SHARP

POCKET COMPUTER

MODEL **PC-1260**
PC-1261

INSTRUCTION MANUAL

MODEL PC-1260/PC-1261 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 warrantied 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.

TABLE OF CONTENTS

INTRODUCTORY NOTE

Welcome to the world of SHARP owners!

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

Your new SHARP PC-1260/1261 was designed to bring you all of the latest state of the art features of this computing revolution. As one of the most sophisticated hand held computers in the world today it incorporates many advanced capabilities:

* MEMORY SAFE GUARD—the PC-1260/1261 remembers stored programs and variables even when you turn it off.
* Battery powered operation for true portability.
* AUTO POWER OFF function which conserves the batteries by turning the power off if no activity takes place within a specified time limit.
* Programmable functions which allow the PC-1260/1261 to be used as a "smart" calculator.
* An expanded version of BASIC which provides formatted output, two-dimensional arrays, variable length strings, program chaining and many other advanced features.
* EASY SIMULATION PROGRAM function is used for changing individual values in a continuing programmed equation for fast calculation.
* HELP FUNCTION is used if you get lost and need an explanation of your error or if you just want a display of control data and operating procedures
  (1) The Reference Function displays BASIC commands and their correct usage.
  (2) The Character Code Table Function displays character codes used in hexadecimal notation.
  (3) The Error Message Function explains the type of error and locates it on the display with the cursor.
* An optional printer/microcassette recorder (Model CE-125) for long term storage and hard copy printout of programs and data.

Congratulations on entering an exciting and enjoyable new world. We are sure that you will find this purchase one of the wisest you have ever made. The SHARP PC-1260/1261 is a powerful tool, designed to meet your specific mathematical, scientific, engineering, business and personal computing needs. With the SHARP PC-1260/1261 you can begin NOW providing the solutions you'll need tomorrow!

CHAPTER 1
HOW TO USE THIS MANUAL

This manual is designed to introduce you to the capabilities and features of your PC-1260/1261 and to serve as a valuable reference tool. Whether you are a "first time user" or an "old hand" with computers, you should acquaint yourself with the PC-1260/1261 by reading and working through Chapters 2 through 6.

- Chapter 2 describes the physical features of the PC-1260/1261.
- Chapter 3 demonstrates the use of the PC-1260/1261 as a calculator.
- Chapter 4 defines some terms and concepts which are essential for BASIC programming, and tells you about the special considerations of these concepts on the PC-1260/1261.
- Chapter 5 introduces you to BASIC programming on the PC-1260/1261, showing you how to enter, correct, and run programs.
- Chapter 6 discusses some shortcuts that make using your new computer easier and more enjoyable.

Experienced BASIC programmers may then read through Chapter 8 to learn the specific features of BASIC as implemented on the PC-1260/1261. Since every dialect of BASIC is somewhat different, read through this material at least once before starting serious programming.

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

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

The remainder of the manual consists of:

- Chapter 7 —Basic information on the optional CE-125 Printer/Microcassette Recorder and other options.
- Chapter 9 —Description for the use of the easy simulation program.
- Chapter 10—Description for the use of the help function.
- Chapter 11—A troubleshooting guide to help you solve some operating and programming problems.
- Chapter 12—The care and maintenance of your new computer.

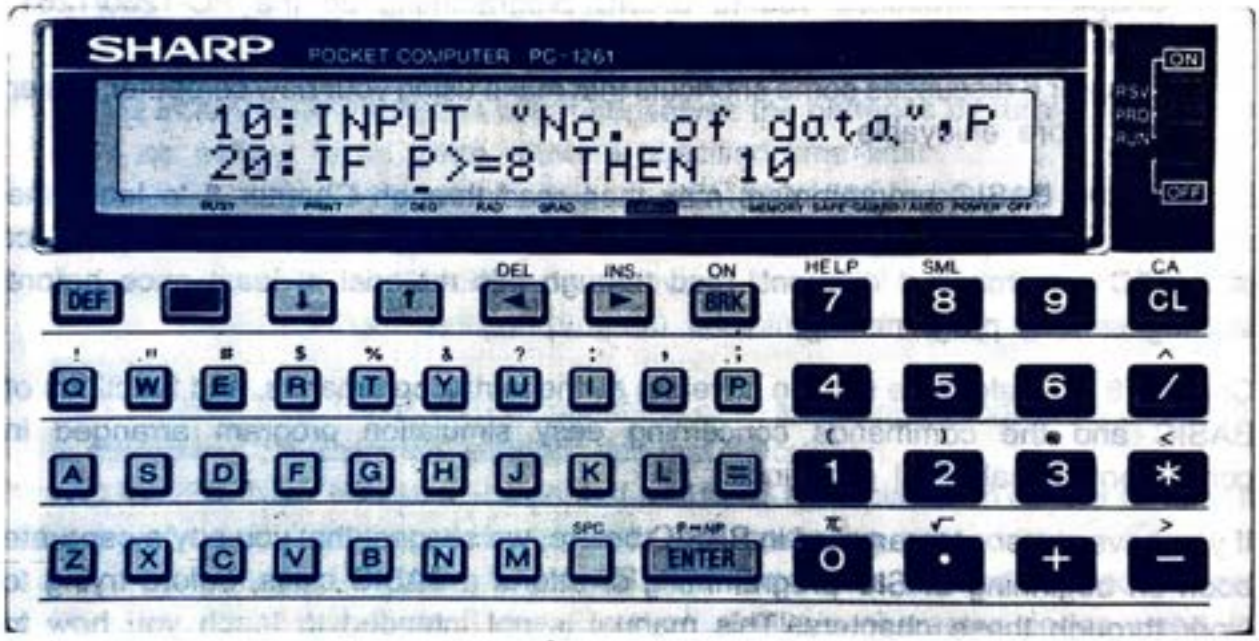
Detailed Appendices, at the end of the manual, provide you with useful charts, and special discussions concerning the use and operation of the PC-1260/1261.

Note: Unless otherwise specified, the text material applies to the both models.

CHAPTER 2
INTRODUCTION TO THE PC-1260/1261

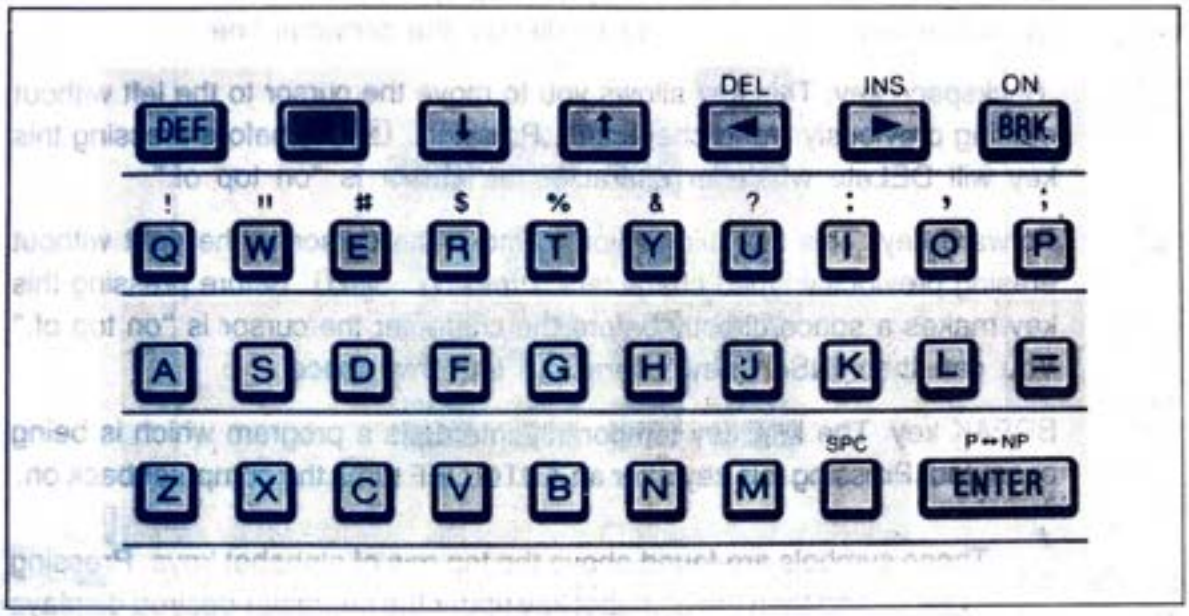
The SHARP PC-1260/1261 system consists of:

* 52-character keyboard.
* 24-digit dual line display.
* 8-bit CMOS processor.
* 40KB ROM.
* 4.4KB RAM (PC-1260) 10.4 KB RAM (PC-1261).
* Optional CE-125 Printer/Microcassette Recorder.



To familiarize you with the placement and functions of parts of the PC-1260/1261 keyboard, we will now study each section of the keyboard. For now just locate the keys and read the description of each. In Chapter 3 we will begin using your new machine.

DESCRIPTION OF KEYS



[A] ~ [Z] Alphabet keys. You are probably familiar with these keys from the standard typewriter keyboard.

When these keys are pressed, upper case characters are entered. When [SHIFT] and then [SML] key are pressed "SMALL" appears on the left end of the display. If an alphabet key is pressed, it is entered as a lower case character. If [SHIFT] and then [SML] key are pressed again, "SMALL" disappears and upper case characters can be entered.

[=] Equals key. On the PC-1260/1261 this key is not used to indicate the end of a calculation; in BASIC programming this symbol has a special function.

[SPC] SPaCe key. Pressing this key advances the cursor leaving a blank space. Pressing [SPC] while the cursor is positioned over a character, erases that character.

P↔NP
[ENTER] [ENTER] key. When you type this key, whatever you previously typed is "entered" into the computer's memory. This key is similar to the Carriage Return key on a typewriter. You must press [ENTER] before the PC-1260/1261 will act on alphanumeric input from the keyboard. Pressing [SHIFT] before pressing this key will cause the PC-1260/1261 to switch on and off the printing of calculations on the optional printer.

[DEF] [DEF] key. This is a special key used to execute BASIC programs.

SHIFT SHIFT key. Press this key before pressing any key which has a character above it and the character above is displayed.

↓ Down Arrow key. Press this key to display the next line.

↑ Up Arrow key. Press this key to display the previous line.

DEL ◄ Backspace key. This key allows you to move the cursor to the left without erasing previously typed characters. Pressing SHIFT before pressing this key will DELete whatever character the cursor is "on top of."

INS ► Forward key. This key allows you to move the cursor to the right without erasing previously typed characters. Pressing SHIFT before pressing this key makes a space directly before the character the cursor is "on top of." You can then INSert new characters into this space.

ON BRK BREAK key. The BRK key temporarily interrupts a program which is being executed. Pressing this key after an AUTO OFF turns the computer back on.

! " # $ % & ? : ' ;

These symbols are found above the top row of alphabet keys. Pressing SHIFT and then the alphabet key under the character desired displays these symbols.

Use this power slide switch to turn the PC-1260/1261 ON and OFF. Notice that the machine is ON when this switch is positioned in any one of three modes; RUN, PROgram, and ReSerVe.

[0] ~ [9] [•] Number keys. The layout of these keys is similar to that found on the standard calculator.

HELP When [SHIFT] and then [SML] are pressed, "SMALL" is displayed and lower
SML case letters can be entered. If [SHIFT] and then [SML] are pressed again, the lower case character mode is disengaged.
The HELP function can be operated by first pressing [SHIFT], and then [HELP]. The [CL] key returns the computer to the previous mode.

CA / [CL] Clear key. Pressing Clear erases the characters you have just typed in and "releases" errors. Pressing [SHIFT] before pressing this key activates the CA (reset) function. CA clears the display and resets the computer.

[/] Division key. Press this key to include the division operator in calculations. Pressing [SHIFT] and then this key will display the "power" symbol, indicating that a number is to be raised to a specific power.

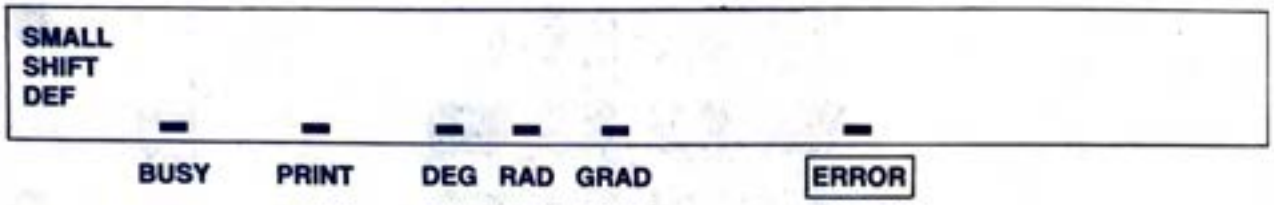
< / [*] Multiplication key. Press this key to include the multiplication operator in calculations. Pressing [SHIFT] and then this key displays the "less than" character.

> / [−] Subtraction key. Press this key to include the subtraction operator in calculations. Pressing [SHIFT] and then this key displays the "greater than" character.

[+] Addition key. Press this key to include the addition operator in calculations.

π √ ( ) @ These five characters are found above the zero, decimal point, 1, 2 and 3 keys. They are displayed by pressing [SHIFT] and then the character under the symbol desired.

DESCRIPTION OF DISPLAY



The liquid crystal display of the SHARP PC-1260/1261 shows up to 48 characters, including the cursor at one time. Although you may enter up to 80 characters including [ENTER] in one line, only the first 48 characters are displayed. To review the remaining characters in a line, press the [↓] key and the display will "scroll up".

The display consists of:

› The prompt. This symbol appears when the computer is awaiting input. As you type, the prompt disappears and is replaced by the cursor.

— The cursor. This symbol (the underline) tells you the location of the next character to be typed in. As you begin typing, the cursor replaces the prompt. The cursor is also used to position the computer over certain characters when using the INSert and DELete functions.

SMALL Lower case mode indicator. "SMALL" is displayed when [SHIFT] and then [SML] are pressed. When this indicator is shown, the alphabet keys are entered as lower case characters. When [SHIFT] and then [SML] are pressed while the "SMALL" indicator is shown, the indicator disappears and the computer returns to the upper case character mode.

SHIFT Shift Key Indicator. This indicator displays when the [SHIFT] key has been depressed. Remember, the [SHIFT] key must be released before depressing any other key.

DEF Definable Mode Indicator. This symbol displays whenever you press the [DEF] key.

BUSY Program Execution Indicator. When the PC-1260/1261 is executing a program this indicator is on (except when characters are displayed). The

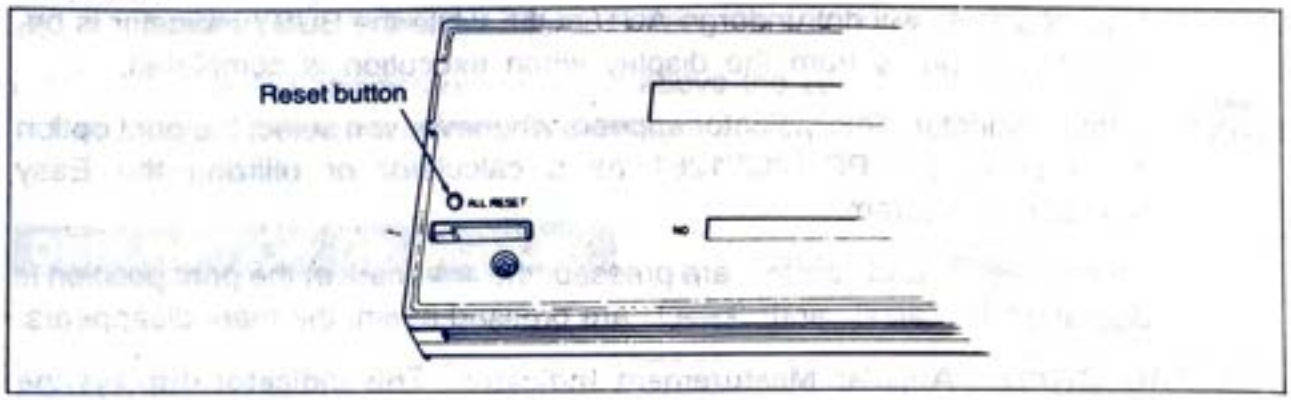
PC-1260/1261 will not undergo AUTO OFF while the BUSY indicator is on. BUSY disappears from the display when execution is completed.

PRINT Printer Indicator. This indicator appears whenever you select the print option when using the PC-1260/1261 as a calculator or utilizing the Easy Simulation Program.

When [SHIFT] and [P↔NP] are pressed, the ■ mark at the print position is displayed. If [SHIFT] and [P↔NP] are pressed again, the mark disappears.

DEG RAD GRAD Angular Measurement Indicator. This indicator displays the current unit of angles for the input of trigonometric functions. Depending on the mode in use, the indicator will appear on DEG (degrees), RAD (radians), or GRAD (gradients).

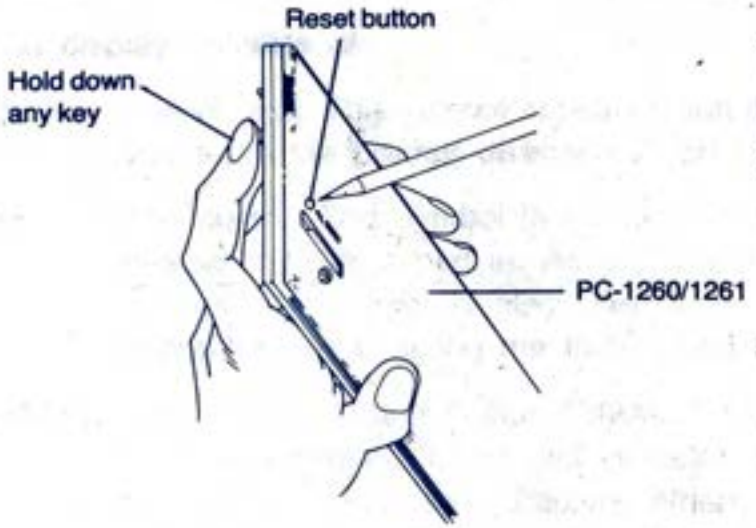
ERROR Error Indicator. Whenever an error is encountered this indicator is displayed. When an error occurs, reset with the [CL] key.



ALL RESET: Reset button. This button is used to reset the computer when CLear or CA is not sufficient to correct the problem.

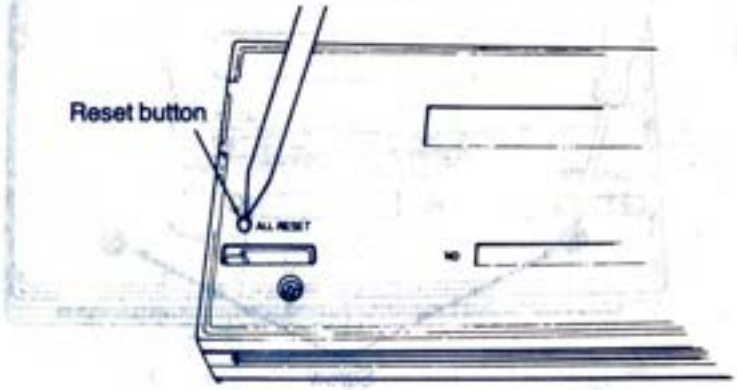
NOTE

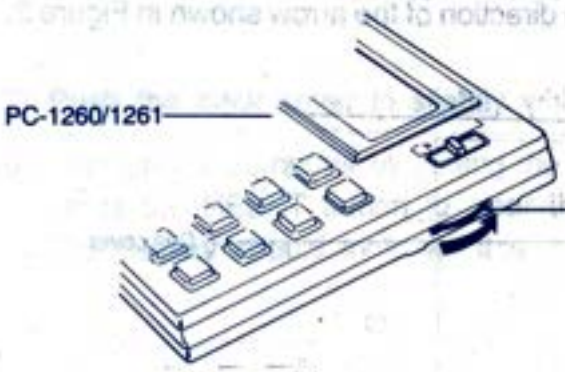
To reset the PC-1260/1261 hold down any key on the keyboard and simultaneously press the RESET button on the back. This preserves all programs, variables, and reserve memory.



Press the ALL RESET button with any pointed object such as a ballpoint pen. Do not use easily broken points such as mechanical pencils or the tips of needles.

If you get no response from any key even when the above operation is performed, push RESET button without any key. This operation will clear the program, data and all reserved contents, so do not press the RESET button without any key except when necessary.





Turn the control in the direction of the arrow to darken the display, and turn it in the opposite direction to lighten the display.
Adjust the control so that the display is easy to see.

CELL REPLACEMENT

The PC-1260/1261 operates on the lithium cells alone. When connected to the CE-125, the PC-1260/1261 can also be supplied from the CE-125 when it has enough power and the lithium cell power decreases. This minimizes the power consumption of the lithium cell.

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

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

INSTALLING THE CELLS

When the display is dim and difficult to see when viewed from the front even by turning the contrast control on the right of the computer counterclockwise as far as it goes, the cell power is consumed. In this case, replace the cells promptly. (Using the optional CE-125 or CE-126P peripheral equipment, record programs and data on tape in advance.)

(1) Turn off the computer by setting the power slide switch to the OFF position.

(2) Remove the screws from the back cover with a small screw driver. (Fig. 1)

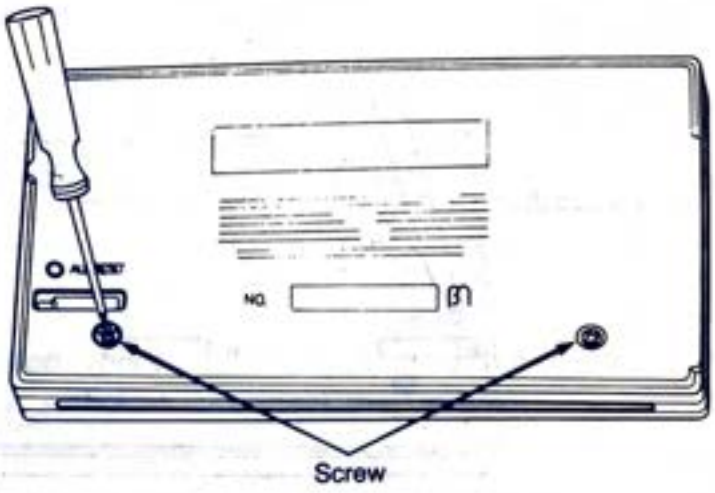


Fig. 1

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

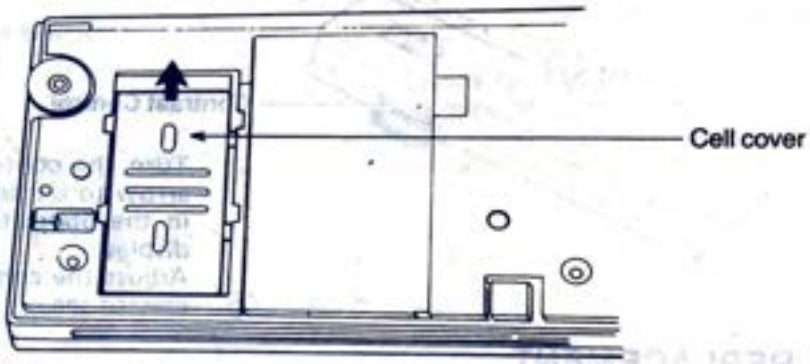


Fig. 2

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

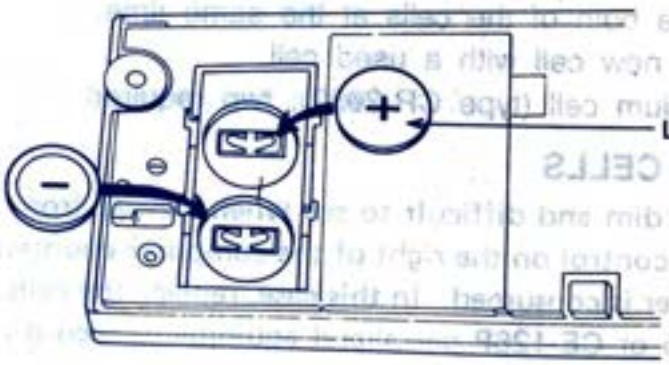


Fig. 3

(5) Replace the cell cover by sliding it in the reverse direction of the arrow shown in figure 2.

(6) Hook the claws of the back cover into the slits of the computer body. (Fig. 4)

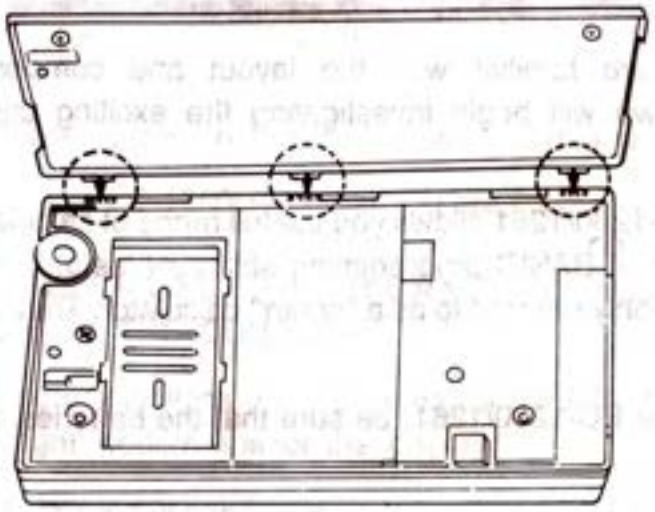
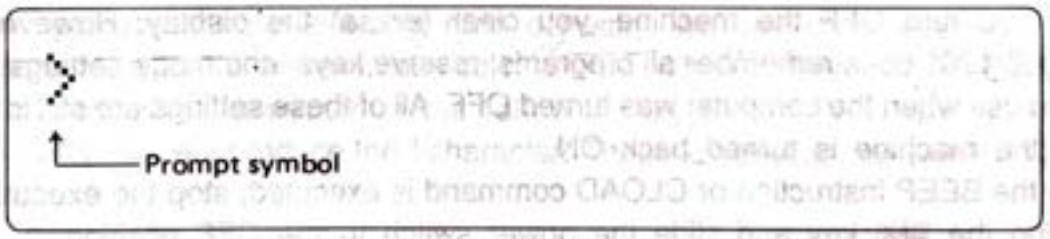


Fig. 4

(7) Push the back cover in slightly while replacing the screws.

(8) Turn on the computer by setting the power slide switch to the ON position and press the RESET button to clear the computer.
The display should look like this:



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

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

CAUTION: Keep cells out of reach of children.

CHAPTER 3
USING THE PC-1260/1261 AS A CALCULATOR

Now that you are familiar with the layout and components of the SHARP PC-1260/1261, we will begin investigating the exciting capabilities of your new computer.

Because the PC-1260/1261 allows you the full range of calculating functions, plus the increased power of BASIC programming abilities (useful in more complex calculations), it is commonly referred to as a "smart" calculator. That, of course, makes you a "smart" user!

(Before using the PC-1260/1261, be sure that the batteries are correctly installed.)

Start Up

To turn ON the PC-1260/1261, slide the power switch up and select one of three modes, RUN, PRO, or RSV. For use as a calculator, the PC-1260/1261 must be in the RUN mode. When the machine is ON, the prompt (>) will appear on the display.

Shut Down

To turn OFF the PC-1260/1261, slide the power switch to the OFF position.

When you turn OFF the machine, you clear (erase) the display. However, the PC-1260/1261 does remember all programs, reserve keys, and mode settings which were in use when the computer was turned OFF. All of these settings are still in effect when the machine is turned back ON.
When the BEEP instruction or CLOAD command is executed, stop the execution by pressing the BRK key and slide the power switch to the OFF position.

Auto Off

In order to save battery wear, the PC-1260/1261 automatically powers down when no keys have been pressed for about 11 minutes. (Note: The PC-1260/1261 will not AUTO OFF while you are executing a program.)

To restart the computer after an AUTO OFF, press the ON BRK key. All settings will be exactly as they were when the AUTO OFF occured.

Some Helpful Hints

Until you are used to your new machine, you are bound to make mistakes while

entering data. Later we will discuss some simple ways to correct these mistakes. For now, if you get an Error Message, press CLear and retype the entry. If the computer "hangs up"—you cannot get it to respond at all—press the ALL RESET button (See Chapter 2).

The PROMPT (>) tells you that the PC-1260/1261 is awaiting input. As you enter data the prompt disappears and the CURSOR (_) moves to the right indicating the next available location in the display.

The right [►] and left [◄] arrows move the cursor within a line.

Pressing [ENTER] informs the PC-1260/1261 that you are finished entering data and signals the computer to perform the indicated operations. **YOU MUST PRESS [ENTER] AT THE END OF EACH LINE OF INPUT OR YOUR CALCULATIONS WILL NOT BE ACTED UPON BY THE COMPUTER.**

When performing numeric calculations input appears on the left of the screen: the results appear on the right' bottom line of the display.

When using the [SHIFT] key in conjunction with another key (to access square root for example) press [SHIFT], release the [SHIFT], then press the other key. [SHIFT] is active for only one key at a time.

Do not use dollar signs or commas when entering calculations into the PC-1260/1261 These characters have special meaning in the BASIC programming language.

In this manual we use Ø to indicate zero, so that you can distinguish between the number (Ø) and the letter (O).

To help get you started entering data correctly, we will show each keystroke necessary to type in the example calculations. When [SHIFT] is used we will indicate the desired character in the following keystroke. For example pressing [SHIFT] and [!/O] will produce the ! character. These keystrokes are written [SHIFT] [!].

Be sure to enter CLear after each calculation (unless you are performing serial calculations). CLear erases the display and resets the error condition. It does not erase anything stored in the computer's memory.

Simple Calculations

The PC-1260/1261 performs calculations with ten-digit precision. If you have not already done so, turn ON your computer by setting it in the RUN mode. Now try these simple arithmetic examples. Remember to CLear between calculations.

Input	Display
5 0 + 5 0 ENTER	100.
1 0 0 − 5 0 ENTER	50.
6 0 * 1 0 ENTER	600.
3 0 0 / 5 ENTER	60.
1 0 SHIFT ^ 2 ENTER	100.
2 * SHIFT π ENTER	6.283185307
SHIFT √ 6 4 ENTER	8.
4 E 3 ENTER	4000.

Recalling Entries

The left arrow ◄ recalls the expression with the cursor positioned after the last character.

The right arrow ► recalls the expression with the cursor positioned "on top of" the first character.

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

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

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

Input	Display
3 0 0 / 6 ENTER	50.

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

Input	Display
◄	300/6_

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

Input	Display
◄	300/6

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

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

Input	Display
5	300/5_
ENTER	60.

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

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

Input	Display
3 Ø / 5 ENTER	6.

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

Input	Display
►	30/5

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

Input	Display
►	30/5

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

Input	Display
SHIFT INS	3 0/5

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

Input	Display
0	300/5
ENTER	60.

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

Input	Display
3 0 0 0 / 5 ENTER	600.

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

Input	Display
►	3000/5

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

Input	Display
►	3▮00/5

Now use the DELete key to get rid of one of the zeros.

Input	Display
SHIFT DEL	3▮0/5

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

Input	Display
ENTER	60.

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

Errors

Recalling your last entry is essential when you get the dreaded ERROR message. Let us imagine that, unintentionally, you typed this entry into the PC-1260/1261:

Input	Display
[3] [Ø] [Ø] [/] [/] [5] [ENTER]	ERROR 1

Naturally you are surprised when this message appears! ERROR 1 is simply the computer's way of saying, "I don't know what you want me to do here".
At this time, when the [◄] key is pressed, the flashing cursor appears at the location where the error occurred. The kind of error will be shown on the line below.

Input	Display
[◄]	300/5 SYNTAX ERROR

To correct this error use the DELete key.

Input	Display
[SHIFT] [DEL] [ENTER]	300/5 60.

If, upon recalling your entry after an ERROR 1, you find that you have omitted a character, use the INSert sequence to correct it.
When using the PC-1260/1261 as a calculator, the majority of the errors you encounter will be ERROR 1 (an error in syntax). For a complete listing of error messages, see APPENDIX A.

Serial Calculations

The PC-1260/1261 allows you to use the results of one calculation as part of the following calculation.
Part of your responsibility in planning this conference is to draw up a detailed budget for approval. You know that your total budget is $150.00 for each attendant. Figure your total budget:

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

Of this amount you plan to use 15% for the final night's awards presentation. When performing serial calculations it is not necessary to retype your previous results, but DO NOT CLear between entries. What is the awards budget?

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

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

NOTE: The [%] key cannot be used in the calculation. The [%] key should be used as a character only.

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

Input	Display
[ENTER]	6750.

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

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

Decorations will be $1225:

Input	Display
[−] [1] [2] [2] [5] [ENTER]	1525.

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

Input	Display
[−] [2] [2] [0] [0] [ENTER]	−675.

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

Negative Numbers

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

Input	Display
[*] [−] [1]	−675. * −1_
[ENTER]	675.

Now you add this result to your original presentation budget:

Input	Display
[+] [6] [7] [5] [0] [ENTER]	7425.

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

Input	Display
[/] [4] [5] [0] [0] [0] [ENTER]	0.165

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

Compound Calculations and Parentheses

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

675+6750/45000

Compound calculations, however, must be entered very carefully:

675+6750/45000 might be interpreted as

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

When performing compound calculations, the PC-1260/1261 has specific rules of expression evaluation and operator priority (see APPENDIX D). Be sure you get the calculation you want by using parentheses to clarify your expressions:

(675+6750)/45000 or 675+(6750/45000)

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

Input

[SHIFT] [(] [6] [7] [5] [+] [6]
[7] [5] [Ø] [SHIFT] [)] [/] [4]
[5] [Ø] [Ø] [Ø] [ENTER]
[6] [7] [5] [+] [SHIFT] [(] [6]
[7] [5] [Ø] [/] [4] [5] [Ø] [Ø]
[Ø] [SHIFT] [)] [ENTER]

Display

```
0.165
```

```
675.15
```

Using Variable in Calculations

The PC-1260/1261 can store up to 26 simple numeric variables under the alphabetic characters A to Z. If you are unfamiliar with the concept of variables, they are more fully explained in Chapter 4. You designate variables with an Assignment Statement:

A=5
B=−2

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

C=A+3
D=E

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

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

Input

[R] [=] [4] [5] [Ø] [Ø] [Ø]
[−] [7] [4] [2] [5]

[ENTER]

Display

```
R=45000−7425_
```

```
37575.
```

As you press [ENTER] the PC-1260/1261 performs the calculation and displays the new value of R. You can display the current value of any variable by entering the alphabetic character it is stored under:

Input	Display
[R] [ENTER]	37575.

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

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

Input	Display
[R] [*] [.] [6] [0]	R*.60_
[ENTER]	22545.

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

Input	Display
[R] [*] [.] [2] [5] [ENTER]	9393.75

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

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

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

Chained Calculations

In addition to combining several operators in one calculation, the PC-1260/1261 also allows you to perform several calculations one after the other—without having to press [ENTER] before moving on. You must separate the equations with commas. Only the result of the final calculation is displayed. (Remember too, that the maximum line length accepted by the computer is 80 characters including [ENTER] .)

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

Input	Display
R = . 8 5 * 4 5 Ø Ø Ø SHIFT . R * . 6 Ø	R=.85*45000, R*.60_

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

Input	Display
ENTER	22950.

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

Input	Display
R ENTER	38250.

Scientific Notation

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

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

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

$10^0=1$, $10^1=10$, $10^2=100$, $10^3=1000$, etc.

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

1234 becomes 1.234 times 10^3 (3 places to the right)
654321 becomes 6.54321 times 10^5 (5 places to the right)
.000125 becomes 1.25 times 10^{-4} (4 places to the left)

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

1.0×10^{87} or 1.0E 87

The PC-1260/1261 uses scientific notation whenever numbers become too large to display using decimal notation. This computer uses the capital letter E to mean "times ten to the":

1234567890000 is displayed as 1.23456789 E 12
.000000000001 is displayed as 1. E −12

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

Limits

The largest number which the PC-1260/1261 can handle is ten significant digits, with two digit exponents. In other words the largest number is:

9.999999999 E 99= 999999999000000000000000000000000
000000000000000000000000000000000
000000000000000000000000000000

and the smallest number is:

9.999999999 E −99= .000000000000000000000000000000000
000000000000000000000000000000000
000000000000000000000000000000009
999999999

Last Answer Feature

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

Input	Display
3 [+] 4 [ENTER]	7.
[*] 5	7.*5_
[ENTER]	35.

Press [CL], then the [↓] or [↑] key. If you operated these keys just after completing the calculation example above, you should see "35." in your display. The numeric data displayed is the result of your last calculation.

The PC-1260/1261 can "remember" the last answer (result) obtained through manual calculation, and recall it on its display with the [↓] or [↑] key.

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

(Example) Use the result (6.25) of the operation, 50÷8, to compute 12×5÷6.25+24×3÷6.25 =:

Input	Display
50 [/] 8 [ENTER]	6.25 Last answer
12 [*] 5 [/] [↑]	12*5/6.25_ Last answer recalled
[+] 24 [*] 3 [/] [↓]	/6.25+24*3/6.25 Last answer recalled
[ENTER]	21.12
[CL] [↓]	21.12_

The last answer is replaced with the result of the previous calculation by performing a manual calculation with the [ENTER] key.

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

The last answer is not cleared by the [CL] or key operation.

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

Length of Formula

The length of a formula you can put into your computer has a certain limitation. With the PC-1260/1261, up to 79 key strokes can be used to enter a single calculation formula (excluding the [ENTER] key). If you attempt the 80th key stroke, the cursor (▮) will start blinking on that character, indicating that the 80th key entry is not valid.

Scientific Calculations

The PC-1260/1261 is equipped with the basic functions shown below. Note that the notation of the functions in BASIC may differ from conventional mathematical notations.

Function	Conventional notation	Key operation	Remarks
Trigonometric functions	sin cos tan	SIN COS TAN	
Inverse trigonometric functions	$\sin^{-1}$ $\cos^{-1}$ $\tan^{-1}$	ASN ACS ATN	
Common logarithm	log	LOG	$\log_{10}x$ (logarithm based on 10)
Natural logarithm	ln	LN	$\log_e x$ (logarithm based on e)
Exponential function	e^x	EXP	e≒2.718281828
Exponentiation		^	A^B for A ^ B
Square root	$\sqrt{\ }$	$\sqrt{\ }$ or SQR	
Degrees (decimal) → degrees (degrees, minutes, seconds) conversion		DMS	Angle conversion (Do not leave out the 0 as in DEG.5 instead of DEG 0.5)
Degrees (degrees, minutes, seconds) → degrees (decimal) conversion		DEG	
Integer		INT	In INT(x), obtains the largest integer less than or equal to x.
Absolute	|X|	ABS	In ABS(x), obtains the absolute value of x.
Signum		SGN	Results in 1 when x>0, −1 when x<0, and 0 when x=0 for SGN(x).
Pi	π	π or PI	π≒3.141592654

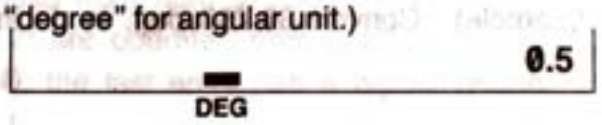
Angular unit	Command	▬Mark display position	Description
Degree	DEGREE	DEG	Represents a right angle as 90[°].
Radian	RADIAN	RAD	Represents a right angle as π/2.[rad].
Grad	GRAD	GRAD	Represents a right angle as 100[g].

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

(Example) sin 30°=

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

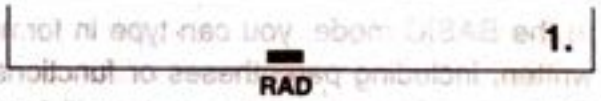
SIN 30 [ENTER]



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

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

TAN (PI/4) [ENTER]



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

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

ACS −0.5 [ENTER]

(Example) log 5+ln 5=

(Operation) LOG 5+LN 5 [ENTER]

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

(Operation) EXP (2+3) [ENTER]

(Example) $\sqrt{4^3 + 6^4} =$

(Operation) √ (4 ^ 3 + 6 ^ 4) ENTER

```
36.87817783
```

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

(Operation) DEG 30.30 ENTER

```
30.5
```
(30.5 degree)

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

(Operation) DMS 30.755 ENTER

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

Priority in Manual Calculation

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

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

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

Notes: * If parentheses are used in a formula, the operation given within the parentheses has the highest priority.

* Composite functions are operated from right to left (sin $\cos^{-1}$ 0.6).
* Chained power $(3^4)^2$ or 3^4^2) are operated from right to left.
* For the above items 3) and 4), the last entry has a higher priority.
  (e.g.) $-2\hat{}4 \rightarrow -(2^4)$
  $3\hat{}-2 \rightarrow 3^{-2}$

CHAPTER 4
CONCEPTS AND TERMS OF BASIC

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

String Constants

In addition to numbers, there are many ways that the SHARP PC-1260/1261 uses letters and special symbols. These letters, numbers, and special symbols are called characters. These characters are available on the PC-1260/1261:

1 2 3 4 5 6 7 8 9 0
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
a b c d e f g h i j k l m n o p q r s t u v w x y z
! " # $ % & ( ) * + , − . / : ; <=> ? @ √ π ^

In BASIC, a collection of characters is called a string. In order for the PC-1260/1261 to tell the difference between a string and other parts of a program, such as verbs or variable names, you must enclose the characters of the string in quotation marks (") The following are examples of string constants:

"HELLO"
"Goodbye"
"SHARP PC-1261"

The following are not valid string constants:

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

Hexadecimal Numbers

The decimal system is only one of many different systems to represent numbers. Another which has become quite important when using computers is the hexadecimal system. The hexadecimal system is based on 16 instead of 10. To write hexadecimal numbers you use the familiar 0 ~ 9 and 6 more "digits": A,B,C,D,E, and F. These correspond to 10, 11, 12, 13, 14, and 15. When you want the PC-1260/1261 to treat a number as hexadecimal put an ampersand '&' character in front of the numeral:

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

Variables

Computers are made up of many tiny memory areas called bytes. Each byte can be

thought of as a single character. For instance, the word byte requires four bytes of memory because there are four characters in it. To see how many bytes are available for use, simply type in MEM [ENTER] . The number displayed is the number of bytes available for writing programs. This technique works fine for words, but is very inefficient when you try to store numbers. For this reason, numbers are stored in a coded fashion. Thanks to this coding technique, your computer can store large numbers in only eight bytes. The largest number that can be stored is +9.999999999E+99.

