

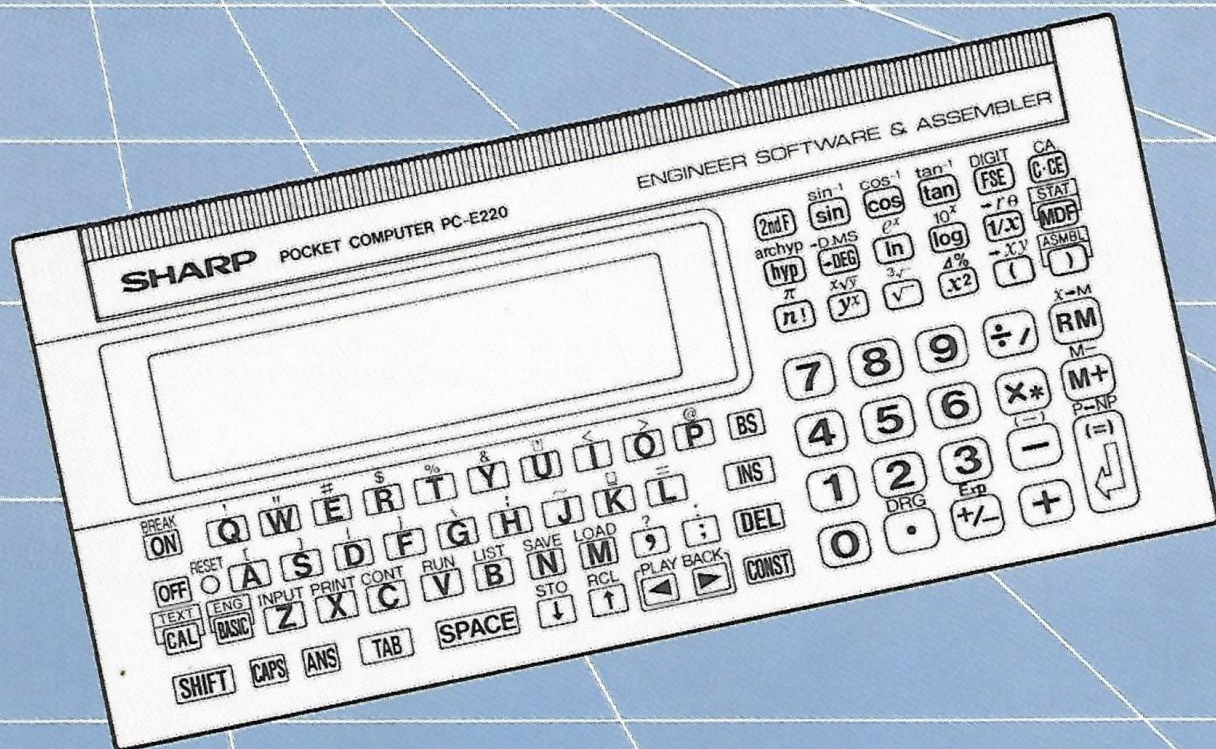
SHARP®

POCKET COMPUTER

MODEL

PC-E220

OPERATION MANUAL



INTRODUCTION

Your new SHARP PC-E220 Pocket Computer was designed with state-of-the-art technology, and incorporates many advanced features and capabilities:

- **Built-in assembler:** The PC-E220 comes standard with a built-in assembler, so that you can make yourself familiar with machine language programming by first writing your source program in TEXT mode, then assembling it in machine code. The PC-E220 uses a CMOS Central Processing Unit (CPU), which is equivalent to the popular Z80 processor.
- **Scientific calculations:** You can perform scientific calculations with ease and efficiency.
- **Statistical and regression calculations:** You can perform single- and two-variable statistic calculations or linear regression calculations.
- **Engineer Software:** The Engineer Software is resident in the computer, and lets you recall mathematical formulas, physical constants and calculate metric conversions and complex numbers.
- **Compatibility with existing software:** Software packages developed for our previous pocket and portable computers are compatible with the PC-E220. See Appendix H.
- **RAM disk:** Part of the internal memory can be used as a RAM disk, which lets you save and load your programs just as you would using a diskette.
- **Serial I/O interface:** You can transfer data directly between the PC-E220 and a personal computer.

Z80 is a registered trademark of Zilog, Inc.

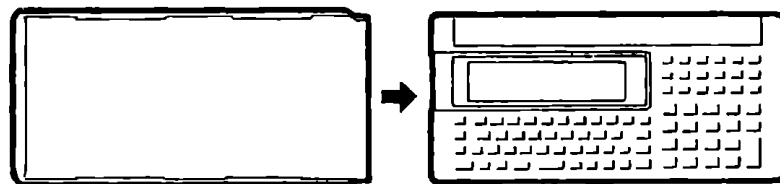
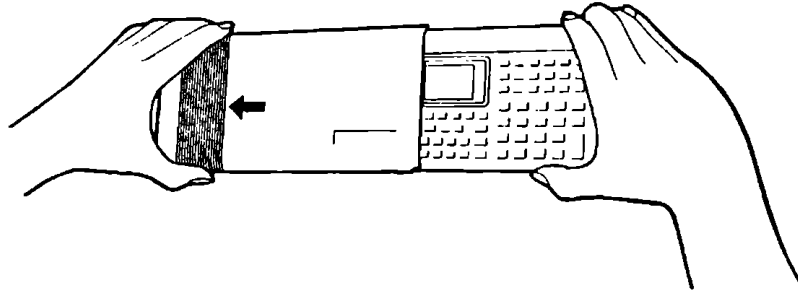
NOTICE

- SHARP strongly recommends that separate permanent written records be kept of all important data. Data may be lost or altered in virtually any electronic memory product under certain circumstances. Therefore, SHARP assumes no responsibility for data lost or otherwise rendered unusable whether as a result of improper use, repairs, defects, battery replacement, use after the specified battery life has expired, or any other cause.
- SHARP assumes no responsibility, directly or indirectly, for financial losses or claims, such as the loss of or alteration of stored data, etc., from third persons resulting from the use of this product and all of its functions.
- The information provided in this manual is subject to change without notice.

The Protective Cover (Hard Cover)

Your computer is supplied with a cover to protect the operation panel when the computer is not being used.

When the computer is to be used, remove the protective cover from the computer, as shown below.



When the computer is not being used, slide the protective cover over the operation panel, as shown below.

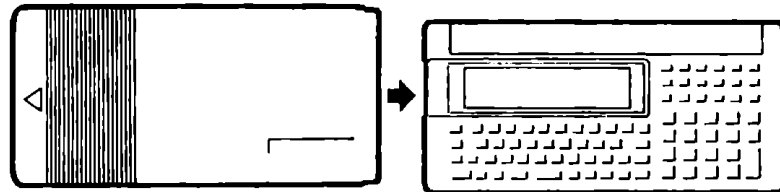


TABLE of CONTENTS

INTRODUCTION	i
USING THE PC-E220 FOR THE FIRST TIME	1
USING THIS MANUAL	4

PART 1 HARDWARE

1. HARDWARE: OVERVIEW	6
2. HARDWARE: IN DETAIL	8
Turning on the Computer	8
Auto OFF	8
Key Notation in this Manual	8
Selecting Modes	9
The Display	10
Battery Replacement	12
3. PERIPHERAL DEVICES	16
CE-126P Thermal Printer/Cassette Interface	16
Cassette Interface	17
Using the Cassette Interface	17
Using the CE-126P Printer/Cassette Interface	19

PART 2 DIRECT INPUT OPERATION

4. CAL MODE	22
Calculations	22
Basic Operations	26
Scientific Calculations	27
Use of Parentheses	31
Decimal Places	31
Modify Function	32
Priority Levels	32
5. ENGINEER SOFTWARE MODE	34
Engineer Software List	35
Factorization	36
Trigonometric Function	38
Integration	40
Greek	42
Physical Constant	43
Metric Conversion	45
Complex Number	48
6. STAT MODE	50
Selecting and Cancelling the STAT Mode	50
Single-Variable Statistical Calculations	50
Two-Variable Statistical Calculations	54

7. RUN MODE	58
Some Helpful Hints	58
Simple Calculations	59
Compound Calculations and Parentheses	59
Recalling Entries	60
Errors	62
Serial Calculations	62
Constant Calculations	63
Using Variables in Calculations	64
Last Answer Feature	65
Maximum Calculation Length	66
Scientific Calculations	66
Priority in Direct Input Calculations	70
Printing of Direct Input Calculations	71
Calculation Errors	71

PART 3 PROGRAM OPERATION

8. CONCEPTS AND TERMS OF BASIC	74
String Constants	74
Hexadecimal Numbers	74
Variables	75
Program Files (RAM disk)	80
Filenames	80
Extension	80
Expressions	80
Numeric Expressions	80
String Expressions	81
Relational Expressions	81
Logical Expressions	82
Parentheses and Operator Precedence	83
9. PROGRAMMING	84
Programs	84
BASIC Statements	84
Line Numbers	84
Labelled Programs	85
BASIC Commands	85
Direct Commands	86
Modes	86
Beginning to Program	86
Storing Programs in Memory	92
10. DEBUGGING	94
Debugging Procedures	95

PART 4 ASSEMBLER OPERATION

11. TEXT MODE (TEXT EDITOR)	98
Text Mode Functions	98
Selecting the TEXT Mode	99
Edit	99
Program Editing	100
Deleting a TEXT Program (Del)	101
Printing a TEXT Program Listing (Print)	101
Saving, Loading, and Verifying a TEXT Program with	
Cassette Tape (Cmt)	101
Serial I/O (Sio)	104
Program File (File)	107
BASIC Converter (Basic)	109
12. MACHINE LANGUAGE MONITOR	111
Using the Machine Language Monitor	111
Machine Language Monitor Command Reference	112
Error Messages in Monitor Mode	117
13. ASSEMBLER	118
Let's try Assembling	119
Source Program Coding and Editing	122
Assembling	126
Pseudo Instructions of the Assembler	130
Assembly Errors	134

PART 5 BASIC REFERENCE

14. SCIENTIFIC & MATHEMATICAL CALCULATIONS	136
Calculation Ranges	145
15. BASIC COMMAND DICTIONARY	147

PART 6 APPENDICES

A CE-T801 DATA TRANSFER CABLE	210
B ERROR MESSAGES	212
C CHARACTER CODE CHART	214
D KEY FUNCTIONS IN BASIC	216
E TROUBLESHOOTING	219
F MEMORY MAPS	221
G SPECIFICATIONS	222
H USING PROGRAMS FROM OTHER SHARP COMPUTERS	224
I CARE OF THE PC-E220	226

COMMAND INDEX

INDEX

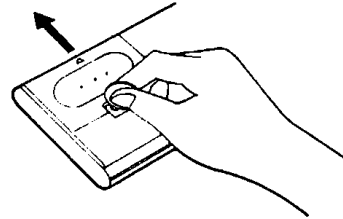
USING THE PC-E220 FOR THE FIRST TIME

Installing Batteries

The PC-E220 uses two types of batteries: four AA batteries to power the computer itself, and a lithium battery for memory backup.

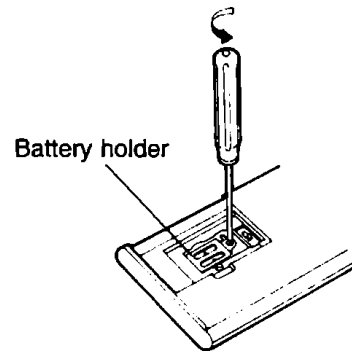
Install the four AA operating batteries and one memory backup lithium battery which are supplied, as follows:

1. Place the computer face-down on its protective cover, and loosen the screw securing the battery compartment lid using a coin or flat-head screwdriver.

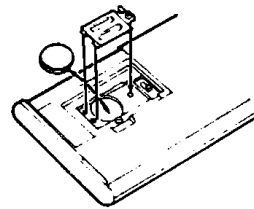


2. Remove the battery compartment lid by sliding it in the direction of the arrow.

3. Using a small screwdriver, unscrew the battery holder securing the backup lithium battery, and remove the holder.

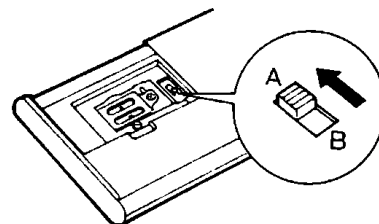


4. Wipe the supplied lithium battery with a dry cloth, and insert it into the compartment with the positive side down.



5. Replace the lithium battery holder in the compartment and secure it with the screw.

6. Set the Memory Backup switch to position A.



3 Press any key, and you will see the following display:

```
RUN MODE
>
```

Checking for Normal Computer Operation

To confirm that the PC-E220 is functioning normally, enter:

F **R** **E** **←**

```
RUN MODE
FRE                                     30435.
```

If the display appears as shown above, the PC-E220 is functioning normally and is in the Command Prompt mode.

The number "30435." indicates the memory capacity available for program or data.

Note:

If the PC-E220 display does not appear as shown for any of the steps above, read the description of the relevant step again and try to perform correct operation for that step.

If the **BATT** indicator is on, see page 12.

USING THIS MANUAL

This manual is design to introduce you to the capabilities and features of your PC-E220 and to serve as a reference tool. It has been divided into six parts, each of which introduces a particular aspect of the PC-E220. Although the manual is intended ultimately as a reference, we suggest that you carefully read the introductory parts, Part 1 (Hardware) and Part 2 (Operation), before use. The PC-E220 is a powerful tool and has many valuable and time-saving functions that even the seasoned computer use will be pleased to discover!

Part 1: Hardware

The first chapter in Part 1 provides an introduction to the features of the PC-E220. Chapter 2 describes basic handling of the computer, the operation of keys, and the meanings of various display symbols. Both chapters 1 and 2 are essential reading. Subsequent chapters explain use of peripheral devices (such as the printer and cassette tape recorder).

Part 2: Direct Input Operation

Part 2 is devoted to use of the PC-E220 as a calculator (scientific, engineering, and statistic) or "direct input" computer. Direct input refers to the independent use of BASIC commands (that is, commands not used within a BASIC program).

Part 3: Program Operation

Part 3 introduces the BASIC programming language as implemented on the PC-E220. Even if you have programmed in BASIC before, we hope you will read this part thoroughly, since there are many versions of the BASIC language. This chapter also contains time-saving information in the form of programming shortcuts and debugging techniques.

Part 4: Assembler

Part 4 explains the PC-E220's monitor and assembler functions, which are designed to help you learn machine language programming. The assembler running on the PC-E220's CPU, which is equivalent to the Zilog Z80, will help make you proficient in Z80 machine language

Part 5: BASIC Reference

Part 5 is an alphabetic listing of the numeric functions and BASIC commands used in programming on the PC-E220. Many of these commands can be used in the direct input modes of the PC-E220.

Part 6: Appendices

Part 6 mainly contains reference material, such as code tables, error messages and specifications. You will also find tips on how to keep your PC-E220 in good condition.

This manual is not intended to be a textbook for teaching yourself BASIC, which is beyond the scope of this manual. If you have never programmed in BASIC before, it is recommended you buy a separate book or attend a class on the subject before trying to work through this manual.

HARDWARE

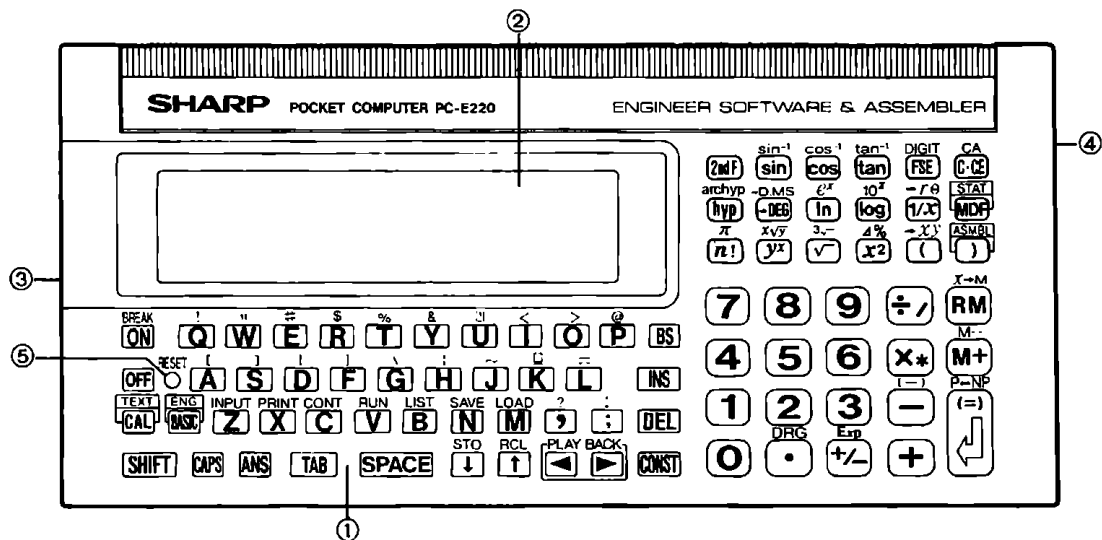
The first chapters of Part 1 introduce the features of the PC-E220*, explain the handling of the computer, operation of the keys, and meanings of the various display symbols. Subsequent chapters explain the use of peripheral devices (such as the printer and the cassette tape recorder).

* The PC-E220 is hereafter referred to as "the computer".

1. HARDWARE: OVERVIEW

The SHARP computer features a QWERTY keyboard layout that is similar to conventional typewriters, a liquid crystal display (LCD) with adjustable contrast, and an 11-pin interface connector.

The following pages describe the individual parts of the computer to acquaint you with their location and functions.



The **BREAK ON** and **OFF** keys have been recessed into the keyboard so that they will not be pressed by mistake.

1. Keyboard

The keys on the keyboard are laid out like those on a typewriter. There is also a numeric keypad, for a total of 82 keys.

2. LCD Screen

The display has four lines of 24 characters each. The characters are extra-large and the contrast is fully adjustable for comfortable viewing.

3. 11-Pin Connector

The 11-pin connector can be used to link the computer directly to optional peripheral equipment, such as the printer (CE-126P) and the cassette tape recorder.

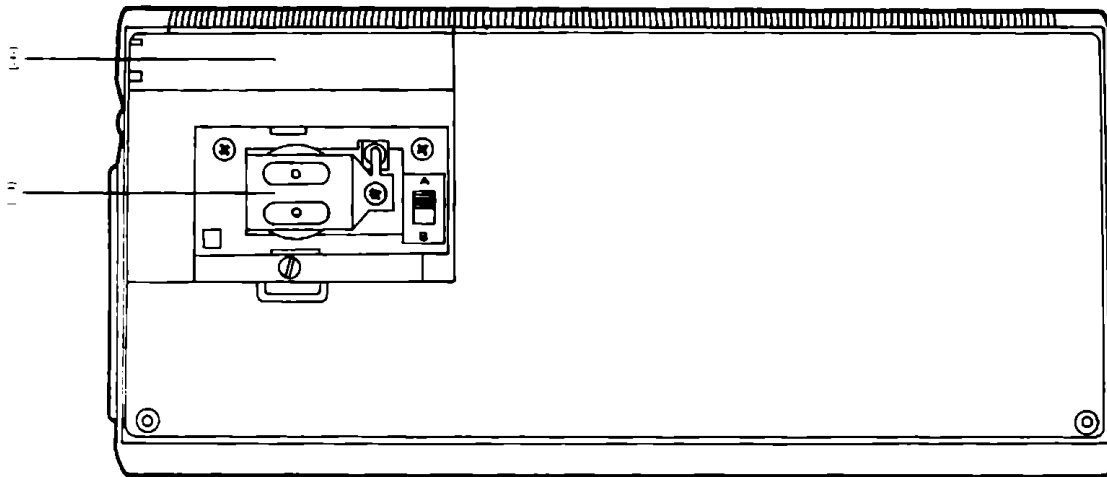
The PC-E220 can also be connected to a personal computer using the optional CE-T801 data transfer cable.

4. LCD Contrast Dial

Use this dial to lighten or darken the display to suit viewing conditions.

3. RESET Button

If the computer "hangs up" for some reason during operation, the keyboard will be inoperative. It may be impossible to use the power switch or **BREAK** key to restart the computer. Press the RESET button to clear the memory and restore the computer to the condition at power-on. Use this button with caution as you risk losing data and programs. See Appendix E for details on the use of this button.



3. Operating Battery Compartment

This compartment houses the four AA batteries required to power the computer. See Chapter 2 for details on replacing batteries.

3. Backup Battery Compartment

This compartment houses one lithium battery required to backup the internal memory.

2. HARDWARE: IN DETAIL

Turning on the Computer

Press the **ON** key, which is located on the left of the computer keyboard. The computer will be in the RUN mode when turned on.

When the computer is turned on or off, the display may black out, or dots, lines, or symbols may appear on it momentarily. This is not a malfunction.

If the **BATT** indicator is on, see page 12.

Auto OFF

To conserve battery power, the computer automatically turns itself off if no key has been pressed for about eleven minutes. Press the **ON** key to bring the computer back up to power if it has turned itself off.

The Auto OFF feature is disabled when the computer is executing the INKEY\$ command. However, it is enabled when the computer is executing the INPUT command.

If the computer is left unused for a long period with the Auto OFF feature disabled, battery power will eventually be consumed, causing any stored programs or data to be lost.

Key Notation in this Manual

ON : Turns the power on.
BREAK : Interrupts an operation or calculation.

CAL : Sets the computer to the CAL mode.
SHIFT + **TEXT** : Sets the computer to the TEXT mode.

sin : sin key
2nd F **sin⁻¹** : sin⁻¹ key

When numerals, characters, or symbols printed on the keys or just above each key are referred to in this manual, only letters that are relevant to the explanation will appear, with key boxes or the **SHIFT** key omitted, as shown in the following example:

S **H** **A** **R** **P** → SHARP
SHIFT + **R** **4** **5** → \$45

In the explanation of the CAL mode, the **(=)** key will appear as **=**, while in explanations of any other mode it will appear as **←**.

The Difference between the **SHIFT** and **2nd F** Keys

The **SHIFT** and **2nd F** keys are functionally identical. The only difference is that the **SHIFT** key must be pressed and held down when you press any other key, whereas the **2nd F** key is pressed and then released before any other key is pressed. In this manual, the **2nd F** key is used in explanations of calculations in the **CAL** mode and calculations of functions, whereas the **SHIFT** key is used for all other explanations.

Where a space must be entered with the **SPACE** key, it is indicated by symbol $_$, for example:

"SHARP_EL-865_WN-104"

Keys may appear in their full boxed image in this manual, such as \sin^{-1} , whenever needed.

To differentiate the number zero from the capital letter "O", zero appears as "0" on the computer's display. In explanations in this manual when necessary to avoid confusion, the number zero will also be shown as "0".

Selecting Modes

Before starting to use your computer, you must decide which mode to use. The computer has six operational modes.

- BASIC mode:** The BASIC mode is subdivided into the RUN and PRO modes.
- PRO mode:** Allows you to write or correct a BASIC program.
- RUN mode:** Allows you to execute a BASIC program or BASIC commands.
- CAL mode:** Allows you to use the computer as you would an ordinary calculator.
- ENG mode:** Allows you to use the Engineer Software.
- STAT mode:** Allows you to do statistical and regression calculations.
- TEXT mode:** Allows you to enter, edit, delete, save, or load text programs in ASCII format, or to convert them to BASIC, and vice versa.
- ASMBL mode:** Allows you to assemble a source (text) program into an object (machine code) program.

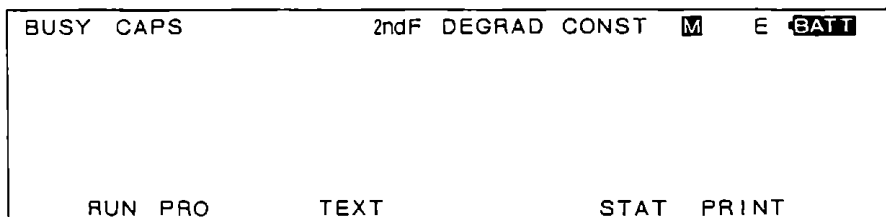
Key Table

A table showing the keys that call each mode is provided below.

Mode to select	Key operation
RUN mode	BASIC
PRO mode	BASIC or BASIC BASIC
CAL mode	CAL
ENG mode	SHIFT + ENG
STAT mode	SHIFT + STAT
TEXT mode	SHIFT + TEXT
ASMBL mode	SHIFT + ASMBL

← Press **BASIC** twice to switch to the PRO mode from within any mode but RUN.

The Display



The computer has a four line 24-character dot-matrix liquid crystal display with a status line above and below. Each character is made up of a 5 × 7 dot matrix. The display shows key labels, and calculation processes.









Display examples in this manual show only the symbols relevant to the explanation of the function.

In BASIC mode the display shows:

- > This symbol is the prompt that appears when the computer is waiting for input in the BASIC mode. As you type, the prompt disappears and is replaced by the cursor.
- ┌ █ The Cursor is the symbol that shows you the location of the next character to be typed in. As you begin typing, the cursor replaces the prompt. The cursor can be positioned over particular characters when using the INSert and DELete functions. The block cursor changes to an underline cursor when not positioned over an existing character.

The status lines consists of:

- BUSY** Displayed while the computer is executing a program or command.
- CAPS** Indicates that the computer is in the CAPS Lock mode, in which all alphabetic characters are entered in uppercase. When this indicator is not on the display, all alphabetic characters are entered in lowercase. The **CAPS** key lets you select or deselect the CAPS Lock mode.
- 2ndF** Displayed when the **2nd F** key is pressed and disappears with subsequent key entry. Remember, the **2nd F** key must be released before pressing any other key if that key's second function is to be used.
- DEG** Indicates that the Degree mode is selected for angular functions.
- RAD** Indicates that the Radian mode is selected for angular functions.
- GRAD** Indicates that the Gradient mode is selected for angular functions.
- CONST** Indicates that a constant is stored in the computer for constant calculation (see page 63). When this symbol is displayed, the computer performs constant calculation every time you press the **←** key. When the constant is no longer needed, clear it using **SHIFT** + **CA**.

-  Indicates that a number other than 0 is in memory for manual calculations.
-  Indicates that an ERROR has occurred.
-  Indicates that the operating batteries are low and should be replaced as soon as possible.
-  Indicates that the computer is in the RUN mode.
-  Indicates that the computer is in the PROgram mode.
-  Indicates that the computer is in the TEXT mode. To select the TEXT mode, use the **[SHIFT]** + **[TEXT]** keys.
-  Indicates that the computer is in the STATistics mode. This mode is alternately selected and deselected each time you press the **[2nd F]** **[STAT]** (or **[SHIFT]** + **[STAT]**) keys.
-  Indicates the computer is ready to send data to the printer in the RUN mode. Press **[SHIFT]** + **[P-HP]** keys to toggle on and off. (Only available when the optional printer is connected.)

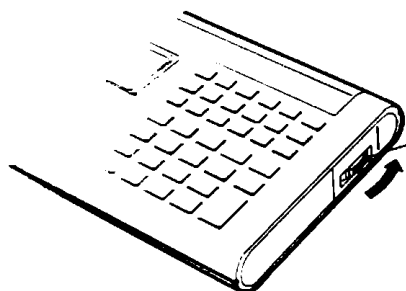
In the CAL mode, "<CAL>," "HYP," "FIX," or other symbols will be displayed on the top line. "<CAL>" indicates that the computer is in the CAL mode. See the explanation on page 24 regarding "FIX."

“CAN” or the Japanese symbols “カナ” and “小” may appear faintly on the top line of the display. These symbols do not affect any function or operation of the calculator.

Displays that Exceed four Lines

The display consists of four lines (24 character per line). Key entries or calculated results are displayed from the top line. If the characters to be displayed exceed four lines, the displayed contents will be scrolled up by one line (the first displayed line will scroll off the top of the display).

