

**SHARP**  
PC-1500A POCKET COMPUTER  
INSTRUCTION MANUAL

WWW.  
PC-1500  
INFO

Developed By American Micro Products, Inc.



**MODEL PC-1500A OPTIONAL BOARD AND PERIPHERALS  
LIMITED WARRANTY**

Sharp Electronics Corporation warrants each of these products to the original purchaser to be free from defective materials and workmanship. Under this warranty the product will be repaired or replaced, at our option, without charge for parts or labor, with the exception of supplies, such as batteries, ribbons, inked rollers, etc., when returned to a SHARP FACTORY SERVICE CENTER listed in the instruction booklet supplied with your product.

This warranty does not apply to cassette tapes, software programs or appearance items nor to any product whose exterior has been damaged or defaced, nor to any product subjected to misuse, abnormal service or handling, nor to any product altered or repaired by other than a SHARP FACTORY SERVICE CENTER. This warranty does not apply to any product purchased outside the United States, its territories or possessions.

The period of the warranty shall be ninety (90) days on parts and labor from the date of the original purchase.

This warranty entitles the original purchaser to have the warranted parts and labor rendered at no cost for the period of the warranty described above when the unit is carried or shipped prepaid to a SHARP FACTORY SERVICE CENTER together with proof of purchase.

THIS SHALL BE THE EXCLUSIVE WRITTEN WARRANTY OF THE ORIGINAL PURCHASER AND NEITHER THIS WARRANTY NOR ANY OTHER WARRANTY, EXPRESSED OR IMPLIED SHALL EXTEND BEYOND THE PERIOD OF TIME LISTED ABOVE. IN NO EVENT SHALL SHARP BE LIABLE FOR CONSEQUENTIAL ECONOMIC DAMAGE OR CONSEQUENTIAL DAMAGE TO PROPERTY. SOME STATES DO NOT ALLOW A LIMITATION ON HOW LONG AN IMPLIED WARRANTY LASTS OR AN EXCLUSION OF CONSEQUENTIAL DAMAGE, SO THE ABOVE LIMITATION AND EXCLUSION MAY NOT APPLY TO YOU. IN ADDITION, THIS WARRANTY GIVES SPECIFIC LEGAL RIGHTS AND YOU MAY HAVE OTHER RIGHTS WHICH VARY FROM STATE TO STATE.

## **SHARP PC-1500A INSTRUCTION MANUAL**

Prepared by American Micro Products, Inc.

### **DISCLAIMER**

American Micro Products, Inc. and Sharp Electronics Corporation make no warranty either expressed or implied, including but not limited to any implied warranties of merchantability and fitness of purpose, regarding these instructional materials or any programs derived therefrom, and make such materials available on an "as is" basis.

American Micro Products, Inc. and Sharp Electronics Corporation assume no liability or responsibility of any kind arising from the use of these instructional materials or any part thereof.

**SHARP PC-1500A INSTRUCTION MANUAL****TABLE OF CONTENTS**

<b>INTRODUCTION</b> .....	1
Operational Notes .....	2
Troubleshooting .....	3
PC-1500A Specifications .....	4
Use of this Manual .....	7
Overview of Sharp PC-1500A .....	8
Sharp PC-1500A Keys .....	9
 <b>PART I: USING THE SHARP PC-1500A</b> .....	13
 <b>SECTION 1: GETTING STARTED</b> .....	13
Manual Problem Solving .....	13
Simple Calculations .....	14
Display Editing .....	16
Clearing the Display .....	16
Variables .....	17
Writing Your Own Program .....	21



<b>SECTION 2: KEYBOARD AND DISPLAY CONTROL</b> .....	26
The Keyboard .....	26
The Display .....	38
Resetting the Computer .....	42
 <b>SECTION 3: EXPRESSION AND KEYBOARD OPERATIONS</b> .....	43
Keyboard Arithmetic .....	43
Standard Number Format .....	47
Variables .....	50
TIME Function .....	54
 <b>SECTION 4: MATH FUNCTIONS AND STATEMENTS</b> .....	56
Number Alteration .....	57
General Math Functions .....	59
Logarithmic Functions .....	68
Trigonometric Functions .....	72
Inverse Trigonometric Functions .....	76
Rectangular and Polar Coordinate Conversions .....	77
Total Math and Logic Hierarchy .....	81

<b>PART II: BASIC PROGRAMMING WITH THE SHARP PC-1500A</b> .....	83
<b>SECTION 5. SIMPLE PROGRAMMING</b> .....	84
What is a BASIC Program? .....	84
Writing a Program .....	87
Entering a Program .....	97
Order of Program Execution .....	98
Multistatement Lines .....	113
Abbreviations .....	114
<b>SECTION 6. PROGRAM EDITING</b> .....	120
Editing Program Statements .....	120
Adding Statements .....	122
Deleting Statements .....	123
LIST (Listing a Program) .....	123
Interrupting Program Execution .....	123
Error Messages .....	124
<b>SECTION 7: BRANCHES AND LOOPS</b> .....	126
IF... THEN .....	126
GOTO .....	128
GOSUB .....	132
FOR ... NEXT Loops .....	135



<b>SECTION 8: MORE BRANCHING</b> .....	138
The Computed GOSUB Statement .....	138
Branching Using INKEY\$ .....	143
Branching Using Timers .....	144
 <b>SECTION 9: USING VARIABLES: STRINGS AND ARRAYS</b> .....	147
Array Concepts .....	147
DIM (Dimensioning Variables) .....	149
String Expressions .....	150
String Functions .....	154
String Variables .....	157
DATA .....	158
READ .....	159
RESTORE .....	160
INKEY\$ .....	161
DEF Key .....	162
Preassigned Keyword Keys .....	162
The AREAD Statement .....	163
Automatic Program Initiation .....	164
 <b>SECTION 10: DISPLAY PROGRAMMING</b> .....	165
BEEP (Programming Tones) .....	165
Display Functions .....	167

<b>SECTION 11: DEBUGGING</b> .....	184
STOP, CONT. ....	188
LOCK, UNLOCK .....	189
 <b>PART III: EXPANDING THE PC-1500A WITH THE PRINTER/CASSETTE INTERFACE CE-150</b> .....	190
 <b>SECTION 12: INTRODUCING THE CE-150</b> .....	190
Connecting the Computer to the Interface .....	190
Power .....	194
Connecting a Tape Recorder to the Interface .....	195
Loading the Paper .....	198
Replacing the Pens .....	202
 <b>SECTION 13: USING A CASSETTE RECORDER</b> .....	206
Tape Recorder Operation .....	206
Recording and Retrieving Programs .....	207
Using Data Files .....	212
Using Two Tape Recorders .....	213



<b>SECTION 14: USING THE PRINTER</b> .....	218
<b>CE-150 Printer Specifications</b> .....	218
<b>TEST Command</b> .....	219
<b>Printing Calculations</b> .....	219
<b>TEXT and GRAPH (Printer Modes)</b> .....	221
<b>Plotting Operations</b> .....	234
 <b>APPENDIX A: ASSEMBLY LANGUAGE PROGRAMMING</b> .....	244
 <b>APPENDIX B: I/O PORTS</b> .....	244
 <b>APPENDIX C: ABBREVIATIONS</b> .....	247
 <b>APPENDIX D: BATTERY REPLACEMENT</b> .....	255
 <b>APPENDIX E: KEY CODE TABLE</b> .....	259
 <b>APPENDIX F: ERROR CODE LIST</b> .....	261
 <b>APPENDIX G: FURTHER READING</b> .....	270
 <b>APPENDIX H: ORDER OF EXPRESSION EVALUATION</b> .....	274

<b>APPENDIX I:</b>	<b>COMMAND REFERENCE TABLE .....</b>	<b>279</b>
<b>APPENDIX J:</b>	<b>HEXADECIMAL NUMBER AND BOOLEAN FUNCTIONS .....</b>	<b>298</b>
<b>APPENDIX K:</b>	<b>DIFFERENCES BETWEEN PC-1500A AND PC-1500 .....</b>	<b>301</b>
<b>INDEX .....</b>		<b>303</b>



## SHARP PC-1500A INSTRUCTION MANUAL

### INTRODUCTION

We at Sharp Electronics Corporation (hereafter referred to as Sharp) are very pleased you have chosen the PC-1500A because we created it with you in mind. We know you wanted a very powerful state-of-the-art computer for simple and advanced programming, fast and easy editing, and creating outstanding graphics, yet one that is very affordable and economical.

We also thought you wanted a truly portable and personal computer, one you could have on hand twenty-four hours a day.

Finally, we know you wanted an instruction manual that is clearly and logically presented which can serve as both a beginning manual and as a reference for advanced applications. Please let us know if we have met your expectations.

## Operational Notes

Since the liquid crystal display of the PC-1500A is made of glass, reasonable care must be taken in handling the computer. To ensure trouble free operation of the Sharp PC-1500A, we recommend that:

1. You keep the computer in an area free from extreme temperature changes, moisture, and dust. During warm weather, vehicles left in direct sunlight are subject to high temperatures; prolonged exposure to such high temperatures can damage the computer.
2. Use only a soft, dry cloth to clean the computer. Never use a cloth wet with solvents or water.
3. To avoid battery leakage, remove the batteries when the computer is not going to be in use for an extended period of time.
4. If service is required, send only the computer to an authorized Sharp service center.
5. Keep this manual as a reference.

**SPECIAL NOTE:** The Sharp PC-1500A cannot be harmed in any way by pressing the computer keys. The worst that could possibly happen is that you would lose the contents of the memory. Experiment with the computer and enjoy it.

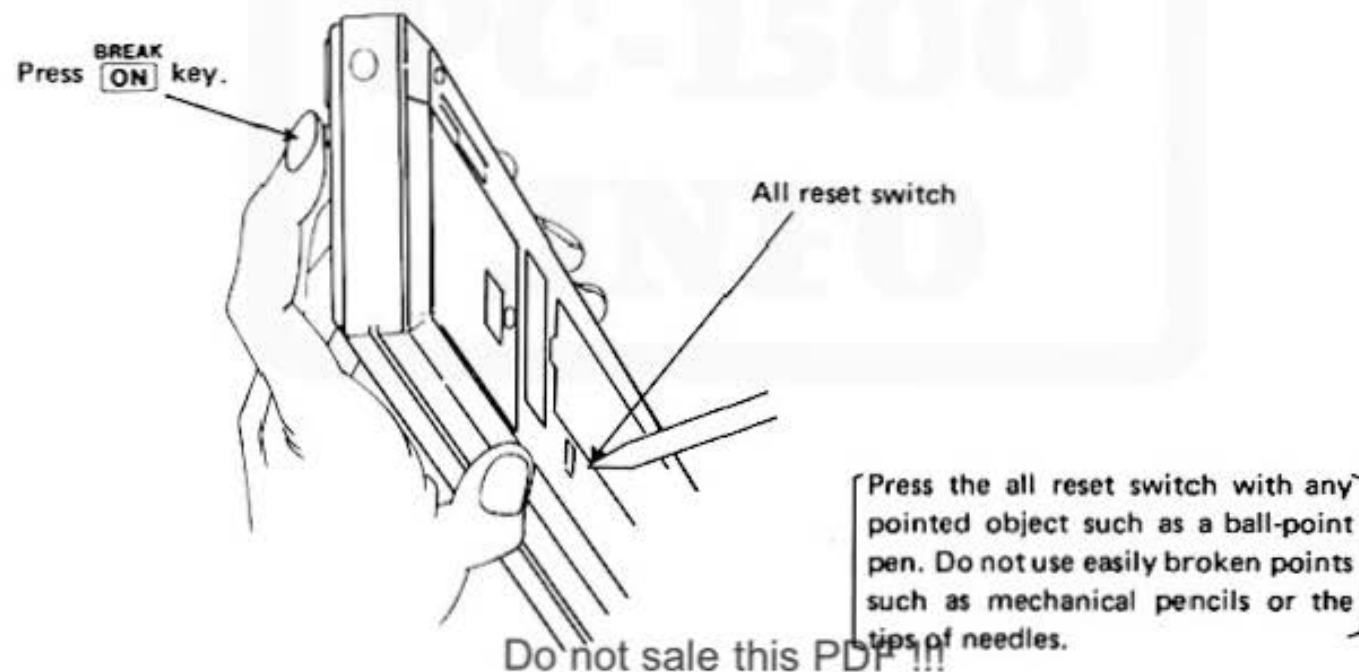


## Troubleshooting

NOTE: READ THIS SECTION TO TURN ON YOUR COMPUTER.

The PC-1500A, if subjected to strong external noise or impact during operation, may be rendered inoperative; this affects all the keys including the **ON** key. This is especially prone to happen during shipping. It is not serious and can easily be corrected using the following procedures.

Should the computer lock up, press the ALL RESET button on the back of the unit for approximately fifteen seconds while holding down the **ON** key.



Next, check the display to ensure that the statement

**NEWØ? : CHECK**







is showing. Then press the **CL** key and enter the statement

**NEWØ** **ENTER**

If the display does not read **NEWØ? : CHECK**, perform the reset operation over again. This operation will clear the program, the data, and the contents of the Reserve Keys, so only use the ALL RESET switch when it is absolutely necessary.

## PC-1500A Specifications

Model:	PC-1500A Pocket Computer
Numeric precision:	10 digits (mantissa) + 2 digits (exponent)
Calculation system:	According to mathematical formula (with priority judging function)
Program language:	BASIC

Capacity:	CPU:	CMOS 8 bit
	System ROM:	16 K Bytes
	Memory Capacity RAM:	8.5 K Bytes
	System area	1.9 K Bytes
	Input buffer:	80 Bytes
	Stack:	196 Bytes
	Machine language free area:	1023 Bytes (7C01H ~ 7FFFH)
	Others:	
	User area	6.6 K Bytes
	Fixed memory area	
	(A ~ Z, A\$ ~ Z\$):	624 Bytes
	Basic program data area:	5946 Bytes
	Reserve area:	188 Bytes
Calculations:	Four arithmetic calculations, power calculation, trigonometric and inverse trigonometric functions, logarithmic and exponential functions, angular conversion, extraction of square root, sign function, absolutes, integers, and logical calculations	
Editing function:	Cursor shifting  	
	Insertion	
	Deletion	
	Line up and down	 
5 Memory protection:	CMOS battery back-up Do not sale this PDF !!!	



6	Display:	Liquid Crystal Display (LCD) with 26 character width and $7 \times 156$ dot graphics.
	Keys	65 Keys including alphabetic, numeric, user-definable function, and pre-programmed keys)
	Power supply:	6.0 V, DC: 4 dry batteries (Type UM-3, AA, or R6)
	Power consumption:	6.0 V, DC: 0.13 w
	Operating time	Approximately 50 hours on dry batteries (Type UM-3, AA, or R6)
	Operating temperature:	0°C ~40°C (32°F~104°F)
	Dimensions:	195 mm (width) $\times$ 86 mm (depth) $\times$ 25.5 mm (height) 7 $\frac{1}{16}$ in (width) $\times$ 3 $\frac{3}{8}$ in (depth) $\times$ 1 in (height)
	Weight:	Approximately 375 gm (0.83 lbs) with batteries
	Accessories:	Soft case, four dry batteries, two keyboard templates, name label, and instruction manual
	Options:	Printer/Cassette Interface (CE-150) Expansion Memory Module (Plug-in type 4 K Byte RAM CE-151, 8 K Byte RAM CE-155/CE-159, 16 K Byte RAM CE-161) Cassette Tape Recorder (CE-152) RS-232C and Parallel Interface (CE-158)

## Use of this Manual

The Sharp PC-1500A Instruction Manual is divided into three basic sections which, when read in order, will give the first time computer user all the information necessary for programming the PC-1500A. These sections include:

1. Introduction to the Sharp PC-1500A
2. Programming in BASIC
3. Programming Graphics

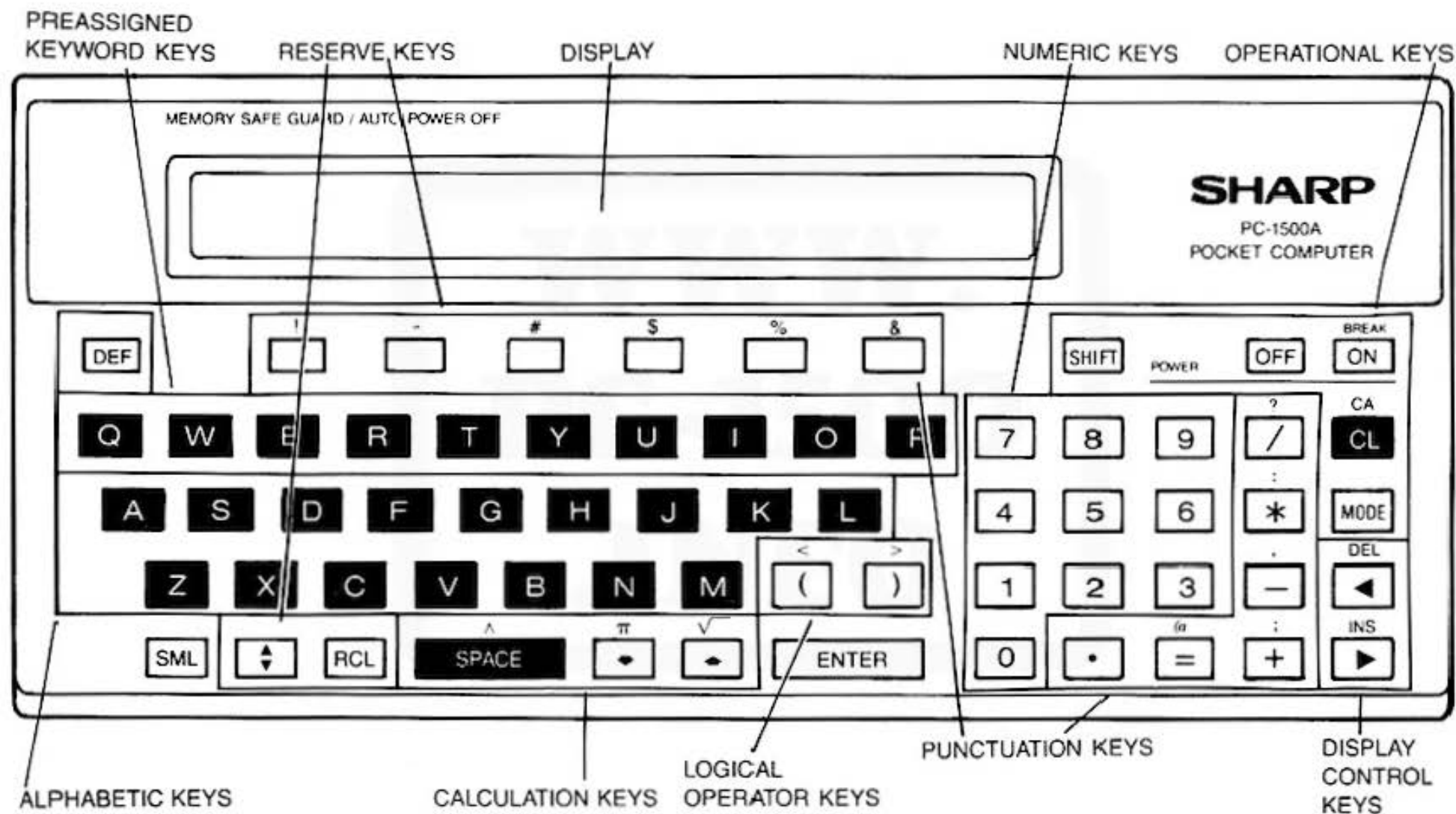
The experienced programmer should review the manual and then use it as a reference book.

The appendices in the manual are:

- A. Assembly Language Programming
- B. I/O Ports
- C. Abbreviations
- D. Battery Replacement
- E. Key Code Table
- F. Error Code List
- G. Further Reading
- H. Order of Expression Evaluation
- I. Command Reference Table
- J. Hexadecimal Numbers and Boolean Functions

Do not sale this PDF !!!

## 8 Overview of Sharp PC-1500A



## Sharp PC-1500A Keys

This is a brief overview of the Sharp PC-1500A keyboard. A page reference is given after each key where more information may be found in this manual.

### Operational Keys

<b>ON</b>	Turns on the power to the PC-1500A, and acts as a program interrupt. Page 27.
<b>OFF</b>	Turns power off. Page 27.
<b>SHIFT</b>	Used to access the secondary functions of the keys. Page 28.
<b>ENTER</b>	Used to pass information to the computer. Page 28.
<b>CL</b>	Clears the display. The secondary function, CA, clears the display and the memory stack. Page 28.
<b>MODE</b>	Used to place computer in RUN, PROGRAM, and RESERVE modes. Page 29.



## Alphabetic Keys

**A-Z**

The letters of the alphabet are arranged in the same order as typewriter keys. Page 29.

**SML**

The Small key allows the user to enter lower case letters. Page 30.

**SPACE**

A typewriter style spacebar. Page 30.

## Numeric Keys

**0-9**






The PC-1500A has a standard 10-key pad for numbers. Page 30.




## Punctuation Keys





All Punctuation Keys listed here (except for the period) are secondary functions (found printed above the other keys) and must have the **SHIFT** key pressed before they can be accessed. Page 31.

## Calculation Keys



     These five Calculation Keys listed are for normal arithmetic operations. Page 32.

   The Exponentiation, PI, and Square Root keys are all available as secondary function keys. Page 32.

## Logical Operator Keys

  Less than and greater than signs are available for logical comparisons. Page 33.

## Display Control Keys

  The Left and Right Arrow keys move the cursor to the left or right in the display. Page 33.

The Delete and Insert functions are used to insert or remove characters from the display. Page 34.

  The Up and Down Arrow keys are used for program editing. Page 35.

## Reserve Keys

The Reserve Keys are the six blank keys that may be defined for up to eighteen functions by the user. Page 36.



The Reserve Select key is used to select from the three assignments for each reserve key. Page 36.



The Recall key is used to recall the function menus for the Reserve Keys. Page 36.

## Preassigned Keyword Keys

The Preassigned Keyword Keys are those which Sharp has designated to hold frequently used BASIC statements. These keys may be more easily used after you place the included template over the keyboard. Page 37.

## **PART I: USING THE SHARP PC-1500A**

### **SECTION 1: GETTING STARTED**

This section is to introduce you to the Sharp PC-1500A computer. All the topics discussed in this section will be covered in more depth in later sections of the manual; this section will provide you with an overview of the machine and some of its capabilities. Topics covered by this section include:

- Simple Calculations
- Display Editing
- Variables
- Creating Your Own Programs
- Entering a Program

#### **Manual Problem Solving**

The PC-1500A, aside from being a handheld computer, is also a calculator. All calculations that can be performed by the PC-1500A as a computer can also be performed by the PC-1500A as a calculator.



## Simple Calculations

For simple calculations, such as the addition of two numbers, the first number would be entered

**103**

then the arithmetic operator

**103 +**

followed by the other number

**103 + 57**

The **ENTER** key is then pressed and the result is displayed on the computer, as follows.

**160**

Note that the equal sign was not used in the calculation; it is only used when defining a variable or as a logical operator. For the remainder of the manual, examples will be demonstrated using the following notation.

**103 + 57**

**ENTER**

**160**

Examples of different types of calculations are shown below.

ADDITION: find the sum of 12 plus 7 plus 5 plus 10.

**12 + 7 + 5 + 10      ENTER      34**

SUBTRACTION: find the value of 155 minus 23.

**155 - 23      ENTER      132**

MULTIPLICATION: find the value of 11 multiplied by 34.

**11 \* 34      ENTER      374**

DIVISION: find the value of 125 divided by 5.

**125 / 5      ENTER      25**

EXPONENTS: find the value of 11 squared.

**11 ^ 2      ENTER      121**

LOGARITHMS: find the common logarithm of 1000.

**LOG 1000      ENTER      3**

16 TRIGONOMETRIC FUNCTIONS: find the sine of 30.

**SIN 30**

**ENTER**

0.5

## Display Editing

At some time you may want to change what is in the display. This can be accomplished quickly and easily using the Display Control Keys. The four operations possible are: (1) forward cursor movement, (2) backward cursor movement, (3) character insert, and (4) character delete.

Cursor movement is performed using the Left and Right Arrow keys, **◀** and **▶**. The **▶** key is used to move the cursor from left to right across the display. The **◀** key is used to move the cursor backward across the display from right to left. A character can only be added to the display in the space occupied by the cursor. By typing the statement "COMPUTERS ARE FUN" you can see the cursor move across the screen in front of the letters. When the cursor is over a previously used space, it will appear as a flashing box.

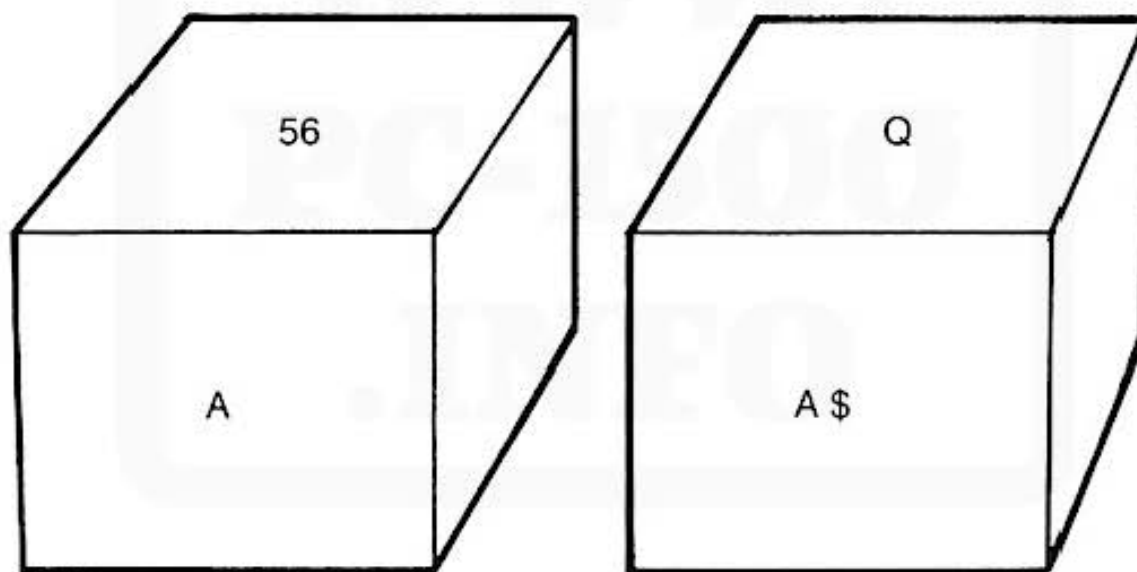
To insert or delete a character, place the cursor at the appropriate location and press the **SHIFT** key and then the **◀** or the **▶** key. When deleting a character, the character will disappear and the display will close up the deleted space. When inserting a character, the display will expand, thereby creating a space at the cursor location. A new character can be added at this time.

## Clearing the Display

The display may be cleared by pressing the **CL** key.

## Variables

A variable is a letter or combination of letters which can be assigned either a numeric value or a character string. A classic way to visualize variables is to picture them as boxes, as in the illustration below.





The PC-1500A can use variables when it is employed as a calculator as well as when it is used as a computer. By keying in the following expression, the variable A is set to a value of 56.

**A = 56**    **ENTER**

At this point the display will appear as:

56

Another value may be placed in the variable B in a like manner.

**B = 10**    **ENTER**

The value of a variable may be recalled by entering the variable into the PC-1500A, as in the example below.

**A**

**ENTER**

56

The variables may be used in any of the calculations available on the PC-1500A. Examples include:

<b>A*B</b>	<b>ENTER</b>	560
<b>A/B</b>	<b>ENTER</b>	5.6
<b>LOG B</b>	<b>ENTER</b>	1

Returning to the illustration with the boxes, it is easy to see the similarity between a box and a variable. As with a variable, each box contains whatever was last placed in it. Since the contents may be unknown, they are, therefore, not useful and must be discarded. In the case of a numeric variable, it is set equal to zero as in the statement below.

**A = 0**

In the case of a string variable, it is set equal to the null string.

**A\$ = ""**

20

By clearing the variable (emptying the box) you will always know the value of the variable as long as you know what you put into it. In the program below, variables are used for both the input and as a counter in the program.

```
10: A = 0
20: B = 0
30: I = 0
40: INPUT "NUMBER OF COUNTS = ",A
50: FOR I = 1 TO A
60: B = B + 1
70: PAUSE B
80: NEXT I
90: GOTO 10
```

The next program listing demonstrates the use of string variables.

```
10: A$ = ""
20: B$ = ""
30: C$ = ""
40: A$ = "GOOD"
50: B$ = " MORNING "
60: INPUT "ENTER YOUR NAME ",C$
70: PRINT A$;B$;C$
80: END
```

## Writing Your Own Programs

### Creating a Program

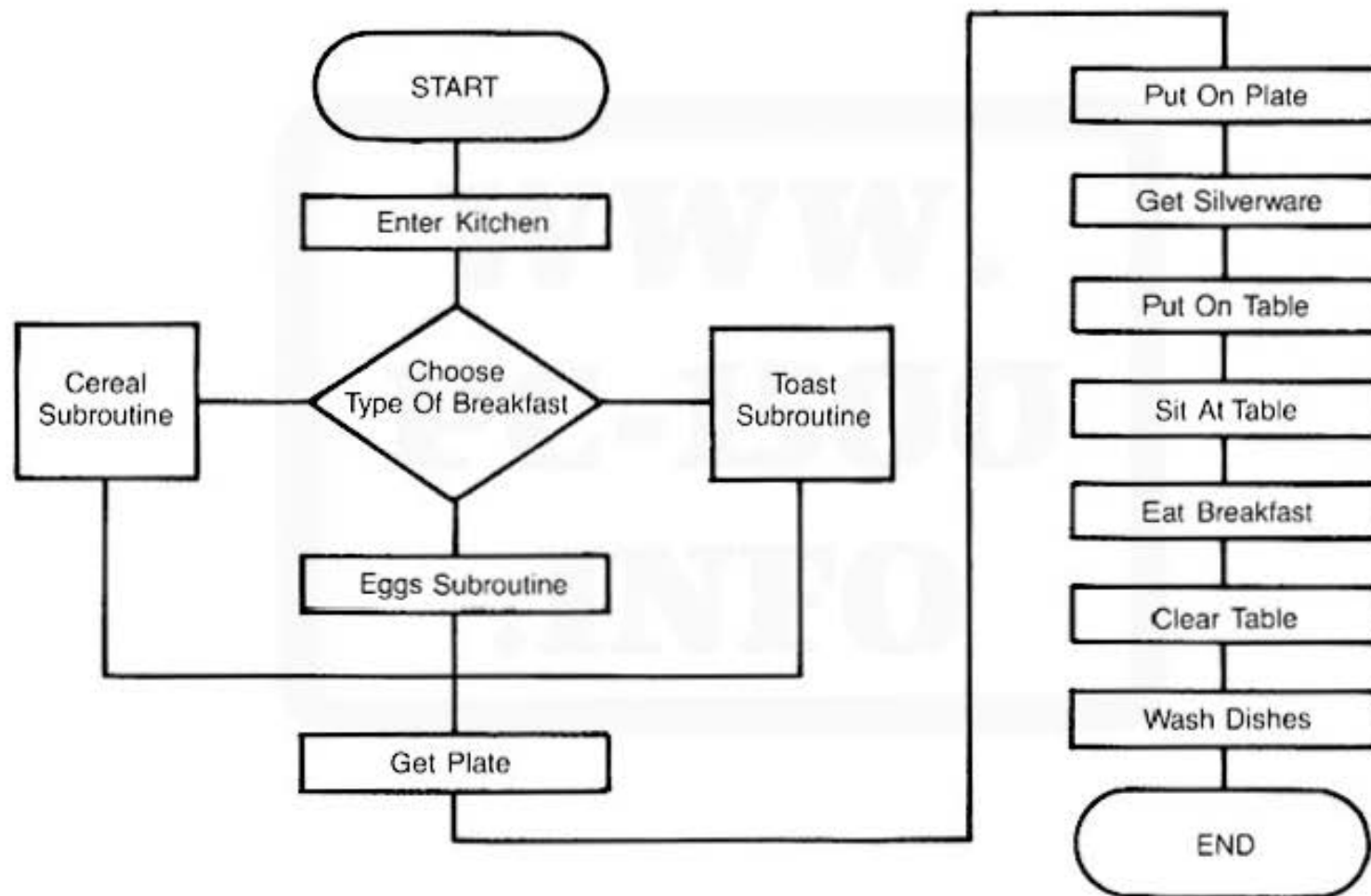
As a beginning programmer, there are several fundamental steps that you (as well as every other programmer) must follow. These steps include:

1. Problem definition
2. Approach definition
3. Logic and program design
4. Program coding
5. Program testing

The *problem definition* is a complete description of what you wish to do on the computer. As part of the problem definition, it is important to know: (1) what information is available for you to give to the computer as input (make sure that the input information is relevant to the problem); (2) what intermediate information is going to be needed to find the final solution; and (3) what kind of information you want the computer to give you as output.

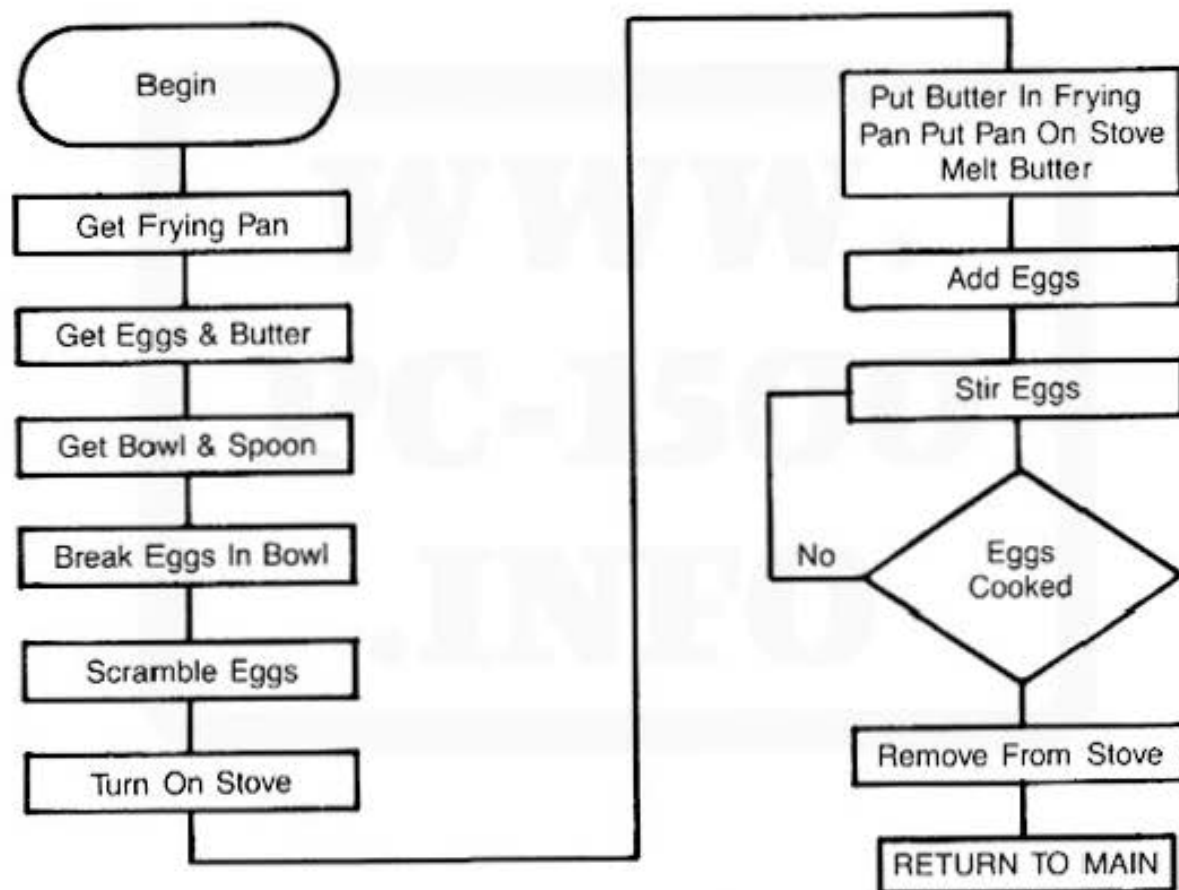
The *approach definition* is a full description of the approach you will take to solve the problem. This will include the mathematical formulas needed for the calculations and a description of the decision that the program or operator will have to make during the running of the program. Up to this point, the program has been defined in everyday language; however, when the logic of the program (an accurate description of how the program is to be

run) is defined, it is generally much simpler to draw a picture of the program logic. This picture is commonly called a *flowchart* because it represents the flow of the program. Below is a flowchart for preparing breakfast.





### SCRAMBLED EGGS SUBROUTINE



## Entering a Program

A program may be entered into the PC-1500A by first placing the computer into the PROGRAM mode using the **MODE** key. Each statement is then keyed into the display as in the example below.

**10PRINT** **SHIFT** **"HELLO** **SHIFT"**

At this point the display prompt will appear as below.

**10PRINT"HELLO"**

The next step is to press the **ENTER** key to enter the statement into the computer. When the **ENTER** key has been pressed, the display prompt will look like this.

**10: PRINT "HELLO"**

Notice that a colon has been added to the statement immediately after the statement number; also, spaces have been added to make the line easier to read. Remember that the computer will add the colon after the statement number; you do not have to add it.

## Running a Program

After entering the program in the PROGRAM mode, place the PC-1500A in the RUN mode by pressing the **MODE** key; then start the program by typing the command RUN and pressing the **ENTER** key.

## Sample Program

The program listed below is a simple guessing game. The object of the game is to guess a number between 0 and 1,000. The PC-1500A will pick a random number in line 30; then, you try to guess that number in line 40.

```
10: PAUSE "GUESSING GAME"  
20: RANDOM  
30: N = INT(RND(1000)):X = 0  
40: "G":INPUT "GUESS A NUMBER ":G:X = X + 1  
50: IF G = N THEN PAUSE "YOU WIN IN ":X:" TRIES":BEEP 10  
60: IF G > N THEN PAUSE "TOO HIGH" : GOTO "G"  
70: IF G < N THEN PAUSE "TOO LOW" : GOTO "G"  
80: INPUT "PLAY AGAIN? (Y/N) ":Y$  
90: IF Y$ = "Y" THEN GOTO 20  
100: PAUSE "HAVE A NICE DAY" : END
```

## SECTION 2: KEYBOARD AND DISPLAY CONTROL

### The Keyboard

The 65 keys on the Sharp PC-1500A can be grouped into the following nine classes:

- Operational Keys
- Alphabetic Keys
- Numeric Keys
- Punctuation Keys
- Calculation Keys
- Logical Operator Keys
- Display Control Keys
- Reserve Keys
- Preassigned Keyword Keys

These nine classes of keys are depicted on the keyboard illustration found in the introduction on page 8.

## Operational Keys

The Operational Keys are the basic keys to get you started and keep you moving smoothly along.

### BREAK

**ON**

The **ON** key applies power to your Sharp PC-1500A pocket computer. To conserve power, the PC-1500A will automatically shut off if nothing is entered within a seven minute period (unless a program is running). If you wish to interrupt, or break into, a program that is running, merely press **ON** and the program will stop and display

BREAK AT LINE ###

The three space symbols, ###, represent the line number that was being executed. To start running the program again from the point of interruption, simply type in **CONT** and press the **ENTER** key. NOTE: There is no need to press the **SHIFT** key to invoke the break function.