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

9 E 99*9 [ENTER]

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

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

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

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

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

left side of the screen, instead of the right.

Variables handled by the SHARP PC-1260/1261 are divided into the followings:

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

Fixed Variables

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

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

You get the message:

```
ERROR 9
```

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

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

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

Figure:

```
A = A$ = A(1)  =  A$(1)
B = B$ = A(2)  =  A$(2)
C = C$ = A(3)  =  A$(3)
D = D$ = A(4)  =  A$(4)
E = E$ = A(5)  =  A$(5)
F = F$ = A(6)  =  A$(6)
G = G$ = A(7)  =  A$(7)
```

H = H$ = A(8)	=	A$(8)
I = I$ = A(9)	=	A$(9)
J = J$ = A(10)	=	A$(10)
K = K$ = A(11)	=	A$(11)
L = L$ = A(12)	=	A$(12)
M = M$ = A(13)	=	A$(13)
N = N$ = A(14)	=	A$(14)
O = O$ = A(15)	=	A$(15)
P = P$ = A(16)	=	A$(16)
Q = Q$ = A(17)	=	A$(17)
R = R$ = A(18)	=	A$(18)
S = S$ = A(19)	=	A$(19)
T = T$ = A(20)	=	A$(20)
U = U$ = A(21)	=	A$(21)
V = V$ = A(22)	=	A$(22)
W = W$ = A(23)	=	A$(23)
X = X$ = A(24)	=	A$(24)
Y = Y$ = A(25)	=	A$(25)
Z = Z$ = A(26)	=	A$(26)

Simple Variables

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

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

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

Note: ● The function or BASIC instruction names to the PC-1260/1261 computer are not usable for variable names. (e.g.) PI, IF, TO, ON, SIN, etc.

● Each simple character variable can hold up to 16 characters or symbols.

Array Variables

For some purposes it is useful to deal with numbers as an organized group, such as a

list of scores or a tax table. In BASIC these groups are called arrays. An array can be either one-dimensional, like a list, or two-dimensional, like a table.

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

DIM numeric-variable-name (size)

where:

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

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

Examples of legal numeric DIMension statements are:

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

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

It is important to know that an array-variable X and a variable X are separate and distinct to the PC-1260/1261. The first X denotes a series of numeric storage locations, and the second a single and different location.

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

```
X(4)=8
```

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

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

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

DIM character-variable-name (size) * length

where:

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

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

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

Example of legal character array declarations are:

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

The first example creates an array of five strings each able to store 16 characters. The second DIM statement declares an array NM with eleven strings of 10 characters each.

Explicit definition of strings smaller than the default helps to conserve memory space. The third example declares a two element array of 80-character strings and the last example declares a single string of twenty-six characters.

Besides the simple arrays we have just studied, the PC-1260/1261 allows "two-dimensional" arrays. By analogy, a one-dimensional array is a list of data arranged in a single column. A two-dimensional array is a table of data with rows and columns.

The two-dimensional array is declared by the statement:

DIM numeric-variable-name (row, columns)

or

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

where:

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

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

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

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

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

Arrays are very powerful programming tools.

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

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

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

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

Variables in the Form of A( )

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

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

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

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

* Using subscripts in excess of 26:

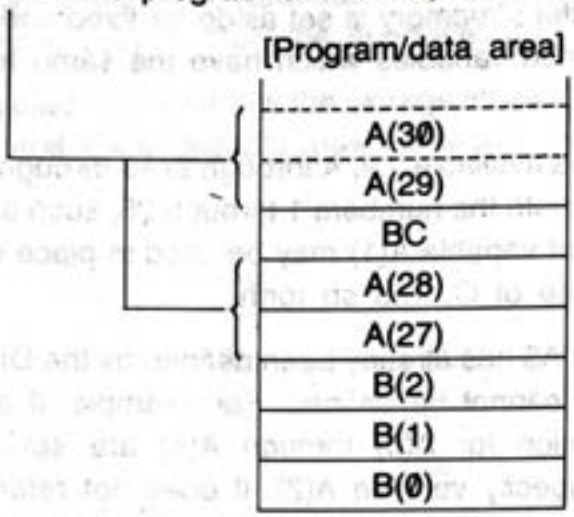
  If subscripts greater than 26 are used for subscripted variables A( ) when array A is not defined by a DIM statement, the corresponding locations in the program/data area are set aside for these A( ) variables. For instance, if you execute A(35)=5, locations for variables A(27) to A(35) will be reserved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

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

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

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

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



(2) Numeric array variables and character array variables with the same subscript cannot be defined at the same time. For example, A(30) and A$(30) cannot be defined at the same time, since they use the same location in the program/data area.

(3) Two dimensional arrays cannot be defined, nor is it possible to specify the length of character strings to be held in character array variables. For example, the length of a character string which can be held in the character array variable A$( ) is limited to seven characters or less.

(4) Variables subscripted with zero (0) cannot be defined. If A(0) or A$(0) is defined, an error will result.

Expressions

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

Numeric Operators

The PC-1260/1261 has five numeric operators. These are the arithmetic operators which you used when exploring the use of the PC-1260/1261 as a calculator in Chapter 3:

+ Addition
− Subtraction
* Multiplication
/ Division
^ Power

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

```
(A*B)^2
A(2,3)+A(3,4)+5.0-C
(A/B)*(C+D)
```

String Expressions

String expressions are similar to numeric expressions except that there is only one string operator—concatenation (+). This is the same symbol used for plus. When used with a pair of strings, the + attaches the second string to the end of the first string and makes one longer string. You should take care in making more complex string concatenations and other string operations because the work space used by the PC-1260/1261 for string calculations is limited to only 79 characters.

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

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

Relational Expressions

A relational expression compares two expressions and determines whether the stated relationship is true or false. The relational operators are:

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

The following are valid relational expressions:

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

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

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

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

Relational expressions evaluate to either true or false. The PC-1260/1261 represents true by a 1; false is represented by a 0. In any logical test an expression which evaluates to 1 or more will be regarded as true while one which evaluates to 0 or less will be considered false. Good programming practice, however, dictates the use of an explicit relational expression instead of relying on this coincidence.

Logical Expressions

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

A AND B

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

A OR B

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

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

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

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

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

NOT	0000000000000001
(Negative) →	1111111111111110

Thus, 1 is inverted to 0, and 0 to 1 for each bit, which is called "to take negative (NOT)".
Then, the following will result when 1 and NOT 1 are added together:

```
  0000000000000001 (1)
+) 1111111111111110 (NOT 1)
-------------------
  1111111111111111 (−1)
```

Thus, all bits become 1. According to the above number list, the bits become −1 in decimal notation, that is 1 + NOT 1=−1.
The relationship between numerical value X and its negative (NOT X) is:

X + NOT X=−1

This results in an equation of NOT X=−X−1
i.e. NOT X=−(X + 1)
From the equation the following are found to result.

NOT 0=−1
NOT −1=0
NOT −2=1

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

```
(A<9) AND (B>5)
(C=5) OR (C=6) OR (C=7)
```

The PC-1260/1261 implements logical operators as "bitwise" logical functions on 16 bit quantities. (See note on relational expressions and true and false). In normal operations this is not significant because the simple 1 and 0 (true and false) which result from a relational expression uses only a single bit. If you apply a logical operator to a value other than 0 or 1, it works on each bit independently. For example if A is 17, and B is 22, (A OR B) is 23:

```
17 in binary notation is    10001
22 in binary notation is    10110
17 OR 22 is                 10111 (1 if 1 in either number, otherwise 0)
10111 is 23 in decimal.
```

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

Parentheses and Operator Precedence

When evaluating complex expressions the PC-1260/1261 follows a predefined set of priorities which determine the sequence in which operators are evaluated. This can be quite significant:

5+2*3 could be

```
5+2=7          or          2*3=6
7*3=21                     6+5=11
```

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

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

```
(5+2)*3          or          5+(2*3)
```

RUN Mode

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

Input	Display
(5>3) AND (2<6)	1.

The 1 means that the expression is True.

Functions

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

```
ABS(−5)    is 5
ABS(6)     is 6
```

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

```
LOG (100)     is    2
LOG (1000)    is    3
```

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

```
LOG 100        is the same as        LOG (100)
```

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

See Chapter 8 for a complete list of functions available on the PC-1260/1261.

CHAPTER 5
PROGRAMMING THE PC-1260/1261

In the previous chapter we examined some of the concepts and terms of the BASIC programming language. In this chapter you will use these elements to create programs on the PC-1260/1261. Let us reiterate however, this is not a manual on how to program in BASIC. What this chapter will do is familiarize you with the use of BASIC on your PC-1260/1261.

Programs

A program consists of a set of instructions to the computer. Remember the PC-1260/1261 is only a machine. It will perform the exact operations that you specify. You, the programmer, are responsible for issuing the correct instructions.

BASIC Statements

The PC-1260/1261 interprets instructions according to a predetermined format. This format is called a statement. You always enter BASIC statements in the same pattern. Statements must start with a line number:

```
10: INPUT A
20: PRINT A * A
30: END
```

Line Numbers

Each line of a program must have a unique line number—any integer between 1 and 65279. Line numbers are the reference for the computer. They tell the PC-1260/1261 the order in which to perform the program. You need not enter lines in sequential order (although if you are a beginning programmer, it is probably less confusing for you to do so). The computer always begins execution with the lowest line number and moves sequentially through the lines of a program in ascending order.

When programming it is wise to allow increments in your line numbering (10, 20, 30, . . . 10, 30, 50 etc). This enables you to insert additional lines if necessary.

CAUTION: Do not use the same line numbers in different programs you plan to merge.

If you use the same line number, the oldest line with that number is deleted when you enter the new line.

Note: Do not use numbers 8960 to 9215 as line numbers. If they are used as line numbers, they may not be displayed properly on the lower line of the display (instead, other character(s) may appear) when the program is listed. If this is the case, use the ↓ key and move the line to the upper line of the display to properly display it.

The incorrect display does not affect the execution or printing of the program.

BASIC Verbs

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

```
10: INPUT A
20: PRINT A * A
30: END
```

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

```
10: INPUT A
20: PRINT A * A
30: END
```

Operands provide information to the computer telling it what data the verb will act upon. Some verbs require operands, with other verbs they are optional. Certain verbs do not allow operands. (See Chapter 8 for a complete listing of BASIC verbs and their use on the PC-1260/1261.

Note: Verbs, commands and functions must be typed in the upper case character mode.

BASIC Commands

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

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

```
RUN
NEW
RADIAN
```

Some verbs may also be used as commands. (See Chapter 8 for a complete listing of BASIC commands and their use on the PC-1260/1261).

Modes

You will remember that when using the PC-1260/1261 as a calculator, it is set in the RUN mode.

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

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

The RSV or ReSerVe mode enables you to designate and store predefined string variables and is used in more advanced programming (See Chapter 6).

Beginning to Program on the PC-1260/1261

After all your practice in using the PC-1260/1261 as a calculator you are probably quite at home with the keyboard. From now on, when we show an entry, we will not show every keystroke. Remember to use [SHIFT] to access characters above the keys and END EVERY LINE BY PRESSING THE [ENTER] KEY.

Now you are ready to program! Set the slide switch to the PROgram mode and enter this command:

Input	Display
NEW	>

The NEW command clears the PC-1260/1261 memory of all existing programs and data. The prompt appears after you press [ENTER], indicating that the computer is awaiting input.

Example 1—Entering and Running a Program

Make sure the PC-1260/1261 is in the PRO mode and enter the following program:

Input	Display
10 PRINT "HELLO"	**10: PRINT "HELLO"**

Notice that when you push [ENTER] the PC-1260/1261 displays your input, automatically inserting a colon (:) between the line number and the verb. Verify that the statement is in the correct format.

Now slide the selector switch to the RUN mode:

Input	Display
RUN	HELLO

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

Example 2—Editing a Program

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

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

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

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

The display unit, a combination of an upper and lower lines, can display 48 columns. However, if 48 columns or more are written in one line, the part beyond 48 columns cannot be seen. To look at this part, continue to press the [►] key.
The cursor moves to the bottom right end and the part which cannot be seen begins to appear in the bottom line. A maximum of up to 79 characters can be entered for one line.

Input	Display
[↑]	10: PRINT "HELLO
[◄] [◄] [◄] [◄] [◄]	10 PRINT "ELLO"

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

Input	Display
GOODBYE"!	10 PRINT "GOODBYE"!_

Don't forget to press [ENTER] at the end of the line. Switch into the RUN mode.

Input	Display
RUN [ENTER]	ERROR 1 IN 10

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

Switch back into the PROgram mode. You must be in the PROgram mode to make changes in a program. Using the [↑], recall the last line of your program.

Input	Display
[↑]	10: PRINT "GOODBYE" SYNTAX ERROR

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

Input	Display
DEL	10 PRINT "GOODBYE"_

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

Input	Display
[◄]	10 PRINT "GOODBYE"

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

Input	Display
INS	10 PRINT "GOODBYE ⌐"

Type in the !. The display looks like this:

Input	Display
!	10 PRINT "GOODBYE!"

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

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

Example 3—Using Variables in Programming

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

Using variables in programming allows much more sophisticated use of the PC-1260/1261's computing abilities.

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

```
A=5
```

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

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

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

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

```
10 INPUT "WORD?";A$
20 B=LEN (A$)
30 PRINT "WORD␣IS␣";B;"␣LETTERS"
40 END
```

␣ means space

Since line 30 of this program is longer than 24 columns, the remaining part is displayed in the next line.

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

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

Try listing this program:

Input	Display
LIST	**10: INPUT "WORD?"; A$**

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

Input	Display
RUN	**WORD?_**
HELP	**WORD IS 4. LETTERS**
ENTER	>

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

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

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

```
40: GOTO 10
```

Now RUN the modified program.

The GOTO statement causes the program to loop (keep repeating the same operation). Since you put no limit on the loop it will keep going forever (an "infinite" loop). To stop this program hit the BREAK (BRK) key.
When you have stopped a program using the BRK key, you can restart it using the CONT command. CONT stands for CONTinue. With the CONT command the program will restart on the line which was being executed when the BRK key was pressed.

Example 4—More Complex Programming

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

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

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

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

Also in this program we have used the FOR verb in line 120 and the NEXT verb in line 150 to create a loop. In Example 3 you created an "infinite" loop which kept repeating the statements inside the loop until you pressed the BRK key. With this FOR/NEXT loop the PC-1260/1261 adds 1 to N each time execution reaches the NEXT verb. It then tests to see if N is larger than the limit L. If N is less than or equal to L, execution returns to the top of the loop and the statements are executed again. If N is greater than L, execution continues with line 160 and the program stops.

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

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

```
RUN 100
```

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

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

Storing Programs in the PC-1260/1261's Memory

You will remember that settings, ReSerVe keys, and functions remain in the computer even after it is turned OFF. Programs also remain in memory when you turn off the PC-1260/1261, or it undergoes an AUTO OFF. Even if you use the BRK, CLear or CA keys the programs will remain.

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

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

This brief introduction to programming on the PC-1260/1261 should serve to illustrate the exciting programming possibilities of your new computer. For more practice in programming exercises, please see Program examples.

CHAPTER 6
SHORTCUTS

The PC-1260/1261 includes several features which make programming more convenient by reducing the number of keystrokes required to enter repetitive material.

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

This chapter discusses two additional features which can eliminate unnecessary typing—the DEF key and the ReSerVe mode.

The DEF Key and Labelled Programs

Often you will want to store several different programs in the PC-1260/1261's memory at one time. (Remember that each must have unique line numbers). Normally, to start a program with a RUN or GOTO command, you need to remember the beginning line number of each program (see Chapter 8). But, there is an easier way! You can label each program with a letter and execute the program using only two keystrokes. This is how to label a program and execute it using DEF:

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

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

Any one of the following characters can be used: A, S, D, F, G, H, J, K, L, =, Z, X, C, V, B, N, M, and SPC. Notice that these are the keys in the last two rows of the darkened on your keyboard to make it easier for you to remember.

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

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

Note: To start a different program with the DEF key during execution of the INPUT command, press the ENTER or BRK key to interrupt the INPUT command, and then start the new program.

ReSerVe Mode

Another timesaving feature of the PC-1260/1261 is the ReSerVe mode.

Within the memory of the PC-1260/1261, 48 characters are designated for "Reserve Memory". You can use this memory to store frequently used expressions, which are then recalled by a simple two keystroke operation.

Note: ● You store the strings in the ReSerVe mode and recall them for use in the RUN and PROgram modes.

● The PC-1260/1261 has a reserve memory of 48 bytes. Set up to 48 bytes, including the reserve key, in the reserve memory. A BASIC command, function, number, or a alphabetic character is 1 byte.

Example: A: S: + − 1 2 A B SIN COS INPUT RUN.....
(1 byte each)

● The length of a reserved string for one key is within 48 bytes. A maximum of up to 80 key commands, including the reserve key and [ENTER] key, can be reserved.

Try this example of storing and recalling a reserved string.

Switch the PC-1260/1261 into ReSerVe mode by moving the slide switch to the RSV position.

Type NEW followed by the [ENTER] key. This will clear out any previously stored characters in the same way NEW clears out stored programs in the PROgram mode.

Type [SHIFT] followed by 'A':

Input	Display
[SHIFT] A	A: _

Notice that the 'A' appears in the display at the left followed by a colon.

Enter the word 'PRINT' and press the [ENTER] key:

Input	Display
PRINT [ENTER]	A: PRINT_

A space appears after the colon signalling you that 'PRINT' is now stored in the reserve memory under the letter A.

Switch the PC-1260/1261 into PROgram mode. Type NEW followed by [ENTER] to clear the program memory. Type '10' as a line number and then press [SHIFT] and the 'A' key:

Input	Display
10 [SHIFT] A	10 PRINT_
[ENTER]	10: PRINT

Immediately the word 'PRINT' will appear in the display after the line number.

Any character sequence can be stored in ReSerVe Memory. The stored strings can be recalled at any time in either the PROgram or the RUN mode by typing [SHIFT] and the key that the string is stored under. The keys available are the same as those used with DEF, i.e., those in the dark area of the keyboard.

To edit a stored character sequence, switch into the ReSerVe mode and press [SHIFT] and the key under which the sequence is stored. You can then edit using the Left Arrow, Right Arrow, DEL, and INS keys in the same way as in other modes.

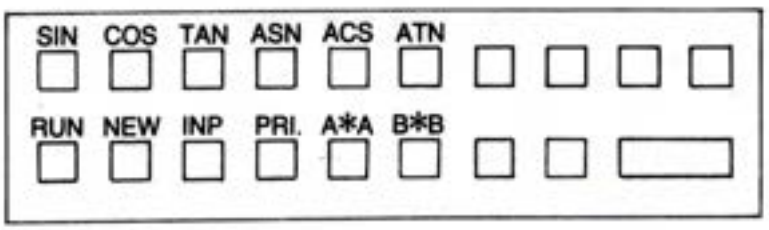
When the last character in a stored sequence is a '@' character, it is interpreted as [ENTER] when the sequence is recalled. For example, if you store the string "GOTO 100@" under the 'G' key, typing [SHIFT] and 'G' in the RUN mode immediately starts execution of the program at line 100. Without the '@' character, you must press [ENTER] after the [SHIFT] and 'G' to begin execution.

Templates

Two templates are provided with the PC-1260/1261. You can use these templates to help you remember frequently used ReSerVe sequences or DEF key assignments. After you have labelled the programs or created the sequences, mark the templates so you know what is associated with each key. You can then execute programs or recall sequences using the two-keystroke operation.

For example, if you have one group of programs which you often use at the same time, label the programs with letters and mark the template so that you can easily begin execution of any of the programs with two keystrokes. You might also store frequently used BASIC commands and verbs in the Reserve Memory and mark a template to speed up entering BASIC programs:

Example:



CHAPTER 7
USING THE PRINTER/MICROCASSETTE RECORDER AND OTHER OPTIONS

The following optional peripheral equipment can be used with the PC-1260/1261.

CE-125	Printer/microcassette recorder
CE-126P	Printer/cassette interface
CE-152	Cassette recorder (CE-126P or CE-124 is required)
CE-124	Cassette interface (CE-152 can be used)

Note: When using either the CE-125 or CE-126P optional printer, be aware that the 39 (&27), 91(&5B), and 93(&5D) character codes for the PC-1260/1261 (displayed characters) and printer (printed characters) are different characters.

Example:

PRO mode

```
10 B$ = CHR$ 91
20 PRINT B$
30 LPRINT B$
```

RUN mode

RUN [ENTER] [← Content of the character code in the pocket computer is displayed.

[ENTER] √‾ ← Content of the character code in the CE-125 or CE-126P is printed.

USING THE CE-125 PRINTER/MICROCASSETTE RECORDER

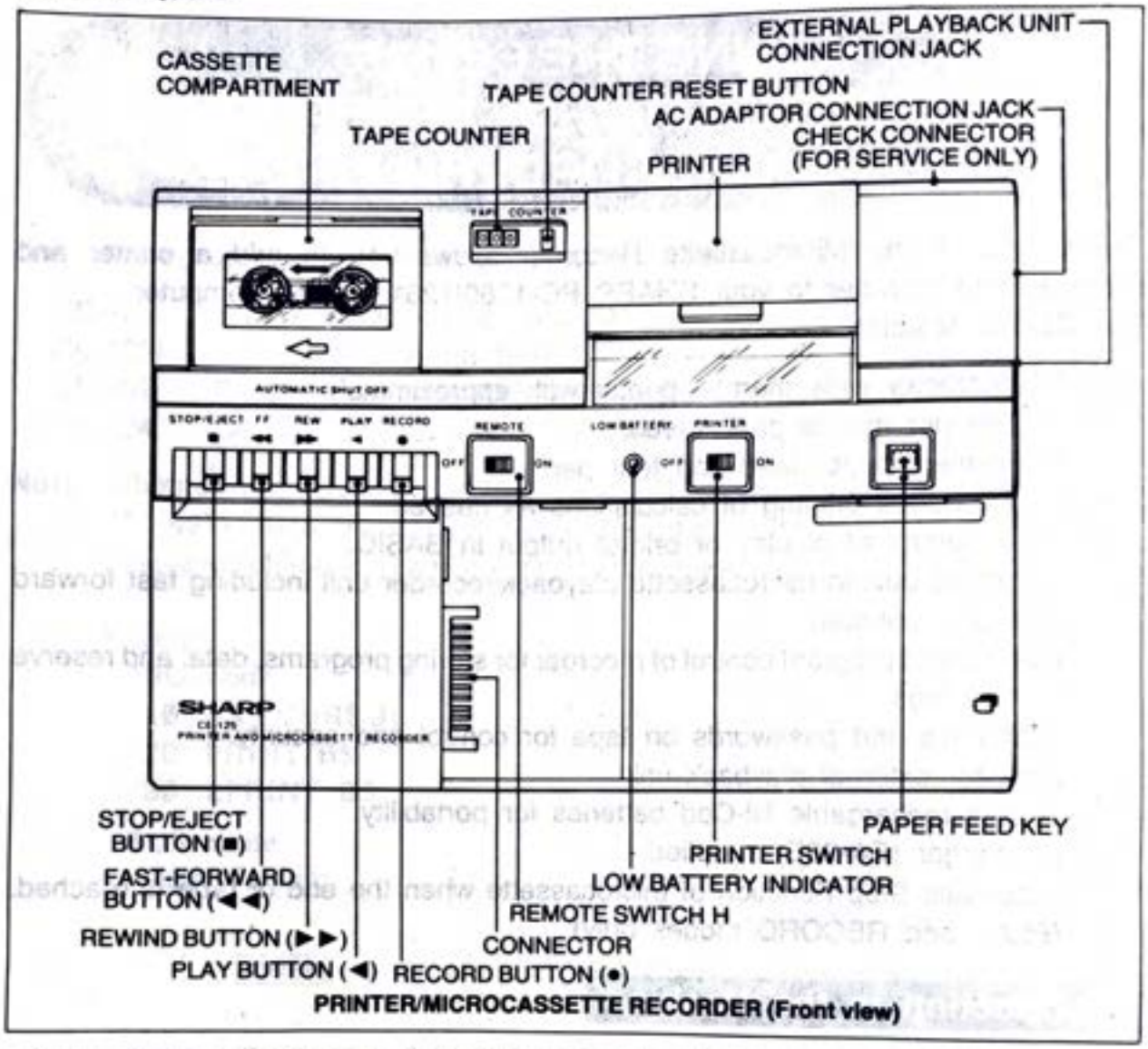
The CE-125 Printer/Microcassette Recorder allows you to add a printer and microcassette recorder to your SHARP PC-1260/1261 Pocket Computer.
The CE-125 features:

* 24 character wide thermal printer with approximately 48 line per minute print speed.
* Convenient paper feed and tear bar.
* Simultaneous printing of calculations as desired.
* Easy control of display or printer output in BASIC.
* Complete built-in microcassette playback/recorder unit including fast forward and tape counter.
* Manual and program control of recorder for storing programs, data, and reserve key settings.
* Filenames and passwords on tape for control and security.
* Jack for external playback unit.
* Built-in rechargable Ni-Cad batteries for portability.
* Recharger (EA-23E) supplied.
* Automatic Stop Function of microcassette when the end of tape is reached. (PLAY and RECORD modes only)

Introduction to the Machine

Before you begin to use the CE-125 you should first become familiar with its components. Let's examine the front of the machine:

The commands for the printer are only available on the CE-125. Also, if the printing is executed when the printer switch of the CE-125 is set at OFF position, printing causes an error (ERROR code 8). In this case, turn the printer switch to ON position, and press the [CL] key. Then, execute the printing again.



PRINTER/MICROCASSETTE RECORDER (Front view)

In the lower left corner of the machine is the cradle where you will connect the PC-1260/1261 to the CE-125. The connector at the left end of this area will mate with the corresponding plug on the PC-1260/1261. Notice the controls which run across the machine in the darkened strip. From left to right these are:

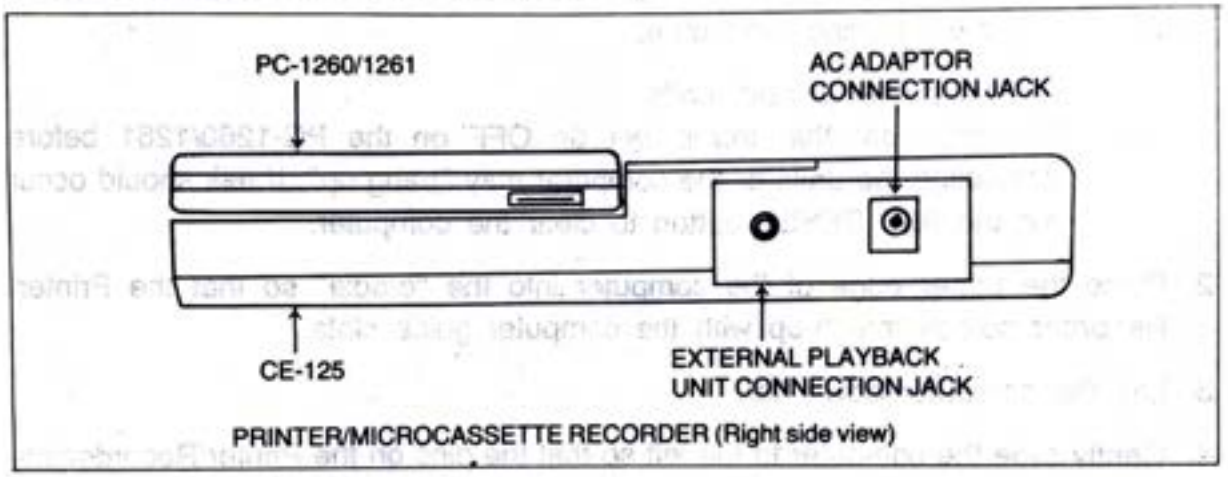
* Microcassette recorder controls—STOP/EJECT, FF, REW, PLAY, and RECORD. These should be familiar from standard tape recorders.
* REMOTE switch. This switch is used to operate the microcassette manually.
* LOW BATTERY indicator. This indicates when there is insufficient power to operate the CE-125.
* PRINTER ON/OFF. This switch is used to turn the printer on and off to conserve batteries when not in use.
* Paper feed key. Pressing this key will feed the paper in the printer.

At the top edge of the machine, between the microcassette and the printer, is the tape counter.

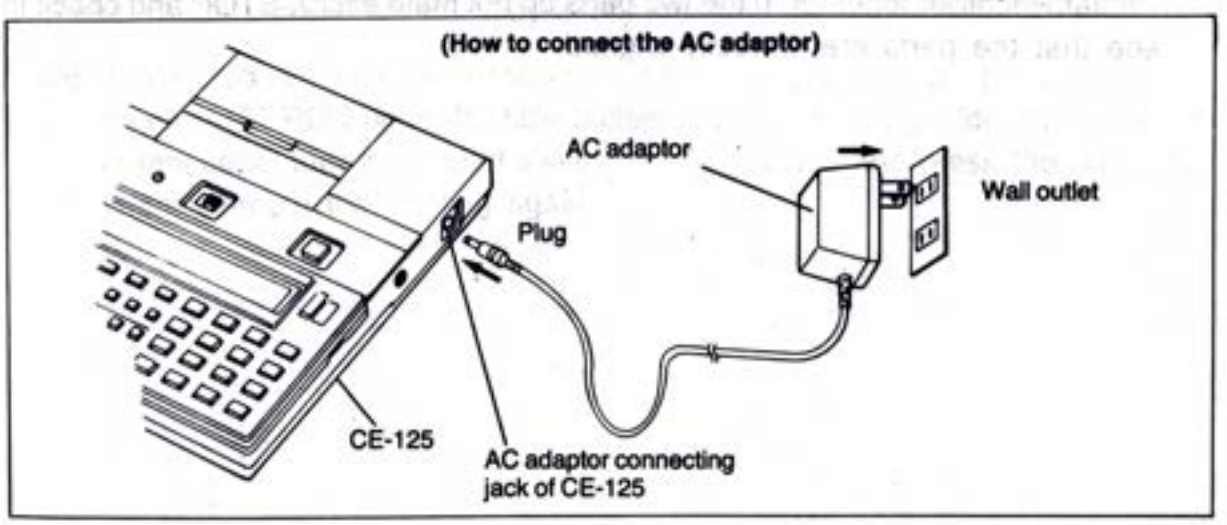
Power

The CE-125 is powered by a rechargeable Ni-Cad battery. It is necessary to recharge the battery when the low battery indicator comes ON. To recharge the battery, turn the computer and printer/recorder power OFF, connect the AC adaptor (EA-23E) to the printer/recorder, and plug the AC adaptor into a wall outlet. It will take about 15 hours before the battery is fully changed.

Important Note! Using any AC adaptor other than the one supplied (EA-23E) may damage the Printer/Recorder.



PRINTER/MICROCASSETTE RECORDER (Right side view)



Always connect the recharger to the CE-125 first. Then plug the recharger into the wall socket.

When the batteries in the CE-125 become discharged, the low battery indicator on the front of the unit lights up and the unit will not function. At this point you must recharge the batteries. When you first receive your CE-125 it is likely that the batteries will have inadequate charge due to storage. The unit will require charging before its first use.

Note: When the computer is used with the CE-125 and the battery power of the computer decreases, the power will be supplied to the computer from the CE-125.

Connecting the PC-1260/1261 to the CE-125

To connect the PC-1260/1261 Pocket Computer to the CE-125 Print/Microcassette Recorder use the following procedure:

1. Turn OFF the power in both units.
   Note: It is important that the power be OFF on the PC-1260/1261 before connecting the units or the computer may "hang up". If this should occur use the ALL RESET button to clear the computer.
2. Place the upper edge of the computer into the "cradle" so that the Printer/Recorder guides match-up with the computer guide slots.
3. Lay the computer down flat.
4. Gently slide the computer to the left so that the pins on the Printer/Recorder are inserted into the plug on the computer. DO NOT FORCE the computer and Printer/Recorder together. If the two parts do not mate easily, STOP and check to see that the parts are correctly aligned.

Fig. 1

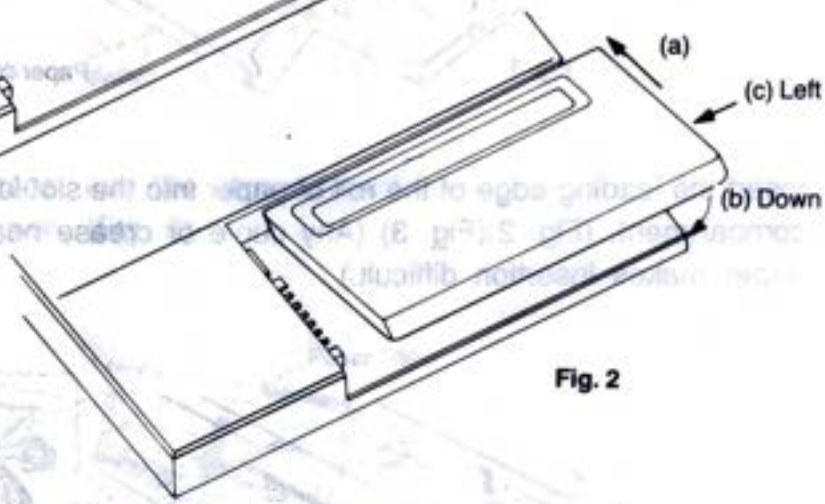


Fig. 2

5. To use the printer, turn on the PC-1260/1261 and then the CE-125. Press the [CL] key. If the [CL] key is not pressed, the printer may not operate.

Note: If executed when the printer switch is set at the OFF position, printing causes an error (ERROR code 8). (Low battery indicator may light for the moment.) In this case, turn the printer switch to ON position, and press the [CL] key. Then, execute the printing again.

Loading the Paper

(1) Turn off the printer switch.
(2) Open the paper cover. (Fig. 1)

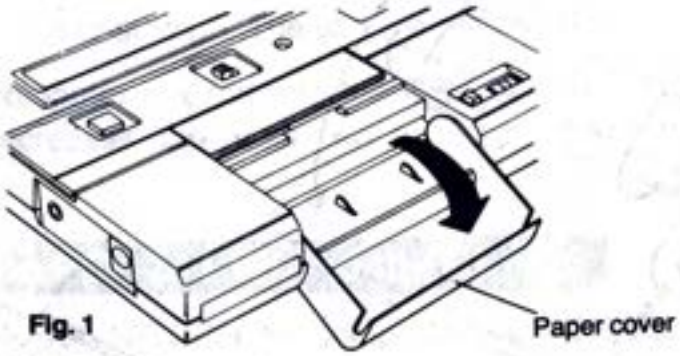


Fig. 1

(3) Insert the leading edge of the roll of paper into the slot located in the paper tape compartment. (Fig. 2)(Fig. 3) (Any curve or crease near the beginning of the paper makes insertion difficult.)

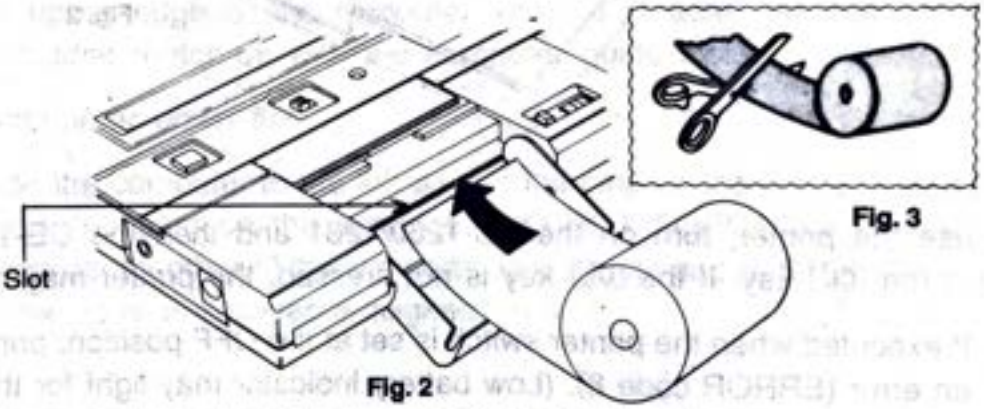
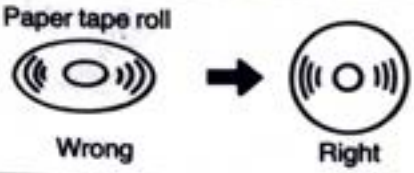


Fig. 2

Fig. 3

Note: Use of incorrect paper tape may cause irregular paper feeding or paper misfeed. Be sure to tighten the roll before using as shown in the figure.



(4) Turn on the printer switch and press the paper feed key until the paper comes out of the printer mechanism. (Fig. 4)

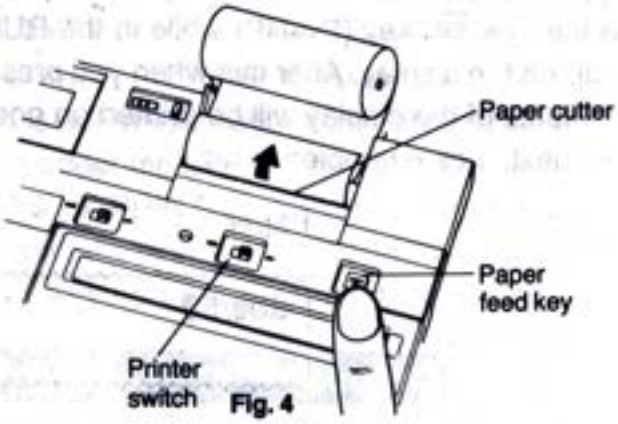


Fig. 4

(5) Install the roll of paper into the compartment.
(6) Close the paper cover. (Fig. 5)

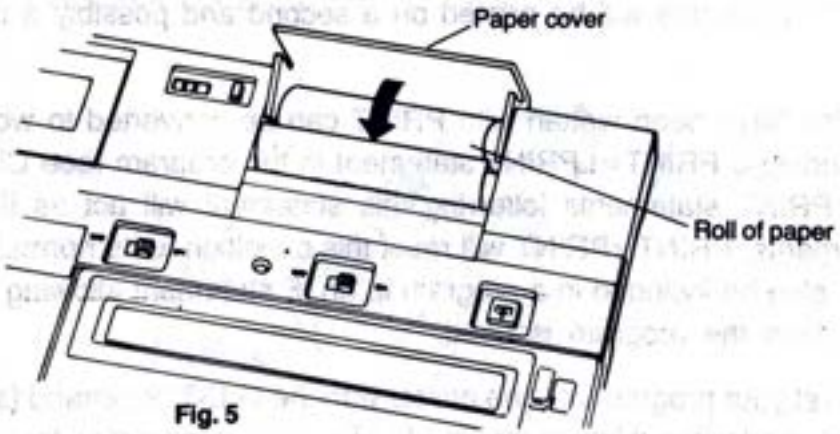


Fig. 5

- If it is necessary to remove the paper, cut the paper on the paper roll compartment side and pull the remaining paper through the printer in the direction of normal paper movement.
  Do not pull the paper backwards as this may cause damage to the printer mechanism.

CAUTION:
Paper tape is available wherever the CE-125 is sold. Please order product No EA-1250P (5 rolls per package) when reordering the paper tape. The paper tape is specifically designed for this unique printer. Use of any other paper tape may cause damage to the unit.

Using the Printer

If you are using the PC-1260/1261 as a calculator, you may use the CE-125 to simultaneously print your calculations. This is easily accomplished by pressing the [SHIFT] key and then the [ENTER] key (P↔NP) while in the RUN mode. The printer indicator "P" will light up on the display. After this,when you press [ENTER] at the end of a calculation, the contents of the display will be printed on one line and the results will be printed on the next. For example:

Input	Paper
300/50	300/50 6.

You may print output on the printer from within BASIC programs by using the LPRINT statement (see Chapter 8 for details). LPRINT functions is exactly the same fashion as the PRINT statement since both the display and the printer are 24 characters wide. The only difference is that if you PRINT something to the display which is longer than 24 characters, there is no way for you to see the extra characters. With the LPRINT verb, the extra characters will be printed on a second and possibly a third line as required.

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

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

Caution:

- If an error (ERROR code 8) occurs due to a paper misfeed, tear off the paper tape, and pull the remaining part of the paper tape completely out of the printer. Then press the [CL] key to clear the error condition.
- When the printer/recorder is exposed to strong external electrical noise, it may print numbers at random. If this happens, depress the [BRK] key to stop the printing.

then press the [CL] key.

Pressing the [CL] key will return the printer to its normal condition.

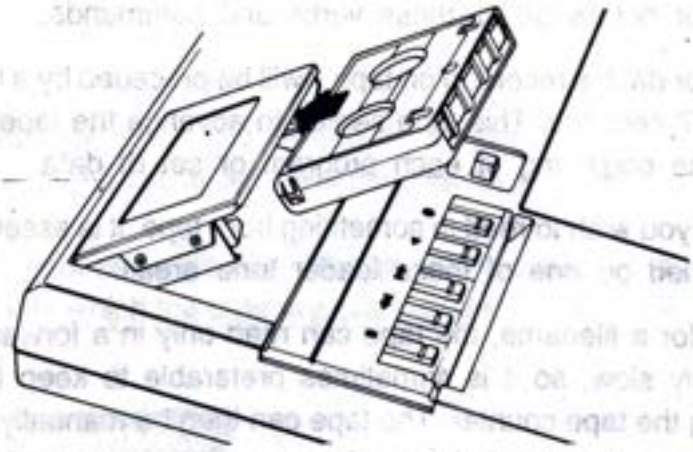
When the printer causes a paper misfeed or is exposed to strong external electrical noise while printing it may not operate normally and only the symbol "BUSY" will be displayed. If this happens, depress the [BRK] key to stop the printing. (Release the paper misfeed.) Press the [CL] key.

- When the CE-125 is not in use, turn off the printer switch to save the battery life.
- Even while printing under the LPRINT command, the entry can be executed when an INPUT, INKEY$ or PRINT command is performed.
  In this case, however, the printer will stop if the [CL] key is pressed. Therefore, only press the [CL] key upon completion of printing.

