

SHARP

POCKET COMPUTER

PC-1401
MODEL PC-1402

INSTRUCTION MANUAL

Scanned by Dale

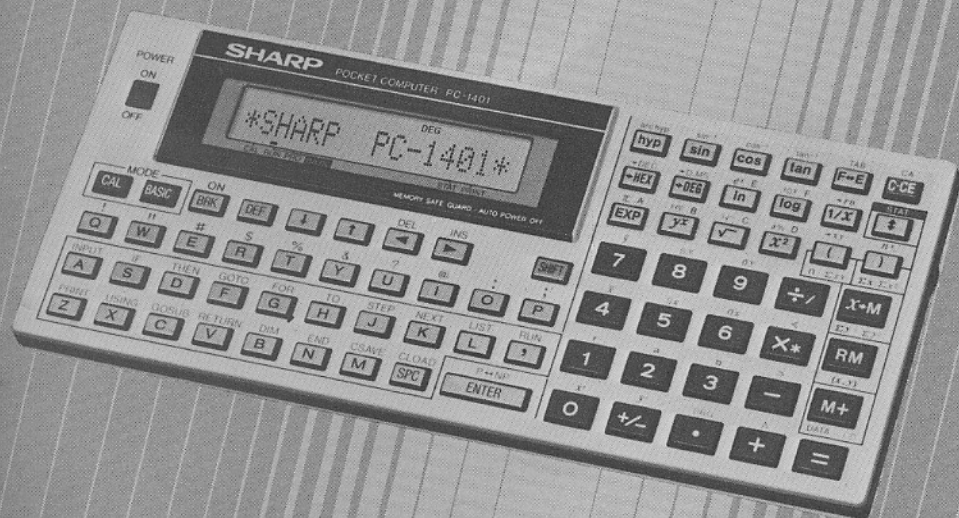


TABLE OF CONTENTS

	Page
INTRODUCTORY NOTE	4
CHAPTER 1. HOW TO USE THIS MANUAL.	5
CHAPTER 2. INTRODUCTION TO THE PC-1401/1402	7
Description of System	7
Key and Switch Operations	8
Modes	9
ALL RESET button	12
Cell Replacement	14
CHAPTER 3. USING THE PC-1401/1402 AS A CALCULATOR	17
Start Up	17
Shut Down	17
Auto Off	17
Calculation in the CAL Mode	18
How to Read the Display	22
Basic Calculations	24
Scientific Calculations	26
Conversion between Decimal and Hex Numbers, and Hex Calculation	34
Statistical Calculation	38
1. One-variable Statistical Calculation	41
2. Two-variable Statistics and Linear Regression	42
Calculation Range	45
Manual Calculation in the BASIC Mode	48
How to Manually Calculate	48
Recalling Entries	49
Errors	53
Serial Calculations	54
Negative Numbers	55
Compound Calculations and Parentheses	56
Using Variables in Calculations	56
Chained Calculations	58
Scientific Notation	59
Limits	60
Last Answer Feature	60
Scientific Calculations (in the BASIC mode)	62
Direct Calculation Feature	68
Priority (in Manual Calculation)	69

Table of Contents

	Page
CHAPTER 4. CONCEPTS AND TERMS OF BASIC	70
String Constants	70
Variables	71
Fixed Variables	72
Simple Variables	73
Array Variables	74
Variables in the Form of A ()	77
Expressions	79
Numeric Operators	79
String Expressions	79
Relational Expressions	80
Logical Expressions	81
Parentheses and Operator Precedence	83
RUN mode	83
Functions	83
CHAPTER 5. PROGRAMMING THE PC-1401/1402.	85
Programs	85
BASIC Statements	85
Line Numbers	85
BASIC Verbs	85
BASIC Commands	86
Modes	86
Beginning to Program on the PC-1401/1402.	87
Example 1 — Entering and Running a Program	87
Example 2 — Editing a Program	88
Example 3 — Using Variables in Programming	90
Example 4 — More Complex Programming	92
Storing Programs in the PC-1401/1402's Memory	93
CHAPTER 6. SHORTCUTS.	94
The DEF Key and Labelled Programs	94
Template	95
CHAPTER 7. USING THE CE-126P PRINTER/CASSETTE INTERFACE	96
Using the Printer	96
Using the Cassette Interface	98
Connecting the CE-126P to a Tape Recorder	98
Cassette Tape Recorder	98
Operating the Cassette Interface and Recorder	100
Tape Notes	104

	Page
CHAPTER 8. BASIC REFERENCE	105
Commands	108
Verbs	118
Functions	162
Pseudovariables	162
Numeric Functions	163
String Functions	171
CHAPTER 9. TROUBLESHOOTING	173
Machine Operation	173
BASIC Debugging	174
CHAPTER 10. MAINTENANCE OF THE PC-1401/1402	175
APPENDICES	
Appendix A: Error Messages	176
Appendix B: ASCII Character Code Chart	178
Appendix C: Formatting Output	180
Appendix D: Expression Evaluation and Operator Priority	185
Appendix E: Key Functions in BASIC Mode	187
Appendix F: Specifications	192
Appendix G: Feature Comparison of the PC-1211, PC-1245, PC-1251, PC-1401/1402, and PC-1500	194
Appendix H: Using Programs Developed for the PC-1245 or PC-1251	199
PROGRAM EXAMPLES	203
INDEX	250

INTRODUCTORY NOTE

Welcome to the world of **SHARP** owners!

Few industries in the world today can match the rapid growth and technological advances being made in the field of personal computing. Computers which just a short time ago would have filled a huge room, required a Ph.D. to program, and cost thousands of dollars, now fit in the palm of your hand, are easily programmed, and cost so little that they are within the reach of nearly everyone.

Your new **SHARP PC-1401/1402** was designed to bring you all of the latest state of the art features of this computing revolution and it incorporates many advanced capabilities:

- * **SCIENTIFIC CALCULATOR** — It has been normal to use two different tasks, scientific calculation (including statistics) and computing, before the **PC-1401/1402**. But now only one tool is enough. The **PC-1401/1402** operates both as a scientific calculator and a pocket computer incorporating 59 programmed scientific functions plus BASIC command keys for simple programming.
- * **MEMORY SAFE GUARD** — the **PC-1401/1402** remembers stored programs and variables even when you turn it off.
- * Battery powered operation for true portability.
- * **AUTO POWER OFF** function which conserves the batteries by turning the power off if no activity takes place within a specified time limit.
- * An expanded version of BASIC which provides formatted output, two-dimensional arrays, variable length strings, and many other advanced features.
- * An optional printer/cassette interface (Model CE-126P) with the printer, you can have "hard-copies" of programs and data. The cassette interface lets you connect a cassette recorder to store programs and data to cassette recorder.

Congratulations on entering an exciting and enjoyable new world. We are sure that you will find this purchase one of the wisest you have ever made. The **SHARP PC-1401/1402** is a powerful tool, designed to meet your specific mathematical, scientific, engineering, business and personal computing needs. With the **SHARP PC-1401/1402** you can begin **NOW** providing the solutions you'll need tomorrow!

One of the models described in this manual may not be available in some countries.

CHAPTER 1

HOW TO USE THIS MANUAL

This manual is designed to introduce you to the capabilities and features of your **PC-1401/1402** and to serve as a valuable reference tool. Whether you are a "first time user" or an "old hand" with computers, you should acquaint yourself with the **PC-1401/1402** by reading and working through Chapters 2 through 6.

- * Chapter 2 describes the physical features of the **PC-1401/1402**.
- * Chapter 3 demonstrates the use of the **PC-1401/1402** as a scientific calculator.
- * Chapter 4 defines some terms and concepts which are essential for BASIC programming, and tells you about the special considerations of these concepts on the **PC-1401/1402**.
- * Chapter 5 introduces you to BASIC programming on the **PC-1401/1402**, showing you how to enter, correct, and run programs.
- * Chapter 6 discusses some shortcuts that make using your new computer easier and more enjoyable.

Experienced BASIC programmers may then read through Chapter 8 to learn the specific features of BASIC as implemented on the **PC-1401/1402**. Since every dialect of BASIC is somewhat different, read through this material at least once before starting serious programming.

Chapter 8 is a reference section covering all the verbs, commands, and functions of BASIC arranged in convenient alphabetical groupings.

If you have never programmed in BASIC before, we suggest that you buy a separate book on beginning BASIC programming or attend a BASIC class, before trying to work through these chapters. This manual is not intended to teach you how to program.

The remainder of the manual consists of:

- * Chapter 7 — Basic information on the optional CE-126P Printer/Cassette Interface.
- * Chapter 9 — A troubleshooting guide to help you solve some operating and programming problems.
- * Chapter 10 — The care and maintenance of your new computer.

Detailed Appendices provide you with useful charts, comparisons, and special discussions concerning the use and operation of the **PC-1401/1402**.

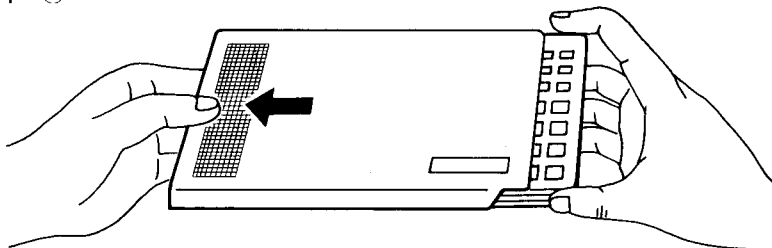
Using the Hard Cover

When the computer is not being used, mount the hard cover on the operation panel of the computer.

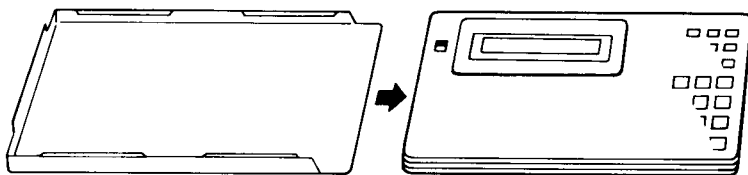
- When the computer is to be used.

To remove the hard cover from the computer, remove it as shown in figure below.

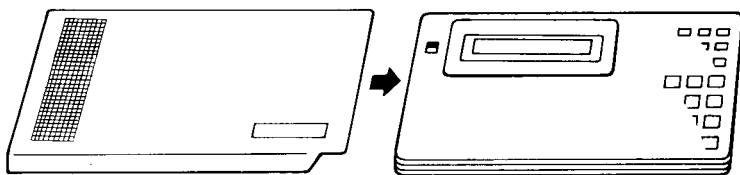
Step ①



Step ②



- When the computer is not used.



CHAPTER 1

HOW TO USE THIS MANUAL

This manual is designed to introduce you to the capabilities and features of your **PC-1401/1402** and to serve as a valuable reference tool. Whether you are a "first time user" or an "old hand" with computers, you should acquaint yourself with the **PC-1401/1402** by reading and working through Chapters 2 through 6.

- * Chapter 2 describes the physical features of the **PC-1401/1402**.
- * Chapter 3 demonstrates the use of the **PC-1401/1402** as a scientific calculator.
- * Chapter 4 defines some terms and concepts which are essential for BASIC programming, and tells you about the special considerations of these concepts on the **PC-1401/1402**.
- * Chapter 5 introduces you to BASIC programming on the **PC-1401/1402**, showing you how to enter, correct, and run programs.
- * Chapter 6 discusses some shortcuts that make using your new computer easier and more enjoyable.

Experienced BASIC programmers may then read through Chapter 8 to learn the specific features of BASIC as implemented on the **PC-1401/1402**. Since every dialect of BASIC is somewhat different, read through this material at least once before starting serious programming.

Chapter 8 is a reference section covering all the verbs, commands, and functions of BASIC arranged in convenient alphabetical groupings.

If you have never programmed in BASIC before, we suggest that you buy a separate book on beginning BASIC programming or attend a BASIC class, before trying to work through these chapters. This manual is not intended to teach you how to program.

The remainder of the manual consists of:

- * Chapter 7 — Basic information on the optional CE-126P Printer/Cassette Interface.
- * Chapter 9 — A troubleshooting guide to help you solve some operating and programming problems.
- * Chapter 10 — The care and maintenance of your new computer.

Detailed Appendices provide you with useful charts, comparisons, and special discussions concerning the use and operation of the **PC-1401/1402**.

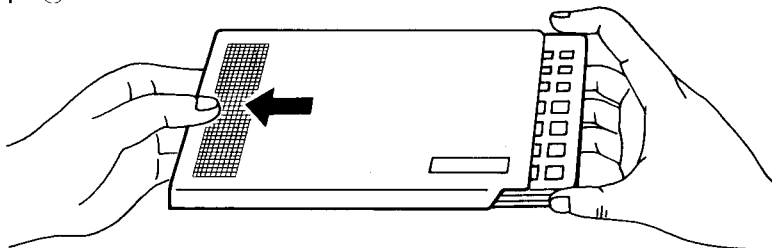
Using the Hard Cover

When the computer is not being used, mount the hard cover on the operation panel of the computer.

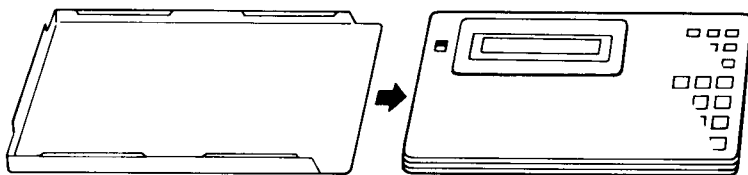
- When the computer is to be used.

To remove the hard cover from the computer, remove it as shown in figure below.

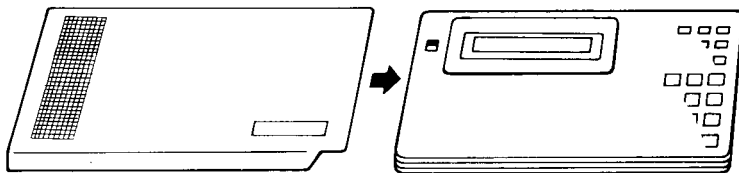
Step ①



Step ②



- When the computer is not used.



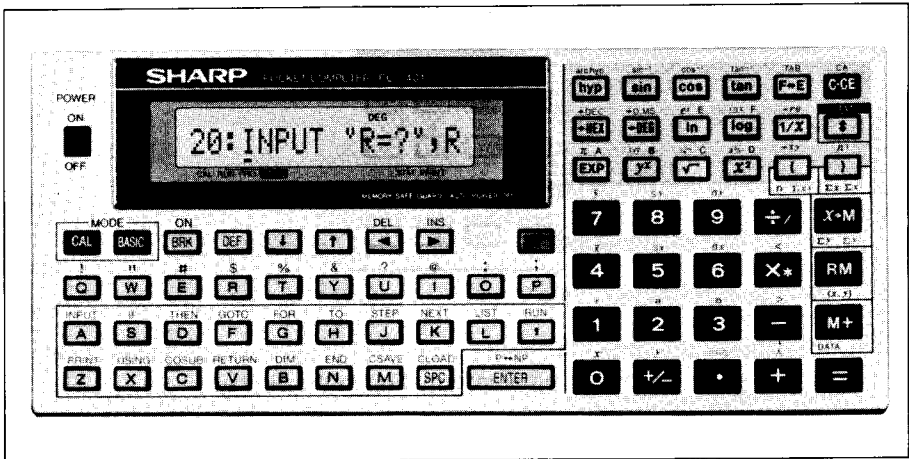
CHAPTER 2

INTRODUCTION TO THE PC-1401/1402

Description of System

The **SHARP PC-1401/1402** system consists of:

- * 76-character keyboard.
- * 16-character display.
- * Powerful BASIC in 40 K Bytes ROM.
- * 8-bit CMOS processor.
- * 4.2K Bytes RAM (**PC-1401**), 10.2K Bytes RAM (**PC-1402**)
- * Option: CE-126P Printer/Cassette Interface



To familiarize you with the placement and functions of parts of the **PC-1401/1402** keyboard, we will now study each section of the keyboard. For now just locate the keys and read the description of each. In Chapter 3 we will begin using your new machine.

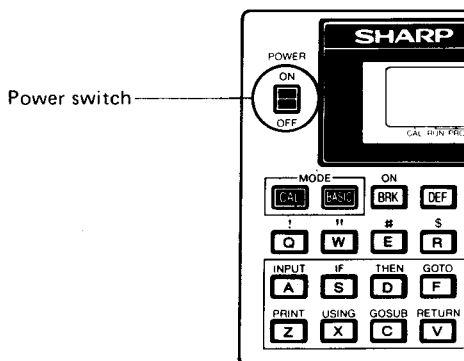
Key and Switch Operations

The **PC-1401/1402** has 76 keys and one slide switch on its panel. Each key function is identified by various characters, numbers or symbols labeled on or beside the keys.

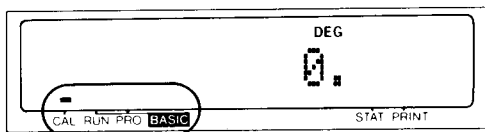
(1) Power on

To begin with, turn your computer on.

Set the **POWER** switch in the upper left corner of the computer to the **ON** position.



You will see the following initial information in the display:



A dash (=) indicator in the lower left area of the display shows the mode which the computer is now set. When this computer has just been turned on, it functions as a calculator. To show that the computer is set at the calculator mode, a dash indicator appears above the **CAL** (CALculator) label.

For calculator operation at the **CAL** mode, refer to **CHAPTER 3, USING THE PC-1401/1402 AS A CALCULATOR**.

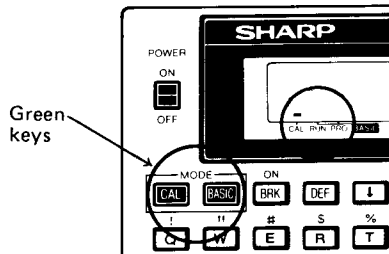
Modes

The **PC-1401/1402** can operate basically in the three different modes. One mode is the **CAL** mode, in which you can use your computer just like a calculator.

Another mode is the **RUN** mode, in which you can execute your program or manual calculation using **BASIC** commands.

The third mode is the **PRO** mode, which allows you to store your program into the computer or correct or amend a stored program.

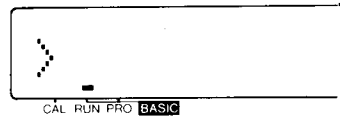
Switching between these modes can be accomplished by the green **CAL** and **BASIC** keys. The selected mode is identified with a dash (—) indicator displayed just above the **CAL**, **RUN**, or **PRO** label in the lower left area of the display.



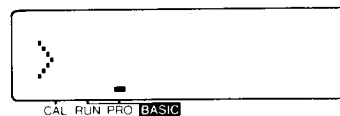
Now switch your computer off, then switch it on again. The **CAL** mode will be selected.



If you press the **BASIC** key when in the **CAL** mode, the **RUN** mode will be selected.



If you press the **BASIC** key when in the **RUN** mode, the **PRO** mode will be selected.

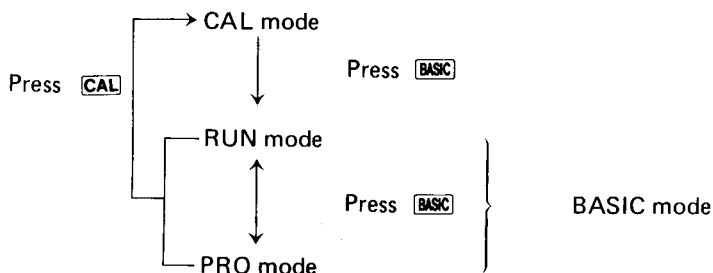


Introduction

Thus the RUN and PRO modes are selected alternately each time you press the **BASIC** key.

The computer will return to the CAL mode if you press the **CAL** key.

Mode switching



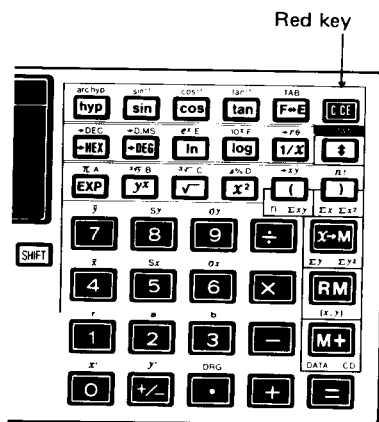
1. CAL mode

Now let's operate the keys.

Set the mode to CAL mode first.

In CAL mode the keys at right can be used for calculation.

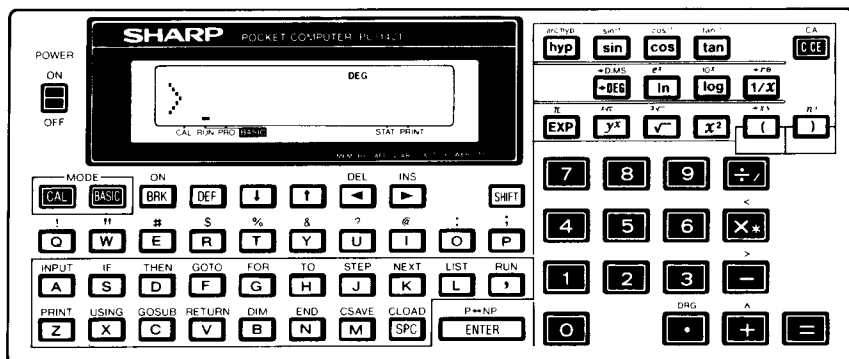
	Display
C-CE (Red key)	→ 0.
1 2	→ 12.
+	→ 12.
3	→ 3.
=	→ 15.



2. RUN and PRO modes

Change the mode to RUN or PRO mode by using the **BASIC** key, and press the following keys while watching the display:

In RUN and PRO modes the keys shown below can be used for calculations.



Example:

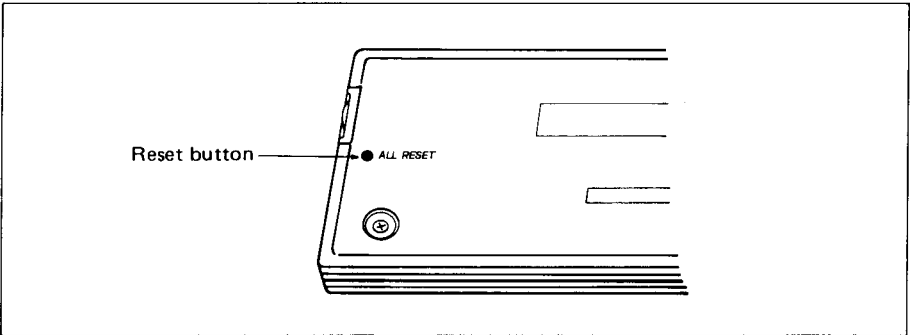
PRINT **Z** **USING** **X** **GOSUB** **C** → ZXC _
r **a** **DRG** **b** → ZXC12.3_
1 **2** **•** **3**
CA → >
C-CE
INPUT **A** **=** **x̄** **4** **+** **Sx** **5** → A = 4 + 5 _
↑
Cursor

If you press an alphabet or number key, the item denoted on the key will be entered. When you wish to enter the character or symbol denoted in brown above each key, press the yellow **SHIFT** key before operating the key.

CA **C-CE** **SHIFT** **PRINT** **Z** → PRINT _
SHIFT **W** **(** **√** → PRINT " (√ _

The **SHIFT** key is used to enter the characters or symbols labeled in brown above each key that has two or three functions. If you repeatedly press the **SHIFT** key, the SHIFT symbol in the top of the display will go on and off. The SHIFT symbol indicates that the **SHIFT** key is activated and the characters labeled in brown can be entered.

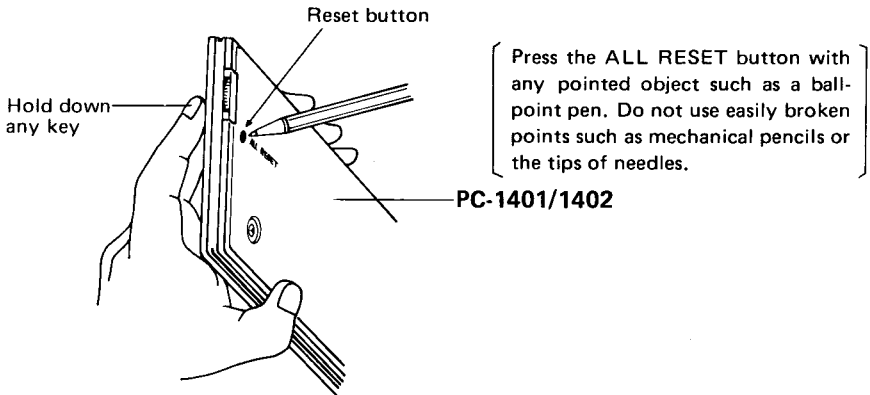
ALL RESET



Reset button. This button is used to reset the computer when C·CE or CA is not sufficient to correct the problem.

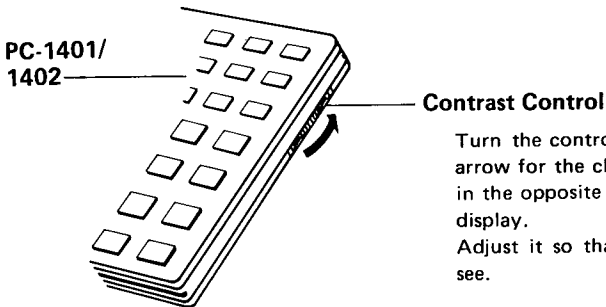
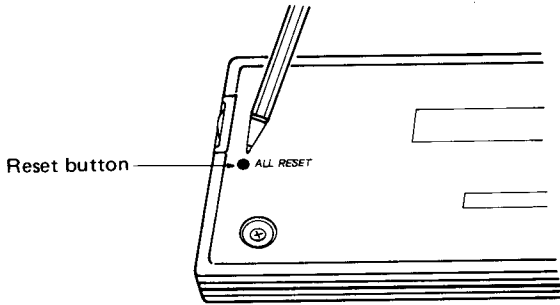
NOTE

- ① To reset the **PC-1401/1402** hold down any key on the keyboard and simultaneously press the **RESET** button on the back. This preserves all programs and variables.



- ② If you get no response from any key even when the above operation is performed, push RESET button without any key.

This operation will clear the programs and data, so do not press the RESET button without any key except when necessary.



Turn the control in the direction of the arrow for the clearer display, and turn it in the opposite direction for the dimmer display.

Adjust it so that the display is easy to see.

CELL REPLACEMENT

The **PC-1401/1402** operates on the lithium cell alone.

When replacing the cells these cautionary instructions will eliminate many problems:

- Always replace both of the cells at the same time.
- Do not mix a new cell with a used cell.
- Use only: Lithium cell (type CR-2032), two required

INSTALLING THE CELLS

The display is dim and uneasy to see when viewed from the front even by turning the contrast control on the right of the computer counterclockwise as far as it goes. This indicates that the cell power is consumed. In this case, replace the cell promptly.

- (1) Turn off the computer by setting the power slide switch to the OFF position.
- (2) Remove the screws from the back cover with a small screw driver. (Fig. 1)

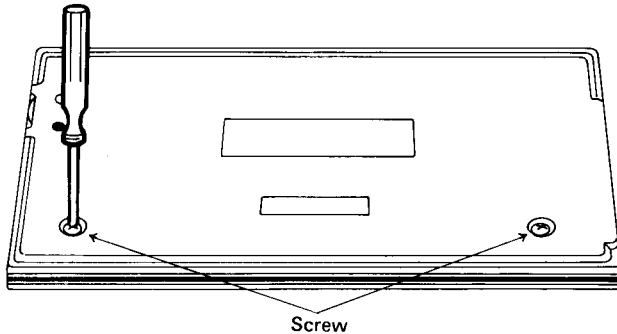


Fig. 1

- (3) Remove the cell cover by sliding it in the direction of the arrow shown in Figure 2.

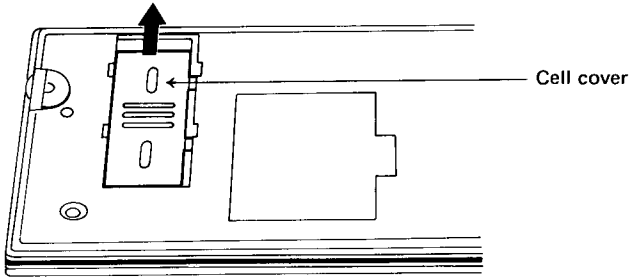


Fig. 2

- (4) Replace the two cells. (Fig. 3)

Always replace both of the cells at the same time.

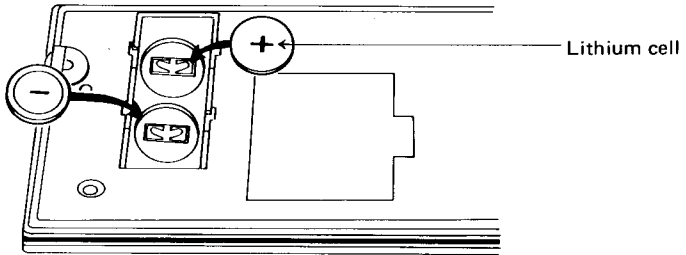


Fig. 3

- (5) Replace the cell cover by sliding it in the reverse direction of the arrow shown in Figure 2.
- (6) Hook the claws of the back cover into the slits of the computer proper. (Fig. 4)

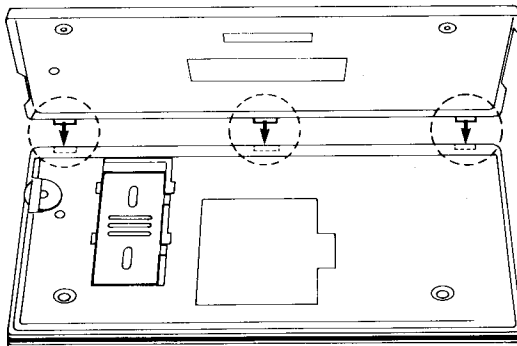
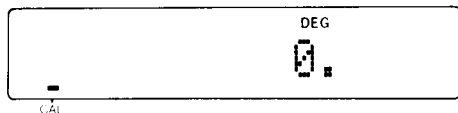


Fig. 4

Introduction

- (7) Push the back cover in slightly while replacing the screws.
- (8) Turn on the computer by setting the power slide switch to the ON position and press the RESET button to clear the computer.
The display should look like this:



If the display is blank or displays any other symbol than “ = 0.”, remove the cells and install them again, then check the display.

NOTE:

Keeping a dead cell in the computer may result in damage to the computer from solvent leakage of the cell. Remove a dead cell promptly.

CAUTION: Keep cell out of reach of children.

CHAPTER 3 USING THE PC-1401/1402 AS A CALCULATOR

Now that you are familiar with the layout and components of the **SHARP PC-1401/1402**, we will begin investigating the exciting capabilities of your new computer.

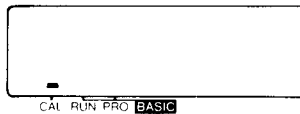
Because the **PC-1401/1402** allows you the full range of calculating functions, plus the increased power of BASIC programming abilities (useful in more complex calculations), it is commonly referred to as a "smart" calculator. That, of course, makes you a "smart" user!

(Before using the **PC-1401/1402**, be sure that the batteries are correctly installed.)

Start Up

To turn **ON** the **PC-1401/1402**, slide the power switch up.

When you wish to use your **PC-1401/1402** as a scientific calculator, place the computer in the **CAL** mode. The **CAL** mode may be selected when the computer is switched on or the **[CAL]** key is pressed. Once the **CAL** mode has been selected, a dash (**-**) indicator will appear just above the **CAL** label in the lower left area of the display.



If the dash (**-**) indicator is at the **RUN** or **PRO** label, press the **[CAL]** key to select the **CAL** mode.

Shut Down

To turn **OFF** the **PC-1401/1402**, slide the power switch to the **OFF** position.

When you turn **OFF** the machine, you clear (erase) the display.

Auto Off

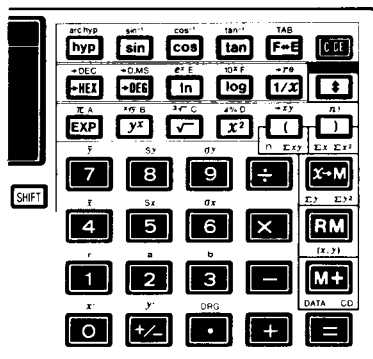
In order to conserve on battery wear, the **PC-1401/1402** automatically powers down when no keys have been pressed for about 10 minutes. (Note: The **PC-1401/1402** will not **AUTO OFF** while you are executing a program.)

To restart the computer after a **AUTO OFF** press the **ON [BRK]** key located at right hand side of the green keys.

Please note that the **CAL** mode may be selected when the **ON [BRK]** key is pressed.

Calculations in the CAL Mode

In the CAL mode the keys and functions shown at right can be used for calculation.



Now let us try some simple calculations. Press the following keys while watching the display:

Input

r	a	b
1	2	3

+

σx	Sx	\bar{x}
6	5	4

=



Press the equals key like a normal calculator.

Display

123.

123.

654.

777.

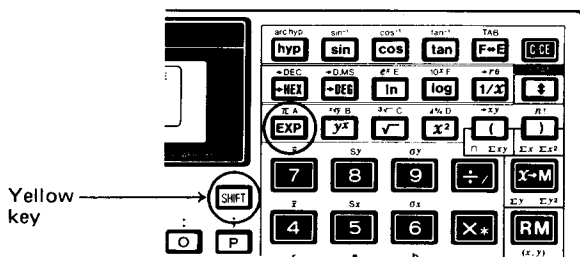
(123 + 654 = 777)

Did you get the correct answer? If you didn't, turn the computer off, then turn it on again and try the same calculation.

Now let us call the value of pi (π).

Symbol " π " is labeled just above the **EXP** key in brown. The functions identified by brown letters can be used by first pressing the yellow **SHIFT** key, and then pressing the required function key.

Now press **SHIFT** **EXP**.

Input

SHIFT π A
EXP

Display

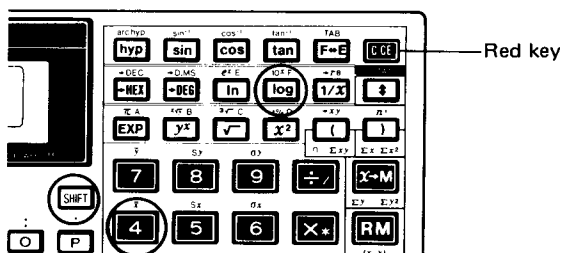
3.141592654

($\pi \approx 3.141592654$)

What you see in the display is the value of π .

Next, let us compute 10^4 . For this calculation, you should use the function 10^x .

This function is also identified by a brown letter, so the SHIFT key must be pressed before the function key is pressed:

Input

\bar{x} 4 SHIFT 10^x F
log

Display

10000.

($10^4 = 10000$)

The following outlines the major key functions:

* C-CE (clear) (red key)

If this key is pressed immediately after numeric data is entered or the contents of the memory is recalled, that data will be cleared. In any other case, operation of the C-CE key will clear the operators and/or numeric data which have been entered. The contents of the memory is not cleared with the C-CE key operation.

Using as a Calculator

Input	Display	Input	Display
123 $+$ 456	456.	6 \times 2 $+$	12.
C-CE	0.	C-CE	0.
789 $=$	912.	6 \div 2 $+$	3.
(123 + 789 = 912)		5 $=$	8.

The C-CE key may also be used to clear an error.

Input	Display
5 \div 0 $=$	0. E ← Error symbol
C-CE	0.

* $\text{F}\leftrightarrow\text{E}$ (display mode switch)

This key is used to switch the display mode from fixed point (normal mode) to floating point mode (exponent mode) or vice versa.

Input	Display
23 \times 1000 $=$	23000.
$\text{F}\leftrightarrow\text{E}$	2.3 E 04
$\text{F}\leftrightarrow\text{E}$	23000.

* TAB (specifies the number of decimal places)

This key is used to specify the number of decimal places when used in conjunction with the numeric data key.

	Input	Display
(1) Specifies 2 decimal places.	C-CE SHIFT TAB $\text{F}\leftrightarrow\text{E}$ 2	0.00 (1)
	5 \div 8 $=$	0.63

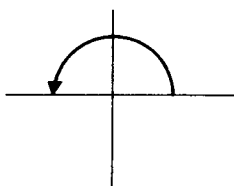
	Input	Display
(2) Specifies 5 decimal places.	SHIFT TAB F↔E 5	0.62500
		(2)
(3) Clear TAB data.	SHIFT TAB F↔E .	0.625
		(3)

* **DRG** (specifies angular data.)

This key is used to specify the angular units for data used in trigonometric functions, inverse trigonometric functions, or coordinate conversion.

	Input	Display	
		DEG	(Degree)
SHIFT DRG .		RAD	(Radian)
SHIFT DRG .		GRAD	(Grad)
SHIFT DRG .		DEG	(Degree)

$$180^{\circ} = \pi \text{ (rad)} = 200^g$$



DEG: Degree [°]
 RAD: Radian [rad]
 GRAD: Grad [g]

* **0** to **9** , **.** , **EXP** and **±/−**

EXP : Used to enter floating decimal point data (the display shows "E" for floating decimal point data).

Using as a Calculator

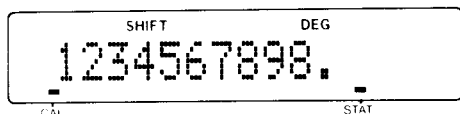
Input	Display
4 $\frac{\pi}{A}$ $\frac{EXP}{EXP}$ 3	4 . E 03 (4 × 10 ³)
=	4000.
+/-	-4000.

+/- : Used to enter negative numeric data (or to reverse the sign from negative to positive).

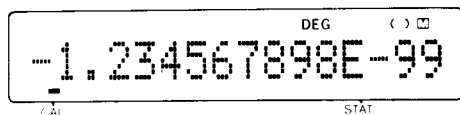
Input	Display
1.23 +/-	-1.23
$\frac{EXP}{EXP}$ 5 +/-	-1.23E-05 (-1.23 × 10 ⁻⁵)
=	-0.0000123
+/-	0.0000123

How to Read the Display

This section describes the display formats and symbols used in the CAL mode.



Fixed decimal point display format
(normal format)



Floating decimal point display
format

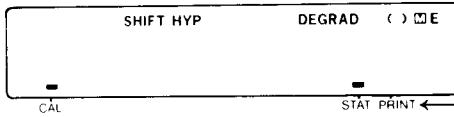
Mantissa (12 digits)

Exponent (4 digits)

The **PC-1401/1402's** display has 16 digits. In the CAL mode, calculation results are normally displayed in fixed decimal point mode. If the result is smaller than 0.000000001 or greater than 9999999999 (greater than -0.000000001 or smaller than -9999999999), it is displayed in floating decimal point mode. In floating decimal point mode, the mantissa is displayed to 12 significant digits, while the exponent is displayed to 4 significant digits. (including a decimal point, sign, and symbol)

Display symbols

The following describes the symbols and status marks shown in the display.



Refer to "Using the Printer" in the CHAPTER 7, USING THE CE-126P PRINTER/CASSETTE INTERFACE.

The CAL mode uses the symbols and marks shown above, whose meanings are described in the following:

SHIFT: This word comes on when the **SHIFT** key is activated, indicating that the key functions identified by brown labels can be selected.

HYP: This word comes on when the **hyp** key is pressed, indicating that a hyperbolic function has been selected. If **SHIFT** **hyp** are pressed, a phrase, SHIFT HYP, come on to indicate that an inverse hyperbolic function has been selected.

DEG
RAD :
GRAD These words are selected sequentially each time **SHIFT** **DRG** keys are operated. Each of these words indicates the angular units for trigonometric functions, inverse trigonometric functions, and coordinate conversion, respectively.

DEG: degree [$^{\circ}$]

RAD: radian [rad]

GRAD: grad [g]

($180 \text{ deg.} = \pi \text{ rad} = 200^g$)

(): This symbol comes on when parentheses are used in a calculation formula by means of the **()** key.

M : This symbol comes on when numeric data other than zero is stored in the calculation memory, to indicate that the memory is occupied.

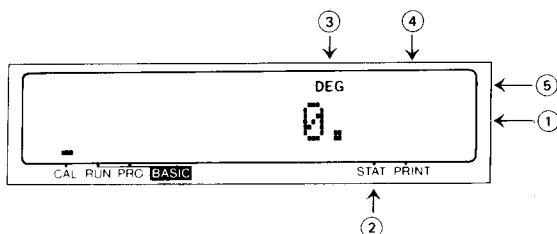
E : This symbol comes on if an error has occurred. The error can be cleared by operating the **C-CE** key.