**OFF**

The **OFF** key turns the PC-1500A off. Whenever a program is running, the busy indicator will be displayed. To turn the PC-1500A off when this indicator is on, you must first interrupt the program by pressing the **ON** key, and then pressing the **OFF** key.

**SHIFT**

The **SHIFT** key has several important uses. First, if the **SHIFT** key is pressed before a key with a secondary function printed above it, the secondary function will be called in (e.g., pressing the **SHIFT** key before pressing the **CL** key will call in the Clear All (CA) function). Second, if the **SHIFT** key is pressed before pressing the **MODE** key, the RESERVE mode will be selected. Ordinarily, pressing the **MODE** key will alternately select the RUN mode or the PROGRAM mode. Using the **SHIFT** key makes a third mode possible. (See Mode key explanation in this section for more detail.) Third, if the **SHIFT** key is pressed before an Alphabetic Key (all Alphabetic Keys are normally upper case letters) a lower case letter will appear. Fourth, if the **SHIFT** key is pressed before an Alphabetic Key when the Small **SML** key has invoked the lower case mode, an upper case letter will appear. (See Small key explanation under Alphabetic Keys in this section.)

**ENTER**

The **ENTER** key must be pressed to pass information into the computer. Always press it after giving an instruction to the PC-1500A. Also be sure to press it after editing a statement or your changes will not be recognized.

## CA

**CL**

The red Clear **CL** key allows you to clear the display screen. Pressing **SHIFT** before **CL** will activate the Clear All secondary function. This will reset the computer, clearing the display and the memory stack. NOTE: The Clear All function will not erase all programs, reserve memories, and all variables. To erase all programs, type in **NEW** and press **ENTER** while in the PROGRAM mode. To



erase all reserve memories, type in **NEW** and press **ENTER** while in the RESERVE mode. To clear all variables by setting them to zero (or null) type in **CLEAR** and press **ENTER**.

## **MODE**

When you turn the PC-1500A on, check the display for the operational mode it is in (RUN, PROGRAM, or RESERVE). Press **MODE** to change the selection from RUN to PROGRAM or from PROGRAM to RUN. Press **SHIFT** and then press **MODE** to select the RESERVE mode. To leave the RESERVE mode, press **MODE**. All manual calculations (using the computer as a calculator) are done in the RUN mode, and all programs are run in the RUN mode. All programs are written and edited in the PROGRAM mode. The RESERVE mode is used for storing frequently used functions on single keys and for storing menus to help identify these functions quickly. (See Reserve Keys in this section and also Section 9 for detailed information on using the RESERVE mode.)

## Alphabetic Keys

### **A-Z**

The Sharp PC-1500A has the same 26-letter arrangement found on most typewriters. The letters are normally upper case (the opposite of most typewriters) but this is convenient because the PC-1500A only recognizes statements and commands in upper case letters. The letters may be forced one at a time to lower case by pressing **SHIFT** before pressing the letter desired.

**SML**

The Small **SML** key is used when all lower case letters are desired. Pressing **SML** again will return to the UPPER CASE mode. Note: While in the SMALL KEY mode, pressing the **SHIFT** key before pressing a letter will produce a single upper case letter.

**SPACE**<  
(>  
)

The Spacebar and Left and Right Parenthesis keys are included with the Alphabetic Keys because of their location and frequent usage with letters.

### Numeric Keys

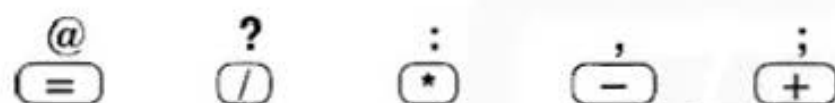
**0-9**

.

The Sharp PC-1500A has a typical 10-key numeric arrangement for your convenience located just to the right of the Alphabetic Keys.

## Punctuation Keys

The Sharp PC-1500A Punctuation Keys are all secondary operations, found typed above other keys. Press **SHIFT** before pressing any of these keys. Five are found immediately to the right of the Numeric Keys and six are found below the display and above the Reserve Keys.



Pressing **SHIFT** before pressing the Equal key, Division key, Multiplication key, Subtraction key, or Addition key will produce the at (@) sign, question mark, colon, comma, and semicolon, respectively.



Pressing **SHIFT** followed by the corresponding blank Reserve Key will produce the exclamation point, double quotation mark, pound sign, dollar sign, percent sign, and ampersand, respectively.



NOTE: The decimal point in the Numeric Keys doubles as the period. The left and right parentheses are grouped with the Alphabetic Keys.

## Calculation Keys



The Equal, Division, Multiplication, Subtraction, and Addition keys are located immediately to the right of the Numeric Keys for ease in calculations. These keys have secondary punctuation functions printed above them.

If you need more advanced calculations than the basic five, the Exponentiation

key (key with  $\wedge$  symbol above and **SPACE** below), Pi key (key with  $\pi$  symbol above and  below), and Square Root key (key with  $\sqrt{\phantom{x}}$  symbol above and  below) keys are available as secondary functions on the bottom row of the keyboard. Press **SHIFT** and then the appropriate key. If you prefer, instead of using the Pi and Square Root keys, you can type in PI and SQR and achieve the same results.



When doing calculations, two very helpful keys are the Left and Right Arrow keys. Although these keys have editing uses in the PROGRAM mode (discussed later in this section under Display Control Keys) for calculations in RUN mode they allow you to recall the calculation after you have pressed **ENTER** and only the result will remain on the screen. Pressing either the Left or Right Arrow key will redisplay the calculation. You must press **ENTER**, however, to get the result again.

## Logical Operator Keys



The Less Than key and the Greater Than key are secondary functions on the Left and Right Parenthesis keys, respectively. They are located on the keyboard just above the **ENTER** key. To use them, you must press **SHIFT** first.

## Display Control Keys


The Display Control Keys are for editing purposes and consist of the Left Arrow and Right Arrow keys with their secondary functions, Delete and Insert, respectively, and also the Down and Up Arrow keys, found just to the left of the **ENTER** key. The Clear key, **CL**, is mentioned here because it clears the display, but it is fully discussed under the Operational Keys elsewhere in this section.




The Left Arrow key is used when writing a program to move the cursor to the left in a statement until the cursor is over the position desired. If you need to move left more than one position, simply hold down the Left Arrow key and it will continue moving to the left until you release the key. As the cursor backs up, it becomes a blinking grid so that you can easily tell which position it is in.

The Left Arrow key, when pressed while reentering a program for editing, will move the cursor immediately to the first character to the right of the statement number. To edit the statement number itself, press the Left Arrow key once more.

DEL

The Left Arrow key has a powerful secondary function, the Delete function .

DEL

To delete a character, press **SHIFT** and then .

INS




The Right Arrow key is used when writing a program to move the cursor to the right in a statement until the cursor is over the position desired. When you first write a statement the cursor will not move to the right of what you have already keyed in; but, after finishing the line you can go to the left using the Left Arrow key and then back to the right, if desired. If you need to move right more than one position, simply hold down the Right Arrow key and it will continue moving to the right until you release the key. As the cursor moves forward, it becomes a blinking grid so you can easily tell which position it is in.

The Right Arrow key, when pressed while reentering a program for editing, will move the cursor immediately to the first digit in the statement number.

INS



The Right Arrow key has a powerful secondary function, the Insert function .


INS



To insert a character, press **SHIFT** and .







The Down and Up Arrow keys are also used for editing. The Down Arrow key  will display the first line of a program in memory if you are in PROGRAM mode, but not writing or editing a program. If you are writing or editing a program, pressing  will display the next program line. Holding the key will continue scrolling down program lines until you release the key or the last program line is reached.

The Down Arrow  key is very useful in debugging techniques as described fully in Section 11.

The Up Arrow  key will display the last line of a program in memory if you are in PROGRAM mode, but not while you are writing or editing a program. If you are writing or editing a program, pressing  will display the previous program line. Holding the key down will continue scrolling up program lines until you release the key or the first program line is reached.

The Up Arrow  key is also very useful in debugging techniques as described fully in Section 11.

## Reserve Keys

Reserve Keys are those six blank keys found immediately under the display with !, ", #, \$, %, and & printed above them. Using the Reserve Mode key (see Mode key discussion under Operational Keys in this section) the Reserve Select key, , and the Recall key, **RCL**, up to 18 functions can be assigned to these six keys. (See Section 9 for detailed instructions on using the RESERVE mode.)



The Reserve Select key controls three sets of six keys available for assignment in the RESERVE mode denoted by I/II/III on the display. Pressing the Reserve Select key repeatedly will select the set of six keys you desire. You can enter a function on each of the six keys. (See Mode key under Operational Keys in this section and Section 9 for detailed information on using the RESERVE mode.)



The Recall key calls up the function menus for the current RESERVE mode (I/II/III) when in RUN mode. These menus let you know at a glance what type of information is on the six function keys for the RESERVE mode you selected. Pressing the key again will return to the original display. (See Mode key under Operational Keys and Section 9 for detailed information on using the RESERVE mode.)

## Preassigned Keyword Keys

Preassigned Keyword Keys are those keys the PC-1500A has assigned BASIC statement values to and those to which you can assign line number labels. By using the PC-1500A's preassigned BASIC statements, you can save valuable time by pressing one key instead of typing out the whole keyword each time you want to use it. By preassigning line number labels, you can begin program execution at up to 18 different line numbers; this is another time saver.

The ten keys already preassigned by the PC-1500A correspond to the top line of Alphabetic Keys and are labeled by placing one of the two preassigned keyword templates provided with the PC-1500A over the Alphabetic Keys.

INPUT	PRINT	USING	GOTO	RETURN	CSAVE	CLOAD	MERGE	LIST
<b>Q</b>	<b>W</b>	<b>E</b>	<b>R</b>	<b>T</b>	<b>Y</b>	<b>I</b>	<b>O</b>	<b>P</b>

NOTE: CSAVE, CLOAD, and MERGE can only be used with the optional extra printer/cassette interface.

The remaining Alphabetic (letter) Keys, Space Bar, and Equals keys have been left for you to preassign line number labels to. (For detailed instructions on using Preassigned Keyword Keys, see Section 9.)

**DEF**

Press the Define key, followed by the key that has been preassigned a BASIC statement by the PC-1500A or a statement line number by you. (See Section 9 for more details on using the Define key.)

## The Display

The Sharp PC-1500A display will do everything from displaying single characters you key in, to producing extensive graphic designs with sound effects!

### Display Indicators

After turning on the PC-1500A, look carefully at the Display Indicators. Some of the information they can tell you is illustrated below.

1	2	3	4	5	6	7	8
BUSY	SHIFT	SMALL	RAD	RUN	DEF	I	•
			or	or		or	
			DEG	PRO		II	
			or	or		or	
			GRAD	RESERVE		III	

#### 1 BUSY

#### Program Execution Indicator

If a program is running, **BUSY** will appear on the display. If no program is running, the **BUSY** space on the display will be blank. (See **OFF** under Operational Keys in this section for how to turn off the computer while program is running.)

2 SHIFT

Shift Key Indicator

This indicates the Shift key has been pressed. NOTE: If **SHIFT** had not been pressed, nothing would have been displayed in this location. (See **SHIFT** under Operational Keys elsewhere in this section for more information on the Shift key.)

3 SMALL

Small Key Indicator

This indicates the computer is in the LOWER CASE mode. NOTE: If the mode had been the regular UPPER CASE mode, nothing would have been displayed in this location. (See **SML** under Alphabetic Keys in this section for more information on the LOWER CASE mode.)

4 RAD  
DEG  
GRAD

Angular Measurement Indicator

This indicates the current unit of angle for the input of trigonometric functions: RAD for radians, DEG for degrees, and GRAD for gradients. One of these three units will always be displayed in this location. (See Section 4 for more details on trigonometric functions.)

5 RUN  
PROGRAM  
RESERVE

This indicates which operational mode the PC-1500A is in. One of these three modes will always be displayed in this location. (See Mode key under Operational Keys in this section for more details on mode.)


## 6 DEF

### Definable Mode Indicator

This indicates **DEF** has been pressed. If **DEF** had not been pressed, this location would be blank. (See the Define key under Preassigned Keyword Keys in this section and also see Chapter 9 for more details.)

## 7 I/II/III

### RESERVE Mode Indicator

This indicates which reserve mode the PC-1500A is in. (See the Reserve Select key  under Reserve Keys in this section and also Chapter 9 for more details on the RESERVE mode.)

## 8 •

### Battery Indicator

This indicates the battery has enough power to operate. If the indicator should go off, replace the batteries or connect the PC-1500A to an external power supply.

See how much you can learn from your PC-1500A display by just turning it on!

NOTE: The PC-1500A remembers the last display indicator settings it had when turned off, and when turned on again it should present you with these same indicator settings. However, if you should remove the batteries without first connecting the PC-1500A to an external power supply, it will develop amnesia, forgetting its display indicator settings, its variable values, and even its programs. So be careful when changing batteries!

The Sharp PC-1500A can also produce sound effects to accompany the display. (See Chapter 5 for a detailed explanation of this feature.)



## Prompt and Cursor

At the far left of the display, find the prompt symbol (>). This prompt indicates that the PC-1500A is asking, or prompting, you to tell it what to do next.

Type in any character. Notice the prompt disappears and a cursor (—) appears to the right of your letter. This cursor is a place marker that indicates where your next character will appear.

You can type in up to 80 characters in a line, but the PC-1500A can only show you 26 characters at a time. To see the others, use your Left and Right Arrow keys. (See Left and Right Arrow keys under Display Control Keys in this section.)

## Display Capabilities

Now look more closely at the display itself. It is a  $7 \times 156$ , programmable, dot-matrix Liquid Crystal Display. You can display up to 26 characters at a time. Each character occupies a  $5 \times 7$  dot matrix.

For graphic purposes, the entire display may be utilized as a  $7 \times 156$  dot matrix. Individual dots within any of 156 columns may be energized to create graphics, figures, or special symbols.

A speaker and tone generator allow the programmer to add the dimension of sound to the man-machine interaction. Tones may be created at any of the 256 frequencies (range 230 Hz to about 7 kHz). Automatic repetition of a tone and control of the duration of a tone are also possible under program control. If your interest has been piqued and you have to know more right now about programming the display, turn to Chapter 10 on Display Programming.

## Resetting the Computer

The PC-1500A, if subjected to strong external noise or impact during operation, may render all its keys, including the **(ON)** key, inoperative. Should this occur, press the ALL RESET switch on the back of the unit for approximately eight seconds with the **(ON)** key held down, as shown in the figure on page 3. Then, check the display; the prompt should read

NEWØ? : CHECK

Next, press the **(CL)** key, type **NEWØ**, and press the **(ENTER)** key. If the display prompt did not read

NEWØ? : CHECK

perform the above operation again. With this operation, the program, data, and all the reserved contents are cleared. Thus, do not attempt to press the ALL RESET switch except when the above trouble occurs.

## SECTION 3: EXPRESSIONS AND KEYBOARD OPERATIONS

This section of the manual will discuss simple keyboard arithmetic and the use of expressions with the Sharp PC-1500A. When used in the context of computing, an expression is any combination of numbers, variables, characters, operators, or functions. All these terms will be explained in detail in this and following sections within the manual. Before explaining the procedures for programming the PC-1500A in BASIC, all operations will be covered by demonstrating their use in a "calculator" format. The PC-1500A is capable of running BASIC programs and evaluating expressions entered into the display as though it were a calculator.

Discussed in this section are:

- Keyboard Arithmetic
- Parentheses
- Standard Number Format
- Variables
- Logical Evaluation
- TIME Function

### Keyboard Arithmetic

#### Arithmetic Operators

The arithmetic functions for the PC-1500A are the same as those found on the simplest of calculators: add, subtract, multiply, and divide.

## Addition

To add a series of numbers, type in the numbers separated by a plus sign, +, as demonstrated below and then press the **ENTER** key. The result will appear in the right hand portion of the display.

EXAMPLE: add the numbers 12, 15, 76, 42.

**12 + 15 + 76 + 42**

**ENTER**

**145**

## Subtraction

To subtract a series of numbers, the minus sign, −, is used.

EXAMPLE: subtract 23, 12, 65, and 26 from 1250.

**1250 − 23 − 12 − 65 − 26**

**ENTER**

**1124**

## Multiplication

The times sign in both BASIC and the Sharp PC-1500A is an asterisk, \*.

EXAMPLE: multiply 43 times 56.

**43\*56**

**ENTER**

**2408**

## Division

A slash mark, /, is used as a division sign in BASIC and on the PC-1500A.

EXAMPLE: divide 1230 by 6.

**1230/6**

**ENTER**

**205**

It is important to note that none of the arithmetic operations are followed by an equal sign. The use of the equal sign will be discussed further in this section. The higher mathematical functions will be further discussed in Section 4.

## Arithmetic Hierarchy

When an expression has more than one arithmetic operation being performed, the order in which the operations are performed takes place in the following manner.

1. Multiplication and division
2. Addition and subtraction

In the case where two or more operations of the same priority appear in the same expression, the operations will be performed from left to right.

EXAMPLE: subtract 3 divided by 12 from 6.

$$6 - 3/12$$

**ENTER**

5.75

### Parentheses

When evaluating compound calculations, it is often convenient to change the calculation hierarchy to ensure clarity and simplicity. The use of parentheses allows this to be done: all parts of the expression enclosed in parentheses will be evaluated or calculated first, having priority over any other calculation or operation.

EXAMPLE 1: subtract 3 divided by 12 from 6; then, divide the value of 6 minus 3 by 12.

$$6 - 3/12$$

**ENTER**

5.75

$$(6 - 3)/12$$

**ENTER**

0.25

EXAMPLE 2: find the value of 9 plus 7 divided by 6 minus the value of 12 multiplied by 5; then, find the value of 9 plus 7 divided by the value of 6 minus 12, multiplied by 5.

$$9 + 7/6 - 12 \cdot 5$$

**ENTER**

- 49.83333333

$$(9 + 7)/(6 - 12) \cdot 5$$

**ENTER**

- 13.33333333

## Standard Number Format

Although all numbers are calculated with ten digits of precision, numbers may be displayed with up to ten digits of precision.

EXAMPLE: find the value of 1 divided by 3.

**1/3**

**ENTER**

**3.33333333E - 01**

Extra zeros to the right of the decimal point will not be displayed.

EXAMPLE: enter the value 2.500000000.

**2.500000000**

**ENTER**

**2.5**

Leading zeros are not displayed.

EXAMPLE: enter the value 000023.

**000023**

**ENTER**

**23**



Numbers greater than  $10^{-10}$  and less than  $10^{10}$  will be displayed without an exponent.

EXAMPLE: enter the number 123400009.

**123400009**

**ENTER**

123400009

EXAMPLE: enter the number 0.000000009.

**0.000000009**

**ENTER**

0.000000009

Numbers less than  $10^{-10}$  and greater than  $10^{10}$  will be displayed in scientific notation.

### Scientific Notation

Scientific notation is a simple and convenient method for representing very small and very large numbers. In this method, each number consists of a mantissa and an exponent. The mantissa will always be greater than one but less than ten (i.e., one digit to the left of the decimal point followed by up to nine digits to the right of the decimal point). The exponent is designated by an "E" followed by an exponent of ten. In the PC-1500A, the exponent must be between +99 and -99; if a number outside this range is entered, then only the last two numbers entered will be used.

EXAMPLE 1: enter the number 1.25E7.

**1.25E7**

**ENTER**

12500000

EXAMPLE 2: enter the number 12380000000000.

**12380000000000**

**ENTER**

1.238E 13

EXAMPLE 3: enter the number 0.000000000000000125.

**0.000000000000000125**

**ENTER**

1.25E - 15

EXAMPLE 4: enter the number 4E - 6.

**4E - 6**

**ENTER**

0.000004

EXAMPLE 5: enter the number 1.347E 453.

**1.347E 453**

**ENTER**

1.347E 53

The final example illustrates how an error can result when a number outside the permitted range is entered.

### Range of Numbers

The range of numbers which may be used by the Sharp PC-1500A is from a maximum value of 9.999999999E 99 to a minimum of 9.999999999E - 99.

## Variables

To find the area of a room, the width must be multiplied by its length. If the width is designated by the letter W, the length by L, and the area by A, then the mathematical formula for finding the area of a room is  $A = W * L$ . The letters A, W, and L are variables. No matter how the width and length of the room varies, the area of a room will always be equal to its length times its width (i.e., the same formula can be used for a closet as well as a family room).

EXAMPLE: enter the values 15 and 20 for the variables W and L, respectively; then, multiply W times L.

<b>W = 15</b>	<b>ENTER</b>	15
<b>L = 20</b>	<b>ENTER</b>	20
<b>A = W * L</b>	<b>ENTER</b>	300

The value for a variable may be observed by entering the variable name into the PC-1500A.

EXAMPLE: enter the name for the variable W.

<b>W</b>	<b>ENTER</b>	15
----------	--------------	----

The value of a variable may be changed without affecting the value of any other variable.

EXAMPLE: enter the value 25 for the variable W; then multiply W times L.

<b>W = 25</b>	<b>ENTER</b>	25
<b>A = W * L</b>	<b>ENTER</b>	500

A simple method for figuring the price of an item, taking the sales tax into consideration, is shown in the next example.

EXAMPLE: the price of an item is \$29.95 and the sales tax is 5%

<b>P = 29.95</b>	<b>ENTER</b>	29.95
<b>P = P*1.05</b>	<b>ENTER</b>	31.4475

Thus, the final price is \$31.45.

Variables are not limited to numerical values; they may also be used with character strings. A character string is a grouping of alphabetic characters, punctuation marks, and numbers. A variable is designated as a string variable by adding a dollar sign (\$) to the variable name. A value is given to the string variable by enclosing the character string in quotation marks.

EXAMPLE:

<b>A\$ = "GOOD MORNING"</b>	<b>ENTER</b>	GOOD MORNING
<b>A\$</b>	<b>ENTER</b>	GOOD MORNING

The naming scheme for variables on the PC-1500A follows the simple and easy to use rules listed below.

- The name may be a letter: A through Z for numbers, and A\$ through Z\$ for strings.
- The name may be a letter followed by another letter or by a single digit (0 through 9)
- A and A\$ are separate variables (the same letter designation may be used for numerical variables and for string variables)
- Because of conflicts with abbreviations in the BASIC programming language, the following variable names may not be used: LE, IF, LN, PI, TO, LF\$, IF\$, LN\$, PI\$, and TO\$.

It is important to note that the character variables will hold only sixteen characters each. The example below illustrates what can happen if attention is not paid to the string length.

EXAMPLE:

**A\$ = "AL DRINKS BUTTERMILK"**

**A\$**

**ENTER**

**ENTER**

**AL DRINKS BUTTER**

## String Concatenation

String concatenation is the process of joining two or more strings together. The PC-1500A uses the plus sign to accomplish this.

EXAMPLE:

```
A$ = "GOOD "  
B$ = "MORNING"  
A$ + B$  
C$ = A$ + B$  
C$
```

ENTER  
ENTER  
ENTER  
ENTER  
ENTER

```
GOOD  
MORNING  
GOOD MORNING  
GOOD MORNING  
GOOD MORNING
```

## The Null String

The null string is a string which contains no characters or blanks. A variable may be defined as a null string by setting the variable equal to a set of quotation marks with no space between them.

EXAMPLE:

```
A$ = ""
```

ENTER

## TIME Function

The PC-1500A has an internal, twenty-four hour clock that may be set by using the statement

TIME = MONTH DAY HOUR . MINUTES SECONDS

The parameters associated with TIME are:

MONTH	A one or two digit number between 1 and 12
DAY	A two digit number between 01 and 31
HOUR	A two digit number between 00 and 23
MINUTE	A two digit number between 00 and 59
SECOND	A two digit number between 00 and 59

To set the time at March 16, 3:57:21 P.M. enter the following.

**TIME = 31615.5721**      **ENTER**

Notice that 3 P.M. was entered in the 24 hour notation as 15. If the month and day are not needed, then they may be left out.

**TIME = 15.5721**      **ENTER**



If the clock is to be used to measure elapsed time, then the TIME function is set at zero.

**TIME = 0**                      **ENTER**

After setting the time, it may be recalled by simply entering the statement

**TIME**                      **ENTER**

EXAMPLE: measure elapsed time by first setting the clock; then, wait ten seconds and recall the time.

<b>TIME = 0</b>	<b>ENTER</b>	0
<b>TIME</b>	<b>ENTER</b>	0.0010

The TIME function may be used both independently and in BASIC programs.

## SECTION 4: MATH FUNCTIONS AND STATEMENTS

Many predefined math functions are available on the Sharp PC-1500A. These functions are also available in the Sharp PC-1500A dialect of the BASIC programming language; each function will act in exactly this way and with the same math hierarchy whether the PC-1500A is being used as a calculator or running a program written in BASIC.

To use the functions as on a calculator, simply enter the function (e.g., type the function into the display) and then enter its argument or the number or variable the function is to act on.

Discussed in this section are:

- Number alteration
- General math functions
- Logarithmic functions
- Trigonometric functions
- Rectangular and polar coordinate conversions
- Total math and logic hierarchy

## Number Alteration

### ABS (Absolute Value)

The ABS function calculates the absolute value of a number or variable. This function will change the sign of all negative values to positive.

EXAMPLE 1: find the absolute value of  $-57$ .

**ABS  $-57$**

**ENTER**

57

EXAMPLE 2: find the absolute value of  $-36.4$ .

**ABS ( $-36.4$ )**

**ENTER**

36.4

EXAMPLE 3: find the absolute value of 67.

**ABS 67**

**ENTER**

67

EXAMPLE 4: find the absolute value of the expression 43 minus 57.

**ABS ( $43 - 57$ )**

**ENTER**

14

EXAMPLE 5: find the absolute value of 43 minus 57.

**ABS 43 – 57**

**ENTER**

**– 14**

As shown in Example 4, if the absolute value is to be determined for an expression, then that expression must be enclosed in parentheses.

### INT (Integer)

The INT function is used to determine the integer portion of a number or variable. To do this, the decimal or fraction portion of a number is dropped from the number.

EXAMPLE 1: find the integer value of 43.671.

**INT 43.671**

**ENTER**

**43**

EXAMPLE 2: find the value of 43.671 divided by 7.0.

**43.671/7.0**

**ENTER**

**6.238714286**

EXAMPLE 3: find the integer value of the expression 43.671/7.0.

**INT (43.671/7.0)**

**ENTER**

**6**

EXAMPLE 4: find the result of dividing the integer value of 43.671 by 7.0.

**INT 43.671/7.0**

**ENTER**

6.142857143

## General Math Functions

The functions to be discussed in this subsection are:

- Sign of a Number
- Square Roots
- Exponentiation
- Pi
- Random Numbers

### SGN (Sign of a Number)

The SGN function will determine the sign of a number or variable and return a value of one (1) for positive numbers, negative one ( - 1) for negative numbers, and zero (0) for zero.

1 if  $X > 0$

0 if  $X = 0$

- 1 if  $X < 0$

Do not sale this PDF !!!

EXAMPLE 1: find the sign of 54.

**SGN 54**

**ENTER**

1

EXAMPLE 2: find the sign of the expression 43 minus 54.

**SGN (43 - 54)**

**ENTER**

- 1

EXAMPLE 3: find the sign of the absolute value of - 99.

**SGN (ABS(- 99))**

**ENTER**

1

EXAMPLE 4: find the sign of the expression 22 minus 22.

**SNG (22 - 22)**

**ENTER**

0

EXAMPLE 5: find the sign of the expression 14 minus 27 multiplied by 52.

**SGN (14 - 27)\*52**

**ENTER**

- 52

**SQR** or  $\sqrt{\square}$  (Square Root)

The Square Root function may be used in two separate ways. One is by entering the letters **SQR** (as with the other math functions) or by using the key marked  $\sqrt{\square}$ : each provides the same result.

EXAMPLE 1: find the square root of 16.

**SHIFT**  $\sqrt{\square}$  **16** **ENTER** 4

EXAMPLE 2: find the square root of the expression 12 plus 13.

**SQR (12 + 13)** **ENTER** 5

EXAMPLE 3: find the square root of the square root of 256.

**SHIFT**  $\sqrt{\square}$  **SHIFT**  $\sqrt{\square}$  **256** **ENTER** 4



EXAMPLE 4: find the square root of 4 multiplied by 4 plus 6 multiplied by 6.

**SQR (4\*4 + 6\*6)**

**ENTER**

7.211102551

EXAMPLE 5: find the square root of 4E minus 50.

**SQR (4E - 50)**

**ENTER**

2E - 25

### Exponentiation (Powers)

To raise a number or a variable to a power, the  $\wedge$  sign is used. Raising a number to a power of two will provide the square of that number. The laws of exponents are listed in the following table.

Table 4.1. Laws of Exponents.

$$A \wedge X * A \wedge Y = A \wedge (X + Y)$$

$$1/(A \wedge X) = A \wedge -X$$

$$(A*B) \wedge X = A \wedge X * B \wedge X$$

$$(A \wedge X)/(A \wedge Y) = A \wedge (X - Y)$$

$$(A \wedge X) \wedge Y = A \wedge (X*Y)$$

$$A \wedge 0 = 1$$

EXAMPLE 1: find the value of 4 squared.

**4  $\wedge$  2**

**ENTER**

16

EXAMPLE 2: find the value of 10 raised to the 25th power

**10 ^ 25**

**ENTER**

**1E 25**

EXAMPLE 3: find the value of 4.5 raised to the power of 6.32.

**4.5 ^ 6.32**

**ENTER**

**13437.02382**

A root of a number may be found by entering the inverse of the root as the power.

EXAMPLE 4: find the value of the square root of 64.

**64 ^ (1/2)**

**ENTER**

**8**

OR

**64 ^ 0.5**

**ENTER**

**8**

OR

**SQR 64**

**ENTER**

**8**

EXAMPLE 5: find the value of the cube root of 64.

$$64 \wedge (1/3)$$

**ENTER**

4

OR

$$64 \wedge 0.33333$$

**ENTER**

3.999944549

It is important to note that rounding decimal numbers can cause inaccurate results, as demonstrated in Example 5.

EXAMPLE 6: find the value of 23 raised to the 0.234 power.

$$23 \wedge 0.234$$

**ENTER**

2.082784388

The inverse of a number raised to a power is found by entering the number and raising it to a negative power.

EXAMPLE 7: find the value of the inverse of 4.3 raised to a power of 3.

$$1/(4.3 \wedge 3)$$

**ENTER**

1.25775089E - 02

OR

$$4.3 \wedge -3$$

**ENTER**

1.25775089E - 02

OR

**(1/4.3) ^ 3**

**ENTER**

1.25775089E - 02

### EXP (Power of e)

The EXP function raises the constant,  $e$  (2.718281828) to a power. The result of this calculation is the natural antilogarithm.

$$e^x = 1 + x + (x^2)/2! + (x^3)/3! + \dots$$

EXAMPLE 1:

**EXP 1**

**ENTER**

2.718281828

EXAMPLE 2:

**EXP 0**

**ENTER**

1

EXAMPLE 3:

**EXP 6.2146**

**ENTER**

499.9959508

EXAMPLE 4:

**EXP (4 + 20/6)****ENTER**

1530.474862

EXAMPLE 5:

**EXP - 58****ENTER**

6.470234927E - 26

EXAMPLE 6:

**EXP (LN 40)****ENTER**

40

**PI**

The value of Pi (3.141592654) is stored in both the PI function and the  $\pi$  key as a fixed constant. Either the function or the key can be used in calculations where the value of Pi is needed.

EXAMPLE: find the area of a circle with a 5-foot diameter.

**PI\*(5/2) ^ 2****ENTER**

19.63495408

OR

 $\pi$   
**↵\*(5/2) ^ 2****ENTER**

19.63495408

## Random Numbers

There may be times when you want to provide your program with random numbers. The RND function allows the PC-1500A to generate random numbers in a range from one to a specified number. NOTE: The range always starts with one.

EXAMPLE 1: find a random number between 1 and 5.

**RND 5**

**ENTER**

3

Random numbers are generated through a mathematical formula and are accessible by using the RND function. When the PC-1500A is turned on, a series of random numbers is generated by the computer. This list remains unchanged unless the RANDOM function is used. This means that a program will use the same series of "random" numbers each time the computer is turned on. To prevent this, the RANDOM function resets the "seed" used by the formula to generate its random numbers.

EXAMPLE 2: find a random number between 1 and 10.

**RANDOM**

**ENTER**

**RND 10**

**ENTER**

9

## Logarithmic Functions

The logarithm of a number is the exponent of a base number (10 or e) raised to a power which generates the original number. The laws of logarithms are in the following table.

Table 4.2. Laws of Logarithms.

$$\text{LN } (Y^X) = X * \text{LN } Y$$

$$\text{LOG } (A * B) = \text{LOG } A + \text{LOG } B$$

$$\text{LN } (A/B) = \text{LN } A - \text{LN } B$$

### LN (Natural Logarithms)

The function LN computes the natural logarithm (base e, 2.718281828) of a number. Any attempt to take the logarithm of a negative number will result in an error.

EXAMPLE 1:

**LN 2.3**

**ENTER**

8.329091229E - 01

EXAMPLE 2:

**LN (7 + 24)**

**ENTER**

3.433987204

EXAMPLE 3:

**LN (EXP 10)**

**ENTER**

10



EXAMPLE 4:

**LN (12E - 16)**      **ENTER**      - 34.35645484

### LOG (Common Logarithms)

The function LOG computes the common logarithm (base 10) of a number. Any attempt to take the logarithm of a negative number will result in an error.

EXAMPLE 1:

**LOG 2.3**      **ENTER**      0.361727836

EXAMPLE 2:

**LOG (7 + 24)**      **ENTER**      1.491361694

EXAMPLE 3:

**LOG (10 ^ 52)**      **ENTER**      52

EXAMPLE 4:

**LOG (12E - 16)**      **ENTER**      - 14.92081875

EXAMPLE 5:

**LOG (LOG 106)****ENTER**

3.064906204E – 01

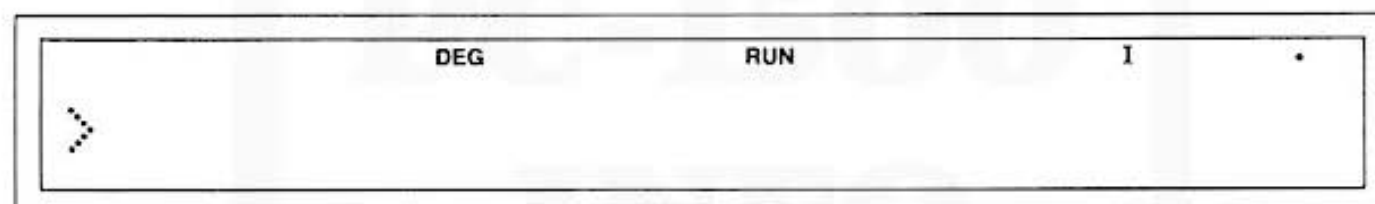
EXAMPLE 6:

**LOG (93 ^ 23)****ENTER**

45.27510782

### Angle Designations

The PC-1500A allows the trigonometric and angular functions to be calculated in degrees, radians, or gradients. The display shows the angular designation as:



To designate degrees:

**DEGREE****ENTER**

To designate radians:

**RADIAN****ENTER**

To designate gradients:

**GRAD** **ENTER**

### DMS (Degrees, Minutes, and Seconds)

Normally, when the PC-1500A is designated to calculate in degrees, the entry is in decimal form. To convert degrees from the decimal form to degrees, minutes, and seconds, the DMS function is used.

EXAMPLE 1: convert 30.66 to degrees, minutes, and seconds.

**DMS 30.66** **ENTER** 30.3936

Example 1 demonstrates this conversion; 30.66 is equivalent to  $30^{\circ} 39' 36''$ . The integer portion of the DMS result represents degrees, the first two digits of the decimal portion represent minutes, and the remainder is seconds in decimal form.

EXAMPLE 2: convert 22.2165 to degrees, minutes, and seconds.

**DMS 22.2165** **ENTER** 22.12594

In Example 2, the DMS form of 22.2165 is  $22^{\circ} 12' 59.4''$ .

**DEG (Decimal Degrees)**

To convert the degree, minute, and second form of an angle to the decimal degree form, the DEG function is used.

Example 1: convert  $20^{\circ} 5' 22''$  to decimal degrees.

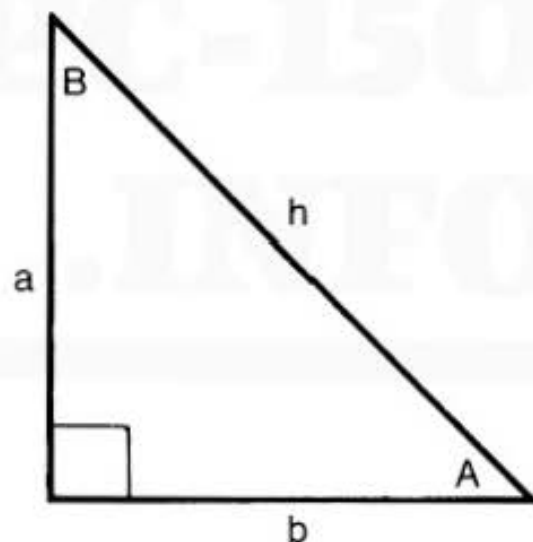
**DEG 20.0522**

**ENTER**

20.08944444

**Trigonometric Functions**

The trigonometric functions can be defined geometrically by analyzing a right triangle as defined by exponential derivations.



Sine	=	SIN A	=	a/h
Cosine	=	COS A	=	b/h
Tangent	=	TAN A	=	a/b
Cotangent	=	COT A	=	1/TAN A
Secant	=	SEC A	=	1/COS A
Cosecant	=	CSC A	=	1/SIN A
Exsecant	=	EXSEC A	=	SEC A - 1
Versine	=	VERS A	=	A - COS A
Coversine	=	COVERS A	=	1 - SIN A
Haversine	=	HAV A	=	1/2* VERS A

The inverse trigonometric functions return the value of an angle if the ratios of two sides are given.

$$\text{Arcsine} \quad = \quad \text{ASN (b/h)} \quad = \quad A$$

Do not sale this PDF !!!

$$\text{Arccosine} \quad = \quad \text{ACS (a/h)} \quad = \quad A$$

$$\text{Arctangent} \quad = \quad \text{ATN (b/a)} \quad = \quad A$$

The hyperbolic trigonometric functions are defined by the following exponential relationships.

$$\text{SINH X} \quad = \quad 1/2 * (\text{EXP X} - \text{EXP} - X)$$

$$\text{COSH X} \quad = \quad 1/2 * (\text{EXP X} + \text{EXP} - X)$$

$$\text{TANH X} \quad = \quad (\text{EXP X} - \text{EXP} - X) / (\text{EXP X} + \text{EXP} - X)$$

The inverse hyperbolic functions are defined by the following logarithmic relationships.

$$\text{SINH}^{-1} X \quad = \quad \text{LN}(X + \text{SQR}(X^2 + 1))$$

$$\text{COSH}^{-1} X \quad = \quad \text{LN}(X + \text{SQR}(X^2 - 1))$$

$$\text{TANH}^{-1} X \quad = \quad 1/2 * \text{LN}[(1 + X)/(1 - X)]$$

Since all the trigonometric functions can be derived from the three basic functions (sine, cosine, and tangent) or by using the LN and EXP functions, only the basic trigonometric and inverse functions are directly available on the PC-1500A and in the BASIC programming language.

## SIN

The SIN function derives the value of sine for a given angle or variable.

EXAMPLE 1: find the sine of 30.