Using the Micro Cassette Recorder

LOADING THE CASSETTE TAPE

1. Depress the STOP/EJECT (■) button to open the cassette compartment lid.
2. Load the cassette tape into the compartment so that the title ("A" or "B") of the tape track to be used is facing upwards. The open edge of the cassette should be facing forward.
3. Press the cassette compartment lid down.



UNLOADING THE CASSETTE TAPE

1. Depress the STOP/EJECT (■) button to open the cassette compartment lid and remove the tape.
2. Press the cassette compartment lid down.

Note: In the PLAY mode press the STOP/EJECT (■) button once to stop the tape movement. Press it again to eject the tape.

Use the manual controls for positioning the tape. Set the 'REMOTE' switch to OFF. In

this position you may use the fast forward (FF), and rewind (REW) buttons in combination with the tape counter to position the tape as desired. To return control of the cassette unit to the PC-1260/1261, set the REMOTE switch to the ON position.

The facilities which are available with your cassette include:

CSAVE	Saves the contents of program or reserve memory on tape.
CLOAD	Retrieves a program or reserve memory from tape.
CLOAD?	Compares the program on tape with the contents of memory to insure that you have a good copy.
MERGE	Combines a program on tape with one already in memory.
PRINT #	Saves the contents of variables on tape.
INPUT #	Retrieves the contents of variables from tape.
CHAIN	Starts execution of a program which has been stored on tape.

Programs may be assigned filenames which will be stored on the tape. This allows the unambiguous storage of many programs on one tape. Programs can then be retrieved by name and the tape will be searched to find the appropriate file. If programs have been password protected in memory, they cannot be stored on tape, but a password can be assigned at the time that unprotected programs are CSAVEd. Such password protected programs can be used by other persons, but they will not be able to LIST or modify the programs in any way.

See Chapter 8 for details on all these verbs and commands.

When a program or data is recorded on tape it will be preceded by a high pitched tone of approximately 7 seconds. This tone serves to advance the tape past any leader and to identify the beginning of each program or set of data.

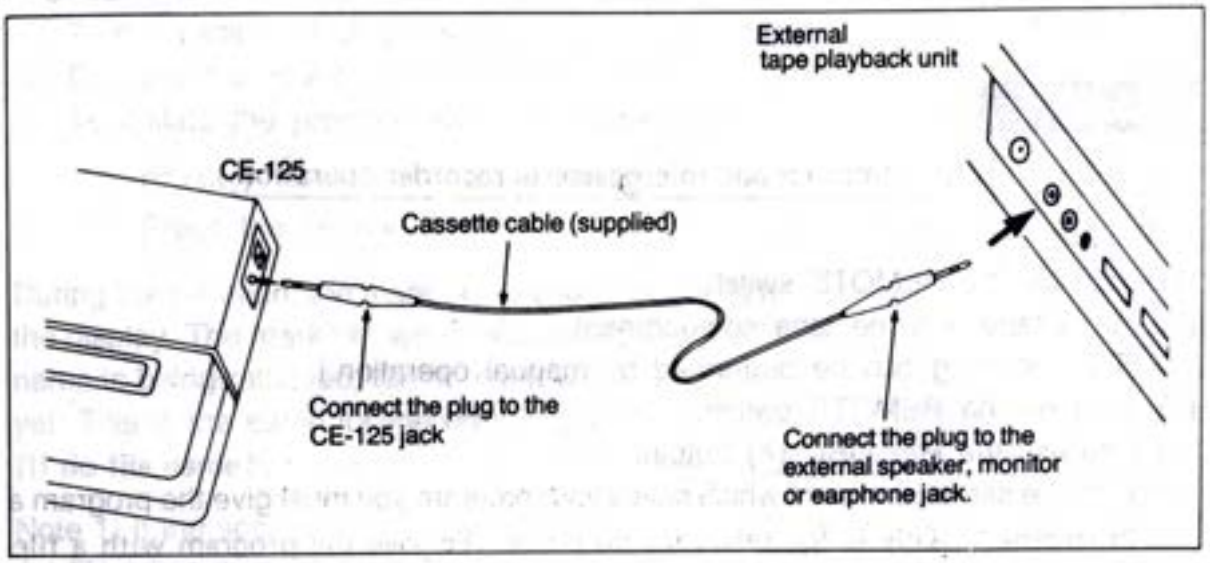
Note: Whenever you wish to read in something from tape, it is essential that the tape be positioned on one of these leader tone areas.

When searching for a filename, the tape can read only in a forward direction. This search is relatively slow, so it is sometimes preferable to keep track of program locations by using the tape counter. The tape can then be manually positioned using fast forward, reverse, and play, to the leader tone area of the correct program before the retrieval is started. While scanning the tape you will be able to hear the high tones which begin each program. In between these high tones will be a mixed high and low tone sound which indicates programs or data.

Using an External Tape Playback Unit

The CE-125 also provides a jack to connect an external tape playback unit. The external playback unit can only be used for reading from tape, i.e. CLOAD, MERGE, and INPUT #. The main purpose of this jack is to load tapes created on some other SHARP Pocket Computer.

Use the supplied cassette cable to connect the CE-125 and an external tape playback unit.



When the external tape unit is connected, it automatically takes the place of the internal microcassette for the appropriate commands and may be used in the same way.

- To transfer program and data from the tape of the external playback unit, use the tape recorder with which the tape was prerecorded. Other tape recorders may not work.

Care and Maintenance

* Be sure that the power is OFF on both units when connecting or disconnecting the CE-125 and the PC-1260/1261.
* The printer should be operated on a level surface.
* The CE-125 should be kept away from extreme temperatures, moisture, dust, and loud noises.
* Use a soft, dry cloth to clean the CE-125. DO NOT use a solvent or a wet cloth.
* Keep foreign objects out of the CE-125.
* Clean tape heads periodically with any standard head cleaning kit.

Errors

If the batteries become low, or if the CE-125 is subjected to strong noise, the unit may cease to function and the PC-1260/1261 may "hang up." This can also occur if the units are connected and the power of the CE-125 is not turned on when a LPRINT or LLIST command is used. In some cases ERROR 8 may be displayed on the PC-1260/1261.

The CLear key usually clears this condition, but in some cases the ALL RESET may be required. Be sure to restore adequate power to the CE-125 before attempting to use it again.

Examples

The procedures for computer and microcassette recorder operation

1. Saving

(1) Turn off the REMOTE switch.
(2) Put a tape into the tape compartment.
(Tape winding can be performed by manual operation.)
(3) Turn on the REMOTE switch.
(4) Depress the RECORD (●) button.
(5) With the same command which saves your program you must give the program a "filename." This is for reference purposes. To save the program with a file-name type:

CSAVE [SHIFT] "PRO-1 [SHIFT] "

Your program will be saved with the name "PRO-1". You can assign any name you desire, whatever is easiest for you to keep track of. Also, note that there is a 7-character length limit for your filename. If the name is longer than 7 characters, the excess is ignored. A good practice is to maintain a program log, which includes the program name, starting and stopping location on tape (use the counter numbers), and a brief description of what the program does.

Press the [ENTER] key. At this time you should hear a shrill buzzing sound, and the tape should be turning. Also the "BUSY" indicator should light up. This tells you that the computer is "busy" transferring your program from memory to the tape. If this does not happen, start again from the beginning of the section.

Once the computer arrives at the end of the program, the "BUSY" indicator light will go off, the recorder will stop, and the "prompt" will re-appear on the display. In order to insure that this has in fact been accomplished we can read it back into memory from the tape as explained in the next section.

Note: When saving a program on the used tape, erase the portion (approx. 5

counter numbers) before writing and execute the recording command. (Make sure that the previous program is completely erased without any portion remaining.)

2. Collating the Computer and Tape Contents

Now that your program is saved on tape, you will no doubt want to see if it is really there. To do this is relatively simple; use the CLOAD? command.

(1) Turn off the REMOTE switch to clear remote control functions.
(2) Rewind the tape to the place at which you started, again using the number counter.
(3) Turn on the REMOTE switch to set remote control functions.
(4) Depress the PLAY (◀) button.
(5) To collate the program with a filename type:

CLOAD [SHIFT] ? [SHIFT] "PRO-1 [SHIFT] "
Press the [ENTER] key.

During the collation, the mark "*" is shown at the rightmost digit of the bottom line of the display. The mark "*" will disappear when the collation is completed. While a file name is being retrieved, no "*" mark will be displayed as the collation is not started yet. This is the same for the MERGE, CHAIN, CLOAD, and INPUT # commands. (If no file name has been specified, this will occur during reading of the first program.)

Note 1) If the specified file name is not found, the computer will continue to retrieve the file name even after the tape stops. In such a case, press [ON BRK] to stop retrieval. This is the same for the MERGE, CHAIN, CLOAD, and INPUT # commands.
The computer compares the CSAVEd program with the one in its memory. If all went well, it will display the "prompt" and end its check. If all did not go well, an error message will be displayed, usually ERROR 8. This tells you that the program on tape is somehow different from the program in SHARP's memory. Erase that portion of tape and start again.

3. Transfer from Tape

(1) Turn off the REMOTE switch.
(2) Rewind the tape to the place at which you started, again using the number counter.
(3) Stop rewinding.
(4) Turn the REMOTE switch back ON.
(5) Press the PLAY (◀) button.
(6) Type:

CLOAD [SHIFT] "PRO-1 [SHIFT] "
and press the [ENTER] key.

(Remember "PRO-1" is the filename we have given to your program. If you saved the program under another name you must use that name instead of PRO-1)

(7) The mark "*" appears while loading the designated CSAVEd program from the tape to the computer's memory.
(If no file name has been specified, this will occur during reading of the first program.) The mark "*" disappears when the load is performed completely.

(8) The cassette retains a copy of the program, so you can CLOAD the same program over and over again.
If an error message ERROR 8 is displayed, while loading, start again from step (1).

USING THE CE-126P PRINTER/CASSETTE INTERFACE

The optional CE-126P Printer/Cassette Interface allows you to add a printer and to connect a cassette recorder to your SHARP PC-1260/1261 Computer.
The CE-126P features:

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

For connecting the PC-1260/1261 to the CE-126P, refer to the instruction manual which is supplied with the CE-126P.

Using the Printer

If you are using the PC-1260/1261 for manual calculation, you may use the CE-126P to simultaneously print your calculations.

> **CAUTION:**
> The result which is obtained by the direct calculation feature in manual calculation cannot be printed.

This is easily accomplished by pressing the [SHIFT] key and then the [ENTER] key (P ↔ NP) while in the RUN mode.
The printer indicator (a dash symbol) will appear just above the "PRINT" label in the lower left area of the display. After this, when you press the [ENTER] at the end of a calculation, the contents of the display will be printed on one line and the results will be printed on the next. For example:

Input	Paper
300/50 [ENTER]	300/50 6.

Using the Options

You may print output on the printer from within BASIC programs by using the LPRINT statement (see Chapter 8 for details). LPRINT can be used in the same form as the PRINT statement.

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

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

Caution:

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

  When the printer causes a paper misfeed or is exposed to strong external electrical noise while printing, it may not operate normally and only the symbol "BUSY" will be displayed. If this happens, depress the [BRK] key to stop printing. (Release the paper misfeed.) Turn the CE-126P power off and on, and then press the [CL] key.
- When the CE-126P is not in use, turn off the printer switch to save the battery life.

Using the Cassette Interface

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

Connecting the CE-126P to a Tape Recorder

Only three connections are necessary:

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

Cassette Tape Recorder

We recommend that you use the optional cassette tape recorder CE-152 for your pocket computer system. The CE-152 is designed to match the PC-1260/1261 for recording programs and data via the CE-126P cassette interface. Any recorded program can be retrieved and reloaded into the PC-1260/1261.

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

The following is a description of the minimum tape recorder specifications necessary for interfacing with the CE-126P if any cassette tape recorder other than the CE-152 is used:

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

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

Note: Some tape recorders may not perform properly due to different specifications. Additionally, 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.

Operating the Cassette Interface and Recorder

Recording (saving) onto magnetic tape

See Tape Notes.

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

   First set the unit to "RUN" or "PRO" mode. Next push the following keys:
   [C] [S] [A] [V] [E] [SHIFT] ["] file name [SHIFT] [#] [ENTER].
   (To write the contents of data memory onto tape, push as follows:
   [P] [R] [I] [N] [T] [SHIFT] [#] [ENTER].)
   E.g., [C] [S] [A] [V] [E] [SHIFT] ["] [A] [A] [SHIFT] ["] [ENTER]

   When you press the [ENTER] key, tape motion will begin, leaving about an 8-second non-signal blank. (Beep tone is recorded.) After that, the file name and its contents are recorded.
8. When the recording is complete, the PROMPT symbol (>) will be displayed and the tape recorder will automatically stop. The program is now saved on tape. (The program also remains in the computer's memory.)

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

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

Collating the Computer and Tape Contents

See tape Notes.

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

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

   The file name which you used previously.

   [C] [L] [O] [A] [D] [SHIFT] [?] [SHIFT] ["] [A] [A] [SHIFT] ["] [ENTER]

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

   During the collation, the mark "*" is shown at the rightmost digit of the bottom line of the display. The mark "*" will disappear when the collation is completed. While a file name is being retrieved, no "*" mark will be displayed as the collation is not started yet.
   (If no file name has been specified, this will occur during reading of the first program.)

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

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

Loading from a magnetic tape

See Tape Notes.

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

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

3. Connect the gray plug to the EAR jack on the tape recorder, and the black plug to the REM jack.
   [In using a tape recorder having no REM terminal, press the PAUSE button to make a temporary stop.]

4. Turn on the REMOTE switch.

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

   Set the VOLUME control to middle or maximum.
   Set Tone to maximum treble.

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

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

   E.g., [C] [L] [O] [A] [D] [SHIFT] ["] [A] [A] [SHIFT] ["] [ENTER]

   The specified file name will be automatically searched for and its contents will be transferred into the Pocket Computer.
   The mark "*" appears while loading the designated CSAVEd program from the tape to the computer's memory.
   (If no file name has been specified, this will occur during reading of the first program.) The mark "*" disappears when the load is performed completely.

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

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

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

Tape notes

1) For any transfer or collation, use the tape recorder that was used for recording. If the tape recorder for transfer or collation is different from that used for recording, transfer or collation may not be possible.

2) Always use only the highest quality tape for programs and data storage (economy grade audio type tape may not provide the proper characterstics for digital recordings).

3) Keep the tape heads and tape handling parts clean—use a cassette cleaner tape to keep everything clean.

4) Volume setting—set to middle or maximum level.
Volume level can be very important when reading in data from the recorder; make slight adjustments as required to obtain error-free data transfer. A slight adjustment either up or down may result in perfect recordings every time.

5) Be sure all connections between the pocket computer and cassette interface are secure. And be sure the connections between interface and recorder are secure and dirt-free.

6) If problems occur when using AC power for the CE-126P and/or the recorder, use battery power instead (sometimes the AC power connection also adds some "hum" to the signal which upsets proper digital recordings).

   - To connect the AC adaptor to the CE-126P, turn the CE-126P power off and then connect the adaptor to the CE-126P.

7) Tone control—set to maximum treble.

8) When recording programs or data on the used tape, erase the portion before writing and execute the recording command. (Make sure that the previous program is completely erased without any portion remaining.)

CHAPTER 8
BASIC REFERENCE

The following chapter is divided into three sections:

Commands: Instructions which are used outside a program to change the working environment, perform utilities, or control programs.

Verbs: Action words used in programs to construct BASIC statements.

Functions: Special operators used in BASIC programs to change one variable into another.

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

Note: Commands, verbs, and functions must be typed in the upper case character mode (normal mode).

Commands

Program Control
- CONT
- GOTO*
- NEW
- RUN

Cassette Control
- CLOAD
- CLOAD?
- CSAVE
- INPUT #*
- MERGE
- PRINT #*

Debugging
- LIST
- LLIST
- TROFF*
- TRON*

Variables Control
- CLEAR*
- DIM*

Angle Mode Control
- DEGREE*
- GRAD*
- RADIAN*

Other
- BEEP*
- PASS
- RANDOM*
- USING*
- WAIT*

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

Verbs

Control and Branching

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

Assignment and Declaration

- CLEAR
- DIM
- LET

Input and Output

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

Other

- BEEP
- CLS
- DEGREE
- GRAD
- RADIAN
- RANDOM
- REM
- TROFF
- TRON

Functions

Pseudovariables

INKEY$
MEM
PI

String Functions

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

Numeric Functions

ABS
ACS
ASN
ATN
COS
DEG
DMS
EXP
INT
LN
LOG
RND
SGN
SIN
SQR
TAN

Commands concerning easy simulation program.

EQU #
LIST #
LLIST #
MEM #
NEW #

Logic (See page 44)

AND
NOT
OR

COMMANDS

1 **CLOAD**
2 **CLOAD** "filename"

Abbreviations: CLO., CLOA.

See also: CLOAD?, CSAVE, MERGE, PASS

Purpose

The CLOAD command is used to load a program saved on cassette tape. It can only be used with the optional CE-125 or CE-152.

Use

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

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

If the PC-1260/1261 is in PROgram or RUN mode, program memory is loaded from the tape. When the PC-1260/1261 is in the ReSerVe mode, reserve memory is loaded. Care should be taken not to load programs into reserve memory or reserve characters into program memory.

Examples

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

Notes:
1. The computer cannot identify the stored contents as a program or a reserve. Therefore, if a mode is designated incorrectly, the reserved contents may be transferred to the program area or the program to the reserve area, causing all computer housekeeping to remain inoperative. If this happens, reset the computer by pressing the RESET button on the back of the computer.

2. If the designated file name is not retrieved, the computer will continue to search the file name even after the tape reaches the end. In this case, stop the retrieval function by pressing the ON/BRK key. This applies to MERGE, CHAIN, CLOAD? and INPUT # commands to be described later.

3. If an error occurs during CLOAD or CHAIN command (to be described later) execution, the program stored in the computer will be invalid.

1 **CLOAD?**
2 **CLOAD?** "filename"

Abbreviations: CLO.?,CLOA.?

See also: CLOAD, CSAVE, MERGE, PASS

Purpose

The CLOAD? command is used to compare a program saved on cassette tape with one stored in memory. It can only be used with the optional CE-125 or CE-152.

Use

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

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

Examples

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

1 **CONT**

Abbreviations: C.,CO.,CON.

See also: RUN, STOP verb

Purpose

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

Use

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

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

Examples

CONT Continues an interrupted program execution.

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

Abbreviations: CS., CSA., CSAV.

See also: CLOAD, CLOAD?, MERGE, PASS.

Purpose

The CSAVE command is used to save a program to cassette tape. It can only be used with the optional CE-125 or CE-152.

Use

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

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

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

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

If the PC-1260/1261 is in PROgram or RUN mode, program memory is loaded to the tape. When the PC-1260/1261 is in the ReSerVe mode, reserve memory is loaded.

Examples

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

1 **GOTO** expression

Abbreviations: G., GO., GOT.

See also: RUN

Purpose

The GOTO command is used to start execution of a program.

Use

The GOTO command can be used in place of the RUN command to start program execution at the line number specified by the expression.

GOTO differs from RUN in six respects:

1) The value of the interval for WAIT is not reset.
2) The display format established by USING statements is not cleared.
3) Variables and arrays are preserved.
4) PRINT=LPRINT status is not reset.
5) The pointer for READ is not reset.
6) The cursor specification is maintained.

Execution of a program with GOTO is identical to execution with the DEF key.

Examples

GOTO 100 Begins execution of the program at line 100.

1 **LIST**
2 **LIST** expression

Abbreviations: L., LI., LIS.

See also: LLIST

Purpose

The LIST command is used to display a program.

Use

The LIST command may only be used in the PROgram mode. The first form of the LIST command displays the statement with the lowest line number.

The second form displays the statement with the nearest line number equal to or greater than the value of the expression. The Up Arrow and Down Arrow keys may then be used to examine the program.

When several programs are stored with the MERGE command, the LIST command works for the last merged program (read with the MERGE command). However, when executing the LIST command in the form:

LIST "label"

and if the specified label is not found in the last merged program, the other programs are searched. If the specified label is found, the line is displayed.
If a password has been set, the LIST command is ignored.

Examples

LIST 100 Displays line number 100.

1 **LLIST**
2 **LLIST** expression
3 **LLIST** expression 1, expression 2
4 **LLIST** expression,
5 **LLIST**, expression

Abbreviations: LL., LLI., LLIS.

See also: LIST

Purpose

The LLIST command is used for printing a program on the optional CE-125 or CE-126P.

Use

The LLIST command may be used in the PROgram or RUN mode.

The first form prints all of the programs in memory.

The second form prints only the program line whose line number is given by expression.

The third form prints the statements from the line number with the nearest line equal to or greater than the value of expression 1 to the nearest line equal to or greater than the value of expression 2. There must be at least two lines between the two numbers.

The fourth form prints program lines beginning with the line whose number is given by the expression.
The fifth form prints all program lines up to, and including, the line whose number is given by the expression.

When several programs are stored with the MERGE command, the LLIST command works for the last merged program.
However, if a label is specified in the form:
(1) LLIST "label"
(2) LLIST, "label"
and if the specified label is not found in the last merged program, the other programs are searched. If the specified label is found, the line is printed (for (1)) or the program after and including the line is printed (for (2)).
If a password has been set the LLIST command is ignored.

Examples

LLIST 100,200 Lists the statements between line numbers 100 and 200.

1 MERGE
2 MERGE "filename"
(effective for the manual operation in the PROgram or RUN mode)

Abbreviations: MER., MERG.

See also: CLOAD

Purpose

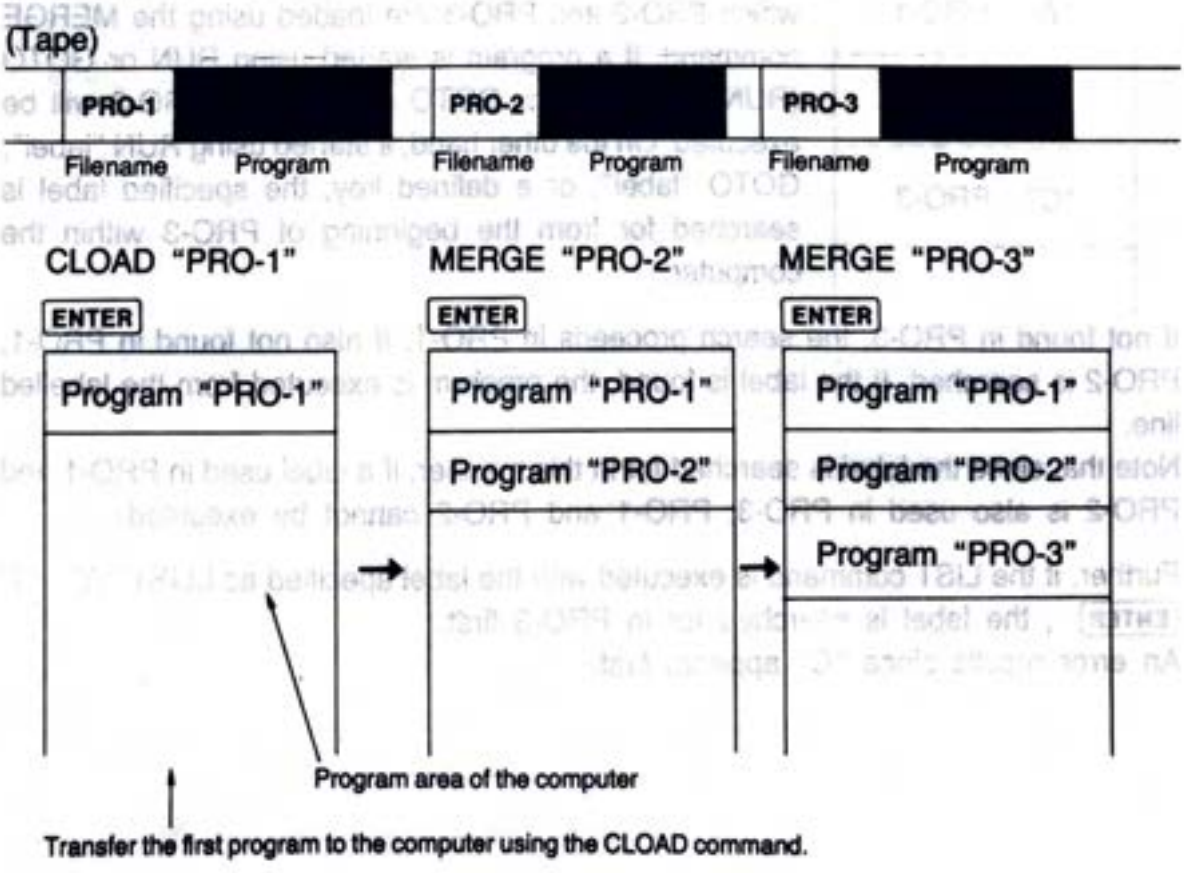
The MERGE command is used to load a program saved on cassette tape and merge it with the program existing in memory.

Use

The MERGE command, while maintaining the program already in the computer, loads the contents recorded on tape by adding to it. Therefore, a completely different program can be stored in the computer simultaneously.

Example

When programs with file names PRO-1, PRO-2 and PRO-3 are to be stored, PRO-1 is stored using the CLOAD command, and PRO-2 and PRO-3 are transferred to the computer using the MERGE command. The state of the storage is as follows.



When loading programs using the MERGE command, there may be two or more programs with the same line numbers within the computer. In this case, the RUN or GOTO commands (which execute programs) will be valid only for the last merged program (transferred using the MERGE command). Since the first, second, etc, programs cannot be executed, these programs must be defined for keys such as [A], [B], [C], etc., beforehand, as "defined programs."

After executing the MERGE command, only the last merged program can be edited. Therefore, define a program before merging the next program.

Merging password protected programs

When loading programs with passwords (password protected programs) using the MERGE command, the handling of the programs differ as follows depending on whether the programs within the computer are protected.

When protected

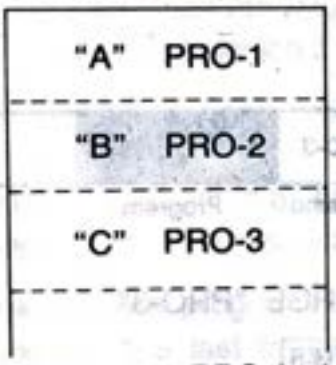
Password protected programs cannot be loaded.

When not protected

If password protected programs are loaded using the MERGE command, all programs within the computer become protected.

When the programs within the computer are protected, even programs without passwords become password protected when loaded using the MERGE command.

Executing merged programs



The figure shows the memory when PRO-1 is loaded after which PRO-2 and PRO-3 are loaded using the MERGE command. If a program is started using RUN or GOTO (RUN expression or GOTO expression), PRO-3 will be executed. On the other hand, if started using RUN "label", GOTO "label", or a defined key, the specified label is searched for from the beginning of PRO-3 within the computer.

If not found in PRO-3, the search proceeds in PRO-1. If also not found in PRO-1, PRO-2 is searched. If the label is found, the program is executed from the labelled line.

Note that since the label is searched for in this manner, if a label used in PRO-1 and PRO-2 is also used in PRO-3, PRO-1 and PRO-2 cannot be executed.

Further, if the LIST command is executed with the label specified as LLIST "A", "C" [ENTER], the label is searched for in PRO-3 first.

An error results since "C" appears first.

1 **NEW**

Abbreviations: none

See also: NEW #

Purpose

The NEW command is used to clear existing program or reserve memory.

Use

When used in the PROgram mode the NEW command clears all programs and data which are currently in memory.

When used in the ReSerVe mode the NEW command clears all existing reserve memory.

The NEW command is not defined in the RUN mode and will result in an ERROR 9.

Examples

NEW Clears program or reserve memory

1 **PASS** "character string"

Abbreviations: PA., PAS.

See also: CSAVE, CLOAD

Purpose

The PASS command is used to set and cancel passwords.

Use

Passwords are used to protect programs from inspection or modification by other users. A password consists of a character string which is no more than seven characters long. The seven characters must be alphabetic or one of the following special symbols: ! # $ % & ( ) * + − / , . : ; < = > ? @ √ π ^

Once a PASS command has been given, the programs in memory are protected. A password protected program cannot be examined or modified in memory. It cannot be sent to tape or listed with LIST or LLIST, nor is it possible to add or delete program lines. If several programs are in memory and PASS is entered, all programs in memory are protected. The only way to remove this protection is to execute another PASS command with the same password or to enter NEW (which erases the programs).

Examples

PASS "SECRET" Establishes the password 'SECRET' for all programs in memory.

1 **RUN**
2 **RUN** line number

Abbreviations: R., RU.

See also: GOTO

Purpose

The RUN command is used to execute a program in memory.

Use

The first form of the RUN command executes a program beginning with the lowest numbered statement in memory.

The second form of the RUN command executes a program beginning with the specified line number.

RUN differs from GOTO in six respects:

1) The value of the interval for WAIT is reset.
2) The display format established by USING statements is cleared.
3) Variables and arrays other than the fixed variables are cleared.
4) PRINT=PRINT status is set.
5) The pointer for READ is reset to the beginning DATA statement.
6) The cursor specification is cleared.

Execution of a program with GOTO is identical to execution with the DEF key. In all three forms of program execution FOR/NEXT and GOSUB nesting is cleared.

Examples

RUN 100 Executes the program which begins at line number 100.

VERBS

1 **AREAD** variable name

Abbreviations: A., AR., ARE., AREA.

See also: INPUT verb and discussion of the use of the DEF key in Chapter 6

Purpose

The AREAD verb is used to read in a single value to a program which is started using the DEF key.

Use

When a program is labelled with a letter, so that it can be started using the DEF key, the AREAD verb can be used to enter a single starting value without the use of the INPUT verb. The AREAD verb must appear on the first line of the program following the label. If it appears elsewhere in the program, it will be ignored. Either a numeric or string variable may be used, but only one can be used per program.

To use the AREAD verb type the desired value in the RUN mode, press the DEF key, followed by the letter which identifies the program. If a string variable is being used, it is not necessary to enclose the entered string in quotes.

Examples

```
10 "X": AREAD N
20 PRINT N^2
30 END
```

Entering "7 DEF X" will produce a display of "49".

Notes:

1. When the display indicates PROMPT (">") at the start of program execution, the designated variable is cleared.

2. When the contents of the display have been displayed by PRINT verb just prior to the start of program execution, the following is stored:

Example: When the program below is executed;

```
10 "A": PRINT "ABC", "DEFG"
20 "S": AREAD A$: PRINT A$
RUN mode
DEF [A] → ABC DEFG
DEF [$] → DEFG
```

- When the display indicates PRINT numeric expression, numeric expression, numeric expression, numeric expression, or PRINT "String" "String", "String", "String", the contents displayed last are stored.
- When the display indicates PRINT Numeric expression; Numeric expression; Numeric expression ..., the contents displayed first (on the extreme left) are stored.
- When the display indicates PRINT "String"; "String"; "String" ..., the contents of the "String" designated last (on the extreme right) are stored.

1 **BEEP** expression

Abbreviations: B., BE., BEE.

Purpose

The BEEP verb is used to produce an audible tone.

Use

The BEEP verb causes the PC-1260/1261 to emit one or more audible tones at 4 kHz. The number of beeps is determined by the expression, which must be numeric. (Positive number less than 9.999999999E+99) The expression is evaluated, but only the integer part is used to determine the number of beeps.

BEEP may also be used as a command using numeric literals and predefined variables. In this case the beeps occur immediately after the [ENTER] key is pressed.

Examples

```
10 A=5 : B$="9"
20 BEEP 3          Produces 3 beeps.
30 BEEP A          Produces 5 beeps.
40 BEEP(A+4)/2     Produces 4 beeps.
50 BEEP B$         This is illegal and will produce an ERROR 9 message.
60 BEEP -4         Produces no beeps, but does not produce an error message.
```

1 **CHAIN**
2 CHAIN expression
3 CHAIN "filename"
4 CHAIN "filename", expression

Abbreviations: CH., CHA., CHAI.

See also: CLOAD, CSAVE, and RUN

Purpose

The CHAIN verb is used to start execution of a program which has been stored on cassette tape. It can only be used in connection with the optional CE-125 or CE-152.

Use

To use the CHAIN verb one or more programs must be stored on a cassette. Then, when the CHAIN verb is encountered in a running program, a program is loaded from the cassette and executed.

The first form of CHAIN loads the first program stored on the tape and begins execution with the lowest line number in the program. The effect is the same as having entered CLOAD and RUN when in the RUN mode.

The second form of CHAIN loads the first program stored on the tape and begins execution with the line number specified by the expression.

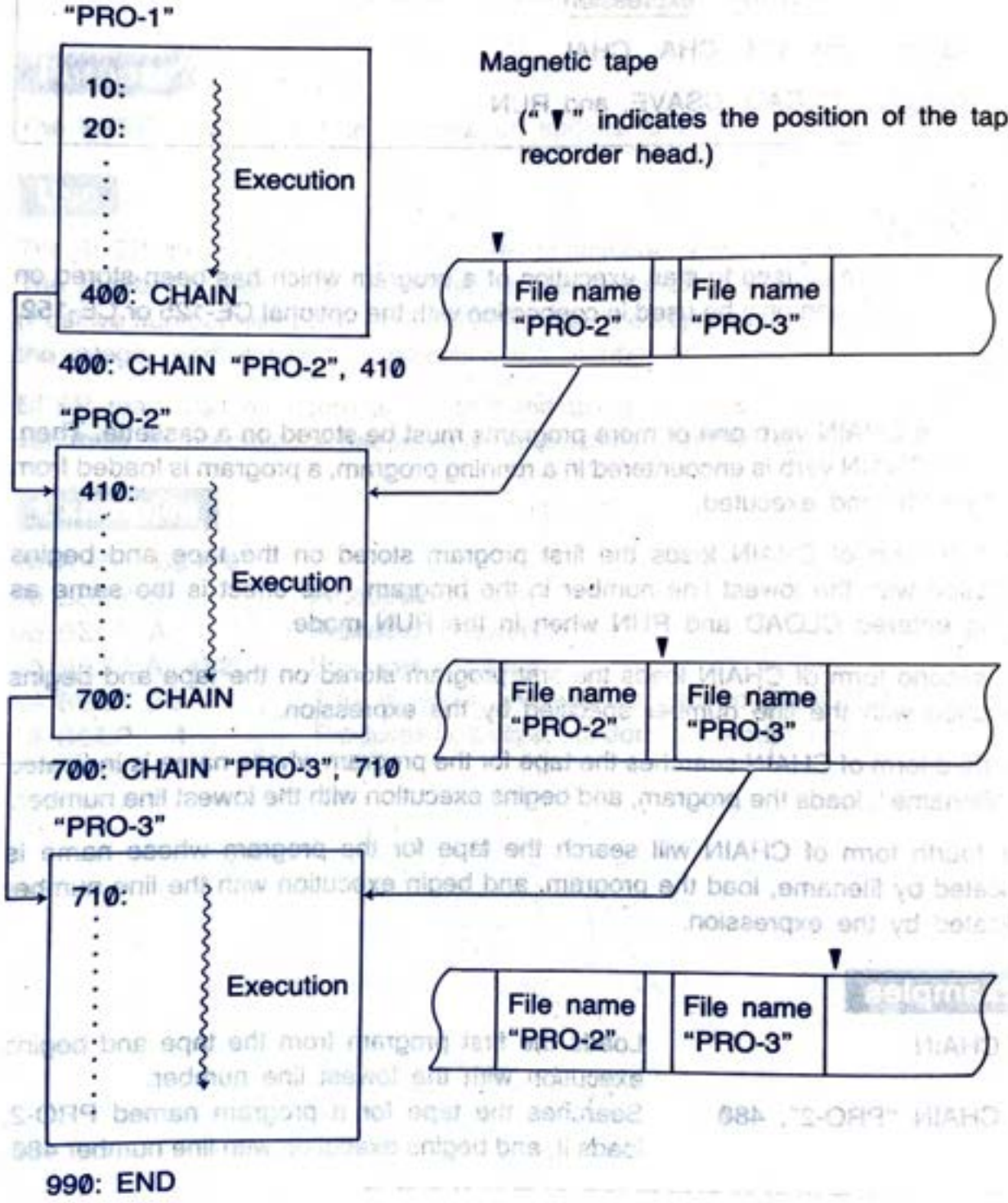
The third form of CHAIN searches the tape for the program whose name is indicated by "filename", loads the program, and begins execution with the lowest line number.

The fourth form of CHAIN will search the tape for the program whose name is indicated by filename, load the program, and begin execution with the line number indicated by the expression.

Examples

10 CHAIN	Loads the first program from the tape and begins execution with the lowest line number.
20 CHAIN "PRO-2", 480	Searches the tape for a program named PRO-2, loads it, and begins execution with line number 480.

For example, let's assume you have three program sections named PRO-1, PRO-2, PRO-3. Each of these sections ends with a CHAIN statement.



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.

1 CLEAR

Abbreviations: CL., CLE., CLEA.

See also: DIM

Purpose

The CLEAR verb is used to erase all variables which have been used in the program and to reset all preallocated variables to zero or null.

Use

The CLEAR verb recovers space which is being used to store variables. This might be done when the variables used in the first part of a program are not required in the second part and available space is limited. CLEAR may also be used at the beginning of a program when several programs are resident in memory and you want to clear out the space used by execution of prior programs.

CLEAR does not free up the space used by the variables A–Z, A$–Z$, or A(1)–A(2) (without DIM declaration) since they are permanently assigned (see Chapter 4). CLEAR does reset numeric variables to zero and string variables to null.

Examples

```
10A=5: DIM C(5)
20 CLEAR
```

Frees up the space assigned to C( ) and resets A to zero.

1 **CLS**

Abbreviations: none

See also: CURSOR

Purpose

The CLS command clears the display.

Use

Clears the display and returns the display start position to 0.

Examples

```
10: WAIT20
20: INPUT A$
30: FOR B=0 TO 47
40: CLS
50: CURSOR B
60: PRINT A$
70: NEXT B
```

This program displays the entry while moving it from left to right on the display unit (from the upper line to the lower line). Each time the FOR-NEXT loop of lines 30–70 is executed, the display is cleared with the CLS command, and display start position is shifted with the CURSOR command, and the contents of A$ are displayed with the PRINT command. By writing and clearing the display in this manner, the display can be made to appear to move. (Delete line 40 and execute. Note the difference.)

1 **CURSOR** expression
2 **CURSOR**

Abbreviations: CU., CUR., CURS., CURSO.

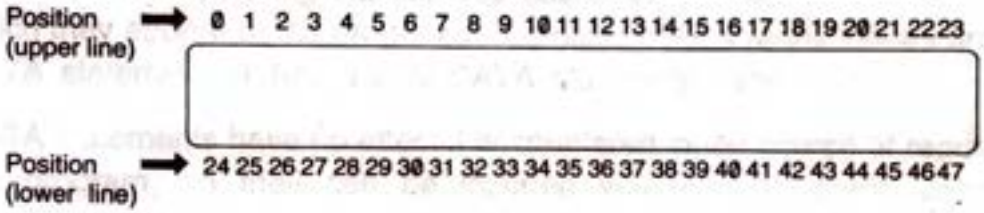
See also: CLS, INPUT, PRINT, PAUSE

Purpose

Specifies the display start position.

Use

Format (1) specifies the display start position of the contents to be displayed with the PRINT command.
The display position is as shown in the following figure.

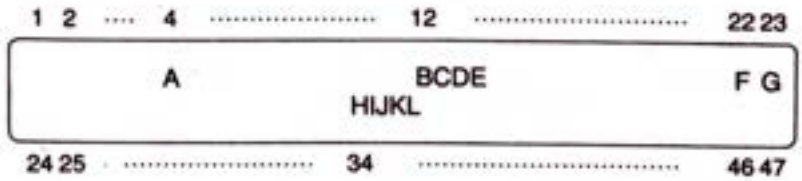


Therefore, let the value of the expression in format (1) be in the range of 0–47. Exceeding this range results in an error (ERROR 9).

Examples

```
10: CURSOR 4 : PRINT "A"
20: CURSOR 22: PRINT "FG"
30: CURSOR 12: PRINT "BCDE"
40: CURSOR 34: PRINT "HIJKL"
```

After starting the program, the following is displayed as the ENTER key is pressed.



When the display start position has been specified with the CURSOR command, the contents of the display in front and behind those displayed by subsequent PRINT commands and PAUSE commands continue to be displayed.
This feature can be used to change only a part of the display for a wide range of applications.
Use the CLS command to clear the display.
Format (2) clears the display start position specification. The CURSOR specifications also work for the INPUT command.

Example:

```
10: WAIT 0
20: PRINT "COST␣␣␣␣VOLUME␣␣PRICE"
30: CURSOR 24:INPUT A
40: CURSOR 31:INPUT B
50: C=A*B
60: CURSOR 38:PRINT C
70: WAIT : PRINT
80: END
```

1 **DATA** expression list
Where: expression list is: expression
or: expression, expression list

Abbreviations: DA., DAT.

See also: READ, RESTORE

Purpose

The DATA verb is used to provide values for use by the READ verb.

Use

When assigning initial values to an array, it is convenient to list the values in a DATA statement and use a READ statement in a FOR ... NEXT loop to load the values into the array. When the first READ is executed, the first value in the first DATA statement is returned. Succeeding READs use succeeding values in the sequential order in which they appear in the program, regardless of how many values are listed in each DATA statement or how many DATA statements are used.

DATA statements have no effect if encountered in the course of regular execution of the program, so they can be inserted wherever it seems appropriate. Many programmers like to include them immediately following the READ which uses them. If desired, the values in a DATA statement can be read a second time by using the RESTORE statement.

Examples

```
10 DIM B(10)
20 WAIT 128
30 FOR I=1 TO 10
40 READ B(I)
50 PRINT B(I)
60 NEXT I
70 DATA 10,20,30,40,50,60
80 DATA 70,80,90,100
90 END
```

Sets up an array.

Loads the values from the DATA statement into B( )
B(1) will be 10, B(2) will be 20, B(3) will be 30, etc.

1 **DEGREE**

Abbreviations: DE., DEG., DEGR., DEGRE.

See also: GRAD and RADIAN

Purpose

The DEGREE verb is used to change the form of angular values to decimal degrees.