STAT : Operation of the **SHIFT** **STAT** keys activates a dash (-) indicator just above the STAT label in the lower right area of the display. The STAT stands for statistics, and indicates that the computer is in the statistical calculation mode.

CAL : If a dash (-) indicator is displayed just above the CAL label, it indicates that the computer is in the CAL mode.

Basic Calculations

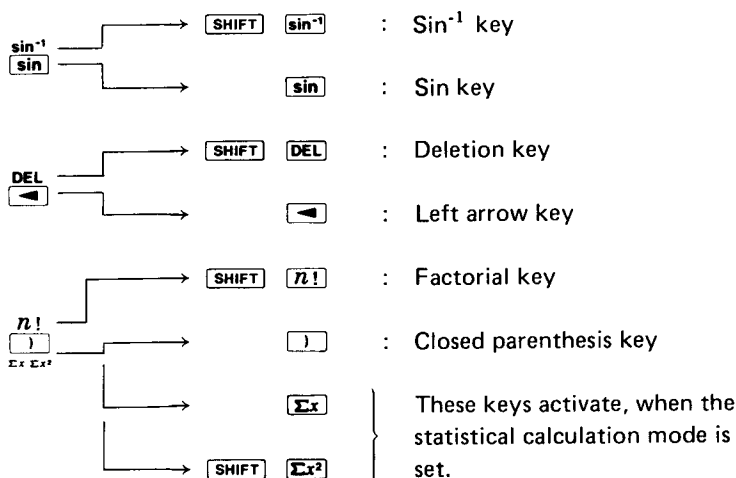
This section describes the basic CAL mode operations for the **PC-1401/1402** computer. Before getting started, correctly set your computer. First, press the **CAL** key to place the computer in the CAL mode. Then press **C-CE** **C-CE** , and make sure that the display shows the following initial information.



If not, read the following description and take the necessary action:

- ① More than one zero is displayed (e.g. 0.00):
The number of fractional digits is being specified. Press **SHIFT** **TAB** **.** to clear this specification.
- ② A dash (-) indicator is displayed at the STAT label:
The computer is in the statistical calculation mode. Press **SHIFT** **STAT** to clear the dash indicator.
- ③ RAD or GRAD is displayed instead of DEG:
The RAD, GRAD, and DEG indicate angular units for display data. Any of these symbols may be displayed unless a trigonometric function, inverse trigonometric function, or coordinate conversion is to be executed. Each of these symbols can be sequentially selected by operating **SHIFT** **DRG** .
- ④ Symbol **M** is displayed:
Numeric data is already in the memory. This symbol can be cleared by **C-CE** **X=M** .
- ⑤ All symbols displayed in the upper area of the display can be cleared with the **C-CE** key, with the exception of those described in the above items ③ and ④ .

In this manual, we will always show key functions as follows;



Press the **SHIFT** **TAB** and \bullet keys to set the floating decimal point system.

1. Addition, Subtraction

Key in the following:

12 **+** 45.6 **-** 32.1 **+** 789 **-** 741 **+** 213 **=**

Answer: 286.5

2. Multiplication, Division

a. Key in the following: 841 **x** 586 \div .12 **=** Answer: 4106883.333

b. Key in the following:

427 **+** 54 **x** 32 \div 7 **-** 39 **x** 2 **=** Answer: 595.8571429

Note that multiplication and division have priority to addition and subtraction. In other words multiplication and division will occur before addition and subtraction.

Constant Multiplication: The first number entered is the constant.

Key in: 3 **x** 5 **=**

Answer: 15

Key in: 10 **=**

Answer: 30

Constant Division: The number entered after the division sign is the constant.

Key in: 15 \div 3 **=**

Answer: 5

Key in: 30 **=**

Answer: 10

Using as a Calculator

Note: The machine retains some calculations depending on priority level.

Accordingly, in successive calculation the operator of the last calculation and the last numerical value are handled as a calculating instruction and a constant for constant calculation, respectively.

$a + b \times c =$	$+bc$	(Constant addition)
$a \times b \div c =$	$\div c$	(Constant division)
$a \div b \times c =$	$\frac{a}{b} \times$	(Constant multiplication)
$a \times b - c =$	$- c$	(Constant subtraction)

3. Memory Calculations

The independently accessible memory is indicated by the three keys: **[X↔M]**, **[RM]**, **[M+]**. Before starting a calculation clear the memory by pressing **[C-CE]** and **[X↔M]**.

Key in: 12 **[+]** 5 **[M+]** Answer: 17

→ To subtract key in: 2 **[+]** 5 **[=]** **[+/-]** **[M+]**

Answer to this equation: -7

Key in **[RM]** to recall memory: 10

Key in: 12 **[X]** 2 **[=]** **[X↔M]**

Answer: 24 (Also takes place of 10 in memory)

Key in: 8 **[÷]** 2 **[M+]**

Answer: 4 **[RM]** : 28

Note: • Memory calculations are impossible in the "STAT" mode. (Statistical calculation mode)

• When subtracting a number from the memory, press the **[+/-]** and **[M+]** keys.

Scientific Calculations in the CAL mode

To perform the trigonometric and inverse trigonometric and coordinate conversion, designate an angular mode. The angular mode is designated by the **[SHIFT]** and **[DRG]** keys.

1. Trigonometric functions

Put the angular mode at "DEG".

Calculate: $\sin 30^\circ + \cos 40^\circ =$

Key in the following: 30 **[sin]** + 40 **[cos]** =

Answer: 1.266044443

Calculate: $\cos 0.25\pi$

Put the angular mode at "RAD".

Key in: .25 **[X]** **[SHIFT]** **[π]** **[=]** **[cos]**

Answer: 0.707106781

(Remember to use the **[SHIFT]** key.)

2. Inverse Trigonometric Functions

Calculate: $\sin^{-1} 0.5$

Put the angular mode at "DEG".

Key in: .5 **[SHIFT]** **[sin⁻¹]**

Answer: 30

Calculate: $\cos^{-1} -1$

Put the angular mode at "RAD". (To enter a negative number, press the)

Key in: 1 **[+/-]** **[SHIFT]** **[cos⁻¹]** **[+/-]** key after numerals.

Answer: 3.141592654 (Value of π)

The answer of the inverse trigonometric functions will be displayed in the following area.

$$\theta = \sin^{-1} x, \quad \theta = \tan^{-1} x$$

$$\text{DEG: } -90 \leq \theta \leq 90 [^{\circ}]$$

$$\text{RAD: } -\frac{\pi}{2} \leq \theta \leq \frac{\pi}{2} [\text{rad}]$$

$$\text{GRAD: } -100 \leq \theta \leq 100 [^{\circ}]$$

$$\theta = \cos^{-1} x$$

$$\text{DEG: } 0 \leq \theta \leq 180 [^{\circ}]$$

$$\text{RAD: } 0 \leq \theta \leq \pi [\text{rad}]$$

$$\text{GRAD: } 0 \leq \theta \leq 200 [^{\circ}]$$

3. Hyperbolic and Inverse Hyperbolic Functions

Calculate: $\sinh 4$

Key in: 4 **[hyp]** **[sin]**

Answer: 27.2899172

Calculate: $\sinh^{-1} 9$

Key in: 9 **[SHIFT]** **[arcsinh]** **[sin]**

Answer: 2.893443986

4. Power Functions

Calculate: 20^2

Key in: 20 **[x²]**

Answer: 400

Calculate: 3^3 and 3^4

Key in: 3 **[y^x]** 3 **[=]**

Answer: 27

Key in: 3 **[y^x]** 4 **[=]**

Answer: 81

Using as a Calculator

5. Roots

Calculate: $\sqrt{25}$

Key in: 25 $\sqrt{\square}$

Answer: 5

Calculate: Cube root of 27

Key in: 27 SHIFT $\sqrt[3]{\square}$

Answer: 3

Calculate fourth root of 81

Key in: 81 SHIFT $\sqrt[4]{\square}$ 4 $=$

Answer: 3

6. Logarithmic Functions

Calculate: $\ln 21$, $\log 173$

Natural Logarithms: Key in: 21 \ln

Answer: 3.044522438

Common Logarithms: Key in: 173 \log

Answer: 2.238046103

7. Exponential Functions

Calculate: $e^{3.0445}$

Key in: 3.0445 SHIFT e^x

Answer: 20.99952881 (21 as in item "6" above)

Calculate: $10^{2.238}$

Key in: 2.238 SHIFT 10^x

Answer: 172.9816359 (173 as in item "6" above)

8. Reciprocals

Calculate: $1/6 + 1/7$

Key in: 6 $1/x$ $+$ 7 $1/x$ $=$

Answer: 0.309523809

9. Factorial

Calculate: 69!

Key in: 69 SHIFT $n!$

Answer: 1.711224524E 98 ($1.711224524 \times 10^{98}$)

Note that the Error section deals with the calculation limits of the calculator.

Calculate: ${}_8P_3 = \frac{8!}{(8-3)!} =$

Key in: 8 **[SHIFT]** **[n!]** **[÷]** (**[)]** 8 **[−]** 3 **[)]** **[SHIFT]** **[n!]** **[=]**

Answer: 336

10. Percent calculations

Calculate: 45% of 2,780 ($2,780 \times \frac{45}{100}$)

Key in: 2780 **[×]** 45 **[SHIFT]** **[Δ%]**

Answer: 1251

Calculate: $\frac{547 - 473}{473} \times 100$

Key in: 547 **[−]** 473 **[SHIFT]** **[Δ%]**

Answer: 15.6448203

11. Angle/Time conversions

To convert an angle given as degrees and minutes/seconds to its decimal equivalent, it must be entered as integer and decimal respectively.

Convert $12^{\circ}47'52''$ to its decimal equivalent

Key in: 12.4752 **[→DEG]**

Answer: 12.79777778

When converting decimal degrees to the equivalent degrees/minutes/seconds, the answer is broken down: integer portion = degrees; 1st and 2nd decimal digits = minutes; 3rd and 4th digits = seconds; and the 5th through end decimal digits are decimal seconds.

Convert 24.7256 to its degree/minute/second equivalent

Key in: 24.7256 **[SHIFT]** **[→DMS]**

Answer: 24.433216 or $24^{\circ}43'32''$

A horse has track times of 2 minutes 25 seconds, 2 minutes 38 seconds, and 2 minutes 22 seconds. What is the average running time?

Key in: .0225 **[→DEG]** **[+]** .0238 **[→DEG]** **[+]** .0222 **[→DEG]** **[=]**

Answer 1: 0.123611111

Key in: **[÷]** 3 **[=]**

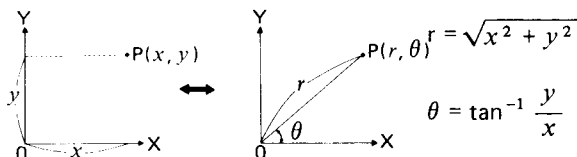
Answer 2: 0.041203703

Key in: **[SHIFT]** **[→DMS]**

Answer 3: 0.022833333 or the average time is 2 minutes 28 seconds

12. Coordinate Conversion

Converting rectangular coordinates to polar ($x, y \rightarrow r, \theta$)



DEG: $0 \leq |\theta| \leq 180$

RAD: $0 \leq |\theta| \leq \pi$

GRAD: $0 \leq |\theta| \leq 200$

Solve for $x = 6$ and $y = 4$ Mode = DEG

Key in: 6 $\frac{\square}{\square}$ 4 **SHIFT** **$\rightarrow r\theta$**

Answer: 7.211102551 (r)

Key in: $\frac{\square}{\square}$

Answer: 33.69006753 (θ)

Calculate the magnitude and direction (phase) in a vector $\vec{I} = 12 + j9$

Key in: 12 $\frac{\square}{\square}$ 9 **SHIFT** **$\rightarrow r\theta$**

Answer: 15 (r)

Key in: $\frac{\square}{\square}$

Answer: 36.86989765 (θ)

Converting polar coordinates to rectangular ($r, \theta \rightarrow x, y$)

Solve for $P(14, \pi/3)$, $r = 14$ $\theta = \pi/3$

Mode = RAD

Key in **SHIFT** **π** \div 3 **=** $\frac{\square}{\square}$ 14 $\frac{\square}{\square}$ **SHIFT** **$\rightarrow xy$**

Answer: 7.000000002 (x)

Key in: $\frac{\square}{\square}$

Answer: 12.12435565 (y)

In the above example
 $\theta = \frac{\pi}{3}$ is inputted first
 and is replaced with $r = 14$
 by pushing the $\frac{\square}{\square}$ key
 after r is inputted.

Use of Parenthesis

The parentheses keys are needed to cluster together a series of operations when it is necessary to override the priority system of algebra. When parentheses are in use on the PC-1401/1402 the symbol “()” will appear in the display.

Calculations in parentheses have priority over other calculations. Parentheses in the CAL mode can be used up to 15 times in a single level. Calculations within the inner-most set of parentheses will be calculated first.

Calculate: $12 + 42 \div (8 - 6)$

Key in: 12 **[+]** 42 **[÷]** **[(]** 8 **[−]** 6 **[)]** **[=]**

Answer: 33

Calculate: $126 \div [(3 + 4) \times (3 - 1)]$

Key in: 126 **[÷]** **[(]** **[(]** 3 **[+]** 4 **[)]** **[×]** **[(]** 3 **[−]** 1
[)] **[)]** **[=]**

Answer: 9

Note: The **[)]** keys located just before the **[=]** or **[M+]** key can be omitted.

Decimal Places

The **[SHIFT]** and **[TAB]** keys are used to specify the number of decimal digits in the calculation result. The number of places after the decimal point is specified by the numeral key (**[0]** ~ **[9]**) pressed after the **[SHIFT]** and **[TAB]** keys. To clear the designation of the decimal places, press the **[SHIFT]** **[TAB]** and **[.]** keys.

[SHIFT] **[TAB]** **[0]** → Designates 0 decimal place.
 (The 1st decimal place is rounded.)

[SHIFT] **[TAB]** **[1]** → Designates 1 decimal place.
 (The 2nd decimal place is rounded.)

}

[SHIFT] **[TAB]** **[9]** → Designates 9 decimal places.
 (The 10th decimal place is rounded.)

[SHIFT] **[TAB]** **[.]** → Designates the floating decimal point system. (Decimal place designation is reset.)
 (The 11th digit from the upper digit is rounded.)

Example: **[SHIFT]** **[TAB]** **[9]**

[.] **[5]** **[÷]** **[9]** **[=]** → 0.055555556
 (The 10th decimal place is rounded.)

[F↔E] → 5.555555556E−02
 (The 10th decimal place of the mantissa is rounded.)

[SHIFT] **[TAB]** **[3]** → 5.556E−02
 (The 4th decimal place of the mantissa is rounded.)

[F↔E] → 0.056

Using as a Calculator

SHIFT TAB .

→ 0.055555555

This is determined by the computer in the form of $5.555555555 \times 10^{-2}$.

Rounding the 11th digit of the mantissa results in $5.55555556 \times 10^{-2}$.

When changed to the floating decimal display, the rounded parts may not be displayed as in this example.

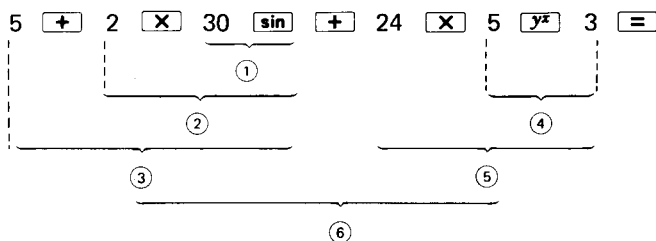
Priority level in the CAL Mode

The machine, provided with a function that judges the priority level of individual calculations, permits keys to be operated according to a given mathematical formula. The following shows the priority level of individual calculations.

Level Operations

- (1) Functions, such as \sin , x^2
- (2) y^x , $\sqrt[n]{y}$
- (3) \times , \div (Calculations which are given the same priority level are executed in sequence.)
- (4) $+$, $-$
- (5) $=$, $M+$, $\Delta\%$

Ex. Key operation and sequence of calculation in $5 + 2 \times \sin 30 + 24 \times 5^3 =$
Mode = DEG



The numbers ① ~ ⑥ indicates the sequence in which the calculations are carried out.

When calculations are executed from higher priority one in sequence a lower priority one must be reserved. The machine is provided with memories of eight levels to meet such requirement.

As the memories can also be used in a calculation including parentheses, calculation can be performed according to a given mathematical formula unless parentheses and pending operation exceed 8 levels in total.

- Single-variable functions are calculated immediately after key operation without being retained. (x^2 , $1/x$, $n!$, \rightarrow DEG, \rightarrow DMS, etc.)

< Calculation without using parentheses >

Ex. 1 $\boxed{+}$ 2 $\boxed{=}$ Pending of 1 level

①

1 $\boxed{+}$ 2 $\boxed{\times}$ 3 $\boxed{=}$ Pending of 2 levels

① ②

1 $\boxed{+}$ 2 $\boxed{\times}$ 3 $\boxed{y^x}$ 4 $\boxed{=}$ Pending of 3 levels

① ② ③

1 $\boxed{+}$ 2 $\boxed{\times}$ 3 $\boxed{y^x}$ 4 $\boxed{\div}$ 5

① ② ③
②

With the $\boxed{y^x}$ pressed, 3 calculations remain pending. Pressing the $\boxed{\div}$ key executes the calculations of " y^x " highest in priority level and " \times " identical in priority level. After the $\boxed{\div}$ key is pressed, the other 2 calculations will remain pending.

< Calculation using parentheses >

Ex. i) 1 $\boxed{+}$ 2 $\boxed{\times}$ 3 $\boxed{y^x}$ $\boxed{(}$ 4 $\boxed{\div}$ 5

① ② ③ ④

4 numerals and calculation instructions are left pending.

ii) 1 $\boxed{+}$ 2 $\boxed{\times}$ $\boxed{(}$ 3 $\boxed{-}$ 4 $\boxed{\div}$ 5 $\boxed{)}$

① ② ③ ④

Pressing the $\boxed{)}$ key executes the calculation of $3 - 4 \div 5$ in the parentheses, leaving 2 calculations pending.

- Parentheses can be used unless pending calculations exceed 8. However, parentheses can be continuously used up to 15 times.

Ex. $a \times (((b - c \times ((d + e) \times f) \div g) \dots\dots\dots$

↑ ↑
Parentheses, if continued, can be used up to 15.

Conversion between Decimal and Hex Numbers, and Hex Calculation

(**↔HEX** , **↔DEC**)

↔HEX : Allows you to convert a decimal number into hexadecimal number and, at the same times, place the **PC-1401/1402** in the HEX mode (display shows the symbol, HEX).

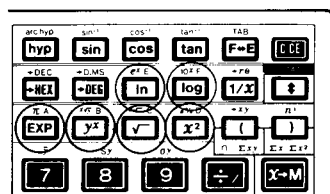
SHIFT **↔DEC** : Allows you to convert a hexadecimal number into decimal number and clear the HEX mode (symbol HEX disappears from the display).

The hexadecimal notation is one of the numeration systems broadly used in the computer field. The radix for the hex notation consists of numerals 0 through 9 and the uppercase letters A through F which are used in place of 10 through 15 in the decimal notation.

(Hexadecimal)		(Decimal)
A	_____	10
f	_____	f
F	_____	15

Hex numbers A through F can be entered by first placing your computer in the Hex mode (with **↔HEX** key), pressing the respective keys shown in figure.

The symbol HEX indicates that the numeric data shown in the display is a hex number, and that you can perform any arithmetic operation on hex numbers.



To clear the Hex mode, operate **SHIFT** **↔DEC** . You cannot clear it with the **C-CE** key.

Decimal to Hex Conversion

(Example) Convert a decimal number 30, into a hex number:

Key in: 30 **↔HEX**

Answer:

1E. HEX

To perform a new conversion, temporarily clear the HEX mode with **SHIFT** **↔DEC** .

(Example)

Convert a decimal number, -2, into a hex number:

Temporarily clear the Hex mode with **C-CE** **SHIFT** **←DEC** .Key in: **2** **+/-** **→HEX**

Answer:

FFFFFFFFFE. HEX

- If you attempt decimal-to-hex conversion on a negative decimal number, the computer internally performs “two’s complement” calculation and shows the result in 16’s complement.
- The **+/-** key may be used to reverse the polarity of the numeric data now in the display. If the polarity of a positive hex number is reversed, its complement will be obtained in the display.

(Example)

Convert a decimal number, 123.4, into hex number.

Key in: **SHIFT** **→DEC**
123.4 **→HEX**

Answer:

7B. HEX

- If a decimal number having a fractional part is converted into a hex number, the fractional part is rounded down and only the integer part is converted into a hex number.

Hex to Decimal Conversion

(Example)

Convert hex 2BC into decimal

Key in: **C-CE** **→HEX** 2 B C **SHIFT** **→DEC**

Answer:

700.

(Example)

Convert hex FFFFFFFF12 into decimal:

Key in: **C-CE** **→HEX** FFFFFFFF 12 **SHIFT** **→DEC**

Answer:

FFFFFFFF12. HEX**-238.**

- If a hex number between FFFFFFFF and FDABF41C01 is converted into decimal, the result will be a negative decimal number.

Using as a Calculator

Hexadecimal Calculation

Hexadecimal calculation can be done after your computer is placed in the Hex mode. Press **C-CE** **⇨HEX** and the symbol **HEX** displayed.

(Example) $A4 + BA =$

Key in: $A4$ **+** BA **=**

Answer:

15E. HEX

(350 in decimal)

(Example) $8 \times 3 =$

Key in: 8 **X** 3 **=**

Answer:

18. HEX

(24 in decimal)

(Example) $(12 + D) \times B =$

Key in: **C-CE** **(** 12 **+** D **)** **X** B **=**

Answer:

155. HEX

(341 in decimal)

(Example) $43A - 3CB =$

$+) A38 - 2FB =$

(Total)

Key in: **C-CE** **X↔M**

$43A$ **-** $3CB$ **(M+)**

Answer:

6F. HEX

$A38$ **-** $2FB$ **(M+)**

Answer:

73D. HEX

RM

Answer:

7AC. HEX

For hex calculation, you should note the following points:

- Hex calculation ignores all fractional parts. This means that the decimal point key, $\boxed{\cdot}$, is meaningless for hex calculation.
- If an intermediate result of a hex calculation includes a fractional part, an error will result.

(Example) B $\boxed{\div}$ 3 $\boxed{\times}$... Error (Symbol "E" is displayed.)

If a fractional part is in the final result, it will be truncated and only the integer part will be displayed.

(Example) B $\boxed{\div}$ 3 $\boxed{=}$... 3. HEX

- In the Hex mode, the $\boxed{+/-}$ key may be used to obtain a complement for the hex number now shown in the display.

(Example) A B $\boxed{+/-}$ → FFFFFFFF55. HEX

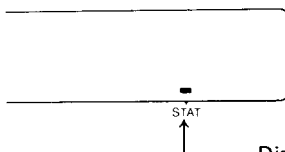
$\boxed{+/-}$ → AB. HEX

- In the Hex mode, the function keys on the computer are not usable.
- When the computer is in the Statistic mode (a dash (–) indicator is shown at the STAT label), conversion between decimal and hex numbers or hex computation is not executable.

Statistical calculation

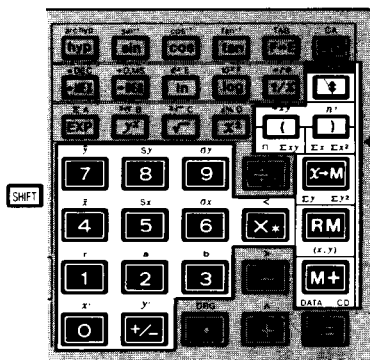
To perform statistical calculation, press the **SHIFT** and **STAT** (under a red **C-CE** key) keys in the CAL Mode, a dash (—) indicator will appear just above the “STAT” label in the lower right area of the display. The “STAT” stands for STATistics, and indicates that the computer is in the statistical calculation mode.

When the computer is in the RUN or PRO mode, press the **CAL** and then **SHIFT** **STAT**.



Display a dash indicator in this position by pressing the **SHIFT** and **STAT** keys.

Keys that are used mainly in the statistical calculation mode.



Intermediate results can be obtained and then additional data may be added.

When the statistical calculation is performed, the following statistics are automatically stored in the fixed variables used in the BASIC mode (memories). And these statistics can be used in the BASIC mode because these statistics are retained even when the statistical calculation mode is reset. These statistics are cleared when the current statistical calculation mode is reset and then this mode is set again.

Memory	Z	Y	X	W	V	U
Statistics	n	Σx	Σx^2	Σxy	Σy	Σy^2

To clear previous statistical inputs and calculations, reset the statistical calculation mode once and set this mode again. Otherwise when a new statistical calculation is performed, incorrect answer will be obtained.

When the statistical calculation mode is set, the followings can not be performed:

- * Memory calculation
- * Calculation with parentheses
- * Coordinate conversion
- * Hexadecimal \longleftrightarrow decimal notation conversions
- * Hexadecimal calculation

CAUTION

Of the statistical data obtained in the CAL mode, the following data is stored in the BASIC mode memory locations (U to Z).

Memory	Z	Y	X	W	V	U
Statistics	n	Σx	Σx^2	Σxy	Σy	Σy^2

When performing calculations using this statistical data, use the RUN mode.

For example, to determine the sum of squares (S^2) of four pieces of data, 205, 221, 226, and 220, operate your computer as follows:

$$\begin{aligned}
 S^2 &= \Sigma (x - \bar{x})^2 \\
 &= \Sigma x^2 - n\bar{x}^2 \\
 &= \Sigma x^2 - \frac{1}{n} (\Sigma x)^2
 \end{aligned}$$

- Enter the data in the CAL mode.

CAL SHIFT STAT

0.

205 DATA 221 DATA

226 DATA 220 DATA

4.

- Change the mode to the RUN and calculate the S^2 .

BASIC

>

X - Y * Y / Z

X - Y * Y / Z

ENTER

246.

1. One-variable statistical calculation

Calculate the following statistics.

- (1) n : Number of samples
- (2) Σx : Total of samples
- (3) Σx^2 : Sum of squares of samples
- (4) \bar{x} : Mean value of samples $\bar{x} = \frac{\Sigma x}{n}$
- (5) Sx : Standard deviation with population parameter taken to be "n-1".

$$Sx = \sqrt{\frac{\Sigma x^2 - n\bar{x}^2}{n-1}} \quad \text{(Used to estimate the standard deviation of population from the sample data extracted from that population.)}$$

- (6) σx : Standard deviation with population parameter taken to be "n".

$$\sigma x = \sqrt{\frac{\Sigma x^2 - n\bar{x}^2}{n}} \quad \text{(Used when all populations are taken to be sample data or when finding the standard deviation of population with sample taken to be a population.)}$$

Data for one-variable statistic calculations are inputted by the following operations

- (1) Data
- (2) Data Frequency (when two or more of the same data are inputted)

Example:

Calculate standard deviation, mean, and variance $(Sx)^2$ from the following data:
Set the computer at the statistical calculation mode.

Value	35	45	55	65
Frequency	1	1	5	2

As each sample is entered the number of that sample will appear on the right hand side of the display.

	Key in:	Display
35	<input type="button" value="DATA"/>	1.
45	<input type="button" value="DATA"/>	2.
55 x 5	<input type="button" value="DATA"/>	7.
65 x 2	<input type="button" value="DATA"/>	9.

Using as a Calculator

Press **SHIFT** **TAB** **2** (Set the Decimal at 2)

	Key in:	Display:
Mean:	SHIFT \bar{x}	53.89
Standard Deviation:	SHIFT Sx	9.28
Variance:	x^2	86.11

Correct Data (CD): The last entry above is an error and must be changed to 60×2 .

	Key in:	Display:
65	X 2 SHIFT CD	7.00
60	X 2 DATA	9.00

2. Two-variable Statistics and Linear Regression

In addition to the same statistical functions for y as for x in single-variable statistics, the sum of the products of samples $\sum xy$ is added in two-variable statistics. Two-variable statistics makes possible the development of a relationship (correlation) between two set of data. Each pair of data has an x and y value. From these sets of data a line of regression can be established. The relationship of the two sets of data by use of the straight line method is called "Linear Regression". In Linear Regression there are three important value; r , a , and b .

The equation of the straight line is $y = a + bx$, where a is the point at which the line crosses the Y-axis and b is the slope of the line.

The correlation coefficient r shows the relationship between two set of data. A perfect correlation between two values is an r equal to 1 (-1 is a perfect negative correlation); in other words, by knowing the value of one variable you can predict with 100% accuracy the value of the other variable. The further the value of r is from 1, the less reliable will your predictions be. The following table can be used as a set of definitions of the values of the correlation coefficient:

	Value of r	Call it
Positive Correlation	+0.80 to +1.00	Extra High
	+0.60 to +0.80	High
	+0.40 to +0.60	Moderate
	+0.20 to +0.40	Low
Negative Correlation	-0.20 to +0.20	Nil
	-0.20 to -0.40	Low
	-0.40 to -0.60	Moderate
	-0.60 to -0.80	High
	-0.80 to -1.00	Extra High

r Correlation coefficient

$$r = \frac{S_{xy}}{\sqrt{S_{xx} \cdot S_{yy}}}$$

a $a = \bar{y} - b\bar{x}$

b $b = \frac{S_{xy}}{S_{xx}}$

Coefficient of linear
regression equation
 $y = a + bx$

$$\left[\begin{array}{l} S_{xx} = \sum x^2 - \frac{(\sum x)^2}{n} \\ S_{yy} = \sum y^2 - \frac{(\sum y)^2}{n} \\ S_{xy} = \sum xy - \frac{\sum x \cdot \sum y}{n} \end{array} \right]$$

Example①: If we know a student's mark in mathematics, can we predict the mark in English?

The exam marks for five students chosen at random are given in the following table:

Student No. n	Mark in Math. x	Mark in English y
1	82	79
2	53	50
3	61	87
4	74	96
5	51	73
6	51	73

Key in:				Display
82	$\{x,y\}$	79	DATA	1.
53	$\{x,y\}$	50	DATA	2.
61	$\{x,y\}$	87	DATA	3.
74	$\{x,y\}$	96	DATA	4.
51	$\{x,y\}$	73	$\{X\}$ 2 DATA	6. (Note: to input multiple identical samples proceed as indicated)

Press **SHIFT** **TAB** **2**

SHIFT **r** 0.57
SHIFT **a** 34.26 (y-axis)
SHIFT **b** 0.68 (slope)

The value of r of .57 indicates that the correlation is moderate. The equation for the straight line for this data is $y = 34.26 + 0.68x$.

If we had a student whose mark in mathematics was 90, based on this analysis, the student would have a mark in English of 95.

Using as a Calculator

Example ②: Is weight a good predictor of longevity among men 65 years of age? In 1950, 10 men, each six feet tall, were selected for an experiment to determine if their weight effected their life span.

Sample	1	2	3	4	5	6	7	8	9	10
Age at death	72	67	69	85	91	68	77	74	70	82
Weight at age 65	185	226	200	169	170	195	175	174	198	172

Key in: STAT Mode, **SHIFT** **TAB** **•**

72 **(x,y)** 185 **DATA**

67 **(x,y)** 226 **DATA**

(Continue to place in all data)

SHIFT **r** -0.792926167

r indicates a relatively high negative correlation. Higher weight indicates a shorter life span. To graph the regression line, a and b are used.

SHIFT **a** 321.9292125 (y-axis)

SHIFT **b** -1.795088908 (slope)

Predict the age of death of a 6-foot man weighing 190 pounds in 1950.

190 **SHIFT** **x'** 73.5 years

To reach age 90, what should a man's weight be in 1960?

90 **SHIFT** **y'** 160.4 pounds

To reach age 150, what should a man's weight be? Obviously the answer will make no sense indicating the danger of carrying a straight-line extrapolation too far.

Calculation range

Four arithmetic calculations:

1st operand, 2nd operand and

calculation result : $\pm 1 \times 10^{-99} \sim \pm 9.999999999 \times 10^{99}$ and 0

Scientific functions:

Functions	Dynamic range	Note
$\sin x$ $\cos x$ $\tan x$	DEG: $ x < 1 \times 10^{10}$ RAD: $ x < \frac{\pi}{180} \times 10^{10}$ GRAD: $ x < \frac{10}{9} \times 10^{10}$ In $\tan x$, however, the following cases are excluded. DEG: $ x = 90 (2n - 1)$ RAD: $ x = \frac{\pi}{2} (2n - 1)$ $n = \text{integer}$ GRAD: $ x = 100 (2n - 1)$	
$\sin^{-1} x$ $\cos^{-1} x$	$-1 \leq x \leq 1$	
$\tan^{-1} x$	$ x < 1 \times 10^{100}$	
$\ln x$ $\log x$	$1 \times 10^{-99} \leq x < 1 \times 10^{100}$	$(\ln x = \log_e x)$
e^x	$-1 \times 10^{100} < x \leq 230.2585092$	$(e \doteq 2.718281828)$
10^x	$-1 \times 10^{100} < x < 100$	
y^x	<ul style="list-style-type: none"> $y > 0$: $-1 \times 10^{100} < x \log y < 100$ $y = 0$: $x > 0$ $y < 0$: x: integer or $\frac{1}{x}$: odd number $-1 \times 10^{100} < x \log y < 100$ 	$y^x = 10^{x \cdot \log y}$
$\sqrt[x]{y}$	<ul style="list-style-type: none"> $y > 0$: $-1 \times 10^{100} < \frac{1}{x} \log y < 100, x \neq 0$ $y = 0$: $x > 0$ $y < 0$: x or $\frac{1}{x}$: integer ($x \neq 0$) $-1 \times 10^{100} < \frac{1}{x} \log y < 100$ 	$\sqrt[x]{y} = 10^{\frac{1}{x} \cdot \log y}$
$\sqrt[3]{x}$	$ x < 1 \times 10^{100}$	
$\sinh x$ $\cosh x$ $\tanh x$	$-227.9559242 \leq x \leq 230.2585092$	
$\sinh^{-1} x$	$ x < 1 \times 10^{50}$	
$\cosh^{-1} x$	$1 \leq x < 1 \times 10^{50}$	
$\tanh^{-1} x$	$ x < 1$	

Using as a Calculator

Functions		Dynamic range	Note
\sqrt{x}		$0 \leq x < 1 \times 10^{100}$	
x^2		$ x < 1 \times 10^{50}$	
$\frac{1}{x}$		$ x < 1 \times 10^{100}$ $x \neq 0$	
$n!$		$0 \leq n \leq 69$ (n: Integer)	
→ DEG		$ x < 1 \times 10^{100}$	
→ DMS		$ x < 1 \times 10^{100}$	
HEX → DEC		$0 \leq x \leq 2540BE3FF$ $FDABF41C01 \leq x \leq FFFFFFFF$	x is an integer in HEX mode
DEC → HEX		$ x \leq 9999999999$	x is an integer.
$x, y \rightarrow r, \theta$		$(x^2 + y^2) < 1 \times 10^{100}$ $\frac{y}{x} < 1 \times 10^{100}$	$r = \sqrt{x^2 + y^2}$ $\theta = \tan^{-1} \frac{y}{x}$
$r, \theta \rightarrow x, y$		$r < 1 \times 10^{100}$, $ r \sin \theta < 1 \times 10^{100}$ $ r \cos \theta < 1 \times 10^{100}$	$x = r \cos \theta$ $y = r \sin \theta$ θ is in the same condition as x of $\sin x, \cos x$.
Statistical calculation	Data CD	$ x < 1 \times 10^{50}$ $ y < 1 \times 10^{50}$ $ \Sigma x < 1 \times 10^{100}$ $\Sigma x^2 < 1 \times 10^{100}$ $ \Sigma y < 1 \times 10^{100}$ $\Sigma y^2 < 1 \times 10^{100}$ $ \Sigma xy < 1 \times 10^{100}$ $ n < 1 \times 10^{100}$	
	\bar{x}	$n \neq 0$	
	Sx	$n \neq 1$ $0 \leq \frac{\Sigma x^2 - n\bar{x}^2}{n-1} < 1 \times 10^{100}$	
	σx	$n \neq 0$ $0 \leq \frac{\Sigma x^2 - n\bar{x}^2}{n} < 1 \times 10^{100}$	
	\bar{y}	$n \neq 0$	
	Sy	$n \neq 1$ $0 \leq \frac{\Sigma y^2 - n\bar{y}^2}{n-1} < 1 \times 10^{100}$	
	σy	$n \neq 0$ $0 \leq \frac{\Sigma y^2 - n\bar{y}^2}{n} < 1 \times 10^{100}$	

Functions		Dynamic range	Note
Statistical calculation	r	$n \neq 0$ $0 < (\sum x^2 - n\bar{x}^2) \cdot (\sum y^2 - n\bar{y}^2) < 1 \times 10^{100}$ $\left \sum xy - \frac{\sum x \cdot \sum y}{n} \right < 1 \times 10^{100}$ $\left \frac{\sum xy - \frac{\sum x \cdot \sum y}{n}}{\sqrt{(\sum x^2 - n\bar{x}^2) \cdot (\sum y^2 - n\bar{y}^2)}} \right < 1 \times 10^{100}$	
	b	$n \neq 0$ $0 < \sum x^2 - n\bar{x}^2 < 1 \times 10^{100}$ $\left \sum xy - \frac{\sum x \cdot \sum y}{n} \right < 1 \times 10^{100}$ $\left \frac{\sum xy - \frac{\sum x \cdot \sum y}{n}}{\sum x^2 - n\bar{x}^2} \right < 1 \times 10^{100}$	
	a	a is the same condition as b, and $ \bar{y} - b\bar{x} < 1 \times 10^{100}$	
	y'	$ a + bx < 1 \times 10^{100}$	
	x'	$\left \frac{y - a}{b} \right < 1 \times 10^{100}$	

For the accuracy of functions other than shown above, the error is ± 1 at the 10th digits, as a rule. (In the scientific notation system, the error is ± 1 at the lowest digit of mantissa display.)

However, the accuracy will become low around-singular points and inflection points of functions.

Therefore, errors are accumulated in each stage of the continuous calculations, causing the accuracy to deteriorate. (The same applies to other continuous calculations made by the computer such as y^x and $\sqrt[x]{y}$.)

Manual Calculation

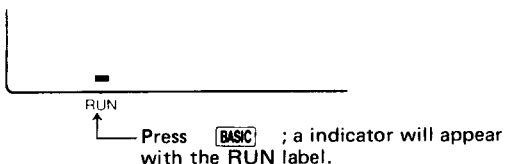
What is Manual Calculation

The PC-1401/1402 Computer may be basically used in two ways. One way lets you store whole calculation procedure or steps into the computer's memory as a program in advance, then lets the computer automatically execute it later. The other way lets you calculate step by step through manual key operations. The latter is called "manual calculation".

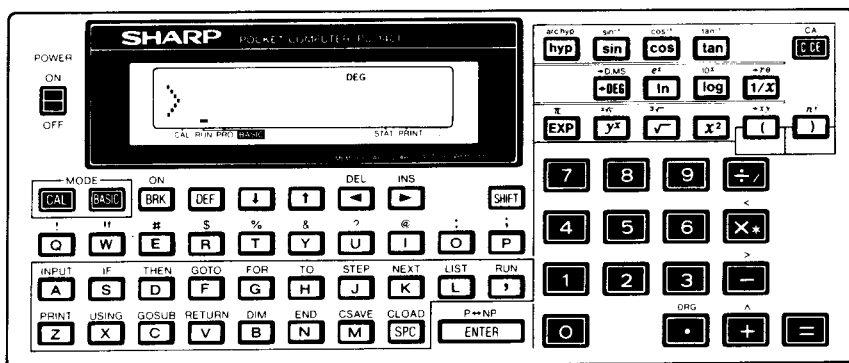
Of course, in the CAL mode, all calculations are performed manually. But here we only call manual calculation performed in the BASIC mode (RUN or PROgram mode) manual calculation.

How to Manually Calculate

Let's try manual calculation in the RUN mode. Press the BASIC key to place your computer in the RUN mode.