Contrast Control



Contrast control
(LCD contrast dial)

Turn the control in the direction of the arrow for darker display, and in the opposite direction for lighter display. Adjust the display so that it is easy to read.

Battery Replacement

The computer uses two types of batteries: four AA batteries for powering the computer itself, and a lithium battery for memory backup.

If the operating batteries are discharged when the CE-126P printer is used with the computer, power will be supplied from the CE-126P, so the current drawn from the operating batteries will be reduced.

Operating Battery Replacement Timing

If the **BATT** indicator appears in the upper right corner of the display, it indicates that the operating batteries are discharged. Immediately replace them with new ones. If you continue to use the computer with the **BATT** warning on the display, the computer will eventually turn off. After this, the computer cannot be turned on even if you press the **ON** key.

If an optional peripheral device is connected to the computer, the computer power may be supplied from the device. Note that, in this case, the **BATT** warning will remain off even if the computer's operating batteries are discharged. Before use, temporarily disconnect the peripheral device and check to make sure that the **BATT** warning does not appear on the display.

Note:

While you are replacing the AA batteries, the memory is backed up by a lithium battery. However, if you have purchased an optional peripheral device, it is advised that you save the program in the computer's memory to the storage device.

Replacing Operating Batteries

If the AA batteries require replacement, follow the replacement procedure below. If the procedure is not followed, the computer may remain inoperative or the life of the memory backup battery may be shortened.

You will need the following types of batteries:

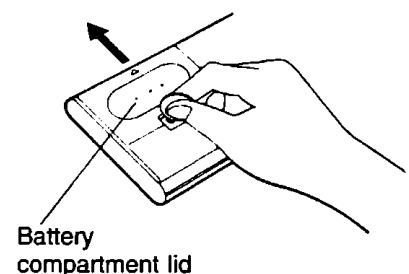
Operating battery: Four AA batteries (R06)

Memory backup battery: One lithium battery (CR2032)

1. Press **ON** then **OFF** to turn the computer off.
2. Using a coin or flat-head screwdriver, loosen the screw securing the battery compartment lid on the back of the computer, and remove the lid.

Note:

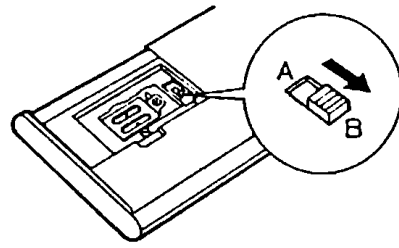
When removing the lid, be careful not to accidentally press the **ON** key on the front of the computer. If this occurs, turn the computer off with the **OFF** key. If you proceed to the following step 3 with the computer turned on, the contents of memory will be destroyed or completely lost.



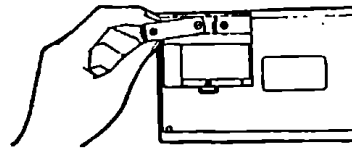
Set the Memory Backup switch to position B.

Note:

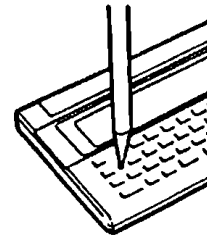
If the batteries are replaced with the switch left in position A, the contents of memory will be destroyed or completely lost.



- Replace the four operating batteries (be sure to replace all four batteries at the same time). Insert them negative end first.

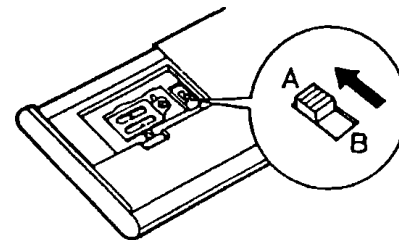


- Temporarily replace the battery compartment lid over the compartment. Turn the computer over, press the RESET button on the front with the tip of a ball-point pen or similar pointed object, and make sure that nothing appears on the display. If anything appears, press the RESET button again.



- Remove the battery compartment lid, and set the Memory Backup switch to position A.

- Replace the battery compartment lid and secure it firmly with the screw.



Note:

If the computer is turned on with the Memory Backup switch left at position B it will remain inoperative. If the switch is at position B, remove the battery compartment lid and set the switch to position A.

Backup Battery Replacement Timing

The memory backup battery has a backup capacity of about five years at room temperature (20°C/68°F). The contents of the memory remain intact for this duration even when the operating batteries are discharged or are being replaced. Note, however, that the backup battery will discharge quickly if the operating batteries are left discharged or are removed.

Note:

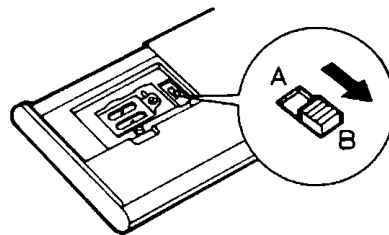
The battery life may be shortened under harsh operating environments with, for example, temperature extremes.

Replacing the Memory Backup Battery

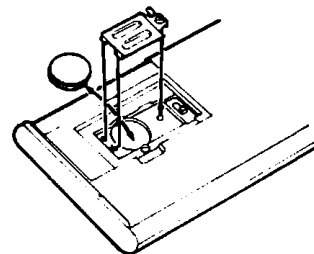
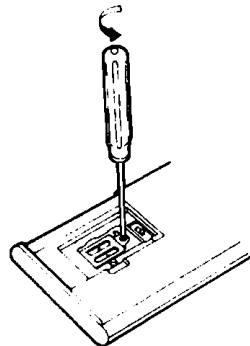
When replacing the memory backup battery, check to make sure that the four AA batteries in the computer are not low (no **BATT** warning appears). If they are low, first replace the four batteries before replacing the backup battery. If you remove the backup battery when the AA batteries are low, contents of the memory may be lost.

Follow the replacement procedure below:

1. Press **ON** then **OFF** to turn the computer off.
2. Using a coin or flat-head screwdriver, loosen the screw on the back of the computer securing the battery compartment lid, and remove the lid.
(When removing the lid, be careful not to accidentally press the **ON** key on the front of the computer. If this occurs, turn the computer off with the **OFF** key. If you proceed to the following step 3 with the computer turned on, the contents of memory will be destroyed or completely lost.)
3. Set the Memory Backup switch to position B.



4. Using a small screwdriver, unscrew the battery holder that secures the backup lithium battery, and remove the holder.
5. Wipe the replacement lithium battery with a dry cloth. Remove the old battery from the compartment and insert the new one with the positive side down.



Note:

Be careful that the operating AA batteries do not pop out of the compartment while replacing the backup battery.

- Replace the backup battery holder in the compartment and secure it with the screw.
- Set the Memory Backup switch to position A.
- Replace the battery compartment lid and secure it firmly with the screw.

Battery Handling Notes

Batteries, if misused, can explode or cause electrolyte leakage. Pay special attention to the following points:

- Be sure to replace all four AA batteries at the same time.
- Do not mix new batteries with old batteries in the same unit.
- Replacement batteries should be of the same type as those to be replaced.
- Some types of batteries are rechargeable, while other types are not. Read the description on the battery carefully and choose the unchargeable type.
- Insert the batteries in the correct position as indicated in the battery compartment.

Caution:

- **Keep batteries out of the reach of children.**
- Remove used batteries from the compartment. Otherwise, the computer may be damaged from electrolyte leakage.
- Do not throw batteries into a fire as this may result in an explosion.

3. PERIPHERAL DEVICES

The computer can be used with the following optional SHARP peripherals:

MODEL	DESCRIPTION	CONNECTOR
CE-126P	Printer/Cassette Interface	11-pin connector
CE-T801	Data Transfer Cable	11-pin connector

A brief description of each device is given below. For detailed information, refer to the individual operation manuals. The CE-T801 data transfer cable is an available option that will enable the PC-E220 to transfer data to and from a personal computer. (See Appendix A.)

A cross-reference of I/O commands that can be used with different types of devices is provided in the table below.

	RAM disk	Cassette tape	SIO	
SAVE LOAD	○	×	×	}
CSAVE CLOAD	×	○	×	
PRINT # INPUT #	×	○	○	}

○: Can be used ×: Cannot be used

When in the TEXT mode, text programs in ASCII format can be transferred to the RAM disk, cassette tape, or through the SIO.

CE-126P Thermal Printer/Cassette Interface

The optional CE-126P Printer/Cassette Interface allows you to add a printer and to connect a cassette recorder to your computer.

The CE-126P features:

- 24-character wide thermal printing.
- Convenient paper feed and tear bar.
- Simultaneous printing of calculations, if desired.
- Easy control of printer output from BASIC programs.
- Built-in cassette interface with remote function.
- Manual and program control of recorder for storing programs and data.
- Battery powered for portability.

To connect the computer to the CE-126P, refer to the operation manual supplied with the CE-126P.

Cassette Interface


The interface is required when interfacing the computer to a cassette recorder to save and load programs to tape (unless the built-in interface on the CE-126P interface is used). It plugs into the 11-pin connector on the computer and into the REMOTE and EARPHONE jacks of the recorder. Use a cassette recorder that meets the following specifications:

Item	Requirements
Recorder Type	Any standard cassette or micro cassette recorder
Input Jack	"MIC" mini input jack (never the "AUX" jack)
Input Impedance	Low input jack impedance (200 – 1000 ohms)
Minimum Input Level	Below 3 mV or -50 dB
Output Jack	EXTERNAL, MONITOR, or EARPHONE mini output jack, or equivalent
Output Impedance	Below 10 ohms
Output Level	Above 1V (maximum practical output above 100 mW)
Distortion	Within 15% (range: 2 kHz to 4 kHz)
Noise and Flutter	0.3% maximum (WRMS)
Other	Stable speed recorder motor


Using the Cassette Interface

Recording (saving) onto magnetic tape
See Tape Notes on page 19.

To save a program to magnetic tape from the computer, use the following procedure:

1. Switch off the REMOTE switch on the CE-126P.
2. Enter a program into the computer.
3. Load the tape in the recorder. Advance the tape to the position where you want the program to be recorded, being careful to avoid the clear tape leader (non-magnetic mylar material) and any programs previously recorded.
4. Connect the red plug on the interface to the MIC jack and the black plug to the REMOTE jack on each device.
5. Switch on the REMOTE switch.
6. Simultaneously press the RECORD and PLAY buttons on the tape recorder.
7. Enter recording instructions (CSAVE command), and press the  key to start execution, as follows:
First, set the computer to the RUN or PRO mode. Next, enter the following key sequence:

CSAVE "filename" 
For example: CSAVE "AA" 

The tape begins to run when you press the  key, leaving a non-signal blank of several seconds. The filename and contents are then recorded. To output data, enter statements in the program, as shown in the following example:


```
100 OPEN "CAS:DATA" FOR OUTPUT
110 PRINT #1, AB,C$,D(5)
120 CLOSE #1
```

8. When recording is completed, the PROMPT symbol (>) will be displayed and the tape recorder will automatically stop. Now you have your program on tape (it also remains in the computer).


Use the tape counter on the recorder to locate programs on tape.

Verifying a Saved Program

After transferring a program to or from tape, you can verify that the program on tape and the program in the computer are identical (and thus be sure that data is correct before continuing programming or running programs) using the CLOAD? command.

1. Switch off the REMOTE switch.
2. Position the tape just before the file that you want to check.
3. Connect the gray plug to the EARphone jack and the black plug to the REMote jack on each device.
4. Switch on the REMOTE switch.
5. Press the PLAY button.
6. Enter the CLOAD? command and start execution with the  key, as follows. First, set the computer to the RUN or PRO mode. Next, enter the following key sequence:

The filename you entered previously.

CLOAD? "AA" 

The computer will automatically search for the specified filename and compare the contents on tape with the contents in memory.

When the specified filename is found on the tape, an asterisk(*) is automatically appended to the line typed in on the screen. This indicates that checking has started. The PROMPT symbol (>) is displayed if the programs are identical. If the programs differ, execution will be interrupted and an error message displayed. If this happens, try again.

Loading from Magnetic Tape

See Tape Notes below.

To load a program from magnetic tape into the computer, use the following procedure:

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


Connect the gray plug to the EARphone jack and the black plug to the REMote socket on each device (if the recorder has no REMote socket, use the PAUSE button to control tape movement manually).

Turn on the REMOTE switch.

Press the PLAY button on the tape recorder (playback mode).

Set the VOLUME control between middle and maximum.

Set TONE to maximum treble.


Enter transfer instructions (CLOAD command), and press the  key in the following manner:

Put the computer into the RUN or PRO mode. Then press the following keys:


TYPE filename" 

For example: CLOAD "AA" 

The specified filename will be automatically searched for and its contents transferred to the computer.

Once the specified file is found on tape, loading begins. This is indicated by an  that is automatically appended to the line typed in on the screen.

When the program has been transferred, the computer will automatically stop the tape and display the PROMPT (>) symbol.

If an error occurs, try loading again from the beginning. If an error occurs again, repeat the process after adjusting the volume up or down a little. If no error code is displayed but the tape does not stop, something is wrong. Press the  to stop the tape and try again.

TIPS

Always use the same tape recorder for checking or loading that was used for saving the program. Using a different model may generate errors.

Use high quality cassette tapes only. Standard audio tapes should not be used.



Using an AC adaptor for the CE-126P interface can occasionally cause hum which will affect the recording signal. If this occurs, switch to battery use.


When reusing an old tape for recording programs, erase all old programs on the tape before recording.

Using the CE-126P Printer/Cassette Interface

Using the Printer


When using the computer for direct input calculations (in the RUN mode only),

you can use the CE-126P to simultaneously print your calculations. Press the  key, and then the  key (P<->NP) while in the RUN mode.

Press the  key at the end of a calculation. This will print the contents of the display on one line and the results on the next.

For example:

100

100 

Paper

300 / 50	6 .
----------	-----

You may use the printer from within BASIC programs by using the LPRINT command (see the BASIC COMMAND DICTIONARY for details). Use LPRINT in the same format as the PRINT command.

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

Notes:

- When the printer is exposed to strong external electrical noise, it may print numbers at random. If this occurs, press the **BREAK** key to stop the printing. Switch the CE-126P off, then on, and then press the **C•CE** key. Pressing the **C•CE** key will return the printer to its normal condition.
- When not in use, switch off the CE-126P to prolong battery life.

DIRECT INPUT OPERATION

The PC-E220* allows direct input calculations in the CAL (calculator), ENG (Engineer Software), STAT (statistics), and RUN modes (as opposed to calculations made as part of a program in PRO mode). PART 2 of this Operation Manual describes the use of these modes.

Since there is some overlapping of functions between modes, PART 2 starts with an overview of PC-E220 operation and mode selection. The PRO mode is then discussed in detail in PART 3.

*The PC-E220 is hereafter referred to as "the computer".

4. CAL MODE

You can use the computer as a 10-digit function calculator. To do this, you must first set the computer in the CAL mode. Press the **CAL** key. The "<CAL>" indicator will appear in the upper left corner of the display.

Note:

In the CAL mode, the results of calculations cannot be output to the printer.

Calculations

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

<u>Enter</u>	<u>Display</u>
123	123.
+	123.
654	654.
=	777.

Did you get the correct answer? If you didn't, press the **C•CE** key, and try the same calculation again.

Now call up the value of pi (π).

The symbol " π " is printed in brown above the **n!** key. The functions identified by brown letters can be used by first pressing the brown **2nd F** key, and then pressing the required function key.

Now press the **2nd F** **n!** keys.

<u>Enter</u>	<u>Display</u>
2nd F π	3.141592654

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

Next, compute 10^4 . For this calculation, you should use the function 10^x . This function is also identified in brown, so the **2nd F** key must be pressed.

Enter

4 **2nd F** **10^x**

Display

10000.
($10^4 = 10000$)

An outline of some the major key functions:

* **C•CE** (clear/clear entry) (red key)

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

Enter

123 **+** 456

Display

456.

C•CE

0.

789 **=**

912.
($123 + 789 = 912$)

6 **X*** 2 **+**

12.

C•CE

0.

6 **÷/** 2 **+**

3.

5 **=**

8.

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

Enter

5 **÷/** 0 **=**

Display

Error symbol

<CAL>	E	
		0 .

C•CE

<CAL>	
	0 .

* **FSE** (display mode switch)

This key is used to switch the display mode for the result of a calculation from the floating point decimal system (normal mode) to the fixed point decimal (FIX), scientific notation (SCI), or engineering notation (ENG) system, or vice versa.

<u>Enter</u>	<u>Display</u>
23 [X*] 1000 [=]	<div style="display: flex; justify-content: space-between;"><CAL></div> <div style="text-align: center; margin-top: 20px;">23000.</div>
FSE	<div style="display: flex; justify-content: space-between;"><CAL>FIX</div> <div style="text-align: center; margin-top: 20px;">23000.000</div>
FSE	<div style="display: flex; justify-content: space-between;"><CAL>SCI</div> <div style="text-align: center; margin-top: 20px;">2.300E 04</div>
FSE	<div style="display: flex; justify-content: space-between;"><CAL>ENG</div> <div style="text-align: center; margin-top: 20px;">23.000E 03</div>

(Normal)

(FIX)

(SCI)

(ENG)

* **DIGIT** (specifies the number of decimal places)

This key is used to specify the number of decimal places when used in conjunction with a numeral key. Turn the power switch off and then on again. Press the **FSE** key and the display will show "0.000" (FIX mode).

Example:

1. Specify 2 decimal places.

<u>Enter</u>	<u>Display</u>
2ndF DIGIT 2	<div style="display: flex; justify-content: space-between;"><CAL>FIX</div> <div style="text-align: center; margin-top: 20px;">0.00</div>
5 [÷/] 8 [=]	<div style="display: flex; justify-content: space-between;"><CAL>FIX</div> <div style="text-align: center; margin-top: 20px;">0.63</div>

2. Specify 5 decimal places.

2nd F **DIGIT** **5**

```
<CAL>          F I X
                0 . 6 2 5 0 0
```

* **DRG** (specifies angular unit)

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

Enter

Display

```
DEG
```

(Degrees)

2nd F **DRG**

```
RAD
```

(Radians)

2nd F **DRG**

```
GRAD
```

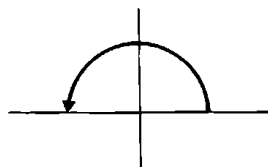
(Grads)

2nd F **DRG**

```
DEG
```

(Degrees)

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



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

* to , , , and

: Used to enter a number in exponential form (the display shows “E” following the number entered).

<u>Enter</u>	<u>Display</u>
<input type="text" value="C•CE"/> 4 <input type="text" value="2nd F"/> <input type="text" value="Exp"/> 3	4.E 03 (4 × 10 ³)
<input "="" type="text" value="="/>	4000.
<input type="text" value="+/-"/>	-4000.

: Used to enter a negative number (or to reverse the sign from negative to positive).

<u>Enter</u>	<u>Display</u>
1.23 <input type="text" value="+/-"/>	-1.23
<input type="text" value="2nd F"/> <input type="text" value="Exp"/> 5 <input type="text" value="+/-"/>	-1.23E-05 (-1.23 × 10 ⁻⁵)
<input "="" type="text" value="="/>	-0.0000123
<input type="text" value="+/-"/>	0.0000123

Basic Operations

1. Addition and Subtraction

Enter: 12 45.6 32.1 789 741 213
 Answer: 286.5

2. Multiplication, Division

Enter: 841 586 12
 Answer: 41068.83333
 Enter: 427 54 32 7 39 2
 Answer: 595.8571429

Note:

Multiplication and division have priority over addition and subtraction. In other words, multiplication and division will occur before addition and subtraction.

Constant Multiplication: The first number entered is a constant.

Enter: 3 $\boxed{\times}$ 5 $\boxed{=}$ Answer: 15

Enter: 10 $\boxed{=}$ Answer: 30

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

Enter: 15 $\boxed{\div}$ 3 $\boxed{=}$ Answer: 5

Enter: 30 $\boxed{=}$ Answer: 10

Note:

The computer places some calculations in pending status depending on their priority levels. Accordingly, in successive calculations the operator and numerical value of the calculation last performed in the computer are handled as a calculation instruction and a constant for the next calculation, respectively.

$a + b \times c =$ + bc (Constant addition)

$a + b \div c =$ $\div c$ (Constant division)

$a \div b \times c =$ $\frac{a}{b} \times$ (Constant multiplication)

$a \times b - c =$ - c (Constant subtraction)

3. Memory Calculations

The independent memory can be accessed using the $\boxed{\times M}$, \boxed{RM} , $\boxed{M+}$, and $\boxed{M-}$ keys. Before starting a calculation, clear the memory by pressing $\boxed{C \div CE}$ and $\boxed{\times M}$. If a value other than 0 is stored into the memory, "M" will be displayed.

Enter: 12 $\boxed{+}$ 5 $\boxed{M+}$

Answer: 17

To subtract, enter: 2 $\boxed{+}$ 5 $\boxed{2nd F}$ $\boxed{M-}$

Answer: 7

Enter \boxed{RM} to recall memory: 10 is displayed.

Enter: 12 $\boxed{\times}$ 2 $\boxed{=}$ $\boxed{2nd F}$ $\boxed{\times M}$

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

Enter: 8 $\boxed{\div}$ 2 $\boxed{M+}$

Answer: 4 \boxed{RM} : 28

In CAL mode, 26 memories, specified with \boxed{STO} \boxed{A} through \boxed{STO} \boxed{Z} , are available, as well as the memory specified with the $\boxed{\times M}$ key.

Pressing the \boxed{STO} \boxed{A} keys assigns data to BASIC numeric variable A.

To read this data, press the \boxed{RCL} \boxed{A} keys.

In RUN mode, pressing the \boxed{A} $\boxed{\leftarrow}$ keys is identical to pressing the \boxed{RCL} \boxed{A} keys in CAL mode.

Scientific Calculations

To perform trigonometric or inverse trigonometric functions, and coordinates conversion, designate the angular unit for the calculation. The angular unit DEG, RAD, or GRAD is specified using the $\boxed{2nd F}$ \boxed{DRG} keys.

Note:

The section on Errors in Appendix B deals with the calculation limits of the computer.

1. Trigonometric Functions

Set the angular unit to DEG.

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

Enter: 30 $\boxed{\sin}$ + 40 $\boxed{\cos}$ $\boxed{=}$

Answer: 1.266044443

Calculate: $\cos 0.25\pi$

Set the angular unit to RAD.

Enter: 0.25 $\boxed{\times}$ $\boxed{2nd F}$ $\boxed{\pi}$ $\boxed{=}$ $\boxed{\cos}$

Answer: 0.707106781

2. Inverse Trigonometric Functions

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

Set the angular unit to DEG.

Enter: 0.5 $\boxed{2nd F}$ $\boxed{\sin^{-1}}$

Answer: 30

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

Set the angular unit to RAD.

Enter: 1 $\boxed{+/-}$ $\boxed{2nd F}$ $\boxed{\cos^{-1}}$

To enter a negative number, press the $\boxed{+/-}$ key after the number.

Answer: 3.141592654 (value of π)

The calculation results of the respective inverse trigonometric functions will be displayed within the following limits:

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

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

RAD: $-\pi/2 \leq \theta \leq \pi/2$ [rad]

GRAD: $-100 \leq \theta \leq 100$ [g]

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

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

RAD: $0 \leq \theta \leq \pi$ [rad]

GRAD: $0 \leq \theta \leq 200$ [g]

3. Hyperbolic and Inverse Hyperbolic Functions

Calculate: $\sinh 4$

Enter: 4 $\boxed{\text{hyp}}$ $\boxed{\sin}$

Answer: 27.2899172

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

Enter: 9 $\boxed{2nd F}$ $\boxed{\text{archyp}}$ $\boxed{\sin}$

Answer: 2.893443986

4. Power Functions

Calculate: 20^2

Enter: 20 $\boxed{x^2}$

Answer: 400

Calculate: 3^3 and 3^4

Enter: 3 $\boxed{y^x}$ 3 $\boxed{=}$

Answer: 27

Enter: 3 $\boxed{y^x}$ 4 $\boxed{=}$

Answer: 81

5. Roots

Calculate: $\sqrt{25}$

Enter: 25 $\boxed{\sqrt{\quad}}$

Answer: 5

Calculate: Cubic root of 27

Enter: 27 $\boxed{2nd F}$ $\boxed{\sqrt[3]{\quad}}$

Answer: 3

Calculate: Fourth root of 81

Enter: 81 $\boxed{2nd F}$ $\boxed{x\sqrt[y]{\quad}}$ 4 $\boxed{=}$

Answer: 3

6. Logarithmic Functions

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

Natural Logarithms:

Enter: 21 $\boxed{\ln}$

Answer: 3.044522438

Common Logarithms:

Enter: 173 $\boxed{\log}$

Answer: 2.238046103

7. Exponential Functions

Calculate: $e^{3.0445}$

Enter: 3.0445 $\boxed{2nd F}$ $\boxed{e^x}$

Answer: 20.99952881 (21 as in Natural Logarithms above)

Calculate: $10^{2.238}$

Enter: 2.238 $\boxed{2nd F}$ $\boxed{10^x}$

Answer: 172.9816359 (173 as in Common Logarithms above)

8. Reciprocals

Calculate: $1/6 + 1/7$

Enter: 6 $\boxed{1/x}$ $\boxed{+}$ 7 $\boxed{1/x}$ $\boxed{=}$

Answer: 0.309523809

9. Factorial

Calculate: 69!

Enter: 69 $\boxed{n!}$

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

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

Enter: 8 $\boxed{n!}$ $\boxed{\div}$ ($\boxed{8}$ $\boxed{-}$ 3 $\boxed{)}$ $\boxed{n!}$ $\boxed{=}$

Answer: 336

10. Percentage Calculations

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

Enter: 2780 $\boxed{\times}$ 45 $\boxed{2nd F}$ $\boxed{\Delta\%}$

Answer: 1251

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

Enter: 547 $\boxed{-}$ 473 $\boxed{2nd F}$ $\boxed{\Delta\%}$

Answer: 15.6448203

11. Angle/Time Conversions

To convert an angle given in the sexagesimal system (degrees/minutes/seconds) to its decimal equivalent, a value in degrees must be entered as an integer and values in minutes and seconds as decimal fractions, respectively.

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

Enter: 12.4752 $\boxed{+DEG}$

Answer: 12.79777778

When converting an angle in decimal degrees to its sexagesimal equivalent (degrees/minutes/seconds), the answer is broken down as integer part = degrees; 1st and 2nd decimal digits = minutes; 3rd and 4th digits = seconds; and the 5th digit and up = fractions of seconds.

Convert 24.7256 to its sexagesimal equivalent (degrees/minutes/seconds)

Enter: 24.7256 **2nd F** **→DMS**

Answer: 24.433216 or 24°43'32"

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

Enter: 0.0225 **→DEG** **+** 0.0238 **→DEG** **+** 0.0222 **→DEG** **=**

Answer 1: 0.123611111

Enter: **÷/** 3 **=**

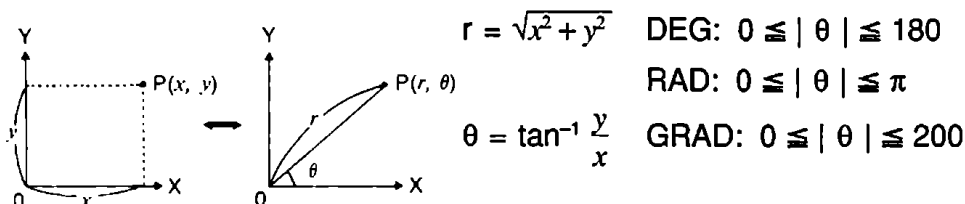
Answer 2: 0.041203703

Enter: **2nd F** **→DMS**

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

12. Coordinate Conversion

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



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

Angular unit: DEG

Enter: 6 **▶** (or **◀**) 4 **2nd F** **→rθ** Answer: 7.211102551 (r)

Enter: **▶** (or **◀**) Answer: 33.69006753 (θ)

Calculate the magnitude and direction (phase) of vector $i = 12 + j9$

Enter: 12 **▶** (or **◀**) 9 **2nd F** **→rθ** Answer: 15 (r)

Enter: **▶** (or **◀**) Answer: 36.86989765 (θ)

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

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

Angular unit: RAD

Enter: **2nd F** **π** **÷/** 3 **=** **▶** (or **◀**) 14 **▶** (or **◀**) **2nd F** **→xy**

Answer: 7.000000002 (x)

Enter: **▶** (or **◀**)

Answer: 12.12435565 (y)

In the above example, $\theta = \pi/3$ is entered first and is replaced with $r = 14$ by pressing the **▶** (or **◀**) key after r is entered.

