the power switch off, loosen the two screws on the back and then remove the rear panel.



(2) While pressing on (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 ⊕ (positive) side facing up.

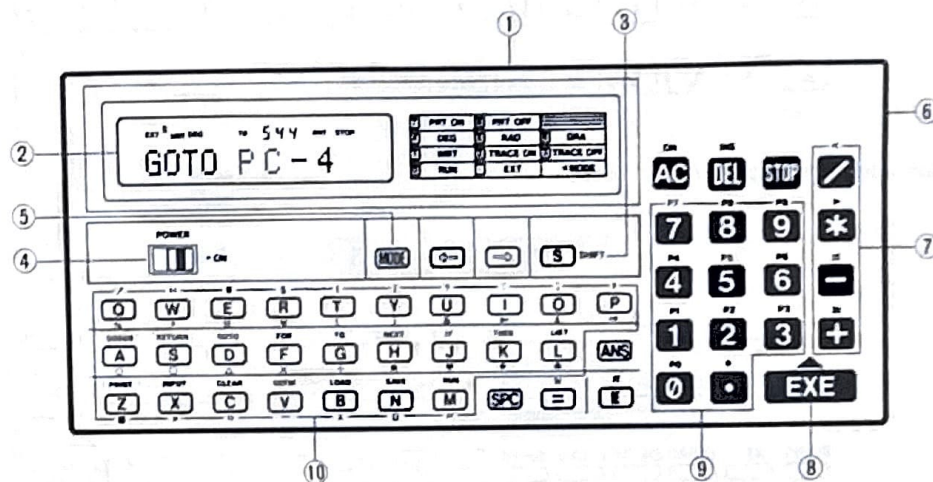
(5) Slide the battery compartment lid closed while pressing down on the batteries.

(6) Replace the rear panel and tighten the screws. After turning the power switch on, press the ALL RESET button with a pointed object.

- 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 ⊕ (positive) and ⊖ (negative) terminals correctly.

Chapter 1


Name and Operation of Each Section



- ① Adapter connector
- ② Display window
- ③ Shift key
- ④ Power switch
- ⑤ Mode key


- ⑥ Display contrast control
- ⑦ Calculation instruction keys
- ⑧ Execute key
- ⑨ Numerical keys and decimal point key
- ⑩ Alphabet keys

1-1 Name of Each Section

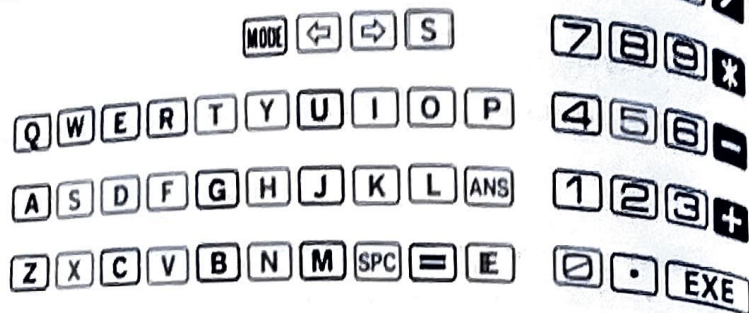
Each key has 1 or 2 operations. The operations can be divided by using the Shift In mode, whereby the keys are pressed directly, and the Shift Out mode, whereby keys are pressed after pressing the  (SHIFT) Key.

Example:

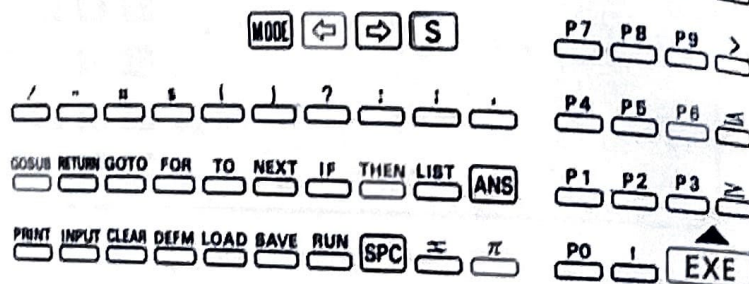
GOSUB — Shift In mode

 A — Shift Out mode

Key operation in the Shift Out mode

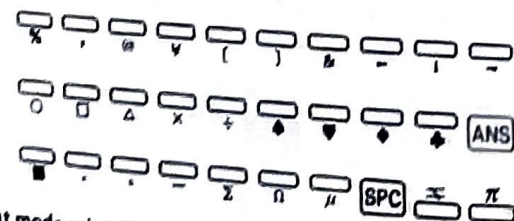


Key operation in the Shift In mode

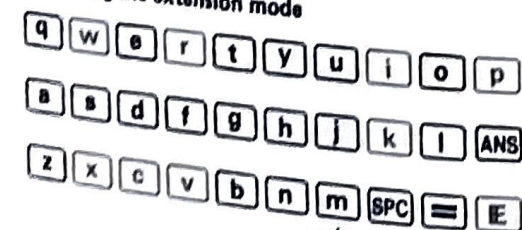


- In the Shift In mode, the alphabet keys become one-key commands and the numerical keys become program area designation keys.
- In addition, using the Extension Mode (press and "EXT" shows on display) while in the Shift In mode results in special symbols displayed for each alphabet key that is pressed. Lower-case letters will be displayed, however, if the Extension mode is used while in the Shift Out mode.

Shift In mode using the extension mode



Shift Out mode using the extension mode



Shift Key (Symbolized by hereafter)

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

Mode Key

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

- "EXT" is displayed. The Extension mode is designated and lower-case letters or special symbols can be used, depending on whether the Shift In or Shift Out mode has been previously selected. To release the Extension mode press again.
- "RUN" is displayed. Manual calculation and program execution can be performed.
- "WRT" is displayed. Program write-in and checking/editing can be performed.
- "TR" is displayed. Execution trace can be performed. (See page 43 for details.)
- "TR" disappears from display. This mode disables the execution trace function().
- "DEG" is displayed. The angular unit will be designated as "degree".
- "RAD" is displayed. The angular unit will be designated as "radian".
- "GRA" is displayed. The angular unit will be designated as "gradient".
- "PRT" is displayed. If a printer is connected, printout can be performed.
- "PRT" disappears from display. This mode disables the printout function ().

Cursor Keys

Press to move the cursor left or right. If pressed once, the cursor moves one character space. If held down, the cursor continues to move in the direction of the arrow on the cursor key.

All Clear/ON 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 is in operation (automatic energy saving function, refer to page 11), 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

If 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 displays the program area number and the line numbers.

Execute Key







- Press  instead of "=" when the result of a manual calculation is required.
- 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.
- While a line of a program is displayed, press   to display the previous line.
- In the "RUN" mode, press for data input during program execution or to continue program execution while "STOP" is displayed.

Answer Key

In manual calculations, press  (answer) to recall the previous calculation result.



Exponent/Pi Key

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





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

* The exponential portion may be a maximum of ± 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.
- 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 the respective keys.
-  is used for multiplication (corresponds to "x").
-  is used for division (corresponds to \div).
- In the Shift In mode, use these keys for comparison of a judgment in an IF statement.

/	"	#	s	()	?	:	.	
Q	W	E	R	T	Y	U	I	O	P
GOSUB	RETURN	GOTO	FOR	TO	NEXT	IF	THEN	LIST	
A	S	D	F	G	H	J	K	L	
PRINT	INPUT	CLEAR	DEFM	LOAD	SAVE	RUN			
Z	X	C	V	B	N	M	SPC		

Alphabet Keys/One-Key Command Keys/Character Keys

- When writing in a program in the Shift Out mode, alphabetical characters are displayed. Press the  Key when a space is required.
-  ~  Keys: In the Shift In mode, the characters written on the panel above the keys are displayed.
-  ~  Keys: In the Shift In mode, the one-key commands which are written on the panel above the keys are displayed.

1-2 How to Read the Display

EXT RUN DEGRADGRA 544 TR PRT STOP
WRT
-1.23456789...

The display shows the calculation value or result. A character may take up an area on the display composed of 5 horizontal and 7 vertical dots. A maximum of 12 positions are available for display of numbers or characters. (Zero is displayed as 0.) However, if a formula or statement exceeds 12 positions, the numbers or characters will move to the left — 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 on the upper portion of the display indicates the number of steps remaining.

In addition a "-" (dash) will be displayed to the right of the 4-position numerical display during operation.

Also, in the Shift In mode, abbreviations for angular units such as "DEG", "RAD" and "GRA" will be displayed. Similarly, "RUN" (RUN mode), "WRT" (WRT mode), "TR" (TR mode), "PRT" (PRT mode), and "STOP" will be displayed to indicate the current mode of operation.

• Alphabet display example

ABCDEFGHIJK

• Symbol display example

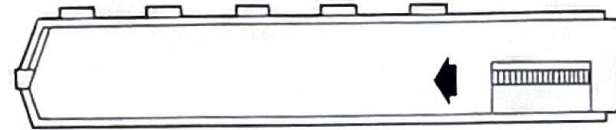
+ - * / = ! " # \$ %

Chapter 2

Prior to Calculating

2-1 Contrast Adjustment

To adjust the display contrast, use the 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.