In the RUN mode, the keys and functions shown in the following figure are operative (the same is true of the PROgram mode)



Before going into operation examples, let's touch on some important points in operation.

While we usually use operators $+$, $-$, \times , or \div for our mathematical calculations on paper, we don't use the operators \times and \div for our arithmetic operations in BASIC. Instead of \times and \div we use an asterisk ($*$) and slash ($/$), respectively.

The operators $*$ and $/$ can be entered by pressing $\boxed{\times}$ and $\boxed{\div}$ keys, respectively. **To get the result of manual calculation, operate the $\boxed{\text{ENTER}}$ key instead of $\boxed{=}$ key.**

Do not use dollar signs or commas when entering calculations into the **PC-1401/1402**. These characters have special meaning in the BASIC programming language. Now try these simple arithmetic examples. Remember to clear with the $\boxed{\text{C-CE}}$ between calculations.

Input	Display
$\boxed{5} \boxed{0} \boxed{+} \boxed{5} \boxed{0} \boxed{\text{ENTER}}$	100.
$\boxed{1} \boxed{0} \boxed{0} \boxed{-} \boxed{5} \boxed{0} \boxed{\text{ENTER}}$	50.
$\boxed{6} \boxed{0} \boxed{*} \boxed{1} \boxed{0} \boxed{\text{ENTER}}$	600.
$\boxed{3} \boxed{0} \boxed{0} \boxed{/} \boxed{5} \boxed{\text{ENTER}}$	60.
$\boxed{1} \boxed{0} \boxed{\text{SHIFT}} \boxed{\wedge} \boxed{2} \boxed{\text{ENTER}}$	100.
$\boxed{2} \boxed{*} \boxed{\text{SHIFT}} \boxed{\pi} \boxed{\text{ENTER}}$	6.283185307
$\boxed{\sqrt{}} \boxed{6} \boxed{4} \boxed{\text{ENTER}}$	8.

Recalling Entries

Even after the **PC-1401/1402** has displayed the results of your calculation, you can redisplay your last entry. To recall, use the left $\boxed{\blacktriangleleft}$ and right $\boxed{\blacktriangleright}$ arrows.

The left arrow, $\boxed{\blacktriangleleft}$, recalls the expression with the cursor positioned after the last character.

The right arrow, $\boxed{\blacktriangleright}$, recalls the expression with the cursor positioned "on top of" the first character.

Using as a Calculator

Remember that the left and right arrows are also used to position the cursor along a line. The right and left arrows are very helpful in editing (or modifying) entries without having to retype the entire expression.

You will become familiar with the use of the right and left arrows in the following examples. Now, take the role of the manager and perform the calculations as we discuss them.


As the head of personnel in a large marketing division, you are responsible for planning the annual sales meeting. You expect 300 people to attend the three day conference. For part of this time, the sales force will meet in small groups. You believe that groups of six would be a good size. How many groups would this be?

Input

3 0 0 / 6 ENTER

Display

50.



On second thought you decide that groups containing an odd number of participants might be more effective. Recall your last entry using the  .

Input



Display

300/6_


To calculate the new number of groups you must replace the six with an odd number. Five seems to make more sense than seven. Because you recalled using the  arrow, the cursor is positioned at the end of the display. Use the  to move the cursor one space to the left.

Input



Display

300/6

Notice that after you move the cursor it becomes a flashing block  . Whenever you position the cursor "on top of" an existing character, it will be displayed as the flashing cursor.

Type in a 5 to replace the 6. One caution in replacing characters — once you type a new character over an existing character, the original is gone forever! You cannot recall an expression that has been typed over.

Input

5

ENTER

Display

300/5_

60.

Sixty seems like a reasonable number of groups, so you decide that each small group will consist of five participants.


Recall is also useful to verify your last entry, especially when your results do not seem to make sense. For instance, suppose you had performed this calculation:

Input

3 0 / 5 ENTER



Display

6.

Even a tired, overworked manager like you realizes that 6 does not seem to be a reasonable result when you are dealing with hundreds of people! Recall your entry using the .

InputDisplay

30/5

Because you recalled using the  the flashing cursor is now positioned over the first character in the display. To correct this entry you wish to insert an added zero. Using the , move the cursor until it is positioned over the zero. When making an INSert, you position the flashing cursor over the character **before** which you wish to make the insertion.

InputDisplay

30/5

Use the INSert key to make space for the needed character.

Input

SHIFT INS

Display

300/5

Using as a Calculator

Pressing **INS**ert moves all the characters one space to the right, and inserts a bracketed open slot. The flashing cursor is now positioned over this open space, indicating the location of the next typed input. Type in your zero. Once the entry is corrected, display your new result.

Input

0

ENTER

Display

300/5

60.

On the other hand, suppose that you had entered this calculation:

Input

3 **0** **0** **0** **/** **5** **ENTER**

Display

600.

The results seem much too large. If you only have 300 people attending the meeting, how could you have 600 "small groups"? Recall your entry using the **▶**.

Input

▶

Display

3000/5

The flashing cursor is now positioned over the first character in the display. To correct this entry eliminate one of the zeros. Using the **▶** move the cursor to the first zero (or any zero). When deleting a character, you position the cursor "on top of" the character to be deleted.

Input

▶

Display

3000/5

Now use the **DEL**ete key to get rid of one of the zeros.

Input

SHIFT **DEL**

Display

300/5

Pressing DELeTe causes all the characters to shift one space to the left. It deletes the character it is "on top of" and the space the character occupies. The flashing cursor stays in the same position indicating the next location for input. Since you have no other changes to make, complete the calculation.

<u>Input</u>	<u>Display</u>
ENTER	60.

(Note: Pressing the SPaCe key, when it is positioned over a character, replaces the character leaving a blank space. DELeTe eliminates the character and the space it occupied.)

Errors

Recalling your last entry is essential when you get the dreaded ERROR message. Let us imagine that, unintentionally, you typed this entry into the **PC-1401/1402**:

<u>Input</u>	<u>Display</u>
3 0 0 / / 5 ENTER	ERROR 1

Naturally you are surprised when this message appears! ERROR 1 is simply the computer's way of saying, "I don't know what you want me to do here". To find out what the problem is, recall your entry using either the ◀ or ▶ arrow.

<u>Input</u>	<u>Display</u>
◀	300 / 5

When you use the ◀ or ▶ key, the flashing cursor indicates the point at which the computer got confused. And no wonder, you have too many operators! To correct this error use the DELeTe key.

<u>Input</u>	<u>Display</u>
SHIFT DEL ENTER	60.

If, upon recalling your entry after an ERROR 1, you find that you have **omitted** a character, use the INSert sequence to correct it.

When using the **PC-1401/1402** as a calculator, the majority of the errors you encounter will be ERROR 1 (an error in syntax). For a complete listing of error messages, see APPENDIX A.

Serial Calculations

The **PC-1401/1402** allows you to use the results of one calculation as part of the following calculation.

Part of your responsibility in planning this conference is to draw up a detailed budget for approval. You know that your total budget is \$150.00 for each attendant. Figure your total budget:

Input	Display
[3] [0] [0] [*] [1] [5] [0] [ENTER]	45000.

Of this amount you plan to use 15% for the final night's awards presentation. When performing serial calculations it is not necessary to retype your previous results, but **DO NOT** Clear between entries (Do not use the **C-CE** in this time.). What is the swards budget?

Input	Display
[*] [.] [1] [5]	45000.*.15_

Notice that as you type in the second calculation (*.15), the computer automatically displays the result of your first calculation at the left of the screen and includes it in the new calculation. In serial calculations the entry must begin with an operator. As always, you end the entry with **[ENTER]** :

NOTE: The $\frac{\%}{\square}$ and $\frac{1\%}{\square}$ keys can not be used in the calculation. The $\frac{\%}{\square}$ key should be used as a character only and the $\frac{1\%}{\square}$ keys is inoperative.

Example: 45000 [*] 15 [SHIFT] [%] → ERROR 1

Input	Display
[ENTER]	6750.

Continue allocating your budget. The hotel will cater you dinner for \$4000:

Input	Display
[-] [4] [0] [0] [0]	6750.-4000_
[ENTER]	2750.

Decorations will be \$1225:

Input

- **1** **2** **2** **5** **ENTER**

Display

1525.

Finally, you must allocate ~~\$2200~~ for the speaker and entertainment:

Input

- **2** **2** **0** **0** **ENTER**

Display

-675.

Obviously, you will have to change either your plans or your allocation of resources!

Negative Numbers

Since you want the awards dinner to be really special, you decide to stay with the planned agenda and spend the additional money. However, you wonder what percentage of the total budget will be used up by this item. First, change the sign of the remaining sum:

Input

***** **-** **1**

Display

-675.*-1_

ENTER

675.

Now you add this result to your original presentation budget:

Input

+ **6** **7** **5** **0** **ENTER**

Display

7425.

Diving by ~~45000~~ gives you the percentage of the total budget this new figure represents:

Input

/ **4** **5** **0** **0** **0** **ENTER**

Display

0.165

Fine, you decide to allocate 16.5% to the awards presentation.

Compound Calculations and Parentheses

In performing the above calculations, you could have combined several of these operations into one step. For instance, you might have typed both these operations on one line:

$$675+6750/45000$$

Compound calculations, however, must be entered very carefully:

$675+6750/45000$ might be interpreted as

$$\frac{675+6750}{45000} \quad \text{or} \quad 675 + \frac{6750}{45000}$$

When performing compound calculations, the **PC-1401/1402** has specific rules of expression evaluation and operator priority (see Page 69). Be sure you get the calculation you want by using parentheses to clarify your expressions:

$$(675+6750) / 45000 \quad \text{or} \quad 675+(6750 / 45000)$$

To illustrate the difference that the placement of parentheses can make, try these two examples:

Input	Display
(6 7 5 + 6 7 5 0) / 4 5 0 0 0 ENTER	0.165
6 7 5 + (6 7 5 0 / 4 5 0 0 0) ENTER	675.15

Using Variables in Calculations

The **PC-1401/1402** can store up to 26 fixed **variables** under the alphabetic characters A to Z. If you are unfamiliar with the concept of variables, they are more fully explained in Chapter 4. You designate variables with an Assignment Statement:

$$A = 5$$

$$B = -2$$

You can also assign the value of one variable (right) to another variable (left):

$$C = A + 3$$

$$D = E$$

A variable may be used in place of a number in any calculation.

Now that you have planned your awards dinner, you need to complete arrangements for your conference. You wish to allocate the rest of your budget by percentages also. First you must find out how much money is still available. Assign a variable (R) to be the amount left from the total:

Input

R **=** **4** **5** **0** **0** **0** **0**
- **7** **4** **2** **5**

ENTER

Display

R=45000-7425_

37575.

As you press **ENTER** the **PC-1401/1402** performs the calculation and displays the new value of R. You can display the current value of any variable by entering the alphabetic character it is stored under:

Input

R **ENTER**

Display

37575.

You can then perform calculations using your variable. The value of (R) will not change until you assign it a new value.

You wish to allocate 60% of the remaining money to room rental:

Input

R ***** **.** **6** **0**

ENTER

Display

R*.60_

22545.

Similarly, you want to allocate 25% of your remaining budget to conduct management training seminars:

Using as a Calculator

Input

R * . 2 5 ENTER

Display

9393.75

Variables will retain their assigned values even if the machine is turned OFF or undergoes an AUTO OFF. Variables are lost only when:

- * You assign a new value to the same variable.
- * You type in CLEAR **ENTER** (not the clear key (C-CE)).
- * You clear the machine using the ALL RESET button.
- * The batteries are changed.

These are certain limitations on the assignment of variables, and certain programming procedures which cause them to be changed. See Chapter 4 for a discussion of assignment. See Chapter 5 for a discussion of the use of variables in programming.

Chained Calculations

In addition to combining several operators in one calculation, the PC-1401/1402 also allows you to perform several calculations one after the other – without having to press **ENTER** before moving on. You must separate the equations with commas. Only the result of the **final** calculation is displayed. (Remember too, that the maximum line length accepted by the computer is 80 characters including **ENTER**.)

You wonder how much money would have been available for rooms if you had kept to your original allocation of 15% for the awards dinner:

Input

R = . 8 5 * 4 5 0 0 0 , R * . 6 0 _
0 0 0 , R *
. 6 0

Display

.85*45000,R*.60_

Although the computer performs all the calculations in the chain, it displays only the final result:

Input

ENTER

Display

22950.

To find the value of R used in this calculation, enter R:

Input

R **ENTER**

Display

38250.

Error Message

If an error occurred as a result of manual calculation, an error message will appear in the display such as:

ERROR 1

or

ERROR 2

The error state can be cleared with either the **C-CE** or **◀** or **▶** key. If the **◀** or **▶** key is used to clear the error state, the portion of the formula where the error occurred is recalled in the display (see the description for the recall feature).

Scientific Notation

People who need to deal with very large and very small numbers often use a special format called exponential or **scientific notation**. In scientific notation a number is broken down into two parts.

The first part consists of a regular decimal number between 1 and 10. The second part represents how large or small the number is in powers of 10.

As you know, the first number to the left of the decimal point in a regular decimal number shows the number of 1's, the second shows the number of 10's, the third the number of 100's, and the fourth the number of 1000's. These are simply increasing powers of 10:

$$10^0 = 1, 10^1 = 10, 10^2 = 100, 10^3 = 1000, \text{ etc.}$$

Scientific notation breaks down a decimal number into two parts: one shows what the numbers are, the second other shows how far a number is to the left, or right, of the decimal point. For example:

1234 becomes 1.234 times 10^3 (3 places to the right)

654321 becomes 6.54321 times 10^5 (5 places to the right)

.000125 becomes 1.25 times 10^{-4} (4 places to the left)

Scientific notation is useful for many shortcuts. You can see that it would take a lot of writing to show 1.0 times 10^{87} — a 1 and 87 zeros! But, in scientific notation this number looks like this:

1.0×10^{87}

or

1.0E 87

The **PC-1401/1402** uses scientific notation whenever numbers become too large to display using decimal notation. This computer uses the capital letter **E** to mean "times ten to the":

1234567890000 is displayed as 1.23456789 E 12

.0000000000001 is displayed as **1. E -12**

Those of you who are unfamiliar with this type of notation should take some time to put in a few very large and very small numbers to note how they are displayed.

Limits

The largest number which the **PC-1401/1402** can handle is ten significant digits, with two digit exponents. In other words the largest number is:

[illegible]

and the smallest number is:

[illegible]

Under certain circumstances, when numbers will be used frequently, the **PC-1401/1402** uses a special compact form. In these cases there are special limits imposed on the size of numbers, usually either 0 to 65535 or -32768 to +32767. Those with some can be represented in 16 binary bits. The circumstances in which this form is used are noted in the Chapter 8.

Last Answer Feature

In the case of the serial calculation, you could use the result of the calculation only as the first member of the subsequent calculation formula.
Refer to the following example.

Input3 $\boxed{+}$ 4 $\boxed{\text{ENTER}}$ $\boxed{*}$ 5 $\boxed{\text{ENTER}}$ Display

7.

7.*5_

35.

Press $\boxed{\text{C-CE}}$, then the $\boxed{\downarrow}$ or $\boxed{\uparrow}$ key. If you operated these keys just after completing the calculation example above, you should see "35." in your display. The numeric data displayed is the result of the your **last** calculation.

The **PC-1401/1402** can "remember" the last answer (result) obtained through manual calculation, and recall it on its display with the $\boxed{\downarrow}$ or $\boxed{\uparrow}$ key.

In the case of the serial calculation described above, you could use the result of the previous calculation only as the first member of the subsequent calculation formula. With the last answer feature, however, you can place the result of the previous calculation in any position of the subsequent calculation.

(Example) Use the result (6.25) of the operation, $50 \div 8$, to compute $12 \times 5 \div 6.25 + 24 \times 3 \div 6.25 = :$

Input50 $\boxed{/}$ 8 $\boxed{\text{ENTER}}$ 12 $\boxed{*}$ 5 $\boxed{/}$ $\boxed{\uparrow}$ $\boxed{+}$ 24 $\boxed{*}$ 3 $\boxed{/}$ $\boxed{\downarrow}$ $\boxed{\text{ENTER}}$ $\boxed{\text{C-CE}}$ $\boxed{\downarrow}$ Display

6.25

Last answer $\longrightarrow \uparrow$

12*5/6.25_

Last answer recalled

/6.25+24*3/6.25_

Last answer recalled

21.12

21.12_

The last answer is replaced with the result of the previous calculation by performing a manual calculation with the $\boxed{\text{ENTER}}$ key.


Using as a Calculator

As shown in this example, the last answer can be recalled anytime and anyplace, but will be replaced with a new last answer resulting from the last calculation.

The last answer is not cleared by the **C-CE** or key operation.

- The last answer cannot be recalled; when the computer is not in the RUN mode, program execution is temporarily halted, or the Trace mode is selected.

Length of Formula

The length of a formula you can put into your computer has a certain limitation. With the **PC-1401/1402**, up to 79 key strokes can be used to enter a single calculation formula (excluding the **ENTER** key). If you attempt the 80th key stroke, the cursor () will start blinking on that character, indicating that the 80th key entry is not valid.

Scientific calculations in the BASIC mode

This computer has many scientific functions which can be used in BASIC mode.

To perform scientific functions you must press **ENTER** at the end of the input or your calculations will not be acted upon by the computer.

These functions will be described as follows:

Functions	Definition in PC-1401/ 1402	Operation	Remark
Trigonometric functions			
sin	SIN	sin	
cos	COS	cos	
tan	TAN	tan	
Inverse trigonometric functions			
\sin^{-1}	ASN	SHIFT sin⁻¹	
\cos^{-1}	ACS	SHIFT cos⁻¹	
\tan^{-1}	ATN	SHIFT tan⁻¹	
Hyperbolic functions			
sinh	HSN	hyp sin	
cosh	HCS	hyp cos	
tanh	HTN	hyp tan	

Functions	Definition in PC-1401/ 1402	Operation	Remark
Inverse hyperbolic functions			
\sinh^{-1}	AHS	SHIFT arChyp sin⁻¹	
\cosh^{-1}	AHC	SHIFT arChyp cos⁻¹	
\tanh^{-1}	AHT	SHIFT arChyp tan⁻¹	
Logarithmic functions			
\ln	LN	ln	$\log_e x$
\log	LOG	log	$\log_{10} x$
Exponential functions			
e^x	EXP	SHIFT e^x	$e \approx 2.718281828$
10^x	TEN	SHIFT 10^x	
Reciprocal			
$\frac{1}{x}$	RCP	1/x	
Square			
x^2	SQU	x²	
Square root			
$\sqrt{\quad}$	$\sqrt{\quad}$ or SQR	√	
Cubic root			
$\sqrt[3]{\quad}$	CUR	SHIFT $\sqrt[3]{\quad}$	
Factorial			
$n!$	FACT	SHIFT n!	
Pi	π or PI	SHIFT π	$\pi \approx 3.141592654$
DMS → DEG	DEG	◀DEG	
DEG → DMS	DMS	SHIFT →DMS	
Power			
y^x	\wedge	SHIFT ^ or y^x	$y \wedge x : y^x$
Power root			
$\sqrt[x]{y}$	ROT	SHIFT $\sqrt[x]{y}$	$y \text{ ROT } x : \sqrt[x]{y}$
Rectangular coordinates → Polar coordinates	POL	SHIFT →Pθ	

Using as a Calculator

Functions	Definition in PC-1401/ 1402	Operation	Remark
Polar coordinates → Rectangular coordinates	REC	[SHIFT] [↔xy]	
Integer	INT	[I] [N] [T]	INT(x)
Absolute x	ABS	[A] [B] [S]	ABS(x)
Sign	SGN	[S] [G] [N]	SGN(x) $x > 0 : 1$ $x = 0 : 0$ $x < 0 : -1$

Of these functions, the INT, ABS, and SGN can be entered by using alphabetic keys. Some other functions may also be entered with alphabetic keys. For examples, "sin 30" may be entered either by operating **[sin]** 30 or **[S]** **[I]** **[N]** 30. For trigonometric and inverse trigonometric functions and coordinate conversion, the desired angular unit must be specified in advance. In manual calculation, angular units may be specified either by operating **[SHIFT]** **[DRG]** or with the following instructions:

Angular unit	Command	Display Symbol	Description
Degree	DEGREE	DEG	Represents a right angle as $90 [^\circ]$.
Radian	RADIAN	RAD	Represents a right angle as $\pi/2$ [rad].
Grad	GRAD	GRAD	Represents a right angle as 100 [g].

These instructions are used to specify angular units in program. For practice, use these instructions to specify angular units in the following calculation examples:

(Example) $\sin 30^\circ =$

(Operation) DEGREE **[ENTER]** (Specifies "degree" for angular unit.)

SIN 30 **[ENTER]**

DEG
0.5

(Example) $\tan \frac{\pi}{4} =$

(Operation) RADIAN **[ENTER]** (Specifies "radian" for angular unit.)

TAN (PI/4) **[ENTER]**

RAD
1.

(Example) $\cos^{-1}(-0.5) =$

(Operation) DEGREE **ENTER** (Specifies "degree" for angular unit.)

ACS - 0.5 **ENTER**

DEG
120.
(120°)

(Example) $\log 5 + \ln 5 =$

(Operation) LOG 5 + LN 5 **ENTER**

2.308407917

(Example) $e^{2+3} =$

(Operation) EXP (2 + 3) **ENTER**
(Do not use the **EXP**)

148.4131591

(Example) $\sqrt[3]{4^3 + 5^3} =$

(Operation) CUR (4 ^ 3 + 5 ^ 3) **ENTER**

5.738793548

(Example) Convert 30 deg. 30 min. in sexagenary notation into decimal notation.

(Operation) DEG 30. 30 **ENTER**

30.5
(30.5 degree)

(Example) Convert 30.755 deg. in decimal notation into sexagenary notation

(Operation) DMS 30.755 **ENTER**

30.4518
(30 deg. 45 min. 18 sec.)

Using as a Calculator

(Example) Conversion from orthogonal into polar coordinates: Determine the polar coordinate (r, θ) for point P (3, 8) on an orthogonal coordinate:

(Operation) DEGREE (Specifies "degree" for angular unit.)

POL (3, 8) (r)

8.544003745

($r \doteq 8.5$)

Z (θ)

69.44395478

($\theta \doteq 69^\circ$)

* The value of θ is transferred to variable Z, and the value of r to variable Y.

(Example) Conversion from polar into rectangular coordinates: Determine rectangular coordinate (x, y) for point P $(12, \frac{4}{5} \pi)$ on a polar coordinate.

(Operation) RADIAN (Specifies "radian" for angular unit.)

REC (12, (4/5*PI))

(x)

-9.708203933

($x \doteq -9.7$)

Z (Y)

7.053423028

($y \doteq 7.1$)

* The values of y and x are transferred to variables Z and Y, respectively.

Note: For coordinate conversion, the conversion result is transferred to variables Z and Y. Therefore, the previous contents of Z and Y will be cleared.

— Reference —

Equations comprised of logical operators ($=$, $>$, $<$, \geq , \leq , $\langle \rangle$) can take on the values listed in the following table: x

x and y represent numeric values.

$=*$	1 if $x = y$ 0 if $x \neq y$	\geq	1 if $x \geq y$ 0 if $x < y$
$>$	1 if $x > y$ 0 if $x \leq y$	\leq	1 if $x \leq y$ 0 if $x > y$
$<$	1 if $x < y$ 0 if $x \geq y$	$\langle \rangle$	1 if $x \neq y$ ($\langle \rangle$ means " \neq ".) 0 if $x = y$

* If, for example, " $A = \text{numeric value}$ " or " $B = \text{formula}$ " is used in a logical equation, the computer will not treat it as a logical equation but as an assignment statement for variables. When using an equal ($=$) sign for a logical equation, use it in the form of " $\text{numeric value} = A$ " or " $\text{formula} = B$ ", with the exception of conditional expressions used in IF statements.

Direct Calculation Feature

In the manual calculation described up to now, we always used the **ENTER** key to terminate a formula and obtain the calculation result of the formula. However, you can directly operate the functions of the **PC-1401/1402** computer with the desired function key (without operating the **ENTER** key) when the objective numeric data is in the display.

(Example) Determine $\sin 30^\circ$ and $8!$.

(Operation) **DEGREE** **ENTER**

C-CE 30

30 _

sin

0.5

(Operation) **C-CE** 8

8 _

SHIFT **7!**

40320.

(Example) For $\tan^{-1} \frac{5}{12}$, first check the result of $\frac{5}{12}$, then determine $\tan^{-1} \frac{5}{12}$.

(Operation) **DEGREE** **ENTER**

5/12 **ENTER**

4.166666667E-01

SHIFT **\tan^{-1}**

22.61986495

It should be noted, however, that this "direct" calculation mode is not available for functions requiring entry of more than one numeric value (binominal functions) such as power, power root, or coordinate conversion.

The direct calculation feature is not effective for formulas:

(e.g.) **C-CE** $5 * 4 \rightarrow 5 * 4$ _

log $\rightarrow 5 * 4 \text{LOG}$ _

The direct calculation feature is effective only for numeric values. Therefore, if hex numbers A to F are entered for hex to decimal conversion, the direct calculation feature will remain inoperative. In such a case, use the ordinary manual calculation using the **ENTER** key.

- After a direct calculation is done, the recall feature is not operative. Operation of the **◀** or **▶** key will only display the cursor.

Priority in Manual Calculation

In the BASIC mode, you can type in formulas in the exact order in which they are written, including parentheses or functions. The order of priority in calculation and treatment of intermediate results will be taken care of by the computer itself.

The internal order of priority in manual calculation is as follows:

- 1) Recalling variables or π .
- 2) Function (sin, cos, etc.)
- 3) Power (\wedge) or power root (ROT)
- 4) Sign (+, -)
- 5) Multiplication or division ($*$, /)
- 6) Addition or subtraction (+, -)
- 7) Comparison of magnitude (>, >=, <, <=, <>)
- 8) Logical AND, OR

- Notes:
- * If parentheses are used in a formula, the operation given within the parentheses has the highest priority.
 - * Composite functions are operated from right to left ($\sin \cos^{-1} 0.6$).
 - * Chained power (3^{4^2} or 3^{4^2}) or power root are operated from right to left.
 - * For the above items 3) and 4), the last entry has a higher priority.

(e.g.) $-2^4 \rightarrow -(2^4)$

$3^{-2} \rightarrow 3^{-2}$

CHAPTER 4

CONCEPTS AND TERMS OF BASIC

In this Chapter we will examine some concepts and terms of the BASIC language.

String Constants

In addition to numbers, there are many ways that the **SHARP PC-1401/1402** uses letters and special symbols. These letters, numbers, and special symbols are called characters. These characters are available on the **PC-1401/1402**:

1 2 3 4 5 6 7 8 9 0
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
! " # \$ % & () * + , - . / : ; < = > ? @ $\sqrt{\quad}$ π ^

In BASIC, a collection of characters is called a **string**. In order for the **PC-1401** to tell the difference between a string and other parts of a program, such as verbs or variable names, you must enclose the characters of the string in quotation marks ("").

The following are examples of string constants:

"HELLO"
"GOODBYE"
"SHARP PC-1401"

The following are **not** valid string constants:

"COMPUTER	No ending quote
"ISN""T"	Quote can't be used within a string

Hexadecimal Numbers

The decimal system is only one of many different systems to represent numbers. Another which has become quite important when using computers is the hexadecimal system. The hexadecimal system is based on 16 instead of 10. To write hexadecimal numbers you use the familiar 0 ~ 9 and 6 more "digits": A, B, C, D, E, and F. These correspond to 10, 11, 12, 13, 14, and 15. When you want the **PC-1401/1402** to treat a number as hexadecimal put an ampersand '&' character in front of the numeral:

&A = 10
&10 = 16
&100 = 256
&FFFF = 65535

Those with some computer background may notice that the last number (65535) is the same as the largest number in the special group of limits discussed in the paragraph "Limits" on page 60. Hexadecimal notation is never required in using the **PC-1401/1402**, but there are special applications where it is convenient.

Variables

Computers are made up of many tiny memory areas called **bytes**. Each byte can be thought of as a single character. For instance, the word byte requires four bytes of memory because there are four characters in it. To see how many bytes are available for use, simply type in MEM **ENTER**. The number displayed is the number of bytes available for writing programs. This technique works fine for words, but is very inefficient when you try to store numbers. For this reason, numbers are stored in a coded fashion. Thanks to this coding technique, your computer can store large numbers in only eight bytes. The largest number that can be stored is +9.99999999 E + 99.

The smallest number is +1.E-99. This gives you quite a range to choose from. However, if the result of a calculation exceeds this range, the computer will let you know by turning on the error annunciator and by displaying the error message in the screen. This annunciator is a small E in the upper right-hand corner of the screen. For the error message refer to the Appendix A. To see it right now type in:

9 **EXP** 99 * 9 **ENTER**

To get the computer working properly again, just press the **C-CE** key. But how do you go about storing all this information? It's really very easy. The computer likes to use names for different pieces of data. Let's store the number 556 into the computer. You may call this number by any name that you wish, but for this exercise, let's use the letter R. The statement, LET, can be used to instruct the computer to assign a value to a variable name but only in a program statement. However, the LET command is not necessary, so we will not use it very often. Now, type in R = 556 and press the **ENTER**. The computer now has the value 556 associated with the letter R. These letters that are used to store information are called **variables**. To see the content of the variable R, press the **C-CE** key, the R key and the **ENTER** key. The computer responds by showing you the value 556 on the right of your screen. This ability can become very useful when you are writing programs and formulas.

Next, let's use the R variable in a simple formula. In this formula, the variable R stands for the radius of a circle whose area we want to find. The formula for the area of a circle is: $A = \pi * R^2$. Type in R **SHIFT** **^** 2 ***** **SHIFT** **π** **ENTER**. The result is 971179.3866. This technique of using variables in equations will become more understandable as we get into writing programs.

So far, we've only discussed numeric variables. What about storing alphabetic characters? Well, the idea is the same, but, so the computer will know the difference between the two kinds of variables, add a \$ to the variable name. For instance, let's store the word BYTE in the variable B\$. Notice the \$ after the B?

Concepts and Terms of BASIC

This tells the computer that the contents of the letter B is alphabetic, or string data.

To illustrate this, key in B **[SHIFT] [R]** = **[SHIFT] [W]** B YTE **[SHIFT] [W] [ENTER]** . The value BYTE is now stored in the variable B\$. To make sure of this, type in B **[SHIFT] [R] [ENTER]** . The screen shows BYTE. This time the display is on the left side of the screen, instead of the right.

Variables handled by the **SHARP PC-1401/1402** are divided into the followings;

Variables	{	Numeric variables	{	Fixed numeric variables (A to Z)
			{	Simple numeric variables (AB, C1, etc.)
	{	String variables	{	Numeric array variables
			{	Fixed character variables (A\$ to Z\$)
			{	Simple character variables (BB\$, C2\$, etc.)
			{	Character array variables

Fixed Variables

The first section, fixed variable, is always used by the computer for storing data. It can be thought of as pre-allocated variable space. In other words, no matter how much memory your program uses up, you will always have at least 26 variables to choose from to store data in. This data can be one of two types: NUMERIC or STRING (alphabetic character). Fixed memory location are eight bytes long and can be used for only one type of data at a time. To illustrate this, type in the following example:

```
A = 123 [ENTER]  
A$ [ENTER]
```

You get the message:

ERROR 9

This means that you have put numeric data into the area of memory called A and then told the computer to show you that information again as STRING data. This confuses the computer so it says that there is an error condition. Press the **[C-CE]** key to clear error condition. Now try the following example:

```
A$ = "ABC" [ENTER]  
A [ENTER]
```

Again, the computer is confused and gives the ERROR 9 message. Look at Figure shown below to see that the variable name A equals the same area in memory as the variable name A\$, and that B equals B\$, and so on for all the letter of the alphabet.

Figure:

A = A\$ = A(1) = A\$(1)
 B = B\$ = A(2) = A\$(2)
 C = C\$ = A(3) = A\$(3)
 D = D\$ = A(4) = A\$(4)
 E = E\$ = A(5) = A\$(5)
 F = F\$ = A(6) = A\$(6)
 G = G\$ = A(7) = A\$(7)
 H = H\$ = A(8) = A\$(8)
 I = I\$ = A(9) = A\$(9)
 J = J\$ = A(10) = A\$(10)
 K = K\$ = A(11) = A\$(11)
 L = L\$ = A(12) = A\$(12)
 M = M\$ = A(13) = A\$(13)
 N = N\$ = A(14) = A\$(14)
 O = O\$ = A(15) = A\$(15)
 P = P\$ = A(16) = A\$(16)
 Q = Q\$ = A(17) = A\$(17)
 R = R\$ = A(18) = A\$(18)
 S = S\$ = A(19) = A\$(19)
 T = T\$ = A(20) = A\$(20)
 U = U\$ = A(21) = A\$(21)
 V = V\$ = A(22) = A\$(22)
 W = W\$ = A(23) = A\$(23)
 X = X\$ = A(24) = A\$(24)
 Y = Y\$ = A(25) = A\$(25)
 Z = Z\$ = A(26) = A\$(26)

Simple Variables

Simple variable names are specified by two (or more) alpha-numeric characters, such as AA or B1. Unlike fixed variables, simple variables have no dedicated storage area in the memory. The area for simple variables is automatically set aside (within the program and data area) when a simple variable is first used.

Since separate memory areas are defined for simple numeric variables and simple character variables even if they have the same name, variables such as AB and AB\$, for example, may be used at the same time.

While alphanumeric characters are usable for simple variable names, the first character of a variable name must always be an alphabetic character. If more than two characters are used to define a variable name, only the first two characters are meaningful.

Concepts and Terms of BASIC

- Note:
- The function or BASIC instruction names to the **PC-1401/1402** computer are not usable for variable names.
(e.g.) PI, IF, TO, ON, SIN, etc.
 - Each simple character variable can hold up to 16 characters or symbols.

Array Variables

For some purposes it is useful to deal with numbers as an organized group, such as a list of scores or a tax table. In BASIC these groups are called **arrays**. An array can be either **one-dimensional**, like a list, or **two-dimensional**, like a table.

To define an array, the DIM (short for dimension) statement is used. Arrays must always be “declared” (defined) before they are used. (Not like the single-value variables we have been using.) The form for the numeric DIMension statement is:

DIM numeric-variable-name (size)

where:

numeric-variable-name is a variable name which conforms to the normal rules for numeric variable names previously discussed.

size is the number of storage locations and must be number in the range 0 through 255. Note that when you specify a number for the size you get one more location than you specified.

Examples of legal numeric DIMension statements are:

```
DIM X (5)
DIM AA (24)
DIM Q5 (0)
```

The first statement creates an array X with 6 storage locations. The second statement creates an array AA with 25 locations. The third statement creates an array with one location and is actually rather silly since (for numbers at least), it is the same as declaring a single-value numeric variable.

It is important to know that an array-variable X and a variable X are separate and distinct to SHARP. The first X denotes a series of numeric storage locations, and the second a single and different location.

Now that you know how to create arrays, you might be wondering how it is that we refer to each storage location. Since the entire group has only one name, the way in which we refer to a single location (called an "element") is to follow the group name with a number in parentheses. This number is called a "subscript". Thus, for example, to store the number 8 into the fifth element of our array X (declared previously) we would write:

$X(4) = 8$

If the use of 4 is puzzling, remember that the numbering of elements begins at zero and continues through the size number declared in the DIM statement.

The real power of arrays lies in the ability to use an expression or a variable name as a subscript.

To declare a character array a slightly different form of the DIM statement is used:

DIM character-variable-name (size) * length

where:

character-variable-name is a variable name which conforms to the rules for normal character variables as discussed previously.

size is the number of storage locations and must be in the range 0 through 255. Note that when you specify a number, you get one more location than you specified.

*length is optional. If used, it specifies the length of each of the strings that comprise the array. Length is a number in the range 1 to 80. If this clause is not used, the strings will have the default length of 16 characters.

Example of legal character array declarations are:

```
DIM X$(4)
DIM NMS(10) * 10
DIM IN$(1) * 80
DIM R$(0) * 26
```

The first example creates an array of five strings each able to store 16 characters. The second DIM statement declares an array NM with eleven strings of 10 characters each. Explicit definition of strings smaller than the default helps to conserve memory space. The third example declares a two element array of 80-character strings and the last example declares a single string of twenty-six characters.

Concepts and Terms of BASIC

Besides the simple arrays we have just studied, the **PC-1401/1402** allows "two-dimensional" arrays. By analogy, a one-dimensional array is a list of data arranged in a single column. A two-dimensional array is a table of data with rows and columns. The two-dimensional array is declared by the statement:

DIM numeric-variable-name (row, columns)

or

DIM character-variable-name (rows, columns) * length

where:

rows specifies the number of rows in the array. This must be a number in the range 0 through 255. Note that when you specify the number of rows you get one more row than the specification.

columns specifies the number of columns in the array. This must be a number in the range 0 through 255. Note that when you specify the number of columns you get one more column than the specification.

The following diagram illustrates the storage locations that result from the declaration `DIM T (2, 3)` and the subscripts (now composed of two numbers) which pertain to each storage location:

	column 1	column 2	column 3	column 4
row 0	T (0, 0)	T (0, 1)	T (0, 2)	T (0, 3)
row 1	T (1, 0)	T (1, 1)	T (1, 2)	T (1, 3)
row 2	T (2, 0)	T (2, 1)	T (2, 2)	T (2, 3)

Note: Two-dimensional arrays can rapidly eat up storage space. For example, an array with 25 rows and 35 columns uses 875 storage locations!

Arrays are very powerful programming tools.

The following table shows the number of bytes used to define each variable and the number used by each program statement.

Variable	Variable name	Data	
Numeric variable	7 bytes	8 bytes	
String variable	7 bytes	Array variable	Specified number
		Simple variable (two-character variable)	16 bytes

- * For example, if DIM Z\$(2, 3) * 10 is specified, 12 variables, each capable of storing 10 characters, are reserved. This requires 7 bytes (variable name) + 10 bytes (number of characters) x 12 = 127 bytes.

Element	Line number	Statement & function	Others, ENTER
Number of bytes used	3 bytes	1 byte	1 byte

Variables in the Form of A ()

While a data area on the computer's memory is set aside for fixed variables, it may also be used to define subscripted variables which have the same form as array variables.

There are 26 fixed variable names available: i.e. A through Z (A\$ through Z\$). Each of these names can be subscripted with the numbers 1 through 26, such as A(1) – A(26) or A\$(1) – A\$(26). This means that variable A(1) may be used in place of variable A, A(2) in place of B, A(3) in place of C, and so forth.

However, if an array named A or A\$ has already been defined by the DIM statement, subscripted variables named A cannot be defined. For example, if an array A is defined by DIM A(5) the location for A(0) through A(5) are set aside in the program/data area. So if you specify variable A(2), it does not refer to the fixed variable B, but refers to the array variable A(2) defined in the program/data area. If you specify A(9), it will cause an error since A(9) is outside the range of the dimension specified by the DIM A(5) statement.

In turn, if subscripted variables are already defined in the form of A(), it is not possible to define arrays A or A\$ by using the DIM statement, unless the definition for the subscripted variables is cleared with the CLEAR statement.

- * Using subscripts in excess of 26:

If subscripts greater than 26 are used for subscripted variables A() when array A is not defined by a DIM statement, the corresponding locations in the program/data area are set aside for these A() variables. For instance, if you execute A(35) = 5, locations for variables A(27) to A(35) will be reserved in the program/data area.

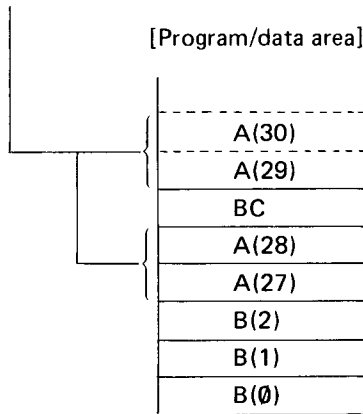
Concepts and Terms of BASIC

While variables subscripted in excess of 26 are treated as array variables, they are subject to the following special restrictions:

- (1) Locations for an array with the same name must contiguous in the program/data area. Otherwise, an error will occur.

```
10 DIM B(2)
20 A(28) = 5
30 BC = 12
40 A(30) = 9
```

If this program is excuted, the array named "A" is not defined in two consecutive segments in the program data area, and an error will result at line 40.