Use of Parentheses

The parentheses keys are needed to cluster a series of operations when it is necessary to override the priority system of algebra. Calculations in parentheses have priority over other calculations. When parentheses are in use in the computer, “()” will be displayed. Parentheses in the CAL mode can be used up to 15 times in succession. The calculation within the innermost set of parentheses will be performed first.

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

Enter: 12 $\boxed{+}$ 42 $\boxed{\div/}$ $\boxed{(}$ 8 $\boxed{-}$ 6 $\boxed{)}$ $\boxed{=}$

Answer: 33

Calculate: $126 \div \{(3 + 4) \times (3 - 1)\}$

Enter: 126 $\boxed{\div/}$ $\boxed{(}$ $\boxed{(}$ 3 $\boxed{+}$ 4 $\boxed{)}$ $\boxed{\times*}$ $\boxed{(}$ 3 $\boxed{-}$ 1 $\boxed{)}$ $\boxed{)}$ $\boxed{=}$

Answer: 9

Note:

The $\boxed{)}$ key operation located just before the $\boxed{=}$ or $\boxed{M+}$ key operation can be omitted.

Decimal Places

The $\boxed{2nd F}$ and \boxed{DIGIT} keys are used to specify the number of decimal places in the calculation result. The number of decimal places after the decimal point is specified by a numeral key ($\boxed{0}$ - $\boxed{9}$) pressed after the $\boxed{2nd F}$ and \boxed{DIGIT} keys. In this case, the display mode must be fixed decimal point (FIX), scientific notation (SCI), or engineer notation (ENG).

- $\boxed{2nd F}$ \boxed{DIGIT} $\boxed{0}$ → Designates 0 decimal places.
(The number is rounded to the nearest integer.)
- $\boxed{2nd F}$ \boxed{DIGIT} $\boxed{1}$ → Designates 1 decimal place.
(The number is rounded to 1 decimal place.)
- $\boxed{2nd F}$ \boxed{DIGIT} $\boxed{9}$ → Designates 9 decimal places.
(The number is rounded to 9 decimal places.)
- $\boxed{2nd F}$ \boxed{DIGIT} $\boxed{\cdot}$ → Clears designation.

Decimal place designation is also cleared when the computer is turned off or another mode is selected. The display will now be in the normal display mode.

Example:

- $\boxed{2nd F}$ \boxed{DIGIT} $\boxed{9}$
0.5 $\boxed{\div/}$ 9 $\boxed{=}$ → 0.055555556 (FIX mode)
(The number is rounded to 9 decimal places)
- \boxed{FSE} → 5.55555556E-02 (SCI mode)
(The mantissa is rounded to 9 decimal places.)
- $\boxed{2nd F}$ \boxed{DIGIT} $\boxed{3}$ → 5.556E-02 (SCI mode)
(The mantissa is rounded to 3 decimal places.)
- \boxed{FSE} → 55.556E-03 (ENG mode)

FSE

→ 0.0555555555

This is held by the computer in the form of $5.5555555555 \times 10^{-2}$. Rounding the 11th digit of the mantissa results in $5.555555556 \times 10^{-2}$. When the display mode is changed to the floating decimal point mode, the rounded part may not be displayed as in this example.

This function cannot be used for statistical or regression calculations.

Modify Function

While the Decimal Place function lets the computer display only the specified number of decimal places, the computer internally stores data in a full 12-digit length, so the display data may have some departure from internal data. To match the internal data with the display data, use the Modify function.

Example:

2nd F	DIGIT	2	
C•CE	5	÷/	9 =
X*	9	=	
C•CE	5	÷/	9 = MDF
X*	9	=	

Display: 0.56 (FIX)

Display: 5.00 (FIX)

Display: 0.56 (FIX)

Display: 5.04 (FIX)

Priority Levels

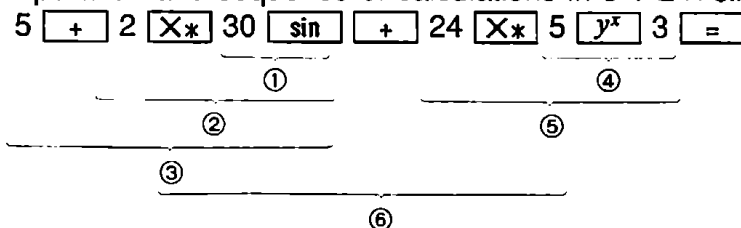
The computer is provided with a function that judges the priority levels of individual calculations, which permits keys to be operated according to a given mathematical formula. The following shows the priority levels of individual calculations.

Level Operations

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

Example:

Key operation and sequence of calculations in $5 + 2 \times \sin 30 + 24 \times 5^3 =$



The numbers ① – ⑥ indicate the sequence in which the calculations are carried out. When calculations are executed in sequence from the higher priority level function, a lower priority one must be set aside.

Note:

The computer has a memory area for up to three pending operations for calculations without parentheses.

As the memory area can also be used for calculations including parentheses, calculations can be performed according to a given mathematical formula, unless the levels of parentheses and/or pending operations exceeds eight in total.

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

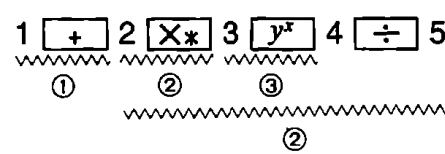
Calculation without Parentheses

Example:

1 $\boxed{+}$ 2 $\boxed{=}$ 1 pending calculation

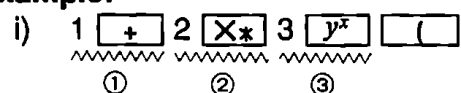


1 $\boxed{+}$ 2 $\boxed{\times}$ 3 $\boxed{=}$ 2 pending calculations

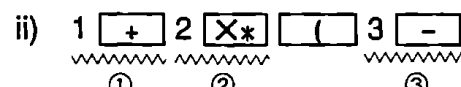
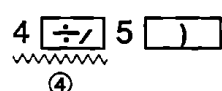

1 $\boxed{+}$ 2 $\boxed{\times}$ 3 $\boxed{y^x}$ 4 $\boxed{=}$ 3 pending calculations


1 $\boxed{+}$ 2 $\boxed{\times}$ 3 $\boxed{y^x}$ 4 $\boxed{\div}$ 5 After $\boxed{y^x}$ is pressed, 3 calculations remain pending. Pressing the $\boxed{\div}$ key executes the calculation of "y^x" highest in priority level and "x" identical in priority level. After $\boxed{\div}$ is pressed, the other 2 calculations remain pending.


Calculation with Parentheses

Example:

i) 1 $\boxed{+}$ 2 $\boxed{\times}$ 3 $\boxed{y^x}$ $\boxed{(}$ 4 numeral and calculation instructions are left pending.

 4 $\boxed{\div}$ 5


ii) 1 $\boxed{+}$ 2 $\boxed{\times}$ $\boxed{(}$ 3 $\boxed{-}$ Pressing the $\boxed{(}$ key executes the calculation of $3 - 4 \div 5$ in the parentheses, leaving 2 calculations pending.

 4 $\boxed{\div}$ 5 $\boxed{)}$


Note:

Parentheses can be used if pending calculations do not exceed eight. However, a maximum of 15 parentheses can be used in succession in a calculation.

Example:

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



5. ENGINEER SOFTWARE MODE

The Engineer Software allows you to view mathematical formulas and physical constants or to calculate metric conversions and complex numbers.

Selecting a Program

The Engineer Software has a two-page program menu screen. You can select either page with the or key. To select and execute a program, type in the program number on the menu screen.

An Engineer Software program must be loaded into the program data area before it can be executed. When you select an Engineer Software program for the first time, the computer will ask if you are sure you want to overwrite any BASIC program in the program data area (the inquiry will not appear if there is no BASIC program in the program area). If you do not want to overwrite, press to exit the Engineer Software mode, then save your BASIC program to the RAM disk or cassette tape. If you are sure you want to overwrite the BASIC program, press at the prompt, and the computer will execute the program you selected. **The initial display for the selected program will appear after a brief delay, during which "GOTO 100" is displayed.**

Note:

Data associated with the BASIC program will also be lost if the program is overwritten.

If you wish, for example, to use the FACTORIZATION program when you are currently using the INTEGRATION program, first press the key once, then the + keys, and the computer will display the Engineer Software program menu. Then press the appropriate numeric key to select the FACTORIZATION program.

You can view or modify the contents of any Engineer Software program in the Program mode, and store the modified contents to the RAM disk or other media. Since the original contents of the program are stored in ROM, you can restore them at any time.

The explanations in this section assume that you are already in the Engineer Software mode, accessed with the + keys. In the Engineer Software mode, you can sequentially recall formulas with the or key. The formulas can be successively recalled by holding down these keys.

Notes:

- If you make a typing error during numeric data entry, delete the wrong entry with the key, then enter the correct data.
- When the computer is in the Engineer Software mode, you may find the key ineffective in turning the power off. In such an event, press the key before pressing the key.

Once you execute any of the Engineer Software programs, any BASIC program and all values assigned to BASIC variables will be cleared.

For fractions or square roots ($\sqrt{\quad}$) contained in formulas, the computer uses the following display notations:

(e.g.) • $\tan 2\theta = \frac{2 \tan \theta}{1 - \tan^2 \theta} \rightarrow \tan 2\theta = 2 \tan \theta / (1 - \tan^2 \theta)$

• $\sin \frac{\theta}{2} = \pm \sqrt{\frac{1 - \cos \theta}{2}} \rightarrow \sin(\theta/2) = \pm \sqrt{(1 - \cos \theta) / 2}$

- Physical constants appearing in this manual are subject to change.
- The Engineer Software may not yield as much data accuracy as the user expects. Carefully check the accuracy you wish for calculation results and that of the data you use for the calculation.

Engineer Software List

Approximate size refers to the number of bytes used by the program. (Variables are excluded.)

Program Name	Approximate Size (bytes)
FACTORIZATION	5,800
TRIGONOMETRIC FUNCTION	10,100
INTEGRATION	7,900
GREEK	2,000
PHYSICAL CONSTANT	8,200
METRIC CONVERSION	6,700
COMPLEX NUMBER	3,200

FACTORIZATION

The FACTORIZATION program displays any of the following 26 factorization formulas:

- (1) $a^2 - b^2 = (a + b)(a - b)$
- (2) $a^3 \pm b^3 = (a \pm b)(a^2 \mp ab + b^2)$
- (3) $a^4 - b^4 = (a + b)(a - b)(a^2 + b^2)$
- (4) $a^4 + b^4 = (a^2 + \sqrt{2}ab + b^2)(a^2 - \sqrt{2}ab + b^2)$
- (5) $a^n - b^n = (a - b)(a^{n-1} + a^{n-2}b + a^{n-3}b^2 + \dots + b^{n-1})$
- (6) $a^n + b^n = (a + b)(a^{n-1} - a^{n-2}b + a^{n-3}b^2 - \dots + b^{n-1})$ (n : an odd number)
- (7) $a^2 \pm 2ab + b^2 = (a \pm b)^2$
- (8) $a^3 \pm 3a^2b + 3ab^2 \pm b^3 = (a \pm b)^3$
- (9) $(a \pm b)^2 \mp 4ab = (a \mp b)^2$
- (10) $a^2 + b^2 + c^2 + 2bc + 2ca + 2ab = (a + b + c)^2$
- (11) $a^4 + a^2b^2 + b^4 = (a^2 + ab + b^2)(a^2 - ab + b^2)$
- (12) $a^3 + b^3 + c^3 - 3abc = (a + b + c)(a^2 + b^2 + c^2 - bc - ca - ab)$
- (13) $(ac - bd)^2 + (ad + bc)^2 = (a^2 + b^2)(c^2 + d^2)$
- (14) $(ac + bd)^2 + (ad - bc)^2 = (a^2 + b^2)(c^2 + d^2)$
- (15) $(ac + bd)^2 - (ad + bc)^2 = (a^2 - b^2)(c^2 - d^2)$
- (16) $(ac - bd)^2 - (ad - bc)^2 = (a^2 - b^2)(c^2 - d^2)$
- (17) $a^2(b - c) + b^2(c - a) + c^2(a - b) = -(b - c)(c - a)(a - b)$
- (18) $(b - c)^3 + (c - a)^3 + (a - b)^3 = 3(b - c)(c - a)(a - b)$
- (19) $a^4 + b^4 + c^4 - 2b^2c^2 - 2c^2a^2 - 2a^2b^2 = (a + b + c)(b - c - a)(c - a - b)(a - b - c)$
- (20) $x^2 + (a + b)x + ab = (x + a)(x + b)$
- (21) $acx^2 + (ad + bc)x + bd = (ax + b)(cx + d)$
- (22) $x^3 + (a + b + c)x^2 + (bc + ca + ab)x + abc = (x + a)(x + b)(x + c)$
- (23) $a^2 - b^2 - c^2 - 2bc = (a + b + c)(a - b - c)$
- (24) $(a + b + c)(bc + ca + ab) - abc = (b + c)(c + a)(a + b)$
- (25) $(a + b + c)^3 - (a^3 + b^3 + c^3) = 3(b + c)(c + a)(a + b)$
- (26) $a^3(b - c) + b^3(c - a) + c^3(a - b) = -(b - c)(c - a)(a - b)(a + b + c)$

Operation:

1

* FACTORIZATION *

$$1: a^2 - b^2 = (a+b)(a-b)$$

↓ ↓ ↓

* FACTORIZATION *

$$4: a^4 + b^4 = (a^2 + \sqrt{2} \cdot ab + b^2)(a^2 - \sqrt{2} \cdot ab + b^2)$$

Use the following keys to recall a specific formula:

- ◀ : Recalls the first formula.
- ▶ : Recalls the 26th (last) formula.
- ↓ : Recalls the following formula.
- ↑ : Recalls the previous formula.

TRIGONOMETRIC FUNCTION

The TRIGONOMETRIC FUNCTION program displays any of the following 49 trigonometric function formulas:

- (1) $\sin^2\theta + \cos^2\theta = 1$
- (2) $1 + \tan^2\theta = \sec^2\theta$
- (3) $1 + \cot^2\theta = \operatorname{cosec}^2\theta$
- (4) $\sin(\alpha \pm \beta) = \sin \alpha \cdot \cos \beta \pm \cos \alpha \cdot \sin \beta$
- (5) $\cos(\alpha \pm \beta) = \cos \alpha \cdot \cos \beta \mp \sin \alpha \cdot \sin \beta$
- (6) $\tan(\alpha \pm \beta) = \frac{\tan \alpha \pm \tan \beta}{1 \mp \tan \alpha \cdot \tan \beta}$
- (7) $\cot(\alpha \pm \beta) = \frac{\cot \alpha \cdot \cot \beta \mp 1}{\cot \beta \pm \cot \alpha}$
- (8) $\sin(\alpha + \beta) \cdot \sin(\alpha - \beta) = \sin^2\alpha - \sin^2\beta$
- (9) $\sin(\alpha + \beta) \cdot \sin(\alpha - \beta) = \cos^2\beta - \cos^2\alpha$
- (10) $\cos(\alpha + \beta) \cdot \cos(\alpha - \beta) = \cos^2\alpha - \sin^2\beta$
- (11) $\cos(\alpha + \beta) \cdot \cos(\alpha - \beta) = \cos^2\beta - \sin^2\alpha$
- (12) $\sin(\alpha \pm \beta) \cdot \cos(\alpha \mp \beta) = \sin \alpha \cdot \cos \alpha \pm \sin \beta \cdot \cos \beta$
- (13) $\frac{\sin(\alpha + \beta)}{\sin(\alpha - \beta)} = \frac{\tan \alpha + \tan \beta}{\tan \alpha - \tan \beta}$
- (14) $\frac{\cos(\alpha + \beta)}{\cos(\alpha - \beta)} = \frac{1 - \tan \alpha \cdot \tan \beta}{1 + \tan \alpha \cdot \tan \beta}$
- (15) $\sin 2\theta = 2 \sin \theta \cdot \cos \theta$
- (16) $\cos 2\theta = \cos^2\theta - \sin^2\theta$
- (17) $\cos 2\theta = 1 - 2 \sin^2\theta$
- (18) $\cos 2\theta = 2 \cos^2\theta - 1$
- (19) $\tan 2\theta = \frac{2 \tan \theta}{1 - \tan^2\theta}$
- (20) $\sin \frac{\theta}{2} = \pm \sqrt{\frac{1 - \cos \theta}{2}}$
- (21) $\cos \frac{\theta}{2} = \pm \sqrt{\frac{1 + \cos \theta}{2}}$
- (22) $\tan \frac{\theta}{2} = \pm \sqrt{\frac{1 - \cos \theta}{1 + \cos \theta}}$
- (23) $\tan \frac{\theta}{2} = \frac{1 - \cos \theta}{\sin \theta}$
- (24) $\tan \frac{\theta}{2} = \frac{\sin \theta}{1 + \cos \theta}$
- (25) $\tan \frac{\theta}{2} = \operatorname{cosec} \theta - \cot \theta$
- (26) $\cot \frac{\theta}{2} = \pm \sqrt{\frac{1 + \cos \theta}{1 - \cos \theta}}$
- (27) $\cot \frac{\theta}{2} = \frac{\sin \theta}{1 - \cos \theta}$
- (28) $\cot \frac{\theta}{2} = \frac{1 + \cos \theta}{\sin \theta}$
- (29) $\cot \frac{\theta}{2} = \operatorname{cosec} \theta + \cot \theta$
- (30) $\sin 3\theta = -4 \sin^3\theta + 3 \sin \theta$
- (31) $\cos 3\theta = 4 \cos^3\theta - 3 \cos \theta$
- (32) $\tan 3\theta = \frac{3 \tan \theta - \tan^3\theta}{1 - 3 \tan^2\theta}$
- (33) $\sin \alpha \cdot \cos \beta = \frac{\sin(\alpha + \beta) + \sin(\alpha - \beta)}{2}$
- (34) $\cos \alpha \cdot \sin \beta = \frac{\sin(\alpha + \beta) - \sin(\alpha - \beta)}{2}$
- (35) $\cos \alpha \cdot \cos \beta = \frac{\cos(\alpha + \beta) + \cos(\alpha - \beta)}{2}$
- (36) $\sin \alpha \cdot \sin \beta = \frac{\cos(\alpha + \beta) - \cos(\alpha - \beta)}{2}$
- (37) $\sin \alpha + \sin \beta = 2 \sin\left(\frac{\alpha + \beta}{2}\right) \cdot \cos\left(\frac{\alpha - \beta}{2}\right)$
- (38) $\sin \alpha - \sin \beta = 2 \cos\left(\frac{\alpha + \beta}{2}\right) \cdot \sin\left(\frac{\alpha - \beta}{2}\right)$
- (39) $\cos \alpha + \cos \beta = 2 \cos\left(\frac{\alpha + \beta}{2}\right) \cdot \cos\left(\frac{\alpha - \beta}{2}\right)$
- (40) $\cos \alpha - \cos \beta = -2 \sin\left(\frac{\alpha + \beta}{2}\right) \cdot \sin\left(\frac{\alpha - \beta}{2}\right)$
- (41) $\tan \alpha \pm \tan \beta = \frac{\sin(\alpha \pm \beta)}{\cos \alpha \cdot \cos \beta}$
- (42) $\cot \alpha \pm \cot \beta = \frac{\sin(\beta \pm \alpha)}{\sin \alpha \cdot \sin \beta}$
- (43) $\cos \alpha \pm \sin \alpha = \sqrt{2} \sin(45^\circ \pm \alpha)$
- (44) $\cos \alpha \pm \sin \alpha = \sqrt{2} \cos(45^\circ \mp \alpha)$
- (45) $\tan(45^\circ \pm \frac{\theta}{2}) = \sec \theta \pm \tan \theta$
- (46) $\tan(45^\circ \pm \frac{\theta}{2}) = \frac{1 \pm \sin \theta}{\cos \theta}$
- (47) $\tan(45^\circ \pm \frac{\theta}{2}) = \cot(45^\circ \mp \frac{\theta}{2})$
- (48) $\tan(45^\circ + \theta) = \frac{1 + \tan \theta}{1 - \tan \theta}$
- (49) $\cot(45^\circ - \theta) = \frac{1 + \cot \theta}{1 - \cot \theta}$

Operation:

2

* TRIGONOMETRY *

$$1: \sin^2 \theta + \cos^2 \theta = 1$$

↓ ↓ ↓

* TRIGONOMETRY *

$$4: \sin(\alpha \pm \beta) \\ = \sin \alpha \cdot \cos \beta \pm \cos \alpha \cdot \sin \beta$$

Use the following keys to recall a specific formula:



: Recalls the first formula.



: Recalls the 49th (last) formula.



: Recalls the following formula.



: Recalls the previous formula.

INTEGRATION

The INTEGRATION program displays any of the following 42 integration formulas:

- | | |
|--|---|
| <p>(1) $\int dx = x + C$</p> <p>(2) $\int x^n dx = \frac{x^{n+1}}{n+1} + C \quad (n+1 \neq 0)$</p> <p>(3) $\int \frac{1}{x} dx = \log x + C$</p> <p>(4) $\int \frac{1}{x \pm a} dx = \log x \pm a + C$</p> <p>(5) $\int e^x dx = e^x + C$</p> <p>(6) $\int e^{ax} dx = \frac{1}{a} e^{ax} + C$</p> <p>(7) $\int a^x dx = \frac{a^x}{\log a} + C \quad (a > 0, a \neq 1)$</p> <p>(8) $\int a^{nx} dx = \frac{a^{nx}}{n \cdot \log a} + C \quad (a > 0, a \neq 1)$</p> <p>(9) $\int \log x dx = x(\log x - 1) + C$</p> <p>(10) $\int x^n \cdot \log x dx = \frac{x^{n+1}}{n+1} \left(\log x - \frac{1}{n+1} \right) + C \quad (n \neq -1)$</p> <p>(11) $\int x e^{ax} dx = \frac{e^{ax}}{a^2} \cdot (ax - 1) + C$</p> <p>(12) $\int \sin x dx = -\cos x + C$</p> <p>(13) $\int \sin ax dx = -\frac{1}{a} \cdot \cos ax + C$</p> <p>(14) $\int \cos x dx = \sin x + C$</p> <p>(15) $\int \cos ax dx = \frac{1}{a} \cdot \sin ax + C$</p> <p>(16) $\int \tan x dx = -\log \cos x + C$</p> <p>(17) $\int \tan ax dx = -\frac{1}{a} \log \cos ax + C$</p> <p>(18) $\int \cot x dx = \log \sin x + C$</p> <p>(19) $\int \cot ax dx = \frac{1}{a} \log \sin ax + C$</p> <p>(20) $\int \sec ax dx = \frac{1}{a} \log \tan \left(\frac{\pi}{4} + \frac{ax}{2} \right) + C$</p> <p>(21) $\int \sec ax dx = \frac{1}{2a} \log \left(\frac{1 + \sin ax}{1 - \sin ax} \right) + C$</p> | <p>(22) $\int \operatorname{cosec} ax dx = \frac{1}{a} \log \tan \frac{ax}{2} + C$</p> <p>(23) $\int \operatorname{cosec} ax dx = -\frac{1}{2a} \log \left(\frac{1 + \cos ax}{1 - \cos ax} \right) + C$</p> <p>(24) $\int \sin^2 x dx = \frac{1}{2} x - \frac{1}{4} \sin 2x + C$</p> <p>(25) $\int \cos^2 x dx = \frac{1}{2} x + \frac{1}{4} \sin 2x + C$</p> <p>(26) $\int \sec^2 ax dx = \frac{1}{a} \cdot \tan ax + C$</p> <p>(27) $\int \operatorname{cosec}^2 ax dx = -\frac{1}{a} \cdot \cot ax + C$</p> <p>(28) $\int \frac{1}{\sin x} dx = \log \tan \frac{x}{2} + C$</p> <p>(29) $\int \frac{1}{\cos x} dx = \log \tan \left(\frac{\pi}{4} + \frac{x}{2} \right) + C$</p> <p>(30) $\int e^{ax} \sin bx dx = \frac{1}{a^2 + b^2} e^{ax} (a \cdot \sin bx - b \cdot \cos bx) + C$</p> <p>(31) $\int e^{ax} \cos bx dx = \frac{1}{a^2 + b^2} e^{ax} (a \cdot \cos bx + b \cdot \sin bx) + C$</p> <p>(32) $\int \sin^{-1} x dx = x \sin^{-1} x + \sqrt{1-x^2} + C$</p> <p>(33) $\int \cos^{-1} x dx = x \cos^{-1} x - \sqrt{1-x^2} + C$</p> <p>(34) $\int \sinh x dx = \cosh x + C$</p> <p>(35) $\int \cosh x dx = \sinh x + C$</p> <p>(36) $\int \tanh x dx = \log \cosh x + C$</p> <p>(37) $\int \frac{1}{\sqrt{a^2 - x^2}} dx = \sin^{-1} \frac{x}{a} + C \quad (x < a)$</p> <p>(38) $\int \frac{1}{a^2 - x^2} dx = \frac{1}{2a} \log \left \frac{a+x}{a-x} \right + C$</p> <p>(39) $\int \frac{1}{a^2 + x^2} dx = \frac{1}{a} \tan^{-1} \frac{x}{a} + C$</p> <p>(40) $\int \frac{1}{\sqrt{x^2 \pm a^2}} dx = \log(x + \sqrt{x^2 \pm a^2}) + C$</p> <p>(41) $\int \sqrt{a^2 - x^2} dx = \frac{1}{2} \cdot (x \sqrt{a^2 - x^2} + a^2 \sin^{-1} \frac{x}{a}) + C$</p> <p>(42) $\int \frac{1}{x^2 - a^2} dx = \frac{1}{2a} \log \left(\frac{x-a}{x+a} \right) + C \quad (x > a)$</p> |
|--|---|

Notes:

- "log" means natural logarithm (\log_e).
- All trigonometric functions are in radians.

Operation:

3

$$\begin{array}{c} * \quad \text{INTEGRATION} \quad * \\ 1: \int dx = x + c \end{array}$$

↓ ↓ ↓

$$\begin{array}{c} * \quad \text{INTEGRATION} \quad * \\ 4: \int 1/(x \pm a) dx \\ = \log |x \pm a| + C \end{array}$$

Use the following keys to recall a specific formula:



: Recalls the first formula.



: Recalls the 42nd (last) formula.



: Recalls the following formula.



: Recalls the previous formula.

GREEK

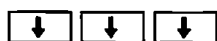
The GREEK program displays the following 24 Greek letters in both capital and small letters, their pronunciation, and corresponding Roman characters:

<i>A</i>	α	Alpha	(a)	<i>N</i>	ν	Nu	(n)
<i>B</i>	β	Beta	(b)	<i>Ξ</i>	ξ	Xi	(x)
<i>Γ</i>	γ	Gamma	(g)	<i>Ο</i>	$ο$	Omicron	(ö)
<i>Δ</i>	δ	Delta	(d)	<i>Π</i>	π	Pi	(p)
<i>E</i>	ϵ	Epsilon	(ě)	<i>P</i>	ρ	Rho	(r)
<i>Z</i>	ζ	Zeta	(z)	<i>Σ</i>	σ	Sigma	(s)
<i>H</i>	η	Eta	(ē)	<i>T</i>	τ	Tau	(t)
<i>Θ</i>	θ	Theta	(th)	<i>Υ</i>	υ	Upsilon	(u)
<i>I</i>	ι	Iota	(i)	<i>Φ</i>	ϕ	Phi	(ph)
<i>K</i>	κ	Kappa	(k)	<i>Χ</i>	χ	Chi	(ch)
<i>Λ</i>	λ	Lambda	(l)	<i>Ψ</i>	ψ	Psi	(ps)
<i>M</i>	μ	Mu	(m)	<i>Ω</i>	ω	Omega	(ö)

Operation:



*	GREEK	*
A	α Alpha	(a)
B	β Beta	(b)
Γ	γ Gamma	(g)



I	ι Iota	(i)
K	κ Kappa	(k)
Λ	λ Lambda	(l)
M	μ Mu	(m)

Use the following keys to recall specific letters:

	: Recalls the first screen.
	: Recalls the last screen.
	: Recalls the following screen.
	: Recalls the previous screen.