2-2 RAM Expansion Pack (optional)

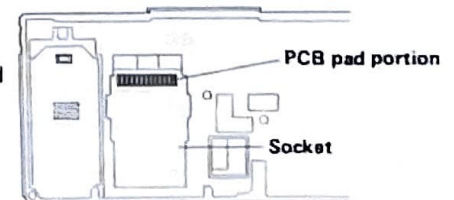
The PC-4 comes with a standard RAM area of 544 possible steps and 26 memories. However, this can be increased to a maximum of 1,568 possible steps and 222 memories with the optional RAM Expansion Pack (Cat. No. 26-3653A). This expanded RAM area can be used the same as the standard area, besides permitting step number increase and memory expansion (Refer to page 10).

• How to install the RAM Expansion Pack

Note: The internal circuitry of the RAM Expansion Pack may be damaged by static electricity. Therefore, before handling the pack, ground yourself to discharge any static electricity by touching a metallic object such as a doorknob.

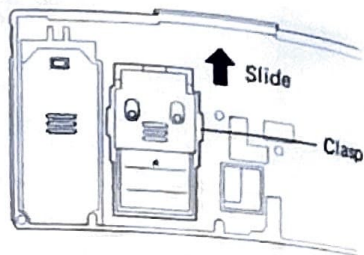
(procedure)

- (1) Turn the power switch off.
- (2) Loosen the two screws on the back 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 turn the power on and press the ALL RESET button with a pointed object. If the ALL RESET button is not pressed, the memory contents may be changed or a meaningless display may be shown.
- Do not 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 memory units (variables). The number of steps at this time is 544. The maximum number of standard memory units is 94. Using the RAM Expansion Pack, the number of memory units can be expanded to 222. For memory expansion, program steps are converted to memory using 8 steps per memory.

Number of Memory Units	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) or the WRT mode (press).

DEFM 30

***VAR: 56

- DEFM can be input by pressing or by pressing .

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

Example:

A total of 56 memories are designated.

Input DEFM

***VAR: 56

The DEFM command can also be written in a program. Use it on the first line of a program.

- If a designation is attempted when a large number of program steps are already in use, ERR 1 appears on the display to indicate there is an insufficient number of available steps and to protect the existing program. (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 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.

Power can be resumed by pressing the 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 designations and mode designations ("WRT", "TR", "PRT", etc.) will be erased.

Chapter 3

How to Calculate

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

"DEG", "RAD" and "GRA", only apply to angular units and their display has no effect for a calculation which has nothing to do with angular units.

3-1 Calculation Priority Sequence (True Algebraic Logic)

The PC-4 has a built-in Calculation Priority Sequence and will perform calculations based on that sequence.

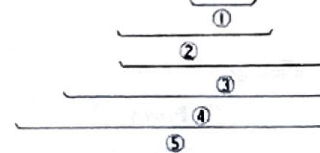
The Calculation Priority Sequence is determined as follows.

- ① Functions (SIN, COS, TAN, etc.)
- ② Power
- ③ Multiplication and division (* and /)
- ④ Addition and subtraction (+ and -)

When the priority is the same, calculation will begin from the left. However, when parentheses are used, they get the highest priority.

Example: (In DEG mode)

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



3-2 Input/Output and Operation Number of Positions

The PC-4 can sustain 12 input 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 extends from 1×10^{-99} to $\pm 9.9999999999 \times 10^{99}$ and 0.

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

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

Example:

$$(1 \times 10^4) \div 7 = 14285.71429$$

$$(1 \times 10^4) \div 7 = 14285 = 0.7142857$$

1 [E] 5 [7] [EXE]

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

14285.71429

0.7142857

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

Example:

$$1234567890 \times 10 = 12345678900$$

1234567890 [x] 10 [EXE]

1.2345678 [E] 10

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

Exponential sign

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

$$1.234 \div 10000 = 0.0001234$$

1.234 [10000] [EXE]

1.234 [E] 04

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

3-3 How to Perform Fundamental Calculations

(1) Calculation symbols and function commands

Calculation symbols used in BASIC include the "+" and "-" signs 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 calculation functions available with the PC-4 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^x	EXP x
Natural logarithm	ln x	LN x
Common logarithm	log x	LOG x
Change to integer	INT x	INT x

Delete the integer portion

Change to absolute value

Symbolize

Round off

Degrees Minutes Seconds

Degrees Minutes Seconds

Random number

* In the case of RND, DEG and DMSS, the argument must be enclosed in parentheses.

FRAC x

|x|

positive number → 1

0 → 0

negative number → -1

(round off x at 10^y)

(Sexagesimal → Decimal)

(Decimal → Sexagesimal)

FRAC x

ABS x

SGN x

RND (x, y)*

DEG (x, y, z)*
Degrees | Minutes | Seconds

DMSS (x)*

RAN #

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:

741 [+ 852]

[EXE]

2431 [- 1593]

[EXE]

741+852 _

1593

2431-1593 _

838

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

Example:

$$25.3 + 13.9 = 39.2$$

$$39.2 \times 7.6 = 297.92$$

25.3 [+ 13.9] [EXE]

[39.2] [7.6] [EXE]

[EXE]

39.2

39.2*7.6 _

297.92

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. The following error messages are displayed for manual calculation.

ERR2	(Syntax error)
ERR3	(Mathematical error)

The following error messages are displayed for program calculation.

ERR2 P0-10	(A syntax error has occurred on line 10 of program P0.)
ERR3 P2-30	(A mathematical error has occurred on line 30 of program P2.)

(Refer to page 69 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 1.0×10^{-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 Input

Example: Input ABC

Operation: 

Example: Input SIN

Operation: 

• Numerical input

Example: Input 123

Operation: 

Example: Input 96.3

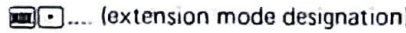


Operation: 

• 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 x 10¹⁵



Operation: 

Example: Input -2 369 x 10⁻⁴⁵

Operation: 



(2) Changing Input Contents (Correction, Deletion and Insertion)

• Correction

Move the cursor to the location to be corrected using arrow keys ( and ); then press the correct character, number or symbol.

Example: Correct "AS" to "BS"

Operation: Move the cursor two character positions to the left.


 Press the  Key.

Example: Correct "LIST" to "RUN"

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


 Press  or 

• **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 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**.

SI_I

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**

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 **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 **INS** and open up one character space.

Press **INS**.

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 **INS INS INS INS INS**.

Press **SIN**.

PRINT X

PRINT _ X

PRINT SINX

The above are methods for changing input contents.

Chapter 4

Manual Calculation

4-1 What Is Manual Calculation?

Manual calculations are not made automatically by storing calculation formulas as a program.

Instead, the 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.

4-2 Operation Method for Manual Calculation

• Addition, subtraction, multiplication and division are performed by true algebraic logic operations. **+**, **-**, **×** (x), **÷** (÷) and **=** (=) are used respectively. The **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

12+36-9*5/4

36.75

EXE

• Calculations involving functions are performed in the same manner as a normal formula. Data which may include addition, subtraction, multiplication and division operations is written in following the function command.

Example: $\log 1.23 = 0.0899051114$

Operation:

LOG 1.23

LOG 1.23

0.0899051114

EXE

• In this manual, the frames around letters and numbers will be omitted.

Example: **SIN(15) + 8** → **SIN 15 + 8**

• When storing a numerical value or a calculation result, letters A through Z of the alphabet, or a combination of letters and numbers (when used as an array), can be used as totalling variables to operate as memories.

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

Example: Store 1234 in variable A

Operation: A \equiv 1234

A = 1234

Example: Add the result of 23 x 56 to variable K.

Operation: K \equiv K \oplus 23 \times 56

K = K + 23 * 56

This manually performed method is 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 17.)

• 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 \oplus 4.5 \ominus 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 \oplus 4 \times 5 EXE

76

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

Operation: 12 \oplus 2.4 $\text{E} 5 \div 42.6 \ominus 78 \times 36.9 \text{ EXE}$

2767.602817

• Memory calculation

Example: $12 \times 45 = 540$

$12 \times 31 = 372$

$75 \div 12 = 6.25$

Operation: A \equiv 12 EXE

A \times 45 EXE

A \times 31 EXE

75 \div A EXE

-
540
372
6.25

Example $23 + 9 = 32$

$53 - 6 = 47$

$45 \times 2 = 90$

$99 \div 3 = 33$

Total 22

Operation: M \equiv 23 \oplus 9 EXE

M \ominus M \oplus 53 \ominus 6 EXE

M \ominus M \times 45 \times 2 EXE

M \ominus M \div 99 \div 3 EXE

M EXE

22

• In this calculation method the results of the respective calculations are not known, yet they are converted to M. When you want to see the calculation results, use the following method:

23 \oplus 9 EXE

32

M \ominus ANS EXE

53 \ominus 6 EXE

47

M \ominus M \oplus ANS EXE

45 \times 2 EXE

90

M \ominus M \ominus ANS EXE

99 \div 3 EXE

33

M \ominus M \oplus ANS EXE

M EXE

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 or inverse trigonometric functions, be sure to designate the angular unit.