Use

The PC-1260/1261 has three forms for representing angular values—decimal degrees, radians and gradient. These forms are used in specifying the arguments to the SIN, COS, and TAN functions and in returning the results from the ASN, ACS, and ATN functions.

The DEGREE function changes the form for all angular values to decimal degree form until a GRAD or RADIAN verb is used. The DMS and DEG functions can be used to convert decimal degrees to degree, minute, second form and vice versa.

Examples

```
10 DEGREE
20 X=ASN 1      X now has a value of 90, i.e. 90 degrees, the Arcsine of 1.
30 PRINT X
```

1 **DIM** dim list

Where: dim list	is: dimension spec.
	or: dimension spec., dim list
and: dimension spec.	is: numeric dim spec.
	or: string dim spec.
and: numeric dim spec	is: numeric name (size)
and: string dim spec	is: string name (dims)
	or: string name (dims)*len
and: numeric name	is: valid numeric variable name
and: string name	is: valid string variable name
and: dims	is: size
	or: size, size
and: size	is: number of elements
and: len	is: length of each string in a string array

Abbreviations: D., DI.

Purpose

The DIM verb is used to reserve space for numeric and string array variables.

Use

Except for the array of the form; A( ), A$( ), two-character ( ), and two-character$ ( ), a DIM verb must be used to reserve space for any array variable.

The maximum number of dimensions in any array is two; the maximum size of any one dimension is 255. In addition to the number of elements specified in the dimension statement, one additional "zeroeth" element is reserved. For example, DIM B(3) reserves B(0), B(1), B(2), and B(3). In two dimensional arrays there is an extra "zeroeth" row and column.

In string arrays one specifies the size of each string element in addition to the number of elements. For example, DIM B$(3)*12 reserves space for 4 strings which are each a maximum of 12 characters long. If the length is not specified each string can contain a maximum of 16 characters.

When a numeric array is dimensioned, all values are initially set to zero; in a string array the values are set to null.

For the array A and A$ DIM declaration, refer to the paragraph discussing variables.

Examples

```
10 DIM B(10)
20 DIM C$(4, 4)*10
```

Reserves space for a numeric array with 11 elements.

Reserves space for a two dimensional string array with 5 rows and 5 columns; each string will be a maximum of 10 characters.

1 **END**

Abbreviations: E., EN.

Purpose

The END verb is used to signal the end of a program.

Use

When multiple programs are loaded into memory at the same time a mark must be included to indicate where each program ends so that execution does not continue from one program to another. This is done by including an END verb as the last statement in the program.

Examples

```
10 PRINT "HELLO"
20 END
30 PRINT "GOODBYE"
40 END
```

With these programs in memory a 'RUN 10' prints 'HELLO', but not 'GOODBYE'. 'RUN 30' prints 'GOODBYE'.

1 **FOR** numeric variable=expression 1 **TO** expression 2
2 **FOR** numeric variable=expression 1 **TO** expression 2
STEP expression 3

Abbreviations: F. and FO.; STE.

See also: NEXT

Purpose

The FOR verb is used in combination with the NEXT verb to repeat a series of operations a specified number of times.

Use

The FOR and the NEXT verbs are used in pairs to enclose a group of statements which are to be repeated. The first time this group of statements is executed the loop variable (the variable named immediately following the FOR) has the value of expression 1.

When execution reaches the NEXT verb the loop variable is increased by the step size and then this value is tested against expression 2. If the value of the loop variable is less than or equal to expression 2, the enclosed group of statements is executed again, starting with the statement following the FOR. In the first form the step size is 1; in the second form the step size is given by expression 3. If the value of the loop variable is greater than expression 2, execution continues with the statement which immediately follows the NEXT. Because the comparison is made at the end, the statements within a FOR/NEXT pair are always executed at least once.

Expression 1, expression 2, and expression 3 must be in the range of −9.999999999E99 ~ 9.999999999E99. When expression 3 is in zero FOR/NEXT loop will be infinite.

The loop variable may be used within the group of statements, for example as an index to an array, but care should be taken in changing the value of the loop variable.

Programs should be written so that they never jump from outside a FOR/NEXT pair to a statement within a FOR/NEXT pair. Similarly, programs must never leave a FOR/NEXT pair by jumping out. Always exit a FOR/NEXT loop via the NEXT statement. To do this, set the loop variable to a value higher than expression 2.

The group of statements enclosed by a FOR/NEXT pair can include another pair of FOR/NEXT statements which use a different loop variable as long as the enclosed pair is completely enclosed; i.e., if a FOR statement is included in the group, the

matching NEXT must also be included. FOR/NEXT pairs may be "nested" up to five levels deep.

Examples

```
10 FOR I=1 TO 5
20 PRINT I
30 NEXT I
```

This group of statements prints the numbers 1, 2, 3, 4, 5.

```
40 FOR N=10 TO 0 STEP-1
50 PRINT N
60 NEXT N
```

This group of statements counts down 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0.

```
70 FOR N=1 TO 10
80 X=1
90 FOR F=1 TO N
100 X=X*F
110 NEXT F
120 PRINT X
130 NEXT N
```

This group of statements computes and prints N factorial for the numbers from 1 to 10.

1 **GOSUB** expression

Abbreviations: GOS., GOSU.

See also: GOTO, ON ... GOSUB, ON ... GOTO, RETURN

Purpose

The GOSUB verb is used to execute a BASIC subroutine.

Use

When you wish to execute the same group of statements several times in the course of a program or use a previously written set of statements in several programs, it is convenient to use the BASIC capability for subroutines using the GOSUB and RETURN verbs.

The group of statements is included in the program at some location where they are not reached in the normal sequence of execution. A frequent location is following the END statement which marks the end of the main program. At those locations in the main body of the program—where subroutines are to be executed, include a GOSUB statement with an expression which indicates the starting line number of the subroutine. The last line of the subroutine must be a RETURN. When GOSUB is executed, the PC-1260/1261 transfers control to the indicated line number and processes the statements until a RETURN is reached. Control is then transferred back to the statement following the GOSUB.

A subroutine may include a GOSUB. Subroutines may be "nested" in this fashion up to 10 levels deep.

The expression in a GOSUB statement may not include a comma, e.g., 'A(1, 2)' cannot be used. Since there is an ON ... GOSUB structure for choosing different subroutines at given locations in the program, the expression usually consists of just the desired line number. When a numeric expression is used it must evaluate to a valid line number, i.e., 1 to 65279, or an ERROR 4 will occur.

Example

```
10 GOSUB 100
20 END
100 PRINT "HELLO"
110 RETURN
```

When this program is run it prints the word 'HELLO' one time.

1 **GOTO** expression

Abbreviations: G., GO., GOT.

See also: GOSUB, ON ... GOSUB, ON ... GOTO

Purpose

The GOTO verb is used to transfer control to a specified line number.

Use

The GOTO verb transfers control from one location in a BASIC program to another location. Unlike the GOSUB verb, GOTO does not "remember" the location from which the transfer occurred.

The expression in a GOTO statement may not include a comma, e.g., 'A(1, 2)' cannot be used. Since there is an ON ... GOTO structure for choosing different destinations at given locations in the program, the expression usually consists of just the desired line number. When a numeric expression is used, it must evaluate to a valid line number, i.e., 1 to 65279, or an ERROR 4 will occur.

Well designed programs usually flow simply from beginning to end, except for subroutines executed during the program. Therefore, the principal use of the GOTO verb is as a part of an IF ... THEN statement.

Examples

```
10 INPUT A$
20 IF A$="Y" THEN GOTO 50
30 PRINT "NO"
40 GOTO 60
50 PRINT "YES"
60 END
```

This program prints 'YES' if a 'Y' is entered and prints 'NO' if anything else is entered.

1 **GRAD**

Abbreviations: GR., GRA.

See also: DEGREE and RADIAN

Purpose

The GRAD verb is used to change the form of angular values to gradient form.

Use

The PC-1260/1261 has three forms for representing angular values—decimal degrees, radians, and gradient. These forms are used in specifying the arguments to the SIN, COS, and TAN functions and in returning the results from the ASN, ACS, and ATN functions.

The GRAD function changes the form for all angular values to gradient form until a DEGREE or RADIAN verb is used. Gradient form represents angular measurement in terms of percent gradient, i.e., a 45° angle is a 50^{g} gradient.

Examples

```
10 GRAD
20 X=ASN 1
30 PRINT X
```

X now has a value of 100, i.e., a 100^{g} gradient, the Arcsine of 1.

1 **IF** condition **THEN** statement
2 **IF** condition statement

Abbreviations: none for IF, T., TH., THE.

Purpose

The IF ... THEN verb pair is used to execute or not to execute a statement depending on conditions at the time the program is run.

Use

In the normal running of BASIC programs, statements are executed in the sequence in which they occur. The IF ... THEN verb pair allows decisions to be made during execution so that a given statement is executed only when desired. When the condition part of the IF statement is true, the statement is executed; when it is false, the statement is skipped.

The condition part of the IF statement can be any relational expression as described in Chapter 4. It is also possible to use a numeric expression as a condition, although the intent of the statement will be less clear. Any expression which evaluates to zero or a negative number is considered false; any which evaluates to a positive number is considered true.

The statement which follows the THEN may be any BASIC statement, including another IF ... THEN. If it is a LET statement, the LET verb itself must appear. Unless the statement is an END, GOTO, or ON ... GOTO, the statement following the IF ... THEN statement is the next one executed regardless of whether the condition is true.

The two forms of the IF statement are identical in action, but the first form is clear.

Examples

```
10 INPUT "CONTINUE?"; A$
20 IF A$="YES" THEN GOTO 10
30 IF A$="NO" THEN GOTO 60
40 PRINT "YES OR NO, PLEASE"
50 GOTO 10
60 END
```

This program continues to ask 'CONTINUE?' as long as 'YES' is entered; it stops if 'NO' is entered, and complains otherwise.

1 **INPUT** input list

Where: input list	is: input group
	or: input group, input list
and: input group	is: var list
	or: prompt, var list
	or: prompt; var list
and: var list	is: variable
	or: variable, var list
and: prompt	is: any string constant

Abbreviations: I., IN., INP., INPU.

See also: INPUT #, READ, CURSOR, PRINT

Purpose

The INPUT verb is used to enter one or more values from the keyboard.

Use

When you want to enter different values each time a program is run, use the INPUT verb to enter these values from the keyboard.

In its simplest form the INPUT statement does not include a prompt string; instead a question mark is displayed on the left edge of the display. A value is then entered, followed by the [ENTER] key. This value is assigned to the first variable in the list. If other variables are included in the same INPUT statement, this process is repeated until the list is exhausted.

If a prompt is included in the INPUT statement, the process is exactly the same except that, instead of the question mark, the prompt string is displayed at the left edge of the display. If the prompt string is followed by a semicolon, the cursor is positioned immediately following the prompt. If the prompt is followed by a comma, the prompt is displayed. Then when a key is pressed, the display is cleared and the first character of the input is displayed at the left edge.

When a prompt is specified and there is more than one variable in the list following it, the second and succeeding variables are prompted with the question mark. If a second prompt is included in the list, it is displayed for the variable which immediately follows it.

When the display starting position has been specified using the CURSOR command

before executing the INPUT command, the input prompt or "?" will be displayed from that position. Further, when a PRINT command ending with a ";" is being executed, its values are displayed after which the input prompt is displayed.

If the [ENTER] key is pressed and no input is provided, the variable retains the value it had before the INPUT statement.

Examples

10 INPUT A	Clears the display and puts a question mark at the left edge.
20 INPUT "A=";A	Displays 'A=' and waits for input data.
30 INPUT "A=",A	Displays'A='. When data is input 'A=' disappears and the data is displayed starting at left edge.
40 INPUT "X=?";X,"Y=?";Y	Displays 'X=?' and waits for first input. After [ENTER] is pressed, display is cleared and 'Y=?' is displayed at left edge.

1 **INPUT #** var list
2 **INPUT #** "filename"; var list

Where: var list is: variable
or: variable, var list

Abbreviations: I. #, IN. #, INP. #, INPU. #

See also: INPUT, PRINT #, READ

Purpose

The INPUT # verb is used to enter values from the cassette tape.

Use and Examples

The following variable types can be specified in the INPUT # statement:
(1) Fixed variables—A, B, C, A(7), D*, A(20)*, etc.
(2) Simple variables—AA, B3, CP$, etc.
(3) Array variables—S(*), HP(*), K$(*), etc.

1) Transferring data to fixed variables

To transfer data from tape to fixed variables, specify the variable names in the INPUT # statement.

```
INPUT # "DATA 1" ; A, B, X, Y
```

This statement transfers data from the cassette file named "DATA 1" to the variables A, B, X, and Y in that order.

To fill all the available fixed variables and, if defined, extended variables (A(27) and beyond) with data transferred from tape, specify the first variable with an asterisk (*) subscripted to it.

```
INPUT # "D-2"; D*
```

This statement transfers the contents of the tape file "D-2" to variables D through Z and to A(27) and beyond.

```
INPUT # A(10)* (without DIM declaration)
```

This statement transfers the data of the first file found after the tape was started, to the variables A(10) and beyond (to J through Z and A(27) and beyond).

Note 1. If an array named A is already defined by the DIM statement, it is not possible to define subscripted fixed variables in the form of A( ). Also no

data transfer to variables A(27) and beyond will occur.

Note 2. Data transfer to fixed variables and extended variables (A(27) and beyond) will continue until the end of the source data file on the tape is reached, but if the computer's memory becomes full, an error (ERROR 6) results.

2) Data transfer to simple variables

Data in a tape file can be transferred to simple variables by specifying the desired simple variable names in the INPUT # statement.

```
INPUT # "DM-1"; AB, Y1, XY$
```

This statement transfers data from the tape file named "DM-1" to simple variable AB, Y1, and XY$.

Note 1. Numeric data must be transferred to numeric simple variables, and character data must be simple character variables. Cross-transfer is not allowed.

Note 2. Locations for simple variables must be set aside in the program data area before the INPUT # statement is executed. If not, an error will result. Use assignment statements to reserve the locations for simple variables.

```
AA=0 [ENTER]
B1$="A" [ENTER]

INPUT # AA, B1$ [ENTER]
```

Use appropriate numeric values or characters in assignment statements to reserve locations for variables.

3) Data transfer to array variables

To transfer data from a tape file to array variables, specify the array name in the INPUT # statement in the form of array name(*).

```
50 DIM B(5)
60 INPUT # "DS-4"; B(*)
```

This statement transfers data from the tape file named "DS-4" to the variables (B(0) through B(5)) in array B.

Note 1. Numeric data must be transferred to numeric array variables with the same length as that of the data, character data must be transferred to character array variables with the same length as that of the data. If this rule is not observed, an error will result.

Note 2. Locations for array variables must be set aside in the program data area before the INPUT # statement is executed. If not, an error will result. Use the DIM statement to define the array in advance.

–CAUTION–

If the number of variables specified in the INPUT # statement does not agree with the amount of data recorded on the tape, the following will happen:

* If the number of pieces of data recorded on the tape file (to be transferred) is greater than the number of specified variables, data transfer will be performed to the last variable, and the remaining data will be ignored.
* If the number of pieces of data recorded in the tape file (to be transferred) is smaller than the number of specified variables, all the file data will be transferred to the variables to the end of the file, and the remaining variables will maintain their previous contents. In this case, however, the computer will continue to wait for data transfer from the tape. To halt this state, you should operate the ON BRK key.
* If the INPUT # statement is executed with no variable name specified in it, an error (ERROR 1) will result.

1 **LET** variable=expression
2 variable=expression

Abbreviations: LE.

Purpose

The LET verb is used to assign a value to a variable.

Use

The LET verb assigns the value of the expression to the designated variable. The type of the expression must match that of the variable, i.e. only numeric expressions can be assigned to numeric variables and only string expressions can be assigned to string variables. In order to convert from one type to the other, one of the explicit type conversion functions, STR$ or VAL, must be used.

The LET verb may be omitted in all LET statements except those which appear in the THEN clause of an IF ... THEN statement. In this one case the LET verb must be used.

Examples

```
10 I=10                          Assignes the value 10 to I.
20 A=5*I                         Assigns the value 50 to A.
30 X$=STR$ (A)                   Assigns the value '50' to X$
40 IF I>=10 THEN LET Y$=X$+".00" Assigns the value '50.00' to Y$.
```

1 **LPRINT** print expr
2 **LPRINT** print expr, print expr, print expr, print expr
3 **LPRINT** print list
4 **LPRINT** print list;
5 **LPRINT**

Where: print list is: print expr
or: print expr; print list
and: print expr is: expression
or: USING clause; expression

The USING clause is described separately under USING

Abbreviations: LP., LPR., LPRI., LPRIN.

See also: PAUSE, PRINT, USING, and WAIT

Purpose

The LPRINT verb is used to print information oh the printer of the optional CE-126P or CE-125.

Use

The LPRINT verb is used to print prompting information, results of calculations, etc. The first form of the LPRINT statement prints a single value. If the expression is numeric, the value will be printed at the far right edge of the paper. If it is a string expression, the print is made starting at the far left. In format (2), one print line of 24 columns is divided into 12 columns on the left and right. The specified values are printed in sequence. In other words, the first specified value is printed on the left side of the first line, the second specified value on the right side of the first line, the third specified value on the left side of the second line, and the fourth specified value on the right side of the second line.

The numeric value within the 12 column (digit) range is printed at the far right end of the display, while the character value (string value) is printed starting at the far left. If the value to be printed exceeds 12 columns, numeric values are printed with the least significant digit(s) of the decimal fraction truncated so value is within 12 digits, and characters are printed from the first 12 (from the left).

Note 1: The number of the values (items) specified in format (2) must be within 2–4.
Note 2: Even if the display starting position has been specified in format (2) with format (4) or the CURSOR command, the specification will be cleared and

printing will be performed in the form as shown above. The values are printed from the left edge of the paper.

In format (3), the values are printed from the left edge of the paper. If the value to be printed exceeds 24 columns, a new line is automatically performed. Up to a maximum of 79 characters can be printed. An error occurs if the 79th column is in the middle of a numeric value.

In format (4), at the specified printing value, the value specified in the LPRINT command to be executed next will be printed in succession. However, due to the structure of the printer, printing occurs in the following instances:

(1) when the value to be printed exceeds 24 columns.
(2) when an LPRINT command not ending with a ";" has been executed.
(3) when the program execution ends.

In format (5), no printing occurs but the paper is fed one line.

Examples

```
10 A=10:B=20:X$="ABCDE":Y$="XYZ"
20 LPRINT A
30 LPRINT X$
40 LPRINT A,B,X$,Y$
50 LPRINT X$:A:B
60 LPRINT
70 LPRINT A*B:
80 LPRINT Y$
```

1 **NEXT** numeric variable

Abbreviations: N., NE., NEX.

See also: FOR

Purpose

The NEXT verb is used to mark the end of a group of statements which are being repeated in a FOR/NEXT loop.

Use

The use of the NEXT verb is described under FOR. The numeric variable in a NEXT statement must match the numeric variable in the corresponding FOR.

Examples

```
10 FOR I=1 TO 10
20 PRINT I
30 NEXT I
```

Print the numbers from 1 to 10 each time the ENTER is pressed.

1 **ON** expression **GOSUB** expression list

Where: expression list is: expression
or: expression, expression list

Abbreviations: O.; GOS., GOSU.

See also: GOSUB, GOTO, ON ... GOTO

Purpose

The ON ... GOSUB verb is used to execute one of a set of subroutines depending on the value of a control expression.

Use

When the ON ... GOSUB verb is executed the expression between ON and GOSUB is evaluated and reduced to an integer. If the value of the integer is 1, the first subroutine in the list is executed as in a normal GOSUB. If the expression is 2, the second subroutine in the list is executed, and so forth. After the RETURN from the subroutine, execution proceeds with the statement which follows the ON ... GOSUB.

If the expression is zero, negative, or larger than the number of subroutines provided in the list, no subroutine is executed and execution proceeds with the next line of the program.

NOTE: Commas may not be used in the expressions following the GOSUB. The PC-1260/1261 cannot distinguish between commas in expressions and commas between expressions.

Examples

```
10 INPUT A
20 ON A GOSUB 100, 200, 300
30 END
100 PRINT "FIRST"
110 RETURN
200 PRINT "SECOND"
210 RETURN
300 PRINT "THIRD"
310 RETURN
```

An input of 1 prints "FIRST"; 2 prints "SECOND"; 3 prints "THIRD". Any other input does not produce any print.

1 **ON** expression **GOTO** expression list

Where: expression list is: expression
or: expression, expression list

Abbreviations: O.; G., GO., GOT.

See also: GOSUB, GOTO, ON ... GOSUB

Purpose

The ON ... GOTO verb is used to transfer control to one of a set of locations depending on the value of a control expression.

Use

When the ON ... GOTO verb is executed the expression between ON and GOTO is evaluated and reduced to an integer. If the value of the integer is 1, control is transferred to the first location in the list. If the expression is 2, control is transferred to the second location in the list; and so forth.

If the expression is zero, negative, or larger than the number of locations provided in the list, execution proceeds with the next line of the program.

NOTE: Commas may not be used in the expressions following the GOTO. The PC-1260/1261 cannot distinguish between commas in expressions and commas between expressions.

Examples

```
10 INPUT A
20 ON A GOTO 100,200,300
30 GOTO 900
100 PRINT "FIRST"
110 GOTO 900
200 PRINT "SECOND"
210 GOTO 900
300 PRINT "THIRD"
310 GOTO 900
900 END
```

An input of 1 prints 'FIRST'; 2 prints 'SECOND'; 3 prints 'THIRD'. Any other input does not produce any print.

1 **PAUSE** print expr
2 **PAUSE** print expr, print expr, print expr, print expr
3 **PAUSE** print list
4 **PAUSE** print list;
5 **PAUSE**

Where: print list is: print expr
or: print expr; print list
and: print expr is: expression
or: USING clause; expression

The USING clause is described separately under USING

Abbreviations: PAU., PAUS.

See also: LPRINT, PRINT, CURSOR, USING, and WAIT

Purpose

The PAUSE verb is used to print information on the display for a short period.

Use

The PAUSE verb is used to display prompting information, results of calculations, etc. The operation of PAUSE is identical to PRINT except that after PAUSE the PC-1260/1261 waits for short preset interval of about .85 seconds and then continues execution of the program without waiting for the ENTER key or the WAI interval.

The first form of the PAUSE statement displays a single value. If the expression is numeric, the value is printed at the far right end of the display. If it is a string expression, the display is made starting at the far left.

However, when the display starting position is specified using format (4) or the CURSOR command, the display starts from that position.
In format (2), the display is divided into 4 groups of 12 columns each. The first specified value is displayed on the upper line at the left side of the display, the second specified value on the upper line at the right side, the third specified value on the lower line at the left side, and the fourth specified value on the lower line at the right side. In this case too, within a range of 12 columns, the numeric value of an expression is displayed from the right end of the display and characters are displayed from the left side.

Even if the display starting position has been specified in format (2) due to the execution of format (4) or the CURSOR command, the specification will be cleared and printing will be performed in the form as shown above.

- The number of the values (items) specified in format (2) must be within 2—4.
- If the specified value exceeds 12 columns, the following is performed.
  1) When the numeric value exceeds 12 digits (when the decimal fraction in the exponential display is 8 digits or more), the least significant digit(s) is truncated.
  2) When the characters exceed 12 columns, only the first 12 characters (from the left) are displayed.

In format (3), the specified value is displayed continuously from the upper line at the left side of the display. However, if the display starting position has been specified using format (4) or the CURSOR command, the display starts from that position.

Note: If the value to be displayed in format (3) exceeds the number of displayable columns, the excess portion is not displayed. When the value to be displayed exceeds 79 columns, and error occurs if the 79th column is in the middle of a numeric value.

In format (4), the specified value is displayed from the upper line at the left side of the display. The column following the end of this displayed value is specified as the display starting position for display commands such as for the next PRINT command.
In format (5), the previously displayed value is displayed as is.

Examples

```
10 A=10:B=20:X$="ABCDEF":
   Y$="XYZ"
```

Display

```
20 PAUSE A
```

```
10.
```

```
30 PAUSE X$
```

```
ABCDEF
```

```
40 PAUSE X$, Y$, A, B
```

```
ABCDEF    XYZ
      10.          20.
```

```
50 PAUSE Y$;X$;
```

```
XYZABCDEF
```

```
60 PAUSE A*B
```

```
XYZABCDEF200.
```

```
1 PRINT print expr
2 PRINT print expr, print expr, print expr, print expr
3 PRINT print list
4 PRINT print list;
5 PRINT
6 PRINT=LPRINT
7 PRINT=PRINT
```

Where: print list is: print expr
or: print expr; print list
and: print expr is: expression
or: USING clause; expression

The USING clause is described separately under USING

Abbreviations: P., PR., PRI., PRIN.

See also: LPRINT, PAUSE, CURSOR, USING, and WAIT

Purpose

The PRINT verb is used to print information on the display or on the printer of the CE-126P or CE-125.

Use

The PRINT verb is used to display prompting information, results of calculations, etc. The first form of the PRINT statement displays a single value. If the expression is numeric, the value is printed at the far right end of the display. If it is a string expression, the display is made starting at the far left.

However, when the display starting position is specified using format (4) or the CURSOR command, the display starts from that position. In format (2), the display is divided into 4 groups of 12 columns each. The first specified value is displayed on the upper line at the left side of the display, the second specified value on the upper line at the right side, the third specified value on the lower line at the left side, and the fourth specified value on the lower line at the right side. In this case too, within a range of 12 columns, the value of an expression is displayed from the right end of the display and characters are displayed from the left side.

Even if the display starting position has been specified in format (2) due to the execution of format (4) or the CURSOR command, the specification will be cleared and printing will be performed in the form as shown above.

- The number of the values (items) specified in format (2) must be within 2–4.
- If the specified value exceeds 12 columns, the following is performed.
  1) When the numeric value exceeds 12 digits (when the decimal fraction in the exponential display is 8 digits or more), the least significant digit(s) is truncated.
  2) When the characters exceed 12 columns, only the first 12 characters (from the left) are displayed.

In format (3), the specified value is displayed continuously from the upper line at the left side of the display. However, if the display starting position has been specified using format (4) or the CURSOR command, the display starts from that position.

Note: If the value to be displayed in format (3) exceeds the number of displayable columns, the excess portion is not displayed. When the value to be displayed exceeds 79 columns, an error (ERROR 6) occurs if the 79th column is in the middle of a numeric value.

In format (4), the specified value is displayed from the upper line at the left side of the display. The column following the end of this displayed value is specified as the display starting position of display commands such as for the next PRINT command.

In format (5), the previously displayed value is displayed as is.

The sixth and seventh forms of the PRINT statement do no printing. The sixth form causes all PRINT statements which follow it in the program to be treated as if they were LPRINT statements. The seventh form resets this condition so that the PRINT statements will again work with the display.

While it is possible to write PRINT statements which would display more than 48 characters, only the leftmost 48 appear in the display. There is no way to see the other characters.

Examples

```
10 A=123:B=5/9:X$="ABCDEF":
   Y$="VWXYZ"
20 PRINT X$,B
```

Display

```
ABCDEF          5.55555E-01
```

```
30 PRINT A;B
```

```
 123.5.555555556E-01
```

```
40 PRINT X$;A;
```

```
ABCDEF123.
```

```
50 PRINT Y$;B
```

```
ABCDEF123.VWXYZ5.5555555
56E-01
```

1 **PRINT** # "var list"
2 **PRINT** # "filename" ; var list

Where: var list is: variable
or: variable, var list

Abbreviations: P. #, PR. #, PRI. #,PRIN. #

See also: INPUT #, PRINT, READ

Purpose

The PRINT # verb is used to store values on the cassette tape.

Use and Examples

The following variable types can be used for variable names:
(1) Fixed variables—A, B, X, A(26), C*, A(10)*,etc.
(2) Simple variables—AA, B2, XYS, etc.
(3) Array variables—B(*), CD(*), N$(*), etc.

1) Saving fixed variable contents onto tape

The contents of fixed variables can be saved onto tape by specifying the desired variable names (separated by commas) in the PRINT # statement.

```
PRINT # "DATA 1"; A, B, X, Y
```

This statement saves contents of variables A, B, X, and Y into tape file named "DATE 1".

If you wish to save the contents of the specified fixed variable and all the subsequent fixed variables, subscript that variable name with an asterisk*.

PRINT # "D-2"; D* — This statement saves the contents of fixed variables D through Z (and of extended variables A(26) and beyond, if defined) into the tape file named "D-2".

PRINT # E, X$, A(30)* — This statement saves the contents of the fixed variables E and X$ and of the extended variables A(30) and all the remaining variables, onto the tape.

Note: Subscripted fixed variable names A(1) through A(26) can be specified in the PRINT # statement in much the same way as A through Z (or A$ through Z$). However, if array A is already defined by the DIM statement, A( ) cannot be used to define subscripted fixed variables.

2) Saving simple variable (two-character variable) contents

The contents of simple variables can be saved onto tape by specifying the desired variable names.

```
PRINT # "DM-1"; AB, Y1, XY$
```

This statement saves the contents of the simple variables AB, Y1, and XY$ into the tape file named 'DM-1'.

3) Saving array variable contents

The contents of all variables of a specific array can be saved onto tape by specifying the array name subscripted by an asterisk enclosed in parentheses(*).

```
PRINT # "DS-2";X(*),Y$(*)
```

This statement saves the contents of all the elements (X(0),X(1), ...) of the array X, and of all the elements (X$(0), Y$(1), ...) of the array Y$, into the tape file name 'DS-2'.

Note: It is not possible to save the contents of only one or more specific elements of an array. While fixed variables or subscripted fixed variables allow you to save only specific parts of them, an array (such as A), defined by the DIM statement does not allow you to save only a specific part of it.

* If the PRINT # statement is executed with no variable names specified, an error (ERROR1) will result.

—CAUTION—

The locations for extended variables such as A(27) and beyond, simple variables, and/or array variables must be set aside in the program/data area before the PRINT # statement is executed. Otherwise, the execution of the PRINT # statement for undefined variables will result in an error.

1 **RADIAN**

Abbreviations: RAD., RADI., RADIA.

See also: DEGREE and GRAD

Purpose

The RADIAN verb is used to change the form of angular values to radian form.

Use

The PC-1260/1261 has three forms for representing angular values—decimal degrees, radians, and gradient. These forms are used in specifying the arguments to the SIN, COS, and TAN functions and in returning the results from the ASN, ACS, and ATN functions.

The RADIAN function changes the form for all angular values to radian form until a DEGREE or GRAD verb is used. Radian form represents angles in terms of the length of the arc with respect to a radius, i.e., 360° is 2 PI radians since the circumference of a circle is 2 PI times the radius.

Examples

```
10 RADIAN
20 X=ASN 1        X now has a value of 1.570796327 or PI/2, the Arcsine of 1.
30 PRINT X
```

1 RANDOM

Abbreviations: RA., RAN., RAND., RANDO.

Purpose

The RANDOM verb is used to reset the seed for random number generation.

Use

When random numbers are generated using the RND function, the PC-1260/1261 begins with a predetermined "seed" or starting number. The RANDOM verb resets this seed to a new randomly determined value.

The starting seed will be the same each time the PC-1260/1261 is turned on, so the sequence of random numbers generated with RND is the same each time, unless the seed is changed. This is very convenient during the development of a program because it means that the behavior of the program should be the same each time it is run even though it includes a RND function. When you want the numbers to be truly random, the RANDOM statement can be used to make the seed itself random.

Examples

```
10 RANDOM
20 X=RND 10
```

When run from line 20, the value of X is based on the standard seed. When run from line 10, a new seed is used.

1 **READ** variable list
Where: variable list is: variable
or: variable, variable list

Abbreviations: REA.

See also: DATA, RESTORE

Purpose

The READ verb is used to read values from a DATA statement and assign them to variables.

Use

When assigning initial values to an array, it is convenient to list the values in a DATA statement and use a READ statement in a FOR ... NEXT loop to load the values into the array. When the first READ is executed, the first value in the first DATA statement is returned. Succeeding READs use succeeding values in the sequential order in which they appear in the program, regardless of how many values are listed in each DATA statement or how many DATA statements are used.

If desired, the values in a DATA statement can be read a second time by using the RESTORE statement.

Examples

```
10 DIM B (10)
20 WAIT 32
30 FOR I=1 TO 10
40 READ B(I)
50 PRINT B(I)*2;
60 NEXT I
70 DATA 10, 20, 30, 40, 50, 60
80 DATA 70, 80, 90, 100
90 END
```

Set up an array

Loads the values from the DATA statement into B( )—B(1) is 10, B(2) is 20, B(3) is 30, etc.

1 REM remark

Abbreviations: none

Purpose

The REM verb is used to include comments in a program.

Use

Often it is useful to include explanatory comments in a program. These can provide titles, names of authors, dates of last modification, usage notes, reminders about algorithms used, etc. These comments are included by means of the REM statement.

The REM statement has no effect on the program execution and can be included anywhere in the program. Everything following the REM verb in that line is treated as a comment, so the REM verb must be the last statement in a line when multiple statement lines are used.

Examples

```
10 REM THIS LINE HAS NO EFFECT
```

1 **RESTORE**
2 **RESTORE** expression

Abbreviations: RES., REST., RESTO., RESTOR.

See also: DATA, READ

Purpose

The RESTORE verb is used to reread values in a DATA statement or to change the order in which these values are read.

Use

In the regular use of the READ verb the PC-1260/1261 begins reading with the first value in a DATA statement and proceeds sequentially through the remaining values. The first form of the RESTORE statement resets the pointer to the first value of the first DATA statement, so that it can be read again. The second form of the RESTORE statement resets the pointer to the first value of the first DATA statement whose line number is greater than the value of the expression.

Examples

```
10 DIM B(10)
20 WAIT 32
30 FOR I=1 TO 10
40 RESTORE
50 READ B(I)
60 PRINT B(I)*I;
70 NEXT I
80 DATA 20
90 END
```

Sets up an array.

Assign the value 20 to each of the elements of B ( ).

1 **RETURN**

Abbreviations: RE., RET., RETU., RETUR.

See also: GOSUB, ON ... GOSUB

Purpose

The RETURN verb is used at the end of a subroutine to return control to the statement following the originating GOSUB.

Use

A subroutine may have more than one RETURN statement, but the first one executed terminates the execution of the subroutine. The next statement executed will be the one following the GOSUB or ON ... GOSUB which calls the subroutine. If a RETURN is executed without a GOSUB, an Error 5 will occur.

Examples

```
10 GOSUB 100
20 END
100 PRINT "HELLO"
110 RETURN
```

When run this program prints the word "HELLO" once.

1 **STOP**

Abbreviations: S., ST., STO.,

See also: END; CONT command

Purpose

The STOP verb is used to halt execution of a program for diagnostic purposes.

Use

When the STOP verb is encountered in program execution, the PC-1260/1261 execution halts and a message is displayed such as 'BREAK IN 200' where 200 is the number of the line containing the STOP. STOP is used during the development of a program to check the flow of the program or examine the state of variables. Execution may be restarted using the CONT command.

Examples

10 STOP Causes "BREAK IN 10" to appear in the display.

1 TROFF

Abbreviations: TROF.

See also: TRON

Purpose

The TROFF verb is used to cancel the trace mode.

Use

Execution of the TROFF verb restores normal execution of the program.

Examples

```
10 TRON
20 FOR I=1 TO 3
30 NEXT I
40 TROFF
```

When run, this program displays the line numbers 10, 20, 30, 30, 30 and 40 as the ↓ is pressed.

1 **TRON**

Abbreviations: TR., TRO.

See also: TROFF

Purpose

The TRON verb is used to initiate the trace mode.

Use

The trace mode provides assistance in debugging programs. When the trace mode is on, the line number of each statement is displayed after each statement is executed. The PC-1260/1261 then halts and waits for the Down Arrow key to be pressed before moving on to the next statement. The Up Arrow key may be pressed to see the statement which has just been executed. The trace mode continues until a TROFF verb is executed.

Examples

```
10 TRON
20 FOR I=1 TO 3
30 NEXT I
40 TROFF
```

When run, this program displays the line numbers 10, 20, 30, 30, 30 and 40 as the [↓] is pressed.

1 **USING**
2 **USING** "editing specification"
3 **USING** character variable

Abbreviations: U., US., USI., USIN.

See also: LPRINT, PAUSE, PRINT

Further guide to the use of USING is provided in Appendix C

Purpose

The USING verb is used to control the format of displayed or printed output.

Use

The USING verb can be used by itself or as a clause within a LPRINT, PAUSE, or PRINT statement. The USING verb establishes a specified format for output which is used for all output which follows until changed by another USING verb.

The editing specification of the USING verb consists of a quoted string composed of some combination of the following editing characters:

Right justified numerfc field character
. Decimal point.
^ Used to indicate that numbers should be displayed in scientific notation.
& Left justfied alphanumeric field.

For example, "####" is an editing specification for a right justified numeric field with room for 3 digits and the sign. In numeric fields, a location must be included for the sign, even if it will always be positive.

Editing specifications may include more than one field. For example "####&&&&" could be used to print a numeric and a character field next to each other.

If the editing specification is missing, as in format 1, special formatting is turned off and the built-in display rules pertain.

Examples

```
10 A=125: X$="ABCDEF"

20 PRINT USING "##.##^";A

30 PRINT USING "&&&&&&&&";X$

40 PRINT USING "####&&&;A;X$
```

Display

```
1.25E02

ABCDEF

125ABC
```

1 **WAIT**
2 **WAIT** expression

Abbreviations: W., WA., WAI.

See also: PAUSE, PRINT

Purpose

The WAIT verb is used to control the length of time that displayed information is shown before program execution continues.

Use

In normal execution the PC-1260/1261 halts execution after a PRINT command until the [ENTER] key is pressed. The WAIT command causes the PC-1260/1261 to display for a specified interval and then proceed automatically (similar to the PAUSE verb). The expression which follows the WAIT verb determines the length of the interval. The interval may be set to any value from 0 to 65535. Each increment is about one fifty-nine of a second. WAIT 0 is too fast to be read reasonably; WAIT 65535 is about 19 minutes. WAIT with no following expression resets the PC-1260/1261 to the original condition of waiting until the [ENTER] key is pressed.

Examples

```
10 WAIT 59
```

Causes PRINT to wait about 1 second.

FUNCTIONS

Pseudovariables

Pseudovariables are a group of functions which take no argument and are used like simple variables wherever required.

1 INKEY$

INKEY$ is a string pseudovariable which has the value of the last key pressed on the keyboard. [ENTER], [CL], [SHIFT], [DEF], [↑], [↓], [►], and [◄] all have a value of null. INKEY$ is used to respond to the pressing of individual keys without waiting for the ENTER key to end the input.

```
10 A$=INKEY$
20 B=ASC A$
30 IF B=0 THEN GOTO 10
40 IF B ...
```

Lines 40 and beyond contain tests for the key and the actions to be taken. On first executing the program, the value of INKEY$ is null, since the last key pressed was [ENTER]. If INKEY$ is used following PRINT or PAUSE, the contents of the display are read instead of a key press.

1 MEM

MEM is a numeric pseudovariable which has the value of the number of characters of program memory remaining. The available program memory will be the total memory less the space consumed by programs and array variables. MEM may also be used as a command.

Immediately after reset, MEM has a value of 3198 bytes in the PC-1260 and 9342 bytes in the PC-1261.

1 PI

PI is a numeric pseudovariable which has the value of PI. It is identical to the use of the special PI character (π) on the keyboard. Like other numbers the value of PI is kept to 10 digit accuracy (3.141592654).

Numeric Functions

Numeric functions are a group of mathematical operations which take a single numeric value and return a numeric value. They include trigonometric functions, logarithmic functions, and functions which operate on the integer and sign parts of a number. Many dialects of BASIC require that the argument to a function be enclosed in parentheses. The PC-1260/1261 does not require these parentheses, except when it is necessary to indicate what part of a more complex expression is to be included in the argument.

LOG 100+100 will be interpreted as:
(LOG 100)+100 not LOG (100+100).

1 **ABS** numeric expression

ABS is numeric function which returns the absolute value of the numeric argument. The absolute value is the value of a number without regard to its sign. ABS −10 is 10.

1 **ACS** numeric expression

ACS is a numeric function which returns the arccosine of the numeric argument. The arccosine is the angle whose cosine is equal to the expression. The value returned depends on whether the PC-1260/1261 is in decimal degree, radian, or gradient mode for angles. ACS.5 is 60 in the decimal degree mode.

1 **ASN** numeric expression

ASN is a numeric function which returns the arcsine of the numeric argument. The arcsine is the angle whose sine is equal to the expression. The value returned depends on whether the PC-1260/1261 is in decimal degree, radian, or gradient mode for angles. ASN.5 is 30 in the decimal degree mode.

1 **ATN** numeric expression

ATN is a numeric function which returns the arctangent of the numeric argument. The arctangent is the angle whose tangent is equal to the expression. The value returned depends on whether the PC-1260/1261 is in decimal degree, radian, or gradient mode for angles. ATN 1. is 45 in the decimal degree mode.

1 **COS** numeric expression

COS is a numeric function which returns the cosine of the angle argument. The value returned depends on whether the PC-1260/1261 is in decimal degree, radian, or gradient mode for angles. COS 60 is .5 in the decimal degree mode.

1 **DEG** numeric expression

The DEG function converts an angle argument in DMS (Degree, Minute, Second) format to DEG (Decimal Degree) form. In DMS format the integer portion of the number represents the degrees, the first and second digits of the decimal represent the minutes, the third and fourth digits of the decimal represent the seconds, and any further digits represent decimal seconds. For example, 55° 10' 44.5" is represented as 55.10445. In DEG format the integer portion is degrees and the decimal portion is decimal degrees. DEG 55.10445 is 55.17902778.

1 **DMS** numeric expression

DMS is a numeric function which converts an angle argument in DEG format to DMS format (see DEG). DMS 55.17902778 is 55.10445.

1 **EXP** numeric expression

EXP is a numeric function which returns the value of e (2.718281828—the base of the natural logarithms) raised to the value of the numeric argument. EXP 1 is 2.718281828.

1 **INT** numeric expression

INT is a numeric function which returns the integer part of its numeric argument. INT PI is 3.