Note:

Each operation of either key scrolls the screen 3 lines upward or downward at a time.

PHYSICAL CONSTANT

The PHYSICAL CONSTANT program displays the following 47 physical constants:

Name and Symbol	Value	Unit
(1) Speed of light in vacuum c	2.99792458×10^8	$\text{m} \cdot \text{s}^{-1}$
(2) Gravitational constant G	6.67259×10^{-11}	$\text{N} \cdot \text{m}^2 \cdot \text{kg}^{-2}$
(3) Gravitational acceleration g	9.80665	$\text{m} \cdot \text{s}^{-2}$
(4) Electron rest mass m_e	$9.1093897 \times 10^{-31}$	kg
(5) Proton rest mass m_p	$1.6726231 \times 10^{-27}$	kg
(6) Neutron rest mass m_n	$1.6749286 \times 10^{-27}$	kg
(7) Muon rest mass m_μ	$1.8835327 \times 10^{-28}$	kg
(8) Atomic mass unit u	$1.6605402 \times 10^{-27}$	kg
(9) Electric charge e	$1.60217733 \times 10^{-19}$	C
(10) Planck constant h	$6.6260755 \times 10^{-34}$	$\text{J} \cdot \text{s}$
(11) Boltzmann constant k	1.380658×10^{-23}	$\text{J} \cdot \text{K}^{-1}$
(12) Magnetic permeability μ_0	$12.566370614 \times 10^{-7}$	$\text{H} \cdot \text{m}^{-1}$
(13) Dielectric permittivity ϵ_0	$8.854187817 \times 10^{-12}$	$\text{F} \cdot \text{m}^{-1}$
(14) Electron charge to mass ratio e/m_e	$1.75881962 \times 10^{11}$	$\text{C} \cdot \text{kg}^{-1}$
(15) Classical electron radius $r_e = e^2/4\pi\epsilon_0 m_e c^2$	$2.81794092 \times 10^{-15}$	m
(16) Fine structure constant $\alpha = e^2/4\pi\epsilon_0 \hbar c$	$7.29735308 \times 10^{-3}$	
(17) Quantum of circulation $h/2m_e$	$3.63694807 \times 10^{-4}$	$\text{J} \cdot \text{s} \cdot \text{kg}^{-1}$
(18) Bohr radius $a_0 = 4\pi\epsilon_0 \hbar^2/m_e e^2$	$5.29177249 \times 10^{-11}$	m
(19) Rydberg constant $R_\infty = e^2/16\pi^2 \epsilon_0 \alpha \hbar c$	1.0973731534×10^7	m^{-1}
(20) Flux quantum $h/2e$	$2.06783461 \times 10^{-15}$	Wb
(21) Bohr magneton $\mu_B = e\hbar/2m_e$	$9.2740154 \times 10^{-24}$	$\text{J} \cdot \text{T}^{-1}$
(22) Electron magnetic moment μ_e	$9.2847701 \times 10^{-24}$	$\text{J} \cdot \text{T}^{-1}$
(23) Free electron g-factor $2\mu_e/\mu_B$	2.002319304386	
(24) Nuclear magneton $\mu_N = e\hbar/2m_p$	$5.0507866 \times 10^{-27}$	$\text{J} \cdot \text{T}^{-1}$
(25) Proton magnetic moment μ_p	$1.41060761 \times 10^{-26}$	$\text{J} \cdot \text{T}^{-1}$
(26) Proton g-factor $2\mu_p/\mu_N$	5.585694772	
(27) Gyromagnetic ratio of proton γ_p	2.67522128×10^8	$\text{s}^{-1} \cdot \text{T}^{-1}$
(28) Neutron magnetic moment μ_n	$9.6623707 \times 10^{-27}$	$\text{J} \cdot \text{T}^{-1}$
(29) Muon magnetic moment μ_μ	$4.4904514 \times 10^{-26}$	$\text{J} \cdot \text{T}^{-1}$
(30) Compton wavelength of the electron $\lambda_c = h/m_e c$	$2.42631058 \times 10^{-12}$	m

Name and Symbol	Value	Unit
(31) Compton wavelength of the proton $\lambda_{cp} = h/m_p c$	$1.32141002 \times 10^{-15}$	m
(32) Stefan-Boltzmann constant $\sigma = \pi^2 k^4 / 60 \hbar^3 c^2$	5.67051×10^{-8}	$\text{W} \cdot \text{m}^{-2} \cdot \text{K}^{-4}$
(33) Avogadro's constant N_A	6.0221367×10^{23}	mol^{-1}
(34) Ideal gas at STP V_0	2.241410×10^{-2}	$\text{m}^3 \cdot \text{mol}^{-1}$
(35) Gas constant $R = N_A k$	8.314510	$\text{J} \cdot \text{mol}^{-1} \cdot \text{K}^{-1}$
(36) Faraday constant $F = N_A e$	9.6485309×10^4	$\text{C} \cdot \text{mol}^{-1}$
(37) Josephson frequency-voltage ratio $2e/h$	4.8359767×10^{14}	$\text{Hz} \cdot \text{V}^{-1}$
(38) Quantum hole resistance R_H	25812.8056	Ω
(39) Electron volt eV	$1.60217733 \times 10^{-19}$	J
(40) Astronomical unit AU	$1.49597870 \times 10^{11}$	m
(41) Parsec pc	3.0856776×10^{16}	m
(42) Sea mile sea mile	1852	m
(43) Angstrom \AA	1×10^{-10}	m
(44) Knot knot	1852/3600	$\text{m} \cdot \text{s}^{-1}$
(45) Torr Torr	101325/760	Pa
(46) Standard atmospheric pressure atm	101325	Pa
(47) Calorie cal	4.1868	J

Operation:

↓ 5

* PHYSICAL CONSTANT *
1: c = 2.99792458E+8
[m · s⁻¹]

↓

* PHYSICAL CONSTANT *
2: G = 6.67259E-11
[N · m² · kg⁻²]

Use the following keys to recall a specific constant:

◀

: Recalls the first physical constant.

▶

: Recalls the last physical constant.

↓

: Recalls the following physical constant.

↑

: Recalls the previous physical constant.

METRIC CONVERSION

The METRIC CONVERSION program converts the units of length, area, volume, weight and energy.

		m	in(inch)	ft(foot)	yd(yard)	
		Units of length I		m	1	39.3701
		in	0.0254	1	0.0833333	0.0277778
		ft	0.3048	12	1	0.333333
		yd	0.9144	36	3	1
		mile	1609.344	63360	5280	1760
		cm	0.01	0.393701	0.0328084	0.0109361
		Å	1×10^{-10}	3.93701×10^{-9}	3.28084×10^{-10}	1.09361×10^{-10}
		pc	3.0856776×10^{16}	$1.214834357 \times 10^{18}$	$1.01236145 \times 10^{17}$	$3.37452788 \times 10^{16}$

		mile	cm	Å	pc(parsec)	
		Units of length II		m	0.000621371	100
		in	1.57828×10^{-5}	2.54	254000000	$8.231579346 \times 10^{19}$
		ft	0.000189394	30.48	3048000000	$9.877895215 \times 10^{18}$
		yd	0.000568182	91.44	9144000000	$2.963368564 \times 10^{17}$
		mile	1	160934.4	1.609344×10^{13}	$5.215528674 \times 10^{14}$
		cm	6.21371×10^{-6}	1	100000000	$3.24077927 \times 10^{19}$
		Å	6.21371×10^{14}	0.00000001	1	$3.24077927 \times 10^{-27}$
		pc	$1.917350576 \times 10^{13}$	3.0856776×10^{18}	3.0856776×10^{26}	1

Reading the table:

To convert from "meter" to "inch", multiply by 39.3701.

Other conversions can be performed in a manner similar to the above operation.

		m ²	a(are)	acre	mile ²	
		Units of area		m ²	1	0.01
		a	100	1	0.0247105	3.86102×10^{-5}
		acre	4046.86	40.4686	1	0.0015625
		mile ²	2589990	25899.9	640	1

		cm ³	m ³	in ³ (inch)	ℓ (litre)	ft ³ (foot)	
		Units of volume		cm ³	1	0.000001	0.0610237
		m ³	1000000	1	61023.7	1000	35.3147
		in ³	16.3871	1.63871×10^{-5}	1	0.0163871	0.000578704
		ℓ	1000	0.001	61.0237	1	0.0353147
		ft ³	28316.8	0.0283168	1728	28.3168	1

Units of weight		g	kg	oz (ounce)	lb (pound)
	g	1	0.001	0.035274	0.00220462
	kg	1000	1	35.274	2.20462
	oz	28.3495	0.0283495	1	0.0625
	lb	453.59237	0.45359237	16	1

Units of energy I		eV	erg	cm ⁻¹	Hz
	eV	1	1.60218 × 10 ⁻¹²	8065.54	2.41799 × 10 ¹⁴
	erg	6.24151 × 10 ¹¹	1	5.03411 × 10 ¹⁵	1.50919 × 10 ²⁶
	cm ⁻¹	0.000123984	1.98645 × 10 ⁻¹⁶	1	2.99792 × 10 ¹⁰
	Hz	4.13567 × 10 ⁻¹⁵	6.62608 × 10 ⁻²⁷	3.33564 × 10 ⁻¹¹	1
	K	8.61738 × 10 ⁻⁵	1.38066 × 10 ⁻¹⁶	0.695039	2.08367 × 10 ¹⁰
	G	5.78838 × 10 ⁻⁹	9.27402 × 10 ⁻²¹	4.66864 × 10 ⁻⁵	1399620
	J/mol	1.03643 × 10 ⁻⁵	1.66054 × 10 ⁻¹⁷	0.0835934	2506070000
kcal/mol	0.0433854	6.9511 × 10 ⁻¹⁴	349.926	1.04905 × 10 ¹³	

Units of energy II		K	G	J/mol	kcal/mol
	eV	11604.5	172760000	96485.3	23.0492
	erg	7.24292 × 10 ¹⁵	1.07828 × 10 ²⁰	6.02214 × 10 ¹⁶	1.43862 × 10 ¹³
	cm ⁻¹	1.43877	21419.5	11.9627	0.00285774
	Hz	4.79922 × 10 ⁻¹¹	7.14478 × 10 ⁻⁷	3.99031 × 10 ⁻¹⁰	9.53241 × 10 ⁻¹⁴
	K	1	14887.4	8.31451	0.00198624
	G	0.000067171	1	0.000558494	1.33418 × 10 ⁻⁷
	J/mol	0.120272	1790.53	1	0.000238889
kcal/mol	503.463	7495250	4186.05	1	

Operation:

↓ 6

```

* METRIC CONVERSION *
1 - LENGTH      4 - WEIGHT
2 - AREA        5 - ENERGY
3 - VOLUME

```

(Menu)

Choose the item for metric conversion with the 1 – 5 number key.

Example 1:

Convert 12 meters to inches:

1

```

* METRIC CONVERSION *
<L>      X = ?

```

Enter the data to convert:

12

```
* METRIC CONVERSION *
<L>      X= 12
from?
<  1:m, 2:in, 3:ft, 4:y d  >
```

Enter the original unit for the data:

1 (m)

```
* METRIC CONVERSION *
<L>      X= 12
from m to?
<  1:m, 2:in, 3:ft, 4:y d  >
```

Enter the target unit:

2 (in)

```
* METRIC CONVERSION *
<L>      X= 12
m          12
in         472.4412
```

The answer is 472.4412 inches.

Example 2:

Convert 10^{-3} inch to angstroms (Å):

When the last result of metric conversion for length is still on the display, press the key, and you can continue metric conversion for length.

```
* METRIC CONVERSION *
<L>      X= ?
```

1 SHIFT + Exp -3 2 (in)

```
* METRIC CONVERSION *
<L>      X= 0.001
from in to?
<  1:m, 2:in, 3:ft, 4:y d  >
```

Since angstrom (Å) is not in the menu on the display, first recall it to the display using the (or) key, then choose angstroms using

7 (in)

```
* METRIC CONVERSION *
<L>      X= 0.001
in        0.001
Å         254000
```

The answer is 254000 Å.

Notes:

- If the result of metric conversion exceeds the range of $10^{-100} < X < 10^{100}$, "Answer not found" will appear. Press the key, then enter appropriate data for conversion.
- If the key is pressed when the prompt "X = ?" is on the display, the computer will return to the previous display.

COMPLEX NUMBER

The COMPLEX NUMBER program performs complex number calculations using the following key operations:

Key	Function
<input type="button" value="X"/>	Stores the value into X (in the order of the real and imaginary parts).
<input type="button" value="Y"/>	Stores the value into Y (in the order of the real and imaginary parts).
<input type="button" value="+"/>	$X + Y \rightarrow X$: Performs addition and stores the result into X.
<input type="button" value="-"/>	$X - Y \rightarrow X$: Performs subtraction and stores the result into X.
<input type="button" value="X*"/>	$X * Y \rightarrow X$: Performs multiplication and stores the result into X.
<input type="button" value="÷"/>	$X / Y \rightarrow X$: Performs division and stores the result into X.
<input type="button" value="M+"/>	$X + M \rightarrow M$: Adds the value of X to the memory.
<input type="button" value="RM"/>	$M \rightarrow X$: Recalls the value in memory and assigns it to X.
<input type="button" value="X→M"/>	$X \rightarrow M$: Stores the value of X into the memory.
<input type="button" value="▶"/> or <input type="button" value="◀"/>	$X \leftrightarrow Y$: Exchanges the values of X and Y.
<input type="button" value="√"/>	$\sqrt{X} \rightarrow X$: Calculates the square root of X and stores the result into X.
<input type="button" value="1/X"/>	$1/X \rightarrow X$: Calculates the reciprocal of X and stores the result into X.
<input type="button" value="X²"/>	$X * X \rightarrow X$: Calculates the square of X and stores the result into X.
<input type="button" value="xy"/> <input type="button" value="("/>	<div style="display: flex; align-items: center;"> <div style="margin-right: 10px;"> Absolute value of X $\rightarrow X$ Argument of X $\rightarrow Y$ </div> <div style="font-size: 2em; margin-right: 10px;">)</div> <div> Stores the absolute value of X ($\sqrt{X_R^2 + X_I^2}$) into X and the argument of X ($\tan^{-1} \frac{X_I}{X_R}$) into Y where X_R is the real part of X and X_I the imaginary part of X. </div> </div>

Operation:

```

*   COMPLEX NUMBER   *
X =  0
<< X, Y, + - * / , M + , RM , x → M >>
    
```


Example:

For $X = 3 + 4i$ and $Y = 6 + 9i$, add Y to X, then square the result:

(X is selected from the menu)

```

*   COMPLEX NUMBER   *
X (real) = 0. ?
<< X, Y, + - * / , M + , RM , x → M >>
    
```


3  4

```
* COMPLEX NUMBER *  
X(real) =3  
X(image)=0. 4  
<< X,Y,+-* / ,M+,RM,x→M >>
```



```
* COMPLEX NUMBER *  
X= 3  
+4i  
<< X,Y,+-* / ,M+,RM,x→M >>
```

 (Y is selected from the menu)

```
* COMPLEX NUMBER *  
Y(real) =0. ?  
<< X,Y,+-* / ,M+,RM,x→M >>
```




6  9 

```
* COMPLEX NUMBER *  
Y= 6  
+9i  
<< X,Y,+-* / ,M+,RM,x→M >>
```



```
* COMPLEX NUMBER *  
Y= 6  
+9i  
<< X,Y,+-* / ,M+,RM,x→M >>
```

You get $(3 + 4i) + (6 + 9i) = 9 + 13i$.
To display the result, press:

```
* COMPLEX NUMBER *  
X= 9  
+13i  
<< X,Y,+-* / ,M+,RM,x→M >>
```

Now square the sum:



```
* COMPLEX NUMBER *  
X=-88  
+234i  
<< X,Y,+-* / ,M+,RM,x→M >>
```

The final answer is: $(9 + 13i)^2 = -88 + 234i$.

Note:

Pressing the  or  key changes the display contents on the 4th line.

6. STAT MODE

The computer can perform statistical and regression calculations on one or two variables. With statistical calculations, you can obtain mean values, standard deviations, and other statistics from sample data. Regression calculation determines the coefficients of linear regression formulas or estimated values from sample data.

Selecting and Cancelling the STAT Mode

Press the **SHIFT** + **STAT** keys to display the STAT menu.

```
* Statistical analysis *
 1:Single-variable stat
 2:Two-variable stat
Select No.?
```

Press the **1** key to select single-variable statistical calculations or the **2** key to select two-variable statistical calculations.

Select any other mode to exit the STAT mode.

Single-Variable Statistical Calculations

The following statistics can be obtained from single-variable statistical calculations.

- n : Sample size of x
- $\sum x$: Sum of samples x
- $\sum x^2$: Sum of squares of samples x
- \bar{x} : Sample mean x
- s : Sample standard deviation with population parameter taken to be $n-1$.

$$s = \sqrt{\frac{\sum x^2 - n\bar{x}^2}{n-1}}$$

This equation is used to estimate the standard deviation of a population from sample data (x) extracted from that population.

- σ : Population standard deviation with population parameter taken to be n .

$$\sigma = \sqrt{\frac{\sum x^2 - n\bar{x}^2}{n}}$$

This equation lets you assume the entire population as sample data (x) or determine the standard deviation of the sample data which is taken to be a population.

Selecting single-variable statistical calculations

After displaying the STAT menu, press the **1** key to select single-variable statistical calculations. The single-variable submenu will be displayed.

SHIFT + **STAT** **1**

```
* Functions *      (x)
 1:Input           2:Delete
 3:Analysis       4:Printer
Select No.?
```

The following can be selected from the submenu:

- | | |
|------------------------|---|
| 1 ... Input: | Used for entering data. |
| 2 ... Delete: | Used for deleting the entered data if the data was incorrect or to start a new calculation. |
| 3 ... Analysis: | Used for displaying the obtained statistics. |
| 4 ... Printer: | Used for printing the obtained statistics. Available only if the optional printer is connected to the computer. |

Press the **BREAK** key to display the STAT menu.

Entering data

Press the **1** key to display the data input prompt. Enter the data.

1

```
* Data input *
1:x= _
↑         ↑
Data to be entered.
Indicates the number of data entered.
```

To enter a single data value, press: data **←**.

To enter multiple identical data values simultaneously, press: data, frequency **←**.

To enter negative values, press: **-** data **←**.

If you make a typing error during data entry, press **C-CE**, then type the correct data.

Press the **BREAK** key to end data entry and display the single-variable submenu.

Note:

In statistics, "frequency" is used to define the number of identical data. For example, if three identical data occur consecutively, frequency 3 is used.

Deleting data

This function can be used for correcting the entered data. Press the following keys in the single-variable submenu to display the data clear prompt.

2

```
* Delete *
 1:Data
 2:All clear
Select No.?
```

1

```
* Delete a data *  
x= _
```

Enter the values of the incorrect data or data to be deleted in the same manner as for data entry. Multiple data values can be deleted using commas (,). After deletion, correct data may be entered from the data input prompt.

Obtaining statistics

Press the **3** key in the single-variable submenu to display the analysis submenu.

3

```
* Analysis *      (x)  
1:n      2:Σx      3:Σx2  4: x̄  
5:s      6:σ  
Select No.?
```

The following statistics can be obtained by pressing keys **1** to **6**:

1 n :	The sample size of data
2 Σx :	Sum of samples
3 Σx^2 :	Sum of squares of samples
4 \bar{x} :	Sample mean
5 s :	Sample standard deviation
6 σ :	Popular standard deviation

Press the **BREAK** key to return to the single-variable submenu.

Starting a new calculation (clearing the previous data)

Perform one of the following:

- Exit the STAT mode and select the STAT mode again. The previous data will be cleared.
- Delete data using the delete/clear function. Press the **2** key in the single-variable submenu to display the delete/clear submenu.

2

```
* Delete *  
1:Data  
2:All clear  
Select No.?
```

The delete/clear submenu will be displayed.

2

```
* All clear *  
1:YES  
2:NO  
Select No.?
```

The all clear submenu will be displayed.

Press the **1** key to clear the previous data or the **2** key to retain the previous data.

Example:

The test scores for 35 randomly selected students are as shown. Determine the mean and standard deviation of these scores.

No.	Score	Frequency	No.	Score	Frequency
1	30	1	5	70	8
2	40	1	6	80	9
3	50	4	7	90	5
4	60	5	8	100	2

Select "1: Single-variable stat" in the STAT menu.

```
* Functions *      (x)
  1:Input          2:Delete
  3:Analysis      4:Printer
Select No.?
```

Select "1: Input" and enter the data.

30 40 50,4
 60,5 70,8 80,9
 90,5 100,2

```
20:x=80.,9.
29:x=90.,5.
34:x=100.,2.
36:x=_
```

Data entry is now complete.

Display the single-variable submenu.

BREAK

Select "3: Analysis".

```
* Analysis *      (x)
  1:n      2:Σx    3:Σx2 4: x̄
  5:s      6:σ
Select No.?
```

Obtain the mean.

```
* Analysis *      (x)
  1:n      2:Σx    3:Σx2 4: x̄
  5:s      6:σ
x̄= 71.42857143
```

Obtain the population standard deviation.

```
* Analysis *      (x)
  1:n      2:Σx    3:Σx2 4: x̄
  5:s      6:σ
σ= 16.23802542
```

Return to the single-variable submenu.

BREAK

- Press , , , or key to obtain the sample size, sum, sum of squares, or standard deviation of samples.
After obtaining intermediate statistics, such as mean and standard deviation, further data can be entered by selecting "1: Input" in the single-variable submenu.

Printing statistics

The calculated statistics can be printed on the optional CE-126P printer. Connect the printer to the computer and turn the power on. Enter the data and select "4: Printer" in the single-variable submenu to print the statistics.

```
* Functions *      (x)
  1:Input         2:Delete
  3:Analysis     4:Printer
Select No.?
```

Printing example:

```
n=          35.
Σx=         2500.
Σx²=        187800.
MEAN(x)=71.42857143
s=          16.47508942
σ=          16.23802542
```

After printing, the display will return to the single-variable submenu.

Two-Variable Statistical Calculations

Operations for two-variable statistical calculations are similar to operations for single-variable statistical calculations. Read the section for the single-variable statistical calculations first.

The following statistics can be determined from two-variable statistical calculations.

n , $\sum x$, $\sum x^2$, and \bar{x} : Same as for single-variable calculations

s_x and σ_x : Same as s and σ .

$\sum y$: Sum of samples y

$\sum y^2$: Sum of squares of samples y

$\sum xy$: Sum of products of samples x and y

\bar{y} : Sample mean y

s_y : Sample standard deviation with population parameter taken to be $n-1$.

$$s_y = \sqrt{\frac{\sum y^2 - n\bar{y}^2}{n-1}}$$

σ_y : Population standard deviation of samples (y) with population parameter taken to be n .

$$\sigma_y = \sqrt{\frac{\sum y^2 - n\bar{y}^2}{n}}$$

- a : $a = \bar{y} - b\bar{x}$
Coefficient for linear regression $y = a + bx$
- b : $b = \frac{S_{xy}}{S_{xx}}$
Coefficient for linear regression $y = a + bx$
- r : $r = \frac{S_{xy}}{\sqrt{S_{xx} \cdot S_{yy}}}$
Correlation coefficient
- x' : $x' = \frac{y - a}{b}$
Estimated value (x estimated from y)
- y' : $y' = a + bx$
Estimated value (y estimated from x)

Note:

$$S_{xx} = \sum x^2 - \frac{(\sum x)^2}{n}$$

$$S_{yy} = \sum y^2 - \frac{(\sum y)^2}{n}$$

$$S_{xy} = \sum xy - \frac{\sum x \cdot \sum y}{n}$$

Selecting two-variable statistical calculations

After displaying the STAT menu, press the $\boxed{2}$ key to select the two-variable statistical calculations.

Entering data

Press the $\boxed{1}$ key at the two-variable submenu to display the data input prompt.

Enter the x and y data as shown in the display.

To enter a single pair of data values, press: data x $\boxed{\leftarrow}$ data y $\boxed{\leftarrow}$.

To enter multiple identical pairs of data values simultaneously, press: data x $\boxed{\leftarrow}$ data y , frequency $\boxed{\leftarrow}$.

To enter a negative value, press the $\boxed{-}$ key before the value.

Press the $\boxed{\text{BREAK}}$ key to end data entry and display the two-variable submenu.

Obtaining statistics

Press the $\boxed{3}$ key on the two-variable submenu to display the analysis submenu.

There are two submenus, pressing the $\boxed{\uparrow}$ or $\boxed{\downarrow}$ key toggles the submenus.

(The symbol \uparrow or \downarrow will appear.)

When the $\boxed{3}$ key is pressed in the two-variable submenu,

(The first analysis submenu)

* Analysis *		(x, y) ↓	
1: n	2: $\sum x$	3: $\sum x^2$	4: \bar{x}
5: s_x	6: σ_x	7: $\sum y$	8: $\sum y^2$
Select No. ?			

↓

(The second analysis submenu)

```

* Analysis *      (x,y) ↑
1:Σxy 2:ȳ      3:sy 4:σy
5:a 6:b 7:r 8:x' 9:y'
Select No.?
```

Pressing the ↑ key will display the first analysis submenu.

Example:

The following table lists the dates (in April) on which migratory birds fly through a certain district, versus the average temperatures in March of the same district. From this table determine the coefficients, a and b , of the linear regression line, $y = a + bx$, and correlation coefficient r . Estimate the date of migration when the mean temperature in March is 9.1°C . Also estimate the mean temperature in March if the date of migration is April 10.

Year	1	2	3	4	5	6	7	8
Mean temp. ($x^{\circ}\text{C}$)	6.2	7.0	6.8	8.7	7.9	6.5	6.1	8.2
Date of migration (y day)	13	9	11	5	7	12	15	7

Select "2: Two-variable stat" in the STAT menu.

2

```

* Functions *      (x,y)
1:Input      2:Delete
3:Analysis   4:Printer
Select No.?
```

Select "1: Input" and enter the data.

```

1
6.2 ← 13 ← 7.0 ←
9 ← 6.8 ← 11 ←
8.7 ← 5 ← 7.9 ←
7 ← 6.5 ← 12 ←
6.1 ← 15 ← 8.2 ←
7 ←
```

```

y=15.
8:x=8.2
y=7.
9:x=_
```

Data entry is now complete. Display the two-variable submenu.

BREAK

Select "3: Analysis" and display the second analysis submenu.

3 ↓

Determine coefficient a .

5

```

* Analysis *      (x,y) ↑
1:Σxy 2:ȳ      3:sy 4:σy
5:a 6:b 7:r 8:x' 9:y'
a= 34.44951017
```


Determine coefficient b .

i

```
* Analysis *      (x,y) ↑
1:Σxy 2:ȳ      3:sy  4:σy
5:a 6:b 7:r 8:x'  9:y'
b=                -3.425018839
```

Estimate the date of migration:

3

```
* Analysis *      (x,y)
x=_
```

Enter the mean temperature.