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

Operation: $\boxed{2} \boxed{1} \boxed{3} \boxed{4} \boxed{5} \boxed{6} \rightarrow \text{"DEG"}$

SIN 12.3456 $\boxed{=}$

0.2138079201

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

Operation: $2 \boxed{\times} \boxed{SIN} \boxed{45} \boxed{\times} \boxed{COS} \boxed{65.1} \boxed{=}$

0.5954345575

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

Operation: ASN 0.5 $\boxed{=}$

30

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

Operation: $\boxed{\pi} \boxed{3} \rightarrow \text{"RAD"}$

COS $\boxed{=}$

0.5

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

Operation: ACS $\boxed{=}$ SQR 2 $\boxed{\div}$ 2 $\boxed{=}$

0.7853981634

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

Operation: $\boxed{3} \boxed{5} \rightarrow \text{"GRA"}$

TAN $\boxed{=}$

-0.612800788

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

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

Operation: LOG 1.23 $\boxed{=}$

0.0899051114

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

Operation: LN 90 $\boxed{=}$

4.49980967

Example: $e^{-3} = 0.0497870683$

Operation: EXP - 3 $\boxed{=}$

0.0497870683

Example: $10^{1.23} = 16.98243652$

(To get the antilogarithm of common logarithm 1.23)

Operation: $10 \boxed{\uparrow} \boxed{1.23} \boxed{=}$

16.98243652

Example: $5.6^{2.3} = 52.58143837$

Operation: $5.6 \boxed{\uparrow} \boxed{2.3} \boxed{=}$

52.58143837

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

Operation: $123 \boxed{\sqrt{x}} \boxed{1} \boxed{7} \boxed{=}$

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: $\boxed{2} \boxed{7} \boxed{8} \boxed{5} \boxed{6} \rightarrow \text{"DEG"}$

LOG SIN 40 $\boxed{+}$ LOG COS 35 $\boxed{=}$

-0.278567983

$10 \boxed{\uparrow} \boxed{=}$

0.5265407845

- Sexagesimal \rightarrow Decimal

Example: $14^\circ 25' 36'' = 14.42666667^\circ$

Operation: DEG $\boxed{14} \boxed{\uparrow} \boxed{25} \boxed{\uparrow} \boxed{36} \boxed{=}$

14.42666667

Example: $12.3456^\circ = 12^\circ 20' 44.16''$

Operation: DMS $\boxed{12} \boxed{\uparrow} \boxed{20} \boxed{\uparrow} \boxed{44.16} \boxed{=}$

12° 20' 44.16

Example: $\sin 63^\circ 52' 41'' = 0.897859012$

Operation: $\boxed{63} \boxed{52} \boxed{41} \rightarrow \text{"DEG designation"}$

SIN DEG $\boxed{63} \boxed{\uparrow} \boxed{52} \boxed{\uparrow} \boxed{41} \boxed{=}$

0.897859012

- Other functions (\sqrt{x} , SGN, RAN #, RND, ABS, INT, FRAC)

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

Operation: SQR 2 $\boxed{+}$ SQR 5 $\boxed{=}$

3.65028154

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

Operation: SGN 6 [EX]
 SGN 0 [EX]
 SGN 2 [EX]

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

Operation: RAN [EX]
 (Example)

Example: The result of 12.3×4.56 is rounded off at 10^{-2} .

$$12.3 \times 4.56 = 56.088$$

Operation: RND [EX] [EX] [EX] [EX]
 * For RND (x, y), y is $|y| < 100$

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

Operation: ABS [EX] [EX] [EX]

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

Operation: INT [EX] [EX] [EX]

* 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 [EX] [EX] [EX]

• **Designation of Number of Significant Digits and of Number of Decimal Places**

Designation of number of significant digits and number of decimal places is performed using a "SET" command.

Designation of number of significant digits SET E n (n = 0 through 8)

Designation of number of decimal places SET D n (n = 0 through 9)

Designation release SET N

* When the designation of the number of significant digits is "SET E 0", the number of digits is 8.

* The last designated digits will be displayed rounded off.

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

Example: $100 \div 6 = 16.66666666\text{.....}$

Operation: SET E 4 [EX] (designates 4 significant digits)
 [EX] [EX]

Example: $123 \div 7 = 17.57142857\text{.....}$

Operation: SET F 2 [EX] (designates 2 decimal places)
 [EX] [EX]

Example: $1 \div 3 = 0.333333333\text{.....}$

Operation: SET N [EX] (releases the designation)
 [EX] [EX]

4-4 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.

$$A = A(0)$$

$$B = A(1) = B(0)$$

$$C = A(2) = B(1) = C(0)$$

$$D = A(3) = B(2) = C(1) = D(0)$$

$$E = A(4) = B(3) = C(2) = D(1) = E(0)$$

$$Y = A(24) = B(23) = C(22) \text{.....} = Y(0)$$

$$Z = A(25) = B(24) = \text{.....} = Y(1) = Z(0)$$

$$A(26) = B(25) = \text{.....} = Y(2) = Z(1)$$

$$A(27) = B(26) = \text{.....} = Y(3) = Z(2)$$

Expanded memories

$$A(93) = B(92) = \text{.....} = Y(69) = Z(68)$$

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 or formula to be executed.
- ② Storing the program in the computer.
- ③ Obtaining the result automatically by simply inputting data in the program.

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

● Programs and Programming

When computer users process a problem, 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. 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.

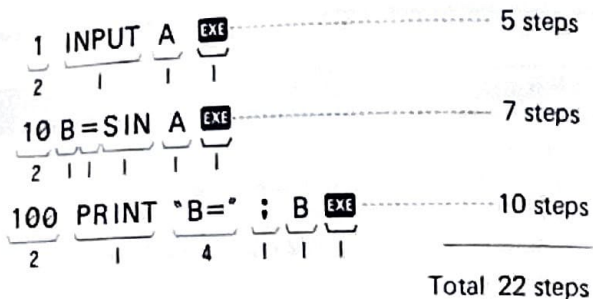
	Command	Operand	
10	INPUT	A,B	Input statement
20	C=A+B		Operation statement
30	PRINT	C	Output statement

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 enter the data. An operation statement is used to process that data. An output statement is used to retrieve the execution result. Line numbers are used at the beginning of each line. The operation statement can include judgment statements and cover many lines to make a long and complex program. Also following the line number, a line consisting of one word appears. This word is called a "command" and it tells the computer what to do next. Following this command is a character string which contains information required to process the command. This is called the "operand".

● **How to Count the Number of Steps**

A command or a function command in a program uses 1 step.
A line number (numerical values from 1 through 9999) uses 2 steps.

Example:



5-2 Program Fundamentals

5-2-1 Constants and Variables

Characters which can be used in BASIC are capital letters (A through Z) and numbers (0 through 9) and certain special characters such as symbols, etc.

● **Constants**

The characters used in BASIC are capital letters (A through Z), and numbers (0 through 9), and certain special characters such as symbols.

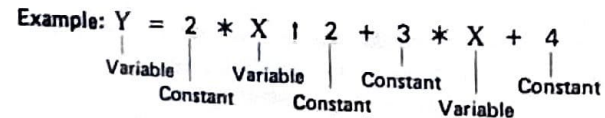
Example: $S = \pi r^2$ becomes $S = \pi * R \uparrow 2$
2 is the constant.

● **Variables**

A variable is a single capital letter (A through Z) or a single capital letter with "\$" attached (character variable).

Variables are also numerical values which are used in a program. They are used during execution to make inputs from the keyboard or to substitute calculation results which were initially unknown.

Example: $S = \pi r^2$ becomes $S = \pi * R \uparrow 2$
R is the variable.



In other words, algebraic terms are "variables" and constant numbers are "constants". In addition, there are character constants and character variables. A character constant is a character string which is written directly. A character string is a group of characters which is enclosed in quotation marks such as "123" or "ABC".

A character variable is not a numerical value — although it may consist of numbers — but a variable which contains a character string. 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.).

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

Comparison or addition of each character variable is possible. Other operations such as subtraction, multiplication and division, however, 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.

In addition to these character variables, there is also an exclusive character variable. The 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 since they use the same memory.

For example:



5-2-2 Substitution Statements

BASIC substitution statements adhere to the following format:

Variable = numerical expression

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

Example: $Y = 2 * X + 3$

In $Y = 2 * X + 3$, the left side is the variable and the right side is the numerical expression.

The "=" does not mean "equal", it means "substitute".

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 $Y \leftarrow 2 * X + 3$.

5-3 Program Writing and Execution

5-3-1 Program Writing

Storing a program in the computer memory is called "program writing". This operation is performed through key input 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 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, including the line number, can be written in. Press at the end of the line.