**SIN 30**

**ENTER**

0.5

EXAMPLE 2: find the sine of 30.5.

**SIN 30.5**

**ENTER**

0.507538363

EXAMPLE 3: find the sine of 30°30'0".

**SIN (DMS 30.3000)**

**ENTER**

5.027182272E – 01

## COS

The COS function derives the value of cosine for a given angle or variable.

EXAMPLE 1: find the cosine of  $\pi/3$ .

**RADIAN**

**ENTER**

**COS ( $\pi/3$ )**

**ENTER**

9.998329795E – 01



**TAN**

The TAN function derives the value of the tangent for a given angle or variable.

EXAMPLE 1: find the tangent for  $27^{\circ} 14' 37''$ .

**TAN (DEG 27.1437)**      **ENTER**      5.148927659E – 01

**Inverse Trigonometric Functions**

The inverse trigonometric functions available on the Sharp PC-1500A are the arcsine, ASN, the arccosine, ACS, and the arctangent, ATN.

EXAMPLE 1: find the arctangent of 0.07.

**ATN .07**      **ENTER**      4.004172941

EXAMPLE 2: find the arcsine of the sine of 30.

**ASN (SIN 30)**      **ENTER**      30

EXAMPLE 3: find the arccosine of  $-0.99$ .

**ACS – .99**      **ENTER**      171.8903855

## Rectangular and Polar Coordinate Conversions

Sometimes, when working with two and three dimensional concepts, it is convenient to view points, lines, planes, and solids in a mathematical form other than the conventional rectangular system in which all the coordinates are in terms of unit length. Figure 4.1 shows a point located by rectangular coordinates  $(x,y)$  while Figure 4.2 shows a point located by two-dimensional polar coordinates  $(r,\theta)$

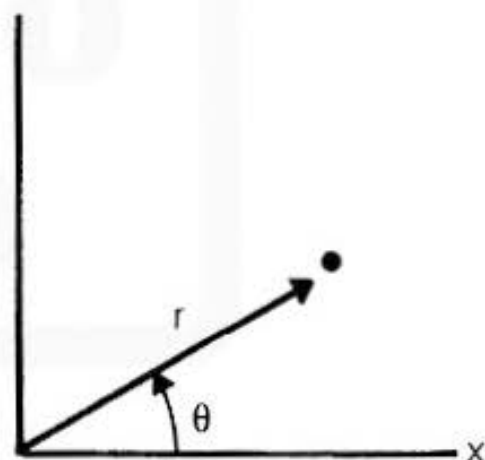
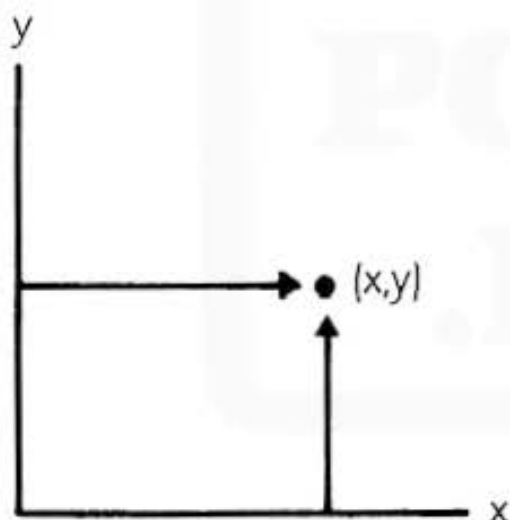


Figure 4.1. Rectangular coordinates.

Figure 4.2. Two-dimensional polar coordinates.

Do not sale this PDF !!!

The formulas for converting two-dimensional rectangular coordinates are in Table 4.3. Figure 4.3 illustrates three-dimensional coordinate systems. The formulas for these conversions are in Table 4.4.

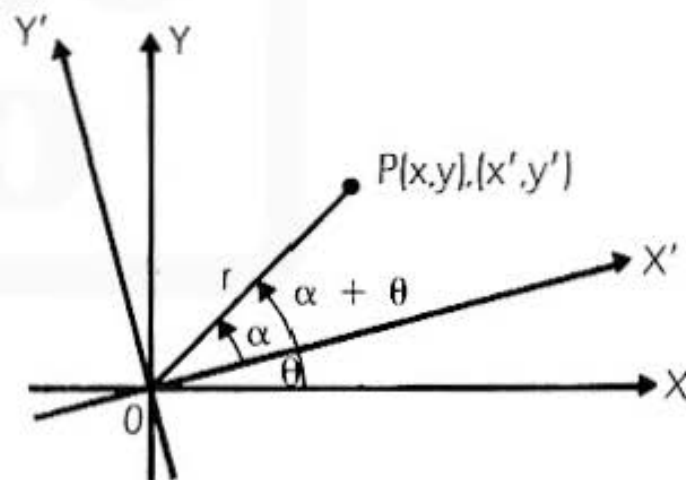
Table 4.3. Two-dimensional coordinate formulas.

Relation between Rectangular and Polar Coordinates

$$\begin{aligned} x &= r \cos \theta & r &= \sqrt{x^2 + y^2} & \sin \theta &= \frac{y}{\sqrt{x^2 + y^2}} \\ y &= r \sin \theta & \theta &= \tan^{-1} \frac{y}{x} & \cos \theta &= \frac{x}{\sqrt{x^2 + y^2}} \end{aligned}$$

Rotation of Polar Axes through Angle  $\theta$

$$\begin{aligned} x' &= r \cos \alpha \\ y' &= r \sin \alpha \\ x &= r \cos (\alpha + \theta) \\ y &= r \sin (\alpha + \theta) \end{aligned}$$



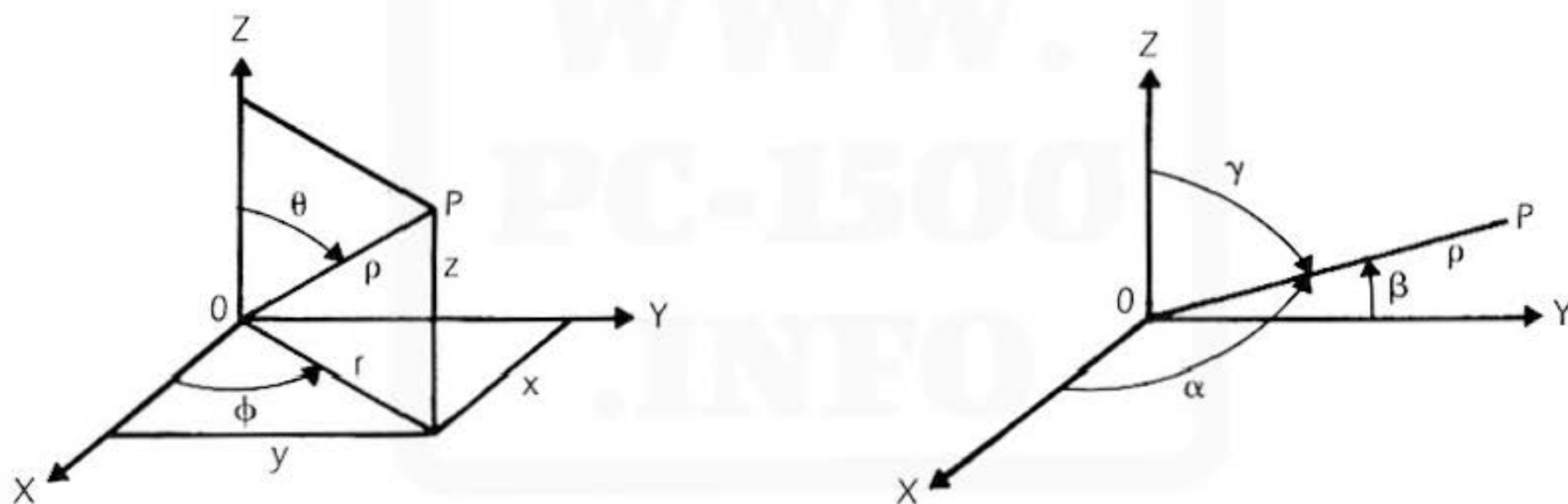


Figure 4.3. Three-dimensional coordinate systems.  
Do not sale this PDF !!!

Table 4.4. Three-dimensional coordinate formulas.

Relations of Coordinates of Systems in Terms of  $x, y, z$ 

Cylindrical	Spherical	Polar Space
$r = \sqrt{x^2 + y^2}$	$\rho = \sqrt{x^2 + y^2 + z^2}$	$\rho = \sqrt{x^2 + y^2 + z^2}$
$\phi = \tan^{-1} \frac{y}{x}$	$\theta = \cos^{-1} \left( \frac{z}{\sqrt{x^2 + y^2 + z^2}} \right)$	$\alpha = \cos^{-1} \left( \frac{x}{\sqrt{x^2 + y^2 + z^2}} \right)$
$z = z$	$\phi = \tan^{-1} \frac{y}{x}$	$\beta = \cos^{-1} \left( \frac{y}{\sqrt{x^2 + y^2 + z^2}} \right)$
		$\gamma = \cos^{-1} \left( \frac{z}{\sqrt{x^2 + y^2 + z^2}} \right)$

## Total Math and Logic Hierarchy

Calculations and logical operations are performed in the order of the hierarchy listed below with expressions in parentheses having the highest priority and the logical operations the lowest. If two or more operations have the same priority, then they are evaluated from left to right.

1. Expressions in parentheses
2. Retrieval of values from variables
3. Functions (e.g., SIN, COS, etc.)
4. Exponentiation
5. Sign of a number
6. Multiplication and division
7. Addition and subtraction
8. Comparison operators (<, <=, =, >, >=, <>)
9. Logical operators

It is important to note that the sign of a number is evaluated after exponentiation and it is, therefore, necessary to enclose a negative number in parentheses.

EXAMPLE:

$$-(6*2)^3 * 6 + \sin 30 / \sqrt{36} + \cos 60 * \pi$$

$$-12^3 * 6 + \sin 30 / \sqrt{36} + \cos 60 * \pi$$

$$-12^3 * 6 + \sin 30 / \sqrt{36} + \cos 60 * 3.141592654$$

$$-12^3 * 6 + 0.5 / 6 + 0.5 * 3.141592654$$

$$-1728 * 6 + 0.5 / 6 + 0.5 * 3.141592654$$

$$-10368 + 8.333333333E-02 + 1.570796327$$

$$-10366.34587$$

## **PART II: BASIC PROGRAMMING WITH THE SHARP PC-1500A**

This part of the manual is dedicated to the use of the Sharp PC-1500A as a stand alone, handheld computer. The PC-1500A dialect of the BASIC programming language is a simple, easy to learn and very powerful language. With the PC-1500A and BASIC programming language, you will find that complex and detailed calculations will become rapid and straightforward. The PC-1500A dialect of BASIC works equally well for both business and technical applications.

Discussed in this part are:

- Simple Programming
- Program Editing
- Branching
- Loops
- Strings
- Arrays
- Display Editing
- Debugging



## SECTION 5: SIMPLE PROGRAMMING

This section will deal with the concepts and instructions needed to begin using the computer. Using this section, you will learn how to organize and write a computer program and how to use the fundamental BASIC statements. A simple program is included as an example.

Discussed in this section are:

- Statements
- Commands
- Writing a Program
- Entering a Program
- Order of Program Execution
- Abbreviations

### What is a BASIC Program?

A BASIC program is a list of instructions to the computer written in the BASIC language. The instructions tell the computer what to do, when to do it, and how to go about doing it. Always keep in mind that the computer is a faithful, though not too bright, servant; it will do exactly what you tell it to do, no more and no less. It is, therefore, important that your program be organized and clearly thought out to obtain the results you want.

## Statements

A program statement is an instruction given to the computer to be executed at the time the program is run. Generally, statements are entered in the program preceded by statement numbers (indicating the order in which they are to be executed).

### EXAMPLE:

```
10: PRINT "GOOD AFTERNOON"  
20: PRINT "HOW ARE YOU?"
```

A list of the instructions found in Sharp PC-1500A BASIC and a brief explanation of each may be found in Appendix I. Most statements may also be executed from the keyboard.

## Commands

A command is an instruction to the computer that is to be executed immediately. Commands are used to manipulate the program and the operation of the computer. Examples of commands are RUN, LIST, and NEW. A complete list of the available commands is in Appendix I.

## Statement Numbers

A BASIC program consists of numbered lines containing one or more statements. The statement number (also referred to as a line number) is used by the computer to run the program steps in the correct sequence. The listing below demonstrates the use of line numbers.

## LISTING

```
10: INPUT "END TIME? ";E
20: TIME = 0
30: X = TIME
40: PAUSE X
50: IF X >= E THEN BEEP 5: GOTO 20
60: GOTO 30
```

On the PC-1500A, a line number can be any number between 1 and 65279. Generally, when writing a program, lines are numbered in increments of 10 so that up to 9 additional lines can be added between each statement. The need for this will become immediately obvious when it comes time for you to write and debug your first program.

**NEW (Clearing the Computer Memory)**

To prepare the computer to begin a new program, the computer memory should be cleared (erased) to make room for the new program and new program data. This is done by placing the computer in the PROGRAM mode and entering the NEW command.

**NEW**     **ENTER**

This will remove any existing program lines from the computer memory and clear all the data registers.

## Writing a Program

As discussed in Section 1, there are five essential steps to writing a program. They are:

- 1) Problem definition
- 2) Approach definition
- 3) Logic and program definition
- 4) Program coding
- 5) Program testing

### Problem Definition

The problem definition is a complete description of what you wish to do with the computer. It includes what information is available to you for solving the problem, what you want the computer to tell you, and what intermediate calculations must be made. As an example, suppose that you wish to buy a house and need to know what the monthly payments on the house will be. The problem definition is broken down in the following manner.

#### Information Available

Price of house	\$75,000
Down payment	5%
Mortgage term	30 years
Interest rate	11.8%
Taxes and Insurance	32% of mortgage payment

### Desired Information

Monthly house payment

### Intermediate Determinations

Amount of house to be financed

Amount of down payment

Mortgage payment

Amount of taxes and insurance

### Approach Definition

The approach definition is a description of how the program is to be solved. In most cases, the approach is closely related to the mathematical formulas to be used in the program. Also included would be the method of data entry and the type of output desired.

Continuing with the house payment example, the desired method of data entry will be through the keyboard since you wish to compare the payments for several different houses. The output will be in the computer display.

## Mathematical Formulas

### Mortgage payment

$$PMT = PV \frac{I}{1 - (1 + I)^{-N}}$$

where

PMT	= Mortgage payment
PV	= Amount of house to be financed
I	= Periodic interest rate ( $I > 0$ )
N	= Number of payment periods

### Amount to be financed

$$PV = P - Dn * P$$

where

PV	= Amount to be financed
P	= Price of house
Dn	= % down payment

### Periodic interest rate

$$I = An/12$$

where

$I$  = Periodic interest rate

$An$  = Annual interest rate

### Payment periods

$$N = Yr * 12$$

where

$N$  = Number of payment periods

$Yr$  = Years in mortgage term

### Taxes and insurance

$$Tx = PMT * Tr$$

where

$T_x$  = Amount of taxes and insurance to be paid  
 $T_r$  = % taxes and insurance of payment  
 $PMT$  = Monthly mortgage payment

Monthly house payment

$HP = PMT + T_x$

where

$HP$  = House payment  
 $PMT$  = Monthly mortgage payment  
 $T_x$  = Taxes and insurance

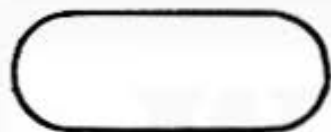


**Logic and Program Definition**

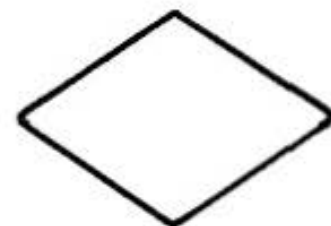
The program logic definition is a description of how the program will run (the program flow). One of the best methods for doing this is with a program flowchart. The following table shows the common flowchart symbols.

Table 5.1. Flowchart symbols.

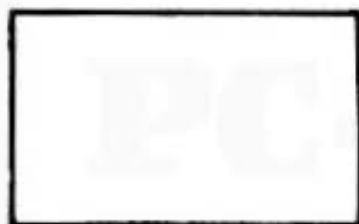
START/STOP



DECISION/BRANCHING



PROCESS INFORMATION



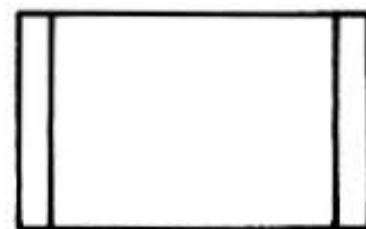
CONNECTOR



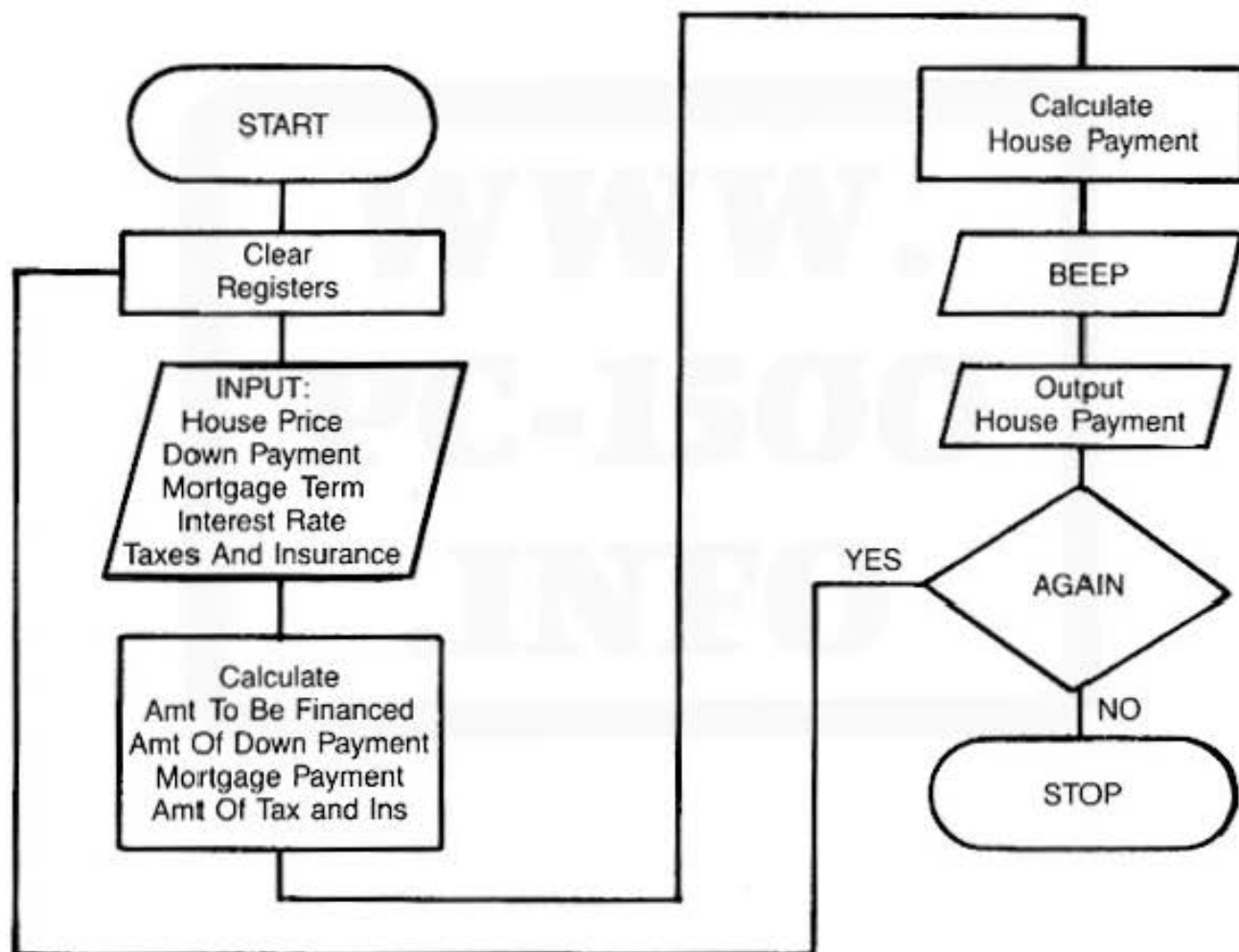
INPUT/OUTPUT



SUBROUTINE



A flowchart of the house payment calculation would appear as:



## Program Coding

The program code is entered into the computer using the commands and statements discussed in this manual. Remember that the computer must be in PRO mode for the code to be entered.

A listing of our sample program is shown here to help you design and write your own programs.

```
10: REM PROGRAM FOR FINDING HOUSE PAYMENTS
20: CLEAR
30: INPUT "PRICE = ";P
40: INPUT "ANNUAL INTEREST % = ";AN
50: INPUT "% DOWN PAYMENT = ";DN
60: INPUT "MORTGAGE TERM = ";YR
70: INPUT "% TAX & INS = ";TR
80: I = AN/1200
90: PV = P - ((DN/100)*P)
100: N = YR * 12
110: PMT = PV*(I/(1 - ((1 + I) ^ - N)))
120: TX = PMT*(TR/100)
130: HP = PMT + TX
140: PRINT "PAYMENT = ";HP
150: INPUT "DO IT AGAIN? ";X$:IF X$ = "Y" THEN GOTO 20
160: END
```

## Program Testing

In a program that involves calculations, the results should always be compared to previously calculated values to verify that the program is coded correctly.

It is quite common for experienced as well as novice programmers to write programs which contain errors or "bugs." These errors can be attributed to a range of problems such as faulty logic or sloppy housekeeping. Or, the error can be the result of a misunderstanding of the computer's functioning. Very rarely can the problem be blamed on the computer.

One of the best insurances against common mistakes is flowcharting. With a flowchart, a particular program's logic can be examined on paper, step by step. With pencil and paper, the exact flow of a program under various conditions can be charted. Although this involves an investment of time, the rewards are generally well worth the effort. In those cases where errors slip by and begin to plague the programmer, there are several methods for determining where and how the error occurs.

One of the most useful methods for testing a program for accuracy is to channel known data (with known and expected results) through the program. If the results are not as intended, then an error has occurred either during programming or data entry. To find the error, a logical approach to determine and find the error must be used. A "scatter-shot" approach often wastes a lot of time and yields minimal results.

This is not to overlook a particular section of a program that was suspect before the program was executed or a particular error in a known region of a program. Using clues given by the program is part of the logical approach. Only when apparent clues are not available does an indepth analysis of the program become necessary. The

following techniques can be applied to get more information out of one clue or to generate clues when there are none.

Incorrect program flow is a common error. The commands TRON and TROFF are useful means to watch the exact flow of a program. The program may be single-stepped, or executed one step or line at a time, in the TRON mode by using the Down Arrow key. The current line of execution can be displayed by pressing the Up Arrow key. While in this mode, and stopped between steps, specific registers can be examined by simply typing their variable names into the display followed by pressing the **ENTER** key.

Another method for following the program's activities is to insert STOP commands at strategic locations such that calculations and data storage can be examined when the STOP is executed. The program may then be continued with the CONT command. Like the TRON method, the data registers may be examined by typing the variable name into the display followed by pressing the **ENTER** key. This method may be used when program flow is not as much in question as data storage.

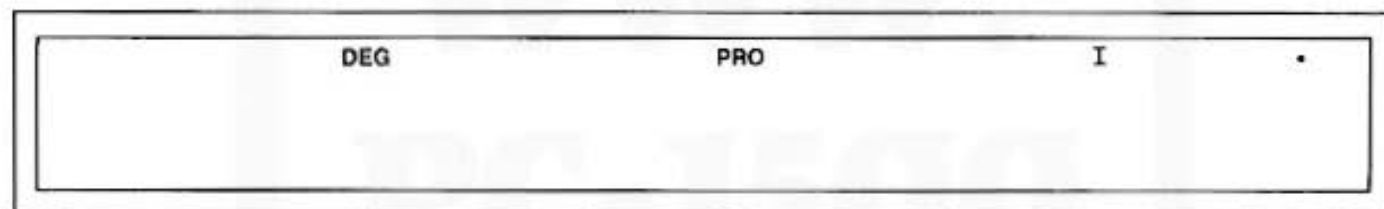
Another variation of the break-point method with the STOP command is to place PRINT commands within the body of the program to display or print out data register values and arithmetic results as the program flows naturally. This method affords minimum user interaction except for the examination of the printout or display. The disadvantage of this method is that very little information about program flow is given.

The final method is one which can be applicably named a "brute-force" method. This method requires that the programmer, with pencil and paper, simulate the computer, step by step, omitting nor assuming any sequence the computer makes. All data register storage and arithmetic are noted on the sheet of paper. The computer serves to

execute the calculations, but **all** of the program flow control is in the hands of the programmer. This method rarely fails to determine where an error may occur, but the time required to execute such a procedure is quite lengthy.

## Entering a Program

To enter program statements into the computer, the computer must first be placed in the PROGRAM mode using the **MODE** key. The display will appear as in the following illustration.



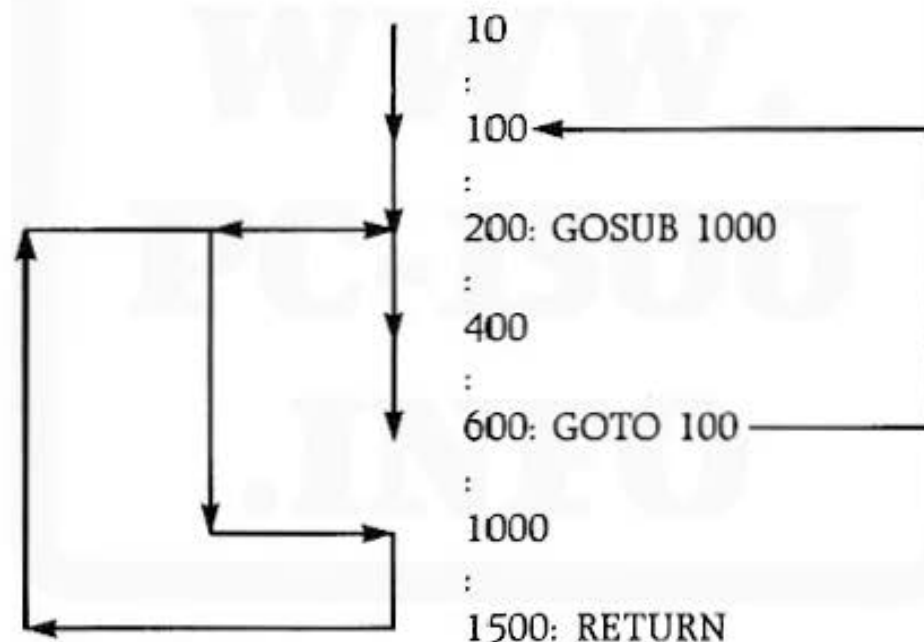
Each numbered statement can be as long as eighty characters in length, including the statement number and punctuation.

## Running a Program

After a program has been entered, it may be run by first placing the computer in RUN mode using the **MODE** key, and then by typing the command, **RUN**, and pressing the **ENTER** key.

## Order of Program Execution

The PC-1500A executes statements in the order of ascending statement numbers. When a branch is encountered the program will jump to that statement number and again process the statements in ascending order.



## Fundamental Program Statements

### REM (Remarks)

The REM statement is used to insert comments among the program statements. These comments are ignored by the PC-1500A. Their purpose is to let the programmer explain the workings of this program to other people who would use the program or to himself. A program fresh in the programmer's mind is a simple thing to run or change; but with the passage of time, the program logic may be forgotten. It is highly recommended that remark statements be used wherever possible. An example of this is shown in the following listing.

```
10: REM PROGRAM F
    OR FINDING HOU
    SE PAYMENTS
20: CLEAR :REM CL
    EAR REGISTERS
25: REM      ***INPUT
    ROUTINES***
30: INPUT    "PRICE =
    ";P
40: INPUT    "ANNUAL
    INTEREST  % = ";
    AN
50: INPUT    "%DOWN P
    MT = ";DN
60: INPUT    "MORTGAG
    E TERM = ";YR
70: INPUT    "% TAX &
    INS = "; TR
75: REM      ***CALCUL
    ATIONS***
80: I = AN/1200 : REM
    PERIODIC INTE
    REST
90: PV = P - ((DN/100)
    *P):REM      FINAN
    CED AMOUNT
```



```
100 100: N=YR*12:REM  #  
      OF PMT PERIOD  
      S  
110: PMT=PV*(I/(1-(  
      (1+I)^-N))):  
      REM MORTGAGE  
      PMT  
120: TX=PMT*(TR/100  
      ):REM TAX&INS  
      AMOUNT  
130: HP=PMT+TX:REM  
      HOUSE PMT  
140: PRINT "PAYMENT  
      = ";HP  
150: INPUT "DO IT A  
      GAIN";X$:IF X$  
      ="Y"THEN GOTO  
      20  
160: END
```

When the REM statement is used in conjunction with other statements on a line, it should be the last statement. The Sharp PC-1500A will ignore everything after REM.

## PRINT

### PRINT item

The PRINT statement is the primary means for letting the PC-1500A communicate with the outside world. The general form of the print statement is the word PRINT followed by a number, character string, variable, or string variable, or any combination of these separated by a comma or a semicolon. The following sample programs show the different uses of the PRINT statement.

```
10: X = 0
20: PRINT X
30: X = X + 1
40: GOTO 20
```

The display prompt shows the value of X in the right-hand portion of the display, as follows.

3

Also, after a PRINT statement, the program will not proceed until the **ENTER** key is pressed.

When printing character strings, the string must be enclosed in quotation marks, as follows.

```
10: PRINT "HELLO"
```

The next example program shows the use of the comma in the PRINT statement. Two different items can be displayed at the same time.

```
10: X$ = "FIRST"  
20: Y$ = "SECOND"  
30: PRINT X$, Y$  
40: END
```

The display prompt will show

FIRST	SECOND
-------	--------

Another example program is as follows.

```
10: X = 10  
20: Y = 20  
30: PRINT X, Y  
40: END
```

The display would be as follows.

10	20
----	----

The most versatile function used with the PRINT statement is the semicolon (;). This form of the PRINT statement will allow you to print more than two items in the same display. The following examples show the different ways that the semicolon may be used.

```
10: X = 10
20: X$ = "X = "
30: PRINT X$;X
40: END
```

```
10: INPUT"YOUR NAME IS?";X$
20: PRINT"GOOD MORNING";X$
30: END
```

```
10: TIME = 0
20: X = 10000*TIME
30: PRINT X;
40: GOTO 20
```

```
10: X = 1
20: Y = X ^ 3
30: PRINT"F(";X;") = ";Y
40: X = X + 1
50: GOTO 20
```

**PAUSE**

The PAUSE statement is identical to the PRINT statement with one exception: the user does not have to press the **ENTER** key to advance to the next program line. The item in the statement will be displayed for a brief and fixed period of time. An example follows.

```
10: TIME = 0
20: X = TIME*10000
30: PAUSE"DO";
40: PAUSE"RE";
50: PAUSE"MI";
60: PAUSE"FA";
70: PAUSE"SO";
80: PAUSE"LA";
90: PAUSE"TI";
100: PAUSE"DO";
110: PAUSE X
120: GOTO 20
```

## INPUT

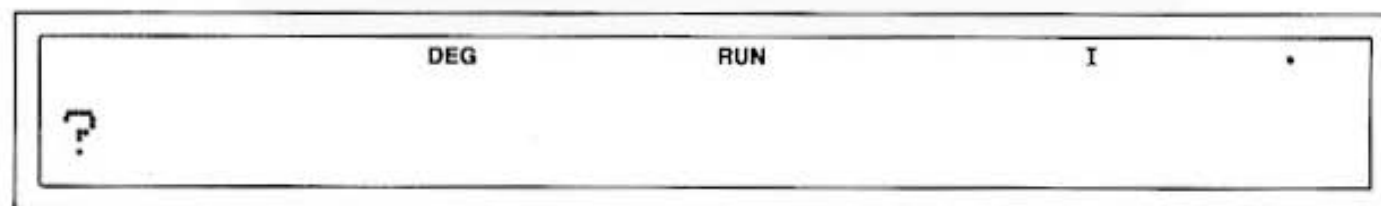
The INPUT statement is the most common way for you to provide information to the PC-1500A through a BASIC program. In its simplest form

INPUT variable

the computer will print a ? (question mark) in the display and then wait for information to be entered. An example follows.

```
10: A = 0
20: INPUT A
30: PRINT A*PI
40: GOTO 10
```

When the program reaches statement number 20, the display will appear as illustrated below.



It is easy to see that a program using this form of the INPUT statement can rapidly become confusing because the ? prompt does not tell you what information you are to enter. It is always a good programming practice to use a descriptive prompting statement. When using this form of INPUT, a PRINT or PAUSE statement may be used to provide a user prompt.

```
10: A = 0
20: PAUSE"ENTER ANY NUMBER"
30: INPUT A
40: PRINT A*PI
50: GOTO 10
```

The next form of the INPUT statement includes a character string followed by a semicolon.

INPUT"character string"; variable

In this form, the character string will replace the ? as the input prompt. The semicolon indicates that the user input will be shown in the same display as the input prompt. This form of the INPUT statement can be demonstrated by rewriting the previous example in the following manner.

```
10: A = 0
20: INPUT"ENTER ANY NUMBER"; A
30: PRINT A*PI
40: GOTO 10
```

The final form of the INPUT statement replaces the semicolon with a comma.

INPUT "character string", variable

The comma directs the display to be cleared before the user input is displayed.

## **BEEP**

The PC-1500A contains a speaker and tone generator that allows the user to add sound to any program. The tones may be created at any of 256 frequencies with a range of 230 Hz to 7kHz. The number of repetitions and duration of each tone are also possible to control from the program.

BEEP repetitions, frequency, duration

Where:

Repetition is the only parameter that must be a part of the BEEP statement: the allowable range is 0 to 65,535.

Frequency is optional; the allowable range is 0 to 255.

Duration is optional; the allowable range is 0 to 65,279 (approximately 30 seconds)



The BEEP statement can also be used to turn the PC-1500A's internal speaker on and off. Thus, the sound made by the cassette CSAVE or CLOAD operation can be eliminated.

The format form of the BEEP statement is

or

BEEP ON

BEEP OFF

When the PC-1500A is turned OFF and then ON, the speaker is restored to the ON position. An example involving electronic music follows.

```
10: DIM A(100)\B(100)
20: FOR X= 1 TO 100
30: A(X)= 50 + INT(RND 25);B(X)= INT(RND 150)
40: NEXT X
50: FOR X= 1 TO 100
60: BEEP 1,A(X)\B(X)
70: NEXT X
80: GOTO 50
```

## WAIT

### WAIT argument

The WAIT statement is used to change the operation of the PRINT statement. The information displayed when using the PRINT statement will remain in the display for the period of time specified by the argument in the WAIT statement.

The WAIT statement argument is optional. When no argument is given, the wait time is infinite and the **ENTER** key must be pressed for the program to proceed past a PRINT statement. The argument for the WAIT statement can be any number between 0 and 65,535. An argument of zero will cause the expression being printed to be displayed so fast that it cannot be read and an argument of 65,535 will cause the expression being printed to be displayed for approximately 17 minutes. A value of 64 will activate the display for one second and a value of 3,840 will activate the display for one minute.

The following program listing shows the effect of the WAIT statement on the PRINT statement.

```
10: FOR W=0 TO 102 STEP 2
20: WAIT W
30: BEEP 1,5
40: PRINT W
50: NEXT W
60: END
```

**LET**

LET variable = expression

The LET statement is used to assign a value to a variable. In most cases, the LET statement is optional. The statement

10: LET X = 15

is the same as the statement

10: X = 15

The only exception to this is when a variable is assigned in an IF...THEN statement

10: IF Z = 15 THEN LET X = 3

When the LET statement is used in conjunction with the IF...THEN statement, the word THEN may be omitted.

10: IF Z = 15 LET X = 3

If the LET statement is omitted from the IF... THEN statement, an ERROR 19 will result.

## GOTO

### GOTO expression

The GOTO statement is used to direct the flow of the program. The GOTO statement is commonly called an unconditional branch; whenever the GOTO statement is encountered, the program will jump to the line number designated by the GOTO statement.

The expression in a GOTO statement may be either a program line number or the label for a program line. The following program listings demonstrate the uses of the GOTO statement.

```
10: X = 0
20: PAUSE"FINDING SQR ROOT"
30: INPUT"ENTER SQUARE";X
40: X = SQR X
50: PRINT"SQR = ";X
60: GOTO 10
```

```
10: X = 0
20: "A": PAUSE"FINDING CUBES"
30: INPUT"ENTER NUMBER ";X
40: X = X*X*X
50: PRINT"CUBE = ";X
60: GOTO"A"
```

The GOTO statement also may be used with an IF... THEN statement. As with the LET statement, the word THEN is optional when using a GOTO statement. The following listing illustrates the use of the GOTO statement along with an IF... THEN statement.

```
10: INPUT"INITIAL BALANCE?".B
20: INPUT"TRANSACTION AMOUNT?".TA
25: IF TA = 0 THEN GOTO 80
30: B = B + TA
40: IF TA < 0 THEN GOTO 70
50: PRINT"DEPOSIT OF $":TA:" POSTED"
60: GOTO 80
70: PRINT ABS(TA);"DOLLARS WITHDRAWN"
80: PRINT"FINAL BALANCE = ";B
90: END
```

## END

The END statement designates the end of the program and stops the program execution.

## Multistatement Lines

line number: statement 1 : statement 2 : statement 3

In the PC-1500A, several statements may share the same line. The computer will execute the statements in sequence from left to right. Each statement is separated by a colon as in the listing below.

```
10: N1 = 0 : N2 = 0 : N3 = 0
20: INPUT "ENTER 3 NUMBERS", N1, N2, N3
30: A = (N1 + N2 + N3) / 3
40: D1 = N1 - A : D2 = N2 - A : D3 = N3 - A
50: SD = D1 + D2 + D3
60: PRINT "SUM OF DIFFERENCE = "; SD
```

The purpose for allowing multistatement lines is so that compact and economical program coding can be created. By writing economical programs, less computer memory is used and, therefore, more detailed or complex programs can be generated. The disadvantage to incorporating multistatement lines in a normal programming style is that the program coding will be difficult to follow.

## Abbreviations

The PC-1500A provides the programmer with the ability to enter program coding with a minimum of keystrokes by using abbreviations for certain commands and statements. Unlike multistatement lines, abbreviations will not save on computer memory. A few of the abbreviations available are shown in Table 5.2. A complete list of abbreviations is found in Appendix C.

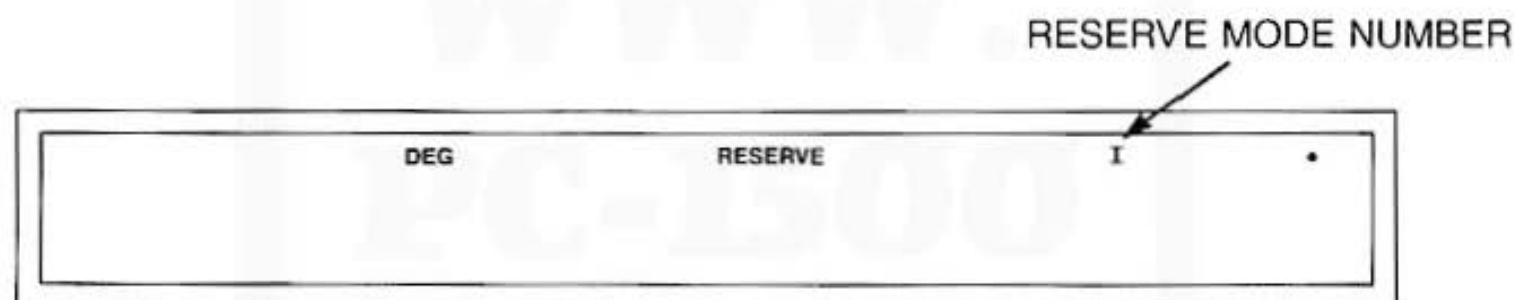
Table 5.2. Partial listing of abbreviations.