1 **LN** numeric expression

LN is a numeric function which returns the logarithm to the base e (2.718281828) of its numeric argument. LN 100 is 4.605170186).

1 **LOG** numeric expression

LOG is a numeric function which returns the logarithm to the base 10 of its numeric argument. LOG 100 is 2.

1 **RND** numeric expression

RND is a numeric function which generates random numbers. If the value of the argument is less than one but greater than or equal to zero, the random number is less than one and greater than or equal to zero. If the argument is an integer greater than or equal to 1, the result is a random number greater than or equal to 1 and less than or equal to the argument. If the argument is greater than or equal to 1 and not an integer, the result is a random number greater than or equal to 1 and less than or equal to the smallest integer which is larger than the argument: (In this case, the generation of the random number changes depending on the value of the decimal portion of the argument.):

	Result	
Argument	Lower Bound	Upper Bound
.5	0<	<1
2	1	2
2.5	1	3

The same sequence of random numbers is normally generated because the same "seed" is used each time the PC-1260/1261 is turned on. To randomize the seed, see the RANDOM verb.

1 **SGN** numeric expression

SGN is a numeric function which returns a value based on the sign of the argument. If the argument is positive, the result is 1; if the argument is zero, the result is 0; if the argument is negative, the result is −1. SGN −5 is −1.

1 **SIN** numeric expression

SIN is a numeric function which returns the sine of the angle argument. The value returned depends on whether the PC-1260/1261 is in decimal degree, radian, or gradient mode for angles. SIN 30 is 5.

1 **SQR** numeric expression

SQR is a numeric function which returns the square root of its argument. It is identical to the use of the special square root symbol ($\sqrt{\ }$) on the keyboard. SQR 4 is 2.

1 **TAN** numeric expression

TAN is a numeric function which returns the tangent of its angle argument. The value returned depends on whether the PC-1260/1261 is in decimal degree, radian, or gradient mode for angles. TAN 45 is 1.

String Functions

String functions are a group of operations used for manipulating strings. Some take a string argument and return a numeric value. Some take a string argument and return a string. Some take numeric value and return a string. Some take a string argument and one or two numeric arguments and return a string. Many dialects of BASIC require the argument of a function to be enclosed in parentheses. The PC-1260/1261 does not require these parentheses, except when it is necessary to indicate what part of a more complex expression is to be included in the argument. String functions with two or three arguments all require the parentheses.

1 **ASC** string expression

ASC is a string function which returns the numeric ASCII code value of the first character in its argument. The chart of ASCII codes and their relationship to characters is given in Appendix B. ASC "A" is 65.

1 **CHR$** numeric expression

CHR$ is a string function which returns the character which corresponds to the numeric ASC11 code of its argument. The chart of ASCII codes and their relationship to characters is given in Appendix B. CHR$ 65 is "A".

1 **LEFT$** (string expression, numeric expression)

LEFT$ is a string function which returns the leftmost part of the string first argument. The number of characters returned is determined by the numeric expression. LEFT$ ("ABCDEF", 2) is "AB".

1 **LEN** string expression

LEN is a string function which returns the length of the string argument. LEN "ABCDEF" is 6.

1 **MID$** (string expression, num. exp. 1, num. esp. 2)

MID$ is a string function which returns a middle portion of the string in the first argument. The first numeric argument indicates the first character position to be included in the result. The second numeric argument indicates the number of characters that are to be included. MID$ ("ABCDEF", 2,3) is "BCD".

1 **RIGHT$** (string expression, numeric expression)

RIGHT$ is a string function which returns the rightmost part of the string first argument. The number of characters returned is determined by the numeric argument. RIGHT$ ("ABCDEF", 3) is 'DEF'.

1 **STR$** numeric expression

STR$ is a string function which returns a string which is the character representation of its numeric argument. It is the reverse of VAL. STR$ 1.59 is '1.59'.

1 **VAL** string expression

VAL is a string function which returns the numeric value of its string argument. It is the reverse of STR$. The VAL of a non-number is zero. VAL "1.59" is 1.59.

Note: The character-string convertible by VAL function to a numerical value consists of numerals (0 to 9), symbols (+ and −) and a symbol (E) indicating an exponential portion. No other characters and symbols are included. If a character-string includes other characters and symbols, any character-string on the right of that character-string will be ignored. If included in a character-string, a space is usually regarded as non-existing.

COMMANDS CONCERNING EASY SIMULATION PROGRAMS

Commands for easy simulation programs are available in addition to those for BASIC programs. The following commands will be described.

EQU #
LIST #
LLIST #
MEM #
NEW #

1 **EQU #** numeric expression
2 **EQU #**

Abbreviations: EQ., EQU.
See also: NEW, NEW #

Purpose

Changes the size (capacity) of the easy simulation program area within the memory.

Use

A basic capacity of 128 bytes is available for easy simulation programs. Format (1) can be specified to change the size of this area. In this case, the area can be specified in units (1 unit is 128 bytes) indicated by the value of the expression. In other words, the size of the easy simulation program area is 128 bytes when the value of the expression is 0, 256 bytes when 1, 384 bytes when 2, etc.

- Format (1) is valid only in the PRO mode and when the BASIC program area is completely cleared (such as after the NEW command is executed).
- If the execution of format (1), while an easy simulation program is stored, would result in the clearing of the program because of a smaller easy simulation program area, an error occurs (ERROR 3) and the size is not changed.
- When expanding the easy simulation program area, the memory formerly used for the program data area is then used for the easy simulation program area. Therefore, the size of the program data area decreases by that much.

Easy simulation program area (basic)
Easy simulation program area (additional 1)
Easy simulation program area (additional 2)
Easy simulation program area (additional 3)
Program data area (The program data area decreases as the easy simulation program area increases.)
Fixed valiable area

When this command is executed in the form of format (2), the number of units added to the current easy simulation program area is indicated.

Note: Since the EQU # command is not a function, a numeric expression cannot be used. Execute separately with manual operation.

Examples

EQU # 1 [ENTER] — The capacity of the easy simulation program area is specified to be 256 bytes (128+128×1).

EQU # 0 [ENTER] — Returns the capacity of the easy simulation program to its basic capacity of 128 bytes (128+128×0).

EQU # [ENTER] 3 — When 3 is displayed, it indicates that 3 units or 384 bytes have been added.

1 **LIST #**
2 **LIST #** name

Abbreviations: L. #, LI. #, LIS. #

See also: LIST, LLIST #

Purpose

Displays the easy simulation program stored in the computer onto the display unit.

Use

In format (1), the first written program (among the ones stored) is listed. In format (2), the easy simulation program starting with the specified name is listed. If the easy simulation program starting with the specified name does not exist, an error results (ERROR 4).

Example

LIST # ENTER — The first written program is listed on the dispaly unit.

LIST # total ENTER — The first easy simulation program starting with name "total" is listed on the dispaly unit.

1 **LLIST #**
2 **LLIST #** name

Abbreviations: LL. #, LLI. #, LLIS. #

See also: LIST #, LLIST

Purpose

The easy simulation programs stored in the computer are printed out.

Use

In format (1), all stored easy simulation programs are printed out. In format (2), the easy simulation program starting with the specified name is printed out. If the easy simulation program starting with the specified name does not exist, an error results (ERROR 4).

Examples

LLIST # [ENTER] — All easy simulation programs stored in the computer are printed out.

LLIST # total [ENTER] — The first easy simulation program starting with the name "total" is printed out on the printer.

1 MEM #

Abbreviations: M. #, ME. #

See also: MEM, EQU #

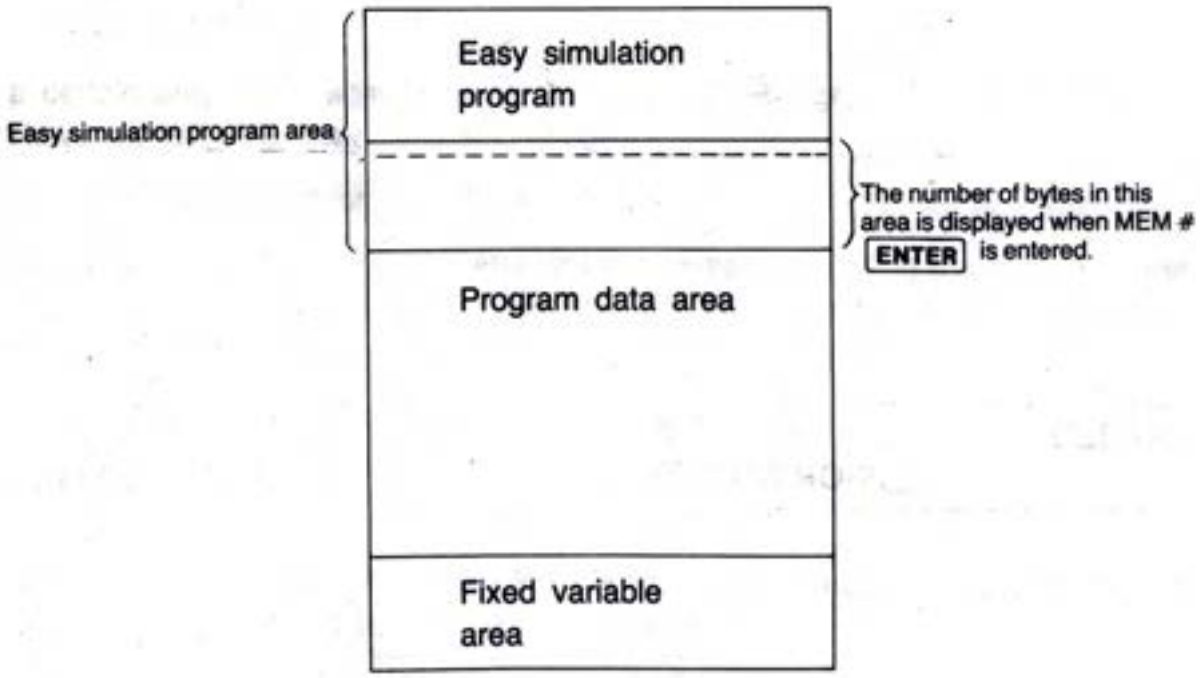
Purpose

Obtains the number of free bytes in the easy simulation program area.

Use

Function which returns the number of bytes of free area (area not written with an easy simulation program) within the easy simulation program area.

Examples



1 **NEW #**

Abbreviations: none

See also: NEW, EQU #

Purpose

Clears the easy simulation program.

Use

The easy simulation programs within the computer are completely cleared when the NEW # command is executed in the PRO mode. The size of the easy simulation program area specified in the EQU # command does not change even if the NEW # command is executed.

CHAPTER 9
EASY SIMULATION PROGRAM

When performing only calculations, a program can be written in the form of an equation without using the BASIC commands. The result can be easily obtained simply by entering the individual values (calculation values) during execution of the calculation. This section describes how to write and execute a program in the calculation format (easy simulation program).

Overview

For example, suppose we want to calculate A=B×C ÷ 2. To manually repeat the calculation for this equation, the key operations for:

value B [*] value C [/] 2 [ENTER]

must be repeated for the desired number of times. As a BASIC program:

```
10 INPUT "B=";B, "C=";C
20 A=B*C/2
30 PRINT "A=";A
40 GOTO 10
```

a certain amount of work is required to write it although it is simple.
However, with the easy simulation program such as:

#A=B*C/2 [ENTER] (In the PROgram mode)

the equation itself can be used as a program simply by adding the # mark (press [SHIFT] and [#]) to the front of the equation. Set the computer to the PROgram mode (PRO) and then enter:

#A [ENTER]
you will see this display:

```
#A                          CLEAR
YES-1 OR NO-2:_
```

This is how you can clear out any formula you have entered. For now, answer the question with a 2 and the > prompt will appear.
Place the switch in the RUN mode, and enter:

A [ENTER]
The calculation starts and the display shows as shown on the right.

```
B          :C          :A
_
```

If the value of B and the value of C are entered, the calculation is executed, and the result is displayed as the value of A.

For example, when B is 17 and C is 23 the calculation proceeds as follows.

17 [ENTER]

```
B          :C          :A
17.        :_
```

12 [ENTER]

```
B          :C          :A
17.        :23.        :195.5
```

If the [ENTER] key is pressed one more time after this, the display returns to the beginning so that the calculation can be repeated. To change individual values, simply use the cursor to insert new values. Instead of a single letter for each variable in the equation, a variable name can be specified using the alphabet such as in:

#AREA=BASE*HEIGHT/2 (BASE: The base of a triangle)

The variable names can be displayed as shown on the right.

```
BASE       :HEIGHT     :AREA
_
```

Writing easy Simulation programs

Easy simulation programs are basically expressed in the following format.

#variable=equation (Equation using numeric values and variables)

For example, equations are expressed as follows.

① #Y=2*X+3 (y=2x+3)
② #AREA=π*RADIUS 2 ($S=\pi r^2$)
③ #PRICE=Cost*(1+Tax/100) (Price=Cost×(1+Tax rate))

Y and X, area and radius, etc, in these easy simulation programs are called variables. These variables are substituted with some numeric values and the equations are calculated. In other words, during a calculation, the variables in manual calculation and in the BASIC program are the same. These variables have features different from the variables introduced until now. These variables can be divided into output variables and input variables.

- **Output variables**
  The output variable, as can be seen from the format of the easy simulation program, is specified on the left side of =, and is substituted with the calculated result.

After the end of the calculation, the result is output (displayed).
In other words, they become variables having an output command. In the easy simulation programs above, Y, area, and price are the output variables.

- **Input variables**

In contrast to the output variables, the input variable is specified on the right side of =. For calculations, these variables must be provided with data (numeric values). Therefore, when executing an easy simulation program, the computer is in the data entry request mode. In other words, they become variables having an input command in order to obtain data.
In the easy simulation programs above, X, radius, cost, and tax are the input variables.
If, in a single easy simulation program, an already specified variable name is specified on the right side of =, the variable which follows does not become an input variable but instead is treated as a value previously entered or as a variable which stores the calculation result.
Next, notes regarding the writing of an easy simulation program are given.

(1) Usable calculation commands and functions

① Calculation commands +, −, *, /
(Logical operators such as >, <, and <= cannot be used.)
② Functions SIN, COS, TAN, ASN, ACS, ATN, LOG, LN, EXP, SQR(or $\sqrt{\ }$), DMS, DEG, INT, ABS, SGN, PI(or π), ^ (Logical functions such as AND and NOT, and the RND functions cannot be used.)
③ Others (,)

(2) Format and length of an easy simulation program

① The # mark is always necessary at the beginning of an easy simulation program.
If the # mark is found at the beginning of an expression when writing a program, the computer determines that it is an easy simulation program. The easy simulation program can be executed by specifying the # mark and the variable name.
In other words, the # mark is a symbol that represents the easy simulation program and also its beginning. Therefore, the # mark can only be used at the beginning of an easy simulation program.
② In an easy simulation program, only the output variable must be on the left side of =. If a calculation command or function command is entered on the left side when writing an easy simulation program, an error (ERROR 1) results. Further, the right side of=is usually a calculation equation.
③ To specify several different output variables and calculation equations within a single easy simulation program, delimit them with: (colons).
Example: #A=A+1: B=2*A*A+4*A+1: C=2*B

④ The length of an easy simulation program (easy simulation program starting with # and ending with [ENTER]), including the # mark and [ENTER] key, is a maximum of 80 bytes. More than 80 bytes cannot be written.

(3) Variables

① A maximum of up to 10 variables can be used in a single easy simulation program.
However, the maximum number of variables usable on the right side of = is 9. If more than 9 variables are used on the right side of =, or if more than 10 variables are used in a single easy simulation program, an error (ERROR 3) results during execution.

② Although upper case and lower case letters and numerals can be used in a variable name, the first character must be an upper case letter.
The spelling of the variable name must not be the same as that of calculation commands, function commands of the computer, or BASIC commands. (However, they can be used if the characters, except the first, are lower case.)
(Example) #SINA=SINA

(Example) #SINA=SINA
↑
Function command

[The variable name must not include a calculation command, function command, or BASIC command.]

[It can be used if the characters INA are lower case]

#TOTAL=DATA1+DATA2
↑ ↑ ↑
BASIC command

[There is no problem if only the first character T is upper case and the rest are lower case.]

Further, a variable name cannot contain the # mark (which indicates the beginning of an easy simulation program) or: (delimiting colon).

③ A variable name of up to 7 columns (7 characters) is valid. If 8 columns or more are specified, the first 7 columns become valid when executing the easy simulation program. The 8th column and beyond are ignored.

④ The variables in an easy simulation program are not the same as those used in BASIC and are stored independently. For example, variable name A is different from the simple numeric variable A handled in BASIC. Therefore, BASIC variables cannot be used in an easy simulation program.

Writing an Easy Simulation Program

Let's write the easy simulation program found on page 000 into the computer. Follow the steps below.

① First, set the power/mode switch to the PRO position (program mode).

② Next, type NEW # and press [ENTER]. This clears the previously stored easy simulation program. However, this step is not necessary if you want to save the previous program.

③ Now enter the easy simulation program.

- #Y=2*X+3

[ENTER]

```
#Y=2*X+3_
```

```
#Y=2*X+3
```

- #AREA=π*RADIUS^2

[ENTER]

```
#AREA=π*RADIUS^2_
```

```
#AREA=π*RADIUS^2
```

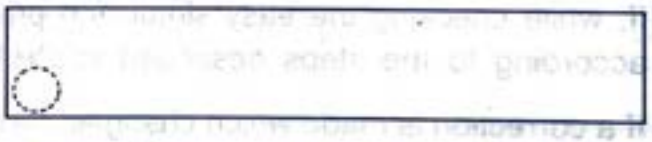
- PRICE=Cost*(1+Tax)

[ENTER]

```
#PRICE=Cost*(1+Tax)_
```

```
#PRICE=Cost*(1+Tax)
```

When the easy simulation program extends over into the second line, a blank column appears at the beginning of the second line.



After entering the easy simulation program as shown, pressing the [ENTER] key writes it to the computer.

Note: If an easy simulation program having the same variable name is written after the # mark, the easy simulation program written after is valid and the one written before is cleared. For example, if an easy simulation program such as:

#AREA=BASE*HEIGHT/2

is already stored and if:

#AREA=π*BASE^2 [ENTER]

is written, the previous easy simulation program will be cleared. Therefore, easy simulation programs starting with the same variable name cannot be stored simultaneously.

Checking the Easy Simulation Program

After writing the easy simulation program, check to see that there are no mistakes. When an easy simulation program is being displayed, the ↑ key or the ↓ key can be used to list the other easy simulation programs.
If the easy simulation program is not being displayed, it can be listed using the LIST # command.

● **Operation of the LIST # command.**
The LIST # command lists the easy simulation program. Enter:

LIST # ENTER

in the program mode lists the easy simulation program which was written first. Further, if executed in the form of:

LIST # Heading name ENTER

(name specified after the # mark at the beginning of the easy simulation program)

the easy simulation program beginning with the specified name is listed.

Example: LIST # AREA ENTER

```
#AREA=π*RADIUS^2
```

If an easy simulation program beginning with the specified name cannot be found, an error results (ERROR 4).

● To check long easy simulation programs which cannot be completely displayed on the display unit, use the → key to move the cursor.

If, while checking the easy simulation program, a mistake is discovered, correct it according to the steps described in CHAPTER ③, ⑤.

If a correction is made which changes the heading name, the program is rewritten as a new and different easy simulation program. The easy simulation program before correction remains. Therefore, if the easy simulation program before the correction is not required, clear it according to the steps described on page 174.

Executing the Easy Simulation Program

Easy simulation programs are executed in the RUN mode. Set the power/mode switch to the RUN position. Then execute in the following format.

Heading name ENTER

(name specified after the mark at the beginning of the easy simulation program)

Let's run the programs we have written.

Example 1:

What is the area of a circle with a radius of 7 cm?

Operation: #AREA [ENTER]

```
RADIUS    :AREA
_
```

Cursor

7

```
RADIUS    :AREA
7_
```

[ENTER]

```
RADIUS    :AREA
7.        :153.93804
```

Example 2:

What is the selling price of a product costing $5000 when it is taxed 5.55%?

Operation: #PRICE [ENTER]

```
Cost      :Tax      :PRICE
_
```

5000 [ENTER]

```
Cost      :Tax      :PRICE
5000.     :_
```

5.55/100

```
Cost      :Tax      :PRICE
5000.     :5.55/100_
```

As shown, values can be entered as expressions.

[ENTER]

```
Cost      :Tax      :PRICE
5000.     :0.0555   :5277.5
```

There may be instances when the result cannot be seen. Check the result with the ► key. Pressing the ◄ key returns it to the original display. As can be seen from running the examples, the name appears on the upper line of the display unit when the execution of the easy simulation program is started. At this time, the displayed columns of the name are automatically set to 7 columns. Therefore, 3 names can be displayed.

7 columns 7 columns 8 columns ——— A maximum of up to 7 colums are displayed for the name.

```
Cost    :Tax    :PRICE
_
```

Therefore, if 4 or more names are used in an easy simulation program, there will be names which cannot be displayed. If a numeric value is entered for the name displayed on the furthest right, the hidden name(s) will appear. If an entry for an item exceeds 7 digits (columns) or if the result (value of the output) exceeds 7 digits, the displayed columns increase due to the excess part. As a result, there may be instances when only 2 values are displayed or part of the contents of the succeeding item cannot be displayed. Therefore, after executing a calculation, be sure to press the ► key and check to see that nothing is hidden. Further, all digits of the entered value or calculated result are usually displayed. However, if the display format is specified (with USING), the display follows the specification. Therefore, even when the number of displayed digits is specified with USING to be 8 digits or more, there may be instances when only 2 values are displayed or part of the contents of the succeeding item cannot be displayed. Specify USING before executing the easy simulation program since it cannot be specified or cleared during execution.

Example:

Example 2 before is specified with USING and then executed.

Operation: USING "######." [ENTER]

(Specifies 6 digits and decimal point to be displayed.)

#PRICE [ENTER]	Cost :Tax :PRICE _
5000 [ENTER]	Cost :Tax :PRICE 5000. :_
5.55/100 [ENTER]	Cost :Tax :PRICE 5000. :0. :5277.

Note: Although the tax in the example above shows 0, this is because it is displayed with the decimal portion truncated as specified with USING. The calculation is executed with 0.0555. Note that this occurs depending on how USING is specified.

USING is cleared by entering USING and [ENTER] or pressing [SHIFT] and then [CA] [CL].

- To stop the execution of an easy simulation program, press [SHIFT] and then [CA] [CL].

Pressing the [CL] key during entry of an input variable simply clears the entry. However, execution can be stopped with the [CL] key when the calculated result has been obtained. Switching the power/mode switch stops execution of an easy simulation program.

Correction a Value Entry and Re-execution (Playback)

This section describes the operations for correcting an entered value during execution of an easy simulation program and then re-executing the program. The entered value can be corrected during entry or after entry when the calculated result (value of the output variable) has been obtained.

(1) Correction during entry of an input variable

① Correcting the value currently being entered

The value currently being entered (value of the variable specified by the cursor) can be cleared with the [CL] key and then corrected. Further, a correction can be made by moving the cursor with the [◄] key or [►] key.

Example 1:

Correcting an erroneous entry of 130 when the value of B is supposed to be 120 for the following easy simulation program. #Total=A+B+C+D.

Execution: RUN mode

#Total [ENTER]

```
A          :B          :C
_
```

100 [ENTER] 130 ← mistake

```
A          :B          :C
100.       :130_
```

Cursor

[◄] [◄]

```
A          :B          :C
100.       :1_0
```

Cursor

2 [ENTER]

```
A          :B          :C
100        :120.       :_
```

② Correcting immediately after completing the entry

Immediately after entering a numeric value using the [ENTER] key, the value can be corrected with the following steps.

Example 2:

In example 1, immediately after entering 135 for C, let's change this value to 145.

135 [ENTER]

```
A          :B          :C
120        :135        :_
```

Cursor

[◄]

```
A          :B          :C
100.       :120        :135
```

If the ◄ key is pressed immediately after entering a numeric value, the cursor moves to the beginning of the numeric value for the variable (in this example C). If a numeric value for D is already entered, the cursor will not return to the value for C.

►

A	:B	:C
100.	:120	:135.

4 ENTER

A	:B	:C
120.	:145.	:_

(2) Correcting after obtaining the calculated result

After entering a value for each input variable and after the calculated result has been obtained, the cursor can be moved to the input variable with the ◄ key. The numeric value can then be changed and the easy simulation program can be executed again.

Example:

After executing the calculation in the previous example, change the value of A to 210 and the value of C to 180 and then run again.

160 ENTER

C	:D	:Total
145.	:160.	:525.

◄◄◄◄

A	:B	:C
100.	:120.	:145.

210 ENTER

A	:B	:C
210.	:120.	:145.

► ENTER

A	:B	:C
210.	:120.	:145

In this example, the value of B is not changed. However, if only the ENTER key is entered, the correction is assumed to be completed and the calculation is executed. The value of C cannot be corrected.

As shown in this example, press [►] and then [ENTER] to move the cursor to the next variable without changing the numeric value.

180 [ENTER]

B	:C	:D
120.	:180.	:160.

[ENTER]

C	:D	:Total
180.	:160.	:670.

After entering data, if the [ENTER] key is entered without any numeric entry when the cursor is at the beginning of the next variable, the calculation is executed.

Note: After executing the calculation, the [◄] key can be used to return the cursor to each variable. However, if the [►] key is pressed or if a number is entered at any variable, the cursor can be moved only within that variable. The cursor cannot be moved to a previous variable with the [◄] key.

However, pressing the [ENTER] key ends the entry of that variable, and the cursor can again be moved to a previous variable with the [◄] key.

Listing the Heading Variable

The output variable written after the # mark at the beginning of an easy simulation program is also called the heading variable.
Executing and listing an easy simulation program is done by specifying this heading variable.
Further, the type of program stored can be determined by looking at this heading variable.
This section describes how to list the heading variable, how to start execution of the easy simulation program from the listed heading variable, and how to list the easy simulation program.

(1) Listing the heading variable

The heading variable can be listed by input:

[ENTER]

in either the PROgram mode or RUN mode.

Example: # [ENTER]

#A	#AREA	#Y
#PRICE		

The heading variables are delimited into 6 names of 8 characters each and displayed with the # mark at the beginning of each name.

Therefore, up to 7 characters per name and up to 6 names can be displayed at one time.

If there are 7 or more heading variables (when 7 or more easy simulation programs have been written), they cannot be all displayed at one time. If so, the hidden heading variables can be listed by moving the cursor with the ► key.

(2) Executing an easy simulation program from the heading variable list

After listing the variables as described in (1) while in the RUN mode (RUN), move the cursor, with the ► key (or ◄ key), to the name of the variable to be executed. Then press the ENTER key to begin execution of the program starting with that variable name.

Example: RUN mode

# ENTER	#A #AREA #Y #PRICE
►	#A #AREA #Y #PRICE
► ►	#A #AREA #Y #PRICE
ENTER	X :Y –

(3) Listing the easy simulation program from the heading variable list

After listing the variables as described in (1) while in the RUN mode (RUN), move the cursor, with the ► key (or ◄ key), to the name of the program to be listed. Then press the ENTER key to list the program.

Example: PRO mode

# ENTER	#A #AREA #Y #PRICE
► ► ►	#A #AREA #Y #PRICE
ENTER	#Y=2*X+3 #PRICE=Cost*(1+Tax)

Clearing the easy Simulation Program

Follow the steps below to clear an easy simulation program.

(1) Clearing all easy simulation programs

To clear all the easy simulation programs within the computer, input:

NEW # ENTER

while in the PROgram mode.

(2) Clearing a single easy simulation program

To clear a single easy simulation program among the serveral stored within the computer, follow the steps below.

① Set to the PROgram (PRO) mode.

② Enter only the heading variable of the easy simulation program to be cleared and then press the ENTER key.

③ The computer then displays a confirmation message. To clear input:

1 ENTER

Doing so will clear that easy simulation program. If the program is not to be cleared, input:

2 ENTER

Example:
Clear #Y=2*X+3 among the easy simulation programs stored within the computer.
Operation: PRO mode

#Y [ENTER]

```
#Y                    CLEAR
YES-1 OR NO -2:_
```

Displays a message to confirm clearing (Clear the easy simulation program #Y? Enter 1 to clear and enter 2 not to clear)

1 [ENTER]

```
>
```

The clear operation is executed by pressing 1 [ENTER]. If a number other than 1 is entered and [ENTER] pressed, the program is not cleared but is retained. However, if a numeric value ending in 1 is entered (such as 21 or 51) and [ENTER] pressed, the program is cleared.

- If an easy simulation program beginning with the specified variable name does not exist within the computer, the confirmation message is not displayed and the prompt is immediately displayed.

Printing When Excecuting an Easy Simulation Program

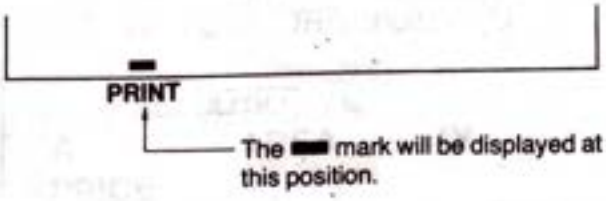
When the optional printer is connected to the PC-1260, the entries during execution of the program and the calculated result can be printed.
Connect the optional printer and simultaneously press:

[SHIFT] [P↔NP ENTER]

A [▬] mark will be displayed at the bottom of the display where PRINT is written. (Switches to the print mode.)

Now, if the easy simulation program is executed, the variable name and entries as well as calculated result will be printed as shown below.

Example: SHIFT P↔NP ENTER



#Y ENTER	X 5.
5 ENTER	Y 13.
#AREA ENTER	RADIUS 12.
12 ENTER	AREA 452.3893421

- There may be cases when the variable name and numeric value cannot be printed in one line such as when USING is used to specify many digits for the numeric value. In cases such as these, they will be printed on 2 lines.

Printing the Heading Variable Names

When the optional printer is connected and in the print mode, enter:

ENTER

to print the entire list of heading variable names stored within the computer. Like the display, up to 7 columns of each variable name are printed.

Example: PRO or RUN mode

ENTER

```
#Y      #AREA    #PRICE
#Total
```

After printing is completed, the display is cleared and the prompt reappears.

Printing an Easy Simulation Program

An easy simulation program can be printed using the LLIST # command. To print all the easy simulation programs stored in the computer, enter:

LLIST # ENTER

Further, if executed in the form of

LLIST # Heading variable (variable name specified after the # mark at the beginning of the easy simulation program) ENTER

the easy simulation program beginning with the specified variable name is printed.

Example: PRO or RUN mode

```
LLIST # [ENTER]  ──────────►  ‡Y=2*X+3
LLIST # Total [ENTER]         ‡AREA=π*RADIUS^2
                              ‡PRICE=Cost*(1+Tax/100)
                              ‡Total=A+B+C+D
        ‡Total=A+B+C+D
```

CHAPTER 10
HELP FUNCTION

There may be times when you cannot recall a program command or have forgotten its use when writing a program. At times like these, you usually look for the command in the instruction manual but you might find it time consuming or perhaps the instruction manual is not within reach. The computer is capable of listing its functions and typical notations for them (reference function). Further, the character code table is built-in and can be listed. Also, if an error occurred, the kind of error can be displayed. This section describes how to list various information.

Reference Function

There are two ways to use the reference function.

① First, when you have forgotten a command or its spelling, you can list the commands of the computer and look for the desired one. All the commands can be listed or by entering the first one or two characters, all commands beginning with the entered character(s) can be listed.

② When you have forgotten a command notation, a sample notation of the command can be listed. A sample notation can be listed by first searching for the command using method ① above or by entering the command name.

(1) Listing BASIC commands and functions

To list the commands of the computer alphabetically from the beginning, input:

[SHIFT] [HELP 7]

The commands starting with A will be listed. To list the other commands, in sequence, press the [↓] key.
(To exit the HELP feature, press the [CL] key.)

Example:

[CL] [SHIFT] [HELP]

```
ABS       ACS       AND
AREAD     ASC       ASN
```

[↓]

```
AREAD     ASC       ASN
ATN
```

[↓]

```
ATN
BEEP
```

Further, a command can be listed by specifying (entering) its first one or two characters. For example, by entering C and then pressing [SHIFT] and [HELP], commands beginning with "C" will be listed.

Note: During a reference display, only the [↓], [↑], and [CL] keys are operable. Therefore, when entering a character, clear the reference display with [CL] key.

[CL] L [SHIFT] [HELP]

```
LEFT$  LEN     LET
LIST   LIST #  LLIST
```

[↓]

```
LIST     LIST#   LLIST
LLIST #  LN      LOG
```

[CL] RE [SHIFT] [HELP]

```
READ   REM      'RESTORE
RETURN
```

Note: If there is no command beginning with the entered character, no command will be listed even though [SHIFT] and [HELP] are pressed. (The [SHIFT] and [HELP] function do not operate.)

(2) Listing a sample notation of a command

After listing the command using method (1) above, move the cursor to the beginning of the command using the [►] key and press the [ENTER] key. A sample notation for the command will be listed.

Example: List the sample notation for the INPUT command.

[CL] I [SHIFT] [HELP]

```
IF      INKEY$   INPUT
INPUT #  INT
```

[►] [►] [►]

```
IF      INKEY$   INPUT
INPUT #  INT
```

[ENTER]

```
INPUT A, B$,C
INPUT "A=";A,"B=";B
```

[↓]

```
INPUT "A=";A,"B=";B
INPUT "KOTAE=";A
```

Note: When there are three sample notations as in the example above, after listing with the [ENTER] key, the other sample notations can be listed with the [↓] key.

Further, after entering the command, the sample notation for the entered command can be listed by pressing [SHIFT] and [HELP] keys.

Example: List the sample notation for the GOTO command.

[CL] GOTO [SHIFT] [HELP]

```
GOTO 200
GOTO "B"
```

[↓]

```
GOTO "B"
GOTO L*M
```

[↓]

```
GOTO L*M
GOTO D$
```

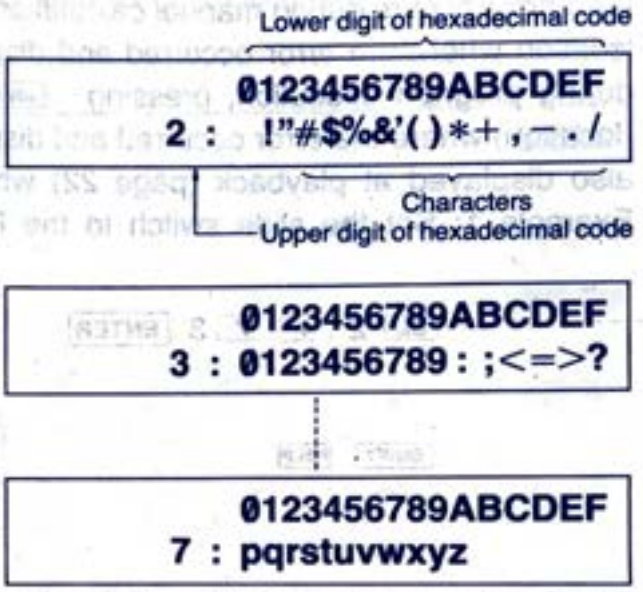
Character Code Table

The computer is equipped with a built-in character code table. The table can be listed by entering:

ASCII [SHIFT] [HELP]

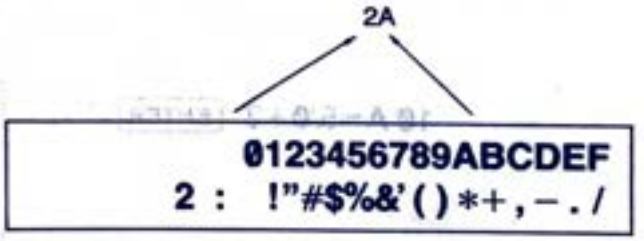
and then using the [↓] or [↑] key. The characters (letters, numerals, symbols, etc.) listed at this time can be specified with the CHR$ command.
Example:

[CL] ASCII [SHIFT] [HELP]



As shown in the example, the lower digit of the hexadecimal code is displayed on the upper line of the display unit and the higher digit of the hexadecimal code on the lower line. The characters are displayed following the : (colon). The character code is determined as follows.

Example: The code for * is 2A



Note: The code here is in hexadecimal notation. (2A is 42 in decimal notation.)

- When specifying a character using CHR$, specify as a decimal code or use the & command and first convert the hexadecimal code into a decimal code and then specify.

Example: When specifying *:

CHR$42 or CHR$&2A

Decimal code Hexadecimal code (converted to decimal code with &)

● See the character code chart on page 190.

Error Message Function

If an error occurs during manual calculation, pressing [SHIFT] and then [HELP] lists the location where the error occurred and displays the kind of error. If an error occurs during program execution, pressing [SHIFT] and then [HELP] lists the program line (location) where the error occurred and displays the kind of error. The kind of error is also displayed at playback (page 22) when an error occurs.

Example 1: Set the slide switch to the RUN mode.

[CL] 2 [*] [/] 3 [ENTER]

```
ERROR 1
```

[SHIFT] [HELP]

```
2*/3
SYNTAX ERROR
```

The flashing cursor at the top of the display indicates the location where the error occurred. The message on the bottom line indicates the kind of error.

Example 2: Set the slide switch to the PROgram mode.

NEW [ENTER]

```
>
```

10 A=5/0+3 [ENTER]

```
10 : A=5/0+3
```

20 PRINT A [ENTER]

```
20 : PRINT A
```

Change the slide switch to the RUN mode.

RUN [ENTER]

```
ERROR 2 IN 10
```

The display above indicates that error 2 occurred in line 10. To find out where in line 2 and the kind of error, press [SHIFT] and [HELP].

[SHIFT] [HELP]

```
10 : A=5/0+3
CALCULATION ERROR
```

The display indicates that the 0 in line 10 is a calculation error. Set the slide switch to the PROgram mode again. Next, press the [↑] or [↓] key. As an example, enter [8] and press [ENTER]. Set the slide switch to the RUN mode and enter RUN and press [ENTER].

```
3.625
```

This is the calculated result of 5/8+3.

CHAPTER 11
TROUBLESHOOTING

This chapter provides you with some hints on what to do when your SHARP PC-1260/1261 does not do what you expect it to do. It is divided into two parts—the first part deals with general machine operation and the second with BASIC programming. For each problem there are a series of suggestions provided. You should try each of these, one at a time, until you have fixed the problem.

Machine Operation

If:	Then You Should:
You turn on the machine but there is nothing on the display	1. Check to see that the slide switch is set to RUN, PRO, or RSV. 2. Push the [ON BRK] key to see if AUTO POWER OFF has been activated. 3. Replace the batteries. 4. Adjust the contrast control.
There is a display, but no response to keystrokes	1. Press [CL] key to clear. 2. Press [CA] ([SHIFT] [CL]) to clear. 3. Turn OFF and ON again. 4. Hold down any key and push RESET. 5. Push RESET without any key.
You have typed in a calculation or answer and get no response	1. Push [ENTER].
You are running a BASIC program and it displays something, and stops	1. Push [ENTER].
You enter a calculation and it is displayed in BASIC statement format (colon after the first number)	1. Switch from the PROgram into the RUN mode for calculations.
You get no response from any keys.	1. Hold down any key and push RESET. 2. If you get no response from any key even when the above operation is performed, push the RESET without pushing any key. This will clear the program, data and all reserved contents.

BASIC Debugging

When entering a new BASIC program, it is usual for it not to work the first time. Even if you are simply keying in a program that you know is correct, such as those provided in this manual, it is usual to make at least one typing error. If it is a new program of any length, it will probably contain at least one logic error as well. Following are some general hints on how to find and correct your errors.

You run your program and get an error message:

1. Go back to the PROgram mode and use the [↑] or the [↓] keys to recall the line with the error. The cursor will be positioned at the place in the line where the PC-1260/1261 got confused.
2. If you can't find an obvious error in the way in which the line is written, the problem may lie with the values which are being used. For example, CHR$(A) will produce an error if A has a value of 1 because CHR$(1) is an illegal character. Check the values of the variables in either the RUN or the PROgram mode by typing in the name of the variable followed by [ENTER].

You RUN the program and don't get an error message, but it doesn't do what you expect.

3. Check through the program line by line using LIST and the [↓] and [↑] keys to see if you have entered the program correctly. It is surprising how many errors can be fixed by just taking another look at the program.
4. Think about each line as you go through the program as if you were the computer. Take sample values and try to apply the operation in each line to see if you get the result that you expected.
5. Insert one or more extra PRINT statements in your program to display key values and key locations. Use these to isolate the parts of the program that are working correctly and the location of the error. This approach is also useful for determining which parts of a program have been executed. You can also use STOP to temporarily halt execution at critical points so that several variables can be examined.
6. Use TRON and TROFF, either as commands or directly within the program to trace the flow of the program through individual lines. Stop to examine the contents of critical variables at crucial points. This is a very slow way to find a problem, but sometimes it is also the only way.

CHAPTER 12
MAINTENANCE OF THE PC-1260/1261

To insure trouble-free operation of your SHARP PC-1260/1261 we recommend the following:

* Always handle the pocket computer carefully as the liquid crystal display is made of glass.
* Keep the computer in an area free from extreme temperature changes, moisture, or dust. During warm weather, vehicles left in direct sunlight are subject to high temperature build up. Prolonged exposure to high temperature may cause damage to your computer.
* Use only a soft, dry cloth to clean the computer. Do not use solvents, water, or wet cloths.
* To avoid battery leakage, remove the batteries when the computer will not be in use for an extended period of time.
* If service is required, the computer should only be returned to an authorized SHARP Service Center.
* If the computer is subjected to strong static electricity or external noise it may "hang up" (all keys become inoperative). If this occurs, press the ALL RESET button while holding down any key. (See Troubleshooting).
* Keep this manual for further reference.

(Note: For maintenance of the CE-125 please see Chapter 7.)

APPENDIX A
ERROR MESSAGES

There are nine different error codes built into the PC-1260/1261. The following table will explain these codes.