- (2) Numeric array variables and character array variables with the same subscript cannot be defined at the same time. For example, A(30) and A\$(30) cannot be defined at the same time. Since they use the same location in the program/data area.
- (3) Two dimensional arrays cannot be defined, nor is it possible to specify the length of character strings to be held in character array variables. For example, the length of a character string which can be held in the character array variable A\$() is limited to seven characters or less.
- (4) Variable subscripted with zero (0) cannot be defined. If A(0) or A\$(0) is defined, an error will result.

Expressions

An **expression** is some combination of variables, constants, and operators which can be evaluated to a single value. The calculations which you entered in Chapter 3 were examples of expressions. Expressions are an intrinsic part of BASIC programs. For example, an expression might be a formula that computes an answer to some equation, a test to determine the relationship between two quantities, or a means to format a set of strings.

Numeric Operators

The **PC-1401/1402** has five **numeric operators**. These are the arithmetic operators which you used when exploring the use of the **PC-1401/1402** as a calculator in Chapter 3:

- + Addition
- − Subtraction
- * Multiplication
- / Division
- ^ Power

A **numeric expression** is constructed in the same way that you entered compound calculator operations. Numeric expressions can contain any meaningful combination of numeric constants, numeric variables, and these numeric operators:

$(A * B) \wedge 2$
 $A(2, 3) + A(3, 4) + 5.0 - C$
 $(A/B) * (C+D)$

String Expressions

String expressions are similar to numeric expressions except that there is only one string operator -- concatenation (+). This is the same symbol used for plus. When used with a pair of strings, the + attaches the second string to the end of the first string and makes one longer string. You should take care in making more complex string concatenations and other string operations because the work space used by the **PC-1401/1402** for string calculations is limited to only 79 characters.

NOTE: String quantities and numeric quantities cannot be combined in the same expression unless one uses one of the functions which convert a string value into a numeric value or vice versa:

"15" + 10 is illegal
"15" + "10" is "1510", not "25"

Relational Expressions

A **relational expression** compares two expressions and determines whether the stated relationship is True or False. The relational operators are:

> Greater Than
>= Greater Than or Equal To
= Equals
<> Not Equal To
<= Less Than or Equal To
< Less Than

The following are valid relational expressions:

A < B
C(1,2) >= 5
D(3) <> 8

If A was equal to 10, B equal to 12, C(1, 2) equal to 6, and D(3) equal to 9, all of these relational expressions would be True.

Character strings can also be compared in relational expressions. The two strings are compared character by character according to their ASCII value starting at the first character (see Appendix B for ASCII values). If one string is shorter than the other, a 0 or NUL will be used for any missing positions. All of the following relational expressions are True:

"ABCDEF" = "ABCDEF"
"ABCDEF" <> "ABCDE"
"ABCDEF" > "ABCDE"

Relational expressions evaluate to either True or False. The **PC-1401/1402** represents True by a 1; False is represented by a 0. In any logical test an expression which evaluates to 1 or more will be regarded as True while one which evaluates to 0 or less will be considered False. Good programming practice, however, dictates the use of an explicit relational expression instead of relying on this coincidence.

Logical Expressions

Logical expressions are relational expressions which use the operators AND, OR, and NOT. AND and OR are used to connect two relational expressions; the value of the combined expression is shown in the following tables:

A AND B

Value of A

Value of B	Value of A	
	True	False
True	True	False
False	False	False

A OR B

Value of A

Value of B	Value of A	
	True	False
True	True	True
False	True	False

(Note: Value of A and B must be 0 or 1)

- Decimal numbers can be expressed in the binary notation of 16 bits as follows:

DECIMAL NOTATION	BINARY NOTATION OF 16-BIT
32767	0111111111111111
⋮	⋮
3	0000000000000011
2	0000000000000010
1	0000000000000001
0	0000000000000000
-1	1111111111111111
-2	1111111111111110
-3	1111111111111101
⋮	⋮
-32768	1000000000000000

The negative (NOT) of a binary number 0000000000000001 is taken as follows:

NOT 0000000000000001
(Negative) → 1111111111111110

Concepts and Terms of BASIC

Thus, 1 is inverted to 0, and 0 to 1 for each bit, which is called "to take negative (NOT)."

Then, the following will result when 1 and NOT 1 are added together:

$$\begin{array}{r} 0000000000000001 \quad (1) \\ +) \quad 1111111111111110 \quad (\text{NOT } 1) \\ \hline 1111111111111111 \quad (-1) \end{array}$$

Thus, all bits become 1. According to the above number list, the bits become -1 in decimal notation, that is $1 + \text{NOT } 1 = -1$.

The relationship between numerical value X and its negative (NOT X) is:

$$X + \text{NOT } X = -1$$

This results in an equation of $\text{NOT } X = -X - 1$

i.e. $\text{NOT } X = -(X + 1)$

From the equation the following are found to result.

$$\text{NOT } 0 = -1$$

$$\text{NOT } -1 = 0$$

$$\text{NOT } -2 = 1$$

More than two relational expressions can be combined with these operators. You should take care to use parentheses to make the intended comparison clear.

$$(A < 9) \text{ AND } (B > 5)$$

$$(A >= 10) \text{ AND NOT } (A > 20)$$

$$(C = 5) \text{ OR } (C = 6) \text{ OR } (C = 7)$$

The **PC-1401/1402** implements logical operators as "bitwise" logical functions on 16 bit quantities. (See note on relational expressions and True and False). In normal operations this is not significant because the simple 1 and 0 (True and False) which result from a relational expression uses only a single bit. If you apply a logical operator to a value other than 0 or 1, it works on each bit independently. For example if A is 17, and B is 22, (A OR B) is 23:

17 in binary notation is 10001

22 in binary notation is 10110

17 OR 22 is 10111 (1 if 1 in either number, otherwise 0)

10111 is 23 in decimal.

If you are a proficient programmer, there are certain applications where this type of operation can be very useful. Beginning programmers should stick to clear, simple True or False relational expressions.

Parentheses and Operator Precedence

When evaluating complex expressions the **PC-1401/1402** follows a predefined set of priorities which determine the sequence in which operators are evaluated. This can be quite significant:

$5 + 2 * 3$ could be

$$5 + 2 = 7$$

$$7 * 3 = 21$$

or

$$2 * 3 = 6$$

$$6 + 5 = 11$$

The exact rules of "operator precedence" are given in Appendix D.

To avoid having to remember all these rules and to make your program clearer, always use parentheses to determine the sequence of evaluation. The above example is clarified by writing either:

$$(5 + 2) * 3$$

or

$$5 + (2 * 3)$$

RUN Mode

In general, any of the above expressions can be used in the RUN mode well as in programming a BASIC statement. In the RUN mode an expression is computed and displayed immediately. For example:

Input

$(5 > 3) \text{ AND } (2 < 6)$

Display

1.

The 1 means that the expression is True.

Functions

Functions are special components of the BASIC language which take one value and transform it into another value. Functions act like variables whose value is determined by the value of other variables or expressions. ABS is a function which produces the absolute value of its argument:

ABS (-5) is 5

ABS (6) is 6

Concepts and Terms of BASIC

LOG is a function which computes the log to the base 10 of its argument.

LOG (100) is 2

LOG (1000) is 3

A function can be used any place that a variable can be used. Many functions do not require the use of parentheses:

LOG 100 is the same as LOG (100)

You must use parentheses for functions which have more than one argument. Using parentheses always makes programs clearer.

See Chapter 8 for a complete list of functions available on the **PC-1401/1402**.

CHAPTER 5

PROGRAMMING THE PC-1401/1402

In the previous chapter we examined some of the concepts and terms of the BASIC programming language. In this chapter you will use these elements to create programs on the **PC-1401/1402**. Let us reiterate however, this is **not** a manual on how to program in BASIC. What this chapter will do is familiarize you with the use of BASIC on your **PC-1401/1402**.

Programs

A **program** consists of a set of instruction to the computer. Remember the **PC-1401/1402** is only a machine. It will perform the exact operations that you specify. You, the programmer, are responsible for issuing the correct instructions.

BASIC Statements

The **PC-1401/1402** interprets instructions according to a predetermined format. This format is called a **statement**. You always enter BASIC statements in the same pattern. Statements must start with a line number:

```
10: PRINT "HELLO"
20: END
30: :
```

Line Numbers

Each line of a program must have a unique line number — any integer between 1 and 65279. Line numbers are the reference for the computer. They tell the **PC-1401/1402** the order in which to perform the program. You need not enter lines in sequential order (although if you are a beginning programmer, it is probably less confusing for you to do so). The computer always begins execution with the lowest line number and moves sequentially through the lines of a program in ascending order.

When programming it is wise to allow increments in your line numbering (10, 20, 30, . . . 10, 30, 50 etc). This enables you to insert additional lines if necessary. **CAUTION: Do not use the same line numbers in different programs.** If you use the same line number, the oldest line with that number is deleted when you enter the new line.

BASIC Verbs

All BASIC statements must contain **verbs**. Verbs tell the computer what action to perform. A verb is always contained within a program, and as such is not acted upon immediately.

Programming

```
10: PRINT "HELLO"  
20: END  
30:  :
```

Some statements require or allow an **operand**:

```
10: PRINT "HELLO"  
20: END  
30:  :
```

Operands provide information to the computer telling it what data the verb will act upon. Some verbs require operands, with other verbs they are optional. Certain verbs do not allow operands. (See Chapter 8 for a complete listing of BASIC verbs and their use on the **PC-1401/1402**).

BASIC Commands

Commands are instructions to the computer which are entered outside of a program. Commands instruct the computer to perform some action with your program or to set modes which effect how your programs are executed.

Unlike verbs, commands have immediate effects — as soon as you complete entering the command (by pressing the **ENTER** key), the command will be executed. Commands **are not** preceded by a line number:

```
RUN  
NEW  
RADIAN
```

Some verbs may also be used as commands. (See Chapter 8 for a complete listing of BASIC commands and their use on the **PC-1401/1402**).

Modes

The RUN mode is also used to execute the programs you create.

The PROgram mode is used to enter and edit your programs.

Beginning to Program on the PC-1401/1402

After all your practice in using the **PC-1401/1402** as a calculator you are probably quite at home with the keyboard. From now on, when we show an entry, we will **not** show every keystroke. Remember to use **[SHIFT]** to access characters above the keys and **END EVERY LINE BY PRESSING THE [ENTER] KEY**.

Now you ready to program.

Set the **POWER SWITCH** to the **ON** position and then press the **[BASIC]** key twice. You will see the following initial information in the display.



The above display shows that the computer is in **PRO**gram mode.

(If a dash indicator is at the **CAL** or **RUN** label, press the **[BASIC]** key once or twice.) Enter the **NEW** command.

Input

NEW

Display

>

The **NEW** command clears the **PC-1401/1402**'s memory of all existing programs and data. The prompt appears after you press **[ENTER]**, indicating that the computer is awaiting input.

Example 1 – Entering and Running a Program

Make sure the **PC-1401/1402** is in the **PRO** mode and enter the following program:

Input

10 PRINT "HELLO"

Display

10: PRINT "HELLO"

Notice that when you push **[ENTER]** the **PC-1401/1402** displays your input, automatically inserting a colon (:) between the line number and the verb. Verify that the statement is in the correct format.

Now press the **[BASIC]** key to set the **RUN** mode.



Input

RUN

Display

HELLO

Programming

Since this is the only line of the program, the computer will stop executing at this point. Press **[ENTER]** to get out of the program and reenter RUN if you wish to execute the program again.

Example 2 — Editing a Program

Suppose you wanted to change the message that your program was displaying, that is you wanted to edit your program. With a single line program you could just retype the entry, but as you develop more complex programs editing becomes a very important component of your programming. Let's edit the program you have just written.

Are you still in the RUN mode? If so change to the PROgram mode.

You need to recall your program in order to edit it. Use the Up Arrow (**↑**) to recall your program. If your program was completely executed, the **↑** will recall the last line of the program. If there was an error in the program, or if you used the BREAK (**[BRK]**) key to stop execution, the **↑** will recall the line in which the error or BREAK occurred. To make changes in your program use the **↑** to move up in your program (recall the previous line) and the **↓** to move down in your program (display the next line). If held down the **↑** and the **↓** will scroll vertically, that is they will display each line moving up or down in your program.

You will remember that to move the cursor within a line you use the **▶** (right arrow) and **◀** (left arrow). Using the **▶** position the cursor over the first character you wish to change:

Input

↑

◀◀◀◀◀

Display

10: PRINT "HELLO"

10 PRINT "HELLO"

Notice that the cursor is now in the flashing block form indicating that it is "on top of" an existing character. Type in:

Input

GOOD"!

Display

10 "GOOD"!_

Don't forget to press **[ENTER]** at the end of the line. Change to the RUN mode with the **[BASIC]**.

Input

RUN

Display

ERROR 1 IN 10

This is a new kind of error message. Not only is the error type identified (our old friend the syntax error) but the **line number** in which the error occurs is also indicated.

Press the **C-CE** and then return into the PROgram mode. You must be in the PROgram mode to make changes in a program. Using the **↑**, recall the last line of your program.

Input

↑

Display

10: PRINT "GOOD" !

The flashing cursor is positioned over the problem area. In Chapter 4 you learned that when entering string constants in BASIC all characters must be contained within quotation marks. Use the DELeTe key to eliminate the "!":

Input

DEL

Display

10 PRINT "GOOD" _

Now let's put the ! in the correct location. When editing programs, DELeTe and INSert are used in exactly the same way as they are in editing calculations (See Chapter 3). Using the **◀** position the cursor on top of the character which will be the first character following the insertion.

Input

◀

Display

10 PRINT "GOOD"

Press the INSert key. A **□** will indicate the spot where the new data will be entered:

Input

INS

Display

10 PRINT "GOOD" □

Type in the !. The display looks like this:

Input

!

Display

10 PRINT "GOOD!"

Programming

Remember to press **ENTER** so the correction will be entered into the program.

NOTE: If you wish to DELeTe an entire line from your program just type in the line number and the original line will be eliminated.

Example 3 – Using Variables in Programming

If you are unfamiliar with the use of numeric and string variables in BASIC, reread these sections in Chapter 4.

Using variables in programming allows much more sophisticated use of the **PC-1401/1402's** computing abilities.

Remember, you assign numeric fixed variables using any letter from A to Z:

```
A = 5
```

To assign string variables you also use a letter, followed by a dollar sign. **Do not use the same letter in designating a numeric and a string fixed variable.** You cannot designate A and A\$ in the same program.

Remember that string fixed variables cannot exceed 7 characters in length:

```
A$ = "TOTAL"
```

The values assigned to a variable can change during the execution of a program, taking on the values typed in or computed during the program. One way to assign a variable is to use the INPUT verb. In the following program the value of A\$ will change in response to the data typed in answering the inquiry "WORD?". Enter this program:

```
10 INPUT "WORD?"; A$
20 B = LEN (A$)
30 PRINT "WORD_IS_"; B; " LTRS"
40 END
```

_____ means space

Before you RUN the program notice several new features. Line 30 of this program exceeds the 16 character maximum of the **PC-1401/1402's** display. When a line is longer than 16 characters (up to the 79 character maximum), **PC-1401/1402** moves the characters to the left as the 16 character maximum is exceeded. This does not destroy the previous input. This move to the left is referred to as horizontal scrolling.

The second new element in this program is the use of the END statement to signal the completion of a program. END tells the computer that the program is completed. It is always good programming practice to use an END statement.

As your programs get more complex you may wish to review them before you begin execution. To look at your program, use the LIST command. LIST, which can only be used in the PROgram mode, displays programs beginning with the lowest line number.



Try listing this program:

Input

LIST

Display

10: INPUT "WORD?"

Use the  and  arrows to move through your program until you have reviewed the entire program. To review a line which contains more than 16 characters move the cursor to the extreme right of the display and the additional characters will appear on the screen. After checking your program, run it:

Input

RUN

HELP



Display

WORD?_

WORD IS 4. LTRS

>


This is the end of your program. Of course you may begin it again by entering RUN. However, this program would be a bit more entertaining if it presented more than one opportunity for input. We will now modify the program so it will keep running without entering RUN after each answer.

Return to the PRO mode and use the up or down arrows (or LIST) to reach line 40.

You may type 40 to Delete the entire line or use the ► to position the cursor over the E in END. Change line 40 so that it reads:

40: GOTO 10

Now RUN the modified program.

The GOTO statement causes the program to loop (keep repeating the same operation). Since you put no limit on the loop it will keep going forever (an "infinite" loop). To stop this program hit the BREAK () key.

Programming

When you have stopped a program using the **BRK** key, you can restart it using the CONT command. CONT stands for CONTINUE. With the CONT command the program will restart on the line which was being executed when the **BRK** key was pressed.

Example 4 – More Complex Programming

Although the **PC-1401/1402** has a factorial function we will use an example of the factorial computation in this section to explain more complex programming.

The following program computes N Factorial (N!). The program begins with 1 and computes N! up to the limit which you enter. Enter this program.

```
100 F = 1: WAIT 118
110 INPUT "LIMIT? "; L
120 FOR N = 1 TO L
130 F = F * N
140 PRINT N, F
150 NEXT N
160 END
```

Several new features are contained in this program. The WAIT verb in line 100 controls the length of time that displays are held before the program continues. The numbers and their factorials are displayed as they are computed. The time they appear on the display is set by the WAIT statement to approximately 2 seconds, instead of waiting for you to press **ENTER**.

Also in line 100, notice that there are two statements on the same line separated by a colon (:). **You may put as many statements as you wish on one line, separating each by a colon, up to the 80 character maximum including **ENTER**.** Multiple statement lines can make a program hard to read and modify, however, so it is good programming practice to use them only where the statements are very simple or there is some special reason to want the statements on one line.

Also in this program we have used the FOR verb in line 120 and the NEXT verb in line 150 to create a loop. In Example 3 you created an "infinite" loop which kept repeating the statements inside the loop until you pressed the **BRK** key. With this FOR/NEXT loop the **PC-1401/1402** adds 1 to N each time execution reaches the NEXT verb. It then tests to see if N is larger than the limit L. If N is less than or equal to L, execution returns to the top of the loop and the statements are executed again. If N is greater than L, execution continues with line 160 and the program stops.

You may use any numeric variable in a FOR/NEXT loop. You also do not have to start counting at 1 and you can add any amount at each step. See Chapter 8 for details.

We have labelled this program with line numbers starting with 100. Labelling programs with different line numbers allows you to have several programs in memory at one time. To RUN this program instead of the one at line 10 enter:

RUN 100

In addition to executing different programs by giving their starting line number, you can give programs a letter name and start them with the DEF key (see Chapter 6).

You will notice that while the program is running, the BUSY indicator is lit at those times that there is nothing on the display. RUN the program a few more times and try setting N at several different values.

Storing Programs in the PC-1401/1402's Memory

Programs remain in memory when you turn off the **PC-1401/1402**, or it undergoes an AUTO OFF. Even if you use the **BRK**, C-CE or CA keys the programs will remain.

Programs are lost from memory only when you perform the following actions:

- * You enter NEW before beginning programming.
- * You initialize the computer using the ALL RESET button.
- * You create a new program using the SAME LINE NUMBERS as a program already in memory.
- * You change the batteries.

This brief introduction to programming on the **PC-1401/1402** should serve to illustrate the exciting programming possibilities of your new computer.

CAUTION:

For programming the preprogrammed command/function keys can not be used within the quotation marks. Using the preprogrammed command/function keys within quotation marks will cause an error. Be sure to use the alphabetic keys within quotation marks as follows;

Examples:

1. 10 : INPUT "INPUT = "; A

Correct: 10 **SHIFT** **INPUT** **SHIFT** **"** **I** **N** **P** **U** **T** **=** **SHIFT** **"** **SHIFT** **:** **A** **ENTER**

Incorrect: 10 **SHIFT** **INPUT** **SHIFT** **"** **SHIFT** **INPUT** **=** **SHIFT** **"** **SHIFT** **:** **A** **ENTER** → ERROR 1

2. 20 : PRINT "SIN = "; A

Correct: 20 **SHIFT** **PRINT** **SHIFT** **"** **S** **I** **N** **=** **SHIFT** **"** **SHIFT** **:** **A** **ENTER**

Incorrect: 20 **SHIFT** **PRINT** **SHIFT** **"** **sin** **=** **SHIFT** **"** **SHIFT** **:** **A** **ENTER** → ERROR 1

CHAPTER 6 SHORTCUTS

The **PC-1401/1402** includes several features which make programming more convenient by reducing the number of keystrokes required to enter repetitive material.

One such feature is in the availability of abbreviations for verbs and commands (See Chapter 8)

This chapter discusses the additional feature which can eliminate unnecessary typing — the DEF key.

The DEF Key and Labelled Programs

Often you will want to store several different programs in the **PC-1401/1402's** memory at one time. (Remember that each must have unique line numbers). Normally, to start a program with a RUN or GOTO command, you need to remember the beginning line number of each program (see Chapter 8). But, there is an easier way! You can label each program with a letter and execute the program using only two keystrokes. This is how to label a program and execute it using DEF:

Note: Put a label on the first line of each program that you want to reference. The label consists of a single character in quotes, followed by a colon :

```
10: "A": PRINT "FIRST"
20: END
80: "B": PRINT "SECOND"
90: END
```

Any one of the following characters can be used: A, S, D, F, G, H, J, K, L, , , Z, X, C, V, B, N, M, and SPC. Notice that these are the keys in the last two rows of the alphabetic portion of the keyboard.

Note: To execute the program, instead of typing RUN 80 or GOTO 10, you need only press the **DEF** key and then the letter used as a label. In the above example, pressing **DEF** and then 'B' would cause 'SECOND' to appear on the display.

When DEF is used to execute a program, variables and mode settings are affected in the same way as when GOTO is used. See Chapter 8 for details.

Template

One template is provided with the **PC-1401/1402**. You can use this template to help you remember frequently used DEF key assignments. After you have labelled the programs, mark the template so you know what is associated with each key. You can then execute programs using the two-keystroke operation.

For example, if you have one group of programs which you often use at the same time, label the programs with letters and mark the template so that you can easily begin execution of any of the programs with two keystrokes.

Example:

AVER- AGE									

CHAPTER 7 USING THE CE-126P PRINTER/ CASSETTE INTERFACE

The optional **CE-126P** Printer/Cassette Interface allows you to add a printer and to connect a cassette recorder to your **SHARP PC-1401/1402** Computer.

The **CE-126P** features:

- * 24 character wide thermal printer.
- * Convenient paper feed and tear bar.
- * Simultaneous printing of calculations as desired. (Except the CAL mode)
- * Easy control of display or printer output in BASIC.
- * Built-in cassette interface with remote function.
- * Manual and program control of recorder for storing programs, data,
- * Dry battery operation for portability.

For connecting the **PC-1401/1402** to the **CE-126P**, refer to the instruction manual which is supplied with the **CE-126P**.

Using the Printer

If you are using the **PC-1401/1402** for manual calculation, you may use the **CE-126P** to simultaneously print your calculations.

CAUTION:

The result which is obtained by the direct calculation feature in manual calculation can not be printed. The calculation in CAL mode can not also be printed.

This is easily accomplished by pressing the **SHIFT** key and then the **ENTER** key (**P ↔ NP**) while in the RUN mode.

The printer indicator (a dash symbol) will appear just above the "PRINT" label in the lower right area of the display. After this, when you press the **ENTER** at the end of a calculation, the contents of the display will be printed on one line and the results will be printed on the next. For example:

Input

300 / 50 **ENTER**

Paper

300/50 6.

You may print output on the printer from within BASIC programs by using the LPRINT statement (see Chapter 8 for details). LPRINT can be used in the same form as the PRINT statement. The difference is that if you PRINT something to the display which is longer than 16 characters, there is no way for you to see the extra characters. With the LPRINT verb, the extra characters will be printed on a second and possibly a third line as is required.

Programs which have been written with PRINT can be converted to work with the printer by including a PRINT=LPRINT statement in the program (see Chapter 8 for details). All PRINT statements following this statement will act as if they were LPRINT statements. PRINT=PRINT will reset this condition to its normal state. This structure may also be included in a program in an IF statement allowing a choice of output at the time the program is used.

You may also list your programs on the printer with the LLIST command (see Chapter 8 for details). If used without line numbers LLIST will list all program lines currently in memory in their numerical order by line number. A line number range may also be given with LLIST to limit the lines which will be printed. When program lines are longer than 24 characters, two or more lines may be used to print one program line. The second and succeeding line will be indented four or six characters so that the line number will clearly identify each separate program line. (Line number, 1 to 999: four, over 999: six)

Caution:

- In case an error (ERROR code 8) occurs due to a paper misfeed, tear off the paper tape, and pull the remaining part of the paper tape completely out of the printer. Then press the **C-CE** key to clear the error condition.
- When the printer is exposed to strong external electrical noise, it may print numbers at random. If this happens, depress the **BRK** key to stop the printing. Turn the **CE-126P** power off and on, and then press the **C-CE** key.

Pressing the **C-CE** key will return the printer to its normal condition.

When the printer causes a paper misfeed or is exposed to strong external electrical noise while printing, it may not operate normally and only the symbol "BUSY" is displayed. If this happens, depress the **BRK** key to stop printing. (Release the paper misfeed.) Turn the **CE-126P** power off and on, and then press the **C-CE** key.

- When the **CE-126P** is not in use, turn off the printer switch to save the battery life.

Using the Cassette Interface

Using this cassette interface will allow you to store programs and data from the computer onto cassette tape (of course you'll also need a cassette recorder such as well sell for this pocket computer system; optional cassette tape recorder, model **CE-152**). Once on tape, you can load these programs and data back into the computer with a simple procedure.

Connecting the CE-126P to a Tape Recorder

Only three connections are necessary:

1. Connect red plug into the MICrophone jack on the cassette recorder.
2. Connect gray plug into the EARphone jack on the cassette recorder.
3. Connect the black plug into the REMote jack on the cassette recorder.

Cassette Tape Recorder

We recommend you to use an optional cassette tape recorder **CE-152** for your pocket computer system. The **CE-152** designed to match the **PC-1401/1402** records programs and data via the **CE-126P** cassette interface. Any recorded program can be retrieved and reloaded into the **PC-1401/1402**.

When you use any other cassette tape recorder than the **CE-152**:

The following is a description of the minimum tape recorder specifications necessary for interfacing with the **CE-126P**:

Item	Requirements
1. Recorder Type	Any tape recorder, standard cassette or micro-cassette recorder, may be used in accordance with the requirements outlined below.
2. Input Jack	The recorder should have a mini-jack input labeled "MIC". Never use the "AUX" jack.
3. Input Impedance	The input jack should be a low impedance input (200 ~ 1,000 OHM.)
4. Minimum Input Level	Below 3 mV or -50 dB.
5. Output jack	Should be a minijack labeled "EXT. (EXTer-nal speaker)", "MONITOR", "EAR (EAR-phone)" or equivalent.
6. Output impedance	Should be below 10 OHM.
7. Output level	Should be above 1V (practical maximum output above 100 mW)
8. Distortion	Should be within 15% within a range of 2 KHz through 4 KHz.
9. Wow and Flutter	0.3% maximum (W.R.M.S)
10. Other	Recorder motor should not fluctuate speed.

* In case the miniplug provided with the **CE-126P** is not compatible with the input/output jacks of your tape recorder special line conversion plugs are available on the market.

NOTE:

- Some tape recorders may reject connection due to different specifications. Or those tape recorders having distortion, increased noise, and power deterioration after long years of use may not show satisfactory results owing to change in their electrical characteristics.

Operating the Cassette Interface and Recorder

Recording (saving) onto magnetic tape

See Tape Notes.

1. Turn off the REMOTE switch on the CE-126P.
2. Enter a program or data into the Computer.
3. Load tape into the tape recorder.
Determine the position on the tape where you want to record the program.
 - When using a tape, be sure the tape moves past the clear leader (non-magnetic mylar material).
 - When using a tape already partially recorded, search for a location where no recording is.
4. Connect the Interface's red plug to the tape recorder's MIC jack and the black plug to the REM jack.
5. Turn on the REMOTE switch.
6. Simultaneously press record and play buttons on the tape recorder (to put it in record mode).
7. Input recording instructions (CSAVE statement, PRINT# statement), and press the **ENTER** key for execution.

First set the unit to "RUN" or "PRO" mode. Next push the following keys:

C **S** **A** **V** **E** **SHIFT** **"** file name **SHIFT** **"** **ENTER** .

(To write the contents of data memory onto tape, push as follows; Eg.

P **R** **I** **N** **T** **SHIFT** **#** **ENTER** .)

Eg. **C** **S** **A** **V** **E** **SHIFT** **"** **A** **A** **SHIFT** **"** **ENTER**

When you press the **ENTER** key, tape motion will begin, leaving about a 8-second non-signal blank. (Beep tone is recorded.) After that, the file name and its contents are recorded.

8. When the recording is complete, the PROMPT symbol (**>**) will be displayed and the tape recorder will automatically stop. Now you have your program on tape (it still is in the Pocket Computer also).

When data is to be automatically recorded by program execution (PRINT # statement, not manual operation), set up steps 1 thru 6 before executing the program.

To aid you in locating programs on tapes, use the tape counter on the recorder.

Collating the Computer and Tape Contents

See tape Notes.

After loading or transferring a program to or from tape, you can verify that the program on tape and program in the Pocket Computer are identical (and thus be sure that everything is OK before continuing your programming or execution of programs).

1. Turn off the REMOTE switch.
2. With cassette in the recorder, operate the tape motion controls to position tape at the point just before the appropriate file name to be checked.
3. Connect gray plug to EARphone and black plug to REMote jacks.
4. Turn on the REMOTE switch.
5. Press PLAY button of recorder.
6. Input a CLOAD? statement and start execution with **ENTER** key. Do this as follows: Set unit to "RUN" or "PRO" mode.

Enter the following key sequence —

↓ The file name which
you used previously.

C **L** **O** **A** **D** **SHIFT** **?** **SHIFT** **"** **A** **A** **SHIFT** **"** **ENTER**

The Pocket Computer will automatically search for the specified file name and will compare the contents on tape with the contents in memory.

During the collation, the mark " * " is shown at the right most digit of the display. The mark " * " will disappear when the collation is completed. While a file name is being retrieved, no " * " mark will be displayed as the collation is not started yet.

(The same occurs when the first program is read without a file name.)

If the programs are verified as being identical, a PROMPT symbol (>) will be displayed on the Pocket Computer.

If the programs differ, execution will be interrupted and an Error code 8 will be displayed. If this occurs, try again.

Loading from a magnetic tape

See Tape Notes.

To load, transfer, or read out programs and data from magnetic tape into the Pocket Computer, use the following procedure.

1. Turn off the REMOTE switch.
2. Load tape in the tape recorder. Position tape just before the portion to be read out.

3. Connect the gray plug to the EAR jack on the tape recorder, and the black plug to the REM jack.

[In using a tape recorder having no REM terminal, press the PAUSE button to make a temporary stop.]

4. Turn on the REMOTE switch.
5. Push the PLAY button on the tape recorder (to put unit in playback mode).

Set the VOLUME control to middle or maximum.

Set Tone to maximum treble.

6. Input transfer instructions (CLOAD statement, INPUT # statement), and press **ENTER** key for execution.

Put the unit into "RUN" mode. Then push the following keys; **C** **L** **O** **A** **D** **SHIFT** **"** file name **SHIFT** **"** **ENTER**. (To load the contents of the data memory, push as follows; Eg. **I** **N** **P** **U** **T** **SHIFT** **#** **ENTER**.)

Eg. **C** **L** **O** **A** **D** **SHIFT** **"** **A** **A** **SHIFT** **"** **ENTER**

The specified file name will be automatically searched for and its contents will be transferred into the Pocket Computer.

The mark "*" appears while loading the designated CSAVED program from the tape to the computer's memory.

(The same occurs when the first program is read without a file name,)

The mark "*" disappears when the load is performed completely.

7. When the program has been transferred the Computer will automatically stop the tape motion and display the PROMPT (>) symbol.

To transfer data (INPUT # statement) in the course of a program, set up steps 1 thru 5 prior to executing the program.

Notes:

- If an error occurs (error code "8" is displayed), start over from the beginning. If the error continues, adjust volume up or down slightly.
- If the error code is not displayed but tape motion continues (while the Pocket Computer displays the symbol "BUSY"), transferring is improper. Press **ON** **BRK** key (to "break") to stop the tape. Repeat steps.
- If the error remains or the tape continues to run after several attempts to correct the problem, try cleaning and demagnetizing the Recorder's tape head.

CAUTION:

After storing a program on a tape, be sure to verify the saved contents by using the statement and the procedure described on page 101. If an error occurs as a result of repeated verification operations, try the following calculation after placing your computer in the RUN mode.

```
CLEAR 
      (3534 - MEM)/120 
      and (3535 - MEM)/120 
```

If the result of either of these calculations is an integer value (1, 2, 3, ..., 29), reduce one line (combine two lines into one) from, or add one line (divide one line into two) to your program, then store it again.

```
(Example)  10 : A = 0, B = 0          10: A = 0, B = 0
            200 : NEXT I              200: NEXT I: END
            210 : END
```

Now verify the stored contents again.

If the error still occurs, relocate the program to another section of the tape, or use another tape. Otherwise there may be some problems with your tape recorder. Check your tape recorder.

Tape notes

- 1) For any transfer or collation, use the tape recorder that was used for recording. If the tape recorder for transfer or collation is different from that used for recording, no transfer or collation may be possible.
- 2) Always use only the highest quality tape for programs and data storage (economy grade audio type tape may not provide the proper characteristics for digital recordings).
- 3) Keep the tape heads and tape handling parts clean—use a cassette cleaner tape to keep everything clean.
- 4) Volume setting — set to middle or maximum level.
Volume level can be very important when reading in data from the recorder; make slight adjustments as required to obtain error-free data transfer. A slight adjustment either up or down may result in perfect results every time.
- 5) Be sure all connections between the pocket computer and cassette interface are secure. And be sure the connections between interface and recorder are secure and dirt-free.
- 6) If problems occur when using AC power for the **CE-126P** and/or the recorder, use battery power instead (sometimes the AC power connection also adds some “hum” to the signal which upsets proper digital recordings).
 - To connect the AC adaptor to the **CE-126P**, turn the **CE-126P** power off and then connect the adaptor to the **CE-126P**.
- 7) Tone control — set to maximum treble.
- 8) When recording programs or data on the used tape, erase the portion before writing and execute the recording command. (Make sure that the previous program is completely erased without any portion remaining.)

CHAPTER 8

BASIC REFERENCE

The following chapter is divided into three sections:

- Commands:** Instructions which are used outside a program to change the working environment, perform utilities, or control programs.
- Verbs:** Action words used in programs to construct BASIC statements.
- Functions:** Special operators used in BASIC programs to change one variable into another.

Commands and verbs are arranged alphabetically. Each entry is on a separate page for easy reference. The contents of each section is shown in the tables below so that you can quickly identify the category to which an operator belongs. Functions are grouped according to four categories and arranged alphabetically within category.

Commands

Program Control

CONT
GOTO*
NEW
RUN

Cassette Control

CLOAD
CLOAD?
CSAVE
INPUT #*
PRINT #*

Debugging

LIST
LLIST
TROFF*
TRON*

Variables Control

CLEAR
DIM*

Angle Mode Control

DEGREE*
GRAD*
RADIAN*

Other

BEEP*
PASS
RANDOM*
USING*
WAIT*

*These commands are also BASIC verbs. Their effect as commands is identical to their effect as verbs so they are not described in the command reference section. See the verb reference section for more information.

VerbsControl and Branching

END
FOR ... TO ... STEP
GOSUB
GOTO
IF ... THEN
NEXT
ON ... GOSUB
ON ... GOTO
RETURN
STOP

Assignment and Declaration

CLEAR
DIM
LET

Input and Output

AREAD
CSAVE
DATA
INPUT
INPUT #
LPRINT
PAUSE
PRINT
PRINT #
USING
READ
RESTORE
WAIT

Other

BEEP
DEGREE
GRAD
RADIAN
RANDOM
REM
TROFF
TRON

FunctionsPseudovariables

INKEY\$
MEM
PI

String Functions

ASC
CHR\$
LEFT\$
LEN
MID\$
RIGHT\$
STR\$
VAL

Numeric Functions

ABS
ACS
AHC
AHS
AHT
ASN
ATN
COS
CUR
DEG
DMS
EXP
FACT
HCS
HSN
HTN
INT
LN
LOG
POL
RCP
REC
RND
ROT
SGN
SIN
SQR
SQU
TAN
TEN

COMMANDS

- 1 CLOAD
- 2 CLOAD "filename"

Abbreviations: CLO., CLOA.

See also: CLOAD?, PASS

Purpose

The CLOAD command is used to load a program saved on cassette tape. It can be used with the optional **CE-126P** Printer/Cassette Interface and Cassette Recorder.

Use

The first form of the CLOAD command clears the memory of existing programs and loads the first program stored on the tape, starting at the current position.

The second form of the CLOAD command clears the memory, searches the tape for the program whose name is given by "filename", and loads the program.

If the **PC-1401/1402** is in PROgram or RUN mode, program memory is loaded from the tape.

Examples

CLOAD Loads the first program from the tape.

CLOAD "PRO3" Searches the tape for the program named 'PRO3' and loads it.

Notes:

1. If the designated file name is not retrieved, the computer will continue to search the file name even after the tape reaches the end. In this case, stop the retrieval function by pressing the **ON** **BRK** key. This applies to CLOAD? and INPUT# commands to be described later.
2. If an error occurs during CLOAD command (to be described later) execution, the program stored in the computer will be invalid.

- 1 CLOAD?
- 2 CLOAD? "filename"

Abbreviations: CLO.?, CLOA.?

See also: CLOAD, PASS

Purpose

The CLOAD? command is used to compare a program saved on cassette tape with one stored in memory. It can be used with the optional **CE-126P** Printer/Cassette Interface and Cassette Recorder.

Use

The first form of the CLOAD? command compares the program stored in memory with the first program stored on the tape, starting at the current position.

The second form of the CLOAD? command searches the tape for the program whose name is given by "filename" and then compares it to the program stored in memory.

Examples

CLOAD? Compares the first program from the tape with the one in memory.

CLOAD? "PRO3" Searches the tape for the program named 'PRO3' and compares it to the one stored in memory.

Commands

CONT

1 CONT

Abbreviations: C., CO., CON.

See also: RUN, STOP verb

Purpose

The CONT command is used to continue a program which has been temporarily halted.

Use

When the STOP verb is used to halt a program during execution, the program can be continued by entering CONT in response to the prompt.

When a program is halted using the **BRK** key, the program can be continued by entering CONT in response to the prompt.

Examples

CONT Continues an interrupted program execution.

- 1 **CSAVE**
- 2 **CSAVE** "filename"
- 3 **CSAVE** , "password"
- 4 **CSAVE** "filename", "password"

Abbreviations: CS., CSA., CSAV.

See also: CLOAD, CLOAD?, PASS

Purpose

The CSAVE command is used to save a program to cassette tape. It can be used with the optional **CE-126P** Printer/Cassette Interface and Cassette Recorder.

Use

The first form of the CSAVE command writes all of the programs in memory on to the cassette tape without a specified file name.

The second form of the CSAVE command writes all of the programs in memory on to the cassette tape and assigns the indicated file name.

The third form of the CSAVE command writes all of the programs in memory on to the cassette tape without a specified file name and assigns the indicated password. Programs saved with a password may be loaded by anyone, but only someone who knows the password can list or modify the programs. (See discussion under PASS command).

The fourth form of the CSAVE command writes all of the programs in memory on to the cassette tape and assign them the indicated file name and password.

If the **PC-1401/1402** is in PROgram or RUN mode, program memory is loaded to the tape.

Examples

CSAVE "PRO3", "SECRET" Saves the programs now in memory on to the tape under the name 'PRO3', protected with the password 'SECRET'.

Commands

GOTO

1 **GOTO** expression

Abbreviations: G., GO., GOT.

See also: RUN

Purpose

The GOTO command is used to start execution of a program.

Use

The GOTO command can be used in place of the RUN command to start program execution at the line number specified by the expression.