PRINT	P. PR. PRI.
PAUSE	PA. PAU.
INPUT	I. IN. INP.
RUN	R.

## Reserve Mode

The RESERVE mode will allow you to assign commands, statements, or values to the function keys (F1 through F6) in any of three RESERVE modes (I, II, and III) for a total of eighteen function keys.

To program the RESERVE mode, simply press **SHIFT** **MODE** and the display mode indicator will display the word RESERVE. To shift from one RESERVE mode to another, press **↕** and observe the RESERVE mode indicator in the upper right-hand portion of the display.



Any of the statements or commands discussed in this manual may be assigned to a RESERVE mode function key as well as any allowable value. Function key assignments are limited to a maximum of 77 characters (one function key cannot exceed 77 characters and the total characters assigned to all eighteen function keys cannot exceed 77 characters).



## Using the @ Sign

The @ (at sign) may be added to the end of a line as a substitute for pressing the **ENTER** key. When the statement

RUN@

is placed in F1, the program will be initiated merely by pressing the F1 function key.

## Using Function Key Menus

A menu display for the RESERVE mode may be set up by first placing the RESERVE mode in I, II, or III and typing the desired character string in quotation marks and then by pressing **ENTER**.

Example:

> "RUN TIME NPV LLIST"

Once the menu has been set up and the PC-1500A is in either RUN or PROGRAM mode, the menu may be seen by pressing **RCL**.

## **STATUS (Check Memory Status)**

The STATUS command on the PC-1500A is used to find the status of certain memory parameters. The argument for STATUS can be any number between 0 and 255.

- |          |   |
|----------|---|
| 0        | returns the number of bytes remaining in memory                                   |
| 1        | returns the number of bytes used by the program                                   |
| 2        | returns the address + 1 of the location where the program ends                    |
| 3        | returns the address of the memory location where the current variables are stored |
| 4 to 255 | returns the line number last executed by the program before stopping              |

The STATUS command may be used in RUN, PROGRAM, or RESERVE mode.

## **MEM (memory)**

The MEM statement will return the amount of free memory available, the same as STATUS 0.

## **STR\$ (Number to String Conversion)**

STR\$ (numeric expression)

The STR\$ function will convert any numeric expression to a character string.

EXAMPLE:

```
10: INPUT "ENTER 3 DIGIT NUMBER", A
20: A$ = STR$(A)
30: B$ = MID$(A$, 2, 1)
40: PRINT "MIDDLE DIGIT = "; B$
50: GOTO 10
```

## **VAL (String to Number Conversion)**

VAL (string)

The VAL function is the reverse of the STR\$ function. It converts character strings to numeric expressions. It is important to remember that the character string must be made up of acceptable numeric characters.

EXAMPLE:

**VAL(" - 1.212")**

**ENTER**

- 1.212

10: INPUT "NUMERIC STRING", C\$

20: C = VAL(C\$)

30: PRINT "NUMBER = "; C







## SECTION 6: PROGRAM EDITING

There may be times when you wish to change or modify a program: since programs are not always perfect, this happens quite often. By following the steps listed in this section, program editing will quickly become second nature.

Discussed in this section are:


- Editing Program Statements
- Deleting Program Statements
- Adding Program Statements
- Listing Program Statements
- Interrupting Program Execution
- Error Messages

### Editing Program Statements




To bring the first program line into the display, simply place the PC-1500A in the PROGRAM mode using the **MODE** key and the  key. Each time the  key is pressed, the next program step will be placed in the display. To scroll backward through the program, use the  key. To view the last line in the program, use the  key immediately after placing the PC-1500A in the PROGRAM mode. Pressing one of the cursor control keys will activate the cursor in the display. By using the display editing techniques discussed in section one, the program line can be changed or corrected. When the edited line is entered, it will replace the old line in the program. The following program listing will be used in the examples within this section.

# LISTING


```
10 I = 0
20 A = 0
30 I = I + 1
40 INPUT "X = ", X
50 B = (A + X) / I
60 PRINT B
70 GOTO 30
```

The program listing shown above is a simple program for calculating a moving average. When the program is in memory and you wish to edit one or more of the program statements, simply place the PC-1500A in the PROGRAM mode and press ; the display will appear as:

10:I = 0

By placing the cursor over the character in question using the cursor controls  and , the line may then be edited using display editing techniques discussed in section one. After the line has been corrected, pressing **ENTER** will put the statement into the program. By pressing , the next statement will appear.

20:A = 0

By pressing , the preceding line number will be displayed.

10:I = 0

Do not sale this PDF !!!

## Adding Statements

To add a statement or line to a program, simply type in the new line number, the statement, and then press **ENTER**. The new line will be inserted at the proper point.

EXAMPLE: Using the same listing as before, you wish to have the program display the number counter. To do this, enter the statement

**45 PRINT I**

The listing will now look like the next listing.

### LISTING

```
10 I=0
20 A=0
30 I=I+1
40 INPUT "X= ".X
45 PRINT I
50 B=(A+X)/I
60 PRINT B
70 GOTO 30
```

## Deleting Statements

To delete a statement, place the statement number into the display and press **ENTER**. The statement will then be deleted from the program.

## LIST (Listing a Program)

The lines within a program may be listed in three different ways using the LIST command (the LIST command may only be used when the computer is in the PROGRAM mode). The first is to use the LIST command alone; the second is to use it with a program statement number; and the last is to use it with a label name.

LIST  
LIST statement number  
LIST label

Using LIST by itself will place the first line of the program into the display.


## Interrupting Program Execution

The STOP statement causes the computer to suspend the execution of a program. When the program stops, the values of all variables are retained and the programmer may inspect and change these. The program may then be continued, at the point where it was halted, with the CONT command.



When the STOP statement is encountered by the computer, a message similar to the following is displayed:

BREAK IN 60

where 60 is the number of the line which contains the STOP statement. To review this line, depress and hold the Up Arrow  key.

When the prompt BREAK appears in the display, you may also review and change the values of variables. When you are ready to resume execution of the program, type in **CONT** and press the **ENTER** key.

## Error Messages

Although abbreviations and colons help you to easily enter programs, they cannot prevent the best of us from making mistakes. Even professional programmers fail to catch errors when reviewing their own programs. What this means is that sooner or later you will encounter an error while running your programs. Most of these errors are easily corrected if you simply accept them as a puzzle to be solved. Several features of the PC-1500A will assist you in tracking down the problem.





Upon encountering an incorrect statement, the PC-1500A will halt operation and indicate the problem with a terse message in the display such as:

ERROR 1 IN 20

which means ERROR 1 has occurred on line 20 of the program listing. ERROR 1 is an error code which tells you that you have a syntax error. In other words, you have incorrectly performed a calculation, have a typing error causing missing information, or have an invalid command. A complete list of error messages is included as Appendix F.

For purposes of illustration, the following program will deliberately cause an error.

```
25 PUASE "HUMPTY DUMPTY"  
50 PRINT "WAS AN EGGHEAD"
```

Upon running the program, an error message will appear on the display. As you depress (and as long as you hold) the Up Arrow  key, the display will show the line on which the PC-1500A became confused. The flashing grid should provide a hint as to the nature of the problem. To correct the statement, press the Clear All  key to quit the program and switch to the PROGRAM mode. Next, press (but do not hold) the Up Arrow  key and the display will again show the erroneous line. You may now proceed to edit the line using the Insert, Delete, and Arrow keys. After completing all editing, press the  key to store the corrected line in the PC-1500A.

## SECTION 7: BRANCHES AND LOOPS

PC-1500A BASIC provides the capability to construct very flexible and powerful programs through the use of control statements. The control statements used are IF... THEN, GOTO, GOSUB, and the FOR... NEXT Loop. Control statements direct the logic flow of the program. Certain statements may or may not be executed depending on the placement of control statements. The following sections describe the control statements of PC-1500A BASIC:

- IF... THEN
- GOTO
- GOSUB
- FOR... NEXT

### IF... THEN

The IF... THEN control statement provides the BASIC program with decision making capability. The conditional part of the control statement, IF, tests the expression for validity. The expression will evaluate to be either true or false. When the expression is true, the THEN statement is executed. Otherwise, the program continues to the next executable instruction. In general, we might express it thusly: IF condition THEN statement.

Let us consider the following sample program.

```
10 PAUSE "ARE YOU OVER 30 YEARS OLD"  
  
20 INPUT "TYPE YES OR NO": X$  
  
30 IF X$ = "YES" THEN PRINT "WELL, IT IS ALL DOWNHILL NOW"  
  
40 IF X$ = "NO" THEN PRINT "YOU ARE STILL JUST A KID"  
  
99 END
```

The PC-1500A first asks the question concerning the age of the user. The user responds by typing either **YES** or **NO**. The expression, X\$, in line 20 is called a character string. When the user responds with a **YES** answer, statement 30 is executed. The IF test for the character string is evaluated to be true. The PC-1500A sets the value of the expression to be positive. The next statement at line 40 is evaluated once again by using the IF test. This time the expression is false. The PC-1500A sets the value of the expression to be non-positive and the statement at line 40 is suppressed. Similarly, if the user had responded with a **NO** answer, line 40 would be executed and line 30 suppressed.

The IF... THEN control statement can also be applied to numbers. Consider the following example:

```
10 INPUT "ENTER A NUMBER" : X  
  
20 IF X + 5 < 0 THEN PRINT "X + 5 IS NEGATIVE"  
  
30 IF X + 5 = 0 THEN PRINT "X + 5 IS ZERO"  
  
40 IF X + 5 > 0 THEN PRINT "X + 5 IS POSITIVE"  
  
99 END
```

The PC-1500A user is prompted for a number. When the number is entered, the expression is evaluated by the computer. If the expression evaluates to be negative, line 20 is printed. When the expression equals zero, line 30 is printed. Correspondingly, when the expression evaluates to be positive, line 40 will be printed. As an example, suppose the PC-1500A user enters a  $-4$  for the value of the number. The PC-1500A evaluates the expression  $(-4 + 5)$  to be a positive 1. The value of 1 is greater than zero and line 40 is printed.

## GOTO

The GOTO control statement provides for transfer of control within a BASIC program. The logic flow of the program can handle all of the necessary paths by using a GOTO statement. The GOTO statement has several different forms. It may perform a "jump" from one statement in a program to another statement located elsewhere

in the program. The GOTO can also act as a looping mechanism. A group of statements may be executed repeatedly by using a GOTO statement. The general form of the GOTO control statement is:

GOTO (statement number)

where            statement number is a valid program line number (it exists in the program and is an integer from 1 to 65279)

WARNING:      If the program causes control to pass to a nonexistent line number, an ERROR 11 will occur.

An example of using the GOTO as a "jump" to another line in the program is as follows:

```
10  INPUT "BEGINNING BALANCE?" , BAL
20  INPUT "DEP OR WITH?" , ANS$
30  IF ANS$ = "DEP" THEN GOTO 60
40  IF ANS$ = "WITH" THEN GOTO 110
50  GOTO 20
60  INPUT "DEPOSIT" , DEP
```

```
65  BAL = BAL + DEP

70  PAUSE "NEW BALANCE:" , BAL

80  INPUT "MORE TRANSACTIONS?" , X$

90  IF X$ = "YES" THEN GOTO 20

100 GOTO 999

110 INPUT "WITHDRAWAL" , WITH

120 BAL = BAL - WITH

130 PAUSE "NEW BALANCE:" , BAL

140 INPUT "MORE TRANSACTIONS?" , X$

150 IF X$ = "YES" THEN GOTO 20

999 END
```

This program allows the PC-1500A user to keep track of his savings account. Start by entering a beginning balance. Deposits are added to the balance, while withdrawals are subtracted from the balance. The GOTO statement provides the PC-1500A user with a means of handling two different transactions within the same program. If a deposit is entered, the program goes to line 60. If a withdrawal is entered, the program goes to line 110. Line 80 and line 140 allow the PC-1500A user to enter multiple transactions at one time.

The GOTO control statement may also be used for repeated execution of one or more statements. This type of action is known as looping. The looping convention allows the PC-1500A user to perform such mundane tasks (e.g., summing numbers from 1 to 1000) by executing a group of statements repeatedly and then transferring control to the start of the loop. As a precaution when using the GOTO, avoid getting trapped into a loop with no end. Every loop should have a condition that will cause transfer of control outside of the loop. This will prevent execution of a loop that will never stop. An example of such an "endless loop" appears below:

```
10  LET X = 1  
  
20  IF X > 0 THEN PRINT "POSITIVE"  
  
30  GOTO 10  
  
99  END
```

This loop will never end and the only way to stop the program will be to hit the BREAK key.



The GOTO statement can be used as a command. The PC-1500A computer must be set to RUN mode. However, unlike the RUN command, GOTO does not clear all the variables before beginning execution. It merely starts at the specific line number and executes the program. An example of the GOTO command is:

GOTO statement number

where statement number is the first line number of the program to be executed and is an integer from 1 to 65279. As noted before, ERROR 11 occurs when the statement number does not exist.

## GOSUB

The PC-1500A user may find a need to perform repeated statements in a program. To enhance the PC-1500A user's programming capability and maximize the utilization of space, all repeated statements are organized in one part of the program. These repeated statements are called a subroutine. This part of the program, the subroutine, may be executed using the GOSUB statement.

A GOSUB statement is similar to a GOTO. The logic flow of the program is transferred to the line specified by GOSUB. The major difference between GOSUB and GOTO is that after the subroutine is executed, the logic flow is returned to the next statement following the calling "GOSUB". As an example, consider the following skeletal program and note the difference between GOTO and GOSUB.

1000 GOTO 4000	1000 GOSUB 4000
	1100 PRINT "GOODBYE"
4000 PRINT "HELLO"	4000 PRINT "HELLO"
.	4100 PRINT "HOW ARE"
.	4200 PRINT "YOU"
.	4300 PRINT "SO LONG"
	4400 RETURN
9999 END	9999 END

The GOSUB statement has this general format:

GOSUB line number

where the line number marks the first statement in the subroutine to be executed. To terminate a subroutine, the RETURN statement is used. The RETURN statement has this format:

line number RETURN

An example in using a subroutine is the following program that calculates student test scores. All test scores are entered. The program will find the class average and also the highest test score. The class average and highest test score are calculated in the subroutine that starts at line 600. This subroutine is located inside a FOR... NEXT loop. The FOR... NEXT loop will be discussed in the next section.

## CLASS AVERAGE PROGRAM

```
100 LET H=0
105 LET T1=0
110 LET T=0
120 INPUT "ENTER NUMBER OF STUDENTS", N
130 FOR I = 1 TO N
140 INPUT "ENTER STUDENT GRADES", X
150 IF I = 1 THEN LET H=X
160 GOSUB 600
170 NEXT I
200 LPRINT "HIGHEST GRADE:", H
210 LPRINT
220 LPRINT "CLASS AVERAGE:", T/N
230 END

600 REM SUBROUTINE TO CALCULATE
610 REM CLASS AVERAGE
620 LET T=T+X
630 T1=X
640 IF T1>H THEN LET H=T1
650 RETURN
```

It is good programming practice to place subroutines at the close of the program after an END statement. Subroutines can be grouped together in sections called "modules." This style of programming makes it easier to solve or "debug" larger programs. Remember, all subroutines must end with a RETURN statement.

## FOR... NEXT Loops

The FOR... NEXT statement is the main looping convention used by the PC-1500A. The FOR... NEXT statement may be used to perform such tasks as repeated calculations, periodic events, and in filling arrays. The general format of the FOR... NEXT statement is:

FOR count-variable = start-value TO end-value STEP increment

where

- |                |   |   |
|----------------|---|---|
| count-variable | = | Variable assigned for the count loop.   |
| start-value    | = | Beginning value for the count-variable the first time through the loop. The possible values are - 32768 to 32767.   |
| end-value      | = | Final value used in testing the count-variable in order to terminate the loop. The possible values are - 32768 to 32767.  |
| STEP increment | = | This clause may be omitted. If omitted, the default step size is 1. The increment may be positive or negative, with the possible values from - 32768 to 32767. The increment counts each time through the loop, updating the count-variable with each pass. |

Do not sale this PDF !!!

The following example illustrates the FOR... NEXT loop and how it is executed. Examine the value of I at the end of the loop.

```
10  FOR I = 1 TO 9
20  PAUSE I;
30  NEXT I
35  PAUSE
40  PAUSE "END OF LOOP.";
50  PRINT " SINCE I=";I
60  END
```

The value of I is initially set equal to one. Next, the count-variable I is compared with the end-value of 9. If the value of I is greater than the end-value, then the loop is terminated. Otherwise the statements inside the loop are performed and I is incremented by the step size. The process is repeated until the loop is terminated.

The increment value used by the FOR... NEXT loop may be negative. In this case, we need to supply the STEP clause in our FOR... NEXT loop with a negative increment. An example of this is as follows:

```
10  LET Q = 10
20  FOR Q = Q/2 TO 1 STEP -1
30  PAUSE Q
40  NEXT Q
50  PAUSE "END OF LOOP.";
60  PRINT " SINCE Q=";Q
70  END
```

The count variable, Q, is assigned the value of 10. The FOR... NEXT loop start-value is 5. (Q/2) The step size or increment is a negative 1. The loop will begin at 5 and decrease to the end-value of 1. We have shown that the FOR... NEXT loop terminates when the end-value is reached, the step size can be positive or negative, and the start-value may be either a number or a variable.

Another important feature of the FOR... NEXT loop is that the loops may be nested, or located within other loops. This procedure is very useful when working with arrays. The example below demonstrates how a large array may be initialized to zero with just two FOR... NEXT loops.

```
10 DIM AR(10,3)
20 FOR I = 1 TO 10
30 FOR J = 1 TO 3
40 LET AR(I,J) = 0
50 PAUSE AR(I,J);
60 NEXT J
70 NEXT I
80 END
```

The nested loops will place a zero in locations (1,1) through (10,3). An important thing to remember when working with nested arrays is to always end the inner loop before the outer loop. A syntax error occurs when the outer loop is ended first. In our example, NEXT J must appear before NEXT I.

## SECTION 8: MORE BRANCHING

### The Computed GOSUB Statement

An easy way to handle multiple branching to subroutines is the computed GOSUB statement. The computed GOSUB behaves in a manner similar to the GOSUB statement. The main difference between the two statements is the computed GOSUB evaluates an expression in order to perform the multiple branching. The general form of the computed GOSUB is:

ON variable GOSUB line number1, line number 2, ... .

The value of "variable" determines which line number will be executed. If "variable" assumes a value greater than the total number of lines following GOSUB, an error condition will occur. For example, consider the following statement:

(X = 1)(X = 2)(X = 3)(X = 4)(X = 5)

ON X GOSUB 500, 600, 700, 800, 900

The variable used in the example is "X". The variable "X" must be an integer with the value of 1, 2, 3, 4, or 5. If "X" is not one of these values, an error condition occurs.

In order to demonstrate the time and effort saved by using the computed GOSUB statement, observe programs A and B in Figure 8.1.

Program A

Program B

10 PAUSE "INDICATE WHICH OPERATION:"

Lines 10 – 40 are the same as Program A

15 PAUSE "ADDITION (+)"

20 PAUSE "SUBTRACTION (-)"

25 PAUSE "MULTIPLICATION (\*)"

30 PAUSE "DIVISION (/)"

35 INPUT "EXPONENTIATION (^)":X

40 IF (X<0) OR (X>5) THEN GOTO 10

45 IF X=1 THEN GOSUB 100

45 ON X GOSUB 100,200,300,400,500

50 IF X=2 THEN GOSUB 200

55 IF X=3 THEN GOSUB 300

60 IF X=4 THEN GOSUB 400



140

65 IF X=5 THEN GOSUB 500

.  
.  
.

#### PROGRAM A SUBROUTINES

100 REM ADDITION SUBROUTINE

.  
.  
.

199 RETURN

200 REM SUBTRACTION SUBROUTINE

.  
.  
.

299 RETURN

300 REM MULTIPLICATION SUBROUTINE

.  
.  
.

399 RETURN

#### PROGRAM B SUBROUTINES

Lines 100 — 599 are the same as Program A.

```
400 REM DIVISION SUBROUTINE
```

```
.  
. .  
. .  
. .
```

```
499 RETURN
```

```
500 REM EXPONENTIATION SUBROUTINE
```

```
.  
. .  
. .  
. .
```

```
599 RETURN
```

Figure 8.1.

The computed GOSUB statement in line 45 of Program B replaced lines 50 — 65. This statement saved four additional lines of code. This could be extremely important for large programs. The computed GOSUB statement substitutes for multiple IF — THEN statements when subroutines are used.

## **ON ... GOTO**

Similarly, the ON GOTO statement provides another control statement of great utility. This statement acts like the GOTO statement discussed previously. The difference, however, lies in its ability to transfer control (i.e., to

execute statements at a different location) automatically. Thus, the GOTO function will "go to" one of several statements depending on the value of a numeric variable. The general form of the ON GOTO statement is:

ON variable GOTO line number1, line number2, ... .

A typical ON GOTO statement might be:

ON X GOTO 100,200,300,400

The variable "X" must be an integer with the value of 1, 2, 3, or 4 because there are only four line numbers listed; any other number will result in an error since there is no corresponding line number.

#### **ON ERROR ... GOTO**

The ON ERROR GOTO statement is another form of controlled transfer which allows a program to detect when an error occurs. Upon detection, the program may execute statements which attempt to recover from the error. These statements usually inform and instruct the user and most likely will save valuable data. The ON ERROR GOTO statement instructs the PC-1500A where to go after detecting the occurrence of an error. The general form of the ON ERROR GOTO statement is:

ON ERROR GOTO line number

where line number is the number of a program line containing instructions to be followed in the event of an error.

## Branching Using INKEY\$

The INKEY\$ function accepts any character from the keyboard and places it in a variable location. The character is accepted immediately, thus eliminating the need to press the **ENTER** key. The general format of the INKEY\$ function is:

variable = INKEY\$

The INKEY\$ function does not show the input character on the screen. An example of using the INKEY\$ function is shown in Figure 8.2.

```
10 WAIT 0
20 X$ = ""
30 X$ = INKEY$
40 IF X$ = "" THEN GOTO 30
50 FOR I = 1 TO 26
60 PRINT X$:
70 NEXT I
80 GOTO 20
90 END
```

Figure 8.2.

## Branching Using Timers

The TIME function can be set or displayed. In order to set the time, the following information must be entered:

**TIME = MMDDHH.MMSS ENTER**

where

- MM — Month of the current year
- DD — Day of the current year
- HH — Hour of the current day
- . — Fractional part of the hour
- (MM) — Minutes of the current hour
- (SS) — Seconds of the current minute

For example, if the current date is November 30, 1982 and the time is 1:30 P.M., the TIME function would be set as follows:

MMDDHH.MMSS

TIME = 113013.3000

NOTE: All afternoon times will be given in military format. For example, 1:00 P.M. is 1300.

To display time, simply type **TIME** and press **ENTER**. The current time will be printed and displayed in the right hand corner of the screen.

113013.3100

The example program performs a calendar and clock simulation. When the program is run, the current date appears on the left hand side of the display and the time on the right hand side. The time display is updated every second. Remember to set the time before executing the example program. Observe the branching in lines 30 and 140 taken by the TIME function. The output of the program will be as follows:

11/30/1983

13.3101

## Example Program

```
10  WAIT 0                                100 PRINT D$
20  X$ = STR$ TIME                        110 T = TIME - D
30  IF TIME > 99999 THEN 50               120 IF T >= 1 THEN 140
40  X$ = "0" + X$                         130 T = T + 12
50  A$ = LEFT$(X$,2)                     140 IF T > 23.5959 GOTO 20
60  B$ = MID$(X$,3,2)                    150 CURSOR 15
70  C$ = MID$(X$,5,2)                    160 PRINT T
80  D$ = A$ + "/" + B$ + "/" + "1983"    170 GOTO 110
90  D = VAL(A$ + B$ + "00")
```

## SECTION 9: USING VARIABLES: STRINGS AND ARRAYS

This section of the manual discusses the use of variables with special emphasis on string variables and the use of arrays. The use of variables and arrays is what gives the BASIC programming language much of its power and versatility. The use of strings allows the PC-1500A to communicate with you in English.

Discussed in this section are:

- Arrays
- Strings
- Substrings
- String Functions
- READ and DATA Statements
- RESTORE Statements
- Define Key

### Array Concepts

Up to this point, the programs considered have had a small number of variables. To take advantage of the full processing potential of the PC-1500A, arrays describing a large number of variables may be used. An array is a much simpler method of handling variables than assigning a different label to each variable.

An array is simply a group of consecutive storage areas, or locations, with a single name. Each storage area can hold a single number or character string. All the storage areas within an array must hold the same type of data (numeric or character strings).



The number of locations within a single array can be as many as 256. When using a string array, you may also specify the length of the string with a maximum length of 80 characters.

The PC-1500A will let you go further than simple arrays by letting you create two-dimensional arrays made of rows and columns. The PC-1500A can have up to three dimensions in defined arrays. Examples of arrays are as follows.

Simple (One-Dimensional) Array:

A(1)  
A(2)  
A(3)  
A(4)  
A(5)

Two-Dimensional Array:

A(1,1)	A(1,2)	A(1,3)	A(1,4)
A(2,1)	A(2,2)	A(2,3)	A(2,4)
A(3,1)	A(3,2)	A(3,3)	A(3,4)
A(4,1)	A(4,2)	A(4,3)	A(4,4)
A(5,1)	A(5,2)	A(5,3)	A(5,4)

## DIM (Dimensioning Variables)

Before an array can be used, it must be defined using a dimension (DIM) statement. A one-dimensional numeric array is defined using the following form.

DIM variable name (size)

A two-dimensional array uses the form:

DIM variable name (rows, columns)

The elements within a numeric array may be assigned separately as if assigning a value to any variable (A(2) = 7) or by using the subscripts as in the following listing.

```
10: DIM SQ(9)
20: FOR I=0 TO 9
30: SQ(I)=I*I
40: NEXT I
```

To define a string array, the following forms are used.

DIM variable name (size) \* length

DIM variable name (row, column) \* length

The length parameter is optional. It has a range of from 1 to 80 characters and, if omitted, has a default value of 16.

## String Expressions

### LEN (Length)

LEN "character string"  
LEN string variable name

When manipulating character strings, it is sometimes helpful to know the number of characters in the string. This is accomplished using the LEN statement. An example follows.

```
10: A$ = "GOOD MORNING"  
20: A = LEN A$  
30: PRINT A
```

The display prompt would show 12.

There are three functions that can be used to extract specific portions of a character string. LEFT\$ extracts characters from the left, RIGHT\$ from the right, and MID\$ from the middle.

## LEFT\$

LEFT\$ "character string", number  
LEFT\$ string variable name, number

The LEFT\$ instruction takes a specified number of characters from the left of a string. The number argument specifies how many characters to extract from the left side of the string. An example follows.

```
10: A$ = LEFT$("DRESSER",5)  
20: PRINT A$
```

The display prompt would show:

DRESS

A second example follows.

```
10: B$ = "THINK BIG"  
20: A$ = LEFT$(B$,4)  
30: PRINT A$
```

The display prompt now shows:

THIN

Do not sale this PDF !!!

**MID\$**

The MID\$ statement is used to extract the middle portion of a character string.

MID\$ ("character string", expression, expression)  
MID\$ (string variable name, expression, expression)

An example follows:

```
10: A$ = "I NEED HELP"  
20: B$ = MID$(A$,3,4)  
30: PRINT B$
```

The display prompt would be:

**NEED**

The first numeric expression represents the first character to be extracted. The second numeric expression tells how many characters to extract.

## RIGHT\$

The RIGHT\$ statement works as the LEFT\$ statement.

RIGHT\$ ("character string", number)

RIGHT\$ (string variable, number)

The number portion of the argument specifies how many characters from the right are to be extracted. An example is as follows.

```
10: X$ = "READ ONLY MEMORY"
```

```
20: Y$ = RIGHT$(X$,6)
```

```
30: PRINT Y$
```

The display prompt would show:

MEMORY

## String Functions

### ASC (ASCII to Decimal Conversion)

ASC "character"  
ASC string variable name

The ASC function is used to convert a single character into ASCII decimal code. The argument of the function is any character string or string variable. The value returned by the function is the ASCII code for the first character in the string.

#### EXAMPLE 1:

```
10: PRINT ASC "W"
```

87

#### EXAMPLE 2:

```
10: LET A$ = "12 APPLES"  
20: Z = ASC A$  
30: PRINT Z
```

49

A list of ASCII characters is provided in Table 9.1.

### **CHR\$ (Conversion ASCII Decimal Code to Characters)**

CHR\$ number

CHR\$ variable

The CHR\$ function is the complement to the ASC function. By using the ASCII decimal code (a number from 0 to 127) an ASCII character will be printed. (NOTE: Some codes represent special characters that do not print.) The following sample program can be used to review the ASCII characters.

#### **EXAMPLE:**

```
10:  FOR I=0 TO 127
20:  PAUSE STR$(I)CHR$ I
30:  NEXT I
40:  END
```



Table 9.1. ASCII Character Code Chart

Upper Bit Positions → b7, b6, b5									
		000	001	010	011	100	101	110	111
Low Bit Positions b4, b3, b2, b1 ↓	Hexa- decimal	0	1	2	3	4	5	6	7
	0000	0		SPACE	0	@	P	E	p
	0001	1		!	1	A	Q	a	q
	0010	2		"	2	B	R	b	r
	0011	3		#	3	C	S	c	s
	0100	4		\$	4	D	T	d	t
	0101	5		%	5	E	U	e	u
	0110	6		&	6	F	V	f	v
	0111	7		□	7	G	W	g	w
	1000	8		(	8	H	X	h	x
	1001	9		)	9	I	Y	i	y
	1010	A		*	:	J	Z	j	z
	1011	B		+	;	K	√	k	{

1100	C			.	<	L	¥	l	!
1101	D			-	=	M	π	m	}
1110	E			.	>	N	^	n	~
1111	F			/	?	O	—	o	■

## String Variables

### String Concatenation

Character strings may be joined together to form new strings by simply adding the strings as in the example.

EXAMPLE:

```
10: S$ = "SUPER"
20: T$ = S$ + "MAN"
30: PRINT T$
```

SUPERMAN

## String Comparisons

Character strings may be compared using many of the same logical operators used in numeric comparisons. The legitimate operators are:

=	True if the two strings are equal in length and contain the same characters in the same order.
<>	True if the two strings differ in length, characters, or ordering of the characters.
>	True if the characters of the first string occur later in ordering than the characters in the second string.
<	True if the characters of the first string occur first in the ordering than the characters in the second string.

NOTE: Forms such as (A\$<B\$) OR (A\$=B\$) are also allowed.

## DATA

DATA item, item, item

DATA statements are another way of entering data into a program. DATA statements begin with the keyword

DATA followed by the data items separated by commas. The data items may be numeric or character strings.

**EXAMPLE:**

10: DATA "GEORGE WASHINGTON", 12, 1776, "S", 4E23

**READ**

READ variable name, variable name, variable name

The READ statement corresponds to the DATA statement; for every DATA statement there must be a READ statement. The READ statement consists of the keyword READ followed by the variables that will be assigned the data separated by commas. The READ statement that corresponds to the DATA statement example is in example 1.

**EXAMPLE 1:**

READ A\$, B, C, S\$, Z

**EXAMPLE 2:**

10: DATA 1, "A", 1  
20: DATA 2, "B", 3  
30: DATA 4, "C", 5  
40: DATA 10, "G", 12  
50: FOR I = 1 TO 4  
60: READ A, A\$, B  
70: X = A + B  
80: PRINT A\$, X  
90: NEXT I  
100: END

**RESTORE**

RESTORE line number

The RESTORE statement is a means of recycling DATA statements. By using the keyword RESTORE and a DATA statement line number or label, the READ statement will begin the data over again beginning at the line number specified. Data will be read in from the first DATA statement by using the keyword RESTORE without a line number.

**EXAMPLE:**

```

10: DATA 7.3
20: DATA 8.5
30: READ S, D
40: READ E, W
50: RESTORE 30
60: READ N,B

```

**NOTE:** When two or more programs are stored in the computer by a MERGE statement, the RESTORE statement in each of the second and subsequent programs must be specified in the form of RESTORE expression (or "label").

When a label in another program is referred to by using "RESTORE label", the RESTORE "label" statement must be followed by GOTO statement to branch to a line with a larger line number.

**Example**

<pre> 10 "A": DATA . . . . . . . </pre>	} Program A	<pre> 10 "B": RESTORE "A": <u>GOTO 20</u> 20 READ X 30 IF X = 0 THEN 20 . . . </pre>	} Program B
---	-------------	--	-------------

Do not sale this PDF !!!

## INKEY\$

The INKEY\$ function takes any single character from the keyboard and stores it in a specified variable. When this function is used, the character is automatically accepted from the keyboard; the **ENTER** key need not be pressed nor is the character printed in the display.

### EXAMPLE:

```
10: WAIT 0
20: A$ = "": A$ = INKEY$
30: IF A$ = "" THEN PRINT "NULL STRING": GOTO 20
40: PRINT A$
50: GOTO 20
```

Good programming practice suggests storing the INKEY\$ function in a variable, then manipulating the variable.

### EXAMPLE:

```
1 WAIT 0
5 FOR X = 1 TO 10: NEXT X
10 IF INKEY$ = "" THEN 10
19 WAIT
20 PRINT INKEY$
30 GOTO 10
```

## DEF Key

### Running DEFInable Programs

The **DEF** key is a method of beginning program execution of a labeled program. By pressing the **DEF** key and one of the following allowable label keys:

**A**, **S**, **D**, **F**, **G**, **H**, **J**, **K**, **L**, **Z**,  
**X**, **C**, **V**, **B**, **N**, **M**, **SPACE**, and **=**

the program will start automatically.

### Preassigned Keyword Keys

A few of the more frequently used keywords have been permanently assigned to a single alphabetic key in the second row of the keyboard. The keywords may be called by pressing the **DEF** key followed by the alphabetic key.

EXAMPLE:

**DEF** **E**

The keywords and their keys are shown below.

<b>Q</b>	INPUT	<b>Y</b>	RETURN
<b>W</b>	PRINT	<b>U</b>	CSAVE
<b>E</b>	USING	<b>I</b>	CLOAD
<b>R</b>	GOTO	<b>O</b>	MERGE
<b>T</b>	GOSUB	<b>P</b>	LIST

### The AREAD Statement

AREAD variable name

The AREAD statement is another method for entering data into a program. By typing the value of a variable and pressing the **DEF** key, the value will be assigned to the variable in the AREAD statement.

#### LISTING

```
10: "X": AREAD TM
20: TIME = TM
30: PRINT "TIME SET TO "; TM
40: END
50: "Z": AREAD D$
60: PRINT "TODAY IS ";D$
70: END
```



EXAMPLE 1:

1130.093045    **ENTER**  
**DEF**    **X**

EXAMPLE 2:

"THURSDAY"    **ENTER**  
**DEF**    **Z**

### Automatic Program Initiation (ARUN)

When ARUN is the first statement in the program, the program will begin running as soon as the computer is turned on.

EXAMPLE:

```
10: ARUN
20: FOR I=1 TO 5
30: PAUSE "HI THERE": BEEP 5
40: CLS
50: NEXT I
60: END
```

## SECTION 10: DISPLAY PROGRAMMING

The PC-1500A display window can be used in a variety of different ways. Each display window shows a maximum of 26 characters. A character is defined by a  $5 \times 7$  dot matrix. The user may invoke the GPRINT command to create and display new characters. The display window may be considered as a  $7 \times 156$  dot matrix. This feature allows the user to develop graphics, figures, and other special symbols. Sound is controlled by use of the speaker and tone generator. The BEEP statement determines the number of times the tone is repeated, the tone frequency, and the tone duration. The POINT command checks to determine which dots of the matrix are currently in use.

### BEEP (Programming Tones)

The BEEP statement creates tones used for error handling, games, and some interactive programs. The general format of the BEEP statement is:

BEEP value1, [ value2 ], [ value3 ]

where

- |            |   |   |
|------------|---|---|
| value1     | — | A required parameter which determines the number of times the tone is repeated. The range is from 0 to 65535. |
| [ value2 ] | — | An optional parameter that determines the frequency of the tone. The range is from 0 to 255.                  |

[ value3 ] — An optional parameter that determines the duration of the tone. The range is from 0 to 65279.

To turn off the internal speaker, use:

BEEP OFF

Conversely, to turn on the internal speaker, use:

BEEP ON

Every time the PC-1500A is turned off, the internal speaker is set to active mode.

The example program below plays music by varying the repetitions and frequencies of the BEEP statement.

```
10 D = 50
20 DATA 14
30 DATA 240,1,240,1,180,1,180,1
40 DATA 150,1,150,1,175,4
50 DATA 180,1,180,1,200,2,200,2
60 DATA 250,1,240,1,230,2
70 READ X
80 FOR I = 1 TO X
```

```
90 READ M,Q
100 BEEP 2,M,(D*Q)
110 NEXT I
120 END
```

## Display Functions

### CLS (Clear Screen)

The CLEAR SCREEN function erases the contents of the entire display. All of the dot locations are turned off. The cursor will be returned to character position 0. The general format of the CLEAR SCREEN function is:

CLS

The following example uses the CLEAR SCREEN function to erase the display after each string is printed. The first string "ABC" will appear for about 4 seconds. The screen will be cleared and "DEF" will be displayed for 2 seconds. Finally, the screen will be cleared and "GHI" will appear for 1 second. This loop is repeated for three times.

```
10 A$ = "ABC"
20 B$ = "DEF"
30 C$ = "GHI"
40 FOR I = 1 TO 3
50 PAUSE A$
60 WAIT 256
```

```
70 CLS
80 PAUSE B$
90 WAIT 128
100 CLS
110 PAUSE C$
120 WAIT 64
130 CLS
140 NEXT I
150 END
```

### **CURSOR Statement**

The CURSOR statement places the cursor in one of the 26 character positions on the screen. The character position may have a value of from 0 to 25. Any value outside of this range, (0-25), results in ERROR 19. The general format of the CURSOR statement is:

CURSOR character position

The main function of the CURSOR statement is to allow the user to print output at any character position on the screen. The CURSOR statement helps the user to design programs that are easier to read. The following example prints "ABC COMPANY" at character positions 0, 2, 4, and 8.

```
10 WAIT 30
20 PAUSE "A"
30 CURSOR 2
40 PAUSE "B"
50 CURSOR 4
60 PAUSE "C"
70 CURSOR 8
80 PAUSE "COMPANY"
90 END
```

The CURSOR statement may also be a variable. The following example prints "N", "NA", "NAM", and "NAME" at locations 4, 8, 12, and 16 on the display.

```
10 C = 0
20 FOR I = 1 TO 4
30 CLS
40 C = C + 4
50 CURSOR C
60 S$ = LEFT$("NAME",I)
70 PAUSE S$
80 WAIT 50
90 NEXT I
100 END
```

**GCURSOR Statement**

There are 156 columns of dots that appear on the display. Each column consists of 7 dots. The GCURSOR statement positions to one of the columns of dots as the starting point for the display. The general format of the GCURSOR statement is as follows:

			GCURSOR position
where			
	position	—	is an integer number between 0 and 155, specifying one of the 156 columns on the display. Any number outside of this range, less than zero or greater than 155, results in ERROR 19.