Error Number **Meaning**

1 Syntax error.

- This means that the PC-1260/1261 can't understand what you have entered. Check for things such as semicolons on the ends of PRINT statements, misspelled words, and incorrect usages.

```
3*/2
```

2 Calculation error

Here you have probably done one of three things:

1. Tried to use too large a number.
   Calculation results are greater than 9.999999999E 99.
2. Tried to divide by zero.

```
5/0
```

3. An illogical calculation has been attempted.

```
LN −30 or ASN 1.5
```

3 Illegal Function (DIMension error/Argument error)

- Array variable already exists.
  Array specified without first dimensioning it.
  Array subscript exceeds size of array specified in DIM statement.

```
DIM B(256)
```

- Illegal function argument. This means that you have tried to make the computer do something that it just can't handle.
  The interval that is greater than 65535.

```
WAIT 66000
```

4 Too Large A Line Number

Here you have probably done one of two things:

1. Tried to use a non-existent line number by the GOTO, GOSUB, RUN, LIST or THEN etc.
2. Tried to use too large a line number. The maximum line number is 65279.

5 Next Without A For ...

Subroutine nesting exceeds 10 levels.

FOR loop nesting exceeds 5 levels.
RETURN verb without a GOSUB, NEXT verb without a FOR, or READ verb without a DATA.

Buffer space exceeded.

6 Memory Overflow.
Generally this error happens when you've tried to DIMension an array that is too big for memory. This can also happen when a program becomes too large.

- The reserve content exceeds 48 bytes.
- The easy simulation program is too large and exceeds the capacity limit.

7 PRINT USING error.
This means that you have put an illegal format specifier into a USING statement.

8 I/O device error.
This error can happen only when you have the optional printer and/or cassette recorder connected to the PC-1260/1261. It means that there is a problem with communication between the I/O device and the PC-1260/1261.

9 Other errors.
This code will be displayed whenever the computer has a problem that isn't covered by one of the other eight error codes. One of the most common causes for this error is trying to access data in a variable is one fashion (e.g. A$) while the data was originally stored in the variable in another fashion (e.g.A).

Regarding Input Errors

When executing a program, an error may occur due to input errors of the program. In this case note the following.

Example: When KPRINT is entered instead of LPRINT

```
                10:K PRINT A$
   [►]          10 K PRINT A$
L  [ENTER]      10:L PRINT A$
                   └─Space
```

When corrected in this manner, the computer does not recognize it as a command. In this sample, erase KPRINT and re-enter LPRINT.

The command can be checked whether entered correctly by using the cursor key.

(Correct input)

10:RADIAN
10 RADIAN
10 RADIAN_

(Wrong input)

10:RADAN
10 RADAN
10 RADAN
10 RADAN

APPENDIX B
CHARACTER CODE CHART

The following chart shows the conversion values for use with CHR$ and ASC. The column shows the first hex character or the first four binary bits, the row shows the second hex character or the second binary bits. The upper left corner of each box contains the decimal number for the character. The lower right shows tne character. If no character is shown then it is an illegal character on the PC-1260/1261.

For examples, the character "A" is a decimal 65 or a hex 41 or binary 01000001. The character '√' is a decimal 252 or a hex FC or a biriary 11111100.

Note: When using either the CE-125 or CE-126P optional printer, be aware that the 39 (&27), 91 (&5B), and 93 (&5D) character codes for the PC-1260/1261 (displayed characters) and printer (printed characters) are different characters.

First 4 bits

The PC-1260/1261 does not recognize codes in the shaded area. If you enter a code number in the shaded area, an error will result.

Second 4 Bits

Hex / Binary	0 0000	1 0001	2 0010	3 0011	4 0100	5 0101	6 0110	7 0111	8 1000	E 1110	F 1111
0 0000	0 NUL	16	32 SPACE	48 0	64 @	80 P	96 SPACE	112 p	128	224	240
1 0001	1	17	33 !	49 1	65 A	81 Q	97 a	113 q	129	225	241
2 0010	2	18	34 "	50 2	66 B	82 R	98 b	114 r	130	226	242
3 0011	3	19	35 #	51 3	67 C	83 S	99 c	115 s	131	227	243
4 0100	4	20	36 $	52 4	68 D	84 T	100 d	116 t	132	228	244
5 0101	5	21	37 %	53 5	69 E	85 U	101 e	117 u	133	229	245 ♦
6 0110	6	22	38 &	54 6	70 F	86 V	102 f	118 v	134	230	246 ♥
7 0111	7	23	39 '	55 7	71 G	87 W	103 g	119 w	135	231	247 ♠
8 1000	8	24	40 (	56 8	72 H	88 X	104 h	120 x	136	232	248 ♣
9 1001	9	25	41)	57 9	73 I	89 Y	105 i	121 y	137	233	249
A 1010	10	26	42 *	58 :	74 J	90 Z	106 j	122 z	138	234	250
B 1011	11	27	43 +	59 ;	75 K	91 [	107 k	123	139	235	251 π
C 1100	12	28	44 ,	60 <	76 L	92 ¥	108 l	124	140	236	252 √
D 1101	13	29	45 –	61 =	77 M	93]	109 m	125	141	237	253
E 1110	14	30	46 .	62 >	78 N	94 ^	110 n	126	142	238	254
F 1111	15	31	47 /	63 ?	79 O	95 –	111 o	127	143	239	255

APPENDIX C
FORMATTING OUTPUT

It is sometimes important or useful to control the format as well as the content of output. The PC-1260/1261 controls display formats with the USING verb. This verb allows you to specify:

* The number of digits
* The location of the decimal point
* Scientific notation format
* The number of string characters

These different formats are specified with an "output mask." This mask may be a string constant or a string variable:

```
10: USING "####"
20: M$="&&&&&&"
30: USING M$
```

When the USING verb is used with no mask, all special formatting is cancelled.

```
40: USING
```

A USING verb may also be used within a PRINT statement:

```
50: PRINT USING M$; N
```

Wherever a USING verb is used, it will control the format of all output until a new USING verb is encountered.

Numeric Masks

A numeric USING mask may only be used to display numeric values, i.e., numeric constants or numeric variables. If a string constant or variable is displayed while a numeric USING mask is in effect, the mask will be ignored. A value which is to be displayed must always fit within the space provided by the mask. The mask must reserve space for the sign character, even when the number will always be positive. Thus a mask which shows four display positions may only be used to display numbers with three digits.

Specifying Number of Digits

The desired number of digits is specified using the '#' character. Each '#' in the mask reserves space for one digit. The display or print always contains as many characters as are designated.in the mask. The number appears to the far right of this field; the remaining positions to the left are filled with spaces. Postive numbers therefore

always have at least one space at the left of the field. Since the PC-1260/1261 maintains a maximum of 10 significant digits, no more than 11 '#' characters should be used in a numeric mask. When the total number of columns of the integer part specified exceed 11, this integer part is regarded as 11 digits in the PC-1260/1261.

Note: In all examples in this appendix the beginning and end of the displayed field will be marked with a 'I' character to show the size of the field.

Statement	Display
10: USING "####"	(Set the PC-1260/1261 to the RUN mode, type RUN, and press [ENTER].)
20: PRINT 25	| 25|
30: PRINT −350	|−350|
40: PRINT 1000	ERROR 7 IN 40

Notice that the last statement produced an error because 5 positions (4 digits and a sign space) were required, but only 4 were provided in the mask.

Specifying a Decimal Point

A decimal point character, '.', may be included in a numeric mask to indicate the desired location of the decimal point. If the mask provides more significant decimal digits than are required for the value to be displayed, the remaining positions to the right will be filled with zeros. If there are more significant decimal digits in the value than in the mask, the extra digits will be truncated (not rounded):

Statement	Display
10: USING "####.##"	
20: PRINT 25	| 25.00|
30: PRINT −350.5	|−350.50|
40: PRINT 2.547	| 2.54|

Specifying Scientific Notation

A " " character may be included in the mask to indicate that the number is to be displayed in scientific notation. The '#' and '.' characters are used in the mask to specify the format of the "characteristic" portion of the number, i.e., the part which is displayed to the left of the E. Two '#' characters should always be used to the left of the decimal point to provide for the sign character and one integer digit. The decimal point may be included, but is not required. Up to 9 '#' characters may appear to the right of the decimal point. Following the characteristic portion, the exponentiation character, E, will be displayed followed by one position for the sign and two positions for the exponent. Thus, the smallest scientific notation field would be provided by a mask of "##^" which would print numbers of the form '2E 99'. The largest scientific notation field would be "##.#########^" which would print numbers such as '−1.234567890 E−12':

Statement	Display
10: USING "###.##^"	
20: PRINT 2	| 2.00E 00|
30: PRINT −365.278	|−3.65E 02|

Specifying Alphanumeric Masks

String constants and variables are displayed using the '&' character. Each '&' indicates one character in the field to be displayed. The string will be positioned at the left end of this field. If the string is shorter than the field, the remaining spaces to the right will be filled with spaces. If the string is longer than the field, the string will be truncated to the length of the field:

Statement	Display
10: USING "&&&&&&"	
20: PRINT "ABC"	|ABC |
30: PRINT "ABCDEFGHI"	|ABCDEF|

Mixed Masks

In most applications a USING mask will contain either all numeric or all string formatting characters. Both may be included in one USING mask, however, for certain purposes. In such cases, each switch from numeric to string formatting characters or vice versa marks the boundary for a different value. Thus, a mask of "#####&&&&" is a specification for displaying two separate values—a numeric value which is allocated 5 positions and a string value which is allocated 4 positions:

Statement	Display
10: PRINT USING "###.##&&";25;"CR"	**|25.00CR|**
20: PRINT −5.789;"DB"	**|−5.78DB|**

Remember: Once specified, a USING format is used for all output which follows until cancelled or changed by another USING verb.

APPENDIX D
EXPRESSION EVALUATION AND OPERATOR PRIORITY

When the SHARP PC-1260/1261 is given a complex expression, it evaluates the parts of the expression in a sequence which is determined by the priority of the individual parts of the expression. If you enter the expression:

100/5+45

as either a calculation or as a part of a program, the PC-1260/1261 does not know whether you mean:

$$\frac{100}{5+45} = 2 \quad \text{or} \quad \frac{100}{5} + 45 = 65$$

Since the PC-1260/1261 must have some way to decide between these options, it uses its rules of operator priority. Because division has a higher "priority" than addition (see below), it will choose to do the division first and then the addition, i.e., it will choose the second option and return a value of 65 for the expression.

Operator Priority

Operators on BASIC of the SHARP PC-1260/1261 are evaluated with the following priorities from highest to lowest:

Level	Operations
1.	Parentheses
2.	Variables and Pseudovariables
3.	Functions
4.	Exponentiation (^)
5.	Unary minus, negative sign (−)
6.	Multiplication and division (*, /)
7.	Addition and subtraction (+, −)
8.	Relational operators (<, <=, =, <>, >=, >)
9.	Logical operators (AND,OR,NOT)

When there are two or more operators at the same priority level the expression will be evaluated from left to right. (The exponentiation will be evaluated from right to left). Note that with A+B−C, for example, the answer is the same whether the addition or the subtraction is done first.

When an expression contains multiple nested parentheses, the innermost set is evaluated first and evaluation then proceeds outward.

For level 3 and 4, the last entry has a higher priority.
For example: $-2\,\hat{}\,4 \rightarrow -(2^4)$
$3\,\hat{}\,-2 \rightarrow 3^{-2}$

Sample Evaluation

Starting with the expression:

((3+5−2)*6+2)/10^ LOG 100

The PC-1260/1261 would first evaluate the innermost set of parentheses. Since '+' and '−' are at the same level it would move from left to right and would do the addition first:

((8−2)*6+2)/10^ LOG 100

Then it would do subtraction:

((6)*6+2)/10^LOG 100

or:

(6*6+2)/10^LOG 100

In the next set of parentheses it would do the multiplication first:

(36+2)/10^LOG 100

And then the addition:

(38)/10^LOG 100

or:

38/10^LOG 100

Now that the parentheses are cleared, the LOG function has the highest priority so it is done next:

38/10^2

The exponentiation is done next:

38/100

And last of all the division is performed:

0.38

This is the value of the expression.

APPENDIX E
KEY FUNCTIONS

[ON/BRK] (ON)

Use to turn the PC-1260/1261 power on when the auto power off function is in effect.

(BREAK)

- Depressing this key during program execution functions as a BREAK([ON/BRK]) key and causes the program to interrupt execution.
- When pushed during manual execution, input/output command such as BEEP, CLOAD, etc., execution of the command is interrupted.

[SHIFT]

- The yellow key marked "SHIFT" must be used to designate a second function. (The material appearing in brown above each key).
  Ex. [SHIFT] [?/U] → ? is entered.

[CL]

- Use to clear the contents of the entry and the display. (Error release)

[SHIFT] [CA]

- Not only clears the display contents, but resets the computer to its initial state.
  —Initial state—
  - Resets the WAIT timer.
  - Resets the display format. (USING format)
  - Resets the TRON state (TROFF).
  - Resets the PRINT=LPRINT.
  - Resets error.

[0] ~ [9]

- Numeric keys.

[.]

- Decimal point.
- Use to enter an abbreviation of a command/verb/function.
- Use to designate the decimal portion in USING format designation.

[E]

- Use to designate an exponent in scientific notation. (This key is a letter E key: upper case)

[/]

- Division key.

[*]

- Multiplication key.
- Use to designate an array variable in the INPUT#, the PRINT#, etc.

[+]

- Addition key.

[−]
- Subtraction key.

[SHIFT] [^]
- Use for power calculation instructions.
- Use to specify the exponent display system for numerical data in USING statement instructions.

[SHIFT] [<]
[SHIFT] [>]
- Use when entering logical operators in IF sentence.

[DEF]
- When any one of eighteen keys (A, S, D, F, G, H, I, K, L, Z, X, C, V, B, N, M, ', SPaCe) is pushed after the depression of the [DEF] key, it starts to execute the program from the program line that has the same label as the key code depressed.

[A] ~ [Z]
- Alphabet keys. You are probably familiar with these keys from the standard typewriter keyboard.
  If pressed as is, an upper case character is displayed. If an alphabet key is pressed after pressing [SHIFT] and [SML], a lower case character is displayed.

[SPC]
- Use to provide space when entering programs or characters.

[=]
- In assignment statements, use to assign the contents (number or character) on the right for the variable specified on the left.
- Use when entering logical operators in IF sentence.

[SHIFT] [!]
- Use to designate these symbols.

[SHIFT] [*]
- " : • Use to designate and cancel characters.
  • Use to specify labels.
- # : • Use with USING statement, to provide the instruction to define the display format of numerical data.
- $: • Use when assigning character variables.
- &: • Use with USING statement, to provide the instruction to define the display format of character string.
  • Use to designate hexadecimal number.
- @: • Used for reserve contents when the reserve key is used as a program key.
  Example: GOTO 100 @
- !%: • Used as a character string within " ".

[SHIFT] [?]
- Use to enter CLOAD?

[SHIFT] [:]
- Use to divide two or more statements in one line.

[,]
- Use to provide pause between two equations, and between variables or comments.

[SHIFT] [;]
- Use to provide multi-display (two or more values/contents/ displayed at a time).
- Use to provide pause between the instruction and the variable.

[SHIFT] [(]
[SHIFT] [)]

- Use to enter parentheses.

[►]

- Shifts the cursor to the right (press once to advance one position, hold down for automatic advance)
- Executes playback instructions.
- Clears an error condition in manual operation.
- After executing an easy simulation program, it is used to list the hidden calculated result.

[◄]

- Shifts the cursor to the left (press once to advance one position, hold down for automatic advance)
- Otherwise the same as the [►] key.

[SHIFT] [INS]

- Inserts one space (⌴ appears) of 1-step capacity between the address (N) indicated by the cursor and the preceding address (N-1).

[SHIFT] [DEL]

- Deletes the contents of the address indicated by the cursor.

[SHIFT] [π]

- Use to designate pi (π).

[SHIFT] [√]

- Use to designate square root.

[ENTER]

- Enters a program line into the computer.
- Use when writing in programs
- Requests manual calculation or direct execution of a command statement by the computer.
- Used to restart a program temporarily stopped by an INPUT or PRINT command and to execute a BASIC or an easy simulation program.

[SHIFT] [P↔NP]

- Use to set print and non-print mode when an optional printer is connected with the PC-1260/1261.

[SHIFT] [HELP]

- Used to call the HELP function.
- Lists sample notation of each command.
- Lists the Character code.
- Resets the error and lists the kind of error.

[SHIFT] [SML]

- Specifies and releases the lower case mode. (Turns the SMALL symbol display on and off.)
- The SMALL symbol is displayed when [SHIFT] and [SML] are pressed. Now, if [A] [B] and [C] are pressed, a b c are displayed. If [SHIFT] and [SML] are pressed again, the SMALL symbol disappears and upper case characters are entered.

The [↑] and [↓] keys have the following functions, depending on designated modes, as well as the state of the computer.

Mode	State	[↓]	[↑]
RUN	Program being executed		
	Program is temporarily interrupted	To execute the next line	To display program line being executed or already executed, hold this key down.
	INPUT statement being executed		
	PRINT statement just now executed		
	Under break		
	Error condition during executing program		To display error-producing line, hold this key down.
	TRON condition	To execute debugging operation	To display program line being executed or already executed, hold this key down.
	Other condition	To display an answer just previously calculated. (Last answer function)	Same as left
PRO	(When the mode is changed from RUN to PRO and program line is not being displayed)		
	Program is temporarily interrupted	To display the line interrupted	Same as left.
	Error condition	To display the line with error	Same as left
	Other condition	To display the first line	To display the last line
	(When the program line or easy simulation program is being display)		
		To display the next program line or easy simulation program	To display the preceding program line or easy simulation program

- On the display, the [ENTER] key is the same as a space.
- If no key is entered in the key input request mode for approximately 11 minutes, the power is automatically turned off (auto-power off function).

APPENDIX F
SPECIFICATIONS

Model:	PC-1260/1261 Pocket Computer
Processor:	8 bit CMOS CPU
Programming Language:	BASIC
Memory Capacity:	System ROM: 40 K Bytes RAM: System: About 600 Bytes User: Fixed Memory Area (A ~ Z, A$ ~ Z$): 208 Bytes Program/Data Area: PC-1260 3198 Bytes PC-1261 9342 Bytes Reserve Area: 48 Bytes
Stack:	Sub-routine: 10 stacks Function: 16 stacks FOR–NEXT: 5 stacks Data: 8 stacks
Operators:	Addition, subtraction, multiplication, division, trigonometric and inverse trigonometric functions, logarithmic and exponential functions, angle conversion, square and square root, sign, absolute, integer, relational operators, logical operators.
Numeric Precision:	10 digits (mantissa) + 2 digits (exponent).
Editing Features:	Cursor left and right, line up and down, character insert, character delete.
Memory Protection:	CMOS Battery backup.
Display:	Dual line 24-digit liquid crystal display with 5×7 dot characters.
Keys:	52 keys: Alphabetic, numeric, special symbols, and functions. Numeric pad. User defined keys.
Power Supply:	6.0V DC: Lithium cells. Type: CR-2032×2
Power Consumption:	6.0V DC: 0.03W Approx. 300 hours when 555555555555555555555555. is on continuous display at an operating temperature of 20°C (68°F); this time may vary slightly with the operation method, etc.

- One hour operation per day allows the battery to be used for approx. 4 months. This is true for one hour operation consisting of 10 minutes of calculations or program executions and 50 minutes of displays.

Operating Temperature: 0°C ~ 40°C (32°F ~ 104°F)

Dimensions: 135(W)×70(D)×9.5(H) mm.
5-5/16" (W)×2-3/4" (D)×3/8" (H)

Weight: Approximately 115g (0.25 lbs.) (with cells)

Accessories: Hard cover, two lithium cells (built-in), two keyboard templates and instruction manual.

Options: Printer/Microcassette Recorder (CE-125)
Printer/Cassette Interface (CE-126P) etc.

Program Examples

Probably you have acquired knowledge on a number of program commands as you have progressed up to this page. It is necessary, however, to generate actual programs by yourself in addition to those given in the instruction manual, so that you can generate programs freely using BASIC language. Like driving a car or playing tennis that can be improved by actual practice, you can improve your programming only by generating as many programs as possible regardless of your skill. It is also important for you to refer to programs generated by others. For your reference, the following pages contain a variety of programs using BASIC commands.

(Sharp Corporation and/or its subsidiaries assume no responsibilities or obligations to any losses or damages that could arise through the use of the software programs employed in this instruction manual.)

CONTENTS

* The number of bytes required by the program is printed at the end of each program listing.

Preface with Remarks for Easy Simulation Program

The operating procedures for the easy simulation program examples described in this collection assume that the formula for all six easy simulation program calculations are already stored in your computer. The following describes how to store and execute the formula for each easy simulation program.

(If the maximum memory space required for each formula is assumed to be 80 bytes, 480 bytes of space is required for the six formulae. This means that three blocks of memory space will be sufficient for these formulae since (480−128)/128=2.75 < 3.)

[Storing the Formula]

(1) Place your computer in the PRO mode.
(2) Type in NEW [ENTER] .
(3) Type in EQU #3 [ENTER] .

(4)
```
‼GrsScor=OutScor+InScor:
 NetScor=GrsScor-Handi
‼Subttl=UP*QTY:Total=Tot
 al+Subttl
‼R=1+Int/100:NumPay=(
 LOG Repay- LOG (Repay-D
 ebt*(R-1)))/ LOG R
‼Off10%=Price*0.9:Off15%
 =Price*0.85:Off20%=Pric
 e*0.8
‼VarRate=VarCost/Sales:B
 rkEven=FixCost/(1-VarRa
 te)
‼Total=Art+English+Math+
 Science+SocStd:Average=
 Total/5
```

.....Golf competition scoring

.....Subtotal and Grand total

.....Monthly repayment calculation

.....Discount price list

.....Break-even analysis

.....School record processing

[Calculating a formula]

Place your computer in the RUN mode, and press # [ENTER] to put the menu on the display. Position the cursor to the formula name you wish to calculate, then strike [ENTER].

(e.g.) Calculation of the subtotal

[ENTER]

```
#GrsScor #Subttl    #R
#Off10%  #VarRate   #Total
```

[►] [►]

```
#GrsScor #Subttl    #R
#Off10%  #VarRate   #Total
```

[ENTER]

```
UP          :QTY          :Subttl
_
```

Easy Simulation Program: GOLF COMPETITION SCORING

■ Liberate Yourself from Cumbersome Scoring.

The formula in this example lets you compute the net scores of individual players from the out score, in score and the handicap.

■ EXAMPLE

Name	Out-Score	In-Score	Handicap	Net-Score
J. Brown	50	48	5	
E. Adams	55	52	10	
F. Marchy	43	50	2	

■ CALCULATION CONTENTS

Gross score=Out Score+In Score
Net score=Gross Sccre−Handicap

■ STORING THE FORMULA

Gross Score=Grs Scor
Out Score=Out Scor
In Score=In Scor
Net Score=Net Scor
Handicap=Handi
#GrsScor=OutScor+InScor : NetScor=GrsScor−Handi

```
#GrsScor=OutScor+InScor:
 NetScor=GrsScor-Handi
```

■ KEY OPERATION SEQUENCE

Step No.	Key Operation	Display	Remarks
1	SHIFT CA (CL)	>	Initialization
2	# ENTER	#GrsScor #Subttl #R #Off10% #VarRate #Total	Formula menu
3	ENTER	OutScor :InScor :Handi _	Prompts for out score entry
4	50 ENTER	OutScor :InScor :Handi 50. :_	Prompts for in score entry
5	48 ENTER	OutScor :InScor :Handi 50. :48. :_	Prompts for handicap entry
6	5 ENTER	InScor :Handi :GrsScor 48. :5. :98.	Displays the gross score
7	►	Handi :GrsScor :NetScor 5. :98. :93.	Display shift
8	ENTER	OutScor :InScor :Handi _	Prompts for out score entry
9	55 ENTER	OutScor :InScor :Handi 55. :_	Repeat the same cycle
10	52 ENTER	OutScor :InScor :Handi 55. :52. :_	
11	10 ENTER	InScor :Handi :GrsScor 52. :10. :107.	
12	►	Handi :GrsScor :NetScor 10. :107. :97.	
13	ENTER	OutScor :InScor :Handi _	

■ Cumulative Operation Too is in Your Reach.

The formula given in this example lets you determine subtotals from unit prices and quantities. You can also cumulate a number of subtotals one by one into a grand total.

■ EXAMPLE

Model	Unit Price	Quantity	Subtotal
XC-1240	17800	20	
XD-1250	29800	30	
XE-1500	64800	40	
Grand total			

Determine the subtotal for each model, and finally the grand total.

■ CALCULATION CONTENTS

Subtotal=unit price×quantity
Grand total=cumulation of subtotals

■ STORING THE FORMULA

Unit price=UP
Quantity=QTY
Subtotal=Subttl
Grand total=Total
#Subttl=UP*QTY: Total=Total+Subttl

```
#Subttl=UP*QTY:Total=Tot
 al+Subttl
```

■ KEY OPERATION SEQUENCE

Step No.	Key Operation	Display	Remarks
1	SHIFT CA/CL	>	Initialzation
2	# ENTER	#GrsScor #Subttl #R #Off10% #VarRate #Total	Formula menu
3	► ► ENTER	UP :QTY :Subttl _	Prompts for unit price
4	17800 ENTER	UP :QTY :Subttl 17800. :_	Prompts for quantity
5	20 ENTER	UP :QTY :Subttl 17800. :20. :356000.	Subtotal display
6	►	QTY :Subttl :Total 20. :356000. :356000.	Display shift
7	ENTER	UP :QTY :Subttl _	Prompts for unit price
8	29800 ENTER	UP :QTY :Subttl 29800. :_	Repetition of same cycle
9	30 ENTER	UP :QTY :Subttl 29800. :30. :894000.	
10	►	QTY :Subttl :Total 30. :894000. :1250000.	
11	ENTER	UP :QTY :Subttl _	
12	64800 ENTER	UP :QTY :Subttl 64800. :_	
13	40 ENTER	UP :QTY :Subttl 64800. :40. :2592000.	
14	►	Subttl :Total 2592000. :3842000.	Grand total display

Easy Simulation Program: MONTHLY REPAYMENT CALCULATION

■ Be Smart in Making a Repayment Plan.

You should make a reasonable repayment plan for your loan. The formula given in this example lets you compute the required number of monthly payments from your ability to repay the loan.

■ EXAMPLE

The following example calculates the required number of monthly repayments, assuming that you owe $3,000,000 at an annual interest of 12% when you can repay $100,000 monthly.

Principal=$3,000,000
Monthly repayment=$100,000
Annual interest=12% (monthly interest=12/12%=1%)

■ CALCULATION CONTENTS

Principal ($A)
Monthly repayment ($a)
Annual interest (r%)

R $(R=1+\frac{r}{100})$

Number of payments $=\frac{\log a-\log(a-A(R-1))}{\log R}$

■ STORING THE FORMULA

Principal=Debt
Monthly repayment=Repay
Annual interest=Int
Number of monthly payments=NumPay
#R=1+Int/100 : NumPay=(LOG Repay−LOG (Repay−Debt*(R−1)))/LOG R

```
#R=1+Int/100:NumPay=(
 LOG Repay- LOG (Repay-D
 ebt*(R-1)))/ LOG R
```

■ KEY OPERATION SEQUENCE

Step No.	Key Operation	Display	Remarks
1	[SHIFT] [CA/CL]	>	Initialization
2	# [ENTER]	#GrsScor #Subttl #R #Off10% #VarRate #Total	Formula menu
3	[►] [►] [►]	#GrsScor #Subttl #R #Off10% #VarRate #Total	Cursor positioning
4	[ENTER]	Int :Repay :Debt _	Prompts for annual interest
5	12/12 [ENTER]	Int :Repay :Debt 1. :_	Prompts for monthly repayment
6	100000 [ENTER]	Int :Repay :Debt 1. :100000. :_	Prompts for principal
7	3000000 [ENTER]	Repay :Debt :R 100000. :3000000. :1.01	Displays R
8	[►]	R :NumPay 1.01 :35.84553611	Displays shift

Easy Simulation Program: DISCOUNT PRICE LIST

■ Why Not Make a Discount Price List.

It's not so easy to wait on good bargainers. So why don't you make a list of cut-off prices and do good business?

■ EXAMPLE

The following example produces a list of cut-off prices for 10%, 15% and 20% discount for machines priced at $169,800 and $198,000.

■ STORING THE FORMULA

```
Price=Price
10% discount=Off10%
15% discount=Off15%
20% discount=Off20%
#Off10%=Price*0.9 : Off15%=Price*0.85 : Off20%=Price*0.8
```

```
#Off10%=Price*0.9:Off15%
 =Price*0.85:Off20%=Pric
 e*0.8
```

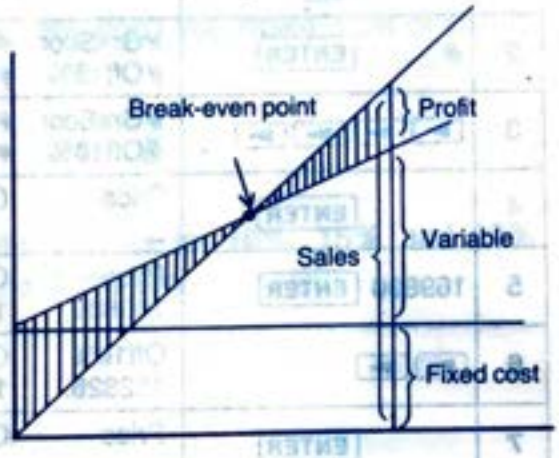
■ KEY OPERATION SEQUENCE

Step No.	Key Operation	Display			Remarks
1	SHIFT CA CL	>			Initialization
2	# ENTER	#GrsScor #Off10%	#Subttl #VarRate	#R #Total	Formula menu
3	► ► ► ►	#GrsScor #Off10%	#Subttl #VarRate	#R #Total	Cursor positioning
4	ENTER	Price _	:Off10%	:Off15%	Prompts for price
5	169800 ENTER	Price 169800.	:Off10% :152820.	:Off15% :144330.	Displays 10% and 15% off prices
6	► ►	Off10% 152820.	:Off15% :144330.	:Off20% :135840.	Display shift
7	ENTER	Price _	:Off10%	:Off15%	Prompts for price
8	198000 ENTER	Price 198000.	:Off10% :178200.	:Off15% :168300.	Repetition of same cycle
9	► ►	Off10% 178200.	:Off15% :168300.	:Off20% :158400.	
10	ENTER	Price _	:Off10%	:Off15%	

Easy Simulation Program: BREAK-EVEN POINT ANALYSIS

■ Let's Check Break-even Point.

The break-even point is determine for the relationship shown on the right.
The following example lets you determine the break-even point from your sales, variable cost and fixed cost. Also let's take a look at how the break-even point changes as your change your variable cost.



■ EXAMPLE

Sales: $10,000,000
Variable cost: $6,000,000
Fixed cost: $3,000,000

Try to change the variable cost to $5,000,000 and see how the break-even point changes.

■ CALCULATION CONTENTS

Variable cost rate=Variable cost/Sales
Break-even point=Fixed cost/(1−Variable cost rate)

■ STORING THE FORMULA

Variable cost rate=VarRate
Variable cost=VarCost
Sales=Sales
Break-even point=BrkEven
Fixed cost=FixCost
#VarRate=VarCost/Sales : BrkEven=FixCost/(1−VarRate)

```
#VarRate=VarCost/Sales:B
 rkEven=FixCost/(1-VarRa
 te)
```

■ KEY OPERATION SEQUENCE

Step No.	Key Operation	Display	Remarks
1	SHIFT CA CL	>	Initialization
2	# ENTER	#GrsScor #Subttl #R #Off10% #VarRate #Total	Formula menu
3	► ► ► ► ►	#GrsScor #Subttl #R #Off10% #VarRate #Total	Cursor positioning
4	ENTER	VarCost :Sales :FixCost _	Prompts for variable cost
5	600 ENTER	VarCost :Sales :FixCost 600. :_	Prompts for entry of sales
6	1000 ENTER	VarCost :Sales :FixCost 600. :1000. :_	Prompts for fixed cost
7	300 ENTER	Sales :FixCost :VarRate 1000. :300. :0.6	Displays the variable rate
8	►	FixCost :VarRate :BrkEven 300. :0.6 :750.	Displays the break-even point
9	◄	Sales :FixCost :VarRate 1000. :300. :0.6	Playback
10	◄	VarCost :Sales :FixCost 600. :1000. :300.	Playback
11	◄ ◄	VarCost :Sales :FixCost 600. :1000. :300.	Prompts for the updated variable cost
12	5 ENTER	VarCost :Sales :FixCost 500. :1000. :300.	
13	ENTER	Sales :FixCost :VarRate 1000. :300. :0.5	
14	►	FixCost :VarRate :BrkEven 300. :0.5 :600.	

Easy Simulation Program: SCHOOL RECORD PROCESSING

■ Making School Records is Now a Simple Thing.

Students' record processing after each examination is rather a tedious task. The formula given in this example allows you to total and average the marks for five subjects.

■ EXAMPLE

Name	Art	English	Mathe-matics	Science	Social studies	Total	Average
P. Smith	84	73	89	69	53		
G. Luis	60	82	93	84	65		
E. White	55	76	56	64	73		

■ CALCULATION CONTENTS

Total mark=Art+English+Mathematics+Science+Social studies
Average=Total mark/5

■ STORING THE FORMULA

Total mark=Total
Social studies=SocStd
#Total=Art+English+Math+Science+SocStd : Average=Total/5

```
#Total=Art+English+Math+
 Science+SocStd:Average=
 Total/5
```

■ KEY OPERATION SEQUENCE

Step No.	Key Operation	Display	Remarks
1	SHIFT CA CL	>	Initialization
2	# ENTER	#GrsScor #Subttl #R #Off10% #VarRate #Total	Formula menu
3	► ► ► ► ► ►	#GrsScor #Subttl #R #Off10% #VarRate ▮Total	Cursor positioning
4	ENTER	Art :English :Math _	Prompts for the Art mark
5	84 ENTER	Art :English :Math 84. :_	Prompts for the English mark
6	73 ENTER	Art :English :Math 84. :73. :_	Prompts for the Mathematics mark
7	89 ENTER	English :Math :Science 73. :89. :_	Prompts for the Science mark
8	69 ENTER	Math :Science :SocStd 89. :69. :_	Prompts for the Social Studies mark
9	53 ENTER	Science :SocStd :Total 69. :53. :368.	Displays total mark
10	►	SocStd :Total :Average 53. :368. :73.6	Display shift
11	ENTER	Art :English :Math _	Prompts for the Art mark

Program Title: GOLF COMPETITION SCORING CALCULATION

Your Golf Competition Manager Needs Help!

Your golf competition manager often has a hard time struggling with enormous amounts of statistics calculated from scores. The program in this example is ideally suited for those managers in trouble. The program lets you subtract preprogrammed handicaps and figure out the ranking.

■ INSTRUCTIONS

1) Using the DATA statement, program the names of players and their handicaps from line 600 of the program. (You may enter the initials of each player name.)
2) Press DEF A to start the program. Type in the par.
3) The computer will display the first player's name and handicap. Type in the in and out scores.
4) Type in the scores for all other players.
5) When you have entered the scores of all players, the computer will print a score card.

■ REFERENCE

In this example, the names of 30 persons and their handicap data are input. For your own application, update the subscript for the array variable in line 20 according to the number of players. Also the names of players and their handicaps specified in the DATA statement on line 600 and beyond must be updated according to your own needs.

■ EXAMPLE

This example determines the handicapped score and ranking of each player from the following score table (PAR=72)

No.	Name	IN	OUT	No.	Name	IN	OUT
1	H. Killy	36	38	16	C. Brown	40	46
2	T. Underson	54	44	17	D. Green	42	39
3	O. Wild	50	44	18	E. White	47	48
4	T. Jefferson	58	58	19	F. Brack	51	49
5	P. Washington	65	69	20	G. Luis	63	56
6	W. Lagan	49	53	21	H. Uleman	59	68
7	W. Genkins	45	51	22	L. Murphy	49	42
8	A. Ford	44	49	23	B. Gallen	50	60
9	B. Kenedy	43	44	24	I. Baker	54	60
10	N. Adams	47	59	25	J. Lincolin	60	66
11	S. Easton	54	52	26	K. Daglus	46	46
12	K. Jefferson	71	75	27	Q. Maple	53	58
13	J. Smith	45	48	28	R. Biggs	58	57
14	A. Homes	46	65	29	N. Wood	45	39
15	H. Themes	51	50	30	M. Bond	43	39

■ KEY OPERATION SEQUENCE

Step No.	Key Input	Display	Remarks
1	DEF A	Input Par_	Prompts entry of par data.
2	72 ENTER	Name Handi In Out H.K 11 ?	Prompts for the first players in-score.
3	36 ENTER	Name Handi In Out H.K 11 36_ ?	Prompts for the first player's out-score.
4	38 ENTER	Name Handi In Out T.U 12 ?	Prompts for second player's in-score.
		Repeat the same entry cycle	
60	43 ENTER	Name Handi In Out M.B 10 43_ ?	
61	39 ENTER	>	Prints a score card on the Printer. Program end

■ PRINTED OUTPUTS

** Golf competition **
Par=72

Name	H.C.	Grss	In	Out
1				
H.K	-9	11	74	36	38
2				
W.L	-6	36	102	49	53
3				
C.B	-5	19	86	40	46
4				
E.W	-4	27	95	47	48
5				
O.W	-3	25	94	50	44
6				
J.S	-2	23	93	45	48
7				
B.G	-1	39	110	50	60
8				
M.B	0	10	82	43	39
9				
K.D	1	19	92	46	46
L.M	1	18	91	49	42
11				
D.G	2	7	81	42	39
12				
N.W	3	9	84	45	39
13				
A.F	4	17	93	44	49
B.K	4	11	87	43	44
N.A	4	30	106	47	59
16				
S.E	7	27	106	54	52
R.B	7	36	115	58	57
18				
A.H	8	31	111	46	65
19				
T.J	10	34	116	58	58
I.B	10	32	114	54	60
W.G	10	14	96	45	51
22				
G.L	11	36	119	63	56
H.T	11	18	101	51	50
24				
T.V	14	12	98	54	44
F.B	14	14	100	51	49
26				
Q.M	17	22	111	53	58
27				
J.L	18	36	126	60	66
28				
H.U	22	33	127	59	68
29				
P.W	26	36	134	65	69
30				
K.J	38	36	146	71	75

■ PROGRAM LIST

```
 10:"A": CLS : CLEAR
 20:D=30
 30:DIM A$(D),C(D),D(D),
    E(D),F(D),G(D)
 40:INPUT "Input Par ? "
    ;Z
 50:RESTORE
 60:FOR I=0 TO D-1
 70:READ A$(I),G(I)
 80:WAIT 0
 90:CLS : PRINT " Name
     Handi  In  Out"
100:CURSOR 26: PRINT A$(
    I)
110:CURSOR 34: PRINT
    STR$ G(I)
120:CURSOR 39: INPUT C(I
    )
130:IF C(I)>=100 GOTO 12
    0
140:CURSOR 44: INPUT D(I
    )
150:IF D(I)>=100 GOTO 14
    0
160:E(I)=C(I)+D(I)
170:F(I)=E(I)-G(I)-Z
180:NEXT I
190:FOR H=0 TO D-2
200:J=H+1
210:FOR K=J TO D-1
220:IF F(H)<=F(K) GOTO 2
    40
230:GOSUB 500
240:NEXT K
250:NEXT H
260:LPRINT " ** Golf com
    petition **"
270:LPRINT "          Par=
    "; STR$ Z
280:LPRINT "Name     H.C.
     Grss In Out"
290:FOR I=0 TO D-1
300:IF I<>0 IF F(I-1)=F(
    I) GOTO 320
310:LPRINT STR$ (I+1)
320:LPRINT USING "&&&":A
    $(I); USING "####":F
    (I);G(I);
330:LPRINT USING "######"
    ;E(I); USING "####";
    C(I);D(I)
340:USING
350:NEXT I
360:LPRINT "": LPRINT ""
370:END
500:P=F(H):F(H)=F(K):F(K
    )=P
510:P=G(H):G(H)=G(K):G(K
    )=P
520:P=E(H):E(H)=E(K):E(K
    )=P
530:P=C(H):C(H)=C(K):C(K
    )=P
540:P=D(H):D(H)=D(K):D(K
    )=P
550:P$=A$(H):A$(H)=A$(K)
    :A$(K)=P$
560:RETURN
600:DATA "H.K",11,"T.V",
    12,"O.W",25,"T.J",34
    ,"P.W",36,"W.L",36
610:DATA "W.G",14,"A.F",
    17,"B.K",11,"N.A",30
    ,"S.E",27,"K.J",36
620:DATA "J.S",23,"A.H",
    31,"H.T",18,"C.B",19
    ,"D.G",7,"E.W",27
630:DATA "F.B",14,"G.L",
    36,"H.U",33,"L.M",18
    ,"B.G",39,"I.B",32
640:DATA "J.L",36,"K.D",
    19,"Q.M",22,"R.B",36
    ,"N.W",9,"M.B",10
650:END
```

1046 bytes

■ MEMORY CONTENTS

A	
B	
C	
D	√
E	
F	
G	
H	√
I	√
J	√
K	√
L	
M	
N	
O	
P, P$	√
Q	
R	
S	
T	
U	
V	
W	
X	
Y	
Z	Par
A$(D)	Name (Initial)
C(D)	In score
D(D)	Out score
E(D)	Gross
F(D)	Score
G(D)	Handicap

Program Title: TYPING PRACTICE

■ OVERVIEW

Quick key operation!
How fast and accurate is your typing?
If you practice with this program, it will make programing much easier for you. Improve your skill!

■ CONTENTS (such as calculation contents)

The number of characters (3 ~ 6) is randomly chosen.
The character arrangement (A ~ Z) is done randomly.
The allotted time depends on the number of characters and the grade level.
3 is the shortest time allotment while 1 is the longest.

■ INSTRUCTIONS

After the buzzer sounds 3 to 6 characters will be displayed. You are to type in the same characters within the allotted time. If they are all correct, you get 10 points. If more than half are correct, you get 5 points.
After the allotted time is over, the next problem is displayed. The allotted time depends on the grade, which has three levels (1, 2, 3).
3 is the shortest time allotment while 1 is the longest. Point competition is done within the same grade category. There are 10 problems, making the maximum score 100 points.