GOTO differs from RUN in five respects:

- 1) The value of the interval for WAIT is not reset.
- 2) The display format established by USING statements is not cleared.
- 3) Variables and arrays are preserved.
- 4) PRINT = LPRINT status is not reset.
- 5) The pointer for READ is not reset.

Execution of a program with GOTO is identical to execution with the **DEF** key.

Examples

GOTO 100 Begins execution of the program at line 100.

- 1 LIST
- 2 LIST expression

Abbreviations: L., LI., LIS.

See also: LLIST

Purpose

The LIST command is used to display a program.

Use

The LIST command may only be used in the PROgram mode. The first form of the LIST command displays the statement with the lowest line number.

The second form displays the statement with the nearest line number equal to or greater than the value of the expression. The Up Arrow and Down Arrow keys may then be used to examine the program.

Examples

LIST 100 Displays line number 100.

Commands

LLIST

- 1 **LLIST**
- 2 **LLIST** expression
- 3 **LLIST** expression 1 , expression 2
- 4 **LLIST** expression ,
- 5 **LLIST**, expression

Abbreviations: LL., LLI., LLIS.

See also: LIST

Purpose

The LLIST command is used for printing a program on the optional CE-126P Printer/Cassette Interface.

Use

The LLIST command may be used in the PROgram or RUN mode.

The first form prints all of the programs in memory.

The second form prints only the program line whose line number is given by expression.

The third form prints the statements from the line number with the nearest line equal to or greater than the value of expression 1 to the nearest line equal to or greater than the value of expression 2. There must be at least two lines between the two numbers.

The forth form prints program lines beginning with the line whose number is given by the expression.

The fifth form prints all program lines up to, and including, the line whose number is given by the expression.

Examples

LLIST 100, 200 Lists the statements between line numbers 100 and 200.

1 NEW

Abbreviations: none

Purpose

The NEW command is used to clear existing program.

Use

When used in the PROgram mode the NEW command clears all programs and data which are currently in memory.

The NEW command is not defined in the RUN mode and will result in an ERROR 9.

Examples

NEW Clears program or data.

Commands

PASS

1 PASS "character string"

Abbreviations: PA., PAS.

See also: CSAVE, CLOAD

Purpose

The PASS command is used to set and cancel passwords.

Use

Passwords are used to protect programs from inspection or modification by other users. A password consists of a character string which is **no more than seven characters long**. The seven characters must be alphabetic or one of the following special symbols: ! # \$ % & () * + - / , . : ; < = > ? @ $\sqrt{\quad}$ π ^

Once a PASS command has been given the programs in memory are protected. A password protected program cannot be examined or modified in memory. It cannot be output to tape or listed with LIST or LLIST, nor is it possible to add or delete program lines. If several programs are in memory and PASS is entered, all programs in memory are protected. The only way to remove this protection is to execute another PASS command with the same password or to enter NEW (which erases the programs).

Examples

PASS "SECRET" Establishes the password 'SECRET' for all programs in memory.

- 1 **RUN**
- 2 **RUN** line number

Abbreviations: R., RU.

See also: GOTO

Purpose

The RUN command is used to execute a program in memory.

Use

The first form of the RUN command executes a program beginning with the lowest numbered statement in memory.

The second form of the RUN command executes a program beginning with the specified line number.

RUN differs from GOTO in five respects:

- 1) The value of the interval for WAIT is reset.
- 2) The display format established by USING statements is cleared.
- 3) Variables and arrays other than the fixed variables are cleared.
- 4) PRINT = PRINT status is set.
- 5) The pointer for READ is reset to the beginning DATA statement.

Execution of a program with GOTO is identical to execution with the DEF key. In all three forms of program execution FOR/NEXT and GOSUB nesting is cleared.

Examples

RUN 100

Executes the program which begins at line number 100.

VERBS

1 AREAD variable name

Abbreviations: A., AR., ARE., AREA.

See also: INPUT verb and discussion of the use of the DEF key in Chapter 6

Purpose

The AREAD verb is used to read in a single value to a program which is started using the **DEF** key.

Use

When a program is labelled with a letter, so that it can be started using the **DEF** key, the AREAD verb can be used to enter a single starting value without the use of the INPUT verb. The AREAD verb must appear on the first line of the program following the label. If it appears elsewhere in the program, it will be ignored. Either a numeric or string variable may be used, but only one can be used per program.

To use the AREAD verb type the desired value in the RUN mode, press the **DEF** key, followed by the letter which identifies the program. If a string variable is being used, it is not necessary to enclose the entered string in quotes.

Examples

```
10 "X": AREAD N
20 PRINT N ^ 2
30 END
```

Entering "7 **DEF** X" will produce a display of "49".

Notes:

1. When the display indicates PROMPT (">") at the start of program execution, the designated variable is cleared.
2. When the contents is displayed by PRINT verb at the start of program execution, the following is stored:

- When the display indicates PRINT numeric expression, numeric expression or PRINT "String", "String", the contents on the right of the display are stored.

Example: When the program below is executed;
10 "A" : PRINT "ABC", "DEFG"
20 "S" : AREAD A\$: PRINT A\$
RUN mode
DEF A → ABC DEFG
DEF S → DEFG

- When the display indicates PRINT Numeric expression; Numeric expression; Numeric expression..., the contents displayed first (on the extreme left) are stored.
- When the display indicates PRINT "String"; "String"; "String"..., the "String" designated last are stored.

Verbs

BEEP

1 BEEP expression

Abbreviations: B., BE., BEE.

Purpose

The BEEP verb is used to produce an audible tone.

Use

The BEEP verb causes the **PC-1401/1402** to emit one or more audible tones at 4 kHz. The number of beeps is determined by the expression, which must be numeric. (Positive number less than 9.999999999E+99) The expression is evaluated, but only the integer part is used to determine the number of beeps.

BEEP may also be used as a command using numeric literals and predefined variables. In this case the beeps occur immediately after the **ENTER** key is pressed.

Examples

10 A = 5 : B\$ = "9"

20 BEEP 3 Produces 3 beeps.

30 BEEP A Produces 5 beeps.

40 BEEP (A+4)/2 Produces 4 beeps.

50 BEEP B\$ This is illegal and will produce an ERROR 9 message.

60 BEEP -4 Produces no beeps, but does not produce an error message.

1 CLEAR

Abbreviations: CL., CLE., CLEA.

See also: DIM

Purpose

The CLEAR verb is used to erase all variables which have been used in the program and to reset all preallocated variables to zero or NUL.

Use

The CLEAR verb recovers space which is being used to store variables. This might be done when the variables used in the first part of a program are not required in the second part and available space is limited. CLEAR may also be used at the beginning of a program when several programs are resident in memory and you want to clear out the space used by execution of prior programs.

CLEAR does not free up the space used by the variables A – Z, A\$ – Z\$, or A(1) – A(26) (without DIM declaration) since they are permanently assigned (see Chapter 4). CLEAR does reset numeric variables to zero and string variables to NUL.

Examples

10 A = 5 : DIM C(5)

20 CLEAR Frees up the space assigned to C() and resets A to zero.

1 **DATA** expression list

Where: expression list is: expression
or: expression , expression list

Abbreviations: DA., DAT.

See also: READ, RESTORE

Purpose

The DATA verb is used to provide values for use by the READ verb.

Use

When assigning initial values to an array, it is convenient to list the values in a DATA statement and use a READ statement in a FOR . . . NEXT loop to load the values into the array. When the first READ is executed, the first value in the first DATA statement is returned. Succeeding READs use succeeding values in the sequential order in which they appear in the program, regardless of how many values are listed in each DATA statement or how many DATA statements are used.

DATA statements have no effect if encountered in the course of regular execution of the program, so they can be inserted wherever it seems appropriate. Many programmers like to include them immediately following the READ which uses them. If desired, the values in a DATA statement can be read a second time by using the RESTORE statement.

Examples

10 DIM B(10)	Sets up an array.
20 WAIT 128	
30 FOR I = 1 TO 10	
40 READ B(I)	Loads the values from the DATA statement into B ()
50 PRINT B(I)	B(1) will be 10, B(2) will be 20, B(3) will be 30,
60 NEXT I	etc.
70 DATA 10, 20, 30, 40, 50, 60	
80 DATA 70, 80, 90, 100	
90 END	

1 DEGREE

Abbreviations: DE., DEG., DEGR., DEGRE.

See also: GRAD and RADIAN

Purpose

The DEGREE verb is used to change the form of angular values to decimal degrees.

Use

The **PC-1401/1402** has three forms for representing angular values — decimal degrees, radians and gradient. These forms are used in specifying the arguments to the SIN, COS, and TAN functions and in returning the results from the ASN, ACS, and ATN functions.

The DEGREE function changes the form for all angular values to decimal degree form until a GRAD or RADIAN verb is used. The DMS and DEG functions can be used to convert decimal degrees to degree, minute, second form and vice versa. The REC and POL functions can be used to convert polar coordinates to rectangular and vice versa.

Examples

10 DEGREE

20 X = ASN 1

30 PRINT X

X now has a value of 90, i.e. 90 degrees, the Arcsine of 1.

1 DIM dim list

Where: <u>dim list</u>	is: <u>dimension spec.</u> or: <u>dimension spec.</u> , <u>dim list</u>
and: <u>dimension spec.</u>	is: <u>numeric dim spec.</u> or: <u>string dim spec.</u>
and: <u>numeric dim spec</u>	is: <u>numeric name (size)</u>
and: <u>string dim spec</u>	is: <u>string name (dims)</u> or: <u>string name (dims) * len</u>
and: <u>numeric name</u>	is: valid numeric variable name
and: <u>string name</u>	is: valid string variable name
and: <u>dims</u>	is: <u>size</u> or: <u>size</u> , <u>size</u>
and: <u>size</u>	is: number of elements
and: <u>len</u>	is: length of each string in a string array

Abbreviations: D., DI.

Purpose

The DIM verb is used to reserve space for numeric and string array variables.

Use

Except for the array of the form; A(), A\$(), two-character(), and two-character\$(), a DIM verb must be used to reserve space for any array variable.

The maximum number of dimensions in any array is two; the maximum size of any one dimension is 255. In addition to the number of elements specified in the dimension statement, one additional "zeroeth" element is reserved. For example, DIM B(3) reserves B(0), B(1), B(2), and B(3). In two dimensional arrays there is an extra "zeroeth" row and column.

In string arrays one specifies the size of each string element in addition to the number of elements. For example, DIM B\$(3)*12 reserves space for 4 strings which are each a maximum of 12 characters long. If the length is not specified each string can contain a maximum of 16 characters.

When a numeric array is dimensioned, all values are initially set to zero; in a string array the values are set to NUL.

Refer to array variables on page 74.

For the array A and A\$ DIM declaration, refer to the paragraph discussing variables.

Examples

- 10 DIM B(10) Reserves space for a numeric array with 11 elements.
- 20 DIM C\$(4,4)*10 Reserves space for a two dimensional string array with 5 rows and 5 columns; each string will be a maximum of 10 characters.

Verbs
END

1 END

Abbreviations: E., EN.

Purpose

The END verb is used to signal the end of a program.

Use

When multiple programs are loaded into memory at the same time a mark must be included to indicate where each program ends so that execution does not continue from one program to another. This is done by including an END verb as the last statement in the program.

Examples

10 PRINT "HELLO"

20 END

30 PRINT "GOODBYE"

40 END

With these programs in memory a 'RUN 10' prints 'HELLO', but not 'GOODBYE'. 'RUN 30' prints 'GOODBYE'.

```

1  FOR numeric variable = expression 1   TO expression 2
2  FOR numeric variable = expression 1   TO expression 2
    STEP expression 3

```

Abbreviations: F. and FO.; STE.

See also: NEXT

Purpose

The FOR verb is used in combination with the NEXT verb to repeat a series of operations a specified number of times.

Use

The FOR and the NEXT verbs are used in pairs to enclose a group of statements which are to be repeated. The first time this group of statements is executed the loop variable (the variable named immediately following the FOR) has the value of expression 1.

When execution reaches the NEXT verb the loop variable is increased by the step size and then this value is tested against expression 2. If the value of the loop variable is less than or equal to expression 2, the enclosed group of statements is executed again, starting with the statement following the FOR. In the first form the step size is 1; in the second form the step size is given by expression 3. If the value of the loop variable is greater than expression 2, execution continues with the statement which immediately follows the NEXT. Because the comparison is made at the end, the statements within a FOR/NEXT pair are always executed at least once.

Expression 1, expression 2, and expression 3 must be in the range of $-9.999999999\text{E}99 \sim 9.999999999\text{E}99$. When expression 3 is in zero, FOR/NEXT loop will be infinite.

The loop variable may be used within the group of statements, for example as an index to an array, but care should be taken in changing the value of the loop variable.

Programs should be written so that they never jump from outside a FOR/NEXT pair to a statement within a FOR/NEXT pair. Similarly, programs must never leave a FOR/NEXT pair by jumping out. Always exit a FOR/NEXT loop via the NEXT statement. To do this, set the loop variable to a value higher than expression 2.

Verbs FOR

The group of statements enclosed by a FOR/NEXT pair can include another pair of FOR/NEXT statements which use a different loop variable as long as the enclosed pair is completely enclosed; i.e., if a FOR statement is included in the group, the matching NEXT must also be included. FOR/NEXT pairs may be "nested" up to five levels deep.

Examples

```
10 FOR I = 1 TO 5  
20 PRINT I  
30 NEXT I
```



This group of statements prints the numbers 1, 2, 3, 4, 5.

```
40 FOR N = 10 TO 0 STEP -1  
50 PRINT N  
60 NEXT N
```



This group of statements counts down 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0.

```
70 FOR N = 1 TO 10  
80 X = 1  
90 FOR F = 1 TO N  
100 X = X * F  
110 NEXT F  
120 PRINT X  
130 NEXT N
```



This group of statements computes and prints N factorial for the numbers from 1 to 10.

1 GOSUB expression

Abbreviations: GOS., GOSU.

See also: GOTO, ON ... GOSUB, ON ... GOTO, RETURN

Purpose

The GOSUB verb is used to execute a BASIC subroutine.

Use

When you wish to execute the same group of statements several times in the course of a program or use a previously written set of statements in several programs, it is convenient to use the BASIC capability for subroutines using the GOSUB and RETURN verbs.

The group of statements is included in the program at some location where they are not reached in the normal sequence of execution. A frequent location is following the END statement which marks the end of the main program. At those locations in the main body of the program — where subroutines are to be executed, include a GOSUB statement with an expression which indicates the starting line number of the subroutine. The last line of the subroutine must be a RETURN. When GOSUB is executed, the **PC-1401/1402** transfers control to the indicated line number and processes the statements until a RETURN is reached. Control is then transferred back to the statement following the GOSUB.

A subroutine may include a GOSUB. Subroutines may be "nested" in this fashion up to 10 levels deep.

The expression in a GOSUB statement may not include a comma, e.g., 'A(1, 2)' cannot be used. Since there is an ON ... GOSUB structure for choosing different subroutines at given locations in the program, the expression usually consists of just the desired line number. When a numeric expression is used it must evaluate to a valid line number, i.e., 1 to 65279, or an ERROR 4 will occur.

Examples

```
10 GOSUB 100
20 END
100 PRINT "HELLO"
110 RETURN
```

When this program is run it prints the word 'HELLO' one time.

Verbs

GOTO

1 GOTO expression

Abbreviations: G., GO., GOT.

See also: GOSUB, ON ... GOSUB, ON ... GOTO

Purpose

The GOTO verb is used to transfer control to a specified line number.

Use

The GOTO verb transfers control from one location in a BASIC program to another location. Unlike the GOSUB verb, GOTO does not "remember" the location from which the transfer occurred.

The expression in a GOTO statement may not include a comma, e.g., 'A(1, 2)' cannot be used. Since there is an ON ... GOTO structure for choosing different destinations at given locations in the program, the expression usually consists of just the desired line number. When a numeric expression is used, it must evaluate to a valid line number, i.e., 1 to 65279, or an ERROR 4 will occur.

Well designed programs usually flow simply from beginning to end, except for sub-routines executed during the program. Therefore, the principal use of the GOTO verb is as a part of an IF ... THEN statement.

Examples

```
10 INPUT A$
20 IF A$ = "Y" THEN GOTO 50
30 PRINT "NO"
40 GOTO 60
50 PRINT "YES"
60 END
```

This program prints 'YES' if a 'Y' is entered and prints 'NO' if anything else is entered.

1 GRAD

Abbreviations: GR., GRA.

See also: DEGREE and RADIAN

Purpose

The GRAD verb is used to change the form of angular values to gradient form.

Use

The **PC-1401/1402** has three forms for representing angular values — decimal degrees, radians, and gradient. These forms are used in specifying the arguments to the SIN, COS, and TAN functions and in returning the results from the ASN, ACS, and ATN functions.

The GRAD function changes the form for all angular values to gradient form until a DEGREE or RADIAN verb is used. Gradient form represents angular measurement in terms of percent gradient, i.e., a 45° angle is a 50^g gradient.

Examples

```
10 GRAD
```

```
20 X = ASN 1
```

```
30 PRINT X
```

X now has a value of 100, i.e., a 100^g gradient, the Arcsine of 1.

Verbs

IF . . . THEN

- 1 IF condition THEN statement
- 2 IF condition statement

Abbreviations: none for IF, T., TH., THE.

Purpose

The IF . . . THEN verb pair is used to execute or not execute a statement depending on conditions at the time the program is run.

Use

In the normal running of a BASIC programs, statements are executed in the sequence in which they occur. The IF . . . THEN verb pair allows decisions to be made during execution so that a given statement is executed only when desired. When the condition part of the IF statement is true, the statement is executed; when it is False, the statement is skipped.

The condition part of the IF statement can be any relational expression as described in Chapter 4. It is also possible to use a numeric expression as a condition, although the intent of the statement will be less clear. Any expression which evaluates to zero or a negative number is considered False; any which evaluates to a positive number is considered True.

The statement which follows the THEN may be any BASIC statement, including another IF . . . THEN. If it is a LET statement, the LET verb itself must appear. Unless the statement is an END, GOTO, or ON . . . GOTO, the statement following the IF . . . THEN statement is the next one executed regardless of whether the condition is True.

The two forms of the IF statement are identical in action, but the first form is clearer.

Examples

```
10 INPUT "CONTINUE? "; A$  
20 IF A$ = "YES" THEN GOTO 10  
30 IF A$ = "NO" THEN GOTO 60  
40 PRINT "YES OR NO, PLEASE"  
50 GOTO 10  
60 END
```

This program continues to ask 'CONTINUE?' as long as 'YES' is entered; it stops if 'NO' is entered, and complains otherwise.

1 INPUT input list

Where: <u>input list</u>	is: <u>input group</u>
	or: <u>input group</u> , <u>input list</u>
and: <u>input group</u>	is: <u>var list</u>
	or: <u>prompt</u> , <u>var list</u>
	or: <u>prompt</u> ; <u>var list</u>
and: <u>var list</u>	is: <u>variable</u>
	or: <u>variable</u> , <u>var list</u>
and: <u>prompt</u>	is: any string constant

Abbreviations: I., IN., INP., INPU.

See also: INPUT #, READ

Purpose

The INPUT verb is used to enter one or more values from the keyboard.

Use

When you want to enter different values each time a program is run, use the INPUT verb to enter these values from the keyboard.

In its simplest form the INPUT statement does not include a prompt string, instead a question mark is displayed on the left edge of the display. A value is then entered, followed by the **ENTER** key. This value is assigned to the first variable in the list. If other variables are included in the same INPUT statement, this process is repeated until the list is exhausted.

If a prompt is included in the INPUT statement, the process is exactly the same except that, instead of the question mark, the prompt string is displayed at the left edge of the display. If the prompt string is followed by a semicolon, the cursor is positioned immediately following the prompt. If the prompt is followed by a comma, the prompt is displayed, then when a key is pressed the display is cleared and the first character of the input is displayed at the left edge.

When a prompt is specified and there is more than one variable in the list following it, the second and succeeding variables are prompted with the question mark. If a second prompt is included in the list, it is displayed for the variable which immediately follows it.

Verbs INPUT

If the **ENTER** key is pressed and no input is provided, the variable retains the value it had before the INPUT statement.

Examples

10 INPUT A

Clears the display and puts a question mark at the left edge.

20 INPUT "A = "; A

Displays 'A =' and waits for input data.

30 INPUT "A = ", A

Displays 'A ='.

When data is input 'A =' disappears and the data is displayed starting at left edge.

40 INPUT "X = ? ";X, "Y = ? "; Y

Displays 'X = ?' and waits for first input.

After **ENTER** is pressed, display is cleared and 'Y = ?' is displayed at left edge.

```
1 INPUT # var list  
2 INPUT # "filename"; var list
```

Where: var list is: variable
or: variable , var list

Abbreviations: I. #, IN. #, INP. #, INPU. #

See also: INPUT, PRINT #, READ

Purpose

The INPUT # verb is used to enter values from the cassette tape.

Use and Examples

The following variable types can be specified in the INPUT # statement:

- (1) Fixed variables — A, B, C, A(7), D*, A(20)*, etc.
- (2) Simple variables — AA, B3, CP\$, etc.
- (3) Array variables — S(*), HP(*), K\$(*), etc.

1) Transferring data to fixed variables

To transfer data from tape to fixed variables, specify the variable names in the INPUT # statement.

```
INPUT # "DATA 1"; A, B, X, Y
```

This statement transfers data from the cassette file named "DATA 1" to the variables A, B, X, and Y in that order.

To fill all the available fixed variables and, if defined, extended variables (A(27) and beyond) with data transferred from tape, specify the first variable with an asterisk (*) subscripted to it.

```
INPUT # "D-2"; D *
```

This statement transfers the contents of the tape file "D-2" to variables D through Z and to A(27) and beyond.

```
INPUT # A(10) * (without DIM declaration)
```

This statement transfers the data of the first file found after the tape was started, to the variables A(10) and beyond (to J through Z and A(27) and beyond).

Verbs

INPUT

- Note 1. If an array named A is already defined by the DIM statement, it is not possible to define subscripted fixed variables in the form of A(). Also no data transfer to variables A(27) and beyond will occur.
- Note 2. Data transfer to fixed variables and extended variables (A(27) and beyond) will continue until the end of the source data file on the tape is reached or the computer's memory becomes full.

2) Data transfer to simple variables

Data in a tape file can be transferred to simple variables by specifying the desired simple variable names in the INPUT # statement.

```
INPUT # "DM-1"; AB, Y1, XY$
```

This statement transfers data from the tape file named "DM-1" to simple variables AB, Y1, and XY\$.

- Note 1. Numeric data must be transferred to numeric simple variables, and character data must be simple character variables. Cross-transfer is not allowed.
- Note 2. Locations for simple variables must be set aside in the program data area before the INPUT # statement is executed. If not, an error will result. Use assignment statements to reserve the locations for simple variables.

```
AA = 0 
```

```
B1$ = "A" 
```

Use appropriate numeric values or characters in assignment statements to reserve locations for variables.

```
INPUT # AA, B1$ 
```

3) Data transfer to array variables

To transfer data from a tape file to array variables, specify the array name in the INPUT # statement in the form of array name (*).

```
50 DIM B(5)
```

```
60 INPUT # "DS-4"; B(*)
```

This statement transfers data from the tape file named "DS-4" to the variables (B(0) through B(5)) in array B.

- Note 1. Numeric data must be transferred to numeric array variables with the same length as that of the data, character data must be transferred to character array variables with the same length as that of the data. If this rule is not observed, an error will result.
- Note 2. Locations for array variables must be set aside in the program data area before the INPUT # statement is executed. If not, an error will result. Use the DIM statement to define the array in advance.

— CAUTION —

If the number of variables specified in the INPUT# statement does not agree with the amount of data recorded on the tape, the following will happen:

- * If the number of pieces of data recorded on the tape file (to be transferred) is greater than the number of specified variables, data transfer will be performed to the last variable, and the remaining data will be ignored.
- * If the number of pieces of data recorded in the tape file (to be transferred) is smaller than the number of specified variables, all the file data will be transferred to the variables to the end of the file, and the remaining variables will maintain their previous contents.
In this case, however, the computer will continue to wait for data transfer from the tape. To halt this state, you should operate the **ON**
BRK key.
- * If the INPUT# statement is executed with no variable name specified in it, an error (ERROR 1) will result.

Verbs

LET

- 1 **LET** variable = expression
- 2 variable = expression

Abbreviations: LE.

Purpose

The LET verb is used to assign a value to a variable.

Use

The LET verb assigns the value of the expression to the designated variable. The type of the expression must match that of the variable, i.e. only numeric expressions can be assigned to numeric variables and only string expressions can be assigned to string variables. In order to convert from one type to the other, one of the explicit type conversion functions, STR\$ or VAL, must be used.

The LET verb may be omitted in all LET statements except those which appear in the THEN clause of an IF ... THEN statement. In this one case the LET verb must be used.

Examples

10 I = 10

Assigns the value 10 to I.

20 A = 5*I

Assigns the value 50 to A.

30 X\$ = STR\$ (A)

Assigns the value '50' to X\$.

40 IF I >= 10 THEN LET Y\$=X\$+" .00"

Assigns the value '50.00' to Y\$.

- 1 **LPRINT** print expr
- 2 **LPRINT** print expr , print expr
- 3 **LPRINT** print list

Where: <u>print list</u>	is: <u>print expr</u>
	or: <u>print expr</u> ; <u>print list</u>
and: <u>print expr</u>	is: <u>expression</u>
	or: <u>USING clause</u> ; <u>expression</u>

The USING clause is described separately under USING

Abbreviations: LP., LPR., LPRI., LPRIN.

See also: PAUSE, PRINT, USING, and WAIT

Purpose

The LPRINT verb is used to print information on the printer of the optional CE-126P Printer/Cassette Interface.

Use

The LPRINT verb is used to print prompting information, results of calculations, etc. The first form of the LPRINT statement prints a single value. If the expression is numeric, the value will be printed at the far right edge of the paper. If it is a string expression, the print is made starting at the far left.

With the second form of the LPRINT statement the paper is divided into two 12 character halves and the two values are printed in each half according to the same rules as above.

With the third form the print always starts at the left edge and each value is printed immediately following the previous value from left to right with no intervening space.

It is possible to cause PRINT statements to work as LPRINT statements. See the PRINT verb for details.

If an LPRINT statement contains more than 24 characters, the first 24 are printed on one line, the next 24 on the next line, and so forth.

Unlike PRINT, there is no halt or wait after execution of an LPRINT statement.

Verbs
LPRINT

Examples

```
10 A=10: B=20: X$ = "ABCDEF"  
20 LPRINT A  
30 LPRINT X$  
40 LPRINT A, B  
50 LPRINT A; B; X$
```

Paper

			10.
ABCDEF			
	10.		20.
10.20.ABCDEF			

1 **NEXT** numeric variable

Abbreviations: N., NE., NEX.

See also: FOR

Purpose

The NEXT verb is used to mark the end of a group of statements which are being repeated in a FOR/NEXT loop.

Use

The use of the NEXT verb is described under FOR. The numeric variable in a NEXT statement must match the numeric variable in the corresponding FOR.

Examples

```
10 FOR I = 1 TO 10  
20 PRINT I  
30 NEXT I
```

Print the numbers from 1 to 10 each
time the **ENTER** is pressed.

1 **ON** expression **GOSUB** expression list

Where: expression list is: expression
 or: expression , expression list

Abbreviations: O. ; GOS., GOSU.

See also: GOSUB, GOTO, ON ... GOTO

Purpose

The ON ... GOSUB verb is used to execute one of a set of subroutines depending on the value of a control expression.

Use

When the ON ... GOSUB verb is executed the expression between ON and GOSUB is evaluated and reduced to an integer. If the value of the integer is 1, the first subroutine in the list is executed as in a normal GOSUB. If the expression is 2, the second subroutine in the list is executed, and so forth. After the RETURN from the subroutine execution proceeds with the statement which follows the ON ... GOSUB.

If the expression is zero, negative, or larger than the number of subroutines provided in the list, no subroutine is executed and execution proceeds with the next line of the program.

NOTE: Commas may not be used in the expressions following the GOSUB. The PC-1401/1402 cannot distinguish between commas in expressions and commas between expressions.

Examples

```
10 INPUT A
20 ON A GOSUB 100, 200, 300
30 END
100 PRINT "FIRST"
110 RETURN
200 PRINT "SECOND"
210 RETURN
300 PRINT "THIRD"
310 RETURN
```

An input of 1 prints "FIRST"; 2 prints "SECOND"; 3 prints "THIRD". Any other input does not produce any print.

1 **ON** expression **GOTO** expression list

Where: expression list is: expression
or: expression , expression list

Abbreviations: O.; G.; GO.; GOT.

See also: GOSUB, GOTO, ON ... GOSUB

Purpose

The ON ... GOTO verb is used to transfer control to one of a set of locations depending on the value of a control expression.

Use

When the ON ... GOTO verb is executed the expression between ON and GOTO is evaluated and reduced to an integer. If the value of the integer is 1, control is transferred to the first location in the list. If the expression is 2, control is transferred to the second location in the list; and so forth.

If the expression is zero, negative, or larger than the number of locations provided in the list, execution proceeds with the next line of the program.

NOTE: Commas may not be used in the expressions following the GOTO. The PC-1401/1402 cannot distinguish between commas in expressions and commas between expressions.

Examples

```
10 INPUT A
20 ON A GOTO 100,200,300
30 GOTO 900
100 PRINT "FIRST"
110 GOTO 900
200 PRINT "SECOND"
210 GOTO 900
300 PRINT "THIRD"
310 GOTO 900
900 END
```

An input of 1 prints 'FIRST'; 2 prints 'SECOND'; 3 prints 'THIRD'. Any other input does not produce any print.

- 1 **PAUSE** print expr
- 2 **PAUSE** print expr , print expr
- 3 **PAUSE** print list

Where: print list is: print expr
or: print expr ; print list
and: print expr is: expression
or: USING clause ; expression

The USING clause is described separately under USING

Abbreviations: PAU., PAUS.

See also: LPRINT, PRINT, USING, and WAIT

Purpose

The PAUSE verb is used to print information on the display for a short period.

Use

The PAUSE verb is used to display prompting information, results of calculations, etc. The operation of PAUSE is identical to PRINT except that after PAUSE the PC-1401/1402 waits for a short preset interval of about .85 seconds and then continues execution of the program without waiting for the ENTER key or the WAIT interval.

The first form of the PAUSE statement displays a single value. If the expression is numeric, the value is printed at the far right end of the display. If it is a string expression, the display is made starting at the far left.

With the second form of the PAUSE statement the display is divided into two 8 character halves. The two values are displayed in each half according to the same rules as above.

With the third form the display starts at the left edge and each value is displayed immediately following the previous value from left to right with no intervening space.

PAUSE statements are not affected by the PRINT = LPRINT statement (see PRINT).

While it is possible to write PAUSE statements which would display more than 16 characters only the left-most 16 appear in the display. There is no way to see the other characters.

Examples

10 A = 10 : B = 20 : X\$ = "ABCDEF"

Display

20 PAUSE A

10.

30 PAUSE X\$

ABCDEF

40 PAUSE A, B

10.

20.

50 PAUSE A; B; X\$

10. 20. ABCDEF

Verbs

PRINT

- 1 **PRINT** print expr
- 2 **PRINT** print expr , print expr
- 3 **PRINT** print list
- 4 **PRINT** = **LPRINT**
- 5 **PRINT** = **PRINT**

Where: print list is: print expr
or: print expr ; print list
and: print expr is: expression
or: USING clause ; expression

The USING clause is described separately under USING

Abbreviations: P., PR., PRI., PRIN.

See also: LPRINT, PAUSE, USING, and WAIT

Purpose

The PRINT verb is used to print information on the display or on the printer of the CE-126P printer/Cassette Interface.

Use

The PRINT verb is used to display prompting information, results of calculations, etc. The first form of the PRINT statement displays a single value. If the expression is numeric, the value is printed at the far right end of the display. If it is a string expression, the display is made starting at the far left.

With the second form of the PRINT statement the display is divided into two 8 character halves and the two values are displayed in each half according to the same rules as above.

With the third form the display starts at the left edge and each value is displayed immediately following the previous value from left to right with no intervening space.

The fourth and fifth forms of the PRINT statement do no printing. The fourth form causes all PRINT statements which follow it in the program to be treated as if they were LPRINT statements. The fifth form resets this condition so that the PRINT statements will again work with the display.

While it is possible to write PRINT statements which would display more than

16 characters, only the left-most 16 appear in the display. There is no way to see the other characters.

Examples

10 A = 10 : B = 20 : X\$ = "ABCDEF"

Display

20 PRINT A

10.

30 PRINT X\$

ABCDEF

40 PRINT A, B

10.

20.

50 PRINT A; B; X\$

10.20.ABCDEF

Verbs

PRINT

- 1 **PRINT** # "var list"
- 2 **PRINT** # "filename"; var list

Where: var list is: variable
 or: variable , var list

Abbreviations: P. #, PR. #, PRI. #, PRIN. #

See also: INPUT #, PRINT, READ

Purpose

The PRINT # verb is used to store values on the cassette tape.

Use and Examples

The following variable types can be used for variable names:

- (1) Fixed variables — A, B, X, A(26), C*, A(10)*, etc.
- (2) Simple variables — AA, B2, XY\$, etc.
- (3) Array variables — B(*), CD(*), N\$(*), etc.

1) Saving fixed variable contents onto tape

The contents of fixed variables can be saved onto tape by specifying the desired variable names (separated by commas) in the PRINT # statement.

PRINT # "DATA 1"; A, B, X, Y

This statement saves contents of variables A, B, X, and Y into tape file named "DATA 1".

If you wish to save the contents of the specified fixed variable and all the subsequent fixed variables, subscript that variable name with an asterisk * .

PRINT # "D-2"; D* This statement saves the contents of fixed variables D through Z (and of extended variables A(26) and beyond, if defined) into the tape file named "D-2".

PRINT # E, X\$, A(30)* This statement saves the contents of the fixed variables E and X\$ and of the extended variables A(30) and all the remaining variables, onto the tape.

Note:

Subscripted fixed variable names A(1) through A(26) can be specified in the PRINT# statement in much the same way as A through Z (or A\$ through Z\$). However, if array A is already defined by the DIM statement, A() cannot be used to define subscripted fixed variables.

2) Saving simple variable (two-character variable) contents

The contents of simple variables can be saved onto tape by specifying the desired variable names.

```
PRINT# "DM-1"; AB, Y1, XY$
```

This statement saves the contents of the simple variables AB, Y1, and XY\$ into the tape file named 'DM-1'.

3) Saving array variable contents

The contents of all variables of a specific array can be saved onto tape by specifying the array name subscripted by an asterisk enclosed in parentheses (*).

```
PRINT# "DS-2"; X(*), Y$(*)
```

This statement saves the contents of all the elements (X(0), X(1),) of the array X, and of all the elements (Y\$(0), Y\$(1),) of the array Y\$, into the tape file name 'DS-2'.

Note:

It is not possible to save the contents of only one or more specific elements of an array.

While fixed variables or subscripted fixed variables allow you to save only specific parts of them, an array (such as A), defined by the DIM statement does not allow you to save only a specific part of it.

- * If the PRINT# statement is executed with no variable names specified, an error (ERROR 1) will result.

— CAUTION —

The locations for extended variables such as A(27) and beyond, simple variables, and/or array variables must be set aside in the program/data area before the PRINT# statement is executed. Otherwise, the execution of the PRINT# statement for undefined variables will result in an error.

1 RADIAN

Abbreviations: RAD., RADI., RADIA.

See also: DEGREE and GRAD

Purpose

The RADIAN verb is used to change the form of angular values to radian form.

Use

The **PC-1401/1402** has three forms for representing angular values — decimal degrees, radians, and gradient. These forms are used in specifying the arguments to the SIN, COS, and TAN functions and in returning the results from the ASN, ACS, and ATN functions.

The RADIAN function changes the form for all angular values to radian form until a DEGREE or GRAD verb is used. Radian form represents angles in terms of the length of the arc with respect to a radius, i.e., 360° is 2 PI radians since the circumference of a circle is 2 PI times the radius.

Examples

10 RADIAN

20 X = ASN 1

30 PRINT X

X now has a value of 1.570796327 or PI/2, the Arcsine of 1.

1 RANDOM

Abbreviations: RA., RAN., RAND., RANDO.

Purpose

The RANDOM verb is used to reset the seed for random number generation.

Use

When random numbers are generated using the RND function, the **PC-1401/1402** begins with a predetermined "seed" or starting number. The RANDOM verb resets this seed to a new randomly determined value.

The starting seed will be the same each time the **PC-1401/1402** is turned on, so the sequence of random numbers generated with RND is the same each time, unless the seed is changed. This is very convenient during the development of a program because it means that the behavior of the program should be the same each time it is run even though it includes a RND function. When you want to have the numbers be truly random, the RANDOM statement can be used to make the seed itself random.

Examples

```
10 RANDOM
20 X = RND 10
```

When run from line 20, the value of X is based on the standard seed. When run from line 10, a new seed is used.

Verbs

READ

1 READ variable list

Where: variable list

is: variable

or: variable , variable list

Abbreviations: REA.

See also: DATA, RESTORE

Purpose

The READ verb is used to read values from a DATA statement and assign them to variables.

Use

When assigning initial values to an array, it is convenient to list the values in a DATA statement and use a READ statement in a FOR . . . NEXT loop to load the values into the array. When the first READ is executed, the first value in the first DATA statement is returned. Succeeding READs use succeeding values in the sequential order in which they appear in the program, regardless of how many values are listed in each DATA statement or how many DATA statements are used.

If desired, the values in a DATA statement can be read a second time by using the RESTORE statement.

Examples

10 DIM B (10) Set up an array

20 WAIT 128

30 FOR I = 1 TO 10

40 READ B(I)

Loads the values from the DATA statement into

50 PRINT B(I)

B () — B (1) is 10, B (2) is 20, B (3) is 30, etc.

60 NEXT I

70 DATA 10, 20, 30, 40, 50, 60

80 DATA 70, 80, 90, 100

90 END

1 REM remark

Abbreviations: none

Purpose

The REM verb is used to include comments in a program.

Use

Often it is useful to include explanatory comments in a program. These can provide titles, names of authors, dates of last modification, usage notes, reminders about algorithms used, etc. These comments are included by means of the REM statement.

The REM statement has no effect on the program execution and can be included anywhere in the program. Everything following the REM verb in that line is treated as a comment, so the REM verb must be the last statement in a line when multiple statement lines are used.

Examples

```
10 REM THIS LINE HAS NO EFFECT
```

Verbs RESTORE

1 **RESTORE**

2 **RESTORE** expression

Abbreviations: RES., REST., RESTO., RESTOR.

See also: DATA, READ

Purpose

The **RESTORE** verb is used to reread values in a **DATA** statement or to change the order in which these values are read.

Use

In the regular use of the **READ** verb the **PC-1401/1402** begins reading with the first value in a **DATA** statement and proceeds sequentially through the remaining values. The first form of the **RESTORE** statement resets the pointer to the first value of the first **DATA** statement, so that it can be read again. The second form of the **RESTORE** statement resets the pointer to the first value of the first **DATA** statement whose line number is greater than the value of the expression.

Examples

```
10 DIM B(10)
```

Sets up an array

```
20 FOR I = 1 TO 10
```

```
30 READ B(I)
```

Assigns the value 10 to each of the elements of B ().

```
40 RESTORE
```

```
50 NEXT I
```

```
60 DATA 10
```

1 RETURN

Abbreviations: RE., RET., RETU., RETUR.

See also: GOSUB, ON . . . GOSUB

Purpose

The **RETURN** verb is used at the end of a subroutine to return control to the statement following the originating **GOSUB**.

Use

A subroutine may have more than one **RETURN** statement, but the first one executed terminates the execution of the subroutine. The next statement executed will be the one following the **GOSUB** or **ON . . . GOSUB** which calls the subroutine. If a **RETURN** is executed without a **GOSUB**, an Error 5 will occur.

Examples

```
10 GOSUB 100
20 END
100 PRINT "HELLO"
110 RETURN
```

When run this program prints the word "HELLO" one time.

1 STOP

Abbreviations: S., ST., STO.,

See also: END; CONT command

Purpose

The **STOP** verb is used to halt execution of a program for diagnostic purposes.