The GCURSOR statement is most commonly used to create graphic displays. The GCURSOR statement defines the column location where the information will be shown on the display. The following program prints "HELLO" beginning at column location 20, incrementing by 20 for each letter.

```

10  GCURSOR 20
20  PAUSE "H"
30  GCURSOR 40
40  PAUSE "E"
50  GCURSOR 60
60  PAUSE "L"
70  GCURSOR 80
80  PAUSE "L"

```

```
90  GCURSOR 100
100 PAUSE "O"
110  END
```

Another example using the GCURSOR statement appears below. In this example, the user is prompted for a message that is less than six characters. The characters are overlapped which gives a very unusual effect.

```
10  WAIT 0
20  DIM B$(0)*10
30  B$(0) = " " : X = 0
40  INPUT "ENTER MESSAGE (< 6 CHARS)".B$(0)
50  CLS
60  FOR I = 1 TO (LEN B$(0))
70  GCURSOR X
80  FOR J = 1 TO 5
90  GCURSOR X
100 PRINT MID$(B$(0),I,1)
110 X = X + 5
120 NEXT J
130 X = X + 5
140 NEXT I
150 WAIT
160 GCURSOR 155
170 GPRINT "00"
180  END
```



**GPRINT Statement**

The energized dots that comprise the display window may be controlled by using the GPRINT statement. The GPRINT statement allows the user to arrange the dots of the 7-dot column in any pattern that he may select. The pattern of energized dots may be defined as a decimal number or a hexadecimal character string.

**Decimal Number**

If we choose to define the pattern of energized dots as a decimal number, each row is numbered as a power of 2. The row begins with 1 (2 raised to the 0 power) and terminates with 64 (2 raised to the sixth power). An example of the numbering system appears below.

1	—	—	—	—	—	—	—	—
2	—	—	—	—	—	—	—	—
4	—	—	—	—	—	—	—	—
8	—	—	—	—	—	—	—	—
16	—	—	—	—	—	—	—	—
32	—	—	—	—	—	—	—	—
64	—	—	—	—	—	—	—	—

There are 128 energized dot patterns possible per column. The general format of the GPRINT statement is:

GPRINT pattern

where

pattern — evaluates to an integer number from 0 to 127 and determines a pattern of energized dots. The patterns are delimited by either a semicolon or a comma, a comma causing one blank column between each printed column.

To illustrate the GPRINT statement, we shall create a new character: the diamond. The grid layout of the diamond appears in Figure 10.1. There are seven columns needed to define the diamond. Columns 1 and 7 have a single dot in row 8. Columns 2 and 6 have one dot each in row 4 and row 16. Columns 3 and 5 have one dot each in row 2 and 32. Column 4 has one dot each in row 1 and in row 64. By adding each row number where a dot appears, the sum will define the contents of that particular column. As an example, columns 1 and 7 sum to 8. The number 8 will define columns 1 and 7 in the GPRINT statement. The final GPRINT statement is:

GPRINT 8:20:34:65:34:20:8

1	—	—	—	*	—	—	—
2	—	—	*	—	*	—	—
4	—	*	—	—	—	*	—
8	*	—	—	—	—	—	*
16	—	*	—	—	—	*	—
32	—	—	*	—	*	—	—
64	—	—	—	*	—	—	—

Figure 10.1

## Hexadecimal String

The seven rows of energized dots can be subdivided into two groups. The first four rows comprise the upper group and the last three rows comprise the lower group. As with decimal numbers, each row is numbered with a power of 2, beginning with 2 raised to the 0 power, or 1.



Figure 10.2.

The hexadecimal string requires two hexadecimal digits to define one column. The first digit corresponds with the lower group and the second digit corresponds to the upper group. The general form of the hexadecimal GPRINT statement is:

GPRINT "hex-string"

where

hex-string      —      Requires two hexadecimal digits to define the energized dot pattern of a single column.

The diamond in Figure 10.1 is defined by the following hexadecimal string:

GPRINT "08142241221408"

The first pair of hex digits is 08. The zero defines the lower group since no dots appear in the lower three rows. The eight defines the upper group because there is one dot in row 8, with no dots above that location. The other six pairs of hex characters may be defined in a similar fashion. A description of all the hexadecimal characters used by the hexadecimal string GPRINT statement appears in Table 10.1.

## **POINT Statement**

The POINT statement scans the pattern of energized dots within any given column and returns a number that the pattern of dots defines. The POINT statement will return a value from 0 to 127. The range of the value returned by the POINT statement is determined exactly like the GPRINT statement. All seven rows of a particular column

Table 10.1. Hexadecimal Characters.

1	_____	___*___	_____	___*___
2	_____	_____	___*___	___*___
4	_____	_____	_____	_____
8	_____	_____	_____	_____
	0	1	2	3
1	_____	___*___	_____	___*___
2	_____	_____	___*___	___*___
4	___*___	___*___	___*___	___*___
8	_____	_____	_____	_____
	4	5	6	7
1	_____	___*___	_____	___*___
2	_____	_____	___*___	___*___
4	_____	_____	_____	_____
8	___*___	___*___	___*___	___*___
	8	9	A	B
1	_____	___*___	_____	___*___
2	_____	_____	___*___	___*___
4	___*___	___*___	___*___	___*___
8	___*___	___*___	___*___	___*___
	C	D	E	F

are added together to determine the value. The general form of the POINT statement is:

POINT position

where

position — is a number between 0 and 155 and describes the column under investigation.

As an example, consider the dots required to display a capital Y in columns 30 through 36. The POINT statement will yield the following expressions:

1	*	—	—	—	—	—	*
2	—	*	—	—	—	*	—
4	—	—	*	—	*	—	—
8	—	—	—	*	—	—	—
16	—	—	—	*	—	—	—
32	—	—	—	*	—	—	—
64	—	—	—	*	—	—	—

POINT 30 = 1

POINT 34 = 4

POINT 31 = 2

POINT 35 = 2

POINT 32 = 4

POINT 36 = 1

POINT 33 = 120

## PRINT USING

The USING statement allows a programmer to rigidly control the format of information on the display. This allows standardized displays and prevents loss of information.

When the USING clause appears, alone or within a PRINT or PAUSE statement, it defines the format for all subsequent PRINT or PAUSE statements until the next USING clause is encountered in the program.

Several USING clauses may appear within a single PRINT or PAUSE statement. In this case, each one defines the format to be used to print the listed variables until the next USING clause is encountered.

A format is specified via a string of special characters called an "editing string." The characters within the editing string define the areas of the display available for information and restrict the type of information which may be printed in these areas. This scheme is the same general scheme employed by other languages such as COBOL and PL/I.

An editing string may be stored in a string variable. The variable's name would then replace the editing string within the USING clause. This allows multiple formats which are selected under program control.

The characters which may be employed within editing strings are summarized below:

Table 10.2. Table of Editing Characters.

Character	Use
#	Specifies a numeric field. Numbers are right-justified within this field. If the field width is insufficient to hold the number, an ERROR 36 will occur. Leading zeroes are converted to blanks.
*	Specifies Asterisk Fill of the specified positions of a numbering field which does not contain data.
.	Causes a decimal point to be displayed within a numeric field.
,	Used at the beginning of a numeric field to specify the insertion of commas after every three digits.
^	Used within a numeric field to cause the number to be displayed in scientific notation.
+	Used in a numeric field to force printing of the sign of the data.
&	Specifies a character field. Characters are left-justified within the field. If the field width is insufficient to hold the data string, the string is truncated.



**NOTE:** The width of a numeric field must always be one more than the width of the data to allow for the sign of the data. This is true regardless of whether you use the + editing character or not.

**NOTE:** The use of the comma requires that you insert one extra # for each comma in the editing string.

**EXAMPLES:**

X=PI Y=1234 A\$="ABCDEF"

**PRINT USING "###"; X**

3

**PRINT USING "+###.###";X**

+3. 141

**PRINT USING "###.## ^";X**

3. 14E 00

**PRINT USING “###. ^”;X**

3.E 00

**PRINT USING “\*#####”;Y**

\*\*1234

**PRINT USING “\*\*\*##”; Y**

1234

**PRINT USING “&&&&&&#####”; A\$; Y**

ABCDEF

1234

**PRINT USING "&&&"; A\$**

ABC

10 U\$ = "\*#####. ###"

20 USING U\$

30 PRINT Y; "\$"

\*\*1234. 00\$

PRINT X; "\$"

\*\*\*\*\*3. 14\$

PRINT USING; A\$; X

ABCDEF 3. 141592654

**PRINT USING "###, ###, ###"; #246813**

246, 813

**NOTE:** Use the number of # (including \*) marks for variable (integer) designation in the following range:  
With 3-digit punctuation (,) not used: within 11  
With 3-digit punctuation (,) used: within 14  
This computer has 10 significant digits for numbers.  
When the format exceeding 10 digits for the integer part is designated by the USING statement, numeric display and printouts exceeding 10 integers under the PRINT statement and LPRINT statement (for the printer) may be incorrect.

RUN mode		Display
<b>USING</b>	<b>"#####"</b> <b>ENTER</b>	→
<b>PRINT</b>	<b>888888888888</b> <b>ENTER</b>	→ 888888888800
<b>(LPRINT</b>	<b>888888888888</b> <b>ENTER</b>	→ 88888888880)

12 digits

## SECTION 11: DEBUGGING

No matter how careful you are, eventually you will create a program which does not do quite what you expect it to. In order to isolate the problem, Sharp's designers have provided a special method of executing programs known as the "Trace" mode. In the Trace mode, the PC-1500A will display the line number of each program line and will halt after the execution of that line. This allows you to follow (or trace) the sequence of instructions as they are actually performed. When the program pauses after the execution of a line, you may inspect or alter the values of variables.

The form of the instruction for initiating the Trace mode is simply: TRON. The TRON instruction may be issued as a command (in RUN mode) or it may be embedded, as a statement, within a program. Used as a command, TRON informs the PC-1500A that tracing is required during the execution of all subsequent programs. The programs to be traced are then started in a normal manner, with a GOTO or RUN command.

If TRON is used as a statement, it will initiate the Trace mode only when the line containing it is executed. If, for some reason, that line is never reached, the Trace mode will remain inactive.

Once initiated, the Trace mode of operation remains in effect until canceled by a TROFF instruction. The TROFF instruction may also be issued as either a command or a statement. The Trace mode can also be canceled by the key sequence:

**SHIFT**

**CL**

As an example in using the Trace mode, enter the following program to compute the length of the hypotenuse


of a triangle given the length of the sides:

Program Listing:


```
10 INPUT A, B
20 A = A*A : B = B*B
30 H =  $\sqrt{A + B}$ 
40 PRINT "HYPOTENUSE = ";H
```

In RUN mode, issue the TRON command, followed by the RUN command. Notice that the INPUT command operates in the usual manner by displaying a question mark for each input value required. As soon as you have entered two values, the line number of the INPUT statement appears:

10:	RUN	I	.
-----	-----	---	---


By pressing the  (Up Arrow) key and holding it, you may review the entire line:

10: INPUT A, B	RUN	I	.
----------------	-----	---	---

To continue the program, press the  (Down Arrow) key once. This causes the next line to be executed and its line number to be displayed. Again, you may review the line with the Up Arrow key. You may also check the contents of any variable by typing its name and pressing **ENTER** :

**A** **ENTER** (when A=4 is input before **A** **ENTER** operation)

RUN		I	.
			4

It is necessary to press the  (Down Arrow) key once for each line to be executed until the program ends. If you do not wish to continue normal line-by-line execution, press the **ENTER** key to suspend execution of the program. If you change your mind again, suspended programs may be continued with the CONT command.

A sample session, using our hypotenuse program, follows:

Keystrokes	Display
	>
<b>T</b> <b>R</b> <b>O</b> <b>N</b>	TRON_
<b>ENTER</b>	>
<b>R</b> <b>U</b> <b>N</b>	RUN_

ENTER	?
3	3_
ENTER	?
4	4_
ENTER	10:
↕	10:INPUT A, B_
↕	20:
↕	20:A = A*A : B = B*B_
A	A_
ENTER	9
B	B_
ENTER	16
↕	30:
H	H_
ENTER	5
↕	HYPOTENUSE = 5
↕	40:PRINT "HYPOTENUSE = ";H_
↕	40:
↕	>




## STOP, CONT

The STOP statement causes the computer to suspend the execution of a program. When the program stops, the values of all variables are retained and the programmer may inspect and change these. The program may then be continued, at the point where it was halted, with the CONT command.

When the STOP statement is encountered by the computer, a message similar to the following is displayed:

BREAK IN 60	RUN	I	.
-------------	-----	---	---

where 60 is the number of the line which contains the STOP statement.

If you wish to review this line, depress and hold the  (Up Arrow) key.

When the BREAK message appears, you may also review and change the values of variables. For example:

H    **ENTER**

	56. 23	.
--	--------	---

**AS**    **ENTER**

"DEDUCTIONS"

Whenever you are ready to resume execution, simply return to the prompt (>) and type **CONT** **ENTER**.

## LOCK, UNLOCK

The LOCK instruction may be used to control the mode (RUN, PROGRAM, or RESERVE) in which the computer operates. Included within a program, it prevents the user from accidentally changing the mode and injuring the program. The LOCK instruction disables the **MODE** key, "locking" the computer into whatever mode it is currently in.

To re-enable the **MODE** key, the UNLOCK instruction is used. UNLOCK restores the normal functioning, allowing changes in mode.

Either instruction may be used as a command or a statement. The forms are simply:

LOCK

UNLOCK

## **PART III: EXPANDING THE PC-1500A WITH THE PRINTER/CASSETTE INTERFACE CE-150**

The Printer/Cassette interface (CE-150) is an option for the Sharp PC-1500A pocket computer. This unit can be connected to one or two cassette tape recorders. The tape recorders can be used to store programs and data on standard audio cassettes. Programs can be "loaded" back into the PC-1500A for use at a later date, saving you the trouble of typing them again.

The use of these features will be discussed in the following sections. First, we would like to bring some operating precautions to your attention.

### **SECTION 12: INTRODUCING THE CE-150**

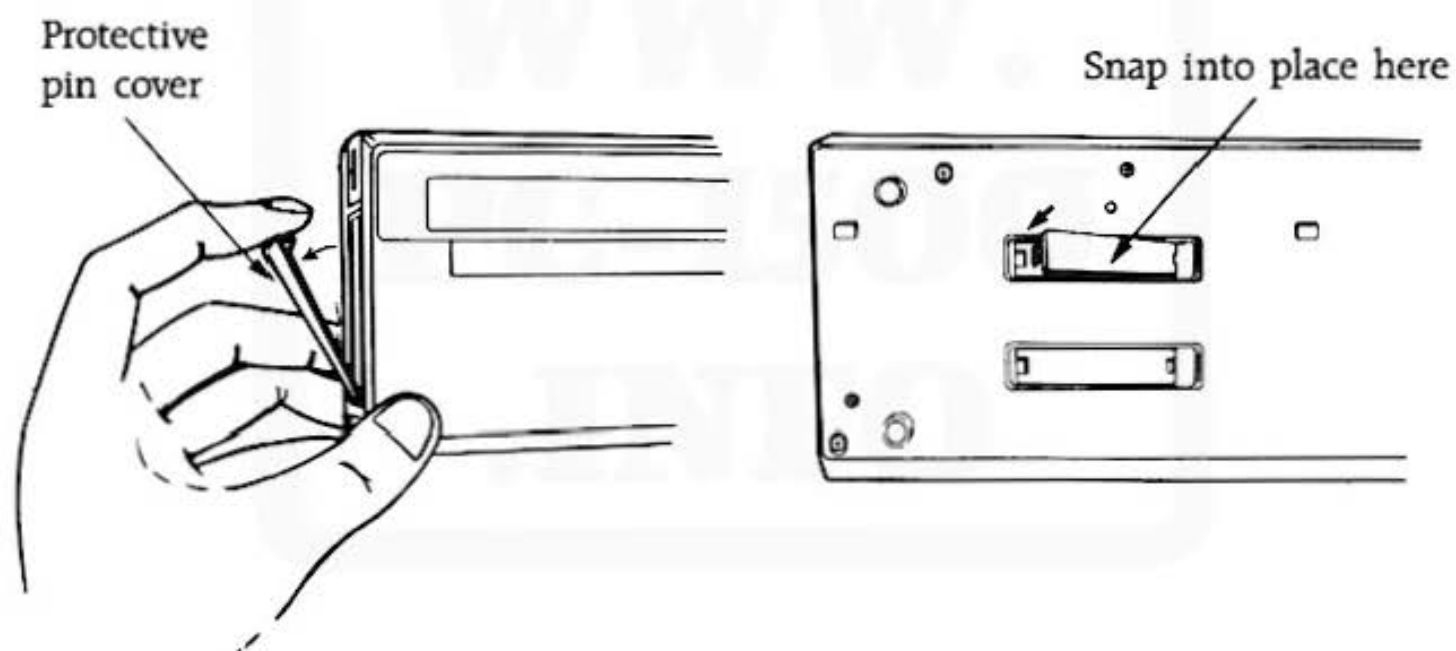
#### **Connecting the Computer to the Interface**

Connect the printer/cassette interface (CE-150) and the computer (PC-1500A) in the following procedure:

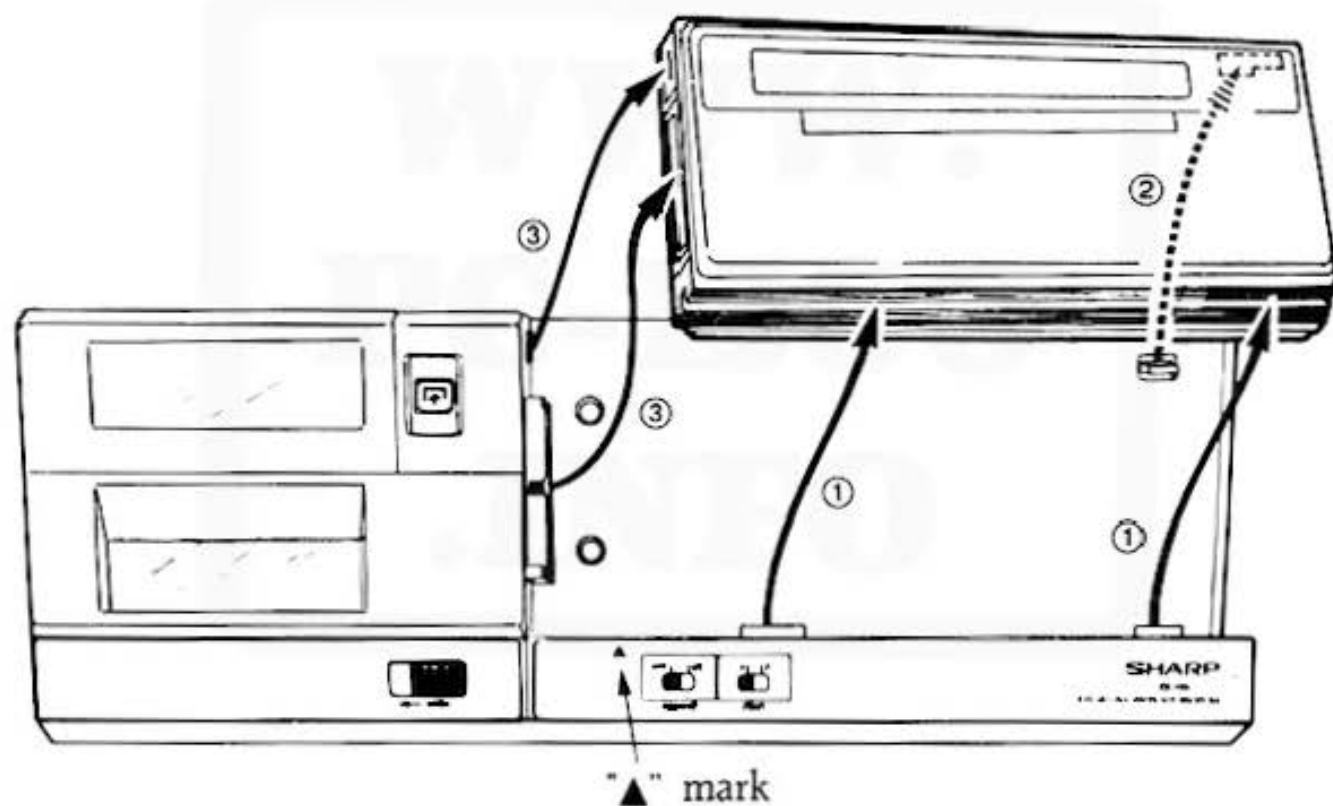
- (1) Turn the computer power OFF.

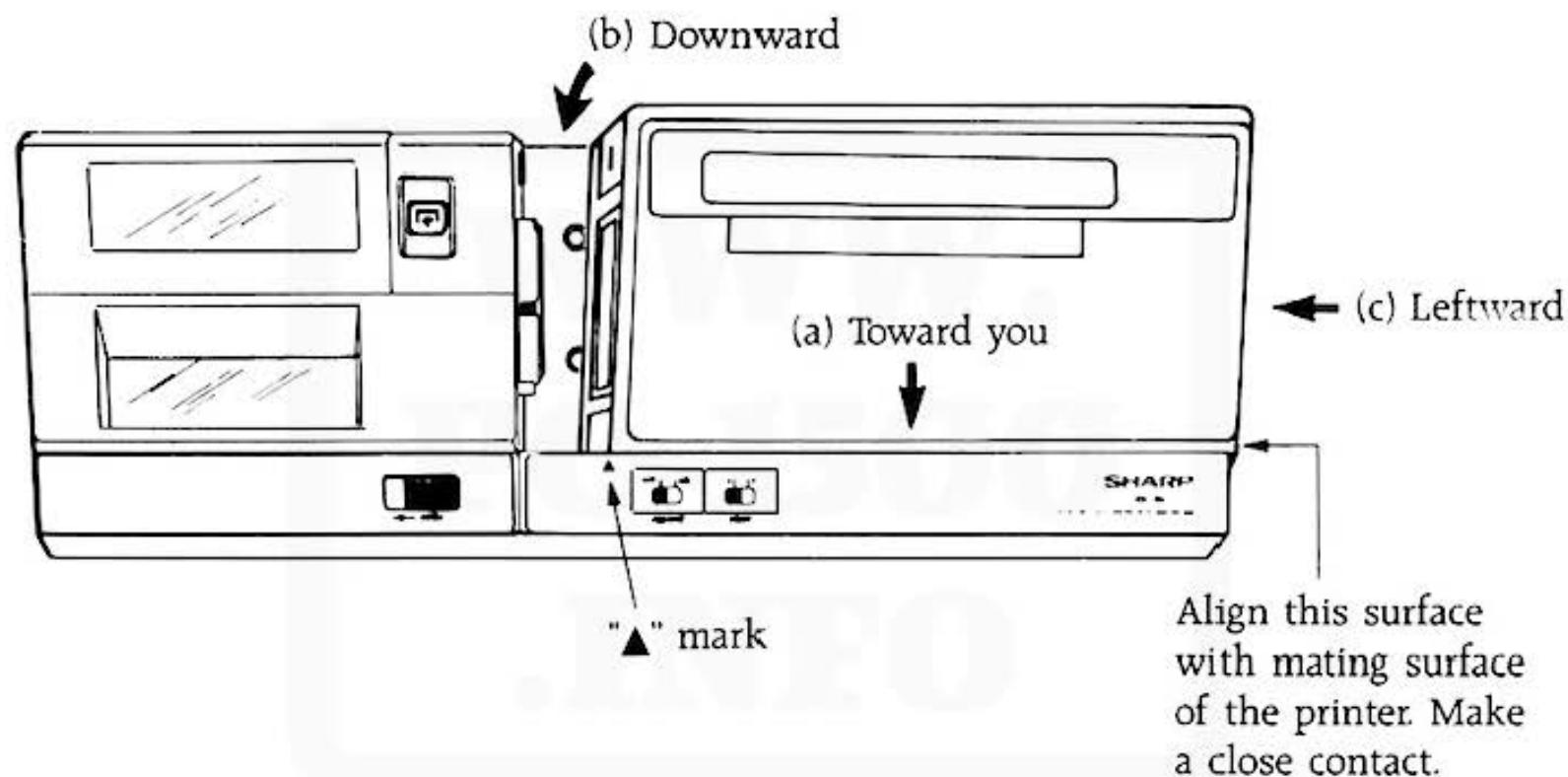
Important Note! It is essential that computer power be OFF. If power is ON, the computer may "hang up" (all keys inoperative). If this occurs, press the ALL RESET switch on the bottom of the computer while pressing the **ON** key.

- (2) Remove the protective pin cover from the left side of the computer and snap it into place on the bottom of the printer (see figure).



- (3) Place the lower edge of the computer into the "cradle" so that the printer guides match-up with the computer guide-slots.
- (4) Lay the computer down flat.
- (5) Gently slide the computer to the left so that the printer pins are inserted into the computer (see figure)





Do not force the computer and printer together. If match-up does not easily take place, carefully shift the computer left and right to correctly position mating surfaces.

Do not sale this PDF !!!

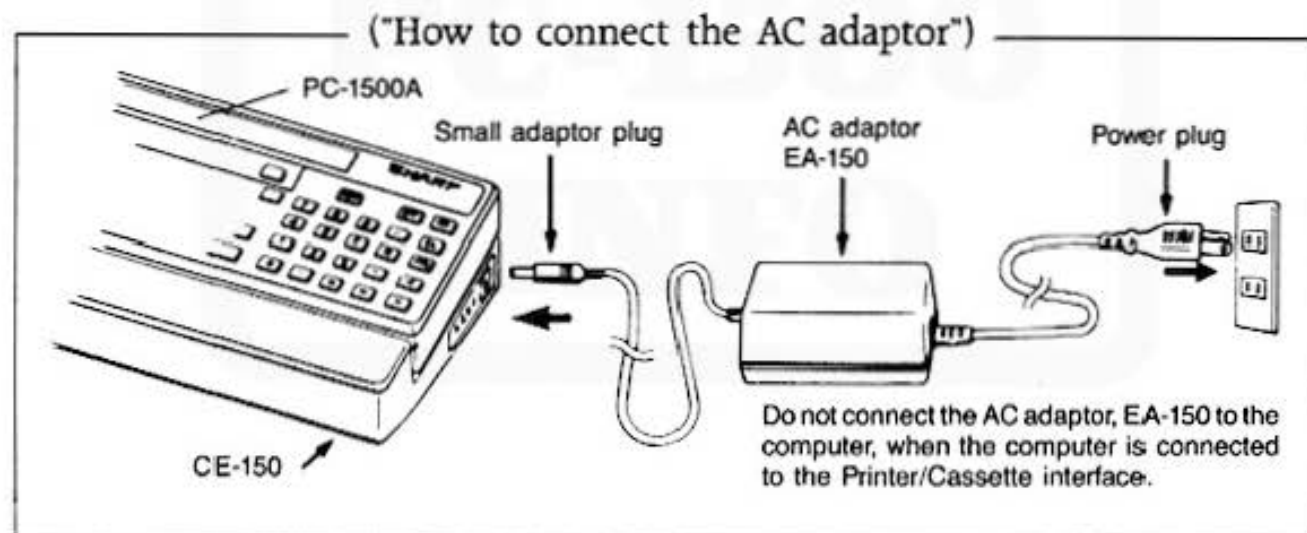
## Power

The CE-150 unit utilizes a rechargeable Ni-Cad battery power source. Therefore it is necessary to recharge the batteries after unpacking, and when the following message is displayed. (In this case, the printer is locked. To unlock the printer, press the **OFF** and **ON** keys of the computer in that order after charging the battery.)

(1) ERROR 80 or ERROR 78

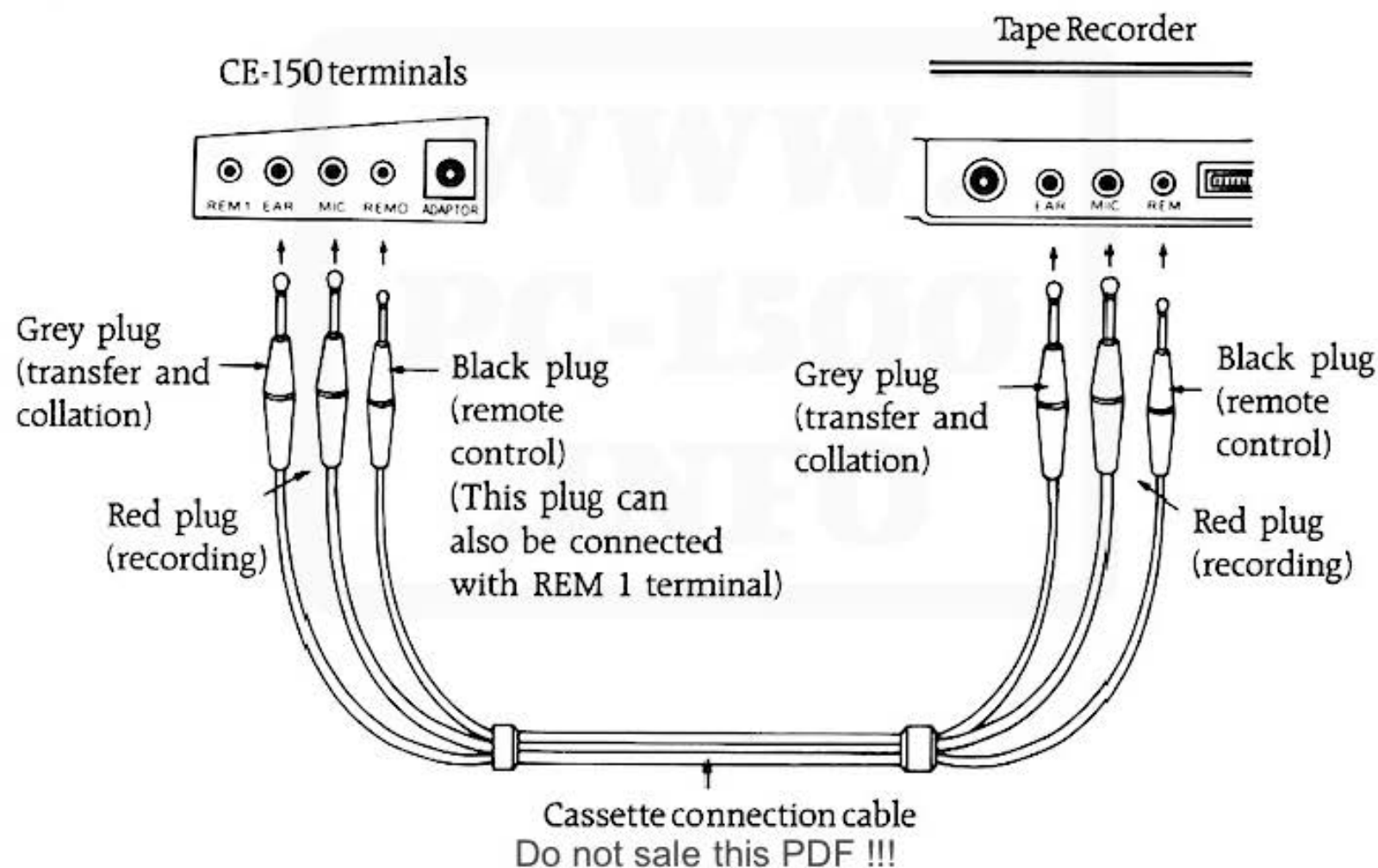
Note: when the printer is in the pen replacement state, the display of ERROR 78 may appear

(2) :CHECK 6 or NEWØ? : CHECK6



## Connecting a Tape Recorder to the Interface

First connect the CE-150 unit and computer, and connect a tape recorder with the CE-150 unit as shown in the following diagram.





The following is a description of the minimum tape recorder specifications necessary for interfacing with the CE-150:

Item	Requirements
1. Recorder Type	Any tape recorder, cassette, microcassette, or open reel recorders 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 mini-jack labeled "EXT" (EXternal speaker), "MONITOR", "EAR" (EARphone) 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 less than 15% within a range of 2 kHz through 4kHz.
9. Wow and Flutter	0.3% maximum (W.R.M.S.)
10. Other	Recorder motor should not fluctuate in speed.

\* In case the mini-plug provided with the CE-150 is incompatible with the input/output jacks of your tape recorder, special line conversion plugs are available on the market.

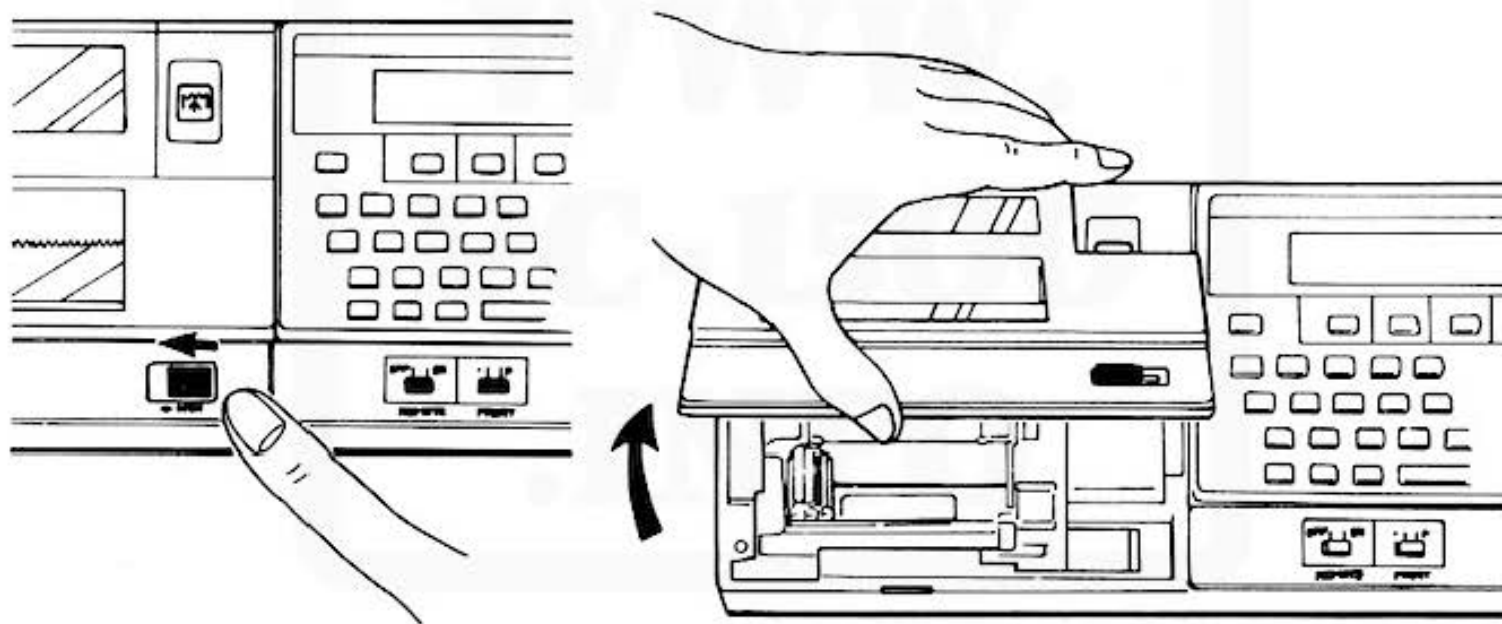
NOTE:

- Some tape recorders may reject the connection due to different specifications. Those tape recorders having distortion, increased noise, and power deterioration after long years of use may not show satisfactory results owing to changes in their electrical characteristics.
- Precautionary Instructions for Tape Recorder:
  - (1) For any transfer or collation, use the tape recorder that was used for recording. If the tape recorder is different from that used for recording, transfer or collation may not be possible.
  - (2) The head of a tape recorder, if dirty, increases distortion and decreases the recording level. Therefore, keep the head clean.
  - (3) Use a tape that is free of dirt, scratches, and creases.

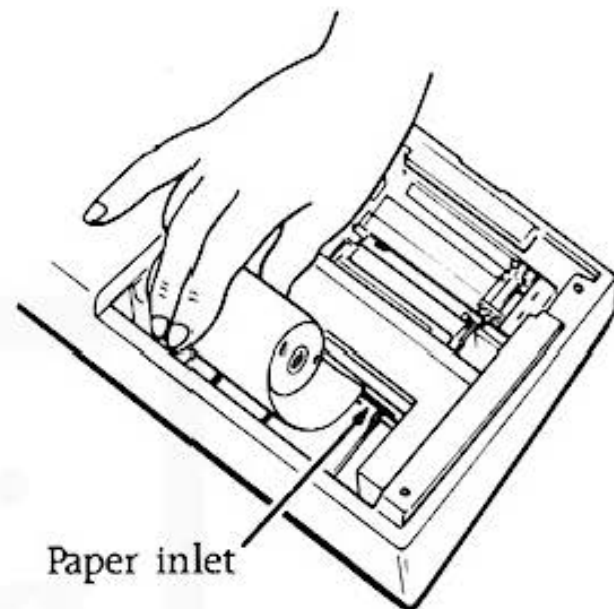
## Loading the Paper


For details refer to the instruction manual for CE-150.

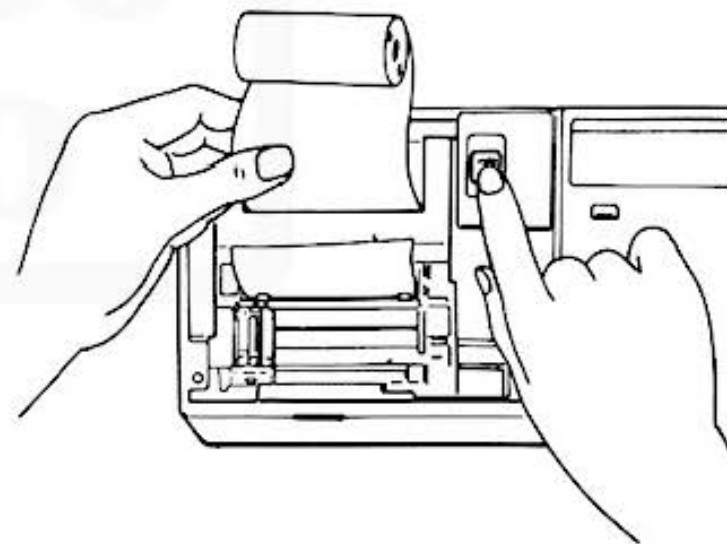
- (1) To remove the printer cover, shift the printer cover lock lever in the direction of the arrow.



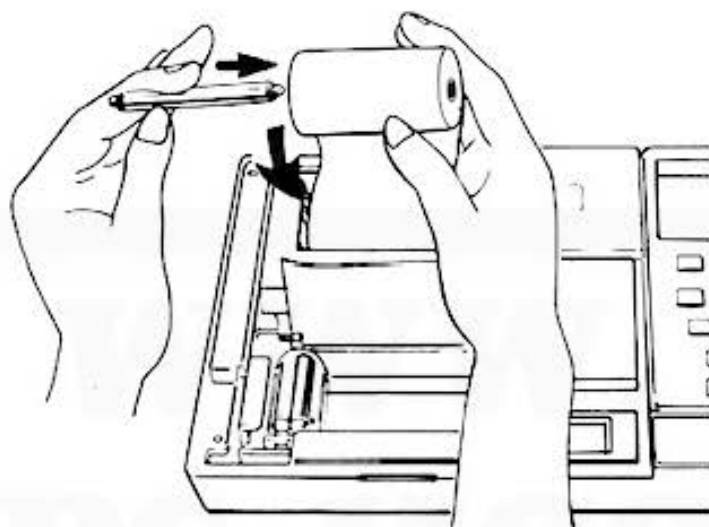
- (2) Cut the tip of roll paper straight, and insert the paper correctly into the paper inlet. (Any curve or crease at the paper tip may prevent paper insertion.)