• The role of the Key

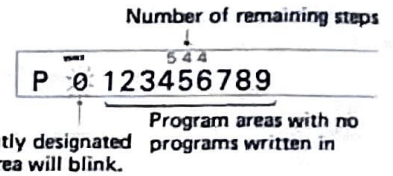
The Key is pressed for program writing, data input, and for obtaining the result of a manual calculation. The Key must also be pressed after making changes, additions or deletions to the stored program. Even when the characters on the display change, if the Key is not pressed immediately after correction, the stored contents will remain unchanged.

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.



- * 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.)

NEW



* To clear all the program areas (P0 through P9), press NEWALL .

- ④ Write line 10.

10 INPUT A,B

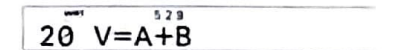
Be sure to press at the end of the line.

Means one character space (May be omitted)



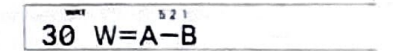
- ⑤ Write line 20.

20 V A + B



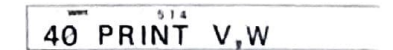
- ⑥ Write line 30.

30 W A - B



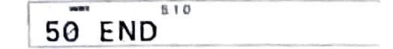
- ⑦ Write line 40.

40 PRINT V,W



- ⑧ Write line 50.

50 END



- * 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, facilitates reading the display. In BASIC language, the spacing has no special meaning (except for a PRINT statement message) and 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, use a NEW command. To clear all the programs in areas P0 through P9, use a NEW ALL command.
- The "NEW ALL" command can be abbreviated as "NEW A".

5-3-2 Program Execution

Program execution is performed in the RUN mode. (Press 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.

{ } (Press then press the desired program area.)

Example: To start the program in the previous example

Operation:

RUN mode
(omitted hereafter)

?

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

② Execution using a RUN command

RUN ("RUN" may be input by pressing either

or .)

?

- When performing Execution using a RUN command, as in the previous example, a " ? " is displayed. When the program is in an input await condition, " ? " will not be released even if is pressed. You must press and then perform operation ② to re-input data.

Also, to begin execution in the middle of the program, input the desired line number after the RUN command and press the Key.

Example: To start from line 20.

Operation: RUN 20

- 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 KEYS function. Key input using the KEYS 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. Execution will be resumed by pressing the Key after data input.

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

Operation:

- To execute the program

?

- Since 2 variables are input, first, input the value of variable A.

47

?

- Next, input the value of variable B.

69

116

-22

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

Incidentally, operations such as manual calculation can be performed during an input await condition.

Also if you want to stop program execution while in an input await condition, press .

5-4 Program Editing

- Program editing consists of changes, additions, or deletions in one or various lines, or even rearranging the order of the program to allow for logical execution.
- 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

(LIST may be input by pressing
LIST or .)

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 you do not want to call out program lines from the beginning, designate the line number where you would like to begin.

To list from line 30

Operation:

LIST 30

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

- * During LIST command execution, each program line will be displayed sequentially. If you want to stop in a particular line, press the Key. To resume the LIST command, press the Key.

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

Designate the WRT mode by pressing .

① Change

Each time the Key is pressed, one line will be displayed starting from the line which was designated using the LIST command. The previous line will be displayed by pressing .

If the line number designation is omitted, the display will automatically begin 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	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

20	V=A+B
----	-------

- Move the cursor below the "+".

20	V=A±B
----	-------

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

- Make the change.

30	W=A-B
----	-------

- * Be sure to press the Key. If it is not pressed, only the display will change but the program will remain unchanged.

- Press to release the change condition.

--	--

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

Let's list the program and check the change.

LIST

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: 1

P 123456789

• Write the new line 30.

30

30 W=V/2

• 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: 1

P 123456789

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

35

35 U=V*2

* 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
50 END
```

List the program to confirm the additions.

LIST

```
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.

1

P 123456789

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

LIST 40

40 PRINT V,W

40 PRINT V,W

• Delete "V," using the Key.

40 PRINT W,U

50 END

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

50

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

- List the program to confirm the deletion.



LIST **EXE**

```
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 **EXE**

```
P 502 123456789
```

- Input line number 30.

30 **EXE**

```
510
_
```

- Confirm the deletion.



LIST **EXE**

```
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=N12
30 L=N10.5
40 PRINT M,L
50 END
```

Move line 20 between lines 30 and 40.

Operation: 1 **EXE**

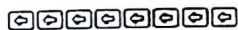
```
P 1 478 3456789
```

- Call out line 20 using a LIST command.

LIST 20 **EXE**

```
20 M=N14782_
```

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



```
20 M=N14782_
```

- Change 20 to 35 and input.

35 **EXE**

```
30 L=N14680.5_
```

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

AC

```
468
_
```

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



LIST **EXE**

```
READY P2
10 INPUT N
20 M=N12
30 L=N10.5
35 M=N12
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.

F1

20

```
P _ 1 3456789
      4 8 8
      4 7 6
```

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

F2

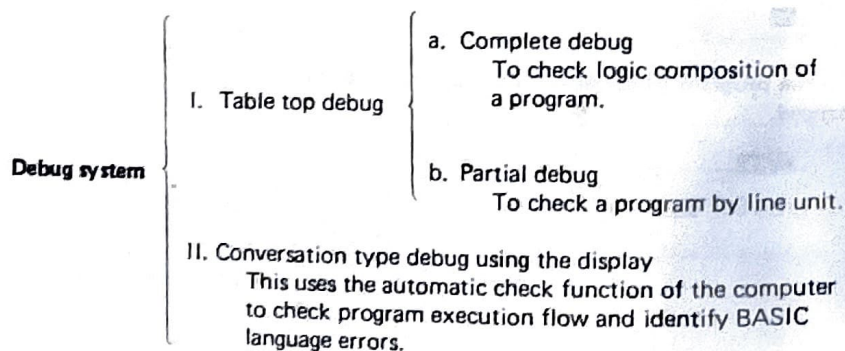
LIST

```
READY P2
10 INPUT N
30 L=N*0.5
35 M=N*2
40 PRINT M,L
50 END
READY P2
```

5-5 Program Debug

(1) Program debug system

The debug system of the PC-4 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 69.

Example:

```
10 INPUT X
20 Y=X*2+3*X+15
30 PRINT Y
40 END
```

Suppose line 20 of the above program is mistakenly input as follows.

```
20 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 P0-20

- This means that "a syntax error occurred on line 20". Confirm the program contents.

LIST 20

P _ 123456789

20 Y=X*2+3X+

- The "*" was omitted between "3" and "X" on line 20. Therefore, correct it by following the procedure for program editing.

20 Y=X*2+3_X

30 PRINT Y_

(3) Debug while executing the program

Conversation type debug is performed by obtaining information from the computer in the form of error messages. However, there may be occasions when an error message is not displayed, yet the calculation or program result is incorrect. In cases like that, program execution can be carried progressively in steps to confirm the calculation results along the way, and thereby isolate the error. There are two ways to do this: (1) the execution process is stopped using a STOP command; (2) execution is performed in one line units 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+X12
50 NEXT I
60 PRINT Y
70 END
```

The value of Y, before each consecutive loop, can be viewed using a STOP statement.

Operation:

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

F1

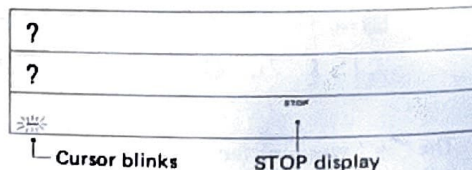
```
45 STOP 
```

- The execution process will stop after the calculation on line 40 is completed and a check can be made then.

F2 RUN

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.

Y

—	STOP
15138	STOP

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

When assembling a complicated program, checking the process using table top debug can be very difficult. However, 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 (2)), the program will sequentially stop at each line and debugging can then 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. (2)

Designate the TR mode. (2)

RUN

Check the execution process.

Continue program execution.

4

87

The value of Y Y

Repeated hereafter

READY P0	
READY ^{TR} P0	
P0-10	STOP
?	TR
?	TR
P0-20	STOP
?	TR
?	TR
—	
P0-20	
P0-30	
P0-40	
7569	
P0-45	

'TR' and 'STOP' will be omitted hereafter.

Debugging using the TR mode is ideal for checking the entire flow of a program and isolating mistakes that may 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 or program execution stops (after display shows a "?").

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

The "character string" may be omitted. However, if it is written, 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.), or the exclusive character variable (\$). These can be written consecutively using a ",".

Example:

```
INPUT A
INPUT 'DATA=' ,A
```

```
?
DATA=?
```

After an INPUT statement, a "? " will be displayed and the PC-4 enters an input await condition. At this time, if data is input and the **OK** Key is pressed, program execution will proceed to the next process.

The input await condition will not be released even if the **AC** Key is pressed. Therefore, when you want to stop a program in execution, press **STOP**.

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

In the case of INPUT A

Numerical value ----- 123 **OK** → A=123

Result of a numerical expression ----- 14 **OK** 25 **OK** → A=350

In the case of INPUT B\$

Character string ----- ABC **OK** → B\$=ABC

789 **OK** → 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 **OK** → A=X

=987654

- 44 -

• KEYS 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 = KEYS

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 KEYS function and a portion of the distribution. However, a determination will be made using the IF statements whether the character data read by the KEYS function on line 10 was input or not. Using the KEYS function, even if the **OK** Key is not pressed, only the first key input 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 judgment commands that perform distribution using character variables which were input using the KEYS function. For details concerning IF statements, refer to page 49.