9.1

```
* Analysis *      (x,y)
x=9.1
y=                3.281838734
x=_
```

(Estimated date: April 3)

Display the second analysis submenu.

BREAK

Estimate the mean temperature.

8

```
* Analysis *      (x,y)
y=_
```

Enter the date of migration.

10

```
* Analysis *      (x,y)
y=10.
x=                7.13850385
y=_
```

(Estimated mean temperature on March 10: approx. 7.1°C)

Note:

The statistical or regression results are automatically stored into the fixed variables U to Z.

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

- The variable contents not calculated will become 0 (zero).
- These statistics are cleared when the statistical calculation mode is set again.

7. RUN MODE

The RUN mode is a very versatile operation mode, and has the ability to run BASIC programs written in the PRO mode.

Selecting RUN Mode

Select the BASIC mode by pressing the **BASIC** key. If PRO is displayed, press the **BASIC** key to select the RUN mode. The prompt (>) tells you that the computer is awaiting entry. The display should now look like this:

```
      CAPS          DEG
R U N  M O D E
>
      RUN
```

Some Helpful Hints

If you make an error during entry and get an error message, the simplest way to clear the error is to press the **C-CE** key and reenter. If the computer "hangs up" (you cannot get it to respond at all), press the RESET button while holding the **ON** key (see Appendix E).

The prompt (>) tells you that the computer is awaiting entry. As you enter data the prompt disappears and the cursor () moves to the right, indicating the next available location in the display.

Pressing the right **▶**, left **◀**, up **↑** and down **↓** keys moves the cursor.

The display of the computer consists of 4 lines (24 characters per line). Key entries and calculated results are displayed from the top line of the display. If the characters to be displayed exceed 4 lines, the displayed contents will be moved up by 1 line (the first line will move off the top of the display).

Press the **←** key to tell the computer that you have finished entering data and to signal the computer to perform the indicated operations. **You must press the ← key at the end of each line of entry or your calculations will not be acted upon by the computer.**

When performing numeric calculations, entries appear on the left of the display; the results appear on the right.

```
- 1 2 3 4 5 6 7 8 9 1 / 1 0
                        - 1 2 3 4 5 6 7 8 9 . 1
```

Do not use dollar signs or commas when entering calculations. These characters have special meanings in the BASIC programming language.

When using the **SHIFT** key to implement another key's second function, press and hold the **SHIFT** key and then press the other key. The **2nd F** key may also be used, as in the CAL mode.

When the **SHIFT** key is used, the character actually produced is represented in the following keystroke. For example pressing **SHIFT** + **Y** will produce the "&" character. This is written **SHIFT** + **&**.

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

Note:

For details regarding the Decimal Place and Modify functions, see the explanation on pages 31, 32 in the chapter on the CAL MODE.

Simple Calculations

The computer performs calculations in the RUN mode with 10-digit precision. Turn the power on and set it to the RUN mode. Now try these simple examples.

Example:

$2 + 3 \times 4 =$

2 **+** 3 **X*** 4 **←**

2 + 3 * 4	14 .
-----------	------

Example:

$5 \times (-6) + 7 =$

5 **X*** **-** 6 **+** 7 **←**

2 + 3 * 4	14 .
5 * - 6 + 7	- 23 .

Compound Calculations and Parentheses

You can combine several operations into one step as in the CAL mode. For example, you can enter:





675 + 6750/45000

Compound calculations, however, must be entered very carefully to avoid ambiguity.



When performing compound calculations, the computer has specific rules of expression evaluation and operator priority (see page 70). Use parentheses to clarify your expressions:

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

Recalling Entries

Even after the computer has displayed the results of your calculation, you can verify that the entry was made correctly, and edit it if necessary. To edit, use the left arrow  and right arrow  keys. Remember that the left and right arrows are also used to position the cursor. Use the left arrow  key to position the cursor after the last character. Use the right arrow  key to position the cursor over the first character.

Example:

300  6 



300 / 6	50 .
---------	------

Change this operation to 300/5.

Recall your last entry using the  key.



300 / 6	50 .
300 / 6 _	

Because you recalled the expression using , the cursor is positioned at the end of the display. Use  to move the cursor one space to the left.



300 / 6	50 .
300 / <u>6</u>	

Notice that after you move the cursor, it becomes a flashing block. Whenever you position the cursor over an existing character, it will flash.

Enter a 5 to replace the 6. An important point in replacing characters is that once you enter a new character over an existing character, the original is gone forever! You cannot recall an expression that has been erased.

5 



300 / 6	50 .
300 / 5	60 .

You can also insert or delete characters in an entry. Change the previous calculation to 3000/5.

Recall your entry using the  key.



300 / 5	50 .
300 / 5	60 .
3000 / 5	

Because you recalled using the  key, the flashing cursor is now over the first character. To make the correction you must insert a zero. Using the  key, move the cursor until it is over a zero. When making an INSert, position the flashing cursor over the character before which you wish to make the insertion.



 


3 0 0 / 5	5 0 .
3 0 0 0 / 5	6 0 .

Pressing INSert moves all the characters one space to the right, and inserts an 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.



3 0 0 / 5	6 0 .
3 0 0 0 / 5	6 0 0 .

To DElete a character, use the  key. Change the previous calculation to 3/5. Recall your entry using .

To correct this entry, eliminate the zeros. Using , move the cursor to the first zero. To delete a character, always position the cursor over the character to be deleted.


 

3 0 0 0 / 5	6 0 .
3 0 0 0 / 5	6 0 0 .

Now use the DElete key to delete the zeros.


3 0 0 0 / 5	6 0 .
3 / 5	6 0 0 .

Pressing the  key deletes the character under the cursor and shifts all the following characters one space to the left. Since you have no other changes to make, complete the calculation by displaying the result.



3 0 0 0 / 5	6 0 0 .
3 / 5	0 . 6

Note:

Pressing the  key when the cursor is positioned over a character erases the character, leaving a blank space. DElete eliminates the character and the space it occupied.

You can also use the **BS** key to delete errors. Note that pressing the **BS** key moves the cursor back one position and deletes the character there, while pressing the **DEL** key deletes the character the cursor is positioned over.

Errors

Recalling your last entry is essential if you get an error message. Let us imagine that, unintentionally, you typed this into the computer:

C•CE 3 0 0 **÷/** **÷/** 5 **←**

```
3 0 0 // 5
ERROR 1 0
```

“ERROR 10” is the computer’s way of saying, “I don’t know what you want me to do here.” Press the **◀** or **▶** key to move the flashing cursor to where the error occurred.

◀ (or **▶**)

```
3 0 0 // 5
ERROR 1 0
3 0 0 / 5
```

Use the **DEL** key to correct this error.

DEL **←**

```
3 0 0 // 5
ERROR 1 0
3 0 0 / 5
6 0 .
```

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

When using the computer as a calculator, the majority of errors you encounter will be syntax errors. For a complete listing of error messages, see Appendix B.

Serial Calculations

The computer allows you to use the results of one calculation as part of the following calculation.

Example:

What is 15% of 300 * 150?

C•CE 3 0 0 **X*** 1 5 0 **←**

```
3 0 0 * 1 5 0
4 5 0 0 0 .
```

In serial calculations it is not necessary to retype your previous results, but **DO NOT** press the **C•CE** key between entries.

$\boxed{\times}$. 1 5 $\boxed{\leftarrow}$

3 0 0 * 1 5 0	4 5 0 0 0 .
4 5 0 0 0 . * . 1 5	6 7 5 0 .

Notice that as you type in the second calculation ($\times .15$), 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 the $\boxed{\leftarrow}$ key.

Note:

The $\boxed{\%}$ and $\boxed{\Delta\%}$ keys cannot be used in percent calculations in the RUN mode. The $\boxed{\%}$ key should be used as a character only, and the $\boxed{\Delta\%}$ key is inoperative.

For example, 4 5 0 0 0 $\boxed{\times}$ 15 $\boxed{\text{SHIFT}}$ + $\boxed{\%}$ $\boxed{\leftarrow}$ → ERROR 10

To change the sign of the previous result, multiply by -1:

$\boxed{\times}$ $\boxed{-}$ 1 $\boxed{\leftarrow}$

4 5 0 0 0 . * . 1 5	6 7 5 0 .
6 7 5 0 . * - 1	- 6 7 5 0 .

Pressing the $\boxed{\text{SHIFT}}$ + $\boxed{-}$ keys or the $\boxed{+/-}$ key will also reverse the sign.

Constant Calculations

The $\boxed{\text{CONST}}$ key lets you use any constant in arithmetic calculations, as described below.

How To Use Constants

Addition: $\boxed{+}$ a $\boxed{\text{CONST}}$ or a $\boxed{+}$ $\boxed{\text{CONST}}$

Subtraction: $\boxed{-}$ a $\boxed{\text{CONST}}$ or a $\boxed{-}$ $\boxed{\text{CONST}}$

Multiplication: $\boxed{\times}$ a $\boxed{\text{CONST}}$ or a $\boxed{\times}$ $\boxed{\text{CONST}}$

Division: $\boxed{\div}$ a $\boxed{\text{CONST}}$ or a $\boxed{\div}$ $\boxed{\text{CONST}}$

where "a" denotes a constant.

Once you use the $\boxed{\text{CONST}}$ key, the "CONST" appears on the upper right of the display.

Note:

When you are not using the Constant Calculation function, make sure that the "CONST" indicator is not on.

Checking the Constant Setting

To check the constant setting you last entered, press the following keys when the "CONST" indicator is on:

$\boxed{2\text{nd F}}$ $\boxed{\text{CONST}}$ ($\boxed{\text{SHIFT}}$ + $\boxed{\text{CONST}}$)

Clearing the Last Constant

To clear the constant setting you last entered, press the following keys. It is also cleared when the computer is turned off.

2nd F **CA** (**SHIFT** + **CA**)

Example:

Store "+ (4.8 + 3.6)" as a constant and calculate "24 - 18.5 + (4.8 + 3.6)" and "8.2 × 6 + (4.8 + 3.6)"

Enter: **+** 4.8 **+** 3.6 **CONST**

The constant need not be enclosed in parentheses.

Enter: 24 **-** 18.5 **←** Answer: 13.9

Enter: 8.2 **X*** 6 **←** Answer: 57.6

Using Variables in Calculations

The computer 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 8. Variables are designated with an Assignment Statement:

A = 5 **←**

B = -2 **←**

Note:

To enter the "=" sign, press the **SHIFT** + **L** keys.

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

C = A + 3 **←**

D = C **←**

As you press **←**, the computer performs the calculation and displays the new value of the variable. You can display the current value of any variable by entering the alphabetic character it is stored under:

C•CE C **←**

C	8 .
---	-----

Variables will retain their assigned values even if the computer is switched OFF or undergoes an Auto OFF. Variables are lost only when:

- You assign a new value to the same variable.
- You enter CLEAR **←** (not the **C•CE** key).
- You clear the computer using the RESET button.

There are certain limitations on the assignment of variables, and certain programming procedures that cause them to be changed. See Chapter 8 for a discussion of assignment and the use of variables in programming.

Last Answer Feature

In a simple calculation, the result of the previous calculation can only be used in continuous calculations as the first number.

Example:

3 $\boxed{+}$ 4 $\boxed{\leftarrow}$
 $\boxed{\times}$ 5 $\boxed{\leftarrow}$

3 + 4	7 .
7 . * 5	35 .

However, the computer has a feature that lets you recall the result of the previous calculation and use it in any location in the current calculation. This is called the last answer feature. It allows the previous answer to be recalled any number of times by pressing the $\boxed{\text{ANS}}$ key. If you entered the last example, press $\boxed{\text{C}\cdot\text{CE}}$ then $\boxed{\text{ANS}}$ and you will see "35." displayed.

Let's look at an example where a previous result is used twice in the current calculation. Note that in this example, the last answer changes and is updated with the current answer each time $\boxed{\leftarrow}$ is pressed.

Example:

Use the result (6.25) of the operation, 50 $\boxed{\div}$ 8, to compute
 $12 \times 5/6.25 + 24 \times 3/6.25 =$

50 $\boxed{\div}$ 8 $\boxed{\leftarrow}$

7 . * 5	35 .
50 / 8	6 . 25

Last answer

12 $\boxed{\times}$ 5 $\boxed{\div}$ $\boxed{\text{ANS}}$

50 / 8	35 .
12 * 5 / 6 . 25	6 . 25

Last answer recalled

$\boxed{+}$ 24 $\boxed{\times}$ 3 $\boxed{\div}$ $\boxed{\text{ANS}}$

50 / 8	35 .
12 * 5 / 6 . 25 + 24 * 3 / 6 . 25	6 . 25

$\boxed{\leftarrow}$

50 / 8	6 . 25
12 * 5 / 6 . 25 + 24 * 3 / 6 . 25	21 . 12

C•CE **ANS**

21.12_

Pressing **←** causes the previous last answer to be replaced with the result of the latest calculation. The last answer is not, however, cleared by pressing the **C•CE** or **SHIFT** + **CA** keys but when the power is turned off.

The last answer can be recalled only when the computer is in the RUN mode, and is replaced when a program is executed.

Maximum Calculation Length

The length of the calculation that can be entered is limited to 255 key strokes before the **←** key is pressed. If you try to exceed this limit, the cursor will start flashing to show that further input is invalid. If this happens, break down the calculation into two or more steps.

Scientific Calculations

The computer has a wide range of numeric functions for use in scientific calculations. PART 5 contains a listing of these functions. Note that the notation of the functions in BASIC may differ from conventional mathematical notations.

All scientific functions may be entered in the RUN mode either by pressing the appropriate function key or entering the BASIC command.

The computer also enables specification of angular units in degrees, radians or gradients using the DEGREE, RADIAN or GRAD commands.

Angular unit	Command	Description
Degrees	DEGREE	Represents a right angle as 90[°].
Radians	RADIAN	Represents a right angle as $\pi/2$ [rad].
Gradients	GRAD	Represents a right angle as 100[g].

For practice, use these instructions to specify angular units when required in the following calculation examples:

Example: $\sin 30^\circ =$

Operation:

DEGREE **←** (specifies "degree" for angular unit)

SIN 30 **←**

or

sin 30 **←**

DEGREE
SIN 30
0.5

Example: $\tan \pi/4 =$

Operation:

RADIAN \leftarrow (specifies "radian" for angular unit)

TAN (PI \div 4)
 \leftarrow

RADIAN	0.5
TAN (PI / 4)	1.

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

Operation:

DEGREE \leftarrow (specifies "degree" for angular unit)

ACS - 0.5 \leftarrow

DEGREE	1.
ACS -0.5	120.

Example: $\log 5 + \ln 5 =$

Operation:

LOG 5 + LN 5 \leftarrow

ACS -0.5	120.
LOG 5+ LN 5	2.308407917

Example: $e^{2+3} =$

Operation:

EXP (2 + 3)
 \leftarrow

LOG 5+ LN 5	2.308407917
EXP (2+3)	148.4131591

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

Operation:

C \cdot CE SQR (4 y^x 3
+ 6 y^x 4) \leftarrow

SQR (4^3+6^4)	36.87817783
---------------	-------------

Example:

Convert 30 deg. 30 min. in sexagesimal notation into decimal notation.

Operation:

DEG 30.30 \leftarrow

SQR (4^3+6^4)	36.87817783
DEG 30.30	30.5

(30.5 DEGREE)

Example:

Convert 30.755 deg. in decimal notation to sexagesimal notation.

Operation:

DMS 30.755 \leftarrow

DEG	30.30	30.5
DMS	30.755	30.4518

(30 DEG. 45 MIN. 18 SEC.)

Example:

Conversion from rectangular into polar coordinates: Determine the polar coordinates (r, θ) for the point P(3, 8) in rectangular coordinates:

Operation:

DEGREE \leftarrow (specifies "degrees" for angular unit)

POL () 3, 8 () \leftarrow

DEGREE	30.4518
POL (3, 8)	8.544003745

Z \leftarrow

POL (3, 8)	8.544003745
Z	69.44395478

The value of θ is stored in variable Z, and the value of r in variable Y.

Example:

Conversion from polar into rectangular coordinates: Determine the rectangular coordinates (x, y) for the point P(12, $4\pi/5$) in polar coordinates.

Operation:

\leftarrow RADIAN \leftarrow (specifies "radians" for angular unit)

REC () 12, () 4 \div 5 \times \leftarrow
PI () () \leftarrow

RADIAN	
REC (12, (4/5*PI))	-9.708203933

Z \leftarrow

REC (12, (4/5*PI))	-9.708203933
Z	7.053423028

The values of y and x are stored in variables Z and Y, respectively.

Note:

For coordinate conversion, the conversion results are stored in variables Z and Y. Therefore, the previous contents of Z and Y will be cleared.

Example:

Convert the hexadecimal number CF8 to its decimal equivalent.

Operation:

C•**CE** **2nd F** **&** **H C F 8** **←**

&HCF8
3320.

“&H” represents a hexadecimal value.

Expressions composed of relational operators (=, >, <, >=, <=, <>) can take on the values listed in the following table (x and y represent numeric values):

=	-1 if $x = y$ 0 if $x \neq y$	>=	-1 if $x \geq y$ 0 if $x < y$
>	-1 if $x > y$ 0 if $x \leq y$	<=	-1 if $x \leq y$ 0 if $x > y$
<	-1 if $x < y$ 0 if $x \geq y$	<>	-1 if $x \neq y$ 0 if $x = y$ (“<>” means “≠”)

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

Note:

Symbols “= >,” “= <,” and “|” do not function as relational operators.

Direct Calculation Feature

In the manual calculations described up to now, the **←** key has always been used to terminate a formula and obtain the calculation result of the formula. However, you can directly operate the functions of the computer with the desired function key (without operating the **←** key) when the objective numeric data is on the display.

Example:

Determine $\sin 30^\circ$ and 8!.

Operation:

DEGREE **←**

C•**CE** **3 0** **sin**

30
0.5

Operation:

C•**CE** **8** **n!**

8
40320.

Example:

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

Operation:

DEGREE \leftarrow
5 \div 12 \leftarrow 2nd F \tan^{-1}

DEGREE	
5 / 12	
	0 . 4 1 6 6 6 6 6 6
	2 2 . 6 1 9 8 6 4 9 5

It should be noted, however, that this “direct” calculation mode is not available for functions requiring the entry of more than one numeric value (binominal functions), such as power, root, or coordinate conversion. The direct calculation feature is effective only for numeric values. Therefore, if hexadecimal numbers A to F are entered for hex to decimal conversion, the direct calculation feature will remain inoperative. In such a case, perform an ordinary manual calculation using the \leftarrow key. The direct calculation feature is not effective for formulas.

Example:

$\text{C}\cdot\text{CE}$ 5 \times 4 \rightarrow 5 \times 4_
 \log \rightarrow 5 \times 4LOG_

If no data is on the display, pressing a function key will display the corresponding BASIC command.

Priority in Direct Input Calculations

You can enter 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 executed by the computer.

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

1. Recalling variables or PI
2. Function (sin, cos, etc.)
3. Power (\wedge), root (ROT)
4. Sign (+, -)
5. Multiplication and division (\times , /)
6. Addition and 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 (3⁴ or 3 \wedge 4 \wedge 2) is operated from right to left.
- For items 3 and 4 above, the last entry has higher priority.

Example:

-2 \wedge 4 \rightarrow -(2⁴)
3 \wedge -2 \rightarrow 3⁻²

Printing of Direct Input Calculations

The calculation steps and results can be printed if the optional printer is connected and switched on, and the **[SHIFT]** + **[P-HP]** keys are pressed (Print mode). Note that calculations made in the CAL mode cannot be printed out.

If a printout is not desired, either switch off the printer, or press **[SHIFT]** + **[P-HP]** again (Non-Print mode).

Calculation Errors

The following types of errors occur in ordinary calculators, pocket computers, and personal computers:

Errors due to Least Significant Digit Processing

Usually, the maximum number of digits that can be calculated in a computer is fixed. For example, $4/3$ results in $1.3333333333\dots$. In a computer with a maximum of eight digits, the first eight digits are significant digits; other least significant digits are either truncated or rounded.

Example:

Computer with 10 significant digits

$$4 \div 3 \rightarrow 1.3333333333\dots$$

10 significant digits

Truncated, rounded

Therefore the calculated result differs from the true value by the amount truncated or rounded. (The difference is the factor of error.)

Example: $4/3 \times 3$

$4 \div 3$	3×3	$\rightarrow 4$	Calculated in succession
$4 \div 3$	3	$\rightarrow 1.333333333$	Calculated independently
$\times 3$	3	$\rightarrow 3.999999999$	

When calculated in succession, the calculation error is reduced.

When calculated independently, the displayed value (10 digits) is used for the calculation.

Errors due to Function Determining Algorithms

The computer uses a variety of algorithms to calculate the values of functions, such as power and trigonometric functions. When calculations use these functions, an additional source of error is introduced. This error factor increases with the number of functions in the calculation. The actual error for each function varies according to the values used and is greatest around singularities and inflection points (e.g., when an angle approaches 90 degrees, the tangent approaches infinity).

PROGRAM OPERATION

Part 3 is devoted to the use of the BASIC programming language as implemented on the PC-E220*. It begins with a general discussion of programming concepts, then moves on to more specific application of these concepts to the PC-E220. Part 3 concludes with some suggestions for programming shortcuts, and for tracing bugs in your program.

This chapter explains the PRO (program) mode and RUN mode (which was discussed in detail in PART 2).

*The PC-E220 is hereafter referred to as "the computer".

8. 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, the computer also uses letters and special symbols in many ways. These letters, numbers, and special symbols are called characters.

In BASIC, a collection of characters is called a "string". In order for the computer to tell the difference between a string and other parts of a program, such as commands or variable names, you must enclose the characters of the string in quotation marks (""). If you wish to use a double quotation mark as a character, enter "CHR\$(H22)".

The following are examples of string constants:

```
"HELLO"  
"Goodbye"  
"SHARP COMPUTER"
```

The following are not valid string constants:


```
"COMPUTER          No ending quotation mark  
"VALUE OF "A"IS"   Quotation mark cannot be used within a string
```

Hexadecimal Numbers





The decimal system is only one of many different systems to represent numbers. Another that 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 to 9 and six more "digits": A, B, C, D, E, and F. These correspond to 10, 11, 12, 13, 14, and 15. When you want the computer to treat a number as hexadecimal, put an ampersand (&) character and "H" in front of the numeral:

```
&HA    = 10  
&H10   = 16  
&H100  = 256  
&HFFFF = 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 enter FRE and press  in the RUN mode. The number displayed is the number of bytes available for writing programs.




This storage technique works well 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, the computer can store large numbers in only 8 bytes. The largest number that can be stored is +9.99999999E + 99. The smallest number is 1.E-99. This gives you quite a wide range. However, if the result of a calculation exceeds this range, the computer will let you know by displaying an error message (see Appendix B). To check this, enter:

9  +  9 9  9 






```
9 E 9 9 * 9
ERROR 2 0
```

To get the computer working properly again, just press the  key. But how do you go about storing all these numbers and strings? It's really very easy. The computer uses names for different pieces of data. Let's store the number 556 in the computer. You may call this number by any name 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. Because the LET command is not usually necessary, we will not use it often.

Enter: R = 556 and press .

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  key, the  key and the  key. The computer responds by showing you the value 556 on the right of the display. This is useful when 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: $A = \pi R^2$. Enter:

R  2 
 +  

```
R = 5 5 6
                    5 5 6 .
R ^ 2 * P I
                    9 7 1 1 7 9 . 3 8 6 6
```

The result is 971179.3866.

This technique of using variables in equations will become more understandable as you get into writing programs.

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

This tells the computer that the contents of variable B\$ are alphabetic, or string data. To illustrate this, enter:

B + = +
 B Y T E +

```
R^2 * P I
                      9 7 1 1 7 9 . 3 8 6 6
B $ = " B Y T E "
B Y T E
```

The string BYTE is now stored in the variable B\$. To make sure of this, press the key and enter the following:

B +

```
B $
B Y T E
```

Types of variables

Variables handled by the computer are divided into the following:

Numeric variables:

- Fixed numeric variables (A to Z)
- Simple numeric variables (AB, C1, etc.)
- Numeric array variables

String variables:

- Fixed string variables (A\$ to Z\$)
- Simple string variables (BB\$, C2\$, etc.)
- String array variables

Fixed Variables

The first type, fixed variables, can be thought of as pre-allocated variables. In other words, no matter how much memory your program uses, you will always have at least 26 variables to choose from in which to store data. This data can be one of two types: NUMERIC or STRING (alphanumeric characters). Fixed memory locations are eight bytes long and can be used for only one type of data at a time.

To illustrate this, enter:

A = 123
 A\$

You get the message:

ERROR 91

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 key to clear the error condition. Now try the following example:

A\$ = "ABC"
 A

Again, the computer is confused and gives the ERROR 91 message. 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.

Simple Variables

Simple variable names are specified by alphanumeric characters, such as AB and C8\$. Unlike fixed variables, simple variables have no dedicated storage area in 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 string 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 used for simple variable names, the first character of a variable name must always be alphabetic and uppercase. More than two characters may be used to define a variable name, however, only the first two will be read by the computer.

Notes:

- The functions and BASIC commands inherent to the computer, for example, PI, IF, TO, ON, SIN, etc., cannot be used as variable names.
- Each simple character variable can hold up to 16 characters and symbols. Each fixed character variable can hold up to 7 characters and symbols.

Array Variables

Sometimes, 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". Arrays can be either one-dimensional, like a list, or two-dimensional, like a table.

Use the DIM (short for dimension) statement to define an array. Arrays must always be declared before they are used (unlike the single-value variables we have been using). The form for the DIMension statement is:

DIM array variable name (size)

where:

array variable name is a variable that conforms to the previously discussed rules for numeric or array variable names.

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

Examples of legal numeric and string DIMension statements are:

```
DIM X(5)           → X (0), X (1), X (2), X (3), X (4), 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 illogical 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 computer. The former denotes a series of numeric storage locations, and the latter denotes a single, entirely different location.

Now that you know how to create arrays, you might be wondering how 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". For example, to store the number 8 in 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 to the number of elements 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 string array, a slightly different form of the DIM statement is used:

DIM string variable name (size) *length

where:

string variable name is a variable name that conforms to the previously discussed rules for normal string variable names.

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

***length** is optional. If used, it specifies the length of each of the strings that compose the array. Length must be a number in the range of 1 to 255. If this clause is not used, the strings will have the default length of 16 characters.