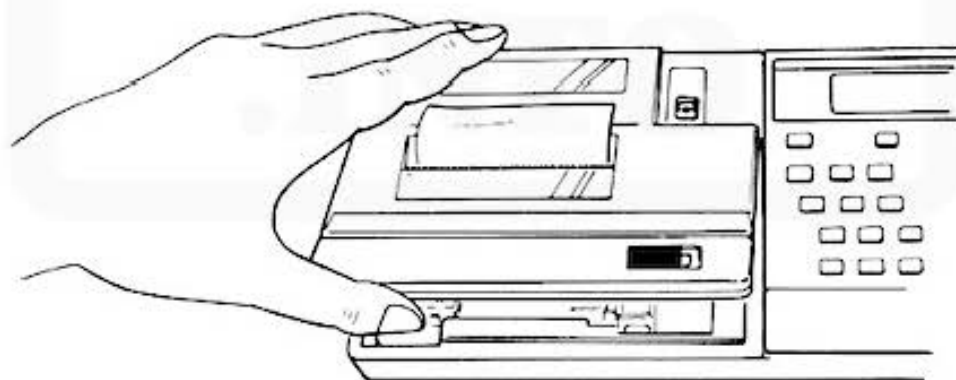
- (3) Press the computer **ON** key to turn it on, and press the  key to feed paper. At this time, feed the paper so that the paper tip may be 1 to 2 in (3 to 5 cm) from the printer.



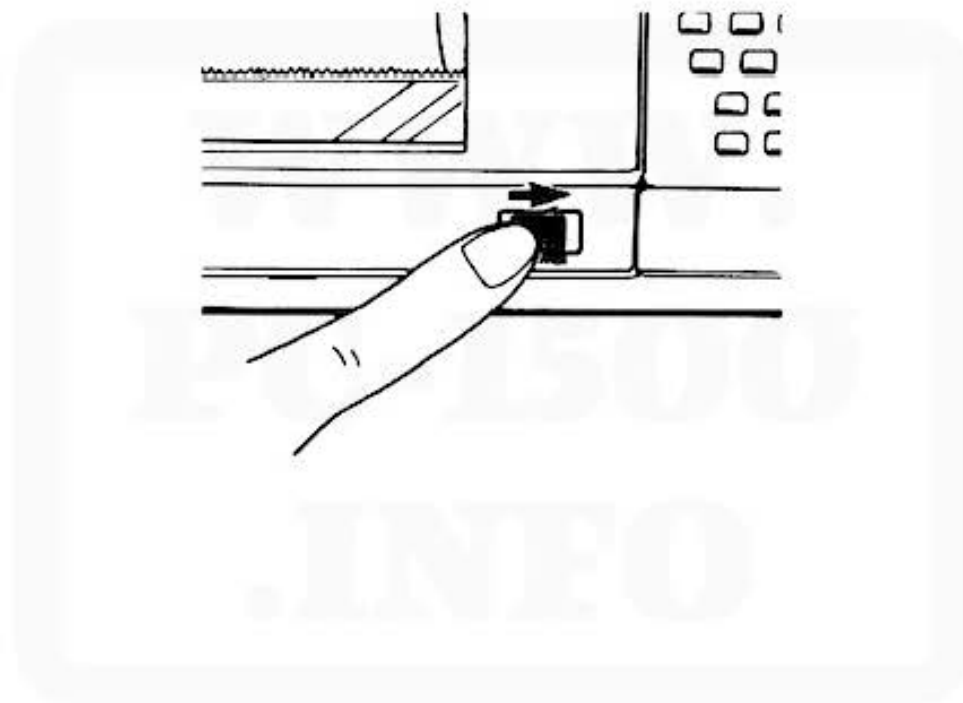
- (4) Insert the shaft into the roll paper and place the paper in the paper case.



- (5) Place the printer cover back in position. At this time, thread the roll paper tip out of the printer through the paper cutter.

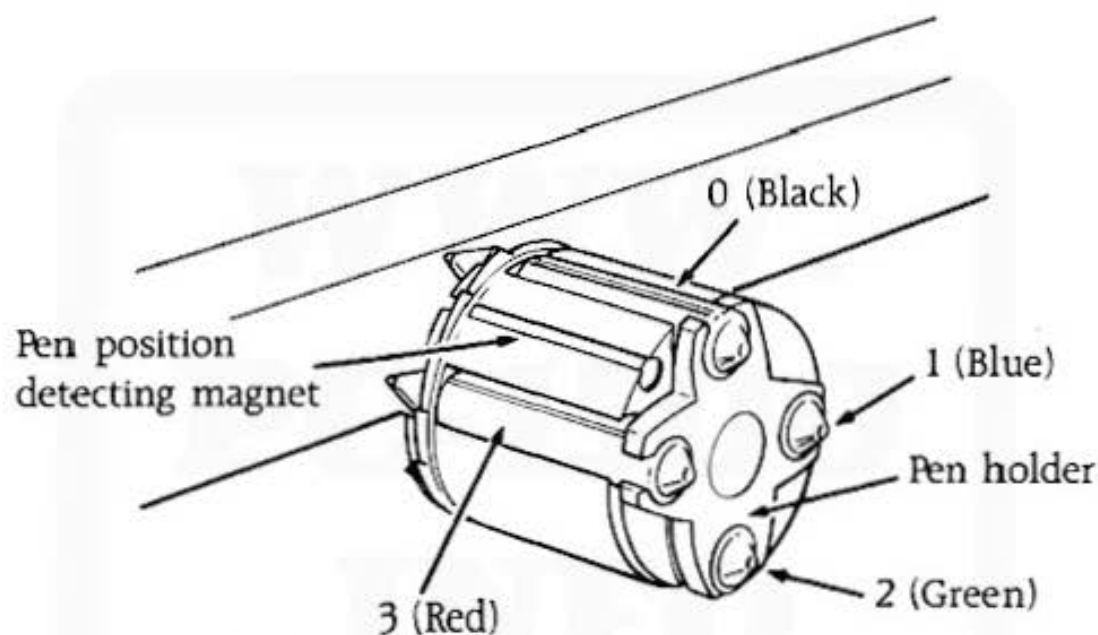


(6) Lock the printer cover.




## Replacing the Pens

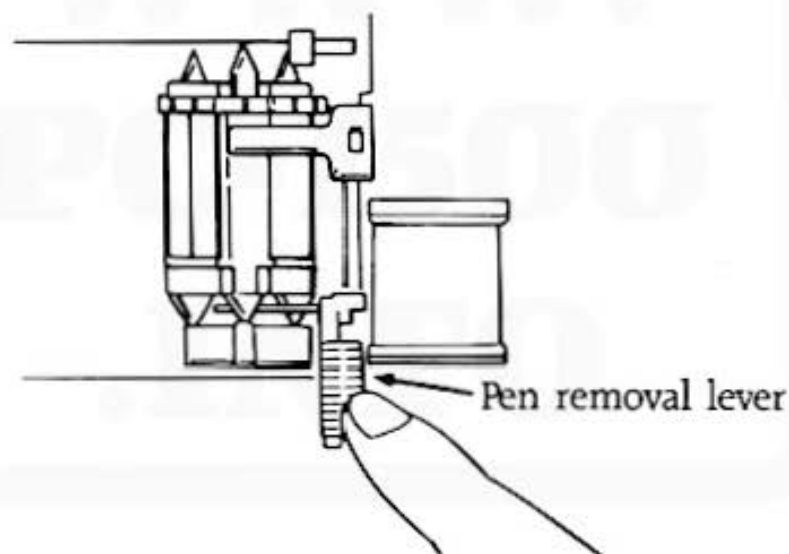
Four kinds of pens can be installed on this unit. Pen installation positions are as illustrated below; for details refer to the instruction manual.



When designated by the COLOR instruction, the pen positions number 0, 1, 2, and 3 clockwise from the pen position detecting magnet as illustrated above. (The pen holder rotates counterclockwise so that the selected pen comes atop.)

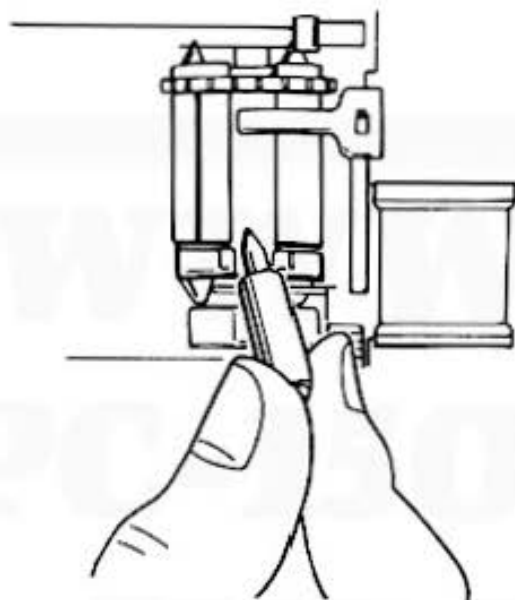
For pen installation or replacement, follow the procedure below:

- (1) With the computer zero (0) key pressed, press the printer . This allows the printer to be in its pen replacement state, when the pen holder shifts to the left and rotates. With the pen on top changed, the pen holder moves to the right. (Release the key when the pen holder starts moving.)
- (2) To remove the pen, press the pen removal lever. This causes the pen on top to come off.







(3) Install a new pen.



Note: Pen replacements are available wherever the CE-150 is sold. Please order product No. EA-850B (4 black pens per package) or EA-850C (1 black, 1 blue, 1 green, and 1 red pen per package) when reordering pens. The pens are specifically designed for this unique printer. Use of any other pen may cause damage to the unit.

- (4) To install or remove the next pen, press the  key. The pen holder returns to the left, rotates so that the next pen comes atop, and shifts to the right again. Remove the pen and replace it with a new one as in steps (2) and (3).
- (5) After the pen replacement or installation, press the printer  with the computer **CL** key pushed down. This causes the printer to be released from its pen replacement state, and the pen to return to the left.

Note: To use this unit, install the four pens on the pen holder. Operation with a lack of even one pen may cause color changes to malfunction.

### Handling the Pens

The pens are installed on the printer when it is used, and removed from the printer after use. Cap the pens and place them in their refill for storage.

Leaving the pens installed on the printer for a long time or placing them on a desk *may cause the ink to dry*.

## SECTION 13: USING A CASSETTE RECORDER

### Tape Recorder Operation

The procedures listed below are to be followed when using a tape recorder for saving or loading programs and data.

1. Turn the remote switch on the CE-150 interface to the OFF position.
2. Put tape into recorder.
3. Advance tape to desired location (take care not to attempt to save a program on the leader portion of the tape)
4. If the recorder has a manual volume control, set it at approximately  $\frac{3}{4}$  of the maximum level. Set the tone control at  $\frac{3}{4}$  of the maximum level.
5. Turn the remote switch to the ON position.
6. Depress both the RECORD and PLAY keys simultaneously.

## Recording and Retrieving Programs

### CSAVE (Saving Programs)

CSAVE "filename"

The CSAVE command is used to save a program on an audio cassette tape. This command may be used in both RUN and PRO modes. A file name should be assigned to the program so that it can be distinguished from any other on the tape. A file name may be a string with up to 16 characters; spaces are not allowed. While the program is being saved onto tape, the BUSY indicator should be present in the display. After the program has been saved, the recorder will stop and the prompt will be displayed.

In order to save the contents of the softkey buffer to tape, the user must use the CSAVE command while in the RESERVE mode. This is done precisely the same as one would save a program to the cassette tape recorder. The format is as follows.

CSAVE "filename" (in RESERVE mode)

### CLOAD (Loading Programs)

CLOAD "filename"

The CLOAD command is the complement to the CSAVE command. It will let you load programs that have been saved on cassette tape. The file name does not have to be specified. When it is, only a program with that file name will be loaded; when it is not, the first program encountered will be loaded. If the program does not load

correctly, an error message will result. In this event, slightly modify the tone and volume levels of the cassette recorder.

### **Verifying Program Files**

CLOAD? "filename"

To verify that a program was saved correctly, rewind the cassette tape to the beginning of the program and use the CLOAD? command. This command will compare the contents of the program memory with what is on the tape. An error message will occur if the files do not match.

### **MERGE (Editing Programs)**

MERGE "filename"

The MERGE command allows you to store many programs in the computer memory at the same time. For example, let's assume the computer memory contains the following program:

```
10: PRINT "DEPRECIATION ALLOWANCE"  
20: INPUT "Enter method: ";A
```

At this point you remember that you have a similar program portion on tape under the filename "DEP1". You will, of course, want to see if this program has sections useful in the program you are currently constructing. The first step

is to find the tape with "DEP1" on it. Cue the tape to the place at which "DEP1" starts. Now type: **MERGE** "**DEP1**" and press the **ENTER** key. The computer will now load "DEP1" into memory IN ADDITION to the above program. After "DEP1" is loaded, you might find something in memory similar to this:

```
10: PRINT "DEPRECIATION ALLOWANCE"  
20: INPUT "Enter method: ";A  
10: "DEP1": REM >> Second Module <<  
20: PRINT "INTEREST CHARGES"  
30: INPUT "Amount Borrowed: ";B  
.  
.  
.
```

Unlike the CLOAD command, the new program DID NOT replace the existing one and that some line numbers have been duplicated. Also, note that a "label" was used on the first line of the merged module. This allows "linking" of the modules together (see discussion below)

It is important that you review the following information before proceeding with any further editing or programing.

Once a MERGE is performed, no insertions, deletions, or changes are allowed to previously existing program lines.

210   EXAMPLE:   10 "A" REM This is existing program  
                  20 FOR T=1 to 100  
                  30 LPRINT T  
                  40 NEXT T

.  
.  
.

Before doing a MERGE of the next program, make any necessary changes to this program. Then MERGE the next program: MERGE "PROG2" (example)

```
10 "B" REM This is MERGED program
20 INPUT "Enter depreciation: ";D
30 INPUT "Number of Years: ";Y
40 . Etc.
```

Now you may make changes to the above program since it was the last MERGED portion. If you need to make further changes to the first program then follow this procedure:

1. CSAVE what you have done to this point: use a new name when saving to tape (e.g., "PROG3")
2. CLOAD the program saved in step one back into memory.
3. Now make any desired modifications or changes. However, keep this in mind: changes can be made ONLY to the first program or to the FIRST OCCURRENCE of any duplicated line numbers (i.e., if program line 10



appears in the first program as well as the second, changes will only affect the first occurrence of line 10). Thus if you attempt to edit the second occurrence of line 10, the change will be erroneously reflected in the first program portion only.

Additional lines may be added at the end of the existing programs only if they have line numbers greater than those previously used. Note that additional program lines may not necessarily appear at the "bottom" of the listing. Since modules must be linked via labels (see below), this should not be of concern.

Since the processor executes your program lines in logical sequence, it will stop when it encounters a break in the sequence in line numbering. Thus, if line numbers 10, 20, and 30 are followed by duplicate line numbers in a second module, the processor will cease execution after line 30 in the first module. To "link" various modules together, the following techniques are valid: GOTO "B", GOSUB "B", IF . . . THEN "B" (B is used for example only; you can use any label except reserved words or letters which appear in row #2 on the keyboard, i.e., Q thru P).

### **The CHAIN Statement**

CHAIN "filename" , line number

The CHAIN statement is used to run programs in sections. This is especially helpful when you wish to run programs that are too large to fit into program memory. The program can be broken down into small, workable sections and saved on tape. By using the CHAIN statement, long and complex programs can be run.

During execution, when the computer encounters the CHAIN statement, the next section is called into memory and executed. In this manner, all of the sections are eventually run. Variable contents are not preserved when chaining to another program.



## Using Data Files

### PRINT# (Saving Data Files)

PRINT# "filename"; variable 1, variable 2

The PRINT# command is similar to the CSAVE command with the exception that it is used to save the values in a set of variables. The file name is optional but highly recommended if more than one file is to be saved on a tape. The first example is for saving the values in a series of variables and the second example is for saving the values in an array.

#### EXAMPLE 1

```
10:REM SAVING A SERIES OF VARIABLES  
20:PRINT# "FILE1":A$,B,C
```

#### EXAMPLE 2

```
10:REM SAVING AN ARRAY  
20:PRINT# "FILE2";B(*)
```

## **INPUT# (Reading Data Files)**

INPUT# "filename"; variable name

This command is the complement of PRINT#. It is used to enter data from the cassette recorder into the PC-1500A. This command can be used either manually or as part of a program.

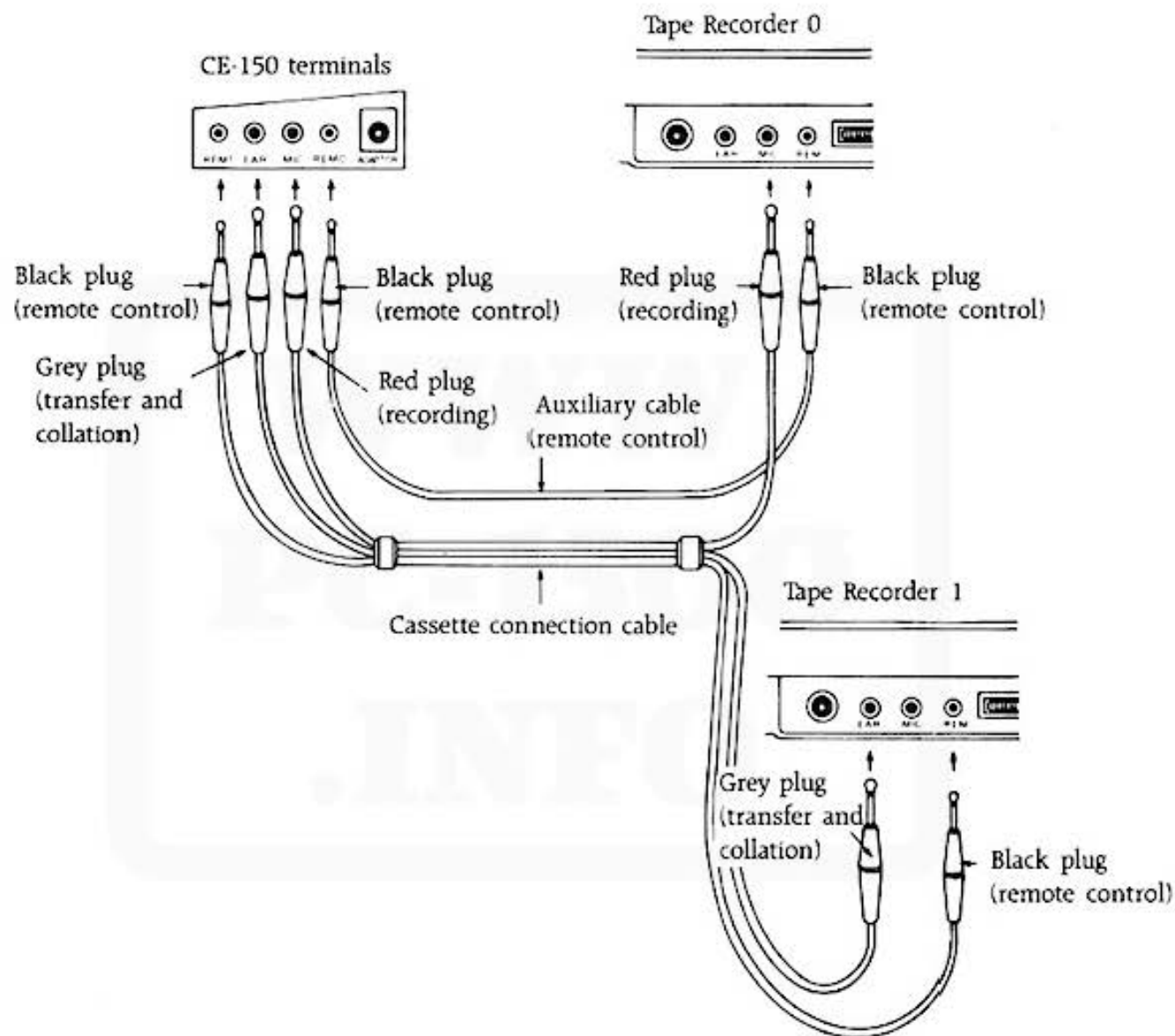
NOTE: If you are specifying more variables than exist on tape, the remaining variables will be assigned a value of 0. If fewer variables are specified, the remaining values will be ignored.

## **Using Two Tape Recorders**

When using two tape recorders, one of them is used for recording and the other for readout (transfer). As illustrated, the Printer/Cassette interface and the two tape recorders are connected by using the connection cords and the auxiliary cord for remote control.

- The CE-150 is equipped with two remote control terminals, REM 0 and REM 1, either of which can be used. In program operation (not manually) however, the program designates the tape recorder connected to the REM 0 terminal or the REM 1 terminal. Therefore, these remote control terminals should be connected according to the program.

In this section, how to operate the second tape recorder connected to the REM 1 terminal of the interface is explained.



## 1. Saving Procedures:

- (1) Type **RMT OFF** and press **ENTER** to reset the second remote control function. (control of Tape Recorder 1 in the illustration above)
- (2) Put a tape into the tape recorder.
- (3) Type **RMT ON** and press **ENTER** to set the second remote control function.
- (4) Set the volume and tone controls in the same manner as that explained previously in the single tape recorder
- (5) Depress both the RECORD and the PLAY buttons simultaneously.
- (6) Execute RECORD instruction.

Program: **CSAVE-1** "*file name*" **ENTER**

Data: **PRINT # -1** "*file name*"; *variable, variable, ...* .

(Example) Designate PRO or RUN mode.

**CSAVE-1"PR-1"** **ENTER**

After saving, the "prompt" will re-appear on the display and the tape will stop. Rewind the tape for collation.

## 2. Collating the Computer and Tape Contents Procedures:

- (1) Type **RMT OFF** **(ENTER)** to clear remote control functions.
- (2) Rewind the tape to the place at which you started, again using the number counter.
- (3) Enter **RMT ON** and the **(ENTER)** key to set remote control functions.
- (4) Set the volume and tone controls in the same manner as that explained previously in the single tape recorder.
- (5) Press **PLAYBACK** button.
- (6) Execute **COLLATION** instructions.

**CLOAD?-1** "*file name*" **(ENTER)**

(Example) Designate **PRO** or **RUN** mode. **CLOAD?-1** "**PR-1**" **(ENTER)**

Execution ends when both contents match, resulting in prompt displays.

### 3. Transfer from Tape Procedures:

- (1) Enter **RMT OFF** and the **(ENTER)** key to clear remote control functions.
- (2) Put a recorded tape into the tape recorder.
- (3) Type **RMT ON** and press the **(ENTER)** to set remote control functions.
- (4) Set the volume and tone controls in the same manner as that explained previously in the single tape recorder.
- (5) Press the PLAYBACK button.
- (6) Execute TRANSFER instruction.

Program: **CLOAD-1** "*file name*" **(ENTER)**

Data: **INPUT # -1** "*file name*"; *variable, variable, .....* **(ENTER)**

(Example) Designate PRO or RUN mode.

**CLOAD-1** "**PR-1**" **(ENTER)**

After the transfer, the prompt displays the result.

## SECTION 14: USING THE PRINTER

### CE-150 Printer Specifications

Characters/line:	4, 5, 6, 7, 9, 12, 18, or 36 depending on character size chosen.
Character Size:	1.2×0.8mm to 10.8×7.2mm depending on character size chosen.
Printing Speed:	Maximum: 11 characters per second when printing smallest characters.
Rotation:	Characters may be printed in either of two directions on either of two axes.
Colors:	4 — Red, Blue, Green, Black.
Graphing System:	X-Y axis plotting.
Paper Feed:	Manual or Programmable.

## TEST Command

The first thing you will want to do is to test the functioning of the CE-150 printer. With the computer and interface ON, type:

**TEST** (and press **ENTER**)

The printer will now draw 4 boxes, each a different color. The color of the boxes, from left to right, is what you chose as Colors 0 through 3 when you inserted the pens (see pp. 202-205).

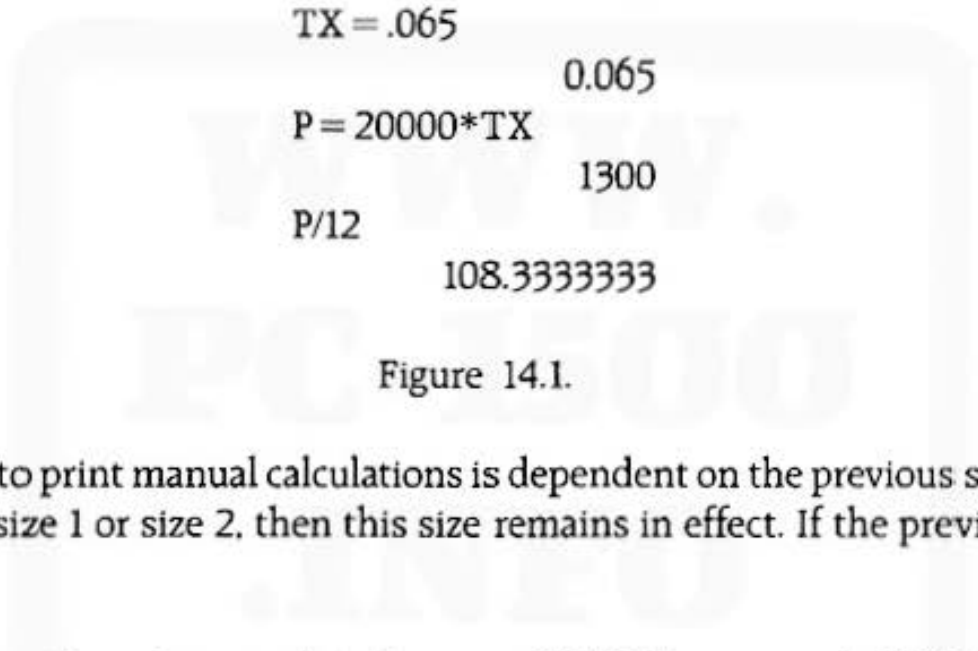
## Printing Calculations

Using the CE-150 interface it is possible to make a printed copy of a series of manual calculations performed on the PC-1500A computer as in Figure 14.1. To do this, simply set the Print Switch on the interface to the "P" position.

To prevent the automatic printing of manual calculations, the Print Switch must be placed in the "." position. With the switch in this position, you may still print out selected results by prefacing the computation with an LPRINT command (see LPRINT). Other commands which cause printing or drawing, such as LLIST, LINE, and others, are also functional in this mode.

When printing, the color used will be the color which was previously specified. If you have just turned the machine on, this will be whatever color pen you have selected to correspond to Color 0. To change the color, you must issue the COLOR command (see p. 233).





```

12000*.065
780
780 + 25.56
805.56
SIN 30
0.5
TX = .065
0.065
P = 20000*TX
1300
P/12
108.3333333

```

Figure 14.1.

The character size used to print manual calculations is dependent on the previous specification. If the previous character size specified was size 1 or size 2, then this size remains in effect. If the previous size was larger than 2, then size 2 is used.

Automatic printing causes the printer mode to be set to TEXT. If you were in GRAPH mode and wish to return to this mode, you must issue the GRAPH command. (Printer modes are explained in the next section.)

## TEXT and GRAPH (Printer Modes)

As the printer is operating, it may be in one of two modes: TEXT or GRAPH. These modes correspond roughly to human typing versus human drawing. Since most commands work only in one mode or the other, it is important to select the proper mode before issuing instructions.

The TEXT mode is used for printing numbers and characters. The width of the printer's paper is divided into columns, the number of which is related to the specified character size. Vertical and horizontal tabbing commands are provided to format text information.

In the GRAPH mode, a variety of diverse figures, charts, and tables may be created. Commands to draw both solid and dashed lines, using either a direct or relative coordinate system, are available. All drawings utilize a normal X — Y coordinate scheme.

To specify TEXT mode, the statement:

TEXT

is sufficient. Certain commands (discussed later) cause an automatic switch to the TEXT mode.

Specifying the GRAPH mode is equally simple. The statement:

GRAPH

**LLIST (Listing Programs)**

The LLIST command causes the current program, or portions of the program, to be printed. Because selective printing of program sections is possible, the LLIST command is extremely helpful during the program development process.

The form of the LLIST command is similar to the form of the LIST command. Because LLIST is more versatile, there are subtle differences. The LLIST forms are as follows:

LLIST

— Prints all program lines currently in the program memory.

LLIST expression

— Prints only the program line whose line number is given by the expression.

LLIST .expression

— Prints all program lines up to, and including, the line whose number is given by the expression.

LLIST expression,

— Prints program lines beginning with the line whose number is given by expression.

LLIST expression 1, expression 2

— Prints program lines beginning with the line whose number is given by expression 1 and ending with the line whose number is given by expression 2. Thus, if the command is:

## LLIST 100, 150

then all lines between 100 and 150 ( if any) will be listed.

### LLIST "label"

— Prints the program line containing the given label.

### LLIST "label",

— Prints program lines beginning with the line containing the given label and continuing to the end.

NOTE: Specification of a label which does not exist will be signaled by an ERROR 11 message.

When printing a program, the color used will be the color which was previously specified. To change the color, you must issue the COLOR command (see p. 233)

The character size used to list a program is dependent on the previous specification. If the previous character size specified was size 1 or size 2, then this size remains in effect. If the previous size was larger than 2, then size 2 is used.

The LLIST command causes the printer mode to be set to TEXT. If you were in GRAPH mode and wish to return to this mode, you must issue the GRAPH command.

While listing a program, the PC-1500A computer attempts to justify the program lines for readability. This is done by leaving spaces in the line numbers. Line numbers which are 1 to 3 digits wide will be right justified with a 3 character field. Line numbers which are 4 or 5 digits wide will be printed in a 5 character field:

```
10: REM WIDTH 3
20: REM WIDTH 3
300: REM WIDTH 3
2001: REM WIDTH 5
2010: REM WIDTH 5
```

### **LPRINT (Printing Text)**

The LPRINT command is the main command for displaying text information on the printer. It is similar in nature to the PRINT command of the PC-1500A, and they share several of the same forms. However, because of the additional features available on the printer, the actions of the LPRINT instruction are more complex. We will therefore concentrate on the subtleties involved in using the LPRINT statement.

The following discussion of the LPRINT command assumes TEXT mode only. Although the given forms may work in GRAPH mode, their operation will be different.

The printing of a single item remains generally the same:

```
LPRINT item
```

where item is an expression, character string, number, or name of a variable whose contents will be printed. As usual, characters are left justified and numbers are right justified.

Like the cursor on the display, if the pen is not positioned at the left of the paper, printing will begin from the pen's position. The position of the pen may be changed by the LCURSOR statement or by the TAB clause (see pp. 228-229).

A problem occurs when one tries to print an item which is too big to fit on one print line because of the current character size. If the item is a number, an ERROR 76 will result. If the item is a character string, the string will be continued on the next line.

Concern for the size of a printed item is also important for the two-item LPRINT statement, whose form is:

LPRINT item 1, item 2

Using CSIZE 1 guarantees that two numeric items will be printed on the same line. In this case, the items will be justified in the usual manner within the two halves of the printed line. For LPRINTs involving strings, the picture is not so clear. If the items both fit, they are justified as usual and printed on the same line. If they do not fit, the results are usually split across two lines. For larger character sizes, the two items are printed on two successive lines.

The semicolon may also be used in the LPRINT statement. It serves both to indicate minimum spacing of items and, at the end of a statement, to group successively printed items on the same line. In either case, if the total

length of the items exceeds the capacity of the print line, the items are printed on successive lines (as many as necessary). The LPRINT with the semicolon has the form:

LPRINT item 1 ; item 2 ; ... (etc)

or the form:

LPRINT item-list;

Several of the aforementioned forms are used in the following demonstration program:

```
10 A$="ABCDEFGF"  
20 B=123456  
30 FOR I=1 TO 3  
40 CSIZE I  
50 LPRINT A$  
60 LPRINT A$, B  
70 LPRINT A$; B  
80 LF 5  
90 NEXT I
```

This program produces the output:

```
ABCDEFGG      123456
ABCDEFGG
ABCDEFGG 123456
```

```
ABCDEFGG
ABCDEFGG
                        123456
ABCDEFGG 123456
```

```
ABCDEFGG
ABCDEFGG
                        123456
ABCDEFGG  1234
56
```

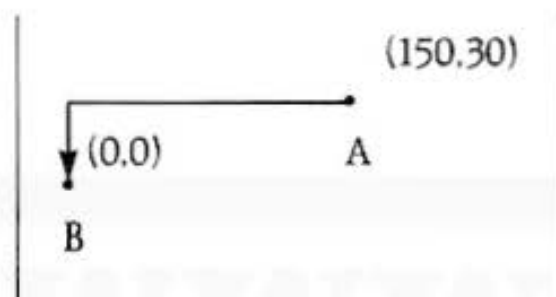
The LPRINT command may be used without an item specification, in either TEXT or GRAPHIC mode:

LPRINT

Used in this manner, it will cause a Carriage Return and a single Line Feed. It does not, however, reset the counters of the GRAPHIC mode. Thus, in the following example, although the LPRINT statement has been used to



move the pen from Point A to Point B, the printer believes itself to be at coordinates (150,30) and will execute all subsequent commands as if it were.



The LPRINT statement also incorporates a USING clause which operates in the same manner as it does in the PRINT statement. The USING clause may only occur in an LPRINT statement which is executed in the GRAPH mode.

## LCURSOR

The LCURSOR statement allows positioning of the pen in a manner analogous to the CURSOR statement of the display. The form of the LCURSOR statement is:

LCURSOR position  
(TEXT mode only)

The character position to which the pen may be moved is, of course, dependent on the character size in effect. In general, the pen may be positioned at one space less than the maximum for the character size. For a list of the print line widths at each character size, refer to the CSIZE command in this section.

## **TAB**

The TAB statement is identical to the LCURSOR statement, except that it may be used within an LPRINT statement. This type of LPRINT statement has the form:

LPRINT TAB position; item-list

The same comments which apply to the position expression of LCURSOR apply to TAB. If the item-list is empty, the net result of the instruction listed above will be a Line Feed.

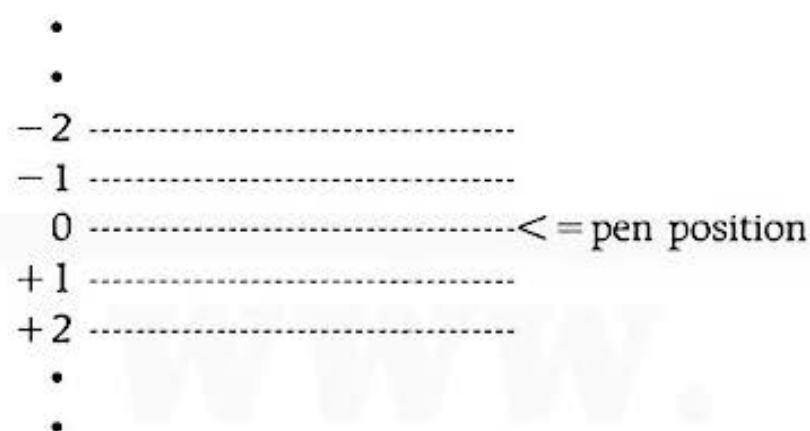
## **LF (Line Feed)**

The LF command causes the paper in the printer to be moved forward or backwards. The form of the command is:

LF expression  
(TEXT mode only)

If the expression evaluates to a positive number, the paper is advanced the number of lines specified by the expression. An expression which evaluates to a negative number will cause the paper to be pulled back the number of lines specified by the expression. This is illustrated by Figure 14.2.

(minus direction)



(plus direction)

Figure 14.2.

The actual distance the paper is moved is related to the character size in effect when the LF instruction is specified.

When the paper is traveling in the reverse direction (i.e., it is being pulled back), an internal counter prevents it from moving backwards more than 10.24 cm (about 4 in)

NOTE: Do not attempt to insert paper while the paper feed mechanism is operating. This can damage the printer

**CSIZE**

The CSIZE command specifies the size of the characters for all subsequent printing. There are nine sizes available, ranging from 36 characters per printed line to 4 characters per printed line. The form of the CSIZE command is:

CSIZE expression  
(either mode)

The expression must evaluate to a number in the range 1 through 9. The width and height of the characters for each size is given in the following table:

Table 14.1. Character size.

CSIZE	1	2	3	4	5	6	7	8	9
Characters per printed line.	36	18	12	9	7	6	5	4	4
Height of each character (mm)	1.2	2.4	3.6	4.8	6.0	7.2	8.4	9.6	10.8
Width of each character (mm)	0.8	1.6	2.4	3.2	4.0	4.8	5.6	6.4	7.2

**ROTATE**

The ROTATE command is used, in GRAPH mode only, to specify the direction in which printing occurs. Four directions are possible; Up, Down, Left to Right, and Right to Left (with the letter upside down). The directions are illustrated in Figure 14.3. The Form of the ROTATE command is:

ROTATE expression  
(GRAPH mode only)

The expression must evaluate to a number in the range 0 to 3. ROTATE 0 designates the normal manner of printing characters from Left to Right.

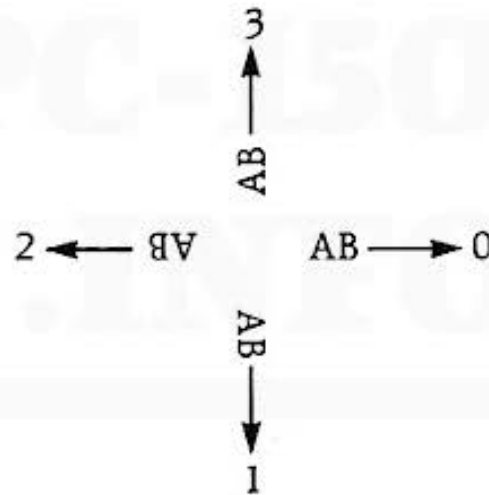


Figure 14.3.

## COLOR

The COLOR command allows the specification of the pen to be used for all subsequent printing and drawing. If each pen position contains a different color pen, then the COLOR command may be used to change pen colors (see pp. 202-205). The form of the COLOR command is:

COLOR expression  
(either mode)

The expression must evaluate to an integer in the range 0 to 3. Each integer corresponds to a different pen. The color represented by the integer will vary depending on the order in which the pens were loaded in the carrier. The correspondence may be determined by the TEST command (described on p. 219).

Numbers which are not integers but which are in the range 0 through 3 will be truncated to integers. All other numbers will cause an ERROR 19.

When the PC-1500A computer is turned OFF and then ON again, the pen which corresponds to zero is selected.

In the TEXT mode, execution of the COLOR command will cause the pen position to be reset to the left side of the paper. In the GRAPH mode, the pen will return to its previous position.

## Plotting Operations

### SORGN (Establishing the Origin)

The SORGN command is used to establish the origin of the X—Y coordinate system for subsequent graphing commands. The SORGN statement makes the current pen position the origin. Thus, this instruction is usually used directly after an instruction which moves the pen to a given spot on the paper. The form of the SORGN command is simply:

SORGN  
(GRAPH mode only)

NOTE: The CE-150 printer allows a pen position to be specified which is outside of the range of drawable positions. In this case, the pen moves as far as it can and then "cuts off." If the pen is moved into this imaginary realm and then the SORGN command is issued, subsequent printing or drawing statements will have no effect. It would thus appear as if the program was in error or that the interface was damaged.