Use

When the **STOP** verb is encountered in program execution the **PC-1401/1402** execution halts and a message is displayed such as 'BREAK IN 200' where 200 is the number of the line containing the **STOP**. **STOP** is used during the development of a program to check the flow of the program or examine the state of variables. Execution may be restarted using the **CONT** command.

Examples

10 STOP

Causes "BREAK IN 10" to appear in the display.

1 TROFF

Abbreviations : TROF.

See also: TRON

Purpose

The TROFF verb is used to cancel the trace mode.

Use

Execution of the TROFF verb restores normal execution of the program.

Examples

```
10 TRON
20 FOR I = 1 TO 3
30 NEXT I
40 TROFF
```

When run, this program displays the line numbers 10, 20, 30, 30, 30 and 40.

1 TRON

Abbreviations : TR., TRO.

See also: TROFF

Purpose

The TRON verb is used to initiate the trace mode.

Use

The trace mode provides assistance in debugging programs. When the trace mode is on, the line number of each statement is displayed **after** each statement is executed. The **PC-1401/1402** then halts and waits for the Down Arrow key to be pressed before moving on to the next statement. The Up Arrow key may be pressed to see the statement which has just been executed. The trace mode continues until a TROFF verb is executed.

Examples

```
10 TRON
20 FOR I = 1 TO 3
30 NEXT I
40 TROFF
```

When run, this program displays the line numbers 10, 20, 30, 30, 30 and 40.

- 1 USING
- 2 USING "editing specification"
- 3 USING character variable

Abbreviations: U., US., USI., USIN.

See also: LPRINT, PAUSE, PRINT

Further guide to the use of USING is provided in Appendix C

Purpose

The USING verb is used to control the format of displayed or printed output.

Use

The USING verb can be used by itself or as a clause within a LPRINT, PAUSE, or PRINT statement. The USING verb establishes a specified format for output which is used for all output which follows until changed by another USING verb.

The editing specification of the USING verb consists of a quoted string composed of some combination of the following editing characters:

- # Right justified numeric field character
 - Decimal point.
- ^ Used to indicate that numbers should be displayed in scientific notation.
- & Left justified alphanumeric field.

For example, "#####" is an editing specification for a right justified numeric field with room for 3 digits and the sign. In numeric fields, a location must be included for the sign, even if it will always be positive.

Editing specifications may include more than one field. For example "####&&&&" could be used to print a numeric and a character field next to each other.

If the editing specification is missing, as in format 1, special formatting is turned off and the built-in display rules pertain.

Examples

	Display
10 A = 125 : X\$ = "ABCDEF"	
20 PRINT USING "##.## ^"; A	1. 25 E 02
30 PRINT USING "&&&&&&&"; X\$	ABCDEF
40 PRINT USING "#####&&"; A; X\$	125 ABC

- Notes:
1. When the total number of digits specified with USING exceeds 16 for "PRINT expression", ERROR 7 results.
 2. When the number of digits for the integer part (sign and decimal point included) exceeds 8 while using the fixed decimal point system for "PRINT expression , expression", ERROR 7 results.
When the character string of the expression in the form of "PRINT expression , expression" exceeds 8 columns, the excess part is not displayed.
 3. When the display contents of the form "PRINT expression ; expression" exceeds 16 columns, the excess part is not displayed.

- 1 WAIT
- 2 WAIT expression

Abbreviations: W., WA., WAI.

See also: PAUSE, PRINT

Purpose

The WAIT verb is used to control the length of time that displayed information is shown before program execution continues.

Use

In normal execution the **PC-1401/1402** halts execution after a PRINT command until the **ENTER** key is pressed. The WAIT command causes the **PC-1401/1402** to display for a specified interval and then proceed automatically (similar to the PAUSE verb). The expression which follows the WAIT verb determines the length of the interval. The interval may be set to any value from 0 to 65535. Each increment is about one fifty-ninth of a second. WAIT 0 is too fast to be read reasonably; WAIT 65535 is about 19 minutes. WAIT with no following expression resets the **PC-1401/1402** to the original condition of waiting until the **ENTER** key is pressed.

Examples

10 WAIT 59

Causes PRINT to wait about 1 second.

FUNCTIONS

Pseudovariables

Pseudovariables are a group of functions which take no argument and are used like fixed variables wherever required.

1 INKEY\$

INKEY\$ is a string pseudovariable which has the value of the last key pressed on the keyboard. **ENTER** , **C-CE** , **SHIFT** , **DEF** , **↑** , **↓** , **▶** , and **◀** all have a value of NUL. INKEY\$ is used to respond to the pressing of individual keys without waiting for the ENTER key to end the input.

```
10 A$ = INKEY$
20 B = ASC A$
30 IF B = 0 THEN GOTO 10
40 IF B ....
```

Lines 40 and beyond contain tests for the key and the actions to be taken. (For example: 40 PRINT A\$). On first executing the program, the value of INKEY\$ is NUL, since the last key pressed was **ENTER** . If INKEY\$ is used following PRINT or PAUSE, the contents of the display are read instead of a key press.

1 MEM

MEM is a numeric pseudovariable which has the value of the number of characters of program memory remaining. The available program memory will be the total memory less the space consumed by programs and array variables. MEM may also be used as a command. Immediately following reset, MEM has a value of 3534 bytes **PC-1401** and 9678 bytes in the **PC-1402**.

1 PI

PI is a numeric pseudovalue which has the value of PI. It is identical to the use of the special PI character (π) on the keyboard. Like other numbers the value of PI is kept to 10 digit accuracy (3.141592654).

Numeric Functions

Numeric functions are a group of mathematical operations. Some take a single numeric value and return a numeric value. Some take two numeric values and return two numeric values. They include trigonometric functions, logarithmic functions, and functions which operate on the integer and sign parts of a number. Many dialects of BASIC require that the argument to a function be enclosed in parentheses. The **PC-1401/1402** does not require these parentheses, except when it is necessary to indicate what part of a more complex expression is to be included in the argument. Numeric functions with two numeric values all require the parentheses.

LOG 100 + 100 will be interpreted as:

(LOG 100) + 100 not LOG (100 + 100).

1 ABS numeric expression

ABS is numeric function which returns the absolute value of the numeric argument. The absolute value is the value of a number without regard to its sign. ABS -10 is 10.

1 ACS numeric expression

ACS is a numeric function which returns the arccosine of the numeric argument. The arccosine is the angle whose cosine is equal to the expression. The value returned depends on whether the **PC-1401/1402** is in decimal degree, radian, or gradient mode for angles. ACS .5 is 60 in the decimal degree mode.

1 AHC numeric expression

AHC is a numeric function which returns arc-hyperbolic cosine of the numeric argument. AHC 5 is 2.29243167.

Functions
Numeric Functions

1 AHS numeric expression

AHS is a numeric function which returns arc-hyperbolic sine of the numeric argument. AHS 6 is 2.491779853.

1 AHT numeric expression

AHT is a numeric function which returns arc-hyperbolic tangent of the numeric argument.

1 ASN numeric expression

ASN is a numeric function which returns the arcsine of the numeric argument. The arcsine is the angle whose sign is equal to the expression. The value returned depends on whether the **PC-1401/1402** is in decimal degree, radian, or gradient mode for angles. ASN .5 is ~~30~~ in the decimal degree mode.

1 ATN numeric expression

ATN is a numeric function which returns the arctangent of the numeric argument. The arctangent is the angle whose tangent is equal to the expression. The value returned depends on whether the **PC-1401/1402** is in decimal degree, radian, or gradient mode for angles. ATN 1. is 45 in the decimal degree mode.

1 COS numeric expression

COS is a numeric function which returns the cosine of the angle argument. The value returned depends on whether the **PC-1401/1402** is in decimal degree, radian, or gradient mode for angles. COS ~~60~~ is .5 in the decimal degree mode.

1 **CUR** numeric expression

CUR is a numeric function which returns the cubic root of its argument.
CUR 8 is 2.

1 **DEG** numeric expression

The DEG function converts an angle argument in DMS (Degree, Minute, Second) format to DEG (Decimal Degree) form. In DMS format the integer portion of the number represents the degree, the first and second digits of the decimal represent the minutes, the third and fourth digits of the decimal represent the seconds, and any further digits represent decimal seconds. For example, $55^{\circ} 10' 44.5''$ is represented as 55.10445. In DEG format the integer portion is degrees and the decimal portion is decimal degrees. DEG 55.10445 is 55.17902778.

1 **DMS** numeric expression

DMS is a numeric function which converts an angle argument in DEG format to DMS format (see DEG). DMS 55.17902778 is 55.10445.

1 **EXP** numeric expression

EXP is a numeric function which returns the value of e (2.718281828 – the base of the natural logarithms) raised to the value of the numeric argument. EXP 1 is 2.718281828. (Press the **[E]** **[X]** **[P]** 1 **[ENTER]**.)

Functions
Numeric Functions

1 **FACT** numeric expression

FACT is a numeric function which returns the factorial of its argument.
FACT 5 is 120.

1 **HCS** numeric expression

HCS is a numeric function which returns hyperbolic cosine of the numeric argument. HCS 5 is 74.20994852.

1 **HSN** numeric expression

HSN is a numeric function which returns hyperbolic sine of the numeric argument. HSN 4 is 27.2899172.

1 **HTN** numeric expression

HTN is a numeric function which returns hyperbolic tangent of the numeric argument. HTN 1 is 0.761594156.

1 **INT** numeric expression

INT is a numeric function which returns the integer part of its numeric argument.
INT PI is 3.

1 **LN** numeric expression

LN is a numeric function which returns the logarithm to the base e (2.718281828) of its numeric argument. LN 100 is 4.605170186.

1 **LOG** numeric expression

LOG is a numeric function which returns the logarithm to the base 10 of its numeric argument. LOG 100 is 2.

1 **POL** (numeric expression, numeric expression)

POL is a numeric function which converts numeric arguments in rectangular coordinates format to polar coordinate format.

The first numeric argument indicates the distance from the y-axis and the second numeric argument indicates the distance from the x-axis. The values converted, the distance and the angle in the polar coordinates, are assigned to the fixed variables Y and Z respectively. The angle converted depend on whether the **PC-1401/1402** is in decimal degree, radian, or gradient mode for angles. POL (3, 4) is (5, 53.13010235) in decimal degree.

1 **RCP** numeric expression

RCP is a numeric function which returns the reciprocal of its numeric argument. RCP 5 is 0.2.

Functions
Numeric Functions

1 REC (numeric expression, numeric expression)

REC is a numeric function which converts numeric arguments in polar coordinates format to rectangular coordinates format.

The first numeric argument indicates the distance and second numeric argument indicates the angle which depends on whether the **PC-1401/1402** is in decimal degree, radian, or gradient mode for angles. The values converted, the distances from the y-axis and the x-axis, are assigned into the fixed variables Y and Z respectively. REC (7, 50) in (4.499513268, 5.362311102) in decimal degree.

1 RND numeric expression

RND is a numeric function which generates random numbers. If the value of the argument is less than one but greater than or equal to zero, the random number is less than one and greater than or equal to zero. If the argument is an integer greater than or equal to 1, the result is a random number greater than or equal to 1 and less than or equal to the argument. If the argument is greater than or equal to 1 and not an integer, the result is a random number greater than or equal to 1 and less than or equal to the smallest integer which is larger than the argument: (In this case, the generation of the random number changes depending on the value of the decimal portion of the argument.):

----- Result -----		
<u>Argument</u>	<u>Lower Bound</u>	<u>Upper Bound</u>
.5	0	<1
2	1	2
2.5	1	3

The same sequence of random numbers is normally generated because the same "seed" is used each time the **PC-1401/1402** is turned on. To randomize the seed, see the **RANDOM** verb.

1 numeric expression **ROT** numeric expression

ROT is a numeric function which returns the power root of its argument.

125 ROT 3 is 5.

(i.e.: $\sqrt[3]{125}$ should be entered as 125 ROT 3.)

1 **SGN** numeric expression

SGN is a numeric function which returns a value based on the sign of the argument.

If the argument is positive, the result is 1; if the argument is zero, the result is 0; if the argument is negative, the result is -1. SGN -5 is -1.

1 **SIN** numeric expression

SIN is a numeric function which returns the sine of the angle argument. The value returned depends on whether the **PC-1401/1402** is in decimal degree, radian, or gradient mode for angles. SIN 30 is .5.

1 **SQR** numeric expression

SQR is a numeric function which returns the square root of its argument. It is identical to the use of the special square root symbol ($\sqrt{\quad}$) on the keyboard. SQR 4 is 2.

1 **SQU** numeric expression

SQU is a numeric function which returns the square of its numeric argument. SQU 3 is 9.

Functions
Numeric Functions

1 **TAN** numeric expression

TAN is a numeric function which returns the tangent of its angle argument. The value returned depends on whether the **PC-1401/1402** is in decimal degree, radian, or gradient mode for angles. TAN 45 is 1 in decimal degree.

1 **TEN** numeric expression

TEN is a numeric function which returns the value of 10 (the base of the common logarithms) raised to the value of its numeric argument.
TEN 3 is 1000.

String Functions

String functions are a group of operations used for manipulating strings. Some take a string argument and return a numeric value. Some take a string argument and return a string. Some take a numeric value and return a string. Some take a string argument and one or two numeric arguments and return a string. Many dialects of BASIC require the argument of a function to be enclosed in parentheses. The **PC-1401/1402** does not require these parentheses, except when it is necessary to indicate what part of a more complex expression is to be included in the argument. String functions with two or three arguments all require the parentheses.

1 **ASC** string expression

ASC is a string function which returns the numeric ASCII code value of the first character in its argument. The chart of ASCII codes and their relationship to characters is given in Appendix B. ASC "A" is 65.

1 **CHR\$** numeric expression

CHR\$ is a string function which returns the character which corresponds to the numeric ASCII code of its argument. The chart of ASCII codes and their relationship to characters is given in Appendix B. CHR\$ 65 is "A".

1 **LEFT\$** (string expression , numeric expression)

LEFT\$ is a string function which returns the leftmost part of the string first argument. The number of characters returned is determined by the numeric expression. LEFT\$ ("ABCDEF", 2) is "AB".

1 **LEN** string expression

LEN is a string function which returns the length of the string argument. LEN "ABCDEF" is 6.

Functions

String Functions

1 **MID\$** (string expression , num. exp. 1 , num. exp. 2)

MID\$ is a string function which returns a middle portion of the string first argument. The first numeric argument indicates the first character position to be included in the result. The second numeric argument indicates the number of characters that are to be included. MID\$ ("ABCDEF", 2, 3) is "BCD".

1 **RIGHT\$** (string expression , numeric expression)

RIGHT\$ is a string function which returns the rightmost part of the string first argument. The number of characters returned is determined by the numeric argument. RIGHT\$ ("ABCDEF", 3) is "DEF".

1 **STR\$** numeric expression

STR\$ is a string function which returns a string which is the character representation of its numeric argument. It is the reverse of VAL. STR\$ 1.59 is '1.59'.

1 **VAL** string expression

VAL is a string function which returns the numeric value of its string argument. It is the reverse of STR\$. The VAL of a non-number is zero. VAL "1.59" is 1.59.

Note: The character-string convertible by VAL function to a numerical value consists of numerals (0 to 9), symbols (+ and -) and a symbol (E) indicating an exponential portion. No other characters and symbols are included. If a character-string includes other characters and symbols, any character-string on the right of that character-string will be ignored. If included in a character-string, a space is usually regarded as non-existing.

CHAPTER 9

TROUBLESHOOTING

This chapter provides you with some hints on what to do when your **SHARP PC-1401/1402** does not do what you expect it to do. It is divided into two parts — the first part deals with general machine operation and the second with BASIC programming. For each problem there are a series of suggestions provided. You should try each of these, one at a time, until you have fixed the problem.




Machine Operation

If:	Then You Should:
You turn on the machine but there is nothing on the display	<ol style="list-style-type: none"> 1. Check to see that the power switch is set to ON position. 2. Press the ON BRK key to see if AUTO POWER OFF has been activated. 3. Replace the batteries.
There is a display, but no response to keystrokes	<ol style="list-style-type: none"> 1. Press C-CE key to clear. 2. Press CA (SHIFT C-CE) to clear. 3. Turn OFF and ON again. 4. Hold down any key and push RESET. 5. Push RESET without any key.
You have typed in a calculation or answer and get no response	<ol style="list-style-type: none"> 1. In RUN mode press ENTER or in CAL mode press = .
You are running a BASIC program and it displays something, and stops	<ol style="list-style-type: none"> 1. Press ENTER .
You enter a calculation and it is displayed in BASIC statement format (colon after the first number)	<ol style="list-style-type: none"> 1. Change from the PROgram into the RUN mode for calculations.
You get no response from any keys.	<ol style="list-style-type: none"> 1. Hold down any key and push RESET. 2. If you get no response from any key even when the above operation is performed, push the RESET <u>without pushing any key</u>. This will <u>clear</u> the program and data.



BASIC Debugging

When entering a new BASIC program, it is usual for it **not** to work the first time. Even if you are simply keying in a program that you know is correct, such as those provided in this manual, it is usual to make at least one typing error. If it is a new program of any length, it will probably contain at least one logic error as well. Following are some general hints on how to find and correct your errors.

You run your program and get an error message:

1. Go back to the PROgram mode and use the  or the  keys to recall the line with the error. The cursor will be positioned at the place in the line where the **PC-1401/1402** got confused.
2. If you can't find an obvious error in the way in which the line is written, the problem may lie with the values which are being used. For example, **CHR\$ (A)** will produce an error if A has a value of 1 because **CHR\$ (1)** is an illegal character. Check the values of the variables in either the **RUN** or the **PROgram** mode by typing in the name of the variable followed by .

You **RUN** the program and don't get an error message, but it doesn't do what you expect.

3. Check through the program line by line using **LIST** and the  and  keys to see if you have entered the program correctly. It is surprising how many errors can be fixed by just taking another look at the program.
4. Think about each line as you go through the program as if you were the computer. Take sample values and try to apply the operation in each line to see if you get the result that you expected.
5. Insert one or more extra **PRINT** statements in your program to display key values and key locations. Use these to isolate the parts of the program that are working correctly and the location of the error. This approach is also useful for determining which parts of a program have been executed. You can also use **STOP** to temporarily halt execution at critical points so that several variables can be examined.
6. Use **TRON** and **TROFF**, either as commands or directly within the program to trace the flow of the program through individual lines. Stop to examine the contents of critical variables at crucial points. This is a very slow way to find a problem, but sometimes it is also the only way.

CHAPTER 10

MAINTENANCE OF THE PC-1401/1402

To insure trouble-free operation of your **SHARP PC-1401/1402** we recommend the following:

- * Always handle the pocket computer carefully as the liquid crystal display is made of glass.
- * Keep the computer in an area free from extreme temperature changes, moisture, or dust. During warm weather, vehicles left in direct sunlight are subject to high temperature build up. Prolonged exposure to high temperature may cause damage to your computer.
- * Use only a soft, dry cloth to clean the computer. Do not use solvents, water, or wet cloths.
- * To avoid battery leakage, remove the batteries when the computer will not be in use for an extended period of time.
- * If service should be required on this equipment, use only a **SHARP** servicing dealer, a **SHARP** approved service facility or **SHARP** repair service where available.
- * If the computer is subjected to strong static electricity or external noise it may "hang up" (all keys become inoperative). If this occurs, press the ALL RESET button while holding down any key. (See Troubleshooting).
- * Keep this manual for further reference.

APPENDIX A

ERROR MESSAGES

There are nine different error codes built into the **PC-1401/1402**. The following table will explain these codes.

Error Number	Meaning
1	<p>Syntax error.</p> <ul style="list-style-type: none">This means that the PC-1401/1402 can't understand what you have entered. Check for things such as semicolons on the ends of PRINT statements, misspelled words, and incorrect usages. <p>$3 * / 2$</p>
2	<p>Calculation error.</p> <p>Here you have probably done one of three things:</p> <ol style="list-style-type: none">Tried to use too large a number. Calculation results are greater than 9.999999999E 99.Tried to divide by zero. $5 / 0$An illogical calculation has been attempted. $LN -30$ or $ASN 1.5$
3	<p>DIMension error/Augument error.</p> <ul style="list-style-type: none">Array variable already exists. Array specified without first dimensioning it. Array subscript exceeds size of array specified in DIM statement. $DIM B (256)$Illegal function argument. This means that you have tried to make the computer do something that it just can't handle. The interval that is greater than 65535. $WAIT 66000$

4 Line Number error.

Here you have probably done one of two things:

1. Tried to use an unexsisting line number by the GOTO, GOSUB, RUN, LIST or THEN etc.
2. Tried to use too large a line number. The maximum line number is 65279.

5 Nesting error.

Subroutine nesting exceeds 10 levels.

FOR loop nesting exceeds 5 levels.

RETURN verb without a GOSUB, NEXT verb without a FOR, or READ verb without a DATA.

Buffer space exceeded.

6 Memory Overflow.

Generally this error happens when you've tried to DIMension an array that is too big for memory. This can also happen when a program becomes too large.

7 PRINT USING error.

This means that you have put an illegal format specifier into a USING statement.

8 I/O device error.

This error can happen only when you have the optional printer and/or cassette recorder connected to the **PC-1401/1402**. It means that there is a problem with communication between the I/O device and the **PC-1401/1402**.

9 Other errors.

This code will be displayed whenever the computer has a problem that isn't covered by one of the other eight error codes. One of the most common causes for this error is trying to access data in a variable in one fashion (e.g. A\$) while the data was originally stored in the variable in another fashion (e.g. A).

APPENDIX B ASCII CHARACTER CODE CHART

The following chart shows the conversion values for use with CHR\$ and ASC. The column shows the first hex character or the first four binary bits, the row shows the second hex character or the second binary bits. The upper left corner of each box contains the decimal number for the character. The lower right shows the character. If no character is shown then it is an illegal character on the PC-1401/1402.

For examples, the character "A" is a decimal 65 or a hex 41 or binary 01000001. The character '√' is a decimal 252 or a hex FC or a binary 11111100.

Note: When using either the **CE-125** or **CE-126P** optional printer, be aware that the 39 (&27), 91 (&5B), and 93 (&5D) character codes for the **PC-1401/1402** (displayed characters) and printer (printed characters) are different characters.

ASCII Character Code Chart

First 4 bits

PC-1401/1402 does not recognize codes in shaded area. If you enter the code number in the shaded area, an error will result.

Second 4 bits

Hex	0	1	2	3	4	5	6	7	8	E	F
Binary	0000	0001	0010	0011	0100	0101	0110	0111	1000	1110	1111
0	0	16	32	48	64	80	96	112	128	224	240
0000	NUL		SPACE	0	@	P					
1	1	17	33	49	65	81	97	113	129	225	241
0001			!	1	A	Q					
2	2	18	34	50	66	82	98	114	130	226	242
0010			"	2	B	R					
3	3	19	35	51	67	83	99	115	131	227	243
0011			#	3	C	S					
4	4	20	36	52	68	84	100	116	132	228	244
0100			\$	4	D	T					
5	5	21	37	53	69	85	101	117	133	229	245
0101			%	5	E	U					
6	6	22	38	54	70	86	102	118	134	230	246
0110			&	6	F	V					
7	7	23	39	55	71	87	103	119	135	231	247
0111			,	7	G	W					
8	8	24	40	56	72	88	104	120	136	232	248
1000			(8	H	X					
9	9	25	41	57	73	89	105	121	137	233	249
1001)	9	I	Y					
A	10	26	42	58	74	90	106	122	138	234	250
1010			*	:	J	Z					
B	11	27	43	59	75	91	107	123	139	235	251
1011			+	;	K	[π
C	12	28	44	60	76	92	108	124	140	236	252
1100			,	<	L	¥					√
D	13	29	45	61	77	93	109	125	141	237	253
1101			-	=	M]					
E	14	30	46	62	78	94	110	126	142	238	254
1110			.	>	N	^					
F	15	31	47	63	79	95	111	127	143	239	255
1111			/	?	O	-					

APPENDIX C FORMATTING OUTPUT

It is sometimes important or useful to control the format as well the content of output. The **PC-1401/1402** controls display formats with the **USING** verb. This verb allows you to specify:

- * The number of digits
- * The location of the decimal point
- * Scientific notation format
- * The number of string characters

These different formats are specified with an "output mask". This mask may be a string constant or a string variable:

10: USING "####"

20: M\$ = "&&&&&&"

30: USING M\$

When the **USING** verb is used with no mask, all special formatting is cancelled.

40: USING

A **USING** verb may also be used within a **PRINT** statement:

50: PRINT USING M\$; N

Wherever a **USING** verb is used, it will control the format of all output until a new **USING** verb is encountered.

Numeric Masks

A numeric **USING** mask may only be used to display numeric values, i.e., numeric constants or numeric variables. If a string constant or variable is displayed while a numeric **USING** mask is in effect, the mask will be ignored. A value which is to be displayed must always fit within the space provided by the mask. The mask must reserve space for the sign character, even when the number will always be positive. Thus a mask which shows four display positions may only be used to display numbers with three digits.

Specifying Number of Digits

The desired number of digits is specified using the '#' character. Each '#' in the mask reserves space for one digit. The display or print always contains as many characters as are designated in the mask. The number appears to the far right of this field; the remaining positions to the left are filled with spaces. Positive numbers therefore always have at least one space at the left of the field. Since the **PC-1401/1402** maintains a maximum of 10 significant digits, no more than 11 '#' characters should be used in a numeric mask.

When the total number of columns of the integer part specified exceed 11, this integer part is regarded as 11 digits in the **PC-1401/1402**.

NOTE: In all examples in this appendix the beginning and end of the displayed field will be marked with a '|' character to show the size of the field.

<u>Statement</u>	<u>Display</u>
10: USING "####"	(Set the PC-1401/1402 to the RUN mode, type RUN, and press ENTER .)
20: PRINT 25	25
30: PRINT -350	-350
40: PRINT 1000	ERROR 7 IN 40

Notice that the last statement produced an error because 5 positions (4 digits and a sign space) were required, but only 4 were provided in the mask.

Specifying a Decimal Point

A decimal point character, '.', may be included in a numeric mask to indicate the desired location of the decimal point. If the mask provides more significant decimal digits than are required for the value to be displayed, the remaining positions to the right will be filled with zeros. If there are more significant decimal digits in the value than in the mask, the extra digits will be truncated (**not rounded**):

<u>Statement</u>	<u>Display</u>
10: USING "####.##"	
20: PRINT 25	25.00
30: PRINT -350.5	-350.50
40: PRINT 2.547	2.54

Specifying Scientific Notation

A “^” character may be included in the mask to indicate that the number is to be displayed in scientific notation. The ‘#’ and ‘.’ characters are used in the mask to specify the format of the “characteristic” portion of the number, i.e., the part which is displayed to the left of the E. Two ‘#’ characters should always be used to the left of the decimal point to provide for the sign character and one integer digit. The decimal point may be included, but is not required. Up to 9 ‘#’ characters may appear to the right of the decimal point. Following the characteristic portion, the exponentiation character, E, will be displayed followed by one position for the sign and two positions for the exponent. Thus, the smallest scientific notation field would be provided by a mask of “##^” which would print numbers of the form ‘2E 99’. The largest scientific notation field would be “##.#####^” which would print numbers such as ‘-1.234567890 E -12’:

<u>Statement</u>	<u>Display</u>
10: USING “###.##^”	
20: PRINT 2	2. 00 E 00
30: PRINT -365.278	-3. 65 E 02

Specifying Alphanumeric Masks

String constants and variables are displayed using the '&' character. Each '&' indicates one character in the field to be displayed. The string will be positioned at the left end of this field. If the string is shorter than the field, the remaining spaces to the right will be filled with spaces. If the string is longer than the field, the string will be truncated to the length of the field:

<u>Statement</u>	<u>Display</u>
10: USING "&&&&&&"	
20: PRINT "ABC"	A B C
30: PRINT "ABCDEFGHI"	A B C D E F

Mixed Masks

In most applications a USING mask will contain either all numeric or all string formatting characters. Both may be included in one USING mask, however, for certain purposes. In such cases, each switch from numeric to string formatting characters or vice versa marks the boundary for a different value. Thus, a mask of "#####&&&&" is a specification for displaying two separate values — a numeric value which is allocated 5 positions and a string value which is allocated 4 positions:

<u>Statement</u>	<u>Display</u>
10: PRINT USING "####.##&&"; 25; "CR"	25. 00CR
20: PRINT -5.789; "DB"	-5. 78DB

Remember: Once specified, a USING format is used for all output which follows until cancelled or changed by another USING verb.

Precaution for USING verb

When display format is designated with USING, please note the followings.

1. When the total number of columns exceed 16 with a USING format in the PRINT expression, ERROR 7 results.

Example: 10: USING "#####.#####"
17 columns or more

20: PRINT A

Note: When the total number of columns specified in the integer part exceed 11 including sign digit, the excess part is ignored. Also when the total number of columns of the integer part specified exceed 11, this integer part is regarded as 11 digits in the PC-1401/1402.

Examples:

1) 10: USING "#####."

This format does not cause the error.

2) 10: USING "#####.####"

This format does not also cause the error.

Integer part: 11 digits, Decimal point: 1 digit and Decimal part: 4 digits

Total: 16 digits (less than 17 digits)

3) 10: USING "#####.####"

This format cause the error 7.

(Total digits are greater than 16 digits.)

- Change the USING format to within 16 columns.

2. When the integer part exceeds 8 columns with a USING format in the PRINT expression, expression, ERROR 7 results.

Example: 10: USING "#####.##"
9 columns or more

20: PRINT A , B

- Change the USING format to within 8 columns.

3. When the display contents of the form PRINT expression ; expression ; expression ; exceeds 16 columns, the excess part is not displayed.
4. Displaying the value converted to hexadecimal notation when floating decimal system (exponential display) is designated with the USING statement, will result in error.
(ERROR 7, format error)

APPENDIX D EXPRESSION EVALUATION AND OPERATOR PRIORITY

When the **SHARP PC-1401/1402** is given a complex expression, it evaluates the parts of the expression in a sequence which is determined by the priority of the individual parts of the expression. If you enter the expression:

$$100 / 5 + 45$$

as either a calculation or as a part of a program, the **PC-1401/1402** does not know if you meant:

$$\frac{100}{5 + 45} = 2 \quad \text{or} \quad \frac{100}{5} + 45 = 65$$

Since the **PC-1401/1402** must have some way to decide between these options, it uses its rules of operator priority. Because division has a higher "priority" than addition (see below), it will choose to do the division first and then the addition, i.e., it will choose the second option and return a value of 65 for the expression.

Operator Priority

Operators on **BASIC** mode of the **SHARP PC-1401/1402** are evaluated with the following priorities from highest to lowest:

Level	Operations
1.	Parentheses
2.	Variables and Pseudovariables
3.	Functions
4.	Exponentiation (^), (ROT)
5.	Unary minus, negative sign (-)
6.	Multiplication and division (*, /)
7.	Addition and subtraction (+, -)
8.	Relational operators (<, <=, =, <>, >=, >)
9.	Logical operators (AND, OR)

When there are two or more operators at the same priority level the expression will be evaluated from left to right. (The exponentiation will be evaluated from right to left). Note that with $A+B-C$, for example, the answer is the same whether the addition or the subtraction is done first.

When an expression contains multiple nested parentheses, the innermost set is evaluated first and evaluation then proceeds outward.

APPENDIX D

Expression Evaluation

For level 3 and 4, the last entry has a higher priority.

$$\begin{aligned}\text{For example: } -2 \wedge 4 &\rightarrow -(2^4) \\ 3 \wedge -2 &\rightarrow 3^{-2}\end{aligned}$$

Sample Evaluation

Starting with the expression:

$$((3+5-2)*6+2) / 10 \wedge \text{LOG } 100$$

The **PC-1401/1402** would first evaluate the innermost set of parentheses. Since '+' and '-' are at the same level it would move from left to right and would do the addition first:

$$((8-2)*6+2) / 10 \wedge \text{LOG } 100$$

Then it would do subtraction:

$$((6)*6+2) / 10 \wedge \text{LOG } 100$$

or:

$$(6*6+2) / 10 \wedge \text{LOG } 100$$

In the next set of parentheses it would do the multiplication first:

$$(36+2) / 10 \wedge \text{LOG } 100$$

And then the addition:

$$(38) / 10 \wedge \text{LOG } 100$$

or:

$$38 / 10 \wedge \text{LOG } 100$$

Now that the parentheses are cleared, the LOG function has the highest priority so it is done next:

$$38 / 10 \wedge 2$$

The exponentiation is done next:

$$38 / 100$$

And last of all the division is performed:

$$0.38$$

This is the value of the expression.

APPENDIX E


KEY FUNCTIONS IN BASIC MODE

ON
BRK



(ON)

Use to turn the **PC-1401/1402** power on when the auto power off function is in effect.

(BREAK)

- Depression this key during program execution functions as a BREAK () key and causes to interrupt the program execution.
- When pushed during manual execution, input/output command such as BEEP, CLOAD, etc., execution of the command is interrupted.
- The yellow key marked "SHIFT" must be used to designate a second function. (The material appearing in brown above each key.

SHIFT

Ex.   → ? is inputted.

C-CE

- Use to clear the contents of the entry and the display. (Error release)

SHIFT CA

- Not only clears the display contents, but resets the computer to its initial state.
 - Initial state —
 - Resets the WAIT timer.
 - Resets the display format. (USING format)
 - Resets the TRON state (TROFF).
 - Resets the PRINT = LPRINT.
 - Resets error.

0 ~ 9

- Numeric keys.
- Decimal point.
- Use to enter an abbreviation of a command/verb/function.
- Use to designate the decimal portion in USING format designation.

.

E

- Use to designate an exponent in scientific notation. (This key is a letter E key)

EXP





























- Use to designate an exponent in scientific notation.

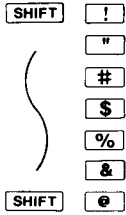
/

- Division key.

APPENDIX E

Key Functions in Basic Mode

- 
 - Multiplication key.
- 
 - Use to designate an array variable in the INPUT#, the PRINT#, etc.
 - Addition key.
- 
 - Subtraction key.
-  
 - Use to enter CLOAD?.
-  
 - Use to divide two or more statements in one line.
- 
 - Use to provide pause between two equations, and between variables or comments.
-  
 - Use to provide multi-display (two or more values/contents/ displayed at a time.
 - Use to provide pause between the instruction and the variable.
- 
 - In assignment statements, use to assign the contents (number or character) on the right for the variable specified on the left.
 - Use when inputting logical operators in IF sentence.
- 
 - When any one of eighteen keys (A, S, D, F, G, H, I, K, L, Z, X, C, V, B, N, M, , SPaCe) is pushed after the depression of the  key, it starts to execute the program from the program line that has the same label as the key code depressed.
-  ~ 
 - Alphabet keys. You are probably familiar with these keys from the standard typewriter keyboard. On the **PC-1401/1402** display the characters appear in the upper case.
-  , 
 - Use to input parentheses.
-   }
  }
 - Use when inputting logical operators in IF sentence.
- 
 - Use to provide space when inputting programs or characters.
-  
 - Use for power calculation instructions.
 - Use to specify the floating decimal point system (exponent display) for numerical data in USING statement instructions.
-  
 - Use to designate pi (π).
- 
 - Use to designate square root.



- Use to designate these symbols.
- " : ● Use to designate and cancel characters.
- # : Use with USING statement, to provide the instruction to define the display format of numerical data.
- \$: ● Use when assigning character variables.
- & : ● Use with USING statement, to provide the instruction to define the display format of character string.
- Use to designate hexadecimal number.



- Shifts the cursor to the right (press once to advance one position, hold down for automatic advance)
- Executes playback instructions.
- Clears an error condition in manual operation.



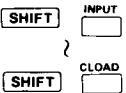
- Shifts the cursor to the left (press once to advance one position, hold down for automatic advance)
- Executes playback instructions.
- Clears an error condition in manual operation.



- Inserts one space (appears) of 1-step capacity between the address (N) indicated by the cursor and the preceding address (N-1).



- Deletes the contents of the address indicated by the cursor.



- Preset command and statement keys. Pressing **SHIFT** and then the alphabet (including comma, space and **ENTER**) key under the command and statement desired enters the designated command and statement to the **PC-1401/1402**.



- Use to set CAL mode.



- Use to set the RUN mode when the CAL mode is set.
- Use to set the PRO mode when the RUN mode is set.
- The RUN and PRO modes are selected alternately each time you press the **BASIC** key.



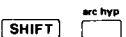
- Use to set print and non-print mode when an optional printer is connected with the **PC-1401/1402**.



- Use to designate an angular mode.



- Use to enter hyperbolic function .



- Use to enter enter inverse hyperbolic function.





- Use to enter a function defined in each key.



APPENDIX E

Key Functions in Basic Mode

ENTER

- Enters a program line into the computer.
- Use when writing in programs.
- Requests manual calculation or direct execution of a command statements by the computer.
- Enters a restart instruction after inputting data required by an INPUT statement or after executing a PRINT statements.

The  and  keys have the following functions, depending on designated modes, as well as the state of the computer.

Mode	State		
RUN	Program being executed		
	Program is temporarily interrupted	To execute the next line	To display program line being executed or already executed, hold this key down.
	INPUT statement being executed		
	PRINT statement just now executed		
	Under break		
	Error condition during executing program		To display error-producing line, hold this key down.
	TRON condition	To execute debugging operation	To display program line being executed or already executed, hold this key down.
	Other condition	To display an answer just previously calculated. (Last answer function)	Same as left
PRO	(When the mode is changed from RUN to PRO and program line is not being displayed)		
	Program is temporarily interrupted	To display the line interrupted	Same as left.
	Error condition	To display the line with error	Same as left
	Other condition	To display the first line	To display the last line
	(When the program line is being display)		
		To display the next program line	To display the preceding program line

Note: The following keys can not be used in the BASIC mode (RUN or PRO mode).

,  , ,  ,  , , , , , and keys used to obtain the statistics (i.e. n, \bar{x} , etc.)

APPENDIX F SPECIFICATIONS

Model:	PC-1401/1402 Pocket Computer		
Processor:	8 bit CMOS CPU		
Programming Language:	BASIC		
Memory Capacity:	System ROM:	40 K Bytes	
	RAM:		
	System	About 500 Bytes	
	User		
	Fixed Memory Area (A ~ Z, A\$ ~ Z\$)	208 Bytes	
	Program/Data Area	PC-1401	3534 Bytes
		PC-1402	9678 Bytes
Stack:	Sub-routine:	10 stacks	Function: 16 stacks
	FOR—NEXT:	5 stacks	Data: 8 stacks
Operators:	Addition, subtraction, multiplication, division, trigonometric and inverse trigonometric functions, logarithmic and exponential functions, angle conversion, square and square root, cubic root, hyperbolic and inverse hyperbolic functions, coordinate conversion, power root, sign, absolute, integer, relational operators, logical operators, etc.		
Numeric Precision:	10 digits (mantissa) + 2 digits (exponent).		
Editing Features:	Cursor left and right, line up and down, character insert, character delete.		
Memory Protection:	CMOS Battery backup.		
Display:	16 character liquid crystal display with 5 x 7 dot characters.		
Keys:	76 keys: Alphabetic, numeric, special symbols, and functions. Numeric pad. User defined keys.		
Power Supply:	6.0V DC: Lithium cells. Type: CR-2032 x 2		
Power Consumption:	6.0V DC @ 0.03W		
	Approx. 300 hours when 55555555. is on continuous display at an operating temperature of 20°C, this time may vary slightly with the operation method, etc.		
	<ul style="list-style-type: none"> One hour operation per day allows the battery to be used for approx. 4 months. This is true for one hour operation consisting of 10 minutes of calculations or program executions and 50 minutes of displays. 		

Dimensions:	170(W) x 72(D) x 9.5(H) mm. 6-11/16"(W) x 2-27/32(D) x 3/8"(H)
Weight:	Approximately (with cells and hard cover) 150g (0.33 lbs.)
Accessories:	Hard cover, two lithium cells (built-in), keyboard-template and instruction manual.
Options:	Printer/Cassette Interface (CE-126P)

APPENDIX G FEATURE COMPARISON OF THE PC-1211, PC-1245, PC-1251, PC-1401, PC-1402 AND PC-1500

The four **SHARP** pocket computers, the **PC-1211**, the **PC-1245**, the **PC-1401/1402**, the **PC-1251**, and the **PC-1500** have many features in common, but are some significant differences. Sometimes the same features are present, yet act in a slightly different fashion. In order to facilitate the use of programs on different models the following comparison charts are provided.