Examples of legal string array declarations are:

```
DIM X$(4)
DIM NM$(10)*10
DIM IN$(1)*255
DIM R$(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 255-character strings, and the last example declares a single string of 26 characters.

Besides the simple array you have just studied, the computer 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 array name (rows, columns) or
DIM string array name (rows, columns) *length
 where:

rows specifies the number of rows in the array. This must be a number in the range of 0 to 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 of 0 to 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) that pertain to each 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 use up storage space. For example, an array with 25 rows and 35 columns uses 875 storage locations!

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

Variable type	Number of bytes used	
	Variable name	Data
Numeric variable Numeric array variable	7 bytes	8 bytes
String variable	7 bytes	16 bytes
String array variable	7 bytes	Specified number

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

Element	Line number	Statement & function	 , others
Number of bytes used	3 bytes	2 bytes	1 byte

Program Files (RAM disk)

Program files are fundamental in the use of your computer.

Part of the computer's internal memory can be used as a RAM disk. Only programs can be stored to the RAM disk; no data can be stored to it. The program stored in the RAM disk must be loaded into the program data area (user area) before it is executed. (See the BASIC COMMAND DICTIONARY for details regarding SAVE, LOAD, KILL, FILES.)

Filenames

Before saving to a storage medium such as a RAM disk, a file must be given a name. This name is used to load the file into computer memory. The filename may be up to 8 characters in length and can include the following characters:

A - Z, a - z, 0 - 9, #, \$, %, &, ', (,), {, }, -, @

Extension

A file extension is an additional way of identifying the type of file (e.g., BASIC program file or text file). The extension consists of three characters added to the end of the filename and separated from it by a period. The extension is specified when the file is saved.

BASIC programs are automatically given the extension .BAS when saved using the SAVE command. When reloaded into memory using the LOAD command, you do not need to specify the .BAS extension.

When the FILES or LFILES command is used to list the files on the RAM disk, BASIC programs will appear with the .BAS extension unless some other extension has been specified by the user when the file was saved.

Expressions

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

Numeric Expressions

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

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

These are the arithmetic operators that you used when exploring the use of the computer as a calculator in Chapter 4.

The following are valid numeric expressions:

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

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 addition. When used with a pair of strings, the + attaches the second string to the end of the first string to make one longer string. You should take care when making complex string concatenations and other string operations because the work space available for string calculations is limited to 255 characters (see page 169).

Note:

String quantities and numeric quantities cannot be combined in the same expression unless one of the functions that converts a string value into a numeric value, or vice versa, is used:

"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
- = Equal to
- < > 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 is equal to 10, B equal to 12, C(1,2) equal to 6, and D(3) equal to 9, all of these relational expressions will 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 C). If one string is shorter than the other, a 0 or NULL 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 true or false. The computer represents true by a -1, and false by a 0.

Logical Expressions

Logical operations use the Boolean algebra functions AND, OR, and NOT to build connections between relational expressions. The logical operations in a single expression are evaluated after arithmetic and relational operations.

In this way, logical operators can be used to make program decisions based on multiple conditions using the IF ... THEN statement.

Example:

```
IF A < = 32 AND B > = 90 THEN 150
```

This statement causes execution to jump to line number 150 if the value of the numeric variable A is less than or equal to 32 and, at the same time, the value of numeric variable B is greater than or equal to 90.

```
IF X < > 13 OR Y = 0 THEN 50
```

This statement causes execution to jump to line 50 unless variable X has the value 13, or if variable Y is equal to 0.

In a logical operation involving two numbers in the range -32768 to +32767, the two numbers are converted into 16-bit binary integers (in two's complement form) and the logical connection is then evaluated for each corresponding pair of bits in the two numbers.

The results returned by the logical operators for these bit evaluations are:

AND			OR			NOT	
X	Y	X AND Y	X	Y	X OR Y	X	NOT X
1	1	1	1	1	1	1	0
1	0	0	1	0	1	0	1
0	1	0	0	1	1	1	0
0	0	0	0	0	0	0	1

After each bit pair has returned the corresponding result (a 1 or a 0) according to the above tables, the resulting 16-bit binary number is converted back to a decimal value. This number is the result of the logical operation.

Example:

$$\begin{array}{l}
 41 \text{ AND } 27 \rightarrow \\
 \text{equals} \\
 9
 \end{array}
 \qquad
 \begin{array}{l}
 41 = 101001 \\
 27 = \underline{011011} \text{ AND} \\
 \leftarrow 001001
 \end{array}$$

$$\begin{array}{l}
 41 \text{ OR } 27 \rightarrow \\
 \text{equals} \\
 59
 \end{array}
 \qquad
 \begin{array}{l}
 41 = 101001 \\
 27 = \underline{011011} \text{ OR} \\
 \leftarrow 111011
 \end{array}$$

$$\begin{array}{l}
 \text{NOT } 3 \rightarrow \\
 \text{equals} \\
 -4 \text{ (two's complement form)}
 \end{array}
 \qquad
 \begin{array}{l}
 3 = 0000000000000011 \text{ NOT} \\
 \leftarrow \underline{1111111111111100}
 \end{array}$$

NOT X can generally be calculated by the equation $\text{NOT } X = -(X+1)$.

Parentheses and Operator Precedence

When evaluating complex expressions, the computer follows a predefined set of priorities that determine the sequence in which operators are evaluated.

$5 + 2 * 3$ could be

$$\begin{array}{l}
 5 + 2 = 7 \quad \text{or} \quad 2 * 3 = 6 \\
 7 * 3 = 21 \quad \quad 6 + 5 = 11
 \end{array}$$

The exact rules of "operator precedence" are given on page 70.

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

$$(5 + 2) * 3 \text{ or } 5 + (2 * 3)$$

9. PROGRAMMING

In the previous chapter, you examined some of the concepts and terms of the BASIC programming language. In this chapter, you will use these elements to create programs. However, this is not a manual on how to program in BASIC. This chapter will provide a general explanation of how to use BASIC on your computer.

Programs

A program consists of a set of instructions to the computer. Remember that the computer 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 computer interprets instructions according to a predetermined format. This format is called a statement. You must always enter BASIC statements in the same pattern. Statements must be preceded by a line number.

Example:

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

Line Numbers

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

You can use the AUTO command to automatically insert line numbers for you. Each time you press the  key, a new line number, with the correct increment, will be automatically inserted. (See the BASIC COMMAND DICTIONARY for a full description of this useful function.)

It is wise to allow increments of several numbers in your line numbering (10, 20, 30, ... 10, 30, 50, etc.). This will enable you to insert additional lines if necessary.

If you use the same line number more than once, the old line will be deleted when you enter the new line.

Labelled Programs

Often you will want to store several different programs in 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. However, there is an easier way. You can label each program with alphanumeric characters and run the program.

Label the first line of each program that you want to reference. The label consists of a letter and alphanumeric characters, with * in front of it or in quotes, followed by a colon.

Example:

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

Although both *label and "label" forms may be used, *label is recommended, since it executes more quickly and is more visible in the program listing.

BASIC Commands

All BASIC statements must contain commands. They tell the computer what action to perform. A command is contained in a program, and as such is not acted upon immediately. Some statements require or allow an operand.

Example:

```
10: DATA "HELLO"  
20: READ B$  
30: PRINT B$  
40: END
```


Operands provide information to the computer telling it what data the command will act upon. Some commands require operands; with other commands they are optional. Certain commands do not allow operands. (See the BASIC COMMAND DICTIONARY for BASIC commands and their uses.)

Note:

Commands, functions and variables entered in lowercase characters will be converted to uppercase characters.

Direct Commands

Direct commands are instructions to the computer that are entered outside of a program. They instruct the computer to perform some immediate action or set modes that affect how your programs are executed.

Direct commands have immediate effect — as soon as you complete entering direct commands (by pressing the  key), the command will be executed. Direct commands are not preceded by a line number.



```
RUN
NEW
RADIAN
```


Modes

Remember that you can set the computer in the CAL or RUN mode when using it as a calculator. The RUN mode is also used to execute the programs you create. The PRO (program) mode is used to enter and edit your programs.

Beginning to Program

Now you are ready to program!


After all your practice in using the computer 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 the  key to access characters above the keys and to end every line by pressing the  key.

To enter program statements into the computer, the computer must first be placed in the PRO mode using the  key. The following will appear:

```
PROGRAM MODE
>
PRO
```

Enter the NEW command.
NEW 

```
PROGRAM MODE
NEW
>
```


The NEW command clears the memory of all existing programs and data. The prompt appears after you press the  key, indicating that the computer is awaiting input.

Entering and Running a Program

Make sure the computer is in the PRO mode and enter the following program:

```
10 PRINT  
[SHIFT] + ["] HELLO  
[SHIFT] + ["]
```

```
PROGRAM MODE  
NEW  
10PRINT "HELLO" _
```

Notice that the computer automatically inserts the colon between the line number and the command when you press the  key.

Check that the statement is in the correct format and then change the mode to RUN by pressing the **BASIC** key.

```
[C•CE] RUN 
```








```
RUN  
HELLO  
>
```



Since this is the only line of the program, the computer will exit the program and return to the BASIC prompt ">".


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 PRO mode.

You need to recall your program in order to edit it. Use the up arrow key  to recall your program. If your program was completely executed, the  key will recall the last line of the program. If there was an error in the program, or if you used the **BREAK** key to stop execution, the  key will recall the line in which the error or break occurred. To make changes in your program, use the  key to move up in your program (recall the previous line) and the  key to move down in your program (display the next line). If held down, the  or  key will scroll vertically (up or down) through your program.

Remember that to move the cursor within the program line you use the  (right arrow) and  (left arrow) keys.

Using the  key, position the cursor over the first character you wish to change:

```

```

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



```
10 PRINT "HELLO"
```

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

GOODBYE **SHIFT** + **"**
SHIFT + **!**

```
10 PRINT "GOODBYE"!_
```

Remember to press the **←** key at the end of the line. Change to the RUN mode.

R U N **←**

```
RUN MODE
RUN
ERROR 10 IN 10
```

The error message indicates the type of error, and the line number in which the error occurred.

Press the **C•CE** key to clear the error condition.

And return to the PRO mode. You must be in the PRO mode to make changes in a program. Using **↑** (or **↓**), recall the line in which the error occurred.

↑ (or **↓**)

```
10 PRINT "GOODBYE"!_
```

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

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. Using **←**, position the cursor on top of the character that will be the first character following the insertion.

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


← **INS**

```
10 PRINT "GOODBYE " _
```



Enter the !. The display looks like this:

SHIFT + **!**

```
10 PRINT "GOODBYE!"
```

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

Notes:

- If you wish to DELETE an entire line from your program, just enter the line number and the original line will be eliminated. The DELETE command can be used to delete more than one line at a time.
- In the PRO mode, if keys are pressed when the cursor is not displayed, their corresponding characters are usually displayed from the leftmost column of the display. However, if the  or  key is pressed when the cursor is displayed, successive key entries are displayed starting from the cursor position.

Using Variables in Programming

If you are unfamiliar with the use of numeric and string variables in BASIC, reread the appropriate sections in Chapter 8.

Using variables in programming allows more sophisticated use of the computer's abilities.

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

```
A = 5
```

To assign string variables you use a letter followed by a dollar sign.

Do not use the same letter in designating a numeric and a string fixed variable. You cannot designate A and A\$ in the same program.

Remember that each string fixed variable must not 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 entered or computed during the program. One way to assign a variable is to use the INPUT command. In the following program, the value of A\$ will change in response to the data entered in answer to the inquiry "WORD?".

Enter this program:

```
10: INPUT "WORD?";A$
20: B=LEN(A$)
30: PRINT "THE WORD (";A$;) HAS"
40: PRINT B;" LETTERS"
50: END
```

Note:

To enter the "=" sign, press the **SHIFT** +  keys.

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 PRO mode, displays program lines beginning with the lowest line number.

Try listing the above program:

LIST

```
10: INPUT "WORD? "; A$
20: B= LEN (A$)
30: PRINT "THE WORD ( "; A$
    ; " ) HAS "
```

Use the and keys to move through your program until you have reviewed the entire program. After checking your program, change to the RUN mode:

RUN

```
RUN
WORD? _
```

HELP

```
WORD?HELP
THE WORD (HELP) HAS
4 . LETTERS
>
```

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 or key (or LIST) to reach line 50, or enter:

LIST 50

```
50: END
```

You may enter 50 to delete the entire line or use the key to position the cursor over the E in END. Change line 50 so that it reads:

```
50: 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 press the key.

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

More Complex Programming

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

```
100: F = 1: WAIT 128
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 command in line 100 controls the 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.

Notice that there are two statements in line 100 separated by a colon (:). You may put as many statements as you wish on one line (separating each by a colon) up to a maximum of 255 characters. Multiple-statement lines can make a program hard to read and modify, 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.

In this program we have used the FOR command in line 120 and the NEXT command in line 150 to create a loop. In a previous program you created an infinite loop that kept repeating the statements inside the loop until you pressed the **BREAK** key. With this FOR...NEXT loop, the computer adds 1 to N each time execution reaches the NEXT command. 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 to line 160 and the program stops.

You may use any fixed numeric variable or single-precision simple numeric variable in a FOR...NEXT loop. You do not have to start counting at 1, and you can increment any amount at each step. (See the BASIC COMMAND DICTIONARY for details.)

We have labeled this program with line numbers starting with 100. Labeling 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, change to the RUN mode and enter:

```
C•CE
R U N 1 0 0
```

You could also give the program a name using a label and start the program with RUN *label.

Notes on the PRINT command:

If more than four lines must be displayed, the first lines will scroll up off the display, and cannot be recalled. Use the WAIT command in the program to display data more slowly, or use the printer. (See the BASIC COMMAND DICTIONARY for details about the WAIT or LPRINT command.)

The WAIT command applies to every PRINT command. Break long PRINT commands into a number of shorter commands if the display scrolls too quickly.

Example:

```
100 PRINT A, B, ..., P
      ↓
100 PRINT A, B, ..., H: PRINT I, J, ..., P
```

Since the WAIT command is not supported by many personal computers, a wait loop such as FOR J=1 TO 500:NEXT J can also be used to extend the display time.

Storing Programs in Memory

You will remember that settings and functions remain in the computer even after it is turned off. Programs also remain in memory when you turn off the computer, or it undergoes an Auto OFF. Even if you use the **BREAK**, **C•CE**, or **SHIFT + CA** keys, the programs will remain in memory.

Programs are lost from memory only when you:

- Enter NEW before beginning programming in the PRO mode.
- Initialize the computer using the RESET button.
- Create a new program using the same line numbers as a program already in memory.

Program Execution

More than one program can be stored in the computer if the memory capacity is not exceeded. Execute the second or subsequent program using one of the following:

The RUN command: RUN line number

The GOTO command: GOTO line number

Execution begins from the specified line number. If a label such as *AB is entered in the program, the program can be executed by entering RUN*AB .

The following lists the differences between the variables and status when a program is executed using the GOTO and RUN commands.

Execution using RUN	Execution using GOTO
<ul style="list-style-type: none">• Clears the WAIT setting.• Clears the USING format.• Clears array and simple variables.• Initializes the DATA statement for the READ statement.• Clears the PRINT=LPRINT setting.• Closes the parallel port.	<ul style="list-style-type: none">• Retains the WAIT setting.• Retains the USING format.• Retains array and simple variables.• Does not initialize the DATA statement for the READ statement.• Retains the PRINT=LPRINT setting.• Leaves the parallel port open.




Note:

When the program is executed using the RUN command, variables for data are cleared. (Fixed variables are retained.) To retain the data, execute using the GOTO command.



10. DEBUGGING

After entering a new BASIC program, it often does not work the first time. Even if you are simply entering a program that you know is correct, such as those in this manual, it is common to make at least one typing error. The program may also contain at least one logic error.

Following are some general hints on how to find and correct your errors. If you run your program and get an error message:

1. Go back to the PRO mode and use the  or  key to recall the line with the error. The cursor will be positioned at the place in the line where the error occurred.
2. If you cannot find an obvious syntax error, the problem may lie with the values that are being used. For example, CHR\$(A) will produce a space if A has a value of 1. Check the values of the variables in either the RUN or PRO mode by entering the name of the variable and pressing the  key.

If you run the program and don't get an error message, but the program doesn't do what you expect:











1. 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 found by just taking another look at the program.
2. 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.
3. 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 from 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.
4. Use TRON (Trace ON) and TROFF (Trace OFF), either as direct commands or 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 it is sometimes the only way.

Trace mode

No matter how careful you are, eventually you will create a program that does not do quite what you expect it to. To isolate the problem, BASIC has a special method of executing programs known as the "Trace" mode.

TRON (Trace ON) starts TRACE mode. The TRON instruction may be issued as a direct command (in RUN mode) or it may be embedded within a program. Used as a direct command, TRON informs the computer that tracing is required during the execution of all subsequent programs. The programs to be traced are then started in a normal manner, with a GOTO or RUN command. If TRON is used as a statement, it will initiate the TRACE mode only when the line containing it is executed. If, for some reason, that line is never reached, TRACE mode will remain inactive.

Debugging Procedures

1. Set the computer to the RUN mode.
2. Enter TRON  to specify the TRACE mode.
3. Enter RUN  to execute the program. After executing each line, the computer will suspend execution, displaying the number of the line just executed.
4. Press the  key to move to the line to be checked. Holding the  key will execute the program step by step. Releasing the key will stop program execution. You can view the contents of the trace line by holding down the  key. (When you release the  key, the command line prompt will appear. To continue trace execution, press the  key.)
5. To resume execution, enter CONT . However, if execution is interrupted during data entry using the INPUT command, just press the  key as for usual program continuation.
6. Continue the trace procedure and check if the program is executing properly by confirming program execution order and variable contents after each line is executed. If the program is not executing properly, correct the logic.
7. After debugging, enter TROFF  to exit the TRACE mode.

Example:





```

10 INPUT "A=";A,"B=";B
20 C=A*2
30 D=B*3
40 PRINT "C=";C;" D=";D
50 END

```

Run the program.




```

RUN mode
TRON 
RUN 
8  (Data entry)
9  (Data entry)




```

>
A=_
B=_ } Execute INPUT command




```

10:
 20:
 30:
 C=16. D=27.
40:


```

When execution is interrupted with the  key, recall the variables manually and check that the values are as expected. Pressing the  key will execute one line at a time and entering CONT  will execute the lines continuously.





Notes:

- If a result or other information is displayed at a location specified by the LOCATE command, the next line number will appear on the line next to that location. (See the BASIC COMMAND DICTIONARY for details of the LOCATE command.)
- If you manually recall a variable or do manual calculation when a location has been specified with the LOCATE command, the specification will be cleared.
- The TRACE mode will remain in effect unless TROFF  is entered, the  +  keys are pressed, or the power is turned off.
- If a comment line is executed in the TRACE mode, the line number for that comment line will not be displayed. In this case, the number of the last executed line other than the comment line will be displayed.

To debug by interrupting program execution, perform one of the following:

- Press the  key during program execution.
- Enter the STOP command at the location to be stopped.

The Break message will be displayed and execution will be interrupted. Then:

1. Check the variable contents manually.
 2. Press the  key to execute subsequent statements line by line.
Enter CONT  to return to previous operation.
- A program interrupted by the  key or the STOP command can be executed line by line by pressing the  key.

ASSEMBLER OPERATION

The PC-E220* has a built-in Assembler that allows the user to learn machine language. Part 4 explains how to write a source program in ASCII format in the TEXT mode, and how to convert TEXT and BASIC programs.

Chapter 12 explains how to execute the assembled program (or "object") in the Monitor mode.

* The PC-E220 is hereafter referred to as "the computer".

11. TEXT MODE (TEXT EDITOR)

The TEXT mode (text editor) allows you to write or edit programs in ASCII format, and input and output them through the SIO.

All BASIC commands for the computer are stored in a 2-byte format called "intermediate code". Because this code will differ depending on the hardware or BASIC interpreter being used, the code cannot be used for communications between personal computers or other devices. ASCII code is generally used for data communications between personal computers since ASCII representations of alphanumeric characters and basic symbols are the same regardless of the hardware being used.

The TEXT mode of the computer allows you to write, edit, or save programs in ASCII, or to translate the program from intermediate code (BASIC) into ASCII, or vice versa.

This chapter describes the TEXT mode functions.

Text Mode Functions

The functions available in the TEXT mode are:

- SHIFT** + **TEXT** → **TEXT mode** (Text Editor)
- **Edit** (Programming and program editing)
 - **Del** (Program deletion)
 - **Print** (Program listing output to printer)
 - **Cmt** (Program I/O to cassette tape)
 - Save (write)
 - Load (read)
 - Verify (compare)
 - **Sio** (Serial I/O)
 - Save (send)
 - Load (read)
 - Format (set parameters)
 - **File** (I/O to RAM disk)
 - Save (register a filename)
 - Load (recall)
 - Kill (delete file)
 - Files (recall a filename)
 - **Basic** (Program conversion between BASIC and TEXT formats)
 - Basic ← text (converts from TEXT to BASIC)
 - Text ← basic (converts from BASIC to TEXT)

Selecting the TEXT Mode

To select the TEXT mode, press the **SHIFT** + **TEXT** keys. The computer will display the TEXT EDITOR screen, as shown. This screen is also called the Main Menu.

```
*** TEXT EDITOR ***
Edit   Del   Print Cmt
Sio    File  Basic
```

From this menu select the desired function by entering the first letter (shown capitalized) of the function name. Once a function is selected, the computer may display a submenu for that function, or directly execute the selected function.

Notes:

- To stop a function from being executed, or to return to a function submenu or the Main Menu, press the **BREAK** key. Use the **C•CE** key to clear an error or delete one or more characters that have been entered (e.g. the filename).
- Exit the TEXT mode by selecting any other mode or by turning the computer off and then on again.

Edit

To select the Edit mode from the Main Menu, press the **E** key.

E

```
TEXT EDITOR
<
```

In the Edit mode, the command line prompt is "<" (instead of ">", as in the BASIC mode).

As with BASIC programs, each line of a TEXT program must be preceded by a line number. However, with a TEXT program, the computer does not automatically insert a colon (:) after each line number as it does in BASIC programs, nor does it insert a space following each entered command. Each line will appear just as you type it.

Notes:

- Program line numbers are automatically sorted into ascending order.
- The range of line numbers that can be used in a program is from 1 to 65279. If this range is exceeded or no line number is entered, an error message (LINE NO. ERROR) will be displayed. Press the **C•CE** key to clear the error.

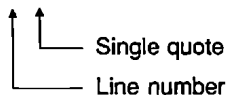
To return to the Main Menu, press **BREAK**.

Note:

A TEXT line cannot begin with a numeral directly following the line number. If you wish to begin a line with a numeral, separate it from the line number with a single quote (').

Example:

50 ' 100 FORMAT (17X, A) 



(Sample Programming) Enter the following program:

```
10INPUT A
20B=A * A
30PRINT A, B
40END
```

```
10INPUT SPACE A
20B=A * A
30PRINT SPACE A, B
40END
```

```
1 0 I N P U T   A
2 0 B = A * A
3 0 P R I N T   A , B
4 0 E N D
```

Note:

See page 120 for details about assembling the sample TEXT program.

Program Editing

You edit a TEXT program in much the same way that you would a BASIC program. (See the explanation of BASIC programming in chapter 9.)

The L (List) and R (Renumber) commands in the TEXT mode take the place of the LIST and RENUM commands in BASIC. (For details of the L and R commands, see the explanations for the LIST and RENUM commands in the BASIC command dictionary.)

However, if you execute the R command in a TEXT program that has been converted from a BASIC program, it will renumber only those line numbers at the beginning of each line, but will not change line numbers within the GOTO, THEN, GOSUB, or RESTORE command statement. As a result, the program will not run properly if it is converted back to BASIC.

L Command Formats:

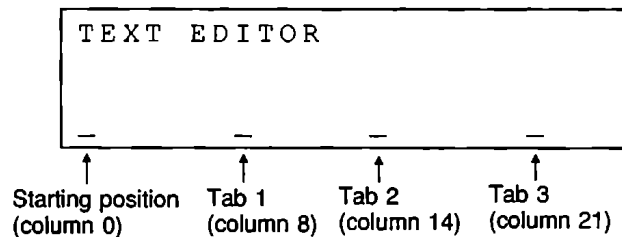
- (1) L
- (2) L line number
- (3) L label

R Command Format:

R [new line number][,[starting line]][,increment]

The **TAB** Key

In the Edit mode, the **TAB** key lets you position the cursor at tab positions.



When you first press the **TAB** key, the cursor advances to the eighth column. Pressing it a second time moves the cursor to the fourteenth column (six columns from the first tab position). Each time you subsequently press the key, the cursor will advance seven columns.

Deleting a TEXT Program (Del)

To select the Delete mode, press the key at the Main Menu.

```
*** TEXT EDITOR ***  
TEXT DELETE OK? (Y)
```

If is pressed, the memory's TEXT area will be cleared of all contents including the TEXT program, and the Main Menu will be displayed.

If any key other than is pressed, the computer returns to the Main Menu without executing deletion.

Note:

If no text has been stored in the text area, nothing will happen when you press at the Main Menu.

Printing a TEXT Program Listing (Print)

Connect the optional CE-126P printer to the computer, and turn on the computer and printer. Press at the Main Menu, and the TEXT program stored in the memory's text area will be printed.

```
*** TEXT EDITOR ***  
--- PRINTING ---
```

When printing is completed, the computer will return to the Main Menu.

Note:

To abort printing, press the key. If the printer is not turned on or is not connected to the computer, nothing will happen when you press at the Main Menu.

Saving, Loading, and Verifying a TEXT Program with Cassette Tape (Cmt)

Press the key at the Main Menu, and the computer will display the Cassette Menu (CMT).

```
<< CMT >>  
Save Load Verify
```

From the Cassette Menu, select the function you want to execute — Save, Load or Verify — by typing in the first letter of the name of the desired function.

Before selecting any function, connect the optional CE-126P printer and cassette tape recorder to the computer and prepare for saving (record) or loading (playback).

Saving a program to cassette tape (Save)

Press the **S** key at the Cassette Menu (CMT), and the computer will prompt you for the name of the file you want to save.

S

```
<< CMT >>
→Save Load Verify
FILE NAME=?
```

Enter the filename, then press **←**, and saving of the program will begin.

Example:

Save a file with the filename "TEXT"

TEXT

```
<< CMT >>
→Save Load Verify
FILE NAME=TEXT_
```

←

```
<< CMT >>
--- SAVING ---
```

When saving is completed, the computer will return to the Cassette Menu (CMT). To verify that the file was successfully saved, select the Verify function from the Cassette Menu.

Notes:

- If you press the **←** key at the filename prompt without entering a filename, the program will be saved to the cassette tape without a filename.
- A filename can be up to eight characters long.
- If no TEXT file has been stored, the computer will return to the Cassette Menu when you press the **←** key at the filename prompt.

Saving/loading the entire contents of the RAM disk

You can use a wild card (*.*) to save all the files stored on the RAM disk. Enter " *.* " at the filename prompt. (You'll find this is convenient to temporarily save the entire contents of the RAM disk in order to arrange the necessary machine code area.)

To load all the files from a cassette tape, use the wild card (*.*) at the filename prompt after you select the Load function at the Cassette Menu. The files will be loaded in the format they were written in (e.g. ASCII or BASIC intermediate code).

Loading a program from a cassette tape

Press the **L** key at the Cassette Menu (CMT), and the computer will prompt you for the name of the file you want to load.

L

```
<< CMT >>
Save →Load Verify
FILE NAME=?
```

Enter the name of the file to load, then press **←**, and the computer will begin loading the program.

Example:

Load a file with the filename "TEXT"

TEXT **←**

```
<< CMT >>
--- LOADING ---
```

The computer searches for the specified filename (TEXT, in this example) on the tape and loads the ASCII contents of the file into the text area.

While the program is loading, an asterisk (*) appears in the lower right corner of the display.

When loading is completed, the asterisk disappears, and the computer will return to the Cassette Menu. To verify that the file was successfully loaded, select the Verify function from the Cassette Menu to check the contents of the loaded file.

Notes:

- If you press the **←** key at the filename prompt without entering a filename, the computer will load the first ASCII file that was encountered after the tape began running.
- If the specified file is not found, the computer will continue searching after the tape stops. Press the **BREAK** key to stop the search.
- If an error occurs during loading or loading is aborted with the **BREAK** key, only the portion of the file that was loaded up to that point will be in the text area.

Verifying the contents of a saved/loaded file

The Verify function lets you confirm that a program has been properly saved to a cassette tape or loaded from it.

Press the **V** key on the Cassette Menu to select the Verify function. The computer will prompt you for the name of the file you want to verify.

V

```
<< CMT >>
Save Load →Verify
FILE NAME=?
```

Enter the name of the file to verify, then press **←**, and the computer will begin the verification operation.

Example:

Verify the contents of a file with the filename "TEXT"

TEXT

```

      << CMT >>
    --- VERIFYING ---
  
```

The computer searches for the specified file (TEXT, in this example) on the tape and compares the ASCII contents of the file with the contents loaded into the text area. During comparison, an asterisk (*) appears in the lower right corner of the display. When verification is completed, the asterisk disappears and the computer will return to the Cassette Menu.

If a data mismatch is found, an error message (VERIFY ERROR) is displayed.

Notes:

- If you press at the filename prompt without typing a filename, the computer will verify the first ASCII file that was encountered after the tape began running.
- If the specified file is not found, the computer will continue searching after the tape stops. Press the **BREAK** key to stop the search.

Serial I/O (Sio)

Press the key at the Main Menu, and the computer will display the Serial I/O (SIO) Menu.