The following program sets the origin to 100 units up and 100 units to the right of the pen's current position. It then draws a 10 unit box with one of the box's corners at the new origin:

```
10 GRAPH
20 LINE (0,0) — (100,100)9
30 SORGN
40 LINE (0,0) — (10,10)0,0,B
```

50 TEXT

60 END

### **GLCURSOR (Moving the Cursor)**

The GLCURSOR statement moves the pen to any X—Y coordinate without drawing a line. The form of the GLCURSOR statement is:

GLCURSOR (expression 1, expression 2)

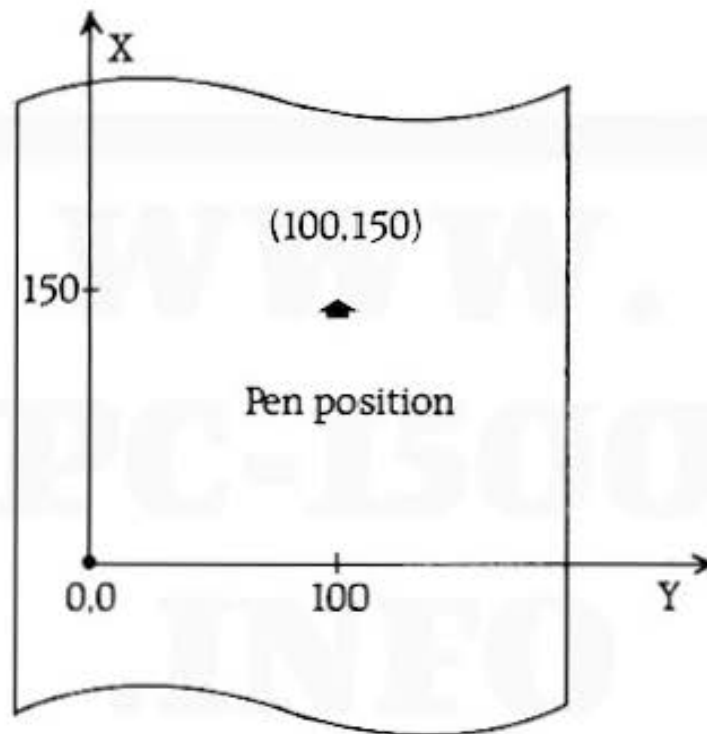
Both expressions must evaluate to a number in the range  $-2047$  to  $+2047$ . Expression 1 represents the X distance to the destination point and expression 2 represents the Y distance to the destination point.

NOTE: If the destination point is outside of the range of points to which the pen may actually move, the pen will stop at one side of the paper. Internally, the counters which control the pen are still counting toward the goal.

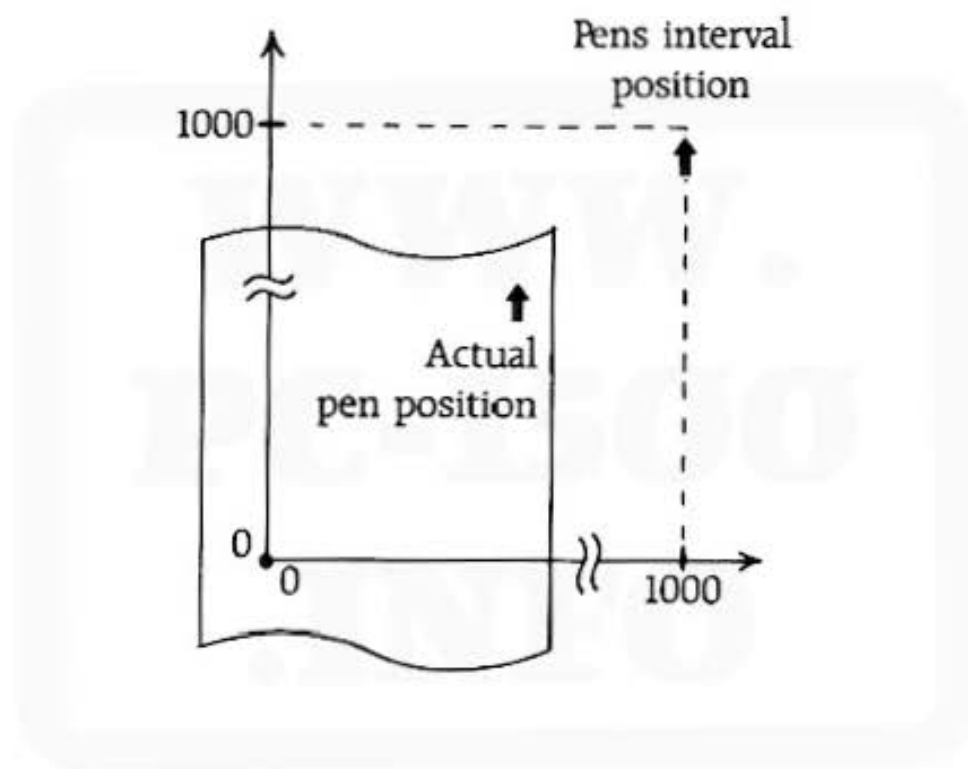
Examples of the use of the GLCURSOR statement follow.



236 In the example below, the pen is moved to position (100, 150)



In the next example, the pen is incorrectly positioned in the "imaginary" region at position (1000,1000)



The pen actually moves toward the right margin and scrolls the paper back. As it reaches the right side it continues upward until it reaches the built-in backing limit of 10cm., at which time it stops.

Do not sale this PDF !!!

**LINE (Drawing Lines)**

The LINE command is the primary command of the GRAPH mode. It specifies the movement of the pen from one point to another. If the pen is down as it moves, a line is drawn. The LINE command also allows dashed lines to be drawn with eight different dash lengths. The first form of the LINE command is:

LINE (X1,Y1) — (X2, Y2) line-type, color

The starting point of the line is determined by the values of the expressions X1 and Y1. The destination point of the line is determined by the values of the expression X2 and Y2. Both the X and the Y values must be in the range - 2048 to + 2047. A value specification which exceeds this range will produce an error.










The line-type and color parameters are optional. If they are omitted, the values used are the values which were in effect before the command. Color is, of course, one of the colors in the range 0 to 3. The line-type must be an expression which evaluates to a number in the range 0 through 9. Table 14.2 identifies the meaning of each option.

Used in another manner, the LINE command interprets the point specifications as the endpoints of a diagonal. It will then proceed to draw the box represented by the diagonal. The form for this LINE statement is:

LINE (X1,X2) — (Y1, Y2) line-type, color, B

The upper case B indicates that this is a box drawing command. The other parameters are the same as for the previous form of the command.

Table 14.2. LINE command options.

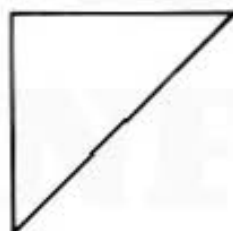
LINE-TYPE Value	RESULTING Line Size	
0	Solid Line	0 
1	0.4mm dash	1 
2	0.6mm dash	2 
3	0.8mm dash	3 
4	1.0mm dash	4 
5	1.2mm dash	5 
6	1.4mm dash	6 
7	1.6mm dash	7 
8	1.8mm dash	8 
9	Pen Up (no line)	9

The final form of the LINE command allows multiple point specifications to be made. Each point, after the first, represents a destination endpoint of the next line segment to be drawn. The current position is assumed as the starting endpoint for the line segment. This form of the LINE command is:

LINE (X1, Y1) — (X2, Y2) — ... (X6, Y6) line-type, color

The three dots in the form above are used to indicate that a series of point specifications (up to six in a row) may be given. Note that the B parameter may NOT be used in this form of the command. An example program to draw a triangle using four line segments is given below. Line 15 serves merely to establish the origin, while the triangle is drawn by line 20:

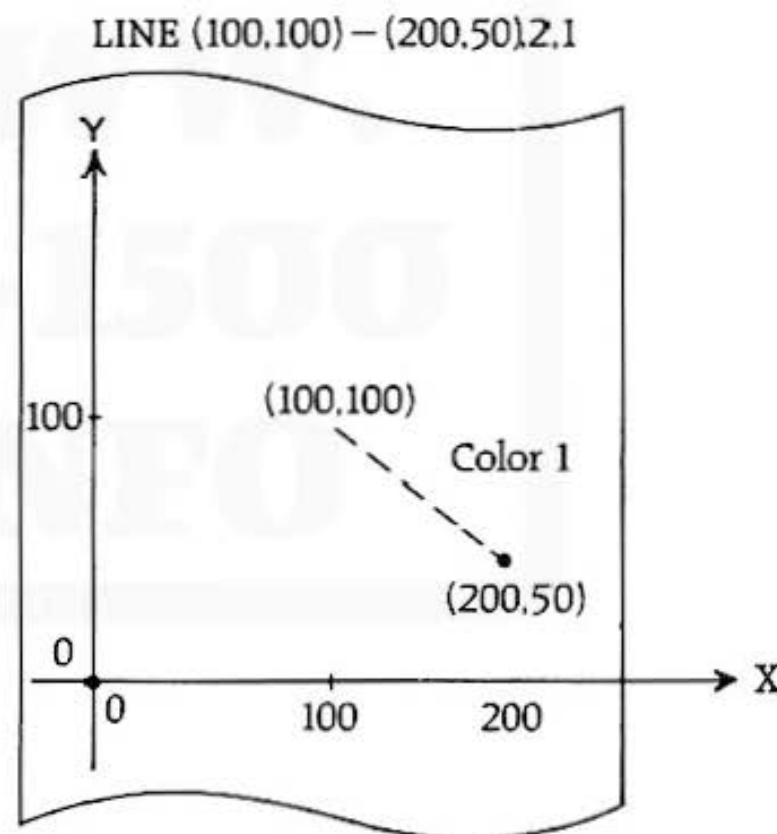
```
10: GRAPH
15: LINE (0, 0) - (10
    0,0) 9:SORGN
20: LINE (0,0) - (50
    , 50) - ( - 50,50) -
    ( - 50, - 50) - (0,0
    )0,0
30: TEXT
```



## RLINE (Drawing Lines)

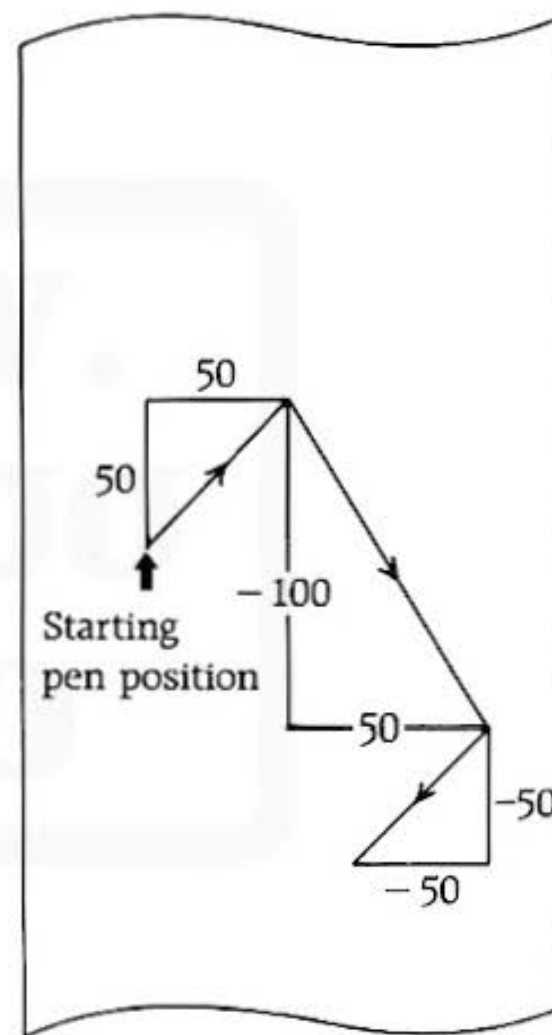
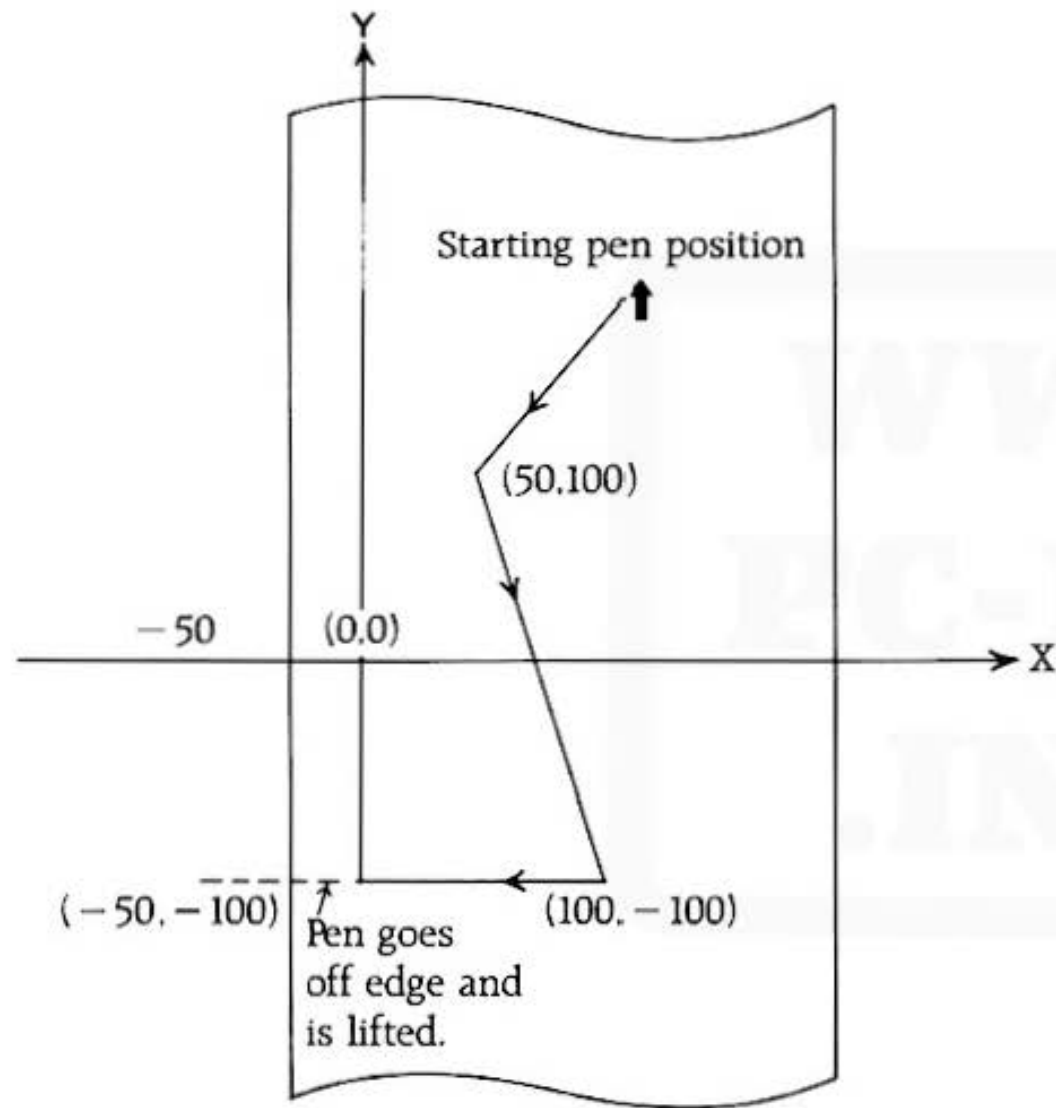
The RLINE command is the same as the LINE command except that all of the point specifications represent a position relative to the current pen position, rather than to the origin. The forms of the RLINE commands are the same as those of the LINE commands with the substitution of the word RLINE for LINE.

Examples follow:



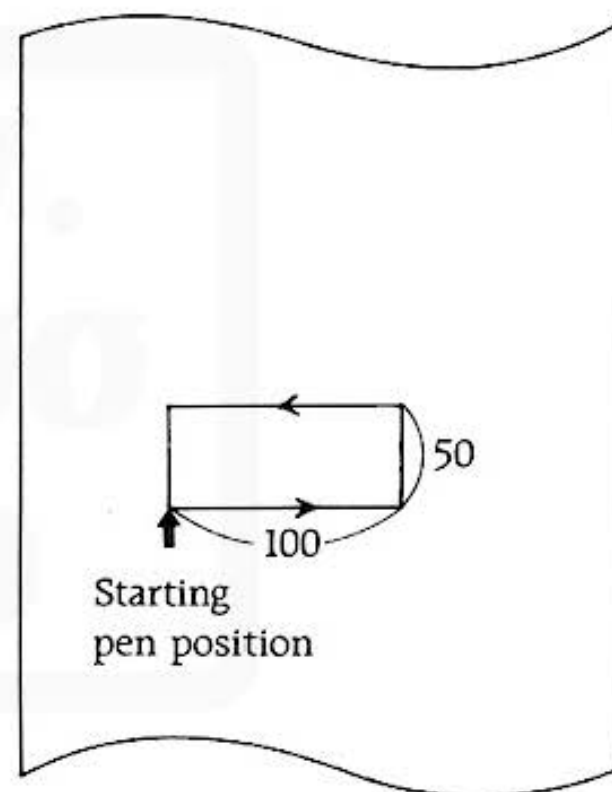
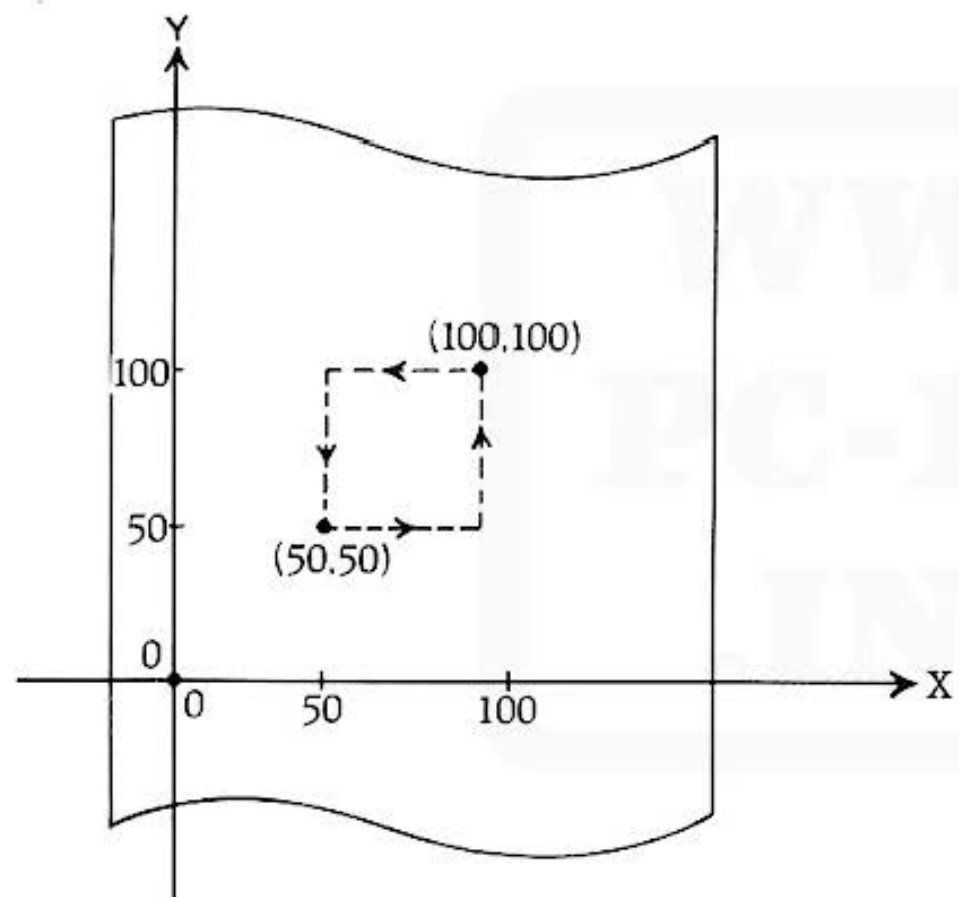
242 LINE - (50,100) - (100, - 100) - ( - 50, - 100)

RLINE - (50,50) - (50, - 100) - ( - 50, - 50)



LINE(50,50)-(100,100),2,,B

RLINE-(100,50),,B





## APPENDIX A: ASSEMBLY LANGUAGE PROGRAMMING

Since the release of the Sharp PC-1500 and its successor — the PC-1500A — on the market we have had a great number of questions from users in regard to the machine language of the PC-1500 and the PC-1500A. To assist such ardent users, Sharp has published a text designed especially to explain the machine language of the Sharp's original design LH5801 Microprocessor and LH5811 Peripheral Control LSI for the PC-1500 and PC-1500A systems.

This text, whose reference is listed below, is available from Sharp Corporation and its products distributors.

Sharp Corporation. 1983. Sharp Pocket Computer PC-1500: Technical Reference Manual. Sharp Corporation, Osaka, Japan. 159 pp.

## APPENDIX B: I/O PORTS

Pin	Signal	In/out	Functional description
1	PA1	In/out	Port A input/output. Key strobe
•	•	•	•
•	•	•	•
•	•	•	•
7	PA7	In/out	Port A input/output. Key strobe
8	GND	In	0V

9	PB0	In/out	Port B input/output.
10	OB1	In/out	Port B input/output.
11	PB2	In	Port B input/output. Cassette tape data input
12	PB3	In	Domestic/export specification select pin
13	PB4	In	User area determination pin
14	PB5	In	Clock input from TP terminal of the timer IC
15	PB6	In	Data input from the DATA OUT terminal of the timer IC
16	PB7	In	BREAK key input (interrupt input)
17	Po	In	PC port latch clock input
18	PC0	Out	Data output to the DATA IN terminal to the timer IC
19	PC1	Out	Strobe output to the STB terminal of the timer IC
20	PC2	Out	Clock output to the LK terminal of the timer IC
21	PC3	Out	Timer IC control signal output
22	PC4	Out	Timer IC control signal output
23	PC5	Out	Timer IC control signal output
24	PC6	Out	
25	PC7		
26	CS0	In	Chip select input connected to AD12
27	CS1	In	Chip select input connected to AD13
28	CS2	In	Chip select input connected to Y3 of the chip select decoder IC
29	RS0	In	Internal register and operation select signal
•	•	•	•
•	•	•	•
•	•	•	•

32	RS3	In	Internal register and operation select signal
33	R/W	In	Read/Write input
34	ME0	In	Memory enable and I/O port controller enable
35	ME1	In	Memory enable
36	W0	In	Wait condition input
37	W1	In	Wait condition input
38	GND	In	0V
39	VCC	In	+5V
40	DME0	Out	ROM enable
41	DME1	Out	ROM enable
42	WAIT	Out	Wait signal to the CPU
43	INT	Out	Interrupt request to the CPU
44	RESET	In	Initial reset signal
45	IRQ	In	Interrupt request input
46	oOS	In	Basic clock input
47	CL1		Not used. Serial data reception clock input
48	SD1		Not used. Serial data reception input
49	LC		Not used. LCD driver synchronizing signal
50	CL0	In	Serial data transmission/reception clock
51	SD0	In	Serial transmission/reception data. Used for the cassette tape data output
52	D0	In/out	Data bus
•	•	•	•
•	•	•	•

•	•	•	•
59	D7	In/out	Data bus
60	PA0	Out	Port A input/output. Used as the key strobe

## APPENDIX C: ABBREVIATIONS

### Printer Commands

COLOR

COL.  
COLD.

LCURSOR

LCU.  
LCUR.  
LCURS.  
LCURSO.

CSIZE

CSI.  
CSIZ.

LF

—

GLCURSOR

GL.  
GLC.  
GLCU.  
GLCUR.  
GLCURS.  
GLCURSO.

LINE

LIN.

LLIST

LL.  
LLI.  
LLIS.

GRAPH

GRAP.

LPRINT	LP. LPR. LPRI. LPRIN.	SORGN	SO. SOR. SORG.
RLINE	RL. RLI. RLIN.	TAB  TEST	—  TE. TES.
ROTATE	RO. ROT. ROTA. ROTAT.	TEXT	TEX.
Cassette Commands			
CHAIN	CHA. CHAI.	CLOAD?	CLO? CLOA.?
CLOAD	CLO. CLOA.	CSAVE	CS. CSA. CSAV.

INPUT #	I.#	PRINT #	P. #
	IN. #		PR. #
	INP. #		PRI. #
	INPU. #		PRIN. #

MERGE	MER.	RMT OFF	RM. OF
	MERG.	RMT ON	RM. O.

# Statements

AREAD	A.	CLEAR	CL.
	AR.		CLE.
	ARE.		CLEA.
	AREA.		

		CLS	—
--	--	-----	---

ARUN	ARU.		
------	------	--	--

BEEP	B.	CURSOR	CU.
	BE.		CUR.
	BEE.		CURS.
			CURSO.

DATA	DA. DAT.	GCURSOR	GCU. GCUR. GCURS. GCURSO.
DEGREE	DE. DEG. DEGR. DEGRE.	GOSUB	GOS. GOSU.
DIM	D. DI.	GOTO	G. GO. GOT.
END	E. EN.	GPRINT	GP. GPR. GPRI. GPRIN.
ERROR	ER. ERR. ERRO.	GRAD	GR. GRA.
FOR	F. FO.	IF	—

INPUT	I. IN. INP. INPU.	RADIAN	RAD. RADI. RADIA.
LET	LE.	RANDOM	RA. RAN. RAND. RANDO.
LOCK	LOC.		
NEXT	N. NE. NEX.	READ	REA.
		REM	—
ON	O.	RESTORE	RES. REST. RESTO. RESTOR.
PAUSE	PA. PAU. PAUS.		
PRINT	P. PR. PRI. PRIN.	RETURN	RE. RET. RETU. RETUR.



	STEP	STE.	UNLOCK	UN. UNL. UNLO. UNLOC.
	STOP	S. ST STO.		
			USING	U. US. USI. USIN.
	THEN	T TH. THE.		
	TRON	TR. TRO.	WAIT	W. WA. WAI.
	TROFF	TROF.		
Commands				
	CONT	C. CO. CON.	NEW  RUN	—  R. RU.
	LIST	L. LI. LIS.		

# Functions

ABS	AB.	INKEY\$	INK. INKE. INKEY.
ACS	AC.		
AND	AN.	INT	—
ASC	—	LEFT\$	LEF. LEFT
ASN	AS.	LEN	—
ATN	AT.	LOG	LO.
CHR\$	CH. CHR.	LN	—
COS	—	MEM	M. ME.
DEG	—	MID\$	MI. MID.
DMS	DM.		
EXP	EX.	NOT	NO.

OR	—	SQR	SQ.
PI	—	STATUS	STA. STAT STATU.
POINT	POI. POIN.	STR\$	STR.
RIGHT\$	RI. RIG. RIGH. RIGHT.	TAN	TA.
		TIME	TI. TIM.
RND	RN.	VAL	V. VA.
SGN	SG.		
SIN	SI.		

## APPENDIX D: BATTERY REPLACEMENT

When replacing the batteries, these cautionary instructions will eliminate many problems:

- \* Always replace all four (4) batteries at the same time.
- \* Do not mix new batteries with used batteries.
- \* Use only: Dry battery (type AA, R6, or SUM-3, 1.5V)  $\times$  4

### Battery Replacement Procedure

Please follow this replacement procedure:

1. Turn off the computer by pressing the **OFF** key.
2. Remove the screw from the battery cover with a coin or a small screwdriver (See Figure D.1)
3. Replace the 4 batteries (Figure D.2)
4. Push the battery cover in slightly while replacing the screw.
5. To proceed, press the **ON** and **CL** keys, type **NEW** and press **ENTER** key.

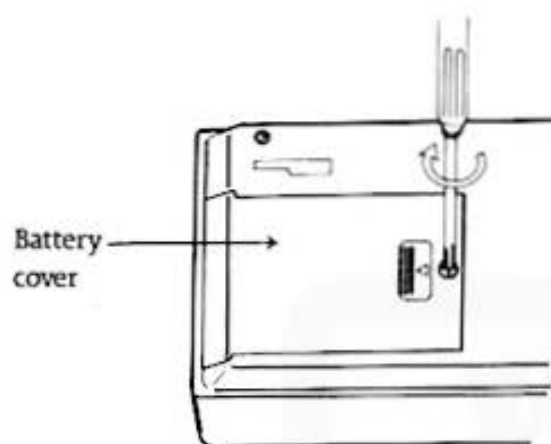


Figure D.1

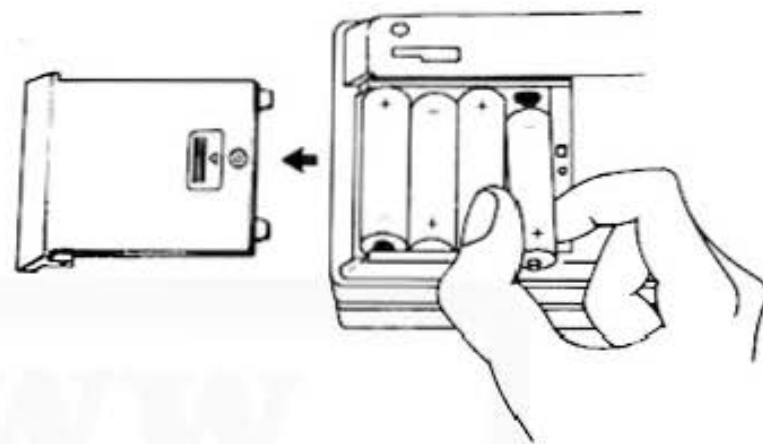
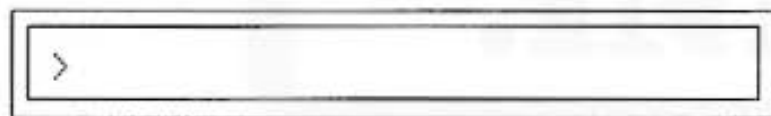


Figure D.2

6. Check the following display.



The prompt symbol is displayed

If the display is blank or displays any symbol other than ">", remove the batteries and install them again, and check the display.

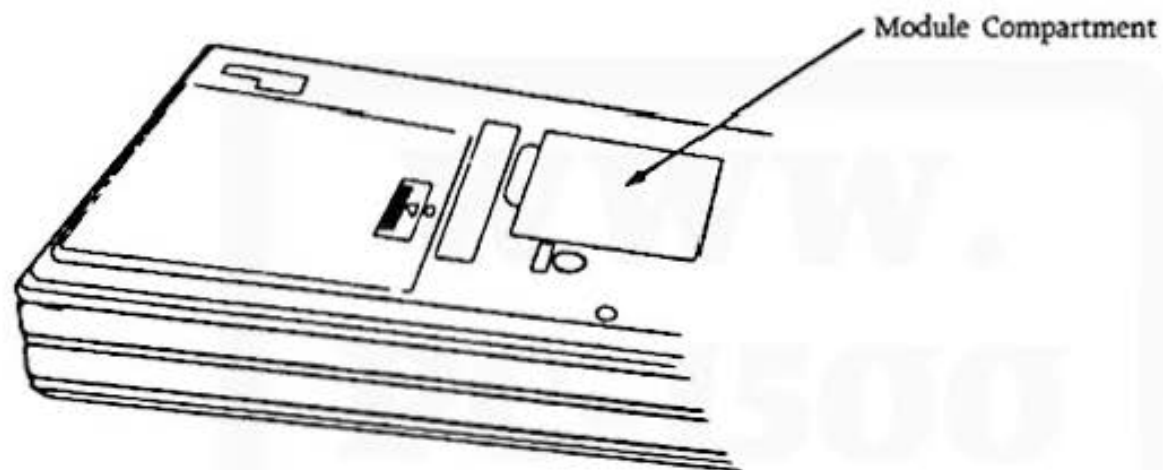
Note:

- \* If the computer will not be operated for an extended period of time, remove the batteries to avoid possible damage caused by battery leakage.
- \* Keeping a dead battery may result in damage to the computer due to solvent leakage of the battery. Remove a dead battery promptly.
- \* A rechargeable battery cannot be used in the PC-1500A.
- \* The AC adaptor, EA-150, for the CE-150 can also be used as an AC adaptor for the PC-1500A when it is used separate from the CE-150.

(Do not connect EA-150 to the PC-1500A computer when the PC-1500A is connected to the printer/cassette interface, CE-150.)

The module compartment is where an optional memory module is accommodated. Do not touch any connector inside the compartment. This may adversely affect the computer due to static electricity.








An optional memory module may be located inside the module compartment (found adjacent to the battery compartment). Do not touch any of the connections inside this compartment as this may adversely affect the computer because of static electricity.



**APPENDIX E: KEY CODE TABLE**

Hexa- decimal	0	1	2	3	4	5	6	7	8	9
0	(Null)		SPACE	0	@	P	E	p	DEF SPC	DEF P
1	SHIFT	<!>	!	1	A	Q	a	q	DEF A	DEF Q
2	↑↓ SML	<">	"	2	B	R	b	r	DEF B	DEF R
3	( )	<#>	#	3	C	S	c	s	DEF C	DEF S
4		<\$>	\$	4	D	T	d	t	DEF D	DEF T
5	( )	<%>	%	5	E	U	e	u	DEF E	DEF U
6		<&>	&	6	F	V	f	v	DEF F	DEF V
7	(BELL)		≡	7	G	W	g	w	DEF G	DEF W



8	 (BS)	 (CAN)	(	8	H	X	h	x	 H	 X
9	 (VT)		)	9	I	Y	i	y	 I	 Y
A	 (LF)	 (SUB)	*	:	J	Z	j	z	 J	 Z
B	 (VT)	 (ESC)	+	;	K	$\sqrt{\quad}$	k	{	 K	
C	 (FF)		,	<	L	¥	l	 	 L	
D	 (CR)		-	=	M	$\pi$	m	}	 M	 =
E		(RS)	.	>	N	^	n	~	 N	
F			/	?	O	—	o	■	 O	

## APPENDIX F: ERROR CODE LIST

### Error Code

### Explanation

#### Computer Related Errors

1. Syntax error: This error results from typing errors such as:  
missing information:  
10: GOTO  
or invalid commands.  
10: 5A = 1 or  
10: NEW  
(because NEW may not be a statement.)  
<Display> ERROR 1 IN 10
2. This error occurs when there is no FOR command corresponding to a NEXT command, or when there is no GOSUB command corresponding to a RETURN command.  
  
Ex. 10: FOR A = 1 TO 10  
.  
.  
.  
100: NEXT B.  
<Display> ERROR 2 IN 100

Do not sale this PDF !!!

4. This error occurs when there is no DATA corresponding to a READ command.

Ex.           10: READ X, Y  
              20: DATA 10  
              30: END  
<Display>   ERROR 4 IN 10

5. This error occurs when an array variable is declared with the name of an existing variable.

Ex.           10: DIM A (10,10)  
              20: DIM A (5)  
<Display>   ERROR 5 IN 20

6. This error occurs when an array variable has been used without a DIM (dimension) statement.

Ex.           10: CLEAR  
              20: A (3) = 1  
<Display>   ERROR 6 IN 20

7. This error occurs when the variable name is inappropriate.

Ex.           10: A\$ = 10 or  
              10: FOR A\$ = 1 TO 10  
<Display>   ERROR 7 IN 10

8. This error occurs when the dimension exceeds 3 in the declaration of array variable.

Ex.           10: DIM A (3,4,5,6)  
<Display>    ERROR 8 IN 10

9. This error occurs when the subscript number of an array variable exceeds the size of the array stated in the DIM command.

Ex.           10: DIM A(3)  
              20: A (4) = 1  
<Display>    ERROR 9 IN 20

10. This error occurs when there is not enough memory available to create more variables.

Ex.	<u>key operation</u>	<u>display</u>
	<b>MEM</b> <sup>(ENTER)</sup>	7
	<b>AD = 10</b> <sup>(ENTER)</sup>	ERROR 10

11. This error occurs when the specified line is not in the program.

Ex.           10: PRINT "X="; X: GOTO 5  
<Display>    ERROR 11 IN 10

12. This error occurs when the USING command specifies incorrect format specifications.

Ex.                   100: PRNT USING "###A#"; 10  
 <Display>       ERROR 12 IN 100

13. This error occurs when a program exceeds program memory capacity or when the Reserve key specification exceeds the reserve-area capacity.

Ex.	<u>key operation</u>	<u>display</u>
	<b>MEM</b> <b>ENTER</b>	7
	<b>15</b> <b>A = A + 1</b> <b>ENTER</b>	ERROR 13

14. (1) FOR statements have been nested too deeply and the stack capacity has been exceeded.  
 (2) While parsing an expression, buffer space has been exceeded.
15. (1) GOSUB statements are nested too deeply and the stack area has been exceeded.  
 (2) While parsing an expression, the string buffer size has been exceeded by the character strings handled.
16. (1) The value specified is over 1E100 or less than - 1E100.

Ex.                   123E 99

- (2) The value set by hexadecimals exceeds 65535.

Ex.                   &1FFAB

17. Data type (numerals, character strings) is inappropriate for calculation expression.  
Ex. **1 + "A" ENTER**
18. Number of arguments inappropriate for expression.  
Ex. **LEFT\$ ("ABC") ENTER**  
**SIN (30,60) ENTER**
19. Specified numeric value is outside the permitted range.  
Ex. 10: DIM A(256)  
<Display> **ERROR 19 IN 10**
20. When fixed memory array variables are specified, there is no '(' following '@' or '@\$'.  
Ex. 100: @\$ = "A"  
<Display> **ERROR 20 IN 100**
21. A variable is required in the expression.  
Ex. 10: FOR 1 = 0 TO 10  
<Display> **ERROR 21 IN 10**
22. This error occurs when the program is loaded, and there is no memory space available for loading.
23. This error occurs when the time is set incorrectly.  
Ex. **TIME = 131005.10 ENTER**

26. This error occurs when the command cannot be executed in the current mode.  
Ex.           <RUN MODE> **NEW** **(ENTER)**
27. This error occurs when an optional printer is not connected and there is no program which corresponds to the given label.  
Ex.           

	<u>key operation</u>	<u>display</u>
	<b>(DEF)</b> <b>I</b> <b>(ENTER)</b>	ERROR 27
28. This error occurs when a command or function code has been inserted inside " ", or when you try to substitute INPUT commands or AREAD commands for character variables.  
Ex.           10 INPUT A\$  
  

	<u>key operation</u>	<u>display</u>
	<b>(DEF)</b> <b>W</b> <b>(ENTER)</b>	ERROR 28
30. This error occurs when a line number exceeds 65539.  
(65280~65539:ERROR 1)  
Ex.           **102235 A=10 (ENTER)**
32. This error occurs when the graphic cursor is between columns 152~155 during execution of input commands (input code cannot be displayed)  
Ex.           100: GCURSOR 152  
              110: INPUT X  
              <Display> ERROR 32 IN 110

36. Data or characters cannot be displayed in accordance with the format specified by USING commands.

Ex. 10: USING "####.##"  
20: PRINT 12345

The integer section together with its sign has exceeded 4 digit spaces.

37. This error occurs in numeric calculations, when the calculation results have exceeded 9.999999999 E99.

38. This error occurs when division has occurred using 0 as the denominator.

Ex. 10: PRINT 5/0

39. This error occurs when an illogical calculation has been made:

* negative number logarithmic calculation	Ex. LN ( - 10)
* ASN, ACS in the case of X = 1	Ex. ASN (1.5)
	ACS (100)
* Square Root of negative numbers	Ex. SQR ( - 10)

177~181 During program creation, the program has overwritten the data area. Overlapping of these two areas occurred.