Verbs and Commands

In the following chart the symbol:

- M indicates that the feature can only be used in manual execution, i.e., as a command;
- P indicates that the feature can only be used within a program;
- B indicates that the feature can be used in both contexts.

When no symbol is shown, the feature is not available on that machine

	PC-1211	PC-1245 PC-1251 PC-1401 PC-1402	PC-1500	Comments
AREAD	P	P	P	See Note 1
ARUN			P	
BEEP	P	B	B	PC-1500 has tone and duration
CHAIN	P	*	P	
CLEAR	B	B	B	* PC-1245/1251 have this verb.
CLOAD	M	M	M	
CLOAD?	M	M	M	
CLS			B	
COLOR			B	
CONT	M	M	M	
CSAVE	M	B	B	
CSIZE			B	
CURSOR			B	
DEGREE	B	B	B	
DATA		P	P	

Verbs and Commands (continued)

	PC-1211	PC-1245 PC-1251 PC-1401 PC-1402	PC-1500	Comments
DEBUG	M			
DIM		B	B	
END	P	P	P	
FOR ... TO ... STEP	P	P	P	
GOSUB	P	P	P	
GOTO	P	B	B	
GCURSOR			B	
GPRINT			B	
GRAD	B	B	B	
GRAPH			B	
IF ... THEN	P	P	P	
INPUT	P	P	P	
INPUT #	B	B	B	
LET	P	P	P	
LF			B	
LINE			B	
LIST	M	M	M	
LLIST		M	M	PC-1211 can emulate with LIST
LOCK			B	
LPRINT		P	B	See Note 2
MERGE	M	*	M	* PC-1245/1251 have this command.
NEW	M	M	M	
NEXT	P	P	P	
ON ... ERROR			P	
ON ... GOSUB		P	P	
ON ... GOTO		P	P	
PAUSE	P	P	P	
PASS		M		
PRINT	P	P	B	See Note 2
PRINT #	B	B	B	
RADIAN	B	B	B	
RANDOM		B	B	
READ		P	P	
REM	P	P	P	

APPENDIX G

Feature Comparison

Verbs and Commands (continued)

	PC-1211	PC-1245 PC-1251 PC-1401 PC-1402	PC-1500	Comments
RESTORE		P	P	
RETURN	P	P	P	
RLINE			B	
RMTOFF			B	
RMTON			B	
ROTATE			B	
RUN	M	M	M	
SORGN			B	
STOP	P	P	P	
TAB			B	
TEST			B	
TEXT			B	
TROFF		B	B	
TRON		B	B	
UNLOCK			B	
USING	P	B	B	See Note 3
WAIT		B	B	

Note 1: There are some minor differences between the **PC-1245/1251/1401/1402** and the **PC-1211** in the behavior of AREAD following PRINT, but these are unlikely to cause problems in ordinary usage.

Note 2: Add PRINT = LPRINT and PRINT = PRINT statements to **PC-1211** programs to achieve the desired results on the **PC-1245/1251/1401/1402**.

Note 3: On the **PC-1211** the USING format applies to all displays on the line in which the USING clause appears, even if the variable precedes the verb. On the other models, the USING format applies only to displays which follow the verb and remains in effect until cancelled by another USING verb.

Example:

```
10 A = -123.456
20 PAUSE USING "####.##"; A
30 PAUSE A, USING "####"; A
```

When excuted, this program displays the following:

● PC-1211 -123.45
-123 -123

● PC-1245/1251/1401/1402	-123.45
	-123.45
	-123

Pseudovariables

In this and the following charts the features are simply marked with a 'Y' when the machine has the feature.

	PC-1211	PC-1245 PC-1251 PC-1401 PC-1402	PC-1500	Comments
INKEY\$		Y	Y	
MEM	Y	Y	Y	
PI or π	Y	Y	Y	PC-1211 has only π
TIME			Y	

Numeric Functions

	PC-1211	PC-1245 PC-1251	PC-1401 PC-1402	PC-1500	Comments
ABS	Y	Y	Y	Y	
ACS	Y	Y	Y	Y	
AHC			Y		
AHS			Y		
AHT			Y		
ASN	Y	Y	Y	Y	
ATN	Y	Y	Y	Y	
COS	Y	Y	Y	Y	
CUR			Y		
DEG	Y	Y	Y	Y	
DMS	Y	Y	Y	Y	
EXP	Y	Y	Y	Y	
FACT			Y		
HCS			Y		
HSN			Y		
HTN			Y		
INT	Y	Y	Y	Y	
LN	Y	Y	Y	Y	
LOG	Y	Y	Y	Y	
NOT		Y	Y	Y	
POINT				Y	
POL			Y		
RCP			Y		

APPENDIX G
Feature Comparison

	PC-1211	PC-1245 PC-1251	PC-1401 PC-1402	PC-1500	Comments
REC			Y		PC-1211 has only $\sqrt{\quad}$
RND		Y	Y	Y	
ROT			Y		
SGN	Y	Y	Y	Y	
SIN	Y	Y	Y	Y	
SQR or $\sqrt{\quad}$	Y	Y	Y	Y	
SQU			Y		
STATUS				Y	
TAN	Y	Y	Y	Y	
TEN			Y		

String Functions

	PC-1211	PC-1245 PC-1251 PC-1401 PC-1402	PC1500	Comments
ASC		Y	Y	
CHR\$		Y	Y	
LEFT\$		Y	Y	
LEN		Y	Y	
MID\$		Y	Y	
RIGHT\$		Y	Y	
STR\$		Y	Y	
VAL		Y	Y	

Operators

	PC-1211	PC-1245 PC-1251 PC-1401 PC-1402	PC-1500	Comments
^	Y	Y	Y	
*, /, +, -	Y	Y	Y	
>, >=, =, <>, <=, <	Y	Y	Y	
AND, OR,		Y	Y	
&		Y	Y	

APPENDIX H

Using Program Developed for the PC-1245 or PC-1250/1251

Modifications Required to PC-1245 Programs

While the PC-1401/1402 uses almost the same set of BASIC statements as that used on the PC-1245, the former has some expanded BASIC features. When using the programs developed for the PC-1245 on the PC-1401/1402, it is necessary to modify the following:

1. Multiplication without using the operator " * ":

On the PC-1245, the operator (*) for multiplication may be omitted, such as AB for A * B or CD for C * D. On the PC-1401/1402, the multiplication operator (*) cannot be omitted since the computer treats two consecutive characters, such as AB or CD, as simple variables. Use the specification on the right hand side of the following example:

(e.g.) $A = \sin BC \rightarrow A = \sin (B * C)$

2. Definition of subscripted variables (such as A()) by using the DIM statement:

On the PC-1245, if, for example, DIM A(30) is executed, memory locations for A(27) through A(30) are set aside as an extension of a fixed variable definition area. On the PC-1401/1402, however, the execution of DIM A (30) reserves a separate memory area for array variables A (0) through A (30) for the array named A.

When defining subscripted variables (such as A ()) as an extension of fixed variables, use the specification on the right hand side of the following example:

$\text{DIM A(30)} \rightarrow \text{A(30) = 0}$

3. Data I/O statement for tape files:

On the PC-1245, the execution of, for instance, the PRINT# C statement saves the contents of the variable C and all the subsequent variables to a tape file. On the PC-1401/1402, however, the execution of the same statement saves the contents of the variable C only.

To save the contents of a specific variable and all the subsequent variables, use the specification on the right hand side of the following examples:

(e.g.) $\text{PRINT\# A} \rightarrow \text{PRINT\# A*}$
 $\text{PRINT\# C} \rightarrow \text{INPUT\# C*}$

4. Definition of the \equiv key:

On the PC-1401/1402, the \equiv key cannot be used as a definition assignment key. If it is defined as a definition assignment key in a program, modify the assignment to some other key.

(e.g.) $100 "=" \rightarrow 100 "N":$

APPENDIX H
Using Program

5. Value of a loop variable after completion of a FOR-NEXT loop:

The value of a loop variable obtained after the execution of a FOR-NEXT loop completed on the PC-1401/1402 is different from that obtained on the PC-1245. If the value of a loop variable is used in a conditional expression in a PC-1245 program, increment it by one when it is used on the PC-1401/1402.

(e.g.) 10 FOR I = 0 TO 10

 50 NEXT I

 60 IF I = 10 THEN 100

 Modify the value of I in line 60 as follows:

 60 IF I = 11 THEN 100

(On the PC-1401/1402, the value of a loop variable must be incremented by one step value. The number of loop execution cycles remains the same, however.)

6. Exponent symbol "E":

The PC-1401/1402 uses the uppercase letter "E" for its exponent symbol. The following change is required:

A = 1.234 IE 5 \longrightarrow A = 1.234E5

B = IE 6 \longrightarrow B = 1E6

If a PC-1245 program is read from a tape file into the PC-1401/1402, the change for the exponent symbol described just above will automatically be done by the PC-1401/1402.

7. The character code of the PC-1245 is partially different from that of the PC-1401/1402. When the following codes are designated by the CHR\$ function, change the codes.

<u>Character Code</u>	<u>PC-1245</u>	<u>PC-1401/1402</u>
39 (&27)	□	,
91 (&5B)	√	[
93 (&5D)	π]
96 (&60)	IE	— (Error)
251 (&FB)	— (Error)	π
252 (&FC)	— (Error)	√

Note: As shown above, the PC-1401/1402 does not have the characters □ and IE and the PC-1245 does not have , , [, and] .

Modifications Required to PC-1250/1251

When programs created for the PC-1250/1251 are executed on the PC-1401/1402, the modifications mentioned in previous paragraph for PC-1245 and the following modifications are required for the program:

1. When the total number of columns exceed 16 with a USING format in the PRINT expression ERROR 7 results.

Example: 10 USING "#####.#####"

17 columns or more

20 PRINT A

Note: When the total number of columns specified in the integer part exceed 11 including sign digit, the excess part is ignored. Also when the total number of columns of the integer part specified exceed 11, this integer part is regarded as 11 digits in the PC-1401/1402.

Examples:

- 1) 10: USING "#####."

This format does not cause the error.

- 2) 10: USING "#####.####"

This format does not also cause the error.

Integer part: 11 digits, Decimal point: 1 digit and Decimal part: 4 digits

Total: 16 digits (less than 17 digits)

- 3) 10: USING "#####.####"

This format cause the error 7.

(Total digits are greater than 16 digits.)

- Change the USING format to within 16 columns.

2. When the integer part exceeds 8 columns with a USING format in the PRINT expression, expression, ERROR 7 results.

Example: 10 USING "#####.##"

9 columns or more

20 PRINT A , B

- Change the USING format to within 8 columns.

3. When a message of the form INPUT "....."; A exceeds 15 columns, the beginning part of the message is not displayed.

- Shorten the message.

APPENDIX H

Using Program

4. When the display contents of the form **PRINT expression ; expression ; expression ;** exceeds 16 columns, when the character string of the expression in the form of **PRINT expression , expression** exceeds 8 columns, or when a message of the form **INPUT "." , A** exceeds 16 columns, the excess part is not displayed.
 - Rewrite so that it fit.

Additional modifications

1. The **PC-1245**, **PC-1250**, and **PC-1251** use a line number ranging from 1—999, whereas this model has an extended line number ranging from 1—65279. Therefore, the line number uses 3 bytes in RAM (**PC-1245** series uses 2 bytes). The modification is carried out automatically when the program is loaded through the cassette tape. There is a possibility of memory overflow (ERROR 6) when loading or executing a long program.
2. This model does not have **CHAIN** and **MERGE** commands, as used in **PC-1245** series. Therefore, key entry of these is not possible, but they can be loaded as a program of the **PC-1245** series through the cassette tape. These commands, if used, will be skipped in execution, and displayed by symbol "**~**".
3. In loading a program of the **PC-1245** series through the cassette tape, the computer will remain **BUSY** for one to two seconds after the tape has stopped due to modification of the line number (2 bytes to 3 bytes) as mentioned previously. During this period, symbol "*****" will be displayed at the right most column of the display as in loading a program.
4. For programs created after execution of the **MERGE** command and loaded through the cassette tape, only the first program is read in.

Note: The **PC-1245**, **PC-1250** or **PC-1251** cannot read from a tape which contains programs developed for the **PC-1401/1402**.

APPLICATION EXAMPLES

By reading the explanations of the various functions, you will probably have acquired a good knowledge of what this machine can do. To become competent in BASIC of what this machine can do. To become competent in BASIC programming, it is important to generate your own programs in addition to understanding this manual. It is recommended to get familiar with CALculator mode by actually using the computer. The following pages contain useful examples of BASIC programs and usage examples of the CALculator mode.

(Sharp Corporation and/or its subsidiaries assume no responsibilities or obligations to any losses or damages that could arise through the use of the software programs employed in this instruction manual.)

NOTE: ● Flowcharts have been added for the programs with complicated structures. At the end of each program, we added the number of bytes used for the program.

CONTENTS

(Title)	(page)
● Dual regression analysis	205
● Two way configuration method	210
(c) ● Comparison of mean values	216
● R-L-C circuits impedance	219
(c) ● Pulse circuits	224
● Roots of equations (Newton's method)	226
● Numerical integration (Simpson's method).	230
● Softlanding game.	237
● Typing practice.	240
● Total amount tables.	243
(c) ● Loan Calculation.	249

Note: Titles marked (c) are examples of the usage of CALculator mode.

Program Title: DUAL REGRESSION ANALYSIS

When there is a relationship between some values (x_i, y_i, z_i) $i = 1, \dots, n$ a regression can be done using these values to find the regression equation $z = ax + by + c$. This method can be applied for finding the relationship between height (x), weight (y) and bust (z).

INSTRUCTIONS

- 1) **DEF** **A** : [Data input]
After the printer output or display output has been selected, input the data to x, y, z . Only press **ENTER** for the end of the input when "x=" indicated.
- 2) **DEF** **B** : [Output of the analysis]
Select the printer output or display output. Then the means, variances, correlation coefficient and the regression coefficients are output.
- 3) **DEF** **C** : [Estimation by the regression equation]
After the printer output or display output has been selected, input the data x, y . Then make estimates using the regression coefficient calculated in **DEF** **B**.

Note: When selecting outputs, "P=P" means the display output, and "P=LP" means the printer output.

CONTENTS

Values are calculated as follows;

n = number of pieces of data

$$S_{xx} = \sum (x_i - \bar{x})^2$$

$$S_{yy} = \sum (y_i - \bar{y})^2$$

$$S_{zz} = \sum (z_i - \bar{z})^2$$

$$S_{xy} = \sum (x_i - \bar{x})(y_i - \bar{y})$$

$$S_{yz} = \sum (y_i - \bar{y})(z_i - \bar{z})$$

$$S_{xz} = \sum (x_i - \bar{x})(z_i - \bar{z})$$

Regression coefficient

$$a = \frac{S_{xz} \cdot S_{yy} - S_{yz} \cdot S_{xy}}{S_{xx} \cdot S_{yy} - (S_{xy})^2}$$

$$b = \frac{S_{yz} \cdot S_{xx} - S_{xz} \cdot S_{xy}}{S_{xx} \cdot S_{yy} - (S_{xy})^2}$$

$$c = \bar{z} - a\bar{x} - b\bar{y}$$

To obtain an estimate, use the following formula;

$$z = ax + by + c.$$

Variances

$$V_x = \frac{S_{xx}}{n}$$

$$V_y = \frac{S_{yy}}{n}$$

$$V_z = \frac{S_{zz}}{n}$$

Correlation coefficients

$$r_{xy} = \frac{S_{xy}}{\sqrt{S_{xx}} \sqrt{S_{yy}}}$$

$$r_{yz} = \frac{S_{yz}}{\sqrt{S_{yy}} \sqrt{S_{zz}}}$$

$$r_{zx} = \frac{S_{zx}}{\sqrt{S_{xx}} \sqrt{S_{zz}}}$$

EXAMPLE

Use the following data for regression analysis.

	1	2	3	4	5
x	24	17	22	21	20
y	2300	1500	1600	1100	900
z	9027	5983	5274	4435	3365

Also, after the regression coefficient has been found, estimate z for:

$$x = 17$$

$$y = 600$$

KEY OPERATION PROCEDURE

<Data input>

Step No.	Key Input	Display	Remarks
1	DEF A	P = P --- 1 P = LP --- 2 _	Selection of display output
2	1 ENTER	X =	Data input
3	24 ENTER	Y =	Data input
4	2300 ENTER	Z =	Data input
5	9027 ENTER	X =	Data input
	⋮	⋮	⋮
	⋮	Z =	
17	3365 ENTER	X =	Data input
18	ENTER	>	END

<Result output>

Step No.	Key Input	Display	Remarks
1	DEF B	P = P --- 1 P = LP --- 2 _	Selection of printer output
2	2 ENTER	>	After the results have output to the printer, END

<Estimate>

Step No.	Key Input	Display	Remarks
1	DEF C	P = P --- 1 P = LP --- 2 _	Selection of display output
2	1 ENTER	X =	Data input
3	17 ENTER	Y =	Data input
4	400 ENTER	Z = 2311.149	Indicate the estimation z
5	ENTER	X =	
6	ENTER	>	END

PRINTED OUTPUTS

(DEF B)

MX= 20.800
MY= 1480.000
MZ= 5616.800

VX= 5.360
VY= 233600.000
VZ= 3669666.561

RXY= 0.551005
RYZ= 0.976378
RZX= 0.481886

A= -66.294
B= 4.043
C= 1012.543

(DEF A)

X= 24.
Y= 2300.
Z= 9027.

X= 17.
Y= 1500.
Z= 5983.

X= 22.
Y= 1600.
Z= 5274.

X= 21.
Y= 1100.
Z= 4435.

X= 20.
Y= 900.
Z= 3365.

(DEF C)

X= 17.
Y= 600.
Z= 2311.149

DEF A and DEF C are print samples when printer output is selected.

PROGRAM LISTING

```

10:"A":REM DATA INPUT
15:CLER:WAIT:V=1
20:E=0
30:GOSUB 1000
40:INPUT "X=":X:GOTO 60
50:END
60:INPUT "Y=":Y:"Z=":Z
70:GOSUB 1200
80:F=F+X:G=G+Y:H=H+Z
90:I=I+X*X:J=J+Y*Y:K=K+
  Z*Z
100:L=L+X*Y:M=M+Y*Z:N=N+
  Z*X
110:E=E+1
115:GOSUB 1100
120:GOTO 40
130:END
140:"B":REM MEANS
150:GOSUB 1000
155:GOSUB 1300
160:X=F/E:PRINT "MX= ";X
  +W
170:Y=G/E:PRINT "MY= ";Y
  +W
180:Z=H/E:PRINT "MZ= ";Z
  +W
190:GOSUB 1100
200:O=I-E*X*X:P=J-E*Y*Y:
  Q=K-E*Z*Z
210:R=L-E*X*Y:S=M-E*Y*Z:
  T=N-E*Z*X
220:REM VARIANCES
230:GOSUB 1300
240:PRINT "VX= ";O/E+W
250:PRINT "VY= ";P/E+W
260:PRINT "VZ= ";Q/E+W
270:GOSUB 1100
280:REM CORRELATION COEF
  FICIENT
290:GOSUB 1400
300:PRINT "RXY=":R/SQR (
  O*P)+W
310:PRINT "RYZ=":S/SQR (
  P*Q)+W
320:PRINT "RZX=":T/SQR (
  O*Q)+W
330:GOSUB 1100
340:REM REGRESSION COEFF
  ICIENTS
350:GOSUB 1300
360:D=O*P-R*R
370:A=(T*P-S*R)/D:PRINT
  "A= ";A+W
380:B=(S*O-T*R)/D:PRINT
  "B= ";B+W
390:C=Z-A*X-B*Y:PRINT "C
  = ";C+W

```


MEMORY CONTENTS

```

400:GOSUB 1100
410:END
420:"C":REM ESTIMATION
425:V=2
430:GOSUB 1000:GOSUB 130
    0
440:INPUT "X=";X:GOTO 46
    0
450:END
460:INPUT "Y=";Y
470:Z=A*X+B*Y+C
480:GOSUB 1200
485:GOSUB 1100
490:GOTO 440
500:END
  1000:INPUT "P=P--1 P=LP
    --2 ";U
  1010:IF U=1 PRINT =
    PRINT :GOTO 1040
  1020:IF U=2 PRINT =
    LPRINT :GOTO 1040
  1030:GOTO 1000
  1040:RETURN
  1100:REM SPACE
  1110:IF U=1 GOTO 1130
  1120:IF U=2 PRINT ""
  1130:RETURN
  1200:REM X-Y-Z PRINT
  1210:IF U=1 ON V GOTO 1
    260,1240
  1220:IF U=2 USING
  1230:PRINT "X= ";X:
    PRINT "Y= ";Y
  1235:IF V=1 LET W=0:
    GOTO 1250
  1240:GOSUB 1300
  1250:PRINT "Z= ";Z+W
  1260:RETURN
  1300:REM USING 1
  1310:USING "#####.##
    #":W=5E-4
  1320:RETURN
  1400:REM USING 2
  1410:USING "##.#####":
    E=5E-7
  1420:RETURN
1500:END

```

1151 bytes

A	a
B	b
C	c
D	$S_{xx} * S_{yy} - (S_{xy})^2$
E	n
F	Σx_i
G	Σy_i
H	Σz_i
I	Σx_i^2
J	Σy_i^2
K	Σz_i^2
L	$\Sigma x_i y_i$
M	$\Sigma y_i z_i$
N	$\Sigma z_i x_i$
O	S_{xx}
P	S_{yy}
Q	S_{zz}
R	S_{xy}
S	S_{yz}
T	S_{zx}
U	Output flag
V	Program flag
W	For rounding
X	x, \bar{x}
Y	y, \bar{y}
Z	z, \bar{z}

Program Title: TWO WAY CONFIGURATION METHOD

This configuration method is for analysing results obtained under different conditions and determining whether there is any relationship between the conditions and the results. To do so we need to analyse the variances and calculate the variance ratio. This is a program for a two way configuration method without repetition.

INSTRUCTIONS

- 1) [Data input and check]
 - Input the level number of the elements A, B, then input the data according to the display.
 - After all of the data have entered, select whether you need to check the data or not. If the indicated data is wrong, press and input the correct data.
- 2) [Results output of the analysis of variances]
 - Select whether you need to print out the entered data.
 - Print out the analysis of variances (variation, the number of degrees of freedom, unbiased variance, unbiased variance ratio).

REFERENCE (Calculation subject etc.,)

Number of levels: a, b

Data: $|x_{ij}|$ ($i = 1 \sim a, j = 1 \sim b$)

1. $[X] = (\sum_{ij} x_{ij})^2 / ab$
 $[A] = \sum_i (\sum_j x_{ij})^2 / b$
 $[B] = \sum_j (\sum_i x_{ij})^2 / a$
 $[ABS] = \sum_{ij} x_{ij}^2$
2. $S_A = [A] - [X]$
 $S_B = [B] - [X]$
 $S_T = [ABS] - [X]$
 $S_E = [ABS] - [A] - [B]$
3. $\phi_A = a - 1$
 $\phi_B = b - 1$
 $\phi_E = ab - a - b - 1$
 $\phi_T = ab - 1$
4. $V_A = S_A / \phi_A$
 $V_B = S_B / \phi_B$
 $V_E = S_E / \phi_E$
5. $F_A = V_A / V_E$
 $F_B = V_B / V_E$

EXAMPLE

Examine relations of the quantity of hormone given, kind of food and weight gain from the following data by analysis of variances.

Data: The weight gain of a piglet fed with different quantities of hormone.

		Kind of food		
		B ₁	B ₂	B ₃
Quantity of Hormone (mg/ind)	A ₁ (10)	79	68	93
	A ₂ (20)	86	90	100
	A ₃ (30)	104	110	118

PRINTED OUTPUTS

LEVEL A=3.
B=3.

DATA

X(1,1)=79.

X(1,2)=68.

X(1,3)=93.

X(2,1)=86.

X(2,2)=90.

X(2,3)=100.

X(3,1)=104.

X(3,2)=110.

X(3,3)=118.

VARIATION

A=1432.889

B=401.556

E=115.111

T=1949.556

DEGREES OF FREEDOM

A=2.

B=2.

E=4.

T=8.

UNBIASED VARIANCES

A=716.444

B=200.778

E=28.778

RATIOS OF

UNBIASED VARIANCE

A=24.896

B=6.977

KEY OPERATION PROCEDURE

<Data input and check>

Step No.	Key Input	Display	Remarks
1	DEF A	LEVEL A = —	Number of level A
2	3 ENTER	LEVEL B = —	Number of level B
3	3 ENTER	X(1, 1) =	Data input
		?	
4	79 ENTER	X(1, 2) =	
		?	
5	68 ENTER	X(1, 3) =	
⋮		? ⋮ Input in the same manner	
11	110 ENTER	X(3, 3) =	
		?	
12	118 ENTER	INPUT END	
		CHECK (Y/N) = —	Check entry
13	Y ENTER	X(1, 1) = 79	
14	ENTER	X(1, 2) = 68	
⋮		⋮ Input in the same manner	
	ENTER	X(3, 1) = 100	Since input-miss was found, correct by DEF B
19	DEF B	X(3, 1) = ?	
		?	
20	104 ENTER	X(3, 1) = 104	
⋮	ENTER	⋮ Check continuously	
23	ENTER	>	END

<Output of the analysis of variance>

Step No.	Key Input	Display	Remarks
1	<input type="button" value="DEF"/> <input type="button" value="C"/>	DATA PRINT (Y/N) _	Print the input data?
2	Y <input type="button" value="ENTER"/>		Printout the analysis of variances.
		>	END

PROGRAM LISTING

```

10:"A":CLEAR
20:INPUT "LEVEL A=";A
30:INPUT "LEVEL B=";B
40:DIM O(B-1),B$(0),X(A
    -1,B-1)
50:FOR I=0 TO A-1:FOR K
    =0 TO B-1
60:B$(0)="X(";STR$(I+1
    );",";STR$(K+1);")=
    "
70:PAUSE B$(0):INPUT X(
    I,K):NEXT K:NEXT I:
    GOTO 90
80:GOTO 70
90:PAUSE "INPUT END":B$(
    0)=""
100:INPUT "CHECK (Y/N)="
    ;B$(0)
110:IF B$(0)="N" END
120:IF B$(0)<>"Y" GOTO 1
    00
125:USING
130:I=0
135:J=0
140:PRINT "X(";STR$(I+1
    );",";STR$(J+1);")=
    ";X(I,J)
150:J=J+1:IF J=B GOTO 17
    0
160:GOTO 140
170:I=I+1:IF I=A GOTO 19
    0
180:GOTO 135
190:END
200:"B":PAUSE "X(";STR$(
    I+1);",";STR$(J+1
    );")=?"
210:INPUT X(I,J)
220:GOTO 140
230:END
240:"C":M$="":INPUT "DAT
    A PRINT(Y/N)";M$
250:IF (M$="Y")+(M$="N")
    <>1 GOTO 240
260:IF M$="N" GOTO 300
270:LPRINT "":LPRINT "LE
    VEL A=";A:LPRINT "
    B=";B
280:LPRINT "DATA":FOR I=
    0 TO A-1:FOR K=0 TO
    B-1
290:LPRINT "X(";STR$(I+
    1);",";STR$(K+1);")
    =" ;X(I,K):NEXT K:
    NEXT I
300:FOR J=0 TO B-1:O(J)=
    0:NEXT J
310:T=0:S=0:Z=0:R=0
320:FOR I=0 TO A-1:P=0:
    FOR J=0 TO B-1
330:E=X(I,J):Z=Z+E:E=O(J
    )=O(J)+E:P=P+E:NEXT
    J
340:S=S+P:P=R=R+P:NEXT I
350:R=R*R/(A*B):S=S/B:
    FOR I=0 TO B-1:T=T+O
    (I)*O(I):NEXT I:T=T/
    A
360:S=S-R:LPRINT "":
    LPRINT "VARIATION"
365:B$(0)="" A="":C=S:
    GOSUB 1000
370:T=T-R:Z=Z-R:P=Z-S-T
375:B$(0)="" B="":C=T:
    GOSUB 1000:B$(0)="" E
    ="":C=P:GOSUB 1000:B$
    (0)="" T="":C=Z:GOSUB
    1000
380:F=A-1: LPRINT "":
    LPRINT "DEGREES OF F
    REEDOM":S=S/F:G=B-1
385:B$(0)="" A="":C=F:
    GOSUB 1000:B$(0)="" B
    ="":C=G:GOSUB 1000
390:T=T/G:O=(A-1)*(B-1):
    P=P/O:D=A*B-1
395:B$(0)="" E="":C=O:
    GOSUB 1000:B$(0)="" T
    ="":C=D:GOSUB 1000
400:LPRINT "":LPRINT "UN
    BIASD VARIANCES"
405:B$(0)="" A="":C=S:
    GOSUB 1000:B$(0)="" B
    ="":C=T:GOSUB 1000:B$
    (0)="" E="":C=P:GOSUB
    1000
410:F=S/P:LPRINT "":
    LPRINT "RATIOS OF":
    LPRINT " UNBIASED
    VARIANCE"
415:B$(0)="" A="":C=F:
    GOSUB 1000:B$(0)="" B
    ="":C=T/P:GOSUB 1000
420:END
1000:REM ROUND
1010:LPRINT B$(0); INT
    (C*1E3+.5)/1E3
1020:RETURN
1030:END

```

1365 bytes

MEMORY CONTENTS

A	Number of Level a
B	Number of Leve b
C	$\sqrt{}$
D	$\sqrt{}$
E	$\sqrt{}$
F	ϕ_a, F_a
G	ϕb
H	
I	$\sqrt{}$
J	$\sqrt{}$
K	$\sqrt{}$
L	
M\$	$\sqrt{}$
N	
O	ϕ_e, ϕ_x
P	$\sqrt{}, S_e, V_e$
Q	
R	$\sqrt{}, [X]$
S	$[A], S_a, V_a$
T	$[B], S_b, V_b$
U	
V	
W	
X	
Y	
Z	$[ABS], S_T$
O(*)	$\sum_i x^2_{ij}$
X(*)	Input Data
B\$(\emptyset)	$\sqrt{}$

This example will introduce the joint of CAL mode statistical treatment and BASIC program. Assume there is a correspondence between the samples and each sample difference d_i is a sample of a population which forms a normal distribution. Value t has a degree of freedom $n-1$ and will be determined by

$$t = \frac{\bar{d}}{\frac{S_{\bar{d}}}{\sqrt{n}}}$$

This t will be used for determining whether two sample populations have statistically equivalent mean values or not.

Here, mean value of difference

$$\bar{d} = \frac{\sum (x_i - y_i)}{n} = \frac{\sum x_i - \sum y_i}{n}$$

standard deviation of difference

$$\begin{aligned} S_{\bar{d}} &= \left(\frac{\sum (x_i - y_i)^2 - (\sum (x_i - y_i))^2 / n}{n - 1} \right)^{\frac{1}{2}} \\ &= \left(\frac{\sum x_i^2 - 2 \sum x_i y_i + \sum y_i^2 - n \cdot \bar{d}^2}{n - 1} \right)^{\frac{1}{2}} \end{aligned}$$

- 1) Input data in the statistical mode of the CAL mode first, then, find t value by using the statistical quantity stored in the U-Z memory in BASIC mode.

Memory	U	V	W	X	Y	Z
Statistical quantity	$\sum y^2$	$\sum y$	$\sum xy$	$\sum x^2$	$\sum x$	n

$$\bar{d} = (Y - V) / Z \rightarrow M \text{ (mean value)}$$

$$S_{\bar{d}} = \text{SQR} ((X - 2 * W + U - Z * M^2) / (Z - 1)) \rightarrow SD \text{ (standard deviation)}$$

$$t = M / SD * \text{SQR } Z \rightarrow T \text{ (t value)}$$

2) The program in BASIC mode is executed , and output following;

SD: standard deviation of the difference ($S_{\bar{d}}$)

M: mean value of the difference (\bar{d})

T: t value (t)

DF: Degree of freedom (Df)

3) The examination needs the t distribution table

EXAMPLE

Students in two classes in one grade (10 students in each classes) are assumed to have the same level of intelligence. After one year of giving them two different types of educational methods, the following results were found;

A	10	4	6	2	7	13	3	11	5	9
B	8	3	4	2	3	11	4	7	4	7

With this data, examine the following hypotheses;

H_0 : Both educational methods have been equally effective.

H_1 : The effectiveness of one method is different with 95% confidence.

ANSWER

The rejection area is $|t| > 2$ under the conditions; degrees of freedom 9, significant level 5% (From t distribution tables)

1) Make the BASIC program

```
10:M=(Y-V)/Z
20:SD=SQR ((X-2*W+U-Z*M
    ^2)/(Z-1))
30:T=M/SD*SQR Z
40:DF=Z-1
50:LPRINT "SD=";SD
60:LPRINT "M=";M
70:LPRINT "T=";T
80:LPRINT "DF=";DF
90:END
```

152 bytes

2) Operation

Set statistical mode in CAL mode.

CAL **SHIFT** **STAT**

0

Input Data

10 **{x,y}** 8 **DATA** 4 **{x,y}** 3 **DATA** ...

... 9 **{x,y}** 7 **DATA**

10

Change to BASIC mode and execute the program.

SHIFT **STAT** **BASIC** RUN **ENTER**

>

The following result is printed out.

SD=1.567021236

M =1.7

T =3.43063125

DF=9.

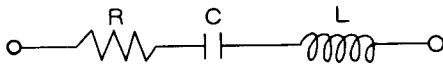
3) Conclusion

Since the t value is in the rejection area, hypothesis H_0 would be rejected. Therefore, there is a difference in the results of different education with 95% confidence.

This program calculates the impedance of a parallel or serial connection of R-L-C circuit elements at a frequency of f .

PROBLEM

1. Serial R-L-C circuit



$$\omega = 2\pi f$$

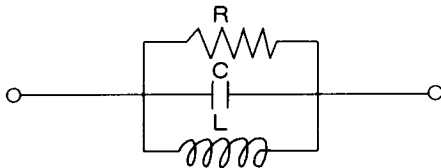
$$Z = R + j\left(\omega L - \frac{1}{\omega C}\right) = x + jy$$

Impedance: $|Z| = \sqrt{R^2 + \left(\omega L - \frac{1}{\omega C}\right)^2}$

Phase: $\theta = \tan^{-1} \left(\frac{\omega L - \frac{1}{\omega C}}{R} \right)$

Then $|Z|$ and θ are obtained by $(x, y) \xrightarrow{\text{POL}} (r, \theta)$ conversion.

2. Parallel R-L-C circuit



$$\omega = 2\pi f$$

$$Z = \frac{1}{\frac{1}{R} + j\left(\omega C - \frac{1}{\omega L}\right)} = x + jy$$

Impedance: $|Z| = \frac{1}{\sqrt{\frac{1}{R^2} + \left(\omega C - \frac{1}{\omega L}\right)^2}}$

Phase: $\theta = \tan^{-1} \left[R \left(\frac{1}{\omega L} - \omega C \right) \right]$

Replace: $\frac{1}{R} + j(\omega C - \frac{1}{\omega L}) = x' + jy'$

and calculate as follows:

$$(x', y') \xrightarrow{POL} (r, \theta) \longrightarrow \left(\frac{1}{r}, -\theta \right) \xrightarrow{REC} (x, y)$$

$$\downarrow \quad \downarrow$$

$$|Z| \quad \theta$$

INSTRUCTIONS

1. **DEF** **A** [The impedance of the serial circuit]
Key-in the values of resistance R (Ω), capacitance C (μF), inductance L (mH), and frequency f (Hz) sequentially. The result of the calculation for x , y , $|z|$, and θ (degree) will be displayed.
2. **DEF** **B** [The impedance of the parallel circuit]
The operation is the same as for **DEF** **A** .

EXAMPLE

Serial circuit

$$\left\{ \begin{array}{l} R = 5 [\Omega] \\ C = 10 [\mu F] \\ L = 25 [\text{mH}] \\ f = 50 [\text{Hz}] \end{array} \right.$$

Parallel circuit

$$\left\{ \begin{array}{l} R = 8 [\Omega] \\ C = 0.5 [\mu F] \\ L = 40 [\text{mH}] \\ f = 60 [\text{Hz}] \end{array} \right.$$

KEY OPERATION PROCEDURE

< Serial circuit >

Step No.	Key Input	Display	Remarks
1	DEF A	SERIAL CIRCUIT	Serial circuit
		R = _	Resistance [Ω]
2	5 ENTER	C = _	Capacitance [μ F]
3	10 ENTER	L = _	Inductance [mH]
4	25 ENTER	F = _	Frequency [Hz]
5	50 ENTER	X	Display for x
6	ENTER	5	
7	ENTER	Y	Display for y
8	ENTER	-310.4559045	
9	ENTER	IMPEDANCE Z	Display for impedance $ Z $
10	ENTER	310.4961653	
11	ENTER	THETA (DEG)	Display for phase θ
12	ENTER	-89.07731137	
13	ENTER	>	END

< Parallel circuit >

Step No.	Key Input	Display	Remarks
1	DEF B	PARALLEL CIRCUIT	Parallel circuit
		R =	Resistance [Ω]
2	8	C =	Capacitance [μF]
3	0.5 ENTER	L =	Inductance [mH]
4	40 ENTER	F =	Frequency [Hz]
5	60 ENTER	X	Display for x
6	ENTER	6.250732478	
7	ENTER	Y	Display for y
8	ENTER	3.306690691	
9	ENTER	IMPEDANCE Z	Display for impedance $ Z $
10	ENTER	7.071482153	
11	ENTER	THETA (DEG)	Display for phase θ
12	ENTER	27.87922142	
13	ENTER	>	END

PROGRAM LISTING

```

10:"A":A=0:PAUSE "SERIAL
   L CIRCUIT":GOTO 30
20:"B":A=1:PAUSE "PARAL
   LEL CIRCUIT"
30:DEGREE :WAIT
40:INPUT "R=":R
50:INPUT "C=":C:C=C*1E-
   6
60:INPUT "L=":L:L=L*1E-
   3
70:INPUT "F=":F
80:W=2*PI*F
90:IF A=1 GOTO 250
100:REM SERIAL
110:U=R:V=W*L-RCP (W*C)
120:Y=POL (U,V)
150:REM PRINT
160:PRINT "X":PRINT U
170:PRINT "Y":PRINT V
180:PRINT "IMPEDANCE Z":
   PRINT Y
190:PRINT "THETA(DEG)":
   PRINT Z
200:END
250:REM PARALLEL
260:Y=RCP R:Z=W*C-RCP (W
   *L)
270:Y=POL (Y,Z):U=RCP Y
280:V=-Z
290:Y=REC (U,V)
300:REM PRINT
310:PRINT "X":PRINT Y
320:PRINT "Y":PRINT Z
330:PRINT "IMPEDANCE Z":
   PRINT U
340:PRINT "THETA(DEG)":
   PRINT V
350:END

```

426 bytes

MEMORY CONTENTS

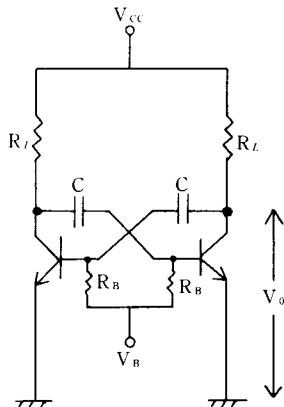
A	Flag	
B		
C	Capacitance	
D		
E		
F	Frequency	
G		
H		
I		
J		
K		
L	Inductance	
M		
N		
O		
P		
Q		
R	Resistance	
S		
T		
U	x	Z
V	y	θ
W	$2\pi f (\omega)$	$2\pi f (\omega)$
X		
Y	Z	x', x
Z	θ	$y' y$

Calculation

Title:

PULSE CIRCUIT

The following equations apply to the astable multivibrator shown as below.



$$T = CR_B \ln \left(\frac{V_{cc} + V_B}{V_{cc}} \right)$$

$$V_0 = V_{cc} \left[1 - \exp \left(-\frac{1}{CRL} t \right) \right]$$

$$tr = -CR_L [\ln(1 - 0.9) - \ln(1 - 0.1)]$$

Where T is the period of the oscillation output, and tr is the rise time of the output pulse.