```

      << SIO >>
    Save  Load  Format
  
```

Select any of the three SIO functions — Save, Load, and Format — from this menu. Enter the first letter of the name of the desired function.

Setting I/O parameters (Format)

Format lets you set the serial communication parameters. The communication parameters must be matched to those on the device the computer is communicating with.

Press the key at the Serial I/O Menu, and a help screen will be displayed. Press any key or wait a short while and a display of the communication parameter setup will follow.

```

      << SIO >>
    Select ←,→,↑,↓ key
    Set      ↓ key
    --- push any key ---
  
```

Press any key or wait a short while and the display shown will appear.

```

    →baud rate =1200
      data bit  =8
      stop bit  =1
      parity    =none
  
```

→ indicates the parameter selection. To move → to select a parameter to change, use the or key. Seven parameters can be set. Parameters can be scrolled onto the display by pressing the key.

⋮

```

parity      =none
end of line =CR LF
end of file =1A
→line number =yes

```

Use the or key to change a setting. The setting for the “end of file” parameter, however, must be entered manually. After making a change, press to store the change in memory. If the new setting is not stored, the computer will use the previous parameter setting.

Explanation of communication parameters

- Baud rate : 300, 600, 1200, 2400, 4800
Baud rate specifies the speed of data transfer. The greater the baud rate, the higher the speed of data transfer. You can select a baud rate of 300, 600, 1200, 2400, or 4800 bps (bits per second).
- Data bit : 7 or 8
The data bit specifies how many bits will be included in a single character of data. Either 7 or 8 bits can be specified.
- Stop bit : 1 or 2
The stop bit specifies the length of the stop bit at the end of each piece of data.
- Parity : none, even, or odd
The parity specifies the type of parity checking (error detection).
none ... No parity bit is added to transferred data (no error checking).
even ... Specifies even parity.
odd ... Specifies odd parity.
- End of line : CR, LF, or CR + LF
End of line specifies the delimiter code used to indicate the end of each program line.
CR ... Specifies a carriage return (CR) code.
LF ... Specifies a line feed (LF) code.
CR + LF ... Specifies CR and LF codes.
- End of file : 00 to FF (2-digit hex numeral)
End of file specifies the end-of-text code used to indicate the end of a program or other file.
- Line number: yes or no
The line number selects whether to send a TEXT program with or without line numbers.
yes ... Program is sent with line numbers.
no ... Program is sent without line numbers.
The line number also selects whether or not line numbers (in increments of 10) are to be automatically assigned to program lines as they are received.
yes ... Line numbers are not assigned. Select “yes” when the program being received already has line numbers.
no ... Line numbers are automatically assigned.
An error message (LINE NO. ERROR) will be displayed if the received file has no line numbers though “yes” was specified.

On the computer, the communication parameters are initially set to the values shown:

Parameter	Condition
baud rate	1200
data bit	8
stop bit	1
parity	none
end of line	CR LF
end of file	1 A
line number	yes

You can change the values of these parameters as described under "Setting I/O parameters (Format)." Once changed and stored in memory, the parameter values are retained until the RESET button is pressed to clear memory, the battery is replaced, or the settings are again changed.

Sending programs (Save)

Press the **S** key at the Serial I/O Menu, and the computer will begin sending a TEXT program through the serial I/O port.

S

```
<< SIO >>
--- SENDING ---
```

When sending is completed, the computer will return to the Serial I/O Menu.

Notes:

- Press the **BREAK** key to abort sending. The computer will return to the Serial I/O Menu.
- If no program is stored in the text area, nothing will happen when the **S** key is pressed.

Receiving programs (Load)

Press the **L** key at the Serial I/O Menu, and the computer will begin receiving a TEXT program through the serial I/O port.

L

```
<< SIO >>
--- RECEIVING ---
```

When receiving is completed, the computer will return to the Serial I/O Menu.

Notes:

- Press the **BREAK** key to abort receiving. The computer will return to the Serial I/O Menu.
- An error message (I/O DEVICE ERROR) will be displayed if the program was not properly received or a parity error occurred. Press the **C•CE** key to clear the error.

Program File (File)

Press the **[F]** key at the Main Menu, and the computer will display the Program File Menu.

[F]

```
<< PROGRAM FILE >>
Save   Load   Kill   Files
```

Select any of the functions — Save, Load, Kill, and Files — from this menu. Enter the first letter of the name of the desired function.

Registering a TEXT program (Save)

Registering a TEXT program means assigning a name to it. The computer then references this name when carrying out operations on the file.

Press the **[S]** key at the Program File Menu, and the computer will prompt you for the name of the file you want to register.

[S]

```
<< PROGRAM FILE >>
→Save   Load   Kill   Files
FILE NAME=?
```

Enter the filename, and press **[←]**. The computer will register the file.

Example:

Register a file with the filename "TEST"

TEST

```
<< PROGRAM FILE >>
→Save   Load   Kill   Files
FILE NAME=TEST_
```

[←]

```
<< PROGRAM FILE >>
Save   Load   Kill   Files
```

The computer registers the file "TEST", then returns to the Program File Menu.

Notes:

- Once the Save function has been selected from the Program File Menu, a filename must be entered. If you press the **[←]** key at the filename prompt without entering a filename, an error message (ILLEGAL FILE NAME) will be displayed. Press the **[C•CE]** key to clear the error.
- The filename can be up to eight characters long, and have a file extension of up to three characters. If no file extension is specified, the computer automatically assigns the file extension ".TXT".
- File registration is not possible if no TEXT program is stored in the text area.

Recalling a TEXT file (Load)

Press the key at the Program File Menu, and the computer will display a list of registered files, with "LOAD →" pointing to the first filename (if no program is registered, nothing will happen when the key is pressed).

(An example of a list of registered files is shown.)

```
LOAD →ABC      .TXT
      PRO       .TXT
      SAMPLE01 .BAS
      TEST      .TXT
```

Use the or key to position the "LOAD →" pointer at the name of the file you want to recall, then press the key. The computer will load the contents of the specified file into the text area, then return to the Program File Menu.

Note:

In the TEXT mode, only programs and files that were registered in the TEXT mode can be recalled. An error message (FILE MODE ERROR) will be displayed if you attempt to load a BASIC program that was saved using the BASIC SAVE command. Press the key to clear the error.

Killing a program file (Kill)

The Kill function deletes a specified file.

Press the key at the Program File Menu, and the computer will prompt you for the name of the file you want to kill.

```
<< PROGRAM FILE >>
Save  Load →Kill  Files
FILE NAME=?
```

Enter the name of the program file to delete, then press . The computer will delete the file.

Example:

Kill a file with the filename "TEST"

TEST

```
<< PROGRAM FILE >>
Save  Load  Kill  Files
```

The computer will delete the file "TEST", then return to the Program File Menu.

Notes:

- If the specified filename has no file extension, ".TXT" is automatically assumed.
- An error message (FILE NOT FOUND) will be displayed if the specified file is not found. Press the key to clear the error.

Listing filenames (Files)

Press the key at the Program File Menu, and the computer will display a list of all registered files, with → pointing to the first filename in the list (if no file is registered, nothing will happen when the key is pressed).

F

(An example of a list of registered files is shown.)

```
→ABC      .TXT
PRO       .TXT
SAMPLE01 .BAS
```

You can view the remaining portion of the list, if any, by scrolling using the **↓** or **↑** key.

To load the program indicated by →, press **SHIFT** + **LOAD** (or **2nd F** **LOAD**).

BASIC Converter (Basic)

The Basic function converts a BASIC program in intermediate code format to a TEXT file in ASCII format, or vice versa. This function is convenient to file or handle your PC-E220 BASIC programs on your personal computer.

Press the **B** key at the Main Menu, and the computer will display the BASIC Converter Menu.

B

```
<< BASIC CONVERTER >>
Basic←text  Text←basic
```

This screen lets you select the direction of conversion, from TEXT to BASIC or from BASIC to TEXT. Type in the first letter of the destination format type.

Conversion between TEXT and BASIC programs (between ASCII and intermediate code formats)

Press the **B** key at the BASIC Converter Menu, and the computer will convert the TEXT program in the text area to a BASIC program and store it to the program data area.

Press the **T** key at the BASIC Converter Menu, and the computer will convert the BASIC program in the program data area to a TEXT program and store it in the TEXT area.

Example:

Convert a TEXT program to BASIC

B

```
<< BASIC CONVERTER >>
--- CONVERTING ---
```

After completing conversion, the computer returns to the Main Menu. (When the size of the program to be converted is small, conversion will be completed almost instantly.)

If there is already a BASIC program in the program data area when a TEXT program is being converted to a BASIC program, or if there is a TEXT program in the text area when a BASIC program is being converted to a TEXT program, the computer will ask if you are sure you want to delete the existing program before beginning conversion.

B

```
<< BASIC CONVERTER >>
➔Basic←text   Text←basic
BASIC DELETE OK? (Y)
```

If Y is pressed, the computer will delete the existing BASIC program and begin conversion.

If any key other than Y is pressed, conversion will be canceled and the computer will return to the Main Menu.

The computer normally retains the original program after it is translated to another format. However, if there is insufficient free memory as a result of converting a program, the computer will ask if you are sure you want to delete the original program.

```
<< BASIC CONVERTER >>
--- CONVERTING ---
TEXT DELETE OK? (Y)
```

If Y is pressed, the computer will delete the original program as it is converted to the other format. The original program will be entirely deleted upon completion of conversion.

If any key other than Y is pressed, conversion will be canceled and the computer will return to the Main Menu.

Notes:

- When a password has been specified, the BASIC Converter function cannot be selected from the Main Menu. First clear the password in the RUN or PRO mode.
- The BASIC Converter does not delete the contents of the data area (variables) during conversion. Conversion may not be possible if there is insufficient free memory. Before starting conversion, ensure sufficient free memory by, for example, clearing variables from the data area with the CLEAR command.
- An error message (MEMORY OVER) will be displayed if the free memory is insufficient when the computer is executing conversion and deleting the original program. Once this error occurs, the program will be divided into converted and unconverted portions. It is therefore advisable to save the program to a cassette tape or other media before beginning translation. This error is more likely to occur when a program is converted from BASIC to TEXT format than from TEXT to BASIC.
- When converting a file from TEXT to BASIC, the computer does the conversion no matter what the contents of the original TEXT file are. So the exact contents of the original TEXT file may not be restored if it is converted to BASIC then back to TEXT again.
- If the converted file exceeds 255 characters (or 255 bytes), the overflow portion is ignored.

Example:

```
TEXT      10FORMULA
↓
BASIC     10:FOR_MULA
↓
TEXT      10FOR_MULA
```


12. MACHINE LANGUAGE MONITOR

Using the computer, you can write programs in machine language, as well as in BASIC. To assist you in programming in machine code, the computer has a machine language monitor (hereafter referred to as "the monitor"). The monitor is one of the system features that allows you to use a set of simple commands to input and output or execute programs written in machine code. This chapter explains the functions of the machine language monitor commands available on the computer.


The CPU of the computer is a Z80 microprocessor (CMOS Z80A equivalent), which is widely used in various high-performance 8-bit computers. A number of reference books about the Z80 processor are available. For details regarding the Z80 machine language, refer to one or more of these reference materials.

A data transfer cable such as the CE-T801 is required for execution of the R and W commands.

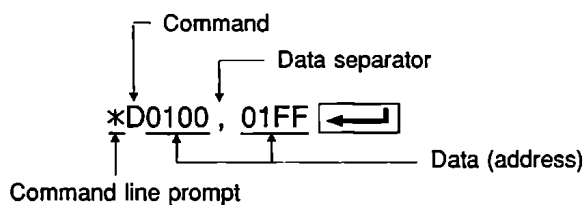
Using the Machine Language Monitor

Select the Monitor mode by entering MON  in the BASIC (RUN or PRO) mode, and the display will appear, as shown.

```
MACHINE LANGUAGE MONITOR
*
```

The asterisk (*) on the display is the command line prompt in the Monitor mode. Enter any command at the prompt. Any necessary addresses or data may be entered following the command. Press the  key at the end of the line to execute the command.

Example:



Notes:




- If a password has been set, the computer will be unable to enter the Monitor mode.
- All addresses and data must be in hex code.
- Use commas (,) to separate more than one address or piece of data.
- An error message (SYNTAX ERROR) will be displayed if hex code is not used or any symbol other than a comma is used as a data separator.
- Exit the Monitor mode by selecting any other mode, or turning the computer off and then on again.

- ① Machine language is extremely complex, and this often results in programming errors. When a machine language program containing errors is executed, it may destroy BASIC programs, data, or other contents of the computer memory. Therefore, it is strongly advised that you save your BASIC programs and other data to cassette tape or other media before executing a machine language program.
- ② When using the monitor, accessing any area other than the machine language area (reserved with the USER command) could destroy BASIC or TEXT programs or any other data within that area, or cause a malfunction. Be sure to use only the area reserved for machine code.

Machine Language Monitor Command Reference

USER — User Area

Purpose: Reserves a machine language area and displays the addresses of the reserved area.

Format:
 (1) USER01FF 
 (2) USER 
 (3) USER00FF 

Remarks:

- Format (1) reserves the memory address area of 0100H (the first address) to 01FFH (the last address) as the machine code area. The first address is automatically set to 0100H.

Reserved area →

```
* USER01FF
FREE: 0100-01FF
*
```

- Format (2) displays the range of addresses reserved as the machine code area.



```
* USER
FREE: 0100-01FF
*
```

“FREE: NOT RESERVED” will be displayed if no machine code area has been reserved.

- Format (3) deletes the existing machine code area from memory and displays the message “FREE: NOT RESERVED”.
- An error message (MEMORY ERROR) will be displayed if an illegal address area for the machine code area is entered.

S — Set Memory

Purpose: Updates the contents of memory.









Format: (1) S0100 
(2) S 

Remarks:

- Format (1) displays the contents of address 0100H (the first address) and prompts you to enter a new setting.




```
* S 0 1 0 0
0 1 0 0 : 1 0 -
```

Existing contents of memory

- To change the data, enter one byte of replacement data (2-digit hex), then press . The computer will then display the contents of the next address and prompt you for the data to be entered. When existing data does not need to be changed, press the  key without entering any data. The computer will then display the contents of the next address and prompt you for the data to be entered.
- A maximum of two digits of hex data can be entered. To cancel an entry before pressing the  key, press the  or  key.
- Press the  key to recall the contents of the preceding address, and the  key to recall the contents of the following address.
- Executing Format (2) recalls the contents of the address that is adjacent to the address that was last displayed with the S command.
- Press the  key to return to the command line prompt.

D — Dump Memory

Purpose: Dumps the contents of memory.

Format: (1) D0100 
(2) D 
(3) D0100, 01FF 

Remarks:






- Format (1) dumps 16 bytes of memory from the address area of 0100H (first address) to 010FH. (The dumped data is printed when in Print mode.)

Example:

```
First address of 16-byte segment → 0 1 0 0 : 3E 01 18 04 > . . .
Check sum → ( 1 D ) 3A 0F 01 3C : . . . <
                                     32 0F 01 C9 2 . . . 8
                                     31 00 00 00 1 . . .
```

ASCII code representations of dumped data are displayed here. Data 00H-1FH appear as periods (.), however.

Notes:

- The size of the area that can be dumped is fixed at XXX0H-XXXFH. The contents of the same segment will be dumped whenever an address within that 16-byte segment is specified. For example, if you specify address 0104H, the contents of the fixed 16-byte segment which includes that address, in this case 0100H-010FH, will be dumped.
- Press the  key to dump the preceding 16-byte segment, and the  key to dump the following 16-byte segment.
- Format (2) dumps the contents of the segment that is adjacent to the segment that was last dumped with the D command.
- If Format (3) is executed in Print mode, the computer will dump the contents of the specified area, 0100H (the first address)-01FFH (the last address), to the printer in 16-byte increments. When dumping is completed, the command line prompt will be displayed.
If Format (3) is executed in non-Print mode, the computer will dump the contents of the 16-byte segment beginning with address 0100H (the first address) to the screen. During screen-dump in non-Print mode, the last address specified is not acknowledged by the computer.
- To enter or exit the Print mode, use the P command (see below) or the  +  keys.
- Press the  key to return to the command line prompt.


Check Sum

Check sum refers to the sum of the values of a specific set of data. This sum is computed and assigned to a data set when that data set is written or dumped. The computer calculates the sum of the contents of a 16-byte segment dumped with the D command, and displays the low-order single byte of the sum as a check sum result.




For example, if you manually enter a machine code program, copying it from a printed program, you can check for errors in the contents of each of the 16-byte segments that are dumped by comparing the displayed check sum result with that for the original program. Note, however, that if your program contains more than one error, the check sum result may erroneously match that of the original program.

P — Print Switch

Purpose: Sets or clears the Print mode.

Format: P 


Remarks:

- The Print mode is alternately selected and deselected each time P  is entered. ("PRINT" will appear in the lower right corner of the display when the Print mode is selected.) The Print mode can also be selected and deselected using the  +  keys.

Note:

- The P command will not work if the printer is not connected or is not turned on.

G — GOSUB

- Purpose:** Executes a machine code program from a specified address.
- Format:** G0100 
- Remarks:**
- The G command is similar to the GOSUB command in BASIC. It executes a machine code program beginning with a specified address until a RET (Return) command is encountered. When the RET command is executed, the command line prompt will be displayed.

Note:

- Be sure to place a RET (Return) command at the end of your program or the program will not execute properly.

Runaway programs




A runaway program is one that is not executing properly and has gone out of control, so that the only way to stop the program is by resetting the system. In many cases, a runaway program will destroy the contents in memory, including the machine code programs, BASIC programs, and other data.

A machine code program can run away if it contains even a single minor bug. Therefore it is advised that any BASIC or other program stored in the computer be saved to cassette tape or other media before executing a machine code program.

Note:

- Press the RESET button to abort program execution.

BP — Break Point

- Purpose:** Sets the break point at a specified address.
- Format:**
- (1) BP011E 
 - (2) BP 
 - (3) BP0 
- Remarks:**
- Format (1) sets the break point at address 011EH. Up to two break points can be specified in one program by using Format (1) twice to specify two different addresses.
 - If you attempt to set a third break point, the break point that was first set will be cleared. Therefore, there cannot be more than two breakpoints in one program.
 - Be sure to set the break point at an instruction (OP code) address. If you set the break point at an operand address, the program will fail to read the breakpoint, and will not execute properly.
 - Format (2) displays the address of the break point. If no break point has been set, only the program line prompt (*) will appear on the following line.
 - Format (3) clears all existing break points.
 - A break point becomes invalid after it has been executed, so if a break point is specified in a program loop, it will be valid only during the first execution of the loop. However, it can be made valid again using the G command.

- The computer retains break points that were specified the last time the Monitor mode was used. When the computer is switched back into Monitor mode from another mode, you can make these break points valid by executing the G command.

Note:

- The contents of an address specified in the BP statement is temporarily replaced with "F7H" when the program is executed. If the RESET button is pressed before a break point specified in the program being run is encountered, the contents will remain "F7H". If this occurs, you will have to replace "F7H" with the original contents.

R — Read SIO

- Purpose:** Reads data through the serial I/O (SIO) port. This command is used for transferring machine code from a personal computer or other devices.
- Format:** (1) R (2) R0100
- Remarks:** The R command reads data in Intel Hex format through the SIO.
- Format (1) loads data that's been received into addresses specified in the received data.
 - Format (2) sequentially loads data that's been received into an area beginning with address 0100H.
 - When all data has been read, the address area into which the data was loaded will be displayed.
 - To abort data reading, press and hold the **BREAK** key until the command line prompt is displayed.

W — Write SIO

- Purpose:** Outputs data through the serial I/O (SIO) port. This command is used for transferring machine code to a personal computer or other devices.
- Format:** W0100, 01FF
- Remarks:**
- When you execute the format example above, the computer sends contents of the memory area in Intel Hex format from 0100H (the first address) to 01FFH (the last address) through the SIO port.
 - To abort data sending, press and hold the **BREAK** key until the command line prompt is displayed.

Note:

- If you connect a printer that's turned on to the peripheral interface connector (11-pin) and execute the W command, both the computer and printer may malfunction. If this occurs, turn the printer off, and press and hold the **BREAK** key until the command line prompt is displayed.

Error Messages in Monitor Mode

The following table lists error messages that can be displayed in the Monitor mode. Press the **CE** key to clear an error.

Error message	Description
SYNTAX ERROR	Illegal command syntax.
MEMORY ERROR	Attempt was made to reserve a machine code area outside the allowable area.
I/O DEVICE ERROR	Data read error or check sum error was detected during serial I/O operation.
OTHER ERROR	Other errors

13. ASSEMBLER

Reference

The following is a summary of technical terms that are often used in machine language programming:

Assemble:

To translate a source program expressed in an assembly language to a machine language. A translated machine code program is called an "object program" or simply "object".

Assembler:

A translation program used to translate a source program into an object program.

Generate:

To generate an object from mnemonic code.

Hand assemble:

To manually translate a source program without using an assembler.

Machine language:

A computer language directly interpreted and executed by a machine. Represented in hex code (internally processed in binary code).

Mnemonic codes:

Symbols designed to assist programmers in remembering instructions for machine code statements; for example, the abbreviation "ADD" for an addition command. A language whose mnemonic statements have specific one-to-one correspondence with machine code is called "assembly language".

Object program:

A fully assembled program that is ready to be loaded into the computer. It generally refers to a machine code program translated from a source program. Sometimes referred to simply as an "object". ("Object" may refer either to individual machine code resulting from translation or an entire machine language program.)

Pseudo instructions:

A set of assembler control commands that is not translated into object code. This set is used to specify an address area, to store object code, or to generate data.

Source program:

A program written in mnemonic code (assembly language). A machine code program is a translation of the source program.

Let's try Assembling

Before we go into the description of the assembler, let's try to assemble a sample program and execute the resulting object (machine code program).

Prior to this, however, you should remember some unusual events that might occur during program execution. If your machine code program has one or more errors, the following may occur:

① The computer executes the program in an endless loop, and all keys are inoperative.

To exit this loop, press the Reset button.

② Program displays random or abnormal characters or behaves strangely.

In some cases, the **BREAK** key will stop the program, but in others, the Reset button may have to be pressed.

③ Part or all of the program is destroyed or lost.

This is the result of a memory problem, and can extend to the source (TEXT) program, BASIC program, and/or all data in the computer, as well as to the machine code program itself.

These problems may occur singly or all together at one time. If any of these problems occurs and is severe enough that you cannot tell what state the computer is in, press the RESET button to clear the entire contents of memory.

- Problems ① and ② above are called "program runaway".

When you execute a machine code program, one or more of the problems described above could occur if the program has even a single error. It is therefore recommended that you save programs and data to a cassette tape or make a printed or written record before running a program. Note that, if a program in memory or other data is destroyed or lost as a result of a machine code program crashing, it cannot be restored.

Entering a Source Program

A source program is entered into the computer as a TEXT program (see page 99).

Press the **SHIFT** + **TEXT** keys to select the TEXT mode and display the Main Menu.

```
*** TEXT EDITOR ***
Edit   Del   Print Cmt
Sio    File  Basic
```

Press the **E** key at the menu to select the Edit function.

```
TEXT EDITOR
<
```

If a TEXT program already exists in the text area, press the **D** key for "Delete" at the Main Menu, then press **Y** to delete the existing program.

Sample Program

The following program loads hex numerals 20H through 9FH into the address area 0400H through 047FH (H suffix denotes hexadecimal notation):

Key operation	Display
10 TAB ORG TAB 0100H ←	10 ORG 0100H
20START: LD TAB A, 20H ←	20START:LD A, 20H
30 TAB LD TAB HL, 0400H ←	30 LD HL, 0400H
40LBL: TAB LD TAB (HL), A ←	40LBL: LD (HL), A
50 TAB INC TAB A ←	50 INC A
60 TAB INC TAB HL ←	60 INC HL
70 TAB CP TAB 0A0H ←	70 CP 0A0H
80 TAB JP TAB NZ, LBL ←	80 JP NZ, LBL
90 TAB RET ←	90 RET
100 TAB END ←	100 END

You may use the **SPACE** key instead of the **TAB** key to insert one or more spaces.

Description of Sample Program

- 10: (Load object into the area beginning with address 0100H.)
- 20: Load 20H into register A.
- 30: Load 0400H into register pair HL.
- 40: Load the contents of register A into the address specified by register pair HL.
- 50: Increment the value of register A by one and load the result into register A.
- 60: Increment the value of register pair HL by one and load the result into HL.
- 70: Compare the contents of register A with value A0H (subtract A0H from the contents of A).
- 80: If the result of the last operation is not zero (contents of register A are not A0H), jump to label LBL. The label will be translated into an address (0105H).
- 90: Return (return from subroutine)
- 100: (End of source program)

Lines 10 and 100 of this source program are called pseudo instructions. They are used for controlling the assembler, and are not translated into machine code (object) (see page 130).

After you've finished entering all the lines in the sample program, check for typing errors. Before you assemble the source program, you have to reserve the machine code area needed to store the object; otherwise it will not be possible to assemble the source program.

Reserving Machine Code Area

To reserve the machine code area, use the USER command in the Monitor mode (see page 112).

First select the Monitor mode.

BASIC MON **←**

```
MACHINE LANGUAGE MONITOR
*
```

Now reserve the machine code area, in this case from 0100H to 04FFH with the USER command.



USER 04FF 

The computer displays the reserved machine code area (user area).

```
MACHINE LANGUAGE MONITOR
*USER04FF
FREE:0100-04FF
*
```

Assembling the Source Program

You can now assemble the sample source program into machine code. Select the Assembler mode.

 + 

(The size of the work area may be different from that shown in this example. See page 127 for details.)

```
***** ASSEMBLER *****
user area=0100H-04FFH
work area=29221bytes
< Asm Display Print >
```

Press the  key to begin assembly.

```
***** ASSEMBLER *****

--- assembling ---
```

When assembly is completed, you will see the display, as shown at right (for an explanation of display information, see page 127).

```
object:0100H-010DH
size :000EH( 14)bytes
label : 2
error : 0 complete !
```

If an error occurs during assembly, the computer will display the relevant error message along with the number of the line in which the error occurred (see page 128).

```
***** ASSEMBLER *****
* FORMAT ERROR (1)
0105 ***** 40
LBL: LD HL),A
```

If this occurs, return to the Editor and correct the source program.


Checking the Generated Object Program

Check the generated object program with the Monitor. The object program is stored in the area from 0100H to 010DH.

To enter the Monitor mode, press the  key (or enter  MON .

```
MACHINE LANGUAGE MONITOR
*
```


Dump the object program with the D command.

D0100 

The computer displays the dumped object program, as shown.

0100	:	3E	20	21	00	>	!	.
(88)		04	77	3C	23	.	w	<#
		FE	A0	C2	05	.	1	.
		01	C9	00	00	.	8	.

(Part of the last contents of memory may be displayed in the area beginning with 010DH (C9).)

(88) is a check sum result (see page 114).

Executing the Object (Machine Code) Program

Now let's execute the generated object program. Use the Monitor's G (GOSUB) command.


Return to the Monitor's command line prompt.

BREAK

```
*

```


Execute the following G command to run the object program.

G0100 

When execution is completed, the Monitor's command line prompt will be displayed.