- "KEYS" function can be abbreviated as "KEY".

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 will be displayed. In the case of a calculation formula, the result will be displayed.

Example:

```
PRINT A (make A = 12345)
PRINT 789
PRINT A*2 (make A = 147)
PRINT B$ (make B$ = "PC-4")
```

12345
789
294
PC-4

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

Example:

```
PRINT "ABC"
PRINT "XYZ" + "123"
```

ABC
XYZ123

The numerical expressions or character expressions may be written consecutively by using a ";" (semicolon) or "," (comma). However, the number of characters which can be written on one line cannot exceed 62, including the line number. The number of characters in the character string enclosed in quotation marks cannot exceed 30. The difference between the ";" and the "," is that, with the ";", the numerical expression or character expression will be displayed following the previous expression, with the ",", the display will go off once and then the numerical expression will appear next.

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 which designates the location where the data is to be displayed.

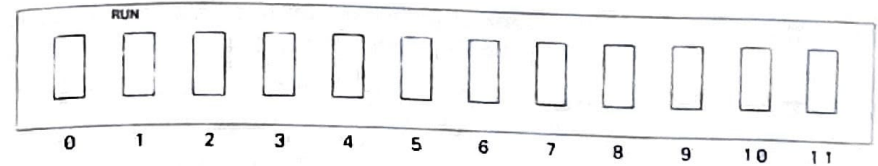
Format: PRINT CSR $\left\{ \begin{matrix} \text{numerical} \\ \text{expression}^* \end{matrix} \right\} \left\{ \begin{matrix} \text{numerical expression} \\ \text{character expression} \end{matrix} \right\} \dots$

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

Items enclosed in () can be omitted

* The value of the numerical expression can vary from 0 to 11.

Using the value of this numerical expression, a position on the display can be designated as the starting place for data to appear. 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 "," (comma) is used instead of a ";" (semicolon) following the CSR function, the display will be cleared once and then successive displays will begin from the left by using the **EXE** Key.

5-6-3 Jump Command

• **GOTO statement**

A GOTO statement, also called an "unconditional jump", causes program execution to continue at 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.

• ON - GOTO statement

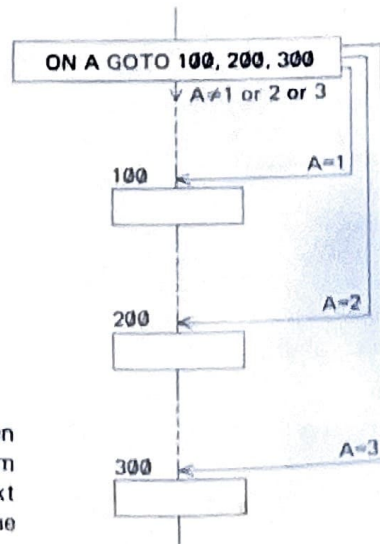
The GOTO statement designates the jump destination directly and causes a jump. However, the ON - GOTO statement determines the jump destination according to the value of the variable. It is used when the jump destination cannot be decided at the beginning of the program; such as, when the operation flow is decided by the data.

Format: { ON variable or numeric expression GOTO line number . . .
ON variable or numeric expression GOTO # program area,
program area . . . }

The value indicated by the variable (A,B,X,Y, etc.) or the numeric expression (A+B, ABS X, etc.) determines which line number or program area to jump to.

Example: ON A GOTO 100, 200, 300

In this program, when A is:
1, the program jumps to line 100
2, the program jumps to line 200
3, the program jumps to line 300



If A is something other than 1, 2 or 3, then the jump is not performed and the program proceeds to the next command or the next line number. This indirect designation is the same as "GOTO A * 100".

5-6-4 Judgment Command

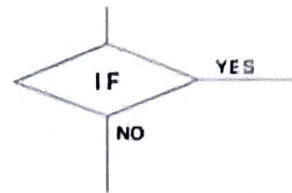
• IF statement

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 "=" (equal to) or "≠" (not equal to) signs, and if the condition is satisfied, program execution proceeds to the specified location. If the condition is not satisfied, execution proceeds to the next line.

This operation is shown in the flowchart below.



As the flowchart shows, if the IF statement condition is fulfilled, the process goes in the "YES" direction, but if the IF statement is not fulfilled, the process continues in the "NO" direction.

In other words, an IF statement indicates a branch and selects the next operation as a direct result of judgment. 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 ≥ 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.	

"THEN" or ";" (semicolon) 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 = n TO m [STEP ℓ]
          }          initial final          increment
          }          value  value
          }
          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.

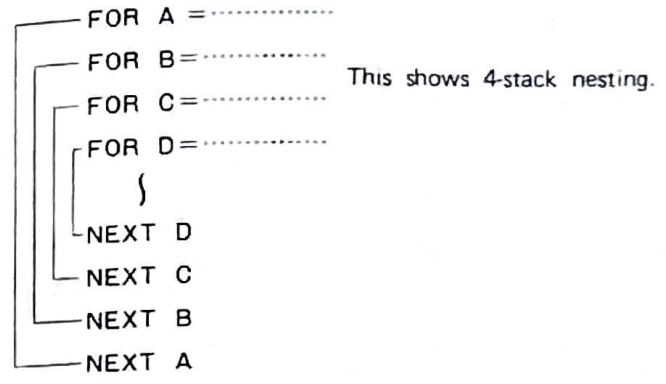
```

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

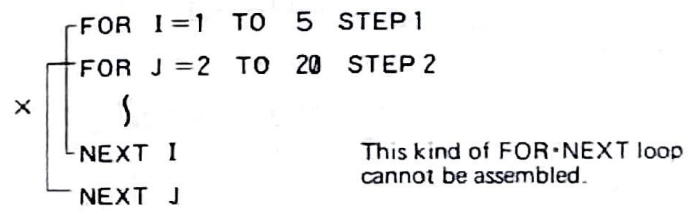
* When increase is performed in increments of 1, "STEP" may be omitted.

* Nesting

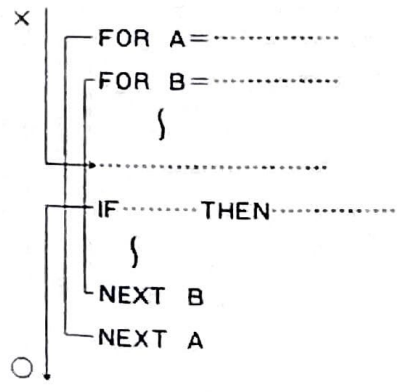
Up to 4 FOR-NEXT loops can be stacked. This stacking is called "nesting".



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

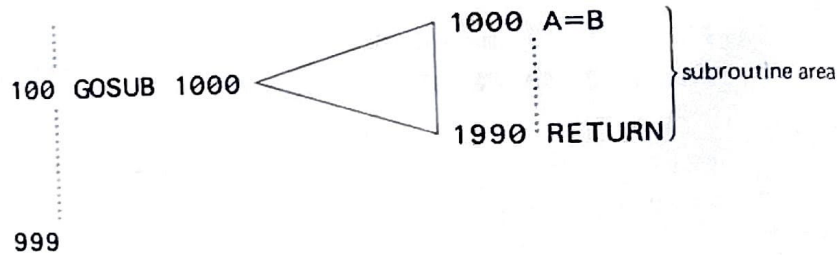
• GOSUB statement

A subroutine is also called a "subprogram". Subroutines are separate programs to be called out from a main routine. The command to call out a subroutine is a GOSUB statement. Using this command, program execution jumps from the main routine to the subroutine. After the subroutine is executed, the program returns to the original location in the main program (using the RETURN statement in the subroutine).

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

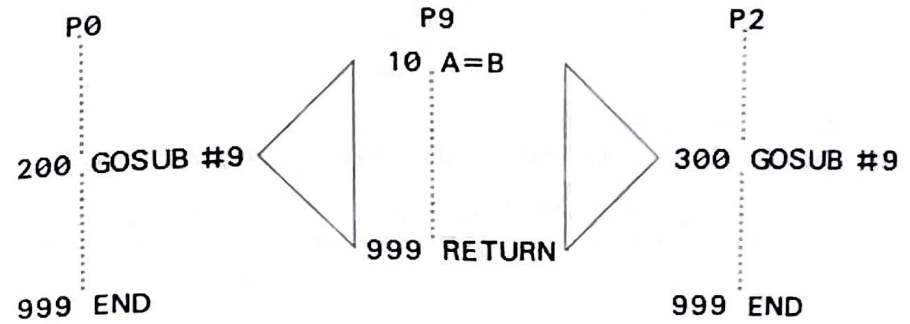
The numerical expression following the GOSUB statement indicates the initial line number of the subroutine area. Without a RETURN statement at the end of the subroutine area, program execution cannot return to the main routine.

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 subroutine.

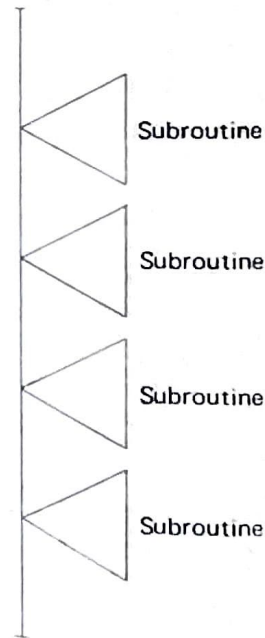
Example:



* Nesting

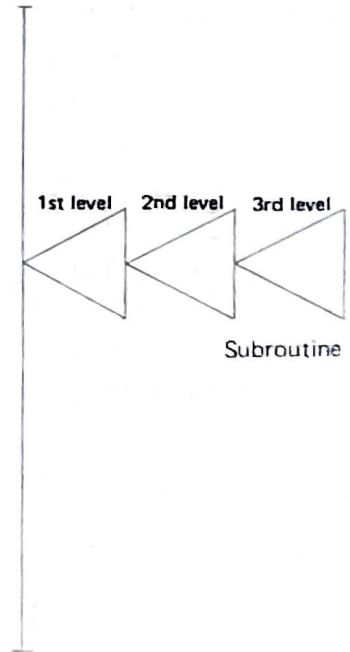
Similar to the FOR-NEXT statement, GOSUB statements can also be stacked. The number of times the subroutine is called out is fixed. This nesting can be performed up to 8 levels. Consequently, a subroutine can be called out from a subroutine.

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.

● **ON – GOSUB statement**

As with the ON – GOTO statement, indirect designation can also be used for a subroutine.

The method of using an ON – GOSUB statement is similar to that of an ON – GOTO statement. It jumps to a designated subroutine and then returns to the original position.

Example:

```
10 INPUT N
20 ON N GOSUB 100,110,120,130,
   140,150,160,170,180,190
30 PRINT X*6
40 END
100 X=5:RETURN
110 X=10:RETURN
120 X=15:RETURN
130 X=20:RETURN
140 X=25:RETURN
150 X=30:RETURN
160 X=35:RETURN
170 X=40:RETURN
180 X=45:RETURN
190 X=50:RETURN
```

In this program, the subroutine is designated indirectly using the value of input N. When N is 1 to 10, it goes to a subroutine between 100 and 190 and determines the value of variable X and displays the computation result of $X * 6$.

5-6-7 Data Processing Command

● **READ and DATA statements**

There are 2 types of data processing.

Normally you enter data through the keyboard, but you can also include data inside the program.

Use "READ" and "DATA" when the data is processed based on a certain constant.

A READ statement reads data from a DATA statement into a variable on a one-to-one basis.

Format: { READ variable, variable,
DATA data, data, }

Numeric variables, character variables, array variables, etc., can be used in a READ statement.

An error will occur when a character variable is read into a numeric variable.

Example:

```
10 READ A
20 FOR I=1 TO A
30 READ K(I)
40 PRINT K(I)
50 NEXT I
100 DATA 5,100,150,200,250,300
```

In this example, the number of data is read into variable A to specify the number of FOR loops, and the data will be read into array variables K(I) respectively by the FOR loops.

If the number of data is less than the number of variables assigned by a READ statement, an error (ERR 4) occurs.

However, if the number of data is more than the number of variables, an error does not occur; the extra data is simply ignored.

```
READ A,B,X$,Y$
DATA 10,20,TANDY,PC-4,USA
```

} An error does not occur.

```
READ M,N,U$,V$,W$
DATA 123,456,TEST,NAME
```

} An error occurs.

- **RESTORE statement**

A RESTORE statement specifies the execution sequence of DATA statements.

Format: RESTORE [line number]

A RESTORE statement has two formats – with or without a line number. If the line number is not given, the READ statement reads data from the first DATA statement when RESTORE is executed.

Example:

```
10 RESTORE 100
20 READ A
30 RESTORE 120
40 READ B
50 RESTORE
60 READ X,Y,Z
100 DATA 100
110 DATA 110
120 DATA 120
130 PRINT A,B,X,Y,Z
```

In this example, 100 is assigned to A and 120 is assigned to B, since the data on lines 100 and 120 are read into variables A and B respectively.

Since a line number is not specified in the RESTORE statement on line 50, the first DATA statement (line 100) is specified. So 100, 110 and 120 are assigned into X, Y and Z respectively.

5-6-8 Multistatement

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

Example:

```
10 A=2
20 B=10
30 C=50
10 PRINT 'NO.' :N;
20 INPUT A
10 A=2:B=10:C=50
10 PRINT 'NO.' :N::INPUT A
```

5-6-9 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 **EXIT** Key to resume program execution.

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

5-6-10 End Command

- **END statement**

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

An 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.

5-6-11 Comment Statement

- **REM statement**

The REM statement gives a comment to a program.

Format: REM comment statement

Unlike other commands, the REM command does not execute anything. Since anything can be freely written after REM, a program explanation can be written at important points in a program so you can see the content of each part of a program by looking at a list.

```
10 REM *PROCESSING RESULTS*
20 A=1
...
100 REM *PRINT OUT*
```

Since all of the characters and symbols written after REM are considered to be comments, other commands should not follow REM on a line.

5-6-12 Program Protection Command

• PASS command

(1) PASS protect

The PASS command protects a program by assigning a program password.

Format: PASS "Password with up to 8 characters" **EXE**

Characters, numerals and symbols can be used for a password with up to 8 characters. When this command is used, a program cannot be edited unless the password is cancelled. The following commands cannot be executed.

(1) LIST

(2) NEW, NEW ALL

Also, a program cannot be newly written. The functions of the PASS command are effective in all of the program areas. Conversely, the PASS command cannot function in only a certain program area. If the commands listed above are executed when the PASS command has been executed, an error (ERR 8) will be displayed. (This command cannot be used in a program.)

(2) PASS release

To release a password, make an entry using exactly the same password as follows.

Format: PASS "Password with up to 8 characters" **EXE**

If a password that was entered has been forgotten, the PASS command function cannot ever be released. Since a password cannot be seen, we suggest you use something which you'll never forget, such as your name. Also, before the password entry (before pressing **EXE**), it is important to confirm that you assigned the password exactly as you intended to.

If you forget passwords, the only solution is to remove the batteries or to press the ALL RESET button; all the programs will be lost, however.

Thus, we recommend that you store programs on cassette tape, using the SAVE command, before you assign a password.

When a password is specified, the number of steps required for the password is the number of its characters plus 1.

(3) Saving a program with a password attached

When a program with a password attached is stored on a cassette tape, the password is also stored at the same time.

So, the same password is specified when the program is recalled from a cassette tape.

A program with another password attached cannot be recalled from a cassette tape when a password is specified, and an error (ERR 8) will occur.

5-6-13 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.

5-6-14 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 ALL

When followed by a line number, the program will be displayed 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 ALL", 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)

LIST EXE	10 A=0
	20 INPUT B
	30 A=A+B
	40 GOTO 20

(WRT mode)

LIST 20 EXE	20 INPUT B_
EXE	30 A=A+B_
EXE	40 GOTO 20_

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


5-6-15 Mode Designation

• MODE command

The MODE command is used to designate the angular unit or 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  Key during manual operation.

5-6-16 Output Format

• SET command

A SET command is used to designate the display output format. It designates the number of significant digits and the number of decimal places.

Format: SET E *n* designation of number of significant digits
(*n* = 0 through 8)

SET F *n* designation of number of decimal places
(*n* = 0 through 9)

SET N designation release

- If SET E 0 is used when designating the number of significant digits, 8 positions will be designated. This command can be performed manually or by writing it in a program. Refer to page 24 for display contents.

5-6-17 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.

• MIDS

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

Format: MIDS (*m* [, *n*]) *m* and *n* are numerical expressions and must be between 1 and 100.
(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.

- "MIDS" function can be abbreviated as "MID".

Example:

If S = "PC-4"

MID (2,3) = "C-4" and MID (4) = "4"

• 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 lacking numbers in the data, "ERR 2" will be displayed, but not the program area and line number.

• STR\$

STR\$ transforms a numerical variable into a character string

Format: STR\$ (Numeric variables)

This function is convenient when a numeric value is used as a character, such as position alignment.

5-6-18 Memory Clear

• CLEAR command

The CLEAR 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, input CLEAR at the beginning of the program.

Example: Writing in the program

```
10 CLEAR
```

```
⋮
```

Manual execution

```
CLEAR [EX]
```

5-6-19 Program Clear

• NEW command

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

Format: NEW

NEW ALL

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

Example:

[EX] NEW [EX]clears a single program

[EX] NEW ALL [EX]clears all programs in all areas.

* "NEW ALL" command can be abbreviated as "NEW A".

5-6-20 Option Specifications

■ Cassette Magnetic Tape

In order to record programs or data stored in this unit on a cassette tape with the PC-4, use the PC-4 cassette interface and an ordinary tape recorder. If the tape recorder has a remote terminal, remote control can be conveniently controlled from the PC-4 through the Cassette Interface.

For tape recorder connection procedures and detailed operating procedures, refer to the PC-4 Cassette Interface operation manual.

• Program recording

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

The filename may be composed of alphabetical letters, numbers, symbols, enclosed in quotation marks and containing up to 8 characters.

Example:

```
"ABC"
```

```
"NO.1"
```

This command starts the tape recorder in the RECORD position.

Operation: SAVE ["filename"] [EX]

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"] [EX]

Display during program load

```
PF:ABC
```

Program file filename

A program that has been written in a designated area will be erased, starting with the initial line number of the program to be loaded, if a callout is performed.

• Recording All Programs

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

This command simultaneously records all 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 ALL ["filename"] [EX]

• Callout of all programs

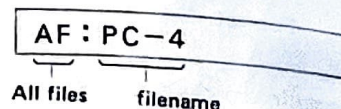
Format: LOAD ALL ["filename"] (Item in brackets may be omitted.)

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

Operation:

LOAD ALL ["filename"] **EXE**

Display during program load



Programs that are already written in program areas prior to callout will be cleared and the new programs will replace them.

The SAVE ALL command and the LOAD ALL command can be used manually.

- "SAVE ALL" and "LOAD ALL" can be abbreviated as "SAVE A" and "LOAD A" respectively.

• 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 `PC` 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 using the PUT command in a program, write it 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 either manually or 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 in along a line number attached and start the program.

• Checking the file which has been recorded on the tape

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

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

The operation sequence is similar to program load

- "VERIFY" command can be abbreviated as "VER".

■ Printer

An exclusive mini printer can be connected to the PC-4. By connecting this printer, program lists, data and even calculation results during execution can be printed out. For printer connection and operating procedures, refer to the Mini Printer Operation Manual.

To print a program list, press **EXE** **Z** and designate the PRT mode.

Program list

EXE **Z** **EXE** **Z**
 LIST **EXE** or LIST ALL **EXE**
EXE **Q** (PRT mode release)

After printout is complete, be sure to press **EXE** **Q** 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.

- "LIST ALL" can be abbreviated as "LIST A".

5-6-21 General Functions

General functions include arithmetic functions, such as trigonometric and others. These functions can be entered by pressing specific characters on the keyboard, or with the "one-key command" using a single key.

General functions may be used manually or incorporated within programs.

Function name		Format	Example
Trigonometric function	$\sin x$	SIN <i>x</i>	SIN 30 SIN A SIN (N+5)
	$\cos x$	COS <i>x</i>	COS 3.14 COS I COS (L-3)
	$\tan x$	TAN <i>x</i>	TAN 70 TAN F TAN (F X 2)
Inverse trigonometric function	$\sin^{-1} x$	ASN <i>x</i>	ASN 0.07 ASN P ASN (Z+Y)
	$\cos^{-1} x$	ACS <i>x</i>	ACS $\pi/5$ ACS D ACS (C-X)
	$\tan^{-1} x$	ATN <i>x</i>	ATN 6.5 ATN V ATN (Q+0.5)
Square root	\sqrt{x}	SQR <i>x</i>	SQR 69 SQR A SQR (R X S)
Exponential function	e^x	EXP <i>x</i>	EXP 5, EXP E, EXP (P+Q)
Natural logarithm	$\log_e x$	LN <i>x</i>	LN 43 LN P LN (U+T)
Common logarithm	$\log_{10} x$	LOG <i>x</i>	LOG 24.6 LOG R LOG (G+15)
Integration (Maximum integer not exceeding <i>x</i>)	INT <i>x</i>	INT <i>x</i>	INT 347.457 INT V INT (Q+U) INT -45.43
Fractionalization (<i>x</i> with its integer part removed)	FRAC <i>x</i>	FRAC <i>x</i>	FRAC 73.54 FRAC N FRAC (H+B)
Absolute value	$ x $	ABS <i>x</i>	ABS -9.43 ABS L ABS (K/P)
Conversion of decimal to sexagesimal		DMSS (<i>x</i>)	DMSS (12.34), DMSS (M)
Conversion of sexagesimal to decimal		DEG (<i>x, y, z</i>)	DEG (12, 34, 45), DEG (A, B, C)
Sign (If $x > 0$, 1) (If $x = 0$, 0) (If $x < 0$, -1)		SGN <i>x</i>	SGN 79 SGN E SGN (P-0)
Significant digit specification (<i>x</i> is determined down to the 10^y -th significant digit place by rounding)		RND (<i>x, y</i>)	RND (123.456, 2) RND (A, C) RND (F+E, G-5)
Random number generation (Uniform random number generation in the range $0 < x < 1$)		RAN #	RAN #
Unit of angular measure	Degree Radian Gradient	MODE 4 MODE 5 MODE 6	One right angle = 90 degrees One right angle = $\frac{\pi}{2}$ radians One right angle = 100 gradients

* *x* and *y* are constants, variables or numerical expressions.

Example 1: Make a program to obtain the length of one side of a triangle using the angle enclosed by the other two sides.

$$[C = \sqrt{a^2 + b^2 - 2ab \cos \theta}]$$

```

10 MODE 4
20 INPUT A,B,C
30 D=SQR (A2+B2-2*A*B*COS C)
40 PRINT D
50 GOTO 20

```

MODE 4 in line 10 designates the angular unit in degrees. Then, the lengths of the two sides and the enclosed angle are input in line 20. Line 30 calculates the square root SQR, and cosine COS, etc., in accordance with the formula.

Example 2: Make a program to obtain the amplifier gain dB with input voltage *X* and output voltage *Y*.

$$[\text{dB} = 20 \cdot \log_{10} \frac{Y}{X}]$$

```

10 INPUT X,Y
20 Z=20*LOG (Y/X)
30 PRINT Z
40 GOTO 10

```

Example 3: Make a program to generate three-digit random numbers using a random number generating function and a significant digit specification function.

```

10 N=RND (RAN#, -4) *1000
20 PRINT N
30 END

```

Since random numbers are generated as 10-digit numbers in the range $0 < x < 1$, three digits are taken as significant digits and multiplied by 1000.

Example 4: Make a program to perform a calculation using the exponential function,

$$\left[\text{Calculate } \frac{A + e^{1.5}}{B} \right]$$

```

10 INPUT A,B
20 C=(A+EXP1.5)/B
30 PRINT C
40 GOTO 10

```

Example 5: Make a program to make a time calculation by entering hours (H), minutes (M) and seconds (S).