Use DEF A when you wish to play in the same grade.

■ KEY OPERATION SEQUENCE

Step No.	Key Input	Display	Remarks
1	DEF Z	Typing Practice Grade (1, 2, 3)?_	Title display Grade input
2	1 ENTER	Practice 1=AZBDC	
3	A	Practice 1=AZBDC A	
4	Z	Practice 1=AZBDC AZ	
		(repeated input)	
		Your-Score=80. Your Score Is Best	After the 10 questions are answered the Score is displayed
	ENTER	>	
1	DEF A	Typing Practice High-Score=80	When you want to play in the same grade
2	B	Practice 1 =BVSW B	
	V	Practice 1 =BVSW BV	
		(repeated input)	
		Your-Score=60.	
		>	

■ PROGRAM LIST

```
 10:"Z": CLEAR : DIM B$(
    5),C$(5): RANDOM :
    WAIT 0
 13:CLS : CURSOR 4:
    PRINT "Typing Practi
    ce": CURSOR 29
 15:INPUT "Grade(1,2,3)?
    ";L: WAIT 0
 17:IF (L=1)+(L=2)+(L=3)
    <>1 THEN 13
 18:GOTO 30
 20:"A": WAIT 0:P=0: CLS
    : CURSOR 4: PRINT "T
    yping Practice":
    CURSOR 30: PAUSE "Hi
    gh-Score=";X
 30:FOR S=1 TO 10
 40:B= RND 4+2:Y$="":R=
    INT (B/2)
 50:FOR C=0 TO B-1
 60:D= RND 26:B$(C)=
    CHR$ (D+&40):Y$=Y$+
    CHR$ (D+&40): NEXT C
 70:BEEP 3:E=0
 75:CLS : PRINT "Practic
    e "; RIGHT$ (" "+
    STR$ (S),2);" ";Y$:
    CURSOR 36
 80:FOR W=1 TO B*10/L*2:
    IF E=B LET W=B*20/L*
    2: GOTO 100
 85:LET C$(E)= INKEY$ :
    IF C$(E)="" THEN 100
 87:LET A$=A$+C$(E):
    PRINT C$(E);
 90:LET E=E+1
100:NEXT W:Q=0
110:FOR W=0 TO B-1: IF B
    $(W)=C$(W) LET Q=Q+1
120:NEXT W: IF Q<=R THEN
    150
130:IF Q=B LET P=P+10:
    GOTO 150
140:P=P+5
150:NEXT S: BEEP 3: CLS
    : WAIT 100: PRINT "
    Your-Score=";P
160:IF P>X LET X=P: WAIT
    : CURSOR 25: PRINT "
    Your Score Is Best"
170:END
```

583 bytes

■ MEMORY CONTENTS

A$	√
B	√
C	Loop counter
D	√
E	√
F	
G	
H	
I	
J	
K	
L	Grade
M	
N	
O	
P	Score
Q	√
R	√
S	Loop counter
T	
U	
V	
W	Loop counter
X	High score
Y$	√
Z	
B$(5)	√
C$(5)	√

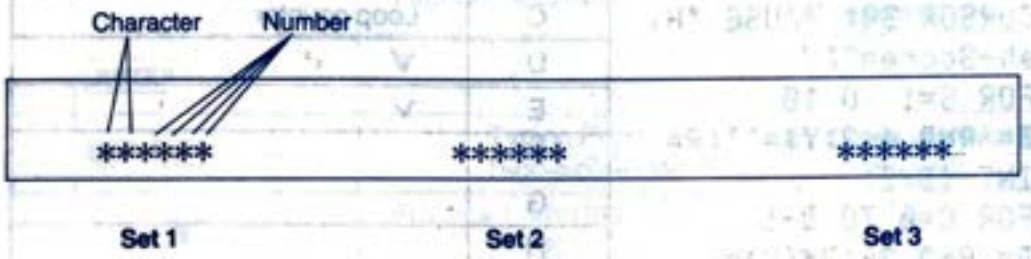
Program Title: MEMORY CHECKER

■ OVERVIEW

Three line with a total of 18 characters will be displayed on the screen for approx. 5 seconds.
Your memory will be tested by how well you input the above line after it has disappeared.

■ CONTENTS

The following type of line will be displayed for approx. 5 seconds. There are 2 characters and 4 numbers in each set.



The 3 sets shown above are to be memorized and then input as answers.
The computer will then analyze your answers and place you in one of the possible 7 categories.
Each set is split into 2 parts of former 3 and latter 3 characters, giving a total of 6 points when all the answers are correct.

Points	Evaluation Message
0	..... NOT SO HOT .....
1	..... RUNNING DOWN .....
2	----- AVERAGE -----
3	----- O K -----
4	+++++ GOOD +++++
5	***** INTELLIGENT *****
6	***** GENIUS *****

■ KEY OPERATION SEQUENCE

Step No.	Key Input	Display	Remarks
1	DEF A	Memory Check	Title
		Memory Check AB1234 PK4398 VC7216	Display of problem line (5 sec.)
		Answer (1)= ?	Waiting for the input of set 1.
2	AB1234 ENTER	Answer (2)= ?	Waiting for the input of set 2.
3	PK4398 ENTER	Answer (3)= ?	Waiting for the input of set 3.
4	VC7215 ENTER	AB1234 PK4398 VC7216 AB1234 PK4398 VC7215	Display of the Problem line and answer input.
		*****INTELLIGENT***** Do you play again (Y/N)_	Display of category. Player input request.
5	Y or N ENTER	>	If Y, goto step 1. If N, end

■ PROGRAM LIST

```
 10:"A": CLS : USING :
    WAIT 0: CURSOR 6:
    PRINT "Memory Check"
    : CLEAR
 20:DIM G$(6)*1,N$(10)*1
    ,V$(3)*3,X$(3)*6,Z$(
    3)*3,Y$(3)*6
 30:FOR I=1 TO 9:N$(I)=
    STR$ I: NEXT I:N$(10
    )="0"
 50:FOR I=1 TO 6
 60:J= RND 26:J=J+64
 70:G$(I)= CHR$ (J):
    NEXT I
 80:FOR I=1 TO 3
 90:Y$(I)=" "
100:FOR J=1 TO 3:K= RND
    9
110:Y$(I)=Y$(I)+N$(K):
    NEXT J
120:L= RND 9:J=(I-1)*2+1
130:A$(I)=G$(J)+G$(J+1)+
    N$(L)
140:H$=Y$(I):A$(I+3)=
    RIGHT$ (H$,3): NEXT
    I
150:CURSOR 25: WAIT 400:
    GOSUB 500: WAIT 0
160:FOR I=1 TO 3
170:CLS : PRINT " Answer
    (";STR$ I;")=";:
    INPUT X$(I):X$(I)=
    LEFT$ (X$(I),6)
180:Z$(I)= LEFT$ (X$(I),
    3)
190:V$(I)= RIGHT$ (X$(I)
    ,3): NEXT I
200:WAIT 200: CURSOR 1:
    GOSUB 500: CURSOR 25
    : GOSUB 520
210:N=0
220:FOR I=1 TO 3
230:IF A$(I)=Z$(I) LET N
    =N+1
240:IF A$(I+3)=V$(I) LET
    N=N+1
250:NEXT I
260:N=N+1
265:WAIT 150
270:CLS : ON N GOTO 300,
    310,320,330,340,350,
    360
300:BEEP 1: PRINT " ....
    . NOT SO HOT .....":
    GOTO 370
310:BEEP 1: PRINT ".....
     RUNNING DOWN ....."
    : GOTO 370
320:BEEP 2: PRINT "  ---
    -- AVERAGE -----":
    GOTO 370
330:BEEP 2: PRINT "
    ----- OK -----":
    GOTO 370
340:BEEP 3: PRINT "    +
    ++++ GOOD +++++":
    GOTO 370
350:BEEP 4: PRINT "*****
     INTELLIGENT *****":
    GOTO 370
360:BEEP 4: PRINT "   **
    *** GENIUS *****"
370:W$="": BEEP 1:
    CURSOR 24: INPUT "Do
     you play again(Y/N)
    ";W$
380:IF W$="N" THEN 600
390:IF W$="Y" THEN 10
395:WAIT 0: GOTO 270
500:BEEP 2: PRINT A$(1);
    A$(4);"  ";A$(2);A$(
    5);"  ";A$(3);A$(6)
510:RETURN
520:BEEP 1: PRINT USING
    "&&&&&&";X$(1);
    USING ;"  "; USING "
    &&&&&&";X$(2); USING
    ;"  "; USING "&&&&&&
    ";X$(3)
530:USING : RETURN
600:END
```

1051 bytes

■ MEMORY CONTENTS

A$	3 columns of characters
B$	
C$	
D$	
E$	
F$	
G	
H$	√
I	Index
J	Random number generation
K	Random number generation
L	Random number generation
M	
N	Counter
O	
P	
Q	
R	
S	
T	
U	
V	
W$	Input for REPLAY
X	
Y	
Z	
G$(6)	Characters(1 ~ 6)
N$(10)	Number table (1 ~ 10)
V$(3)	3 columns after answering (1 ~ 3)
X$(3)	Work (1 ~ 3)
Y$(3)	Work (1 ~ 3)
Z$(3)	3 columns before answering (1 ~ 3)

Program Title: NUMBER OF DAYS CALCULATION

■ OVERVIEW

How many days has it been since you were born? This program is convenient for answering such questions. By setting a certain day, this program will output the number of days that have passed since that day.

■ INSTRUCTIONS

DEF A

Base Year ENTER
Month ENTER
Day ENTER
Target Year ENTER
Month ENTER
Day ENTER

To end the program, type in DEF Z in place of the year.

■ EXAMPLE

from 1976 year 10 month 5 day
to 1982 year 6 month 4 day: 2068 days
to 1985 year 1 month 1 day: 3010 days

■ KEY OPERATION SEQUENCE

Step No.	Key Input	Display	Remarks
1	DEF A	From? Y M D	Waiting for base data
2	1976 ENTER	From 1976_Y? M D	Base date 1976 year 10 month 5 day input
3	10 ENTER	From 1976_Y 10_M? D	
4	5 ENTER	From 1976_Y 10_M 5_D To ? Y M D	
5	1982 ENTER	From 1976_Y 10_M 5_D To 1982_ Y? M D	Target date 1982 year 6 month 4 day input
6	6 ENTER	From 1976_Y 10_M 5_D To 1982_ Y 6_ M? D	
7	4 ENTER	From 1976_Y 10_M 5_D *** 2068 days ***	
8	ENTER	From 1976_Y 10_M 5_D To ? Y M D	
9	1985 ENTER	From 1976_Y 10_M 5_D To 1985_ Y? M D	Target date 1985 year 1 month 1 day input
10	1 ENTER	From 1976_Y 10_M 5_D To 1985_ Y 1_ M? D	
11	1 ENTER	From 1976_Y 10_M 5_D *** 3010 days ***	
12	ENTER	From 1976_Y 10_M 5_D To ? Y M D	
13	DEF Z	>	Program end

■ PROGRAM LIST

```
10:"A": WAIT 0: USING :
   CLS
20:PRINT " From      Y
      M    D"
30:CURSOR 6: INPUT R
40:CURSOR 13: INPUT S
50:CURSOR 18: INPUT T
60:CURSOR 25: PRINT "To
           Y    M    D"
70:CURSOR 30: INPUT F
80:CURSOR 37: INPUT V
90:CURSOR 42: INPUT W
100:H=R
110:G=S:I=T
120:GOSUB 500
130:J=I:H=F
140:G=V:I=W
150:GOSUB 500
160:X=I-J
170:CURSOR 24: PRINT "
      "
180:CURSOR 25: PRINT " *
   ** ";X;" days ***"
190:WAIT : PRINT
200:WAIT 0: GOTO 60
210:END
500:IF G-3>=0 LET G=G+1:
   GOTO 520
510:G=G+13:H=H-1
520:I= INT (365.25*H)+
   INT (30.6*G)+I
530:I=I- INT (H/100)+
   INT (H/400)-306-122:
   RETURN
540:"Z": END
```

397 bytes

■ MEMORY CONTENTS

A	
B	√
C$	√
D	√
E	
F	Year (after calculation)
G	√
H	√
I	√
J	√
K	
L	
M	
N	
O	
P	
Q	
R	√
S	Month of base date
T	Day of base date
U	
V	Month of target date
W	Day of target date
X	Number of days
Y	
Z	

Program Title: HISTOGRAM

■ OVERVIEW

This program graphs the histogram of the data input.

■ CONTENTS

Give the range of the input data A ~ B (A < B) and the interval width D of the frequency and distribution graphs. The various data X_i are valid only when $A \leqq X \leqq B$ and is ignored otherwise. When making a graph, the frequency given to one asterisk (*) is first set. After the graph has been printed the mean value, number of valid input data and standard deviation is also printed.

(Note) Both A and B must be integers and up to 4 digits.

■ INSTRUCTIONS

1. The program is initiated by pressing [R] [U] [N] [ENTER] . The lower bound A, upper bound B and interval with (scale unit size) are input.
2. The data is then input. The input data is printed, so if there was mistaken data entry, use [DEF] [B] when the display shows "Data=" and it is waiting for data entry. The deleted data can be input once again.
   To continue data entry again press [DEF] [A] when display shows "Deletion Data=" and then continue with data input.
3. When all data input is finished, input [DEF] [C] when the display shows "Data=" and set the frequency of one asterisk (*), then printout of the histogram takes place.
4. If when using [DEF] [B] the display shows "Over-deleted, Check", then the data input was mistaken and the entire data set should be checked once again and program should be restarted from the beginning.

■ EXAMPLE

The historgram are desired for the marks of a math exam.

Data range=0~100
Scale unit size=10
"*"=1

78	92	63	70	42	53	45	60	97	82
98	12	24	85	36	49	53	83	72	85
42	23	70	80	95	77	81	19	36	71
29	63	49	55	67	78	62	41	32	68

■ PRINTED OUTPUTS

```
Input Data
Data= 78.
Data= 92.
Data= 63.
Data= 70.
Data= 42.
Data= 53.
Data= 45.
Data= 60.
Data= 97.
Data= 82.
Data= 98.
Data= 12.
Data= 24.
Data= 85.
Data= 36.
Data= 49.
Data= 53.
Data= 83.
Data= 72.
Data= 85.
Data= 42.
Data= 23.
Data= 70.
Data= 80.
Data= 95.
Data= 77.
Data= 81.
Data= 19.
Data= 36.
Data= 71.
Data= 29.
Data= 63.
Data= 49.
Data= 55.
Data= 55.

Deletion Data
**  55.

Input Data
Data= 67.
Data= 78.
Data= 62.
Data= 41.
Data= 32.
Data= 32.
Data= 67.

Deletion Data
**  32.
**  67.

Input Data
Data= 68.


Histgram
*=1.

      0    5   10   15
      +----+----+----+---
    0I
     I
   10I                   2
     I**
   20I                   3
     I***
   30I                   3
     I***
   40I                   6
     I******
   50I                   3
     I***
   60I                   6
     I******
   70I                   7
     I*******
   80I                   6
     I******
   90I                   4
     I****
  100I
     I
  100+----+----+----+---


Number of data= 40.
Mean= 60.425
Std.Dev.=22.82639645
```

■ KEY OPERATION SEQUENCE

Step No.	Key Input	Display	Remarks
1	RUN [ENTER]	Range A=<Data= <B A=? B=	Input data range.
2	0 [ENTER]	Range A=<Data =<B A=0_ B=?	
3	100 [ENTER]	Scale unit size=_	Input scale unit
4	10 [ENTER]	Data=_	
5	78 [ENTER]	Data=_	
6	92 [ENTER]	Data=_	
		≀	
		Input all the data in the same manner	
38	55	Data=_	
39	55 [ENTER]	Data=_	Here, wrong input is done
40	[DEF] [B]	Deletion Data=_	Use [DEF] [B] to delete wrong data
41	55 [ENTER]	Deletion Data=_	
42	[DEF] [A]	Data=_	Resume the data input process
43	67 [ENTER]	Data=_	
		≀	
		Keep going	
48	68 [ENTER]	Data=_	
49	[DEF] [C]	*=_	Frequency input
50	1 [ENTER]	>	Print out of histogram and calculation results

■ PROGRAM LIST

```
10:CLEAR : WAIT 0
20:PRINT "Range A=<Data
   =<B": CURSOR 30:
   PRINT "A=": CURSOR 3
   9: PRINT "B="
25:CURSOR 32: INPUT A:
   CURSOR 41: INPUT B
30:CLS : INPUT "Scale u
   nit size=";D
40:C=B-A:E= INT (C/D)
60:DIM D(E),B$(0)*24
70:LPRINT "Input Data"
80:K=1: WAIT 0:F=0
90:"A" USING : IF F>0
   LET F=0: LPRINT "":
   LPRINT "Input Data"
100:X=-1E99
110:INPUT "Data=";X
120:IF (X<A)+(X>B)=1
   GOTO 100
125:LPRINT "Data= ";X
130:N= INT ((X-A)/D)
140:D(N)=D(N)+1
150:U=U+X:V=V+X*X
160:K=K+1
170:GOTO 100
200:"B" USING : IF F<>1
   LET F=1: LPRINT "":
   LPRINT "Deletion Dat
   a"
205:X=-1E99
210:INPUT "Deletion Data
   =";X
230:IF (X<A)+(X>B)=1
   GOTO 210
235:LPRINT "**  ";X
240:N= INT ((X-A)/D)
250:IF D(N)-1<0 BEEP 2:
   WAIT : PRINT "   Ove
   r-deleted Check"
260:D(N)=D(N)-1:K=K-1
265:U=U-X:V=V-X*X
270:GOTO 210
300:"C"F=2: WAIT 100
305:LPRINT "": LPRINT ""
   : LPRINT "Histgram"
310:P=1: INPUT "*= ";P:
   LPRINT "*=";P
320:LPRINT ""
330:LPRINT " "; USING "#
   ####";0;5*P;10*P;15*
   P
340:LPRINT "     +----+-
   ---+----+---"
345:H=A
350:FOR L=0 TO E
360:IF D(L)=0 GOTO 440
370:Q= INT (D(L)/P): IF
   D(L)>Q*P LET Q=Q+1
380:IF Q>18 LET Q=18
390:B$(0)="     I"
400:FOR M=1 TO Q:B$(0)=B
   $(0)+"*": NEXT M
410:LPRINT USING "#####"
   ;H;"I                ";
   D(L)
420:LPRINT B$(0)
430:GOTO 460
440:LPRINT USING "#####"
   ;H;"I"
450:LPRINT "     I"
460:H=H+D: NEXT L
470:LPRINT USING "#####"
   ;B;"+----+----+----+
   ---"
480:LPRINT "": LPRINT ""
500:W=U/(K-1): USING
505:LPRINT "Number of da
   ta= ";K-1
510:LPRINT "Mean= ";W
520:LPRINT "Std.Dev.=";√
   (-W*W+V/(K-1))
530:END
```

1034 bytes

■ MEMORY CONTENTS

A	Lower bound
B	Upper bound
C	√
D	Interval width
E	Interval number
F	Status flag
G	
H	√
I	
J	
K	Valid input data number
L	Loop counter
M	Loop counter
N	√
O	
P	Frequency of 1 "*"
Q	√
R	
S	
T	
U	Sum of input data (Σx_i)
V	Σx_i^2
W	Mean value ($\Sigma x_i/K$)
X	Input data (x_i)
Y	
Z	
D(E)	Frequency in each small interval
B$(0)	For graph printout

Program Title: BIORHYTHM (SEMI-GRAPHIC)

■ OVERVIEW

You input your name and date of birth, the biorhythm for any specified month will be printed. Watch out for those dangerous days.

■ CONTENTS

Input: name
date of birth
desired year, month
The date of birth and the year, month to be analyzed are input in the western calender.

Output: The biorhythm curve of the month to be analyzed is printed out using the characteres "B", "♥", and "◆".

From the date of birth the biorhythm cycle for the physical is 23 days, for emotions 28 days, and for intellect 33 days.

■ INSTRUCTIONS

DEF A initiates the program. Input the information in the following order: name, date of birth, year and month to be analyzed. After tne above informations are input the biorhytm is printed out using "B", "♥", and "◆".
(B:Physical ♥: Emotional ◆: Intellectual)

■ EXAMPLE

Name SHARP
Date of Birth 1952 Year 1 Month 28 Day.
Desired Year and Month 1982 Year 9 Month.

■ KEY OPERATION SEQUENCE

Step No.	Key Input	Display	Remarks
1	DEF A	Name=_	Waiting for name input
2	SHARP ENTER	Date of Birth ? Year Month Day	Waiting for Date of Birth
3	1952 ENTER	Date of Birth 1952_Year? Month Day	
4	1 ENTER	Date of Birth 1952_Year 1_Month? Day	
5	28 ENTER	Desired Year and Month ? Year Month	
6	1982 ENTER	Desired Year and Month 1982_ Year ? Month	
7	9 ENTER	Desired Year and Month 1982_ Year 9_ Month	
8		>	Biorhythm printed out and program end.

■ PRINTED OUTPUTS

```
Name:SHARP
Birth Date:
  1952Year 1Month 28Day
Desired Month:
  1982Year 9Month

     -    0    +
I---------+---------I
I  B  ♦    +  ♥      I  1
I    ♦B    +    ♥     I  2
I    ♦ B   +      ♥   I  3
I  ♦     B +       ♥  I  4
I--♦------B-------♥--I  5
I  ♦        + B     ♥ I  6
I  ♦        +   B   ♥ I  7
I  ♦        +    B  ♥ I  8
I    ♦      +     ♥B  I  9
I----♦-----+-----♥--B--I 10
I     ♦     +  ♥     B  I 11
I        ♦  +♥       B  I 12
I          ♦♥         B I 13
I          ♥♦     B     I 14
I------♥---+♦-B-------I 15
I      ♥    +B♦        I 16
I    ♥     B+    ♦     I 17
I  ♥    B  +      ♦    I 18
I  ♥ B     +       ♦   I 19
I-♥-B------+--------♦--I 20
I  ♥       +        ♦  I 21
I  ♥       +        ♦  I 22
I  B♥      +        ♦  I 23
I  B  ♥    +        ♦  I 24
I----B-♥---+-------♦---I 25
I      B   ♥+        ♦  I 26
I        B ♥    ♦      I 27
I          B♥  ♦       I 28
I          + ♦♥        I 29
I---------♦---♥------I 30
I---------+---------I
```

■ PROGRAM LIST

```
 5:"A": CLEAR : DEGREE
   : LPRINT "": WAIT 0
10:INPUT "Name=";NAME$
20:LPRINT "Name:";NAME$
30:PRINT "Date of Birth
   ": CURSOR 29: PRINT
   "Year";"    ";"Month
   ";"   ";"Day"
35:CURSOR 24: INPUT U$:
   CURSOR 34: INPUT V:
   CURSOR 42: INPUT W
40:LPRINT "Birth Date:"
   : LPRINT "  ";U$;"Ye
   ar "; STR$ V;"Month
   "; STR$ W;"Day"
50:GOSUB 500:B=X
60:CLS : PRINT "Desired
    Year and Month":
   CURSOR 32: PRINT "Ye
   ar";"     ";"Month"
65:CURSOR 27: INPUT U$:
   CURSOR 38: INPUT V:W
   =0
70:LPRINT "Desired Mont
   h:": LPRINT "  ";U$;
   "Year "; STR$ V;"Mon
   th"
75:LPRINT ""
80:GOSUB 500:A=X
100:C=A-B
110:GOSUB 700
120:GOSUB 800
130:D=C- INT (C/23)*23
140:E=C- INT (C/28)*28
150:F=C- INT (C/33)*33
160:DIM B$(2)*21
170:B$(1)="I---------+--
   -------I"
180:B$(2)="I          +
          I"
200:LPRINT "     -    0
      +"
210:LPRINT B$(1)
215:FOR K=1 TO Z
220:L=0: IF K=5* INT (K/
   5) LET L=1
230:IF L=1 LET B$(0)=B$(
   1): GOTO 250
240:B$(0)=B$(2)
250:G= SIN ((K+D)/23*360
   ):P$="B": GOSUB 900
260:G= SIN ((K+E)/28*360
   ):P$= CHR$ &F6:
   GOSUB 900
270:G= SIN ((K+F)/33*360
   ):P$= CHR$ &F7:
   GOSUB 900
290:LPRINT B$(0); USING
   "###";K
300:NEXT K
310:LPRINT B$(1)
320:END
500:T= VAL U$
560:IF V>=3 LET Q=T:R=V+
   1:S=W: GOTO 580
570:Q=T-1:R=13+V:S=W
580:X= INT (365.25*Q)+
   INT (30.6*R)+S-122
590:X=X- INT (Q/100)+
   INT (Q/400)
600:RETURN
700:IF T<>4* INT (T/4)
   LET Y=0: RETURN
710:IF T<>100* INT (T/10
   0) LET Y=1: RETURN
720:IF T<>400* INT (T/40
   0) LET Y=0: RETURN
730:Y=1: RETURN
800:IF (V=4)+(V=6)+(V=9)
   +(V=11)=1 LET Z=30:
   RETURN
810:IF V=2 LET Z=28+Y:
   RETURN
820:Z=31: RETURN
900:N=11+ INT (8* ABS G)
   * SGN G
910:B$(0)= LEFT$ (B$(0),
   N-1)+P$+ RIGHT$ (B$(
   0),21-N)
920:RETURN
930:END
```

1096 bytes

■ MEMORY CONTENTS

A	√
B	√
C	Total number of days
D	Physical
E	Emotion
F	Intellect
G	Biorhythm curve
H	
I	
J	
K	Loop counter
L	√
M	
N	√
O	
P$	√
Q	√
R	√
S	√
T	√
U	√
V	Month
W	√
X	√
Y	√
Z	Relevant number of days
NAME$	Name
B$(2)	√

Program Title: AVERAGE, VARIANCE AND STANDARD DEVIATION

■ OVERVIEW

When the data are input, the total sum, average, variance, and standard deviation will be calculated for you. Revision of input data as well as data with weights is possible.

■ CONTENTS

Total sum $\Sigma x_i \cdot f_i$

Standard deviation $\sigma=\sqrt{\sigma^2}$

Average $\bar{x}=\dfrac{\Sigma x_i \cdot f_i}{\Sigma f_i}$

Variance $\sigma^2=\dfrac{\Sigma(x_i-\bar{x})^2 f_i}{\Sigma f_i-1}$

(when there are no weights $f_i=1$)

■ INSTRUCTIONS

1. At DEF A , select whether or not there are any weights, then input the data.
2. DEF B is used to find any revision positions in the data. DEF C is used to revise the data.
3. The total sum, average, variance, and standard deviation will be calculated with DEF D .

■ EXAMPLE

x_i	14.1	14.2	14.3	14.4	14.5
f_i	8	19	23	15	10

(data with weights)

■ KEY OPERATION SEQUENCE

[Data Input]

Step No.	Key Input	Display	Remarks
1	DEF A	No. of Data=_	Waiting for No. of Data
2	5 ENTER	No. of Data=5_ Weights=1/No Weights=2_	Select either one.
3	1 ENTER	X(1)=?	Waiting for Data x_1
4	14.1 ENTER	X(1)=14.1_ F(1)=?	Waiting for Data f_1
5	8 ENTER	X(2)=?	⋮
	⋮	≀	⋮
	⋮	Input data in the same manner.	⋮
12	14.5 ENTER	X(5)=14.5_ F(5)=?	⋮
13	10 ENTER	>	Process End

[Data confirmation and correction]

Step No.	Key Input	Display	Remarks
1	DEF B	X(1)=14.1 F(1)=8.	Showing data x_1 and f_1
2	ENTER	X(2)=14.1 F(2)=19.	
3	DEF C	X(2)=14.1 X(2)=?	Use DEF C to correct wrong data
4	14.2 ENTER	F(2)=19. F(2)=?	Correct valve input
5	ENTER	X(2)=14.2 F(2)=19.	
		≀ Operate in the same manner.	

■ KEY OPERATION SEQUENCE

[Output of calculation results]

Step No.	Key Input	Display	Remarks
1	DEF D	Total Sum=1072.5 Average=14.3	
2	ENTER	Variance= 1.432432432E-02	
3	ENTER	Std.Dev.= 1.196842693E-01	
4	ENTER	>	

■ PROGRAM LIST

```
10:"A": CLEAR : WAIT 0:
   CLS
20:INPUT "No. of Data="
   ;P
25:CURSOR 24: INPUT "We
   ights=1/No Weights=2
   ";A
30:CLS
40:IF A=2 DIM X(P-1):
   GOTO 70
50:IF A=1 DIM X(P-1),F(
   P-1): GOTO 70
60:GOTO 30
70:FOR I=0 TO P-1
80:B$="X("+ STR$ (I+1)+
   ")="
85:PRINT B$;: INPUT X(I
   )
100:IF A=2 GOTO 150
120:B$="F("+ STR$ (I+1)+
   ")="
130:CURSOR 24: PRINT B$;
   : INPUT F(I)
135:GOTO 150
150:CLS : NEXT I: END
200:"B": WAIT 0:I=0
205:IF A=2 WAIT
210:B$="X("+ STR$ (I+1)+
   ")=":J=1: PRINT B$;X
   (I): CURSOR 24
230:IF A=1 LET C$="F("+
   STR$ (I+1)+")=":
   WAIT : PRINT C$;F(I)
   :J=2: GOTO 240
240:I=I+1
250:IF I=P END
255:CLS : WAIT 0: GOTO 2
   05
260:"C": WAIT 0: PRINT B
   $;X(I): CURSOR 24:
   PRINT B$;: INPUT X(I
   )
265:IF A=2 GOTO 250
270:CLS : PRINT C$;F(I):
   CURSOR 24: PRINT C$;
   : INPUT F(I)
280:GOTO 250
290:IF J=1 GOTO 230
300:"D":N=0:T=0:S=0: FOR
   I=0 TO P-1:X=X(I)
305:F=1: IF A=1 LET F=F(
   I)
310:N=N+F:T=T+F*X:S=S+F*
   X*X: NEXT I
400:WAIT 0:X=T/N:Q=(S-N*
   X*X)/(N-1):S=√Q:
   BEEP 3
405:CLS : PRINT "Total S
   um=";T
406:CURSOR 24: WAIT :
   PRINT "Average=";X
407:CLS : WAIT 0: BEEP 1
   : PRINT "Variance=":
   CURSOR 32: WAIT :
   PRINT Q
408:CLS : WAIT 0: BEEP 1
   : PRINT "Std.Dev.=":
   CURSOR 32: WAIT :
   PRINT S
410:END
```

722 bytes

■ MEMORY CONTENTS

A	√
B$	√
C$	√
D	
E	
F	√
G	
H	
I	√
J	Flag
K	
L	
M	
N	√
O	
P	No. of Data
Q	Variance
R	
S	Standard deviation
T	Total sum
U	
V	
W	
X	Average
Y	
Z	
X(P−1)	Data x_i
F(P−1)	Data f_i

Program Title: CORRELATION COEFFICIENT AND LINEAR REGRESSION

■ OVERVIEW (Statistics)

Data is for analysis and testing hypotheses. This program finds the covariance and correlation coefficient for related data sets $(x_1 , y_1) \ldots (x_n , y_n)$, as well as the linear regression.

The following data input is put into the equation Y=AX+B, for output to the printer.

■ INSTRUCTIONS

1.: Data input (x_i , y_i).
2.: Correction of mistaken data.
3.: The covariance, correlation coefficient, regression coefficients and means are found and output to the printer.
4.: Y is estimated from the X value and output to the printer.

First input the position of mistaken data, next the corrected data. Next select [Y] or [N] if the print of data list is needed or not respectively.

Remark: Example of data correction.
For example when you correct Y(5), operate [Y] [SHIFT] [(] [1] 5 [SHIFT] [)] [2] [ENTER] in the input of data correction position.

■ CALCULATION

$S_{xx} = \Sigma xi^2 - nx^2$

$S_{xy} = \Sigma xiyi - n\bar{x}\bar{y}$

$S_{yy} = \Sigma yi^2 - n\bar{y}^2$

$C = S_{xy}/(n - 1)$ ..... covariance

$r = S_{xy}/\sqrt{S_{xx}S_{yy}}$ ..... correlation coefficient

$a = S_{xy}/S_{xx}$
$b = \bar{y} - a\bar{x}$
} regression coefficient $(y=ax+b)$

■ EXAMPLE

x_i	6.9	7.6	7.6	9.0	8.1	6.5	6.4	6.9
y_i	12	10	9	5	6	15	14	12

Estimated value
X=7
X=8
X=7.5

■ PRINTED OUTPUTS

```
No. of Data= 8.
X(1)= 6.9
Y(1)= 12.
X(2)= 7.6
Y(2)= 10.
X(3)= 7.6
Y(3)= 9.
X(4)= 9.
Y(4)= 5.
X(5)= 8.1
Y(5)= 6.
X(6)= 6.5
Y(6)= 15.
X(7)= 6.4
Y(7)= 14.
X(8)= 6.9
Y(8)= 12.
Variance=
          -3.060714286
Correlation Coefficient=
        -9.693968513E-01
*Regression Coefficient*
A= -3.942042318
B= 39.4475621
   *** Mean Value ***
X= 7.375
Y= 10.375



   ***Estimation***
X= 7.
Y= 11.85326587
X= 8.
Y= 7.911223556
X= 7.5
Y= 9.882244715
```

■ KEY OPERATION SEQUENCE

[Data Input]

Step No.	Key Input	Display	Remarks
1	DEF A	No. of Data ?_	Input No. of Data.
2	8 ENTER	X(1)= ? Y(1)=	Waiting for x_1 input
3	6.9 ENTER	X(1)= 6.9_ Y(1)= ?	Waiting for y_1 input
4	12 ENTER	X(2)= ? Y(2)=	
		Input in the same manner.	
16	14 ENTER	X(8)= ? Y(8)=	
17	6.9 ENTER	X(8)= 6.9_ Y(8)= ?	
18	12 ENTER	>	Process end

[Data correction]

Step No.	Key Input	Display	Remarks
1	DEF B	Which to correct?	
2	X(2) ENTER	X(2)=5.8 New Data?	Showing current data. Enter new data
3	7.6 ENTER	Which to correct?	
4	ENTER	All data print? (Y/N)_	Want a data list?
5	Y ENTER	>	Data List
	or		
	N ENTER	>	No Data List

■ KEY OPERATION SEQUENCE

[Printout of statistical results]

Step No.	Key Input	Display	Remarks
1	DEF C	** Under Process **	
		>	All the data printed out

[Estimation]

Step No.	Key Input	Display	Remarks
1	DEF D	Estimation X?_	Waiting for X
2	7 ENTER	Estimation X?_	
3	8 ENTER	Estimation X?_	
4	7.5 ENTER	Estimation X?_	
5	ENTER	>	End of process

■ PROGRAM LIST

```
10:"A": CLEAR : CURSOR
20:INPUT "No. of Data ?
   ";N
30:DIM X(N-1),Y(N-1),B$
   (0)
40:LPRINT "No. of Data=
   ";N
50:FOR A=1 TO N
60:B$="X("+ STR$ A+")="
70:C$="Y("+ STR$ A+")="
80:WAIT 0
90:PRINT B$;"
          ";C$
100:CURSOR 8: INPUT X(A-
    1)
110:CURSOR 32: INPUT Y(A
    -1)
120:LPRINT B$;" ";X(A-1)
130:LPRINT C$;" ";Y(A-1)
140:CLS
150:NEXT A
160:END
170:"C": LPRINT "":I=0:J
    =0:K=0:L=0:M=0
175:PAUSE "  ** Under Pr
    ocess **"
180:FOR A=1 TO N
190:Z=A-1
200:I=I+X(Z)
210:J=J+Y(Z)
220:K=K+X(Z)*X(Z)
230:L=L+X(Z)*Y(Z)
240:M=M+Y(Z)*Y(Z)
250:NEXT A
260:I=I/N:J=J/N
270:K=K-N*I*I
280:L=L-N*I*J
290:M=M-N*J*J
300:H=√(K*M)
310:H=L/H
320:LPRINT "Variance=":
    LPRINT L/(N-1)
330:LPRINT "Correlation
    Coefficient=":
    LPRINT H
340:LPRINT "*Regression
    Coefficient*"
350:S=L/K:T=J-S*I
360:LPRINT "A= ";S
370:LPRINT "B= ";T
380:LPRINT "   *** Mean
    Value ***"
390:LPRINT "X= ";I
400:LPRINT "Y= ";J
410:END
420:"B": CURSOR :B$(0)="
    "
430:INPUT "Which to corr
    ect ?";B$(0)
440:IF B$(0)="" GOTO 530
450:V= LEN B$:D$= LEFT$
    (B$(0),1):U= VAL (
    MID$ (B$(0),3,V-1))
460:IF (U(=0)+(U)N)=1
    GOTO 420
470:WAIT 0
480:PRINT " ";B$(0);"=";
    X(U+1-2)
490:IF D$="X" CURSOR 24:
    PRINT "New Data":
    CURSOR 33: INPUT X(U
    -1): LPRINT "X(";
    STR$ U;")=";X(U-1):
    GOTO 420
500:PRINT " ";B$(0);"=";
    Y(U+1-2)
510:IF D$="Y" CURSOR 24:
    PRINT "New Data":
    CURSOR 33: INPUT Y(U
    -1): LPRINT "Y(";
    STR$ U;")=";Y(U-1):
    GOTO 420
520:GOTO 420
530:O$=""
540:INPUT "    All data
    Print ?          (
    Y/N) ";O$
550:IF (O$="Y")+(O$="N")
    ()1 GOTO 540
560:IF O$="N" END
570:LPRINT "No. of Data=
    ";N
580:FOR I=1 TO N
590:LPRINT "X(";STR$ I;
    ")= ";X(I-1)
600:LPRINT "Y(";STR$ I;
    ")= ";Y(I-1)
610:NEXT I
620:END
630:"D": LPRINT "":
    LPRINT "   ***Estima
    tion***"
640:INPUT "Estimation X
    ?";X: GOTO 660
650:END
660:LPRINT "X= ";X
670:Y=S*X+T
680:LPRINT "Y= ";Y
690:GOTO 640
700:END
```

1301 bytes

■ MEMORY CONTENTS

A	√
B$	√
C$	√
D$	√
E	
F	
G	
H	Correlation coefficient
I	√, Mean value on X
J	Mean value on Y
K	S_{xx}
L	S_{xy}
M	S_{yy}
N	Number of data
O$	√
P	
Q	
R	
S	Regression coefficient A
T	Regression coefficient B
U	√
V	√
W	
X	Estimated value X
Y	Estimated value Y
Z	√
X(N−1)	X data
Y(N−1)	Y data
B$(0)	√

Program Title: SIMULTANEOUS EQUATIONS

■ OVERVIEW

Who is Mr. X?

There are a number of Mr. X'es hiding with weird smiles in simultaneous equations. But don't be scared. This program allows you to solve any complex simultaneous euation using the "sweep-out" technique, just by entering the coefficients and constants for each equation.

■ INSTRUCTIONS

1) Press DEF A to enter coefficients:
   First type in the number of degrees, then sequentially type in the data.
2) Press DEF B to check and correct entered data:
   Check the data just entered and correct it if needed. When you have finished correcting and checking the data the computer will ask if you wish to print the data. If so type in "1"; if not, type in "2".
3) Press DEF C to compute the result:
   The computer calculates and prints the solution.

■ REFERENCE

An equation may be expressed in a matrix as follows:

$$Ax=b,\ A=\{a_{ij}\},\ x=\{x_i\},\ b=\{b_i\}\ (i,j=1\sim n)$$
$$P=a_{mm}(m=2\sim n)$$
$$q=a_{im}/P(i=1\sim m-1)$$
$$a_{ij}=a_{ij}-qq\cdot a_{mi}(i=1\sim m)$$
$$b_i=b_i-q\cdot b_m$$

If this operation is executed repeatedly, the coefficient for $i<j$ is: $a_{ij}=0$.
As a result, we obtain $x_i=b_1/a_{11}$ from $a_{11}x_1=b_1$, and $x_2=(b_2-a_{21}x_1)/a_{22}$ from $a_{21}x_1+a_{22}x_2=b_2$, and so on and so forth.
If P becomes zero during the operation, the program execution will be terminated, with the message "Error" is shown on the display.

■ Example

Three simultaneous equations

$$\begin{cases} 2x_1 + 3x_2 + 8x_3 = 0 \\ 3x_1 - 8x_2 + 6x_3 = -24 \\ -2x_1 + 2x_2 - 2x_3 = 1.5 \end{cases}$$

Determine the solution (x_1, x_2, x_3) of the above equation.