0, 224~241 During execution of INPUT commands or AREAD commands, incorrect input data are given.

Ex. 10: INPUT A

	<u>key operation</u>	<u>display</u>
123	PRINT <b>ENTER</b>	ERROR 240

Do not sale this PDF !!!



## Cassette Related Errors

- 40. Inappropriate specification for the expression.
- 41. SAVE and LOAD have been specified for the ROM area.
- 42. The cassette file data is too large and cannot be LOADED.
- 43. While verifying data using the CLOAD? command, the format of data to be loaded does not match the file format.
- 44. A CHECK SUM Error has occurred.

## Printer Related Commands

- 70. (1) The pen has exceeded the coordinate range of:  
 $-2048 \leq X, Y \leq 2047$   
(2) The pen will exceed the range upon execution of further commands.
- 71. (1) The paper has backed up more than 10.24 cm. in the TEXT mode.  
(2) The paper will back up more than 10.24 cm. upon execution of further commands (in TEXT mode)
- 72. The value given is inappropriate for the value of TAB.

73. A command has been used in the wrong printer mode (GRAPH/TEXT)
74. The number of commas (,) in LINE or RLINE command is too large.  
Note: Entry of over seven commas results in an error. Also, if the first comma is omitted, more than six would cause an error
76. For LPRINT, when the printing of calculation results cannot be done on one line (in TEXT mode)
78. (1) One or more pens are in the process of being changed.  
(2) The LOW BATTERY state has not been corrected. If ERROR 78 is due to LOW BATTERY state of the printer, turn OFF the CE-150. After recharging the printer, turn the CE-150 ON. You may now continue.
- This error occurs when, for either of these two reasons, commands that move the pen (such as LPRINT and LINE) are not able to be executed.
79. The color signal has not gone on. The color signal is for COLOR and goes on only when the pen comes to the left side. When the pen is in this position, it is possible to know the number of the present pen color position.
80. Low Battery. After recharging, immediately push the PC-1500A ON key again and start operation.

## APPENDIX G: FURTHER READING

### BASIC Programming

Albrecht, R.L., L. Finkel, and J. Brown. 1978. BASIC. 2nd. Ed. Wiley Interscience, New York, New York.

Coan, J. 1978. Basic BASIC. Hayden Book Co., Rochelle Park, New Jersey.

Cole, J. 1981. 50 Programs in BASIC for Home, School, and Office. ARCsoft Publishers, Woodsboro, Maryland.

Cole, J. 1981. 50 More Programs in BASIC for Home, School, and Office. ARCsoft Publishers, Woodsboro, Maryland.

Dwyer, T.A., and M. Critchfield. 1978. BASIC and the Personal Computer. Addison-Wesley Publishing, Co., Reading, Massachusetts.

Gilder, J. 1980. BASIC Computer Programs in Science and Engineering. Hayden Book Co., Rochelle Park, New Jersey.

Gottfried, B.S. 1975. Schaum's Outline Series: Theory and Problems of Programming with BASIC. McGraw-Hill Book Co., New York, New York.

Hubin, W. 1978. BASIC Programming for Scientists and Engineers. Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Koffman, E., and F. Friedman. 1979. Problem Solving and Structured Programming in BASIC. Addison-Wesley Publishing Co., Reading, Massachusetts.

- Lott, R.W. 1977. BASIC with Business Applications. John Wiley & Sons, New York, New York.
- Miller, A.R. 1981. BASIC Programs for Scientists and Engineers. SYBEX, Inc., Berkeley, California.
- Monro, D.M. 1978. Basic BASIC. Winthrop Publishers, Inc., Cambridge, Massachusetts.
- Nagin, P., and H. Ledgard. 1978. BASIC with Style: Programming Proverbs. Hayden Book Co., Rochelle Park, New Jersey.
- Poole, L. 1980. Practical BASIC Programs. OSBORNE/McGraw-Hill, Berkeley, California.
- Poole, L., and M. Borchers. 1979. Some Common BASIC Programs. 3rd. Ed. OSBORNE/McGraw-Hill, Berkeley, California.
- Ruckdeshel, F.R. 1981. BASIC Scientific Subroutines. Volume 1. Byte/McGraw-Hill, Peterborough, New Hampshire.
- Simon, D.E. 1978. BASIC From the Ground Up. Hayden Book Co., Rochelle Park, New Jersey.

## **General Reference**

- Behrendt, B.L. 1982. Pocket Magic. Micro Text Publications, Inc., New York, New York.
- Billings, K., and D. Moursund. 1979. Are You Computer Literate? Dilithium Press, Beaverton, Oregon.

- Cole, J. 1981. Murder in the Mansion and Other Computer Adventures. ARCsoft Publishers, Woodsboro, Maryland.
- Cole, J. 1982. Pocket Computer Programming Made Easy. ARCsoft Publishers, Woodsboro, Maryland.
- Cole, J. 1982. Practical PC-2/PC-1500 Pocket Computer Programs. ARCsoft Publishers, Woodsboro, Maryland.
- Cole, J. 1982. 99 Tips and Tricks for the New Pocket Computers. ARCsoft Publishers, Woodsboro, Maryland.
- Cole, J. 1982. 101 Pocket Computer Programming Tips and Tricks. ARCsoft Publishers, Woodsboro, Maryland.
- Cole, J. 1982. TRS-80 Pocket Computer Programs. Tandy Corporation, Fort Worth, Texas.
- Craig, J.C. 1982. 119 Practical Programs for the TRS-80 Pocket Computer. TAB Books, Blue Ridge Summit, Pennsylvania.
- Dorf, R.C. 1974. Computers and Man. Boyd & Fraser Publishing Co., San Francisco, California.
- Forsythe, G., M. Malcolm, and C. Moler. 1977. Computer Methods for Mathematical Computations. Prentice-Hall, Inc., Englewood Cliffs, New Jersey.
- Gear, C.W. 1974. Computer Organization and Programming. McGraw Hill, Inc., Englewood Cliffs, New Jersey.
- Inman, D., and J. Conlon. 1982. Problem Solving on the TRS-80 Pocket Computer. Tandy Corporation, Fort Worth, Texas.

- Leventhal, L.A. 1978. Introduction to Microprocessors: Software, Hardware, Programming. Prentice-Hall, Inc., Englewood Cliffs, New Jersey.
- Lewart, C. 1982. Science and Engineering Source Book. Micro Text Publications, Inc., New York, New York.
- Librach, H. 1982. Pocket Computer Primer. Micro Text Publications, Inc., New York, New York.
- Sharp Corporation. 1983. Sharp Pocket Computer PC-1500: Technical Reference Manual. Sharp Corporation, Osaka, Japan.
- Sharp, V.F. 1982. How to Solve Statistical Problems with your Pocket Computer. TAB Books, Blue Ridge Summit, Pennsylvania.
- Shelley, G.B., and T.J. Cashman. 1980. Introduction to Computers and Data Processing. Anaheim Publishing Co., Fullerton, California.
- Spencer, D.D. 1982. Programming the TRS-80 Pocket Computer. Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

## APPENDIX H: ORDER OF EXPRESSION EVALUATION

Calculations are performed in accordance with the following hierarchy: expressions in parentheses having the highest priority and logical operations having the lowest. If two or more operations of the same priority are found in the same expression or sub-expression, they are evaluated from left to right.

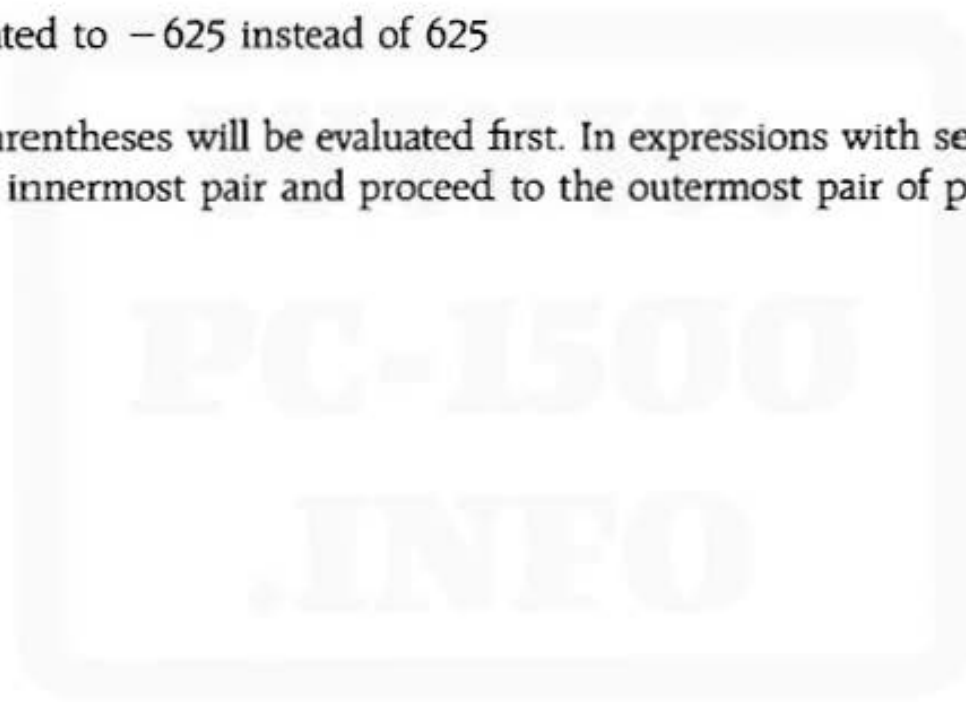
- 1) Expressions in Parentheses
- 2) Retrieval of values from variables, TIME, PI, MEM, INKEY\$
- 3) Functions (SIN, COS, LOG, EXP, etc.)
- 4) Exponentiation (Example:  $2A \wedge 3 = 2*(A \wedge 3)$ )
- 5) Arithmetic Sign (+, -)
- 6) Multiplication, Division (\*, /)
- 7) Addition, Subtraction (+, -)
- 8) Comparison Operators (<, <=, =, >=, >, <>)
- 9) Logical Operators (AND, OR, NOT)

#### NOTES:

When both arithmetic sign and exponents are used in the same expression, the exponent is evaluated before the sign.

Example:  $-5 \wedge 4$  is evaluated to  $-625$  instead of  $625$

Calculations within parentheses will be evaluated first. In expressions with several "layers" of parentheses, calculations start with the innermost pair and proceed to the outermost pair of parentheses.





## Sample Evaluation

$$7 \wedge 2 + 3 * \sqrt{144} / \sqrt{81} + \underbrace{\text{SIN}(120 + 150)} * -3$$

$$7 \wedge 2 + 3 * \sqrt{144} / \sqrt{81} + \underbrace{\sin(270)} * 3$$

$$\underbrace{7 \wedge 2} + 3 * \quad 12 \quad / \quad 9 \quad + \quad \quad -1 \quad * -3$$

$$49 + \underbrace{3 * 12 / 9} + \quad -1 * -3$$

$$49 + \underbrace{36 / 9} + \quad 3$$

$$\underbrace{49 + 4} + \quad 3$$

$$\underbrace{53 + 3}$$

$$56$$

## (CALCULATION RANGE)

Functions	Dynamic range
$y \wedge x (y^x)$	$-1 \times 10^{100} < x \log y < 100$ $\left( \begin{array}{l} y = 0, x \leq 0: \text{ERROR 39} \\ y = 0, x > 0: 0 \\ y < 0, \neq \text{integer: ERROR 39} \end{array} \right)$ see examples p. 278.
SIN x COSx TANx	$\left. \begin{array}{l} \text{DEG: }  x  < \times 10^{10} \\ \text{RAD: }  x  < \frac{\pi}{180} \times 10^{10} \\ \text{GRAD: }  x  < \frac{10}{9} \times 10^{10} \end{array} \right\}$ In TANx, however, the following cases are excluded. DEG: $ x  = 90(2n-1)$ RAD: $ x  = \frac{\pi}{2}(2n-1)$ GRAD: $ x  = 100(2n-1)(n: \text{integer})$
$\text{SIN}^{-1}x$ $\text{COS}^{-1}x$	$-1 \leq x \leq 1$
$\text{TAN}^{-1}x$	$ x  < 1 \times 10^{100}$
LNx LOGx	$1 \times 10^{-99} \leq x < 1 \times 10^{100}$
EXPx	$-1 \times 10^{100} < x \leq 230.2585092$
$\sqrt{x}$	$0 \leq x < 1 \times 10^{100}$

Functions other than shown above can be calculated only when  $x$  stays within the following range.

$$1 \times 10^{-99} \leq |x| < 1 \times 10^{100} \text{ and } 0$$

(Ex.)  $0 \wedge 0$  **ENTER** → ERROR 39  
 $0 \wedge 5$  **ENTER** → 0  
 $(-4) \wedge 0.5$  **ENTER** → ERROR 39  
 $-4 \wedge 0.5$  **ENTER** → -2

- As a rule, the error of functional calculations is less than  $\pm 1$  at the lowest digit of a display numerical value (at the lowest digit of mantissa in the case of scientific notation system) within the above calculation range.

## APPENDIX I: COMMAND REFERENCE TABLE

### 1. Functions

Function	Abbreviations	Remarks
ABS	AB.	Absolute value
ACS	AC.	$\cos^{-1}$
AND	AN.	exp. AND exp. [logical And]
ASC	ASC	Converts characters to ASCII code ASC "character" character variable
ASN	AS.	$\sin^{-1}$
ATN	AT.	$\tan^{-1}$
CHR\$	CH. CHR.	Converts ASCII code to characters CHR\$ ASCII decimal code numeric value
COS	COS	
DEG		Converts decimal degrees to degrees, minutes, seconds.
DMS	DM.	Converts degrees, minutes, seconds to decimal degrees.
EXP	EX.	$e^x$

INKEY\$	INK. INKE. INKEY.	Character variable = INKEY\$ If a key is pushed during execution of INKEY\$ command, the ASCII character will be read into the character variable.
INT		Truncates value to integer <b>INT (10/3) (ENTER) &lt; Display &gt; 3</b>
LEFT\$	LEF. LEFT.	LEFT\$ (character variable, numeric expression) Takes the specified number of characters from the left side of the specified character string.
LEN		LEN "character" character variable Seeks the character count of the specified character string.
LOG	LO.	$\log_{10}X$
LN		$\log_e X$
MEM	M. ME.	Displays the remaining number of steps available in memory. Same as STATUS 0.
MID\$	MI. MID.	MID\$ (character variable, numeric exp 1, numeric exp 2); Takes character(s) from the middle of the specified character string.
NOT	NO.	NOT exp. [logical Negation]
OR		exp. OR exp. [logical Or]

$\pi$ (PI)		Specifies ratio of circumference. (= 3.141592654)
POINT	POI. POIN.	Returns a number which represents the pattern of activated dots within the given column. POINT numeric expression.
RIGHT\$	RI. RIG. RIGH. RIGHT.	RIGHT\$ (character variable, numeric exp) Takes the specified number of characters from the right side of the specified character string.
RND	RN.	RND expression Commands to generate random numbers.
SGN	SG.	Signum functions
SIN	SI.	sin
$\sqrt{\quad}$ (SQR)	SQ.	square root
STATUS	STA. STAT. STATU.	STATUS 0 or 1 (0) Number of program steps available. (1) Number of program steps already used.
STR\$	STR.	STR\$ numeric expression Converts numerals to character string.
TAN	TA.	tan

TIME	TL. TIM.	(1) TIME = month, day, hour, minute, second Time function sets month, day, hour, minute, and second. (2) TIME [calls up date and time]
VAL (value)	V. Va.	VAL { "character" character variable } Converts character strings to numerals.
^		Powers

## 2. Statements

Command	Abbreviation	Remarks
AREAD (auto read)	A. AR. ARE. AREA.	AREAD variable When executing programs by defined keys, AREAD enters display content into specified variables.
ARUN (auto run)	ARU.	ARUN Command to automatically start program execution when the PC-1500A is switched on.

BEEP	B. BE. BEE.	BEEP exp1. exp2. exp3 Sound command. Turns sound generating functions ON/OFF. specifies loudness and length of sounds.
CLEAR	CL. CLE. CLEA.	Command to clear all data (variables)
CLS (clears)		Display clear command. Erases display.
CURSOR	CU. CUR. CURS. CURSO.	(1) CURSOR exp. ( $0 \leq \text{exp} \leq 25$ ) specification of starting position of display. (2) CURSOR cancels previous specification.
DATA	DA. DAT	DATA exp. exp. ... Data to be read in using the READ command.
DEGREE	DE. DEG. DEGR. DEGRE.	Angular mode specification. Degree ( $[\circ]$ ) is designated.



DIM (dimension)	D. DI.	(1) DIM variable name (exp) (2) DIM variable name (exp) * exp3 (3) DIM variable name (exp1, exp2) variable names: A, B, C\$, D\$, etc. ... ( ): specifies size and dimension of the array exp3: digit space specification.
END	E. EN.	Program ending command.
FOR  TO  STEP	E. FO.   STE.	(1) For numerical variable = exp1 to exp2 Start of FOR-NEXT loop. Used in correspondence with NEXT command. (2) FOR numerical variable = exp1 to exp2 STEP exp3 exp1: initial expression. exp2: final value. exp3: interval to increase with each loop.
GCURSOR (graphic cursor)	GC. GCU. GCUR. GCURS. GCURSO.	Specifies display position by dot units. GCURSOR expression ( $0 \leq \text{expression} \leq 155$ ) or ( $\&0 \leq \text{expression} \leq \&9B$ )

GOSUB	GOS. GOSU.	$\text{GOSUB} \left\{ \begin{array}{l} \text{expression} \\ \text{"character"} \\ \text{character variable} \end{array} \right\}$ <p>Subroutine jump commands. Moves execution to specified line or label. Statements at this point are executed as a subroutine. Used in correspondence with RETURN command.</p>
GOTO	B. GO. GOT.	<p>Jump commands.</p> $\text{GOTO} \left\{ \begin{array}{l} \text{expression} \\ \text{"character"} \\ \text{character variable} \end{array} \right\}$ <p>Moves execution to specified line or label.</p>
GPRINT (graphic print)	GP. GPR. GPRI. GPRIN.	<p>Displays on printer content given on display.</p> <p>(1) GPRINT "00 00 00 ..." (inside " " are hexadecimal numbers)</p> <p>(2) GPRINT 0; 0; ...</p> <p>(3) GPRINT &amp; 0; &amp; 0; ...</p>
GRAD	GR. GRA.	<p>Angular mode designation. Gradient ([g]) is designated.</p>

IF		<p>(1) IF conditional expression execution command.</p> <p>(2) IF arithmetic expression execution command.</p> <p>Evaluates given conditions and either moves execution to the next line or executes command.</p>
INPUT	I. IN. INP. INPU.	<p>(1) INPUT variable, variable, ...</p> <p>(2) INPUT "character", variable, "character", variable</p> <p>(3) INPUT "character"; variable, "character"; variable</p>
LET	LE.	<p>(1) LET numeric variable = expression</p> <p>(2) LET character variable = "characters"</p> <p>(3) LET character variable = character variable.</p> <p>LET follows IF commands. It can be omitted in other cases.</p>
LOCK	LOC.	Locks the MODE the computer is in.
NEXT	N. NE. NEX.	<p>NEXT numeric variable</p> <p>Shows the very end of the FOR-NEXT loop.</p> <p>The numeric variable must be the same as the numeric variable following the FOR command.</p>
ON ERROR	O. ER. ERR. ERRO.	<p>ON ERROR GOTO expression</p> <p>Error disposal command.</p>

ON GOSUB	O. GOS. GOSU.	ON expression GOSUB exp1. exp2. exp3 Subroutine jump command. Specifies the place to jump to (exp1. exp2. exp3) by the value of expression.
ON GOTO	O. G. GO. GOT.	On expression GOTO exp1. exp2. exp3 Jump command. Specifies the place to jump to (exp1. exp2. exp3) by the value of the expression following ON.
PAUSE	PA. PAU. PAUS.	Same form as PRINT command. Displays the specified content for about 0.85 seconds, then executes program.
PRINT	P. PR. PRI. PRIN.	<div> <div>(1) PRINT</div> <div> <div>expression</div> <div>"character"</div> <div>character variable</div> </div> <div>}</div> <div>:</div> </div> <div> <div>(2) PRINT</div> <div> <div>expression</div> <div>"character"</div> <div>character variable</div> </div> <div>}</div> <div>.</div> <div> <div>expression</div> <div>"character"</div> <div>character</div> <div>variable</div> </div> <div>}</div> </div>

		<p>(3) PRINT { expression "character" character variable } ; { expression "character" character variable }</p> <p>; ... ; { expression "character" character variable } ;</p>
RADIAN	RAD. RADI. RADIA.	Angular mode designation. Radian ([rad]) is designated.
RANDOM	RA. RAN. RAND. RANDO.	Place the seeds of random numbers prior to the use of RND commands.
READ	REA.	READ variable, variable, ... Data read in command. Enters data from DATA statements into specified variables.
REM (remark)		REM... document notes... Specify remarks not to be executed.

RESTORE	RES. REST. RESTO. RESTOR.	(1) RESTORE expression Changes the order of data read in by READ commands. (2) RESTORE Start at beginning of first DATA statement.
RETURN	RE. RET. RETU. RETUR.	Return to continue execution after GOSUB statement that invoked this subroutine.
STOP	S. ST. STO.	Command to halt program execution.
THEN	T TH. THE.	Defines execution command for IF statement. Jump commands are only possible to be defined as execution commands for IF command.  THEN { expression "character" character variable }
TRON (trace on)	TR. TRO.	Specifies the mode to perform debugging.
TROFF (trace off)	TROF.	Cancels the mode for performing debugging.

UNLOCK	UN. UNL. UNLO. UNLOC.	Cancels LOCK mode.
USING	U. US. USI. USIN.	(1) USING "###.### ^ " (2) USING "&&& ..... &&&" (3) PRINT USING "Format"; .... (4) USING (5) PRINT USING; Specifies display position by the value of the expression.
WAIT	W. WA. WAI.	WAIT expression ( $0 < = \text{expression} < = 65535$ ) Specifies time duration of display when using PRINT commands. WAIT with no argument cancels the previous specification (duration = infinity)

## 3. Commands

Command	Abbreviation	Remarks
CONT (continue)	C. CO. CON.	Restarts execution of programs that have been temporarily halted. Effective in the RUN mode.
LIST	L. LI. LIS.	Command that performs program listing. Effective in the PRO mode.
NEW		(1) NEW    (2) NEWØ In the PRO mode, both NEW and NEWØ clear the program of all variables. In the RESERVE mode, NEW will clear all memory (including the softkey buffer); NEWØ will not clear all memory (including the softkey buffer) in this mode.
RUN	R. RU.	(1) RUN (2) RUN expression (3) RUN "character" character variable Effective in RUN mode.



## 4. Cassette Commands

Command	Abbreviation	Remarks
CHAIN	CHA. CHAI.	Transmission commands. When used in the middle of a program, reads in programs from the tape (transmits) and executes the program.  (1) CHAIN "filename" (CHAIN-1 "filename")  (2) CHAIN "filename", expression (CHAIN-1 "filename", expression)  (3) abbreviated forms of the "filename" for (1) and (2)
CLOAD (cassette load)	CLO. CLOA.	Transmission command. This command transmits program or reserve content from the tape to the computer memory.  (1) CLOAD (CLOAD-1)  (2) CLOAD "filename" (CLOAD-1 "filename")

CLOAD? (cassette load?)	CLO.? CLOA.?	Comparison command. This command compares the program in the memory or the reserve content with the content recorded on tape. (1) CLOAD? (CLOAD?-1) (2) CLOAD? "filename" (CLOAD?-1 "filename")
CSAVE (cassette save)	CS. CSA. CSAV.	This command records on tape the content of program and reserve memory. (1) CSAVE (CSAVE-1) (2) CSAVE "filename" (CSAVE-1 "filename")
INPUT#	I.# IN.# INP.# INPU.#	Data transmission command. This command transmits data recorded on tape into the specified variables. Takes same form as PRINT # command.
MERGE	MER. MERG.	Transmission command. This command transmits programs from tape to the computer. Takes same form as CLOAD command. In this command, previously recorded programs will be retained as they are and programs newly read in will be added.

PRINT #	P.# PR.# PRI.# PRIN.#	Data recording command. This command records onto tape the data stored in the PC-1500A. (1) PRINT # variable name, variable name, ... (PRINT # - 1, variable name, variable name....) (2) PRINT # "filename"; variable name, ... (PRINT # - 1, "filename"; variable name, ...)
RMT OFF (remote off)	RM.OF RMTOF	This command cancels the remote function of REM 1 terminal. (For second tape recorder)
RMT ON (remote on)	RM.O RMTO	This command resets the remote function of REM 1 terminal. (For second tape recorder)

## 5. Printer Commands

Command	Abbreviation	Remarks
COLOR	COL. COLO.	Specifies color of characters. COLOR expression ( $0 < = \text{expression} < = 3$ )
CSIZE (character size)	CSI. CSIZ.	Specifies size of the characters to be printed. CSIZE expression ( $1 < = \text{expression} < = 9$ )

GLCURSOR (graphic line cursor)	GL. GLC. GLCU. GLCUR. GLCURS. GLCURSO.	Command that moves the pen position from the starting point to an X, Y coordinate. Valid for GRAPH mode only. GLCURSOR (exp1, exp2)
GRAPH (graphic)	GRAP.	This mode is used to draw graphs and illustrations.
LCURSOR (line cursor)	LCU. LCUR. LCURS. LCURSO.	Moves pen to desired position on printer.
LF (line feed)		Performs paper feed as far as number of line feeds shown by the expression. Valid for TEXT mode only. LF expression

LINE	LIN.	<p>Line drawing commands.</p> <p>Valid for GRAPH mode only.</p> <p>(1) LINE (exp1, exp2) – (exp3, exp4)</p> <p>(2) LINE (exp1, exp2) – (exp3, exp4) exp5, exp6</p> <p>(3) LINE (exp1, exp2) – (exp3, exp4) exp5, exp6, B</p> <p>where: exp5 specifies line type exp6 specifies color B specifies a box drawing</p> <p>(4) LINE (exp1, exp2) – (exp3, exp4) – ... ...(exp11, exp12)</p>
LLIST	LL. LLI. LLIS.	List program.
LPRINT	LP. LPR. LPRI. LPRIN.	<p>Prints specified content.</p> <p>Valid for TEXT mode only. Takes same form as PRINT commands.</p>
RLINE (relative line)	RL. RLI. RLIN.	<p>Command that draws lines with the pen position as the starting point.</p> <p>Valid for GRAPH mode only. Takes same form as LINE commands.</p>

ROTATE (rotator)	RO. ROT. ROTA. ROTAT.	Specifies direction of characters to be printed (print direction). Valid for GRAPH mode only.  ROTATE expression ( $0 \leq \text{expression} \leq 3$ )
SORGN (set origin)	SO SOR. SORG.	This command specifies the present pen position as the new starting point (point of origin). Valid for GRAPH mode only.
TAB		Specifies pen position. Valid for TEXT mode only. (1) TAB expression (2) LPRINT TAB expression: ...
TEST	TE. TES.	Color check. When executed, test draws a 5mm by 5mm square in each color.
TEXT	TEX.	TEXT mode specification. This mode prints characters and numerals.

## APPENDIX J: HEXADECIMAL NUMBERS AND BOOLEAN FUNCTIONS

### Hexadecimal Numbers

The PC-1500A provides the capability to use a hexadecimal (base 16) number within any expression in which a decimal number may be used. Hexadecimal numbers are distinguished from decimal numbers by preceding them with an & (ampersand). The following are examples of valid hexadecimal numbers:

&16    &F    &7ECA    &08    &99A    &-5B

Hexadecimal numbers may be used in calculations:

10 + &A **ENTER**

RUN		I	.
		20	

Or within programs:

```
35 GPRINT &F, 54, &3E
40 DATA 67, &7F, &2B, 12, 305
```

## AND Function

The AND function provides a boolean AND of the internal representation of two values. The values must be in the range - 32768 through 32767. Numbers which exceed this range will cause an ERROR 19.

Examples:

<b>10 AND &amp;F</b>	<b>ENTER</b>	10
<b>1 AND 0</b>	<b>ENTER</b>	0
<b>- 1 AND 1</b>	<b>ENTER</b>	1
<b>55 AND 64</b>	<b>ENTER</b>	0
<b>16 AND 63</b>	<b>ENTER</b>	16

## OR Function

The OR function performs a boolean OR on the internal representations of two values. The values must be in the range - 32768 through 32767. Numbers which exceed this range will cause an ERROR 19.

Examples:

<b>10 OR &amp;F</b>	<b>ENTER</b>	15
<b>1 OR 0</b>	<b>ENTER</b>	1
<b>- 1 OR 1</b>	<b>ENTER</b>	- 1
<b>55 OR 64</b>	<b>ENTER</b>	119
<b>16 OR 63</b>	<b>ENTER</b>	63



## NOT Function

The NOT function returns the boolean NOT, or complement, of the internal representation of a single value. The value must be in the range  $-32768$  through  $32767$ . If the value exceeds this range an ERROR 19 will occur

Examples:

**NOT 0**

**ENTER** - 1

**NOT &F**

**ENTER** - 16

**NOT 55**

**ENTER** - 56

**NOT 1**

**ENTER** - 2

**NOT - 2**

**ENTER** 1

**APPENDIX K: DIFFERENCES BETWEEN PC-1500A AND PC-1500**

Because the PC-1500A provides more RAM than the PC-1500, some of the descriptions of the chip select signal, memory map, and 40-pin connector (for memory modules) found in the SHARP Pocket Computer PC-1500 Technical Reference Manual require modifications to suit the PC-1500A. A summary of the differences between the PC-1500A and the PC-1500 is given below. Take notice of these differences when using machine language.

**1. Chip Select Signal**

The methods of receiving signals (S0 – S5) are different. (Technical reference manual: page 93)

TC40H138F Output	Memory address	PC-1500	PC-1500A
$\overline{Y0}$	4000H 47FFH	S0 STANDARD USER MEMORY	S0  STANDARD USER MEMORY
$\overline{Y1}$	4800H 4FFFH	S1	
$\overline{Y2}$	5000H 57FFH	S2	
$\overline{Y3}$	5880H 5FFFH	S3 OPTIONAL USER MEMORY	S1  S2 OPTIONAL USER MEMORY  S3
$\overline{Y4}$	6000H 67FFH	S4	
$\overline{Y5}$	6800H 6FFFH	S5	
$\overline{Y6}$	7000H 77FFH	INHIBITED	INHIBITED
$\overline{Y7}$	7800H	7600H STANDARD USER AND SYSTEM MEMORY	7600H STANDARD USER AND SYSTEM MEMORY
	7FFFH	7BFFH  INHIBITED	7C00H  MACHINE LANGUAGE MEMORY

## 2. Memory map

If a memory module, the CE-151 or CE-155, is installed in the PC-1500A, the user memory area is increased accordingly. (Technical reference manual: page 95 ~ 96)

When the CE-151 is used.

3800H	
4000H	STANDARD USER MEMORY (RAM) 6KB
5800H	CE-151 (RAM) 4KB
6800H	NOT USED
7000H	
7600H	STANDARD USER AND SYSTEM MEMORY
7C00H	MACHINE LANGUAGE MEMORY
8000H	← ME 0 →

When the CE-155 is used.

3800H	CE-155 (RAM) 2KB	8KB IN TOTAL
4000H	STANDARD USER MEMORY (RAM) 6KB	
5800H	CE-155 (RAM) 6KB	
6800H		
7000H		
7600H	STANDARD USER AND SYSTEM MEMORY	
7C00H	MACHINE LANGUAGE MEMORY	
8000H	← ME 0 →	

## 3. 40-pin Connector

The signals names and descriptions of pin 5 as well as pins 16 ~ 18 differ on the 40-pin connector used to mount the memory module. (Technical reference manual: page 102)

Pin no.	Signal name	Description
5	S1	Address designation of 5800H ~ 5FFFH
16	S2	Address designation of 6000H ~ 67FFH
17	S3	Address designation of 6800H ~ 6FFFH
18	S4	Not connected

## INDEX

- Abbreviations, 114, 247-254  
ABS, 57, 253, 279  
ACS, 76, 253, 279  
Alphabetic Keys, 10, 26, 29-30, 37  
Ampersand (&), 31, 298  
AND, 253, 274, 279, 299  
AREAD, 163-164, 249, 282  
Arithmetic hierarchy, 45  
Arrays, 147-148  
ARUN, 164, 249, 282  
ASC, 154-157, 253, 279  
ASN, 76, 253, 279  
ATN, 76, 253, 279  
At sign, @, 31, 32, 116  
BEEP, 107-108, 165-167, 249, 283  
BEEP OFF, 108, 283  
BEEP ON, 108, 283  
CA, 9, 28  
Calculation Keys, 11, 26, 32  
CHAIN, 211, 248, 292  
CHR\$, 155, 253, 279  
CLEAR, 29, 249, 283  
Clear key, 4, 9, 16, 28-29, 33  
CLOAD, 37, 207-208, 248, 292  
CLOAD?, 208, 248, 293  
CLS, 167-168, 249, 283  
COLOR, 202-205, 219, 223, 233, 247, 269, 294  
Concatenation ( + ), 53, 157  
CONT, 27, 123-124, 188-189, 252, 291  
COS, 75, 253, 274, 277, 279  
CSAVE, 37, 207, 231, 248, 293  
CSIZE, 228, 247, 294  
CURSOR, 168-169, 249, 283  
DATA, 158-159, 250, 283  
Debugging, 184-189  
DEG, 38-39, 72, 253, 279  
DEGREE, 70, 250, 283  
DIM, 149-150, 250, 284  
Display Control Keys, 11, 26, 33-35  
DMS, 71, 253, 279  
Editing, 5, 16, 33-35, 179  
END, 112, 250, 284

- 304 EXP, 65-66, 253, 274, 277, 279  
 FOR...TO...STEP, 135-137, 250, 284  
 GCURSOR, 170-171, 250, 284  
 GLCURSOR, 235-237, 247, 295  
 GOSUB, 132-135, 138-142, 250, 285  
 GOTO, 37, 111-112, 128-132, 250, 285  
 GPRINT, 172-175, 250, 285  
 GRAD, 38-39, 71, 250, 285  
 GRAPH, 220-221, 223-224, 231-234, 247, 295  
 Hexadecimal (&), 298-300  
 IF...THEN, 126-128, 250, 286  
 INKEY\$, 143, 161, 253, 274, 280  
 INPUT, 37, 105-107, 114, 249, 251, 286  
 INPUT#, 213, 293  
 INT, 58, 253, 280  
 LCURSOR, 225, 228, 229, 247, 295  
 LEFT\$, 151, 253, 280  
 LEN, 150, 253, 280  
 LET, 110, 251, 286  
 LF, 229, 247, 295  
 LINE, 219, 238-240, 241, 247, 296  
 Line numbers, 24, 27, 34-35, 37, 85-86  
 LIST, 37, 123, 252, 291  
 LLIST, 219, 223, 247, 296  
 LN, 68-69, 253, 277, 280  
 LOCK, 189, 251, 286  
 LOG, 15, 19, 69-70, 253, 274, 277, 280  
 Logic operations, 11, 21-22  
 Logical Operator Keys, 11, 26, 33  
 Lower case, 29  
 LPRINT, 219, 224-228, 229, 248, 296  
 MEM, 117, 253, 274, 280  
 MERGE, 37, 208-211, 249, 293  
 MID\$, 152, 253, 280  
 Mode key, 9, 24, 28, 29, 189  
 Multiple statements, 113  
 NEW, 28-29, 86, 252, 291  
 NEWØ, 4, 42, 291  
 NEXT, 251, 286  
 NOT, 253, 274, 280, 300  
 Numeric Keys, 10, 30, 31, 32  
 ON, 3, 9, 27, 42, 251  
 ON...ERROR, 142, 286  
 ON...GOSUB, 138-141, 287  
 ON...GOTO, 141-142, 287  
 Operational Keys, 9, 26-29

OR, 254, 274, 280, 299  
Punctuation Keys, 10, 26, 31, 32  
PAUSE, 104, 114, 251, 287  
PI ( $\pi$ ), 32, 66-67, 254, 274, 281  
POINT, 175-177, 254, 281  
Power on/off, 3-4, 27  
Preassigned Keyword Keys, 12, 26, 37ff, 162-163  
PRINT, 37, 101-103, 109, 114, 251, 287  
PRINT#, 212, 249, 294  
PROGRAM mode, 9, 24, 28, 29, 35  
RADIANT, 70, 251, 288  
RANDOM, 67, 251, 288  
Range of numbers, 49  
READ, 251, 288  
REM, 99-100, 251, 288  
Remote off, 294  
Remote on, 294  
Reserve Keys, 12, 26, 31, 36, 115  
RESERVE mode, 9, 29, 36, 115,  
RESTORE, 160, 251, 289  
RETURN, 37, 251, 289  
RIGHT\$, 153, 254, 281  
RLINE, 241-243, 248, 296  
RMT OFF, 215, 249, 294  
RMT ON, 215, 249, 294  
RND, 254, 281  
ROTATE, 232, 248, 297  
RUN, 24, 39, 114, 252, 291  
RUN mode, 9, 24, 28, 36  
Scientific notation, 48-49  
SGN, 59-60, 254, 281  
SIN, 16, 75, 254, 274, 281  
SORGN, 234-235, 248, 297  
SQR, 32, 61-62, 254, 281  
Square root, 32, 61-62, 281  
Standard number format, 47-48  
STATUS, 117, 254, 281  
STEP, 252  
STOP, 123, 188-189, 252, 289  
STR\$, 118, 254, 281  
TAB, 229, 248, 297  
TAN, 76, 254, 277, 281  
TEST, 233, 248, 297  
TEXT, 220-221, 223-224, 228-229, 231, 233, 248, 297  
THEN, 252, 289  
TIME, 54-55, 144-146, 254, 274, 282  
TROFF, 184, 252, 289  
TRON, 184-187, 252, 289

306 UNLOCK, 189, 252, 290  
USING, 37, 178-183, 252, 290  
VAL, 118, 254, 282  
Variables, 17-20, 28, 50-53  
Variables, string, 19-20, 51-53  
WAIT, 109, 252, 290





**SERVICE CENTER ADDRESS** All and more about Sharp PC-1500 at <http://www.PC-1500.info>

**SHARP ELECTRONICS CORPORATION**

9 Sharp Plaza, Paramus, New Jersey 07652  
(201) 265-5600

**SHARP ELECTRONICS CORPORATION**

6478 I-85 Norcross, Georgia 30093  
(404) 448-5230

**SHARP ELECTRONICS CORPORATION**

430 East Plainfield Road, Countryside, Illinois 60525  
(312) 482-9292

**SHARP ELECTRONICS CORPORATION**

Sharp Plaza  
20600 South Alameda St., Carson California 90810  
(213) 637-9488

**SHARP ELECTRONICS CORPORATION**

1205 Executive Drive, East Richardson, Texas. 75081  
(214) 234-1136

NOTE: Whether or not your unit is still in warranty, you should return it together with all peripherals, 1 if connected to or installed in the PC-1500A when the fault condition (defect) was observed. In this way, we can verify total system performance. Send your equipment to the closest SHARP FACTORY SERVICE CENTER listed above:

1. Applies only to items manufactured by Sharp Electronics Corp.



All and more about Sharp PC-1500 at <http://www.PC-1500.info>



Do not sale this PDF !!!