EXAMPLES

Obtain the period and rise time of the output for the following circuit parameters.

$$V_{cc} = 12 \text{ [V]}$$

$$V_B = 9 \text{ [V]}$$

$$R_L = 1 \text{ [K}\Omega\text{]}$$

$$C = 500 \text{ [pF]}$$

$$R_B = 50 \text{ [K}\Omega\text{]}$$

OPERATION

1. Set the CAL mode and specify the floating point display.
2. Obtain the period T .

500 EXP 12 +/- X 50 EXP 3 X

0.000025

((12 + 9)) ÷ 12) ln =

0.00001399

F↔E

(Switching of the display)

1.39903947 E-05

Answer: Approx 14.0 (μ s)

3. Obtain the rise time T_r .

500 \pm/\mp **EXP** 12 \pm/\mp \times **EXP** 3 \times ((**0.000001098**
 1 \div 0.9 $)$ **ln** $-$ (1 $-$ 0.1 $)$ **ln**
 $)$ **=**

F \leftrightarrow E (Switching of the display) **1.098612289E-06**

Answer: Approx. 1.1 (μs)

OVERVIEW (mathematical)

Finding the roots of equations is usually troublesome, but by using Newton's method the approximate roots of equations can be found.

When 1 root is found, depending on the interval width, by using Newton's method the starting point automatically changes.

CONTENTS

$$X_{n+1} = X_n - \frac{f(X_n)}{f'(X_n)}$$

If the absolute value of the distance between X_n and X_{n+1} is less than 10^{-8} , X_n is considered a root and is displayed. Here the first derivative is defined in the following way:

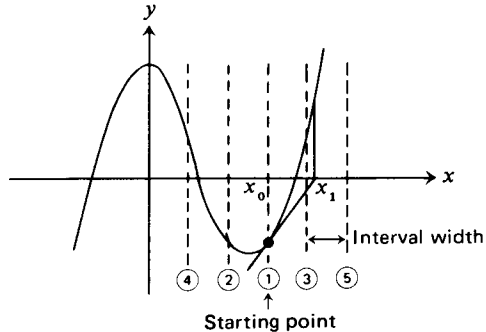
$$f'(X) = \frac{f(X+h) - f(X)}{h} \quad (h \text{ is the minute interval})$$

Change 1E-8 in line 340 to change the value for 10^{-8} .

INSTRUCTIONS

INPUT

Starting point
Minute interval
Interval width



OUTPUTS

Root value (by pressing the **ENTER** key, the next interval's root is found)

EXAMPLE

$$x^3 - 2x^2 - x + 2 = 0 \quad (\text{the roots are } -1, 1, 2)$$

starting point = 0

minute interval = 10^{-4}

interval = 0.5

The above values are used in the calculation.

The functions are to be written into lines after 500 as subroutines.

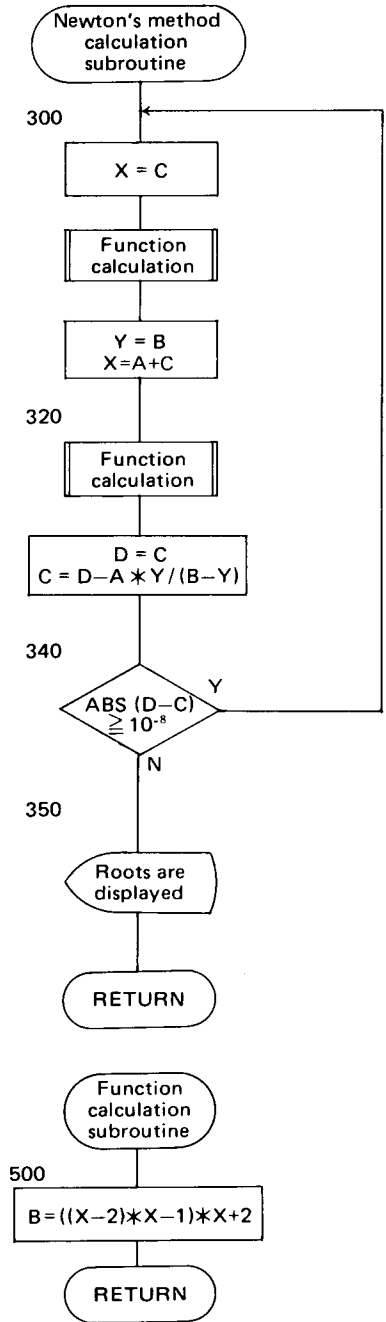
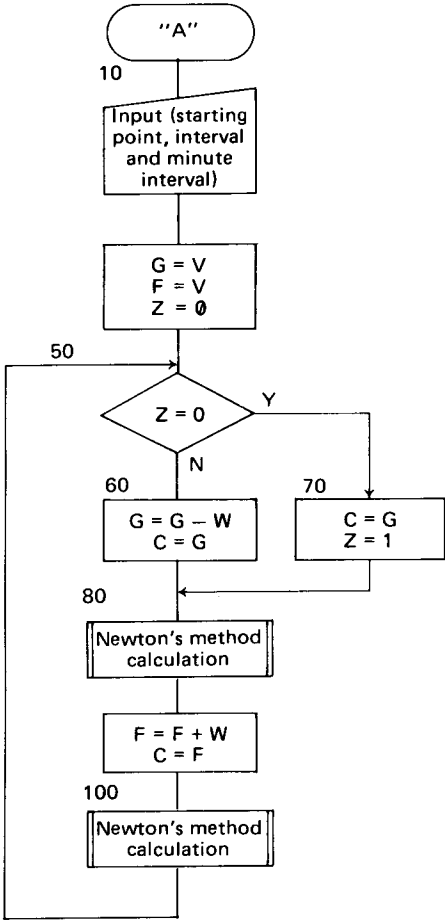
How to type in the example:

1. Go into PRO mode by operating the mode change key.
2. $500B = ((X-2) * X-1) * X+2$ **ENTER**
 510 RETURN **ENTER** That is all that had to be done.

KEY OPERATION PROCEDURE

Step No.	Key Input	Display	Remarks
1	DEF A	START POINT = _	Waiting for starting point input
2	0 ENTER	MINUTE INTV. = _	Waiting for minute interval input
3	0.0001 ENTER	INTERVAL = _	Waiting for interval width input
4	0.5 ENTER		2. Display of roots
5	ENTER		1. By repeatedly pressing the ENTER key the roots of the function are found.
6	ENTER	-1.	
7	ENTER	1.	
8	ENTER	-1.	
9	ENTER	-1.	
10	ENTER	-1.	
11	ENTER	2.	
	:	:	
	:	:	

FLOWCHART



PROGRAM LISTING

```

10:"A":INPUT "START POINT=";V
20:INPUT "MINUTE INTERVAL=";A
30:INPUT "INTERVAL=";W
40:G=V:F=V:Z=0
50:IF Z=0 GOTO 70
60:G=G-W:C=G:GOTO 80
70:C=G:Z=1
80:GOSUB 300
90:F=F+W:C=F
100:GOSUB 300
110:GOTO 50
120:END
300:X=C:GOSUB 500
310:Y=B:X=A+C
320:GOSUB 500
330:D=C:C=D-A*Y/(B-Y)
340:IF ABS (D-C)>=1E-8
    GOTO 300
350:BEEP 3:PRINT C
360:RETURN
500:B=((X-2)*X-1)*X+2
510:RETURN

```

275 bytes

MEMORY CONTENTS

A	Minute interval
B	$f(x)$
C	X_0
D	$f(x+h)$
E	
F	✓
G	✓
H	
I	
J	
K	
L	
M	
N	
O	
P	
Q	
R	
S	
T	
U	
V	Starting point
W	Interval width
X	x
Y	$f(x)$
Z	Initial flag

OVERVIEW

Numerical Integration is done on function values given at equal interval widths of the integration interval.

If the function equation is written in the program the values in the intervals of integration are automatically given.

CONTENTS (Calculation contents)

1. Data are input and integration carried out.

Simpson's 1/3 formula splits the interval $[a, b]$ into n smaller intervals.

The values of the function over the smaller intervals are approximated in 2's ($2i, 2i+1$ units) by using a 2nd order equation to approximate the curve.

After the data (function values) for the smaller intervals are input, the integrated values are printed.

$$\begin{aligned}
 \int_a^b f(x) dx &\doteq \sum_{i=0}^{N/2-1} \int_{x_{2i}}^{x_{2i+2}} P_2^i(x) dx \\
 &= \sum_{i=0}^{N/2-1} I_i \\
 &\doteq \frac{h}{3} (y_0 + 4y_1 + 2y_2 + 4y_3 + \dots \\
 &\quad \dots + 4y_{n-1} + y_n) \\
 &\left\{ \begin{array}{l} h = \frac{b-a}{n} \\ I_i = \frac{h}{3} (y_{2i} + 4y_{2i+1} + y_{2i+2}) \end{array} \right.
 \end{aligned}$$

2. Integration using the function equation written in the program

Using the input function equation as a base, the intervals $[a, b]$ are split up into n smaller intervals and the function values are calculated and printed. The integrated values are also printed.

INSTRUCTIONS

- Using **[DEF] [A]**, the program is started and a selection of either the data input method or the function equation input method of calculation has to be done. The integration interval's starting point, ending point, and number of divisions has to be input.

The data input method of calculating:

as the integration intervals are being input the data is printed.

The function equation input method of calculating:

the function values are printed according to the function equation.

- The **[DEF] [B]** corrects the input data as required. Enter revision number and revision value.

3. **DEF** **C** The integration values are printed according to the integration interval data (function values).

(Note) The number of divisions is from 2 to 254 and even.

EXAMPLE

1. The data is input and the calculations are done

interval [0, 5], 40 divisions

$f(x\ 0)$	4	$f(x11)$	-7	$f(x22)$	-2	$f(x33)$	13
$f(x\ 1)$	5.5	$f(x12)$	-8	$f(x\ 23)$	0	$f(x34)$	12.5
$f(x\ 2)$	6	$f(x13)$	-9	$f(x\ 24)$	2	$f(x35)$	12
$f(x\ 3)$	5.7	$f(x14)$	-9.5	$f(x\ 25)$	4	$f(x36)$	10.5
$f(x\ 4)$	5	$f(x15)$	-10	$f(x\ 26)$	6	$f(x37)$	9
$f(x\ 5)$	2	$f(x16)$	-9.5	$f(x\ 27)$	7	$f(x38)$	7.8
$f(x\ 6)$	0	$f(x17)$	-9	$f(x\ 28)$	8	$f(x39)$	6
$f(x\ 7)$	-1.8	$f(x18)$	-8.5	$f(x\ 29)$	9.7	$f(x40)$	4
$f(x\ 8)$	-3	$f(x19)$	-7	$f(x\ 30)$	11		
$f(x\ 9)$	-5	$f(x20)$	-5.5	$f(x\ 31)$	12		
$f(x10)$	-6	$f(x21)$	-4	$f(x\ 32)$	12.5		

2. The function equation is stored in the program and the calculations are done.

$$Y = ((X-2)X-1)X+2$$

interval [0, 1] 20 divisions

The function is stored after line 900 as a subroutine.

How to store into the program (for the case of the example)

(Put in PRO mode)

900 $Y = ((X-2)*X-1)*X+2$ **ENTER**

910 RETURN **ENTER** this ends the input

PRINTED OUTPUTS

A=0.
B=5.
N=40.

1 F(0)=4.
2 F(0.125)=5.5
3 F(0.25)=6.
4 F(0.375)=5.7
5 F(0.5)=5.
6 F(0.625)=2.
7 F(0.75)=0.
8 F(0.875)=-1.8
9 F(1)=-3.
10 F(1.125)=-5.
11 F(1.25)=-6.
12 F(1.375)=-7.
13 F(1.5)=-8.
14 F(1.625)=-9.
15 F(1.75)=-9.5
16 F(1.875)=-10.
17 F(2)=-9.5
18 F(2.125)=-9.
19 F(2.25)=-8.5
20 F(2.375)=-7.
21 F(2.5)=-5.5
22 F(2.625)=-4.
23 F(2.75)=-2.
24 F(2.875)=0.
25 F(3)=2.
26 F(3.125)=4.
27 F(3.25)=6.
28 F(3.375)=7.
29 F(3.5)=8.
30 F(3.625)=9.7
31 F(3.75)=11.
32 F(3.875)=12.
33 F(4)=12.5
34 F(4.125)=13.
35 F(4.25)=12.5
36 F(4.375)=12.
37 F(4.5)=10.5
38 F(4.625)=9.
39 F(4.75)=7.8
40 F(4.875)=6.
41 F(5)=4.

F=8.291666667

A=0.
B=1.
N=20.

1 F(0)=2.
2 F(0.05)=1.945125
3 F(0.1)=1.881
4 F(0.15)=1.808375
5 F(0.2)=1.728
6 F(0.25)=1.640625
7 F(0.3)=1.547
8 F(0.35)=1.447875
9 F(0.4)=1.344
10 F(0.45)=1.236125
11 F(0.5)=1.125
12 F(0.55)=1.011375
13 F(0.6)=0.896
14 F(0.65)=0.779625
15 F(0.7)=0.663
16 F(0.75)=0.546875
17 F(0.8)=0.432
18 F(0.85)=0.319125
19 F(0.9)=0.209
20 F(0.95)=0.102375
21 F(1)=0.

F=1.083333333

KEY OPERATION PROCEDURE

<When the values are input>

Step No.	Key Input	Display	Remarks
1	DEF A	INP. = 1/CAL. = 2 ? _	Selection of data input or equation input
2	1 ENTER	A = _	Waiting for the starting point of the integration interval
3	0 ENTER	B = _	Waiting for the ending point of the integration interval
4	5 ENTER	N = _	Waiting for the integration interval division number
5	40 ENTER	F (0) =	Waiting for the data to be input
		?	
6	4 ENTER	F (0.125) =	
		?	
7	5.5 ENTER	F (0.25) =	
		?	
	⋮	⋮	
	6 ENTER	F (5) =	
		?	
	4 ENTER	>	

<Data revision>

Step No.	Key Input	Display	Remarks
1	DEF B	REVISION NO. ? = _	Waiting for the revision number input
2	3 ENTER	NEW VALUE = _	Waiting for the revision value input
3	6 ENTER	REVISION NO. ? = _	
4	ENTER	ALL PRINT (Y/N) _	
5	Y ENTER	>	All of the data printed out
	N ENTER	>	Finished without printing

(to be continued to **DEF** **C**)

KEY OPERATION PROCEDURE

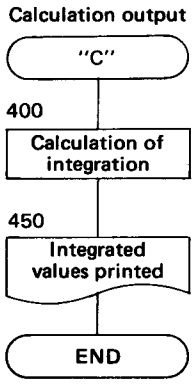
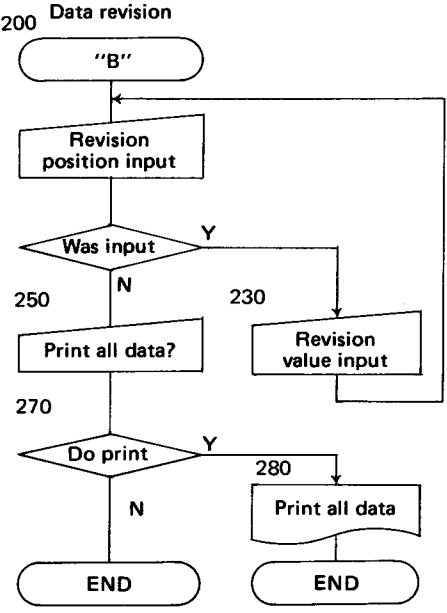
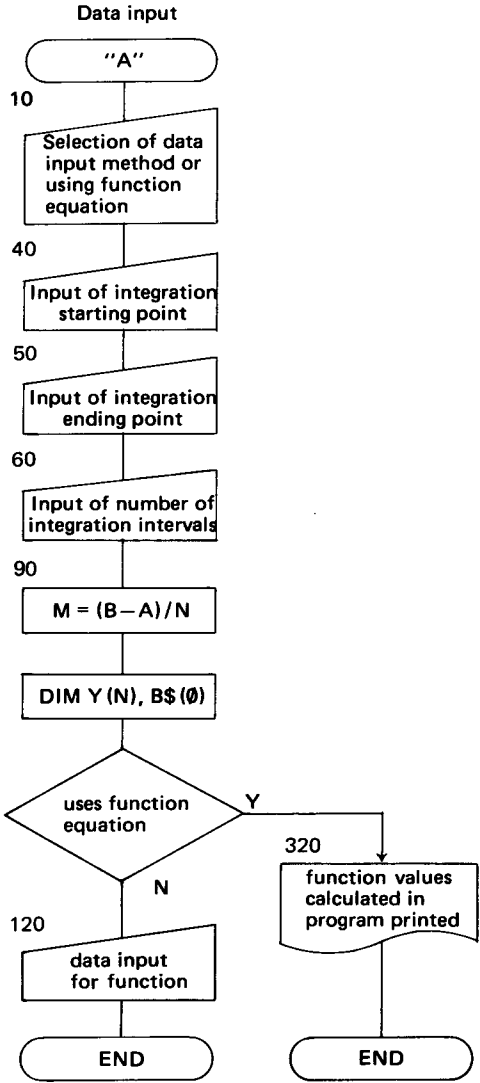
<When the function equation is used>

Step No.	Key Input	Display	Remarks
1	DEF A	INP. = 1/CAL. = 2 ? _	Waiting for the data input method or the equation input method of calculation
2	2 ENTER	A = _	Waiting for the integration starting point input
3	0 ENTER	B = _	Waiting for the integration ending point input
4	1 ENTER	N = _	Waiting for the integration interval division number
5	20 ENTER		Prints the calculations done to find the function values using the function equation as the base
		>	

<Final procedure in both cases>

Step No.	Key Input	Display	Remarks
1	DEF C		Integrated values printed
		>	

FLOWCHART



PROGRAM LISTING

```
10:"A":CLEAR :WAIT 0:
  INPUT "INP.=1/CAL.=2
  ?":I0:GOTO 30
20:END
30:IF (I0=1)+(I0=2)<>1
  GOTO 10
40:INPUT "A=":A:LPRINT
  "A=":A
50:INPUT "B=":B:LPRINT
  "B=":B
60:INPUT "N=":N
70:IF N/2<>INT (N/2)
  GOTO 50
80:LPRINT "N=":N
90:M=(B-A)/N:DIM Y(N):B
  $(0):L=A:IF I0=2 GOTO
  320
100:FOR I=0 TO N:B$(I)="
  F("STR$ L+")="
110:PAUSE B$(0)
120:INPUT Y(I):LPRINT
  USING "###":I+1:" "
  B$(I): USING Y(I):L
  =L+M:GOTO 140
130:GOTO 110
140:NEXT I
150:END
200:"B":INPUT "REVISION
  NO.?" :H:GOTO 220
210:GOTO 250
220:IF (H<0)+(H>N+1)=1
  GOTO 200
230:INPUT "NEW VALUE ?="
  Y(H-1):LPRINT USING
  "###":H:" F("STR$ (
  A+M*(H-1)):"=":
  USING Y(H-1):GOTO 2
  00
240:GOTO 230
250:INPUT "ALL PRINT (Y/
  N)?" :W$
260:IF (W$="Y")+(W$="N")
  <>1 GOTO 250
270:IF W$="N" END
280:LPRINT "":LPRINT "A=
  ":A:LPRINT "B=":B:
  LPRINT "N=":N
290:M=(B-A)/N:L=A:FOR I=
  0 TO N
300:LPRINT USING "###":(
  I+1):" F("STR$ L:")
  =" : USING Y(I)
310:L=L+M:NEXT I:END
320:LPRINT "":FOR I=0 TO
  N:X=L:GOSUB 900
330:LPRINT USING "###":(
  I+1):" F("STR$ L:")
  =" : USING Y
340:Y(I)=Y:L=L+M:NEXT I:
  END
400:"C":S=Y(0):L=S:FOR I
  =0 TO N:IF S>Y(I)
  LET S=Y(I)
```

```
410:NEXT I
420:S=Y(0)+Y(N)
430:FOR I=1 TO N-1:IF I/
  2<>INT (I/2) LET K=
  4:GOTO 450
440:K=2
450:S=S+K*Y(I):NEXT I:S=
  S*M/3:LPRINT "":
  LPRINT "F=":S:END
900:Y=((X-2)*X-1)*X+2
910:RETURN
1000:END
```

859 bytes

MEMORY CONTENTS

A	Integration interval starting point
B	Integration interval ending point
C	
D	
E	
F	
G	
H	✓
I	✓
J	
K	Term coefficient during integration
L	✓, Max. value of Y(i)
M	h
N	Number of divisions
O	Flag for inputting or using equation
P	
Q	
R	
S	✓, Min. value of Y(i), Integration value
T	
U	
V	
W\$	✓
X	✓ Function equation X
Y	Function equation Y
Z	
Y (N)	Input data (function value)
B\$ (0)	✓

OVERVIEW

This game involves landing a rocket, with only a limited amount of fuel, as softly as possible. The rocket is in free fall. The engine is used to slow down the free falling rocket. If ignition takes place too soon or too much fuel is used, then the rocket is thrust back out into space and becomes dust around the planet.

If all the fuel is burned up, the rocket hits the planet and blows up.

The aim is to land the rocket as softly as possible by controlling the engines while watching how much fuel is burned.

CONTENTS

Gravity is set to be $5 \text{ m}/(\text{unit time})^2$.

If 5 units of fuel per a unit time are burnt, then gravity is offset.

Equations

$$H = H_0 + V_0 t + \frac{1}{2} a t^2$$

$$V = V_0 + a t$$

$$V^2 = V_0^2 + 2aH$$

$$H_0 = 500, \quad V_0 = -50, \quad F_0 = 200$$

H : height

V : speed

a : gravitational
acceleration

t : time

V_0 : initial speed

H_0 : initial height

V_0 : initial speed

F_0 : initial fuel

The initial height, initial fuel level, and the wait time is stored in line 30 as data. By changing these values the above variables can be changed.

INSTRUCTIONS

1. It is started by pressing **DEF** **A** . Press **0** ~ **9** keys to adjust the amount of fuel used to land the rocket.
2. When all the fuel is burned up, "FALLING" is displayed.

KEY OPERATION PROCEDURE

Step No.	Key Input	Display	Remarks
1	<input type="button" value="DEF"/> <input type="button" value="A"/>	***START***	
2	Keys <input type="button" value="0"/> ~ <input type="button" value="9"/> designate fuel burned in unit time	500 -50 200 0	Height, speed, fuel left and fuel burnt in unit time are displayed.
	<input type="button" value="9"/>	452 -46 191 9	
	⋮	⋮	
	⋮	Repeat	
	⋮	⋮	
	(If successful)	SUCCESS !!	
		FUEL F = 15	
	(If failed)	GOOD BYE!!	
		REPLAY (Y/N)?	Wait for input on whether you wish to play again
	<input type="button" value="Y"/>		Play again
	<input type="button" value="N"/>	>	End

PROGRAM LISTING

```

10:"A":WAIT 50:CLEAR :
   USING :S=-50:A=0:D#=
   ""
20:BEEP 3:PRINT " ***
   START ***"
30:DATA "WAIT=",50,"FUE
   L=",200,"HEIGHT=",50
   0
40:RESTORE
50:READ B$,W,B$,F,B$,H
60:WAIT W
70:IF F=0 IF S<0 PRINT
   USING "####";H;S;" F
   ALLING":GOTO 90
80:PRINT USING "####";H
   ;S;F;C
90:BEEP 1:D$="":D#=
   INKEY$
100:IF D$="" LET C=A:
   GOTO 130
110:C=VAL D$
120:A=C
130:IF C>F LET C=F
140:F=F-C:X=C-5:H=H+S+X/
   2:S=S+X
150:IF H>5 GOTO 70
160:IF H>0 AND ABS S<5
   BEEP 5:PRINT "SUCCES
   !!":GOTO 180
170:BEEP 3:PRINT "GOOD B
   Y !!":GOTO 190
180:WAIT 150:PRINT USING
   "####";"FUEL F=";F
190:WAIT 50:PRINT "REPLA
   Y (Y/N) ?":Z$=INKEY$
200:IF Z$="Y" OR Z$="N"
   GOTO 220
210:GOTO 190
220:IF Z$="Y" GOTO 10
230:END

```

457 bytes.

MEMORY CONTENTS

A	✓
B\$	✓
C	Fuel burned
D\$	Fuel burned
E	
F	Initial fuel level, fuel left
G	
H	Initial height, height
I	
J	
K	
L	
M	
N	
O	
P	
Q	
R	
S	Speed
T	
U	
V	
W	Wait time
X	✓
Y	
Z\$	✓

OVERVIEW

Quick key operation!

How fast and accurate is your typing?

If you practice with this program, it will make programming much easier for you. Improve your skill!

CONTENTS (such as calculation contents)

The number of characters (4 ~ 6) is randomly chosen.

The character arrangement (A ~ Z) is done randomly.

The allotted time depends on the number of characters and the grade level.

3 is the shortest time allotment while 1 is the longest.

INSTRUCTIONS

After the buzzer sounds 4 to 6 characters will be displayed. You are to type in the same characters within the allotted time.

If they are all correct, you get 10 coins.

If more than half are correct, you get 5 points.

After the allotted time is over, the next problem is displayed. The allotted time depends on the grade, which has three levels (1, 2, 3).

3 is the shortest time allotment while 1 is the longest.

Point competition is done within the same grade category.

There are 10 problems, making the maximum score 100 points.

KEY OPERATION PROCEDURE

Step No.	Key Input	Display	Remarks
1	DEF Z	GRADE (1, 2, 3)?	Grade input
2	1 ENTER	A Z B D C	
3	A	A Z B D C A	
4	Z	A Z B D C A Z	
	⋮	⋮	
	⋮	YOUR – SCORE = 80	After the 10 questions are answered the score is displayed
		YOUR SCORE BEST	If your score is higher than the high score the guidance is displayed
		>	END
1	DEF A	HIGH–SCORE=80	When you want to play in the same grade
		B W V S	
2	B	B W V S B	
	⋮	⋮	
	⋮	YOUR – SCORE = 60	
		>	END

PROGRAM LISTING

```

10:"Z":CLEAR:DIM B$(5)
,C$(5):RANDOM
15:INPUT "GRADE(1,2,3)?
":L:WAIT 0
17:IF (L=1)+(L=2)+(L=3)
<>1 THEN 15
18:GOTO 30
20:"A":WAIT 0:P=0:PAUSE
"HIGH-SCORE=";X
30:FOR S=1 TO 10
40:B=RND 4+2:Y$="":R=
INT (B/2)
50:FOR C=0 TO B-1:C$(C)
=" "
60:D=RND 26:B$(C)=CHR$
(D+640):Y$=Y$+CHR$ (
D+640):NEXT C:A$=""
70:BEEP 3:E=0:WAIT 30:
USING "#####&&"
80:FOR W=1 TO B*10/L:
PRINT Y$:A$:IF E=B
LET W=B*20/L:GOTO 10
0
85:C$(E)=INKEY$:IF C$(
E)=" " THEN 100
87:A$=A$+C$(E)
90:E=E+1
100:NEXT W:Q=0
110:FOR W=0 TO B-1:IF B$
(W)=C$(W) LET Q=Q+1
120:NEXT W:IF Q<=R THEN
150
130:IF Q=B LET P=P+10:
GOTO 150
140:P=P+5
150:NEXT S:USING :BEEP 3
:PAUSE "YOUR-SCORE="
;P
160:IF P>X LET X=P:WAIT
100:PRINT "YOUR SCOR
E BEST"
170:END

```

491 bytes

MEMORY CONTENTS

A\$	✓
B	✓
C	Loop counter
D	✓
E	✓
F	
G	
H	
I	
J	
K	
L	Grade
M	
N	
O	
P	Score
Q	✓
R	✓
S	Loop counter
T	
U	
V	
W	Loop counter
X	High score
Y\$	✓
Z	
B\$ (5)	✓
C\$ (5)	✓

Program Title: TOTAL AMOUNT TABLES

This is a very handy and easy total sum program. Input codes or item name at random and correspond data, and find the sum and percentage for each code (item), then print the total sum at the end.

INSTRUCUCIONS

- 1) **DEF** **Z** : [Registration of item names]
Input the titles in the order of their codes.
- 2) **DEF** **A** : [Input sum data]
By inputting the code numbers, the name of the items will be indicated, then input data. If the code number is unknown, input the item name with "*" at top and the code number will be found.
- 3) **DEF** **S** : [Print out of summation results]
Each of the totals for codes (items) will be summed up and the total result and percentage will be indicated. First input wheter the percentage needs to be printed-out or not.

EXAMPLE

Code table

Code	Item
1	PC-1245
2	PC-1251
3	PA-7050
4	EL-331
5	EL-332
6	WN-106
7	CT-660
8	EL-550
9	CS-2302
10	EA-11E
11	EA-150
12	EA-850C

Input data

Code	Price	Quantity	Sum
4	5800	3	-----
3	----	--	39,800
8	14,800	15	-----
9	19,800	20	-----
11	3,300	40	-----
12	800	18	-----
8	14,800	20	-----
7	-----	--	178,000
8	-----	--	74,000
3	-----	--	597,000
12	800	99	-----
2	29,800	4	-----

Remark: The prices have not concernd with actual prices.

PRINT OUTPUT

Example with percentage

1	PC-1245		
	0	0.00%	
2	PC-1251		
	119200	5.51%	
3	PA-7050		
	636800	29.41%	
4	EL-331		
	17400	0.80%	
5	EL-332		
	0	0.00%	
6	WN-106		
	0	0.00%	
7	CT-660		
	178000	8.22%	
8	EL-550		
	592000	27.34%	
9	CS-2302		
	396000	18.29%	
10	EA-11E		
	0	0.00%	
11	EA-150		
	132000	6.10%	
12	EA-850C		
	93600	4.32%	
TOTAL		2165000	

Example without percentage

1	PC-1245	0
2	PC-1251	119200
3	PA-7050	636800
4	EL-331	17400
5	EL-332	0
6	WN-106	0
7	CT-660	178000
8	EL-550	592000
9	CS-2302	396000
10	EA-11E	0
11	EA-150	132000
12	EA-850C	93600
TOTAL		2165000

KEY OPERATION PROCEDURE

< Registration of the item names >

Step No.	Key Input	Display	Remarks
1	DEF Z	1	
		ITEM NAME = _	Waiting for the input of code No. 1
2	PC-1245 ENTER	2	
		ITEM NAME = _	Waiting for the input of code No. 2
3	PC-1251 ENTER	3	
	⋮	⋮ Input in the same manner	
13	EA-850C ENTER	13	
		ITEM NAME = _	
14	ENTER	>	Finish by pressing the ENTER key only.

< Data input >

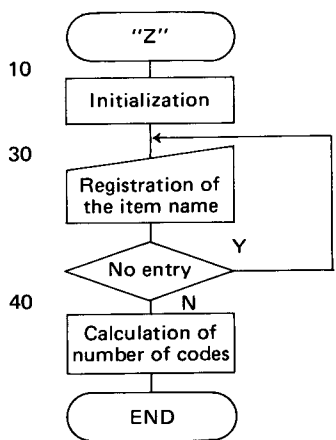
Step No.	Key Input	Display	Remarks
1	DEF A	SUMMATION	Title indication
		CODE = _	
2	4 ENTER	EL-331	
		DATA = _	
3	5800 * 3 ENTER	CODE = _	
4	3 ENTER	PA-7050	
		DATA = _	
5	39800 ENTER	CODE = _	
6	* EL-550 ENTER	8	Input this way if the code number is unknown
		DATA = _	
⋮		⋮ Input in the same manner	
25	29800 * 4 ENTER	CODE = _	
26	ENTER	>	Finish by pressing the ENTER key only.

< Print out the results >

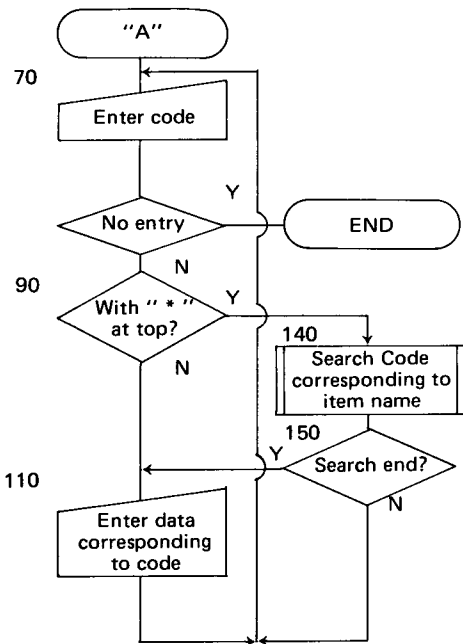
Step No.	Key Input	Display	Remarks
1	DEF S	SUMMATION LIST	Title display
		PERCENT. (Y/N) _	
2	Y ENTER		Output the summation result with percentage
	N ENTER		Output the summation result without percentage

FLOWCHART

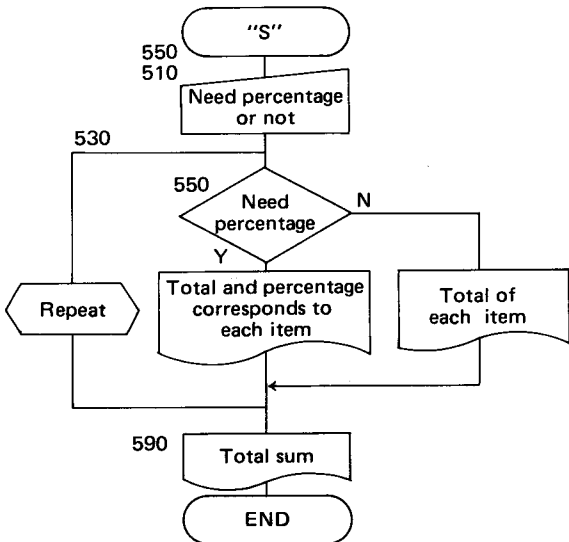
Registration of item name



Summation



Total sum table



PROGRAM LISTING

```

10:"Z":CLEAR :DIM C$(50
)*10,D(50)
20:I=0
30:I=I+1:PAUSE STR$ I:
  BEEP 1:INPUT "ITEM N
  AME=":C$(I):GOTO 30
40:N=I-1
50:END
60:"A":PAUSE "SUMMATION
  "
70:INPUT "CODE=":C$(0):
  GOTO 90
80:END
90:IF LEFT$ (C$(0),1)="
  *" GOTO 140
100:I=VAL C$(0):PAUSE C$
  (I)
110:D=0:INPUT "DATA=":D
120:D(I)=D(I)+D:E=E+D
130:GOTO 70
140:GOSUB 300
150:IF G=0 PAUSE "NO REG
  ISTRATION":GOTO 70
160:I=G:PAUSE STR$ I
170:GOTO 110
300:F=LEN C$(0)-1:C$(0)=
  RIGHT$ (C$(0),F):G=0
310:FOR I=1 TO N
320:IF C$(0)=LEFT$ (C$(I
  ),F) LET G=I:I=N
330:NEXT I
340:RETURN
500:"S":PAUSE "SUMMATION
  LIST"
510:INPUT "PERCENT.(Y/N)
  ":C$
520:IF (C$="Y")+(C$="N")
  <>1 GOTO 510
530:FOR I=1 TO N
540:LPRINT USING "###":I
  ;"  ":C$(I)
550:IF C$="N" LPRINT "
  ":USING "#
  #####":D(I):
  GOTO 580
560:H=D(I)/E*100:H= INT
  (H*100+.5)/100

```

```

570:LPRINT "  ":USING
  "#####":D(I);"
  ":USING "###.##":H
  ;"%
580:NEXT I
590:LPRINT "":LPRINT "TO
  TAL  ":USING "#
  #####":E
600:END

```

660 bytes

MEMORY CONTENTS

A	
B	
C\$	✓
D	Price (Input Data)
E	Total
F	✓
G	
H	Percentage
I	✓
J	
K	
L	
M	
N	Number
O	
P	
Q	
R	
S	
T	
U	
V	
W	
X	
Y	
Z	
C\$(50)*10	Item name
D(50)	Price of each item
C\$(0)	Code No.

Calculation
Title:

LOAN CALCULATION

For most people, expensive goods are too difficult to buy directly, so here we show you a simple loan calculation.

$$P = (PT - R) \times \frac{i}{1 - \frac{1}{(1+i)^n}}$$

Here P : loan rate

PT : price

R : deposit

n : number of payments

i : monthly interest

EXAMPLE

What is the monthly repayment when buying an NC machine costing 550,000 with a deposit of 100,000, a monthly interest of 1.5% and 20 payments?

Answer: $PT = 550,000$

$R = 100,000$

$n = 20$ times

$i = 1.5\% = 0.015$

Calculate the required money as follows;

(550000 - 100000) × 0.015
÷ (1 - (1 + 0.015)²⁰) =

26210.58114

Therefore the monthly repayment is approximately 26,210.

INDEX

&	70	CE-126P	96
*	79	CHR\$	171
+	79	CLEAR	121
-	79	CLOAD	108
/	79	CLOAD?	109
^	79	Clear key	19, 187
$\sqrt{\quad}$	169	CONT	110
<	80	COS	164
◀	49	CSAVE	111
<=	80	CUR	165
<>	80	Cursor	11
=	80	Cassette tape	104
>	80	Commands	86, 105, 108
▶	49	DATA	122
>=	80	DEF key	94
π	162	DEG	165
↑	88	DEGREE	123
↓	88	DELeTe key	89
A() variables	77	DIM	124
ABS	163	Direct calculation feature	68
ACS	163	DMS	165
AHC	163	Debugging	174
AHS	164	Display	22
AHT	164	END	126
ALL RESET	12	ENTER key	49
AND	81	EXP	165
AREAD	118	Edition calculations	49
ASC	171	Editing programs	88
ASCII	178	Error Messages	176
ASN	164	Expressions	79
ATN	164	FACT	166
Array variables	74	Fixed variables	72
Auto off (Auto Power Off)	17	FOR...TO...STEP	127
BASIC key	9	Formatting output	180
BASIC mode	10	Functions	83, 107, 162
BEEP	120	GOSUB	129
Batteries	14	GOTO	112, 130
CA key	187	GRAD	131
CAL key	9	Hard cover	6
CAL mode	9, 18	HCS	166

HSN	166	PAUSE	144
HTN	166	PI	162
IF...THEN	132	PRINT	146
INKEY\$	162	PRINT#	148
INPUT	133	PROgram mode	9
INPUT#	135	Parenthesis	30, 56, 83
INSert key	51, 89	POL	167
INT	166	Printer	96
LEFT\$	171	Priority (CAL mode)	32
LEN	171	Program	85
LET	138	Pseudovariables	162
LIST	113	RADIAN	150
LLIST	114	RANDOM	151
LN	167	RCP	167
LOG	167	READ	152
LPRINT	139	REC	168
Labelled programs	94	Relational expression	80
Last answer feature	60	REM	153
Limits of numbers	60	RESET	12
Line numbers	85	RESTORE	154
Linear regression	42	RETURN	155
Logical expressions	81	RIGHT\$	172
MEM	162	RND	168
MID\$	172	RUN	117
Maintenance	175	RUN mode	9
Manual calculation	48	Range of numbers	60
Masks	180	Relational expressions	80
Memory Protection	93	ROT	169
NEW	115	Scientific notation	59
NEXT	141	SGN	169
NOT	81	SHIFT key	18
Numeric expression	79	Simple variable	73
Numeric function	163	SIN	169
Numeric variables	72	SQR	169
ON (Start up)	17	SQU	169
One-variable statistics	41	STOP	156
ON...GOSUB	142	STR\$	172
ON...GOTO	143	Statements	85
OR	81	Statistical calculation	38
Operator priority (BASIC) mode	185	String expressions	79
Operators	79	String function	171
P ↔ NP	96	String variables	72
PASS	116	Subroutines	129

Index

TAN	170	Two-variable statistics	42
TEN	170	USING	159
TROFF	157	VAL	172
TRON	158	Variables	71
Tape Recorder	98	Verbs	85, 106, 118
Template	94	WAIT	161
Troubleshooting	173		

SHARP CORPORATION

OSAKA, JAPAN

Scanned by Dale