■ PRINTED OUTPUTS

```
Degree=3.
A(1,1)=2.
A(1,2)=3.
A(1,3)=8.
B(1)=0.
A(2,1)=3.
A(2,2)=-8.
A(2,3)=6.
B(2)=-24.
A(3,1)=-2.
A(3,2)=2.
A(3,3)=-2.
B(3)=1.5


X1= 4.5
X2= 3.
X3= -2.25
```

■ KEY OPERATION SEQUENCE

[Entering coefficients]

Step No.	Key Input	Display	Remarks
1	DEF A	Degree=_	Prompts for number of degrees
2	3 ENTER	Degree=3_ 3 Simultaneous Equation	
3	2 ENTER	A(1,1)=? A(1,2)=	Prompts for coefficient x of equation 1
4	3 ENTER	A(1, 1)=2. A(1,2)=?	Prompts for coefficient x of equation 2
5	8 ENTER	A(1,2)=3. A(1,3)=?	⋮
6	0 ENTER	A(1,3)=8. B(1)=?	Prompts for the constant of equation 1
	≀	≀ Repeat the same cycle of data entry	≀
14		A(3,3)=−2. B(3)=?	
15	1.5 ENTER	>	END

[Data correction and check]

Step No.	Key Input	Display	Remarks
1	DEF B	A(1,1)=2. Correct A(1,1)=?	Check the Coefficient for x of equation 1.
2	ENTER	A(1,2)=3. Correct A(1,2)=?	If no correction is required, just press ENTER.
3	ENTER	A(1,2)=8. Correct A(1,3)=?	
4	ENTER	B(1)=0 Correct B(1)=?	
5	ENTER	A(2,1)=−3. Correct A(2,1)=?	
6	3 ENTER	A(2,2)=3. Correct A(2,1)=?	If a correction is required, type in the correct data.
	≀	≀	
12	ENTER	B(3)=1.5 Correct B(3)=?	
13	ENTER	Wish a data printout? Yes=1 No=2?	Asks if you wish a data printout.
14	1 ENTER	>	END

■ KEY OPERATION SEQUENCE

[Result output]

Step No.	Key Input	Display	Remarks
1	DEF C	Wish a data printout? Yes=1 No=2?	Asks if you wish a result printout.
2	1 ENTER	>	Prints the solution. END

■ PROGRAM LIST

```
10:"A": WAIT 0: CLEAR :
   CLS : INPUT "Degree=
   ";N:N=N-1: BEEP 1:
   WAIT 99: CURSOR 24
20:PRINT STR$ (N+1);"
   Simultaneous Equatio
   n": WAIT 0
30:DIM C(N,N),B(N),D$(0
   )
40:FOR I=0 TO N
50:FOR J=0 TO N
60:PRINT "A(";STR$ (I+
   1);",";STR$ (J+1);"
   )=";
70:INPUT C(I,J)
75:CLS : PRINT "A(";
   STR$ (I+1);",";STR$
   (J+1);")=";C(I,J):
   CURSOR 24
80:NEXT J
90:PRINT "B(";STR$ (I+
   1);")=";
100:INPUT B(I)
105:CLS : PRINT "B(";
   STR$ (I+1);")=";B(I)
   : CURSOR 24
110:NEXT I
120:END
200:"B": CLS : WAIT 0
210:FOR I=0 TO N
220:FOR J=0 TO N
230:D$(0)="A("+ STR$ (I+
   1)+","+ STR$ (J+1)+"
   )= "
240:PRINT D$(0);C(I,J)
250:CURSOR 24: PRINT "Co
   rrect ";D$(0);:
   INPUT C(I,J): CLS :
   GOTO 240
260:CLS : NEXT J
270:D$(0)="B("+ STR$ (I+
   1)+")= "
280:PRINT D$(0);B(I)
290:CURSOR 24: PRINT "Co
   rrect ";D$(0);:
   INPUT B(I): CLS :
   GOTO 280
300:CLS : NEXT I
310:CLS : PRINT "Wish a
   data print out ?"
320:CURSOR 24: PRINT "
        Yes=1  No=2  ";
330:INPUT I
340:IF I<>1 END
350:LPRINT "Degree=";N+1
360:FOR I=0 TO N
370:FOR J=0 TO N
380:LPRINT "A(";STR$ (I
   +1);",";STR$ (J+1);
   ")=";C(I,J)
390:NEXT J
400:LPRINT "B(";STR$ (I
   +1);")=";B(I)
410:NEXT I
420:END
500:"C": WAIT 0: CLS :I=
   0:J=0: FOR I=0 TO N
510:J=I
520:IF C(I,J)<>0 THEN 59
   0
530:J=J+1: IF J>N THEN 8
   20
540:IF C(I,J)=0 THEN 530
550:FOR K=0 TO N
560:C=C(I,J):C(I,K)=C(J,
   K):C(J,K)=C
570:NEXT K
580:C=B(I):B(I)=B(J):B(J
   )=C
590:C=C(I,I)
600:IF C=0 THEN 820
610:FOR J=0 TO N
620:C(I,J)=C(I,J)/C
630:NEXT J
640:B(I)=B(I)/C
650:FOR K=0 TO N
660:IF K=I THEN 720
670:C=C(K,I)
680:FOR J=0 TO N
690:C(K,J)=C(K,J)-C*C(I,
   J)
695:NEXT J
700:B(K)=B(K)-C*B(I)
720:NEXT K
730:NEXT I
740:CLS : WAIT 0: BEEP 1
   : PRINT "Wish a data
    print out ?":
   CURSOR 24
750:PRINT "       Yes=1
    No=2  ";: INPUT J
760:IF (J=1)+(J=2)<>1
   THEN 740
770:FOR I=0 TO N
780:IF J=1 LPRINT "X";
   STR$ (I+1);"= ";B(I)
   : GOTO 800
790:CLS : PRINT "X";
   STR$ (I+1);"= ";B(I)
795:WAIT : CURSOR 24:
   PRINT "X"; STR$ (I+2
   );"= ";B(I+1)
797:IF N=(I+1) END
800:WAIT 0: NEXT I
810:END
820:BEEP 3: PRINT "Error
   ": CURSOR 24: WAIT :
   PRINT "    Not Opera
   ble !"
830:END
```

1342 bytes

■ MEMORY CONTENTS

A	
B	
C	√
D	
E	
F	
G	
H	
I	√
J	√
K	√
L	
M	
N	Degree
O	
P	
R	
S	
T	
U	
V	
W	
X	
Y	
Z	
C(N,N)	Enterd data
B(N)	Enterd constants, solutio
D$(0)	√

Some notes on the Microcassette Tape Supplied with the CE-125(s)

The microcassette tape is supplied for use with the PC-1250/1251 computers. When the PC-1260/1261 computer is used with the CE-125(s) Printer/Microcassette Recorder, the microcassette tape supplied with the CE-125(s) can be used except for some part of the tape. The tape contains 20 programs for the PC-1250/1251 Computers. The program description and operating procedures for 9 programs out of the 20 programs are provided in the Instruction Manual of the PC-1250/1251, while those for the remaining 11 programs are provided in the Instruction Manual of the CE-125(s) Printer/Microcassette Recorder. The following describes some notes on the usage of these programs:

1) While the contents of the Instruction Manual for the PC-1250/1251 Computers are duplicated in this collection, some descriptions are simplified here. For more details, about the program descriptions provided in the Instruction Manual of the PC-1250/1251, see pages after this page respectively.

2) Every program line is made so that it can be shown on a single row of display. So they are shown on the first (top) row of the display on the PC-1260/1261 Computers.

3) For the titles of programs contained in the supplied microcassette tape, tape counter, and some other points, see page 21 of the CE-125(s) Instruction Manual.

4) Some portions of the Instruction manual for the CE-125(s) state that you should operate [SHIFT] [(] (or [SHIFT] [)] to enter a parenthesis "("(or ")"). But this instruction is not correct on the PC-1260/1261. Use [SHIFT] [(] (or [SHIFT] [)]) instead.

(Sharp Corporation and/or its subsidiaries assume no responsibilities or obligations to any losses or damages that could arise through the use of the software programs employed in this instruction manual.)

Note: The PC-1260/1261 can read from a tape which contains programs developed for the PC-1250/1251. However, the reverse is not possible. That is, the PC-1250/1251 cannot read from a tape which contains programs developed for the PC-1260/1261.

CONTENTS

Program Title: NEWTON'S METHOD FOR FINDING ROOTS OF EQUATIONS

■ OVERVIEW (mathematical)

Finding the roots of equations is usually troublesome, but by using Newton's Method the approximate roots of equations can be found.
When 1 root is found, depending on the interval width, by using Newton's Method the starting point automatically changes.

■ CONTENTS

$$X_{n+1}=X_n-\frac{f(X_n)}{f'(X_n)}$$

If the absolute value of the distance between X_n and X_{n+1} is less than 10^{-8}, X_n is considered a root and is displayed. Here the first derivative is defined in the following way:

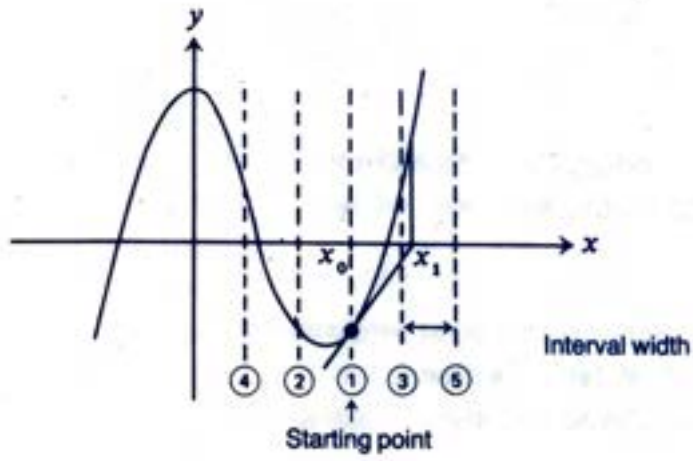
$$f'(X)=\frac{f(X+h)-f(X)}{h}$$ (h is the minute interval)

Change 1E−8 in line 340 to change the value for 10^{-8}.

■ INSTRUCTIONS

INPUT

Starting point
Minute interval
Interval



OUTPUTS

Root value (by pressing the ENTER key, the next interval's root is found)

■ EXAMPLE

$x^3-2x^2-x+2=0$ (the roots are −1, 1, 2)

starting point=0
minute interval=10^{-4}
interval=0.5

The above values are used in the calculation. The functions are to be written into lines after 500 as subroutines.

How to type in the example:

1. Go into PRO mode by operating the mode change key.
2. 500 B=((X−2)*X−1)*X+2 [ENTER]
   510 RETURN [ENTER] That is all that had to be done.

■ KEY OPERATION SEQUENCE

Step No.	Key Input	Display	Remarks
1	[DEF] [A]	STARTING POINT=_	Waiting for starting point input
2	0 [ENTER]	MINUTE INTERVAL=_	Waiting for minute interval input
3	0.0001 [ENTER]	INTERVAL=_	Waiting for interval width input
4	0.5 [ENTER]	ANSWER= 2.	Display of roots
5	[ENTER]	ANSWER= 1.	By repeatedly pressing the [ENTER] key the roots of the function are found.
6	[ENTER]	ANSWER= −1.	
7	[ENTER]	ANSWER= 1.	
8	[ENTER]	ANSWER= −1.	
9	[ENTER]	ANSWER= −1.	
10	[ENTER]	ANSWER= −1.	
11	[ENTER]	ANSWER= 2.	

■ PROGRAM LIST

```
10:"A": INPUT "STARTING
    POINT=";V
20:INPUT "MINUTE INTERV
   AL=";A
30:INPUT "INTERVAL=";W
40:G=V:F=V:Z=0
50:IF Z=0 GOTO 70
60:G=G-W:C=G: GOTO 80
70:C=G:Z=1
80:GOSUB 300
90:F=F+W:C=F
100:GOSUB 300
110:GOTO 50
120:END
300:X=C: GOSUB 500
310:Y=B:X=A+C
320:GOSUB 500
330:D=C:C=D-A*Y/(B-Y)
340:IF ABS (D-C)>=1E-8
    GOTO 300
350:BEEP 3: PRINT "ANSWE
    R=",C
360:RETURN
500:B=((X-2)*X-1)*X+2
510:RETURN
```

■ MEMORY CONTENTS

A	Minute interval
B	f(x)
C	X_0
D	f(x+h)
F	√
G	√
V	Starting point
W	Interval
X	x
Y	f(x)
Z	Initial flag

Program Title: AVERAGE, VARIANCE AND STANDARD DEVIATION

Refer to 248 page about instructions, contents, example and memory contents.

■ KEY OPERATION SEQUENCE

[Data Input]

Step No.	Key Input	Display	Remarks
1	DEF A	NO. OF DATA=_	Waiting for number of data input
2	5 ENTER	WEIGHTS=1/NO WEIGHTS=2?_	Waiting for the selection of weights/no weights
3	1 ENTER	X(1)=	
		?	
4	14.1 ENTER	F(1)=	
		?	
5	8 ENTER	X(2)=	
		?	
	⋮	⋮	
12	14.5 ENTER	F(5)=	
		?	
13	10 ENTER	>	End of the process

■ KEY OPERATION SEQUENCE

[Data revision]

Step No.	Key Input	Display	Remarks
1	DEF B	X(1)=14.1	
2	ENTER	F(1)=8	
3	ENTER	X(2)=14.1	
4	DEF C	X(2)=	DEF C is used to input the revised values when data errors are found
	ENTER	REVISION VALUE=?_	Wraiting for revision value input.
5	14.2 ENTER	F(2)=19	Revised value is input
		⋮	

[Calculation]

1	DEF D	TOTAL SUM=1072.5	Display of total sum
2	ENTER	MEAN VALUE=14.3	Display of average
3	ENTER	VARIANCE=1.432432432E−02	Display of variance
4	ENTER	STD.DEV.=	Display of standard deviation
5	ENTER	1.196842693E−01	
6	ENTER	>	Processing finished

■ PROGRAM LIST

```
 10:"A": CLEAR : WAIT 0
 20:INPUT "NO. OF DATA="
    ;P
 30:INPUT "WEIGHTS=1/NO
    WEIGHTS=2?";A
 40:IF A=2 DIM X(P-1):
    GOTO 70
 50:IF A=1 DIM X(P-1),F(
    P-1): GOTO 70
 60:GOTO 30
 70:FOR I=0 TO P-1
 80:B$="X("+ STR$ (I+1)+
    ")="
 85:PAUSE B$: INPUT X(I)
    : GOTO 100
 90:GOTO 85
100:IF A=2 GOTO 150
120:B$="F("+ STR$ (I+1)+
    ")="
130:PAUSE B$: INPUT F(I)
    : GOTO 150
140:GOTO 130
150:NEXT I: END
200:"B": WAIT :I=0
210:B$="X("+ STR$ (I+1)+
    ")=":J=1: PRINT B$;X
    (I)
230:IF A=1 LET B$="F("+
    STR$ (I+1)+")=":
    PRINT B$;F(I):J=2
240:I=I+1
250:IF I=P END
255:GOTO 210
260:"C": PAUSE B$: IF
    LEFT$ (B$,1)="X"
    INPUT "REVISION VALU
    E=";X(I): GOTO 290
270:IF LEFT$ (B$,1)="F"
    INPUT "REVISION VALU
    E=";F(I): GOTO 290
280:GOTO 250
290:IF J=1 GOTO 230
291:GOTO 210
300:"D":N=0:T=0:S=0: FOR
    I=0 TO P-1:X=X(I)
305:F=1: IF A=1 LET F=F(
    I)
310:N=N+F:T=T+F*X:S=S+F*
    X*X: NEXT I
400:WAIT :X=T/N:Q=(S-N*X
    *X)/(N-1):S=√Q:
    PRINT "TOTAL SUM=";T
    : PRINT "MEAN VALUE=
    ";X
410:PRINT "VARIANCE=";Q:
    PRINT "STD. DEV.=":
    PRINT S: END
```

Program Title: INTERSECTION BETWEEN CIRCLES AND STRAIGHT LINES

■ OVERVIEW

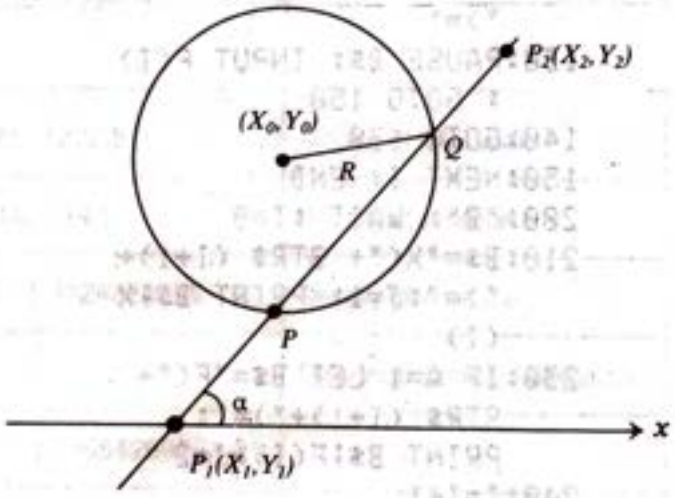
The points of intersection between circles and straight lines in the X–Y plane are found.

■ CONTENTS

The 2 points of intersection between a circle and a straight line are P and Q.

(Note) The angles are in degrees, minutes, and seconds and are to be input in the following way:

123.1423=123 degrees 14 minutes 23 seconds.



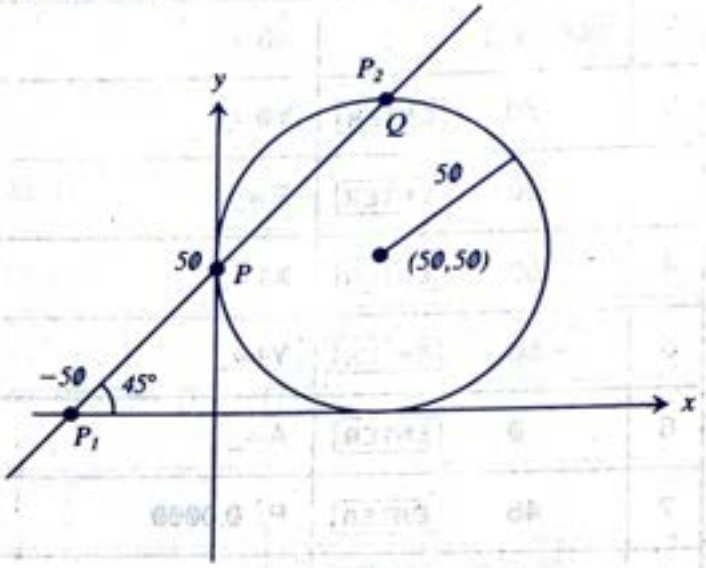
■ INSTRUCTIONS

1. If the straight line is determined by 2 points, DEF A is used.
   If the line is determined by 1 point and 1 direction angle, DEF B is used.
2. After the data are input, the results are displayed.

■ EXAMPLE

$(X_1,\ Y_1)=(-50,\ 0)$
$(X_2,\ Y_2)=(50,\ 100)$ $(X_p,\ Y_p)=(0,\ 50)$
$(X_0,\ Y_0)=(50,\ 50)$ $(X_q,\ Y_q)=(50,\ 100)$
$R = 50$
$\alpha = 45°$

(Note) The coordinate values are accurate up to 5 decimal places.



■ KEY OPERATION SEQUENCE

[When 2 points on the line are known]

Step No.	Key Input	Display		Remarks
1	DEF A	X0=_		
2	50 ENTER	Y0=_		
3	50 ENTER	R=_		
4	50 ENTER	X1=_		
5	−50 ENTER	Y1=_		
6	0 ENTER	X2=_		
7	50 ENTER	Y2=_		
8	100 ENTER	P: 0.0000	49.9999	(x_p, y_p)
9	ENTER	Q: 50.0000	100.0000	(x_q, y_q)
10	ENTER	>		End

■ KEY OPERATION SEQUENCE

[When 1 point on the line and 1 direction angle are known]

Step No.	Key Input	Display	Remarks
1	DEF B	X0=_	
2	50 ENTER	Y0=_	
3	50 ENTER	R=_	
4	50 ENTER	X1=_	
5	−50 ENTER	Y1=_	
6	0 ENTER	A=_	
7	45 ENTER	P: 0.0000 49.9999	(x_p, y_p)
8	ENTER	Q: 50.0000 100.0000	(x_q, y_q)
9	ENTER	>	End

■ PROGRAM LIST

```
10:"A":J=0: GOTO 30
20:"B":J=1
30:DEGREE : INPUT "X0=
   ";A,"Y0= ";B,"R= ";C
40:INPUT "X1= ";D,"Y1=
   ";E
50:IF J<>0 INPUT "A= ";
   H:H= DEG H: GOTO 90
60:INPUT "X2= ";F,"Y2=
   ";G
70:X=F-D:Y=G-E: GOSUB 5
   00
80:H=X
90:X=A-D:Y=B-E: GOSUB 5
   00
100:K=W* SIN (X-H)
110:L= ACS (K/C)
120:M=H-90-L:N=H-90+L
130:GOSUB 600
140:PRINT USING "######.
   ####";"P:";O;P
150:M=N: GOSUB 600
160:PRINT "Q:";O;P
170:END
500:W=√(X*X+Y*Y)
510:X= ACS (X/W): IF Y<0
   LET X=360-X
520:RETURN
600:O=A+C* COS M:P=B+C*
   SIN M: RETURN
```

■ MEMORY CONTENTS

A	X_0
B	Y_0
C	R
D	X_1
E	Y_1
F	X_2
G	Y_2
H	α
J	√
K	h
L	α
M	Q_p
N	Q_q
O	X_p,X_q
P	Y_p,Y_q
W	L
X	ΔX,θ
Y	ΔY

Program Title: NUMBER OF DAYS CALCULATION

Refer to 234 page about instructions, example and memory contents.

■ KEY OPERATION SEQUENCE

Step No.	Key Input	Display	Remarks
1	DEF A	START YEAR=_	
2	1976 ENTER	MONTH=_	Base date 1976 year 10 month 5 day input
3	10 ENTER	DAY=_	
4	5 ENTER	END YEAR=_	
5	1982 ENTER	MONTH=_	Target date 1982 year 6 month 4 day input
6	6 ENTER	DAY=_	
7	4 ENTER	DAYS= 2068.	
8	ENTER	END YEAR=_	
9	1985 ENTER	MONTH=_	Target date 1985 year 1 month 1 day input
10	1 ENTER	DAY=_	
11	1 ENTER	DAYS= 3010.	
12	ENTER	END YEAR=_	
13	DEF Z	>	End

■ PROGRAM LIST

```
10:"A"
20:INPUT "START YEAR=";
   R,"MONTH=";S,"DAY=";
   T
30:INPUT "END YEAR=";F,
   "MONTH=";V,"DAY=";W
50:H=R
60:G=S:I=T
70:GOSUB 500
 80:J=I
100:H=F
110:G=V:I=W
120:GOSUB 500
130:X=I-J
140:WAIT : USING : PRINT
    "DAYS=",X
150:GOTO 30
500:IF G-3>=0 LET G=G+1:
    GOTO 520
510:G=G+13:H=H-1
520:I= INT (365.25*H)+
    INT (30.6*G)+I
530:I=I- INT (H/100)+
    INT (H/400)-306-122:
    RETURN
600:"Z": END
```

Program Title: TYPING PRACTICE

Refer to 227 page about instructions and memory contents.

■ KEY OPERATION SEQUENCE

Step No.	Key Input	Display	Remarks
1	DEF Z	GRADE(1, 2, 3)?_	Grade input
2	1 ENTER	AZBDC	
3	A	AZBDC A	
4	Z	AZBDC AZ	
	⋮	⋮	
	⋮	YOUR-SCORE=80.	After the 10 questions are answered the score is displayed
		YOUR SCORE IS BEST	If your score is higher than the high score the guidance is displayed
		>	
1	DEF A	HIGH-SCORE=80.	When you want to play in the same grade
		BWVS	
2	B	BWVS B	
	⋮	⋮	
	⋮	YOUR-SCORE=60.	
		>	

■ PROGRAM LIST

```
10:"Z": CLEAR : DIM B$(
   5),C$(5): RANDOM
15:INPUT "GRADE(1,2,3)?
   ";L: WAIT 0
17:IF (L=1)+(L=2)+(L=3)
   <>1 THEN 15
18:GOTO 30
20:"A": WAIT 0:P=0:
   PAUSE "HIGH-SCORE=";
   X
30:FOR S=1 TO 10
40:B= RND 4+2:Y$="":R=
   INT (B/2)
50:FOR C=0 TO B-1:C$(C)
   =""
60:D= RND 26:B$(C)=
   CHR$ (D+&40):Y$=Y$+
   CHR$ (D+&40): NEXT C
   :A$=""
70:BEEP 3:E=0: WAIT 30:
   USING "&&&&&&&"
80:FOR W=1 TO B*10/L:
   PRINT Y$;"   ";A$:
   IF E=B LET W=B*20/L:
   GOTO 100
85:C$(E)= INKEY$ : IF C
   $(E)="" THEN 100
87:A$=A$+C$(E)
90:E=E+1
100:NEXT W:Q=0
110:FOR W=0 TO B-1: IF B
   $(W)=C$(W) LET Q=Q+1
120:NEXT W: IF Q<=R THEN
   150
130:IF Q=B LET P=P+10:
   GOTO 150
40:P=P+5
50:NEXT S: USING : BEEP
   3: PAUSE "YOUR-SCORE
   =";P
60:IF P>X LET X=P: WAIT
   100: PRINT "YOUR SCO
   RE IS BEST"
70:END
```

Program Title: SOFTLANDING GAME

■ OVERVIEW

This game involves landing a rocket, with only a limited amount of fuel, as softly as possible. The rocket is in free fall. The engine is used to slow down the free falling rocket. If ignition takes place too soon or too much fuel is used, then the rocket is thrust back out into space and becomes dust around the planet.
If all the fuel is burned up, the rocket hits the planet and blows up.
The aim is to land the rocket as softly as possible by controlling the engines while watching how much fuel is burned.

■ CONTENTS

Gravity is set to be 5 m/(unit time)^2.
If 5 units of fuel per a unit time are burnt, then gravity is offset.

Equations

$H=H_0+V_0t+\frac{1}{2}at^2$

$V=V_0+at$

$V^2=V_0^2+2aH$

$H_0=500,\ V_0=-50,\ F_0=200$

H: height
V: speed
a: gravitational acceleration
t: time

H_0: initial height
V_0: initial speed
F_0: initial fuel
F : fuel burned

The initial height, initial fuel level, and the wait time is stored in line 30 as data. By changing these values the above variables can be changed.

■ INSTRUCTIONS

1. It is started by pressing DEF A . Press 0 ~ 9 keys to adjust the amount of fuel used to land the rocket.

■ KEY OPERATION SEQUENCE

Step No.	Key Input	Display	Remarks
1	DEF A	***START***	
2	Keys 0 ~ 9 designate fuel burned in unit time	H: 500 S:-50 F: 200 C:0	
	9	H: 452 S:-46 F: 191 C:9	
	⋮		
	⋮	Repeat	
	⋮		
	(If successful)	SUCCESS!!	
		FUEL LEFT: F=15	
	(If failed)	GOOD BYE!!	
		REPLAY (Y/N)?	Wait for input on whether you wish to play again
	Y		Play again
	N	>	End

■ PROGRAM LIST

```
10:"A": WAIT 50: CLEAR
   : USING :S=-50:A=0:D
   $=""
20:BEEP 3: PRINT "  ***
    START ***"
30:DATA "TIME=",50,"FUE
   L=",200,"HEIGHT=",50
   0
40:RESTORE
50:READ B$,W,B$,F,B$,H
60:WAIT W
70:PRINT USING "####";"
   H:";H;" S:";S;" F:";
   F;" C:"; STR$ C
80:IF F<=0 GOTO 170
90:BEEP 1:D$= INKEY$
100:IF D$="" LET C=A:
   GOTO 130
110:C= VAL D$
120:A=C
130:IF C>F LET C=F
140:F=F-C:X=C-5:H=H+S+X/
   2:S=S+X
150:IF H>0 GOTO 70
160:IF ( ABS H(5)+( ABS
   S(5)=2 BEEP 5: PRINT
   "SUCCESS!!": GOTO 18
   0
170:BEEP 3: PRINT "GOOD
   BYE!!": GOTO 190
180:WAIT 150: PRINT
   USING "####";"FUEL L
   EFT :F=";F
190:WAIT 50: PRINT "REPL
   AY (Y/N) ?":Z$=
   INKEY$
200:IF (Z$="Y")+(Z$="N")
   <>1 GOTO 190
210:IF Z$="Y" GOTO 10
220:END
```

■ MEMORY CONTENTS

A	V
B$	V
C	Fuel burned
D$	Fuel burned
F	Initial fuel level, fuel left
H	Initial height, height
S	Speed
W	Wait time
X	V
Z$	V

Program Title: MEMORY CHECKER

Refer to 230 page about instructions and memory contents.

Points	Evaluation Message
0	IDIOT
1	BAD
2	AVERAGE
3	OK
4	GOOD!
5	*INTELLIGENT*
6	**GENIUS**

■ KEY OPERATION SEQUENCE

Step No.	Key Input	Display	Remarks
1	DEF A	MEMORY CHECK	Title
2		**XXXX **XXXX **XXXX	Display of problem line (5 sec.) *...character X...number
3		ANS.=_	Waiting for the input of set 1
4	(Example) AB1234 ENTER	ANS.=_	Waiting for the input of set 2
5	**XXXX ENTER	ANS.=_	Waiting for the input of set 3
6	**XXXX ENTER	**XXXX **XXXX **XXXX	Display of the problem line (1.5 sec.)
		**XXXX **XXXX **XXXX	Display of the answer input
		IDIOT	display of category
		BAD	
		AVERAGE	
		OK	
		GOOD!	
		INTELLIGENT	
		GENIUS	
		*REPLAY(Y/N)?_	Player input request
7	Y or N ENTER	>	If Y, go to step 2
			If N, END

■ PROGRAM LIST

```
 10:"A": USING : WAIT 20
    0: PRINT "MEMORY CHE
    CK": CLEAR : RANDOM
 20:DIM G$(6)*1,N$(10)*1
    ,V$(3)*3,X$(3)*6,Z$(
    3)*3,Y$(3)*6
 30:FOR I=1 TO 9:N$(I)=
    STR$ I: NEXT I:N$(10
    )="0"
 50:FOR I=1 TO 6
 60:J= RND 26:J=J+64
 70:G$(I)= CHR$ (J):
    NEXT I
 80:FOR I=1 TO 3
 90:Y$(I)=" "
100:FOR J=1 TO 3:K= RND
    9
110:Y$(I)=Y$(I)+N$(K):
    NEXT J
120:L= RND 9:J=(I-1)*2+1
130:A$(I)=G$(J)+G$(J+1)+
    N$(L)
140:H$=Y$(I):A$(I+3)=
    RIGHT$ (H$,3): NEXT
    I
150:GOSUB 500
160:FOR I=1 TO 3
170:INPUT "  ANS. = ";X$
    (I):X$(I)= LEFT$ (X$
    (I),6)
180:Z$(I)= LEFT$ (X$(I),
    3)
190:V$(I)= RIGHT$ (X$(I)
    ,3): NEXT I
200:GOSUB 520: GOSUB 500
    : GOSUB 520
210:N=0
220:FOR I=1 TO 3
230:IF A$(I)=Z$(I) LET N
    =N+1
240:IF A$(I+3)=V$(I) LET
    N=N+1
250:NEXT I
260:N=N+1
270:WAIT 150: ON N GOTO
    300,310,320,330,340,
    350,360
300:BEEP 1: PRINT "  IDI
    OT": GOTO 370
310:BEEP 1: PRINT "  BAD
    ": GOTO 370
320:BEEP 2: PRINT "  AVE
    RAGE": GOTO 370
330:BEEP 2: PRINT "   OK
    ": GOTO 370
340:BEEP 3: PRINT "   GO
    OD!": GOTO 370
350:BEEP 4: PRINT "* INT
    ELLIGENT *": GOTO 37
    0
360:BEEP 5: PRINT "**GEN
    IUS**"
370:W$="": BEEP 1: INPUT
    "* REPLAY (Y/N) ? ";
    W$
380:IF W$="N" THEN 600
390:IF W$="Y" THEN 50
395:GOTO 370
400:GOTO 370
500:WAIT 300: BEEP 2:
    PRINT A$(1);A$(4);"
      ";A$(2);A$(5);"
    ";A$(3);A$(6)
510:RETURN
520:WAIT 80: BEEP 1:
    PRINT USING "&&&&&&"
    ;X$(1); USING ;"   "
    ; USING "&&&&&&";X$(
    2); USING ;"   ";
    USING "&&&&&&";X$(3)
525:USING
530:RETURN
600:END
```

Program Title: BUGHUNT

■ OVERVIEW

This is a game involving a man chasing after a bug.

■ CONTENTS

The bug moves according to random numbers.
The man chases the bug and kills it.

The man moves by using the [8] [4] [6] [2] keys. (INKEY$ is used)

Each time the man moves one space, so does the bug. (Sometimes the bug will stay in the same place)

Initially the man is in position (0,0).

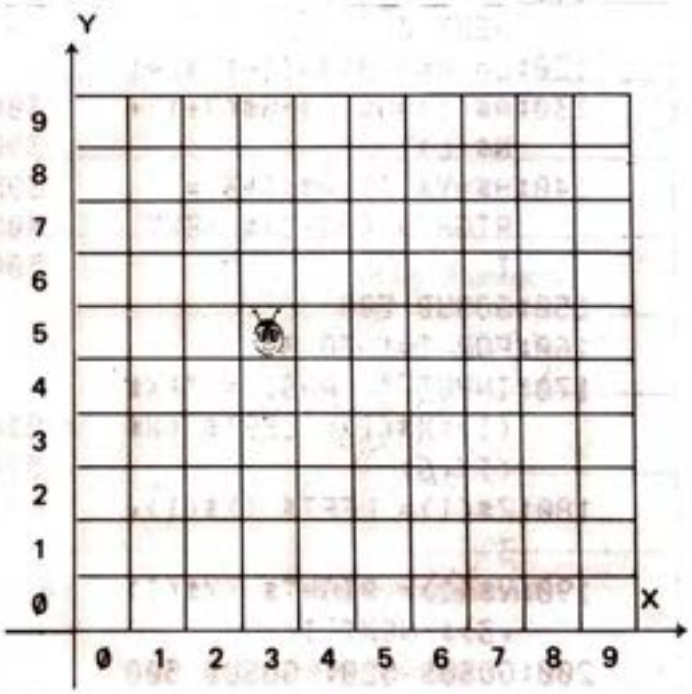
The bug is placed at a position that was chosen randomly. Hints are displayed as distance. The distance is displayed by the ABS(X−a)+ABS(Y−b) equation.

The initial energy level is 100. This decreases by 1 with time. Each time that a bug is killed, the energy increases by 5, 10, or 15. (The amount is chosen randomly.)

The score is determined by how many bugs were killed when the energy level reaches 0.

(The position of the bug may "warp" when cornered.)

The program can be started by either pressing RUN [ENTER] or [DEF] [A].



Position of the man (X, Y)
Position of the bug (a, b)

Concerning the display (Small characters are actual values)

(X, Y)	DISTANCE=*l*	E=e

Present position (X coordinate, Y coordinate) | Hint (distance) | Remaining energy

● Each time the man moves the display changes

Bug is caught

HIT! HIT!

BANG! BANG!

SCORE t ENERGY e

Concerning the BEEP sound

* Hint: When the distance is 1 the BEEP goes off 3 times
  2 " 2
  3 " 1
* If the distance is greater than 3 no BEEP is given.
* When the bug is caught, the BEEP goes off 5 times.

■ KEY OPERATION SEQUENCE

Step No.	Key Input	Display	Remarks
1	DEF A	**BUGHUNT GAME**	
	8	(0,0) DISTANCE=5 E=100	
	6	(0,1) DISTANCE=4 E=99	
	8	(1,1) DISTANCE=2 E=98	2 BEEPs
		HIT! HIT!	5 BEEPs
		BANG! BANG!	
		SCORE1 ENERGY 108	

■ PROGRAM LIST

```
 10:"A": RANDOM : WAIT 2
    50: PRINT "** BUGHUN
    T GAME **": BEEP 3
 20:X=0:Y=0:E=100:F=100:
    T=0:S=0
 30:A= RND 9:B= RND 9
 40:L= ABS (X-A)+ ABS (Y
    -B)
 50:IF X=A AND Y=B GOTO
    400
100:IF L=1 BEEP 3
110:IF L=2 BEEP 2
120:IF L=3 BEEP 1
130:WAIT 50: PRINT "(";
    STR$ (X);",";STR$ (
    Y);") DISTANCE=";
    STR$ (L);" E=";STR$
    (E)
150:S=S+1:E=F- INT (S/2)
153:IF E<=0 THEN 500
155:G$= INKEY$ : IF G$="
    " GOTO 130
157:BEEP 1
160:IF G$="2" LET Y=Y-1:
    GOTO 210
170:IF G$="4" LET X=X-1:
    GOTO 210
180:IF G$="6" LET X=X+1:
    GOTO 210
190:IF G$="8" LET Y=Y+1:
    GOTO 210
200:GOTO 150
210:IF X<0 LET X=0: GOTO
    150
220:IF Y<0 LET Y=0: GOTO
    150
230:IF X>9 LET X=9: GOTO
    150
240:IF Y>9 LET Y=9: GOTO
    150
250:IF X=A AND Y=B GOTO
    400
260:E=F- INT (S/2)
270:IF E<=0 GOTO 500
280:R= RND 5
290:IF R=1 LET B=B-1:
    GOTO 340
300:IF R=2 LET A=A-1:
    GOTO 340
310:IF R=3 LET A=A+1:
    GOTO 340
320:IF R=4 LET B=B+1:
    GOTO 340
340:IF A<0 OR A>9 GOTO 3
    70
350:IF B<0 OR B>9 GOTO 3
    70
360:GOTO 40
370:BEEP 4: PAUSE "*** W
    ARP ***": GOTO 30
400:PAUSE "HIT! HIT!"
410:BEEP 5
420:PAUSE "BANG! BANG!"
430:T=T+1:C= RND 3*5:F=F
    +C
435:E=F- INT (S/2)
440:WAIT 100: PRINT "SCO
    RE ";T;" ENERGY ";E
450:GOTO 30
500:WAIT : PRINT "SCORE
    "; STR$ (T);" *GAME
    OVER!!*"
510:END
```

■ MEMORY CONTENTS

A	Position of bug X coordinate
B	Position of bug Y coordinate
C	Amount of energy added
E	Remaining energy
F	Energy level
G$	Key read in
L	Distance between bug and man
R	Size of bug movement
S	Time spent
T	Score
X	Man position X coordinate
Y	Man position Y coordinate

Program Title: DOUBLE ROTATION

■ OVERVIEW

Quickly put in order A, B, C .....

This is a game that arranges randomly placed characters (A–J) in alphabetical order. When the letters are arranged in the right order, a score is displayed. The trick is to attack from the best place.

The sooner the characters are arranged, the better.

It is fun to race with 2 or 3 of your friends.

■ INSTRUCTIONS

1. After the program is initiated, by pressing DEF A , "DOUBLE ROTATION" is displayed. A random sequence of characters (A–J) is then displayed.
2. The space in between the characters is taken as the breakpoints (1–9) where the numbers are placed. Inputing a break number causes the characters on each side of the breakpoint to be rotated by moving them to the far ends of the row.
3. After the characters have been placed in order, the number of moves required is displayed as the score. The lower the score the better.
4. Start the game at DEF B . When you wish to replay with the same alphabets at the last game. And operate keys in the same manner with DEF A .

■ EXAMPLE

In (1) 4 is input, "F" and "I" move to each side changing the configuration to (2).

If 1 is now input, the "E" moves to the far right but. "F" stays in its place because it is already in the far left position, becoming configuration (3).

■ KEY OPERATION SEQUENCE

Step No.	Key Input	Display	Remarks
1	DEF A	DOUBLE ROTATION	
		A ~ J	Random requence display
2	1 ~ 9	⋮	Numbers between 1 and 9 are selected and input
		Repeated input	
		⋮	
		ABCDEFGHIJ	
		GAME END	
		YOUR SCORE 35.	
		>	
			Does player want to play using the same beginning random alphabets?
1	DEF B	A ~ J	
		Same as DEF A in succession	

■ PROGRAM LIST

```
 10:"A": CLEAR : WAIT 50
    : RANDOM : DIM B$(4)
 20:PAUSE "DOUBLE ROTATI
    ON"
 30:B$(0)="ABCDEFGHIJ"
 40:B$(1)=""
 50:A=0
 60:FOR I=1 TO 10
 70:R= RND 10
 80:S=2^(R-1)
 85:B=S AND A
 90:IF B<>0 GOTO 70
100:A=A OR S
110:B$(1)=B$(1)+ MID$ (B
    $(0),R,1): NEXT I
120:B$(2)=B$(1)
130:N=0
150:BEEP 1
170:D$="": PRINT B$(2):D
    $= INKEY$
180:C= VAL D$
190:IF C=0 GOTO 170
210:B$(3)= LEFT$ (B$(2),
    C)
220:B$(4)= RIGHT$ (B$(2)
    ,10-C)
240:IF C=1 GOTO 260
250:B$(3)= RIGHT$ (B$(3)
    ,1)+ LEFT$ (B$(3),C-
    1)
260:IF C=9 GOTO 280
270:B$(4)= RIGHT$ (B$(4)
    ,9-C)+ LEFT$ (B$(4),
    1)
280:B$(2)=B$(3)+B$(4)
290:N=N+1
300:IF B$(2)<>B$(0) GOTO
    150
310:BEEP 5: PAUSE "GAME
    END"
320:WAIT 200: PRINT
    USING "IIIII";"YOUR S
    CORE";N
330:END
400:"B": WAIT 50: GOTO 1
    20
```

■ MEMORY CONTENTS

A	V
B	V
C	V
D$	Input key
I	V
N	Score
R	Random numbers
S	V
B$(4)	Alphabet sequences

Note:

If the lines 180 and 240 of this program are changed as follows, the game is easier to play, and besides, the left end can be a breakpoint with the break number 0.

```
180: C = VAL  D$: IF  D$ = "0" GOTO
     210
240: IF  C = < 1  GOTO 260
```

INDEX

SERVICE CENTER ADDRESS

SHARP ELECTRONICS CORPORATION
9 Sharp Plaza, Paramus, New Jersey 07652
(201) 265-5600

SHARP ELECTRONICS CORPORATION
6478 Interstate 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

FOR YOUR RECORDS...

For your assistance in reporting this electronic calculator in case of loss or theft, please record below the model number and serial number which are located on the bottom of the unit.
Please retain this information.

Model Number _____ Serial Number _____

Date of Purchase _____ Place of Purchase _____

SHARP ELECTRONICS CORPORATION

CORPORATE HEADQUATERS AND EXECUTIVE OFFICES:
10 Sharp Plaza, Paramus, New Jersey 07652, Phone: (201) 265-5600

REGIONAL SALES OFFICES AND DISTRIBUTION CENTERS:
Eastern: 10 Sharp Plaza, Paramus, New Jersey 07652, Phone: (201) 265-5600
Midwest: 430 East Plainfield Road CountrySide, Illinois 60525, Phone: (312) 482-9292
Western: 20600 South Alameda Street, Carson, California 90810, Phone: (213) 637-9488

Printed in Japan
5G4.9T(TINSE4170CCZZ)⑧