```

10 CLEAR
20 INPUT H,M,S
30 T=T+DEG(H,M,S)
40 PRINT DMS$(T)
50 GOTO 20

```

In this program, hours, minutes and seconds are assigned to variables H, M and S respectively. Then they are calculated as a decimal value, converted to a sexagesimal value and displayed.

Error Message List

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

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. P.44
	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-in entry is made. P.45
Output statement	PRINT	PRINT output control function $\left\{ \begin{matrix} ; \\ \end{matrix} \right\}$ output element $\left\{ \begin{matrix} ; \\ , \\ \end{matrix} \right\} \dots$	Outputs a specified output element in a specified format. P.45
	CSR	CSR $n \left\{ \begin{matrix} ; \\ , \\ \end{matrix} \right\}$ ($0 \leq n \leq 11$)	Displays from the designated n th position. P.46
Branching	GOTO	GOTO {line number #program area}	Causes control to jump to the specified line number. P.47
	ON - GOTO	ON numerical expression GOTO {line number,.....} #program area,....	Causes jump to the specified line number or program area, depending on value of numerical expression. P.48
	IF...THEN...	IF comparison {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. P.49
	GOSUB	GOSUB {line number #program area}	Calls the subroutine with the specified 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. P.52
	ON - GOSUB	ON numerical expression GOSUB {line number,.....} #program area,....	Executes the specified subroutine, depending on the value of numerical expression. P.54
	RETURN	RETURN	Signifies the end of a subroutine; returns control to a line number next to the GOSUB statement. P.52

Classification	Command name	Format	Function
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 $\left\lceil \frac{e_2 - e_1}{e_3} \right\rceil + 1$ times between the FOR and NEXT statements. If the increment e_3 is omitted, e_3 is regarded as "1" P.50
	NEXT	NEXT v	Signifies the end of a FOR loop. If the result of v 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. P.50
Data processing	READ	READ variable, variable,...	Reads data from the DATA statement to the variable. P.55
	DATA	DATA data, data,...	Stores data to be referenced by the READ statement. P.55
	RESTORE	RESTORE (line number)	Specifies the execution sequence of DATA statement. P.56
Comment statement	REM	REM comment statement	Gives a comment to a program. P.57
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 STOP key. P.57
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 STOP key is pressed. P.57
Data clearing	CLEAR	CLEAR	Clears all variable data for a program. P.62
Program listing	LIST	LIST (line number)	Displays a listing of all the statements in a program from the specified line number downward. P.59
Program/data listing	LIST ALL	LIST ALL	Displays a listing of the statements in all programs and the data in all memories. P.59
Program execution	RUN	RUN (line number)	Causes a program to start from the specified line number. P.59

Classification	Command name	Format	Function
Program erasing	NEW	NEW	Clears the currently specified program area of a program. P.62
	NEW ALL	NEW ALL	Clears all the programs. P.62
Program protection	PASS	PASS "character string" (Up to 8 characters)	Protects a program by assigning a program password. P.58
Memory expansion	DEFM	DEFM [number of memories to be expanded]	Expands memories. P.11
Assignment statement	LET	[LET] variable = numerical expression	Assigns the result of a numerical expression to the variable.
Angular unit designation	MODE	MODE $\begin{Bmatrix} 4 \\ 5 \\ 6 \end{Bmatrix}$	Designates trigonometric angular units as degree (4), radian (5) or gradient (6). P.60
Format designation	SET	SET $\begin{Bmatrix} E \\ F \\ N \end{Bmatrix}$ ($0 \leq n \leq 9$)	Designates the number of significant digits or number of decimal places for the displayed numerical value. P.60
Character function	LEN	LEN (character variable)	Calculates the size of the character variable. P.60
	MID\$	MID\$ (m, n)	Extracts n characters from the mth character in the exclusive character variable (\$). P.61
	VAL	VAL (character variable)	Converts the numbers in a character variable to a numerical value. P.61
	STR\$	STR\$ (numerical variable)	Converts the numerical value in a numerical variable to a character. P.61
Option use	SAVE	SAVE ["filename"]	Records only the program in the currently designated program area on tape. P.63
	LOAD	LOAD ["filename"]	Calls out the program from the tape and loads it to the currently designated program area. P.63
	SAVE ALL	SAVE ALL ["filename"]	Records the programs in all program areas on tape at the same time. P.63
	LOAD ALL	LOAD ALL ["filename"]	Calls out all programs from the tape and loads them to the respective program areas. P.63
	PUT	PUT ["filename"] variable	Records the data in the variable on tape. P.64
	GET	GET ["filename"] variable	Calls out the data from the tape and loads it in the variable. P.64
	VERIFY	VERIFY ["filename"]	Checks to confirm that the programs or data have been recorded on the tape properly. P.65

* Items enclosed in [] may be omitted. Either one of the items enclosed in { } may be used.

Function Digit Capacity

$\sin x, \cos x, \tan x$
 $\sin^{-1} x, \cos^{-1} x$
 $\tan^{-1} x$
 $\log x, \ln x$
 e^x
 \sqrt{x}
 x^y ($x \uparrow y$)

Input range	Result accuracy
$ x < 1440^\circ$ (8π rad, 1600 gra)	10th digit ± 1
$ x \leq 1$	"
$x > 0$	"
$-227 \leq x \leq 230$	"
$x \geq 0$	"
When $x < 0$, y is a natural number.	"

(26-3650 or 26-3650A) program

Programs made for the original PC-4 (26-3650 or 26-3650A) can be used with the new PC-4 (26-3650B). The 26-3650B is provided with more commands than the 26-3650/A for more convenient operation.

The BASIC for 26-3650B is almost the same as for the 26-3650/A. Differences are:

• Additional commands

PASS (Program protection)
 READ (Reads data from DATA line)
 DATA (Stores data for READ)
 RESTORE (Specifies the data to read)
 ON - GOTO (Indirect specification of GOTO)
 ON - GOSUB (Indirect specification of GOSUB)
 REM (Comment line)

• Additional functions

DEG (Sexagesimal to decimal conversion)
 DMSS\$ (Decimal to sexagesimal conversion)
 STR\$ (Converts a numerical value to character variable)

• Modified commands

26-3650B	26-3650/A
NEW (NEW ALL)	CLEAR (CLEAR A)
CLEAR	VAC
SAVE ALL	SAVE A
LOAD ALL	LOAD A
VERIFY	VER
DEFM (Can be programmed)	(Manual operation only)
KEY\$	KEY
MID\$	MID

Programs made for the 26-3650/A should basically work on the 26-3650B when you change CLEAR (A) to NEW (ALL), VAC to CLEAR. However, we recommend that you revise your program using 26-3650B enhanced BASIC for easier operation (and editing).

Example:

26-3650 program

```
10 VAC
20 FOR A=1 TO 20
30 INPUT Z(A)
40 IF Z(A)>80: B=B+1:GOTO 90
50 IF Z(A)<60: C=C+1:GOTO 90
60 IF Z(A)>40: D=D+1:GOTO 90
70 IF Z(A)<40: E=E+1:GOTO 90
80 F=F+1
90 NEXT A
...
```

This is part of a program to enter data and distribute them according to their size. The program can be used as is; however, you may correct the following:

Line 10 : Change VAC to CLEAR.

Since memory expansion is necessary on this program, add line 5 stating:

```
5 DEFM 20
```

Or, on the following program:

```
10 INPUT "I=1/O=2/P=3",N
20 IF N<1 THEN 10
30 IF N>3 THEN 10
40 GOTO N*100
...
```

This portion of the program is to determine the branch line number according to the value of N. You can use ON - GOTO on the 26-3650B.

```
10 INPUT "I=1/O=2/P=3",N
20 ON N GOTO 100,200,300
30 GOTO 10
...
```

The program saved on cassette tape by the 26-3650 can be loaded to the 26-3650B. However, the program saved by the 26-3650B may not be able to be loaded to the 26-3650/A.

- When saved on tape using "SAVE ALL" by the 26-3650B, you must use "LOAD ALL" on the 26-3650/A.
- When saved on tape using "SAVE filename" by the 26-3650B, you must use "LOAD filename" on the 26-3650/A.

Specifications

- **Type**
PC-4
- **Fundamental calculation functions**
Negative numbers, exponents, parenthetical addition, subtraction, multiplication and division (with priority sequence judgment function (true algebraic logic))
- **Built-in functions**
Trigonometric/inverse trigonometric functions (angular units – degree/radian/grade), logarithmic/exponential functions, square roots, powers, conversion to integer, deletion of integer portion, absolute value, symbolization, designation of number of significant digits, designation of number of decimal digits, random numbers, π , decimal \leftrightarrow sexagesimal conversion.
- **Commands**
INPUT, PRINT, GOTO, ON-GOTO, FOR-NEXT, IF-THEN, GOSUB, ON-GOSUB, RETURN, READ, DATA, RESTORE, STOP, END, REM, LET, PASS, RUN, LIST, LIST ALL, MODE, SET, CLEAR, NEW, NEW ALL, DEFM, SAVE, SAVE ALL, LOAD, LOAD ALL, PUT, GET, VERIFY.
- **Program functions**
KEY\$, CSR, LEN, MID\$, VAL, STR\$
- **Calculation range**
 $\pm 1 \times 10^{-99}$ to $\pm 9.999999999 \times 10^{99}$ and 0 (internal calculations use 12-digit mantissa)
- **Program system**
Stored system
- **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 variables**
Standard 26, expandable to 94 (maximum 222 variables when optional RAM pack is loaded) and exclusive character variable (\$)
- **Nesting**
Subroutine – 8 levels
FOR-NEXT loop – 4 levels
Numerical value – 6 levels
Operators – 12 levels
- **Display system and contents**
10-digit mantissa (including minus sign) or 8-digit mantissa (7 digits for negative number) and 2-digit exponent.
- **Display elements**
12-digit 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 (Continuous use)**

Mainframe only – approximately 170 hours

With options connected – approximately 100 hours

■ **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 ($\frac{3}{8}$ "H x 6 $\frac{1}{2}$ "W x 2 $\frac{3}{4}$ "D)

■ **Weight**

119 g (4.2 oz) including batteries

The FCC Wants You to Know

This equipment generates and uses radio frequency energy. If not installed and used properly, that is in strict accordance with the manufacturer's instructions, it 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.

Warning

This equipment has been certified to comply with the limits for a Class B computing device, pursuant to Subpart J of Part 15 of FCC Rules. Only peripherals (computer input/output devices, terminals, printers, etc.) certified to comply with the Class B limits may be attached to this computer. Operation with non-certified peripherals is likely to result in interference to radio and TV reception.

2/86

SERVICE POLICY

Radio Shack's nationwide network of service facilities provides quick, convenient, and reliable repair services for all of its computer products. In most instances, Warranty service will be performed in accordance with Radio Shack's Limited Warranty. Non-warranty service will be provided at reasonable parts and labor costs.

Because of the sensitivity of computer equipment, and the problems which can result from improper servicing, the following limitations also apply to the services offered by Radio Shack:

1. If any of the warranty seals on any Radio Shack computer products are broken, Radio Shack reserves the right to refuse to service the equipment or to void any remaining warranty on the equipment.
2. If any Radio Shack computer equipment has been modified so that it is not within manufacturer's specifications, including, but not limited to, the installation of any non-Radio Shack parts, components, or replacement boards, then Radio Shack reserves the right to refuse to service the equipment, void any remaining warranty, remove and replace any non-Radio Shack part found in the equipment, and perform whatever modifications are necessary to return the equipment to original factory manufacturer's specifications.
3. The cost for the labor and parts required to return the Radio Shack computer equipment to original manufacturer's specifications will be charged to the customer in addition to the normal repair charge.