```
* G0100
*
```

Now check the results of program execution.

D0400 

0400	:	20	21	22	23	!	"	#
(78)		24	25	26	27	\$	%	&'
		28	29	2A	2B	()	*+
		2C	2D	2E	2F	,	-	/



Hex numerals 20H through 9FH have been written in the address area from 0400H through 047FH.

0410	:	30	31	32	33	0	1	2	3
(78)		34	35	36	37	4	5	6	7
		38	39	3A	3B	8	9	:	;
		3C	3D	3E	3F	<	=	>	?

Source Program Coding and Editing

The computer's assembler translates (assembles) the source program that's stored in the text area into object code. The assembled object code is sequentially loaded into a memory area beginning with the specified address.

In this section, you will look at the conventions and rules (entry format, etc.) used to create a source program.

Source Program Configuration

Each line of a source program normally contains a single statement. A program is usually made up of at least several of these lines. A source program written in assembly language begins with an ORG statement and ends with an END statement (the ORG and END statements can be omitted).

Example:

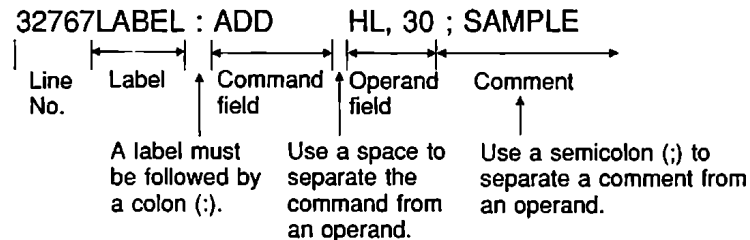
```
10   ORG    0100H
    {
100  END
```

- The ORG statement is used to specify the first address of the memory area where the generated object code is to be stored. In other words, the lines of the object program are sequentially stored in the area beginning with the address specified by the ORG statement. If the address is not specified, the computer assumes address 0100H as the first address.
- The END statement specifies the end of the source program. The computer ceases assembling when it encounters an END statement in the source program.

These statements are used for controlling the assembler, and themselves are not assembled into object code (see pseudo instructions on page 130).

Line (Statement) Configuration

Each line of a source program may consist of a line number, label, command, operand, comment, and/or pseudo instruction.



- A single line may have up to 254 characters including a comment.
- Lowercase alphabetic characters are processed the same as uppercase characters, except when they are used in operands or comments.

① Line numbers

A LINE NO. ERROR will occur if you specify a line number outside the range of 1 through 65279.

② Labels

- A label may be specified immediately following the line number (do not insert a space between the line number and the label; otherwise an error will occur).
- Up to six characters may be used to specify a label. If more than six characters are used, an error will occur (see page 134).
- The following characters may be used for labels:
 - Letters: A to Z (a to z are read as A to Z)
 - Numerals: 0 to 9
 - Symbols: [,], @, ?, and _

- Notice, however, that the first letter of a label must always be a letter or symbol character (if a numeral is used, it will not be distinguished from the line number).
- In labels, you cannot use exactly the same single or pair of characters as the register or conditional code names listed below:
 - Single registers: A, B, C, D, E, H, L, I, R
 - Paired registers: AF, BC, DE, HL, IX, IY, SP
 - Conditional code: NZ, Z, NC, C, PO, PE, P, M
- A label must be followed by a colon (:); otherwise an error will occur. An exception is when you define a value for a label with the EQU pseudo instruction, in which case the label need not be followed by a colon (see page 133).
- When you don't need a label, place one or more spaces between the line number and the following command word. You may use the `TAB` key instead of the `SPACE` key to insert spaces.

③ Command field (OP code)

- The command field accepts Z80 commands in mnemonic symbols. You can also enter any of the pseudo instructions listed on page 130 and a few subsequent pages in this field. A command in a statement is called an operation code or OP code.
- The entered command must be separated by one or more spaces from the operand that follows it. The `TAB` key can be used instead of the `SPACE` key to insert a space.

④ Operand field

Registers, addresses, and constants used in command (OP code) execution are called operands.

- Use commas (,) to separate operands.
- Each operand may have up to 32 characters.
- The following types of constants can be used in operands:

[Numerical constants]

Binary, decimal, and hexadecimal numbers:

Binary numbers: Represented by a string of ones and zeros, with a "B" suffix.

Example: 10111100B, 100000B

Decimal numbers: Represented by radix 0 to 9.

Example: 188, 32

Hexadecimal numbers: Represented by decimal radix 0 to 9 plus uppercase letters A through F, with an "H" suffix. When hex numbers begin with a letter, they must have a zero prefix to distinguish them from a command.

Example: 0BCH, 20H

[String constants]

Character strings in operands must be enclosed in single quotes ('). ASCII representations of characters serve as constants in operands.

Example:

(Specifications)	(Character strings)	(Constants)
'A'	A	41H
'AB'	AB	41H, 42H
'B"C'	B"C	42H, 27H, 43H
""D'	'D	27H, 44H
'E'''	E'	45H, 27H
""	'	27H
"	(NULL)	00H

[Label constants]

If you define a constant for a label with the EQU command, that label can be used as a constant in operands (see page 133).

- Expressions (including arithmetic operators) may be used in operands. You can use the following signs and arithmetic operators in operands. Note, however, that one operator does not have priority over another.
Signs: Positive (+), negative (−)
Operators: *, /, +, −
- The computer performs internal operations on 16-bit data. If an overflow occurs, the overflow is ignored (no error occurs). The object is generated by using the 8- or 16-bit result of operation.
- In statements containing expressions, the computer will not check the validity of entered expressions.

Examples:

LD A, 4142H → Read as LD A, 42H

DB 1234H → Read as DB 34H

⑤ Comments

Each line of a source program may be followed by a comment, separated by a semicolon (;). The portion of a line from the semicolon to the end of the line is regarded as a comment, and will not be translated into machine code (object).

Deleting a Source Program

Press the key for "Delete" at the Main Menu in the TEXT mode to select the Delete function. The computer asks if you are sure you want to delete the contents of the text area. (However, if no program is stored in the text area, nothing will happen when the key is pressed.)

TEXT DELETE OK? (Y)

Press the key, to delete the entire contents of the text area. The display then returns to the Main Menu of the TEXT mode. If any key other than is pressed, the display returns to the Main Menu of the TEXT mode without executing deletion.

Entering a Source Program

Press the key for "Edit" at the Main Menu in the TEXT mode to select the Edit function.

```
TEXT EDITOR
<
```

Press the or key to display any contents of the text area, such as a source program. If nothing is in the text area, the display will not change. A new program can be loaded into the text area after first deleting any previous contents. Press the key to return to the Main Menu, select the Delete function, and delete the contents of the text area by following the steps in the section above about deleting a source program.

To enter a source program:

- ① Enter a line number.
- ② When you don't need a label, press **TAB** to insert one or more separating spaces. The cursor will advance to the command entry field. (The **SPACE** key can be used instead of **TAB**.)
When you need a label, enter it immediately following the line number without inserting a space. Place a colon (:) at the end of the label. The colon may be followed by one or more spaces but this is not required.
- ③ Enter a command. If the command is to be followed by an operand, separate it from the command with one or more spaces using the **TAB** or **SPACE** key.
- ④ Input operands.
Use commas (,) to separate operands.
- ⑤ When you want to add a comment to the line, enter the comment following a semicolon (;).
- ⑥ When the entire line has been entered, press **←** to register the line to memory. The cursor will disappear when you press **←**.

To enter the next line, repeat the steps above. (See the entry example on page 120.)

Assembling

In this section, you will take a closer look at how to assemble a source program entered in the TEXT mode. The following explanation assumes that the sample program shown on page 120 is already loaded in the computer. If not, load it before continuing to the next section.

Menu in the Assembler Mode

To assemble a source program, you must first select the Assembler mode.

SHIFT + **ASMBL**

The Assembler mode menu as shown at the right appears.

```
***** ASSEMBLER *****
user  area=0100H-04FFH
work  area=29221bytes
< Asm  Display  Print >
```

{ User area (machine code area): addresses 0100H to 04FFH
Size of work area: 29221 bytes

- A** executes assembly.
- D** displays an assembly list.
- P** prints an assembly list.

- The menu shows the reserved machine code area on the second line. To reserve a machine code area, use the USER command in the Monitor mode. If a machine code area is not reserved or is too small to store the object, an error message (NOT RESERVED or USER AREA OVER) will display during assembling. If this occurs, enter: **BASIC** MON **←** to select the Monitor mode, and reserve or increase the machine code area with the USER command.

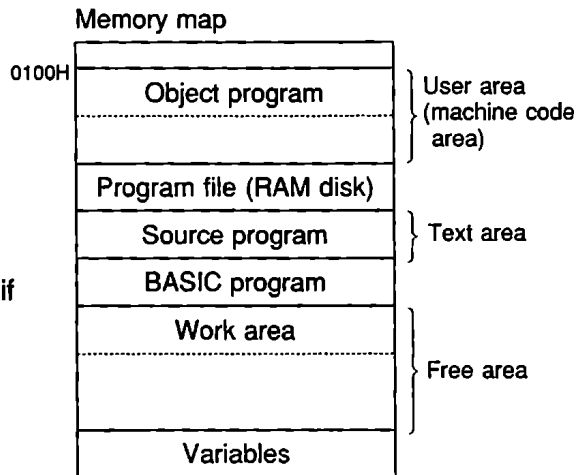
- The third line of the ASSEMBLER display shows the size of the available work area in bytes. This information indicates the byte count of the free area available in memory, and matches the byte count obtained with the FRE command in BASIC.

The work area, used for actual assembly operations within the computer, is automatically reserved in the free area. If the work area cannot be reserved, an error message (WORK AREA OVER) will be displayed. If this occurs, increase the free area by, for example, deleting an existing BASIC program or array variables, or reducing the machine code area.

Note:

An error will occur if a free area of at least 307 bytes is not reserved when the computer enters the Assembler mode.

When a source program contains labels, the assembler will set aside a label work area of the necessary size during assembling. An error will occur if the assembler is unable to set aside the necessary work area.



Executing Assembly

Successful assembly

Press the **A** key for "Asm" at the Assembler menu to begin assembly.

```
***** ASSEMBLER *****
--- assembling ---
```

While assembly is executing, "--assembling--" is displayed. When assembly is completed, a final display with "complete !" will appear, and will include the object area, its size, the number of labels, and error count.

```
object: 0100H-010DH
size   : 000EH ( 14) bytes
label  : 2
error  : 0 complete !

Object area: 0100H to 010DH
Size of object: 0EH (14) bytes
Number of labels: 2
Error count: 0
```

Press the **C•CE** key at the "complete !" display to select the Monitor mode. In the Monitor mode, you can check the assembled object program with the D command, or execute it with the G command.

Unsuccessful assembly

If an error is encountered in the source program when it is being assembled, the assembler will suspend assembly and display the relevant error message along with the number of the line in which the error was found. Press the **↓** key to continue assembly.

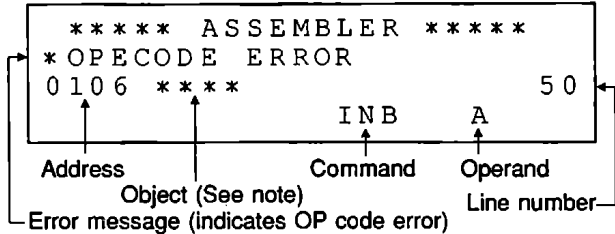
Example:

Assume that the sample program given on page 120 contains errors on lines 50 and 80, as shown below.

50	INB	A"INC A" is correct.
80	JP	NZ, KBL"JP NZ, LBL" is correct.

Press the **A** key at the Assembler menu to assemble the sample program containing the above errors.

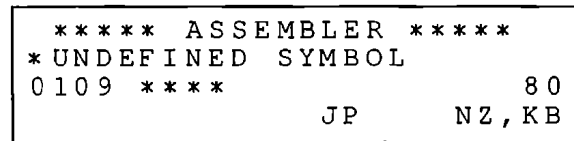
When the first error is encountered, the error message shown at right is displayed.



Note:

If the assembler fails to generate the correct object program due to an error in the source code, an asterisk string (****) will be displayed following the address.

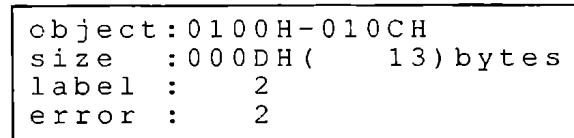
Press the **↓** key to resume assembly, and the error message for the second error on line 80 will be displayed.



Error message (an undefined symbol has been used in the label)

Press the **↓** key again.

The assembler final display will appear, this time without the "complete !" message.



Press the **C+CE** key at the final display to return to the Assembler menu.

Notes:

- The assembler ignores the statement on line 50 and assumes that the label on line 80 specifies address 0000H, at which point it assembles the program in this example.
- If an error is detected in the source program, the generated object will not be a correct program. If you attempt to execute the object, it may cause program runaway or damage contents of memory. Be sure to correct the source program and reassemble it, so you can execute an error-free object.

Displaying an Assembly List

The Display option in the Assembler menu lets you examine an assembly list before you actually assemble your source program. The assembly list contains the object code to be generated, its address, and other object information.

Press the **D** key for "Display" at the Assembler menu to display the first line of an assembly list. You can sequentially view the subsequent lines with the **↓** key. Load the sample source program shown on page 120 if it is not in your computer yet, and examine the assembly list for it.

Press the **D** key at the Assembly menu.

**** ASSEMBLE LIST ****		
0100		10
	ORG	0100H
0100	3E20	20
Address	Object	Line number
Source program		

When the object cell is left blank, there is no object code to be generated. If an object code exceeds eight digits, the remaining digits will be displayed on the next line.

Press the **↓** key repeatedly to view subsequent lines of the assembly list.

**** ASSEMBLE LIST ****		
0100		10
	ORG	0100H
0100	3E20	20
	START: LD	A, 20H
0102	210004	30
	LD	HL, 04
	00H	
0105	77	40
	LBL: LD	(HL),
	A	
0106	3C	50
	INC	A
0107	23	60
	INC	HL
0108	FEA0	70
	CP	0A0H
010A	C20501	80
	JP	NZ, LB
	L	
010D	C9	90
	RET	
010E		100
	END	
**** SYMBOL TABLE ****		
Symbol table(*)	START : 0100	LBL : 0105
Object address	object : 0100H-010DH	
Object size	size : 000EH (14) bytes	
Number of labels	label :	2
Error count	error :	0 complete !

* The symbol table lists the values assigned to labels in hex.

Notes:

- Press the **C•CE** key to return from the assembly list to the Assembler menu.
- The Display function only lets you examine the assembly list, but will not load the object code in the list into the machine code area. To load it, you have to actually assemble the source program.

Printing an Assembly List

The Print option from the Assembler menu lets you print an assembly list. Connect the optional CE-126P printer to your computer, turn the printer on, and press the

P key at the Assembler menu.

Notes:

- If you choose the Print option when the CE-126P printer is not connected or is turned off, an error message (*PRINTER ERROR) will be displayed. If this occurs, clear the error with the **C•CE** key, and check your printer.
- You can print an assembly list whether or not the PRINT indicator is on in the lower right corner of the display.
- When printing is completed, the assembler shows the assembler's final screen. Press the **C•CE** key to return to the Assembler menu.
- The list is printed in the same format that it's displayed.
- To abort printing, press and hold the **BREAK** key until printing stops. "--- break ---" will appear on the display. Press the **C•CE** key to return to the Assembler menu.

Pseudo Instructions of the Assembler

Pseudo instructions are used to control the assembler itself and are not translated into object code. The computer has the following pseudo instructions:

- **ORG**: Specifies the first address of object load area.
- **DEFB/DB/DEFM/DM, DEFS/DS and DEFW/DW**: Define data in operands.
- **EQU**: Defines label values.
- **END**: Specifies the end of assembly.

The following summarizes conventions and rules used in explanations of the format of these pseudo instructions.

- Expression:** Expressions may be numerals, formulas, labels, or "strings".
- Formula:** Formulas may be numerals, labels, and any arithmetic expressions using numerals and labels.
- { }:** When several elements are stacked within braces in a statement, one of these elements must be selected.
- []:** An element within brackets in a statement is optional.
- []...:** A horizontal ellipsis that follows brackets indicates that an element within brackets is optional or may be repeated.

ORG — Origin

- Format:** ORG expression
- Purpose:** Specifies the first address of an object load area.
- Remarks:**
- Expression specifies the first address of the area where the generated object is to be stored. In other words, the lines of the object program are sequentially stored in the area beginning at the first address that is specified by the expression.
 - If the source program has no ORG statement, the assembler assumes "ORG 100H", which makes address 100H the first address from which the object is stored.

Example:

```
ORG 0400H
```

This statement stores the object into the area that begins with address 0400H.

DEFB/DB/DEFM/DM — Define Byte/Define Message

Format: $\left. \begin{array}{l} \text{DEFB} \\ \text{DB} \\ \text{DEFM} \\ \text{DM} \end{array} \right\} \text{expression [,expression]...}$

Purpose: Translates a string or low-order single byte of a numeral given by the expression into object code.

Remarks: • These statements translate the low-order single byte of a numeral given in the expression into object code.

Examples:

DEFB 1234H; Translates 1234H into object code "34H".

DB 1234; Translates 1234 into object code "D2H".

- A character string in the operand must be enclosed in single quotes ('). Up to 32 characters may be used in a string. Individual characters in the operand string are translated into corresponding ASCII codes.

Example:

DEFM 'DATA'; Translates individual characters in string 'DATA' into object codes 44H, 41H, 54H, and 41H.

- Use commas (,) to separate operands.

Example:

DB 32*4+5, 'X2'; 85H, 58H, and 32H are generated in the object.

Sample Program

	Source code	Object code
10	ORG 0100H	
20	LD HL, DATA	21 0C 01
30	LD DE, 300H	11 00 03
40	LD BC, 5	01 05 00
50	LDIR	ED B0
60	RET	C9
70	DATA: DB 'ABCDEFGH'	41 42 43 44
		45 46 47 48
80	END	

Individual characters in the operand string on line 70 are translated into corresponding ASCII code.

This sample program copies five bytes of data located in the area whose first address is specified by label DATA, into the area beginning with address 300H. In short, it copies data 41H, 42H, 43H, 44H, and 45H into address area 300H to 304H.

DEFW/DW — Define Word

Format: [label:] $\left\{ \begin{array}{l} \text{DEFW} \\ \text{DW} \end{array} \right\}$ expression [,expression]...

Purpose: Translates the low-order two bytes of a numeral or a string (two characters or less per data) given in the expression into object code.

Remarks:

- These statements translate the low-order two bytes of a numeral specified in the operand.

Examples:

DW 1234H; Translates 1234H into object codes 34H and 12H (in the order of low- and high-order bytes).

DEFW 34H; Translates 34H into object codes 34H and 00H.

- A character string in the operand must be enclosed in single quotes ('). You can specify up to two characters for a string.

Examples:

DEFW 'DA'; Translates string 'DA' into object codes 41H and 44H.

DW 'Z'; Translates string 'Z' into object codes 5AH and 00H.

- Use commas (,) to separate operands.

Example:

DW 'AB','CD',5678H; 42H, 41H, 44H, 43H, 78H, and 56H are generated in the object.

DEFS/DS — Define Storage

Format: [label:] $\left\{ \begin{array}{l} \text{DEFS} \\ \text{DS} \end{array} \right\}$ numeric expression

Purpose: Generates null codes (00H) by the number specified in the operand.

Remarks:

- These statements generate null codes (00H) by the specified byte count.

Example:

DS 12; Generates 12 bytes of 00H.

Sample Program

	Source code	Object code
10	ORG 0100H	
20	LD HL, DATA	21 10 01
30	LD DE, 300H	11 00 03
40	LD BC, 5	01 05 00
50	LDIR	ED B0
60	RET	C9
65	DS 4	00 00 00 00
70	DATA: DB 'ABCDEFGH'	41 42 43 44
		45 46 47 48
75	NXT00: DS 500H-NXT00	(00H is placed in all the subsequent addresses to 04FFH.)
80	END	

This sample program is identical to the one shown on page 131, except for inclusion of additional lines 65 and 75. Line 65 reserves a memory area for future use. Line 75 places null codes (00H) to clear unnecessary contents in memory.

00H is a no operation (NOP) code that instructs the computer not to do anything.

EQU — EQU

Format: label EQU expression

Purpose: Assigns the value specified by the operand to a label.

Remarks:

- This statement assigns the value given by the expression to the label.
- The value may be a one- or two-byte numeral or string.
- The label should not be followed by a colon (:).

Example:

START EQU 1000H; Assigns value 1000H to label START, which can then be treated as constant 1000H.

OK EQU 'Y'; Assigns value 59H to label OK.

END — End

Format: END

Purpose: Declares the end of the source program.

Remarks:

- The END statement specifies the end of a source program at which point assembly terminates. Anything following this statement will not be assembled.
- If no END statement is placed at the end of a source program, the assembler assembles to the last contents of the text area.

Assembly Errors

This section contains a list of the error messages that can occur during assembly, as well as the meanings of these messages. Use the **C•CE** key to clear an error message. If assembly is suspended upon encountering an error in the source program, press the **↓** key to continue assembly.

Errors are also cleared when another mode is selected.

Error	Meaning (cause)
OPECODE ERROR	Invalid OP code (command code).
FORMAT ERROR (1)	Invalid operand separator.
FORMAT ERROR (2)	Invalid code (ASCII codes 01H-1FH, etc.) or character is in operand (such codes or characters cannot be entered through normal operation, however).
FORMAT ERROR (3)	Invalid number of operands.
FORMAT ERROR (4)	Invalid characters used in the label.
FORMAT ERROR (5)	A label has more than six characters.
FORMAT ERROR (6)	String in operand is not enclosed in single quotes.
FORMAT ERROR (7)	The number of characters used in statement or individual operand exceeds 32 (e.g. value of address, etc. in operand has too many leading zeros).
QUESTIONABLE OPERAND (1)	Invalid operand.
QUESTIONABLE OPERAND (2)	Invalid condition (NZ, Z, NC, etc.).
QUESTIONABLE OPERAND (3)	Operand value exceeds the allowable limit.
QUESTIONABLE OPERAND (4)	String in operand exceeds length of 32 characters.
QUESTIONABLE OPERAND (5)	Division by zero was attempted.
QUESTIONABLE OPERAND (6)	Other invalid values or expressions.
UNDEFINED SYMBOL	An undefined symbol (label) is used.
MULTI DEFINE SYMBOL	The same symbol (label) is defined more than once.
FILE NOT EXIST	Program to be assembled does not exist in text area.
USER AREA OVER	Object could not be loaded into the machine code area. (The first address of the object area specified by the ORG statement is outside the machine code area, or the object has overflowed the machine code area as it was loaded.)
WORK AREA OVER	The size of free area is too small for the work area needed for assembly (when the computer enters the Assembler mode or is assembling).
PRINTER ERROR	Printer is not ready or not functioning. (The printer is not connected, turned off, or is inoperative due to dead printer battery.)

BASIC REFERENCE

Part 5 contains alphabetical listings of all the BASIC commands supported by the PC-E220*. This chapter is designed to be used as a ready reference.

The first section contains an alphabetical listing of numeric functions and pseudo variables.

The second section is an alphabetical listing of all other BASIC commands.

* The PC-E220 is hereafter referred to as "the computer".

14. SCIENTIFIC & MATHEMATICAL CALCULATIONS

The computer has a wide range of built-in functions for scientific, mathematical and statistical calculations. They are listed below alphabetically. All the functions listed below can be used as part of calculations when using the computer in RUN mode. They may also be used as BASIC commands within programs.

For trigonometric functions, entries can be made in degrees, radians or as a gradient value, as appropriate:

DEGREE: Set the computer to degree entry mode by typing DEGREE (the status line on the display shows DEG). This is the default mode.

RADIAN: Set the computer to radian entry mode by typing RADIAN (the status line on the display shows RAD).

GRADIENT: Set the computer to gradient entry mode by typing in GRAD (the status line on the display shows GRAD).

These three modes (DEG, RAD, and GRAD) can also be set from within a program. Once a mode is set, all entries for trigonometric functions must be in the units set (degrees, radians, or gradient values) until another mode is selected either manually or from within a program. The examples given below are all for direct entry of the functions entered in degrees.

Most functions can also be implemented by pressing the corresponding function key. Functions marked with an asterisk (*) have no corresponding key and must be entered through the keyboard.

***ABS**

|x|

Function: Absolute value**Remarks:** Returns the absolute value of the numeric argument. The absolute value is the magnitude of the number irrespective of its sign. ABS-10 is 10.**Example:** ABS -10 $\boxed{\leftarrow}$ 10**ACS** $\cos^{-1}x$ **Function:** Inverse or arc cosine**Remarks:** Returns the arc cosine of the numeric argument. The arc cosine is the angle whose cosine is equal to the argument. The value returned depends on the mode (DEG, RAD or GRAD).**Example:** DEGREE $\boxed{\leftarrow}$
ACS -0.5 $\boxed{\leftarrow}$ 120**AHC** $\cosh^{-1}x$ **Function:** Inverse hyperbolic cosine**Remarks:** Returns the inverse hyperbolic cosine of the numeric argument.**Example:** AHC 10 $\boxed{\leftarrow}$ 2.993222846**AHS** $\sinh^{-1}x$ **Function:** Inverse hyperbolic sine**Remarks:** Returns the inverse hyperbolic sine of the numeric argument.**Example:** AHS 27.3 $\boxed{\leftarrow}$ 4.000369154**AHT** $\tanh^{-1}x$ **Function:** Inverse hyperbolic tangent**Remarks:** Returns the inverse hyperbolic tangent of the numeric argument.**Example:** AHT 0.7 $\boxed{\leftarrow}$ 0.867300527

ASN $\sin^{-1}x$

Function: Inverse or arc sine

Remarks: Returns the arc sine of the numeric argument. The arc sine is the angle whose sine is equal to the argument. The value returned depends on the mode (DEG, RAD or GRAD).

Example: DEGREE
ASN 0.5 30

ATN $\tan^{-1}x$

Function: Inverse or arc tangent

Remarks: Returns the arc tangent of the numeric argument. The value returned depends on the mode (DEG, RAD or GRAD).

Example: DEGREE
ATN 1 45

COS $\cos x$

Function: Cosine

Remarks: Returns the cosine of the angle argument. The value returned depends on the mode (DEG, RAD or GRAD).

Example: DEGREE
COS 120 -0.5

***CUB** x^3

Function: Cube

Remarks: Returns the cube of the argument.

Example: CUB 3 27

CUR $\sqrt[3]{x}$

Function: Cube root

Remarks: Returns the cube root of the argument.


Example: CUR 125 5

DEG

$dd^{\circ}mm'ss'' \rightarrow ddd.dddd^{\circ}$

Function: Deg/min/sec to decimal conversion

Remarks: Converts an angle argument in DMS (Degrees, Minutes, Seconds) format to DEG (Decimal Degrees) format. In DMS format the integer portion of the number represents degrees, the first and second digits after the decimal point represent minutes, the third and fourth digits after the decimal point represent seconds, and any further digits represent fractional seconds.

Example: DEG 30.5230  (30°52'30") 30.875

DMS

$ddd.dddd^{\circ} \rightarrow dd^{\circ}mm'ss''$

Function: Decimal to deg/min/sec conversion


Remarks: Converts an angle argument in DEG format to DMS format (see DEG).

Example: DMS 124.8055  124.48198 (124°48'19"8)

EXP

e^x

Function: Exponential function

Remarks: Returns the value of e (2.718281828... the base of natural logarithms) raised to the value of the numeric argument.
The corresponding function key is .

Example: EXP 1.2  3.320116923

FACT

$n!$

Function: Factorial n

Remarks: Returns the factorial of the argument.

Example: FACT 7  5040

HCS

$\cosh x$

Function: Hyperbolic cosine

Remarks: Returns the hyperbolic cosine of the numeric argument.

Example: HCS 3  10.067662

HSN**sinh x****Function:** Hyperbolic sine**Remarks:** Returns the hyperbolic sine of the numeric argument.**Example:** HSN 4 


27.2899172

HTN**tanh x****Function:** Hyperbolic tangent**Remarks:** Returns the hyperbolic tangent of the numeric argument.**Example:** HTN 0.9 

0.71629787

INT*Function:** Integer**Remarks:** Returns the integer portion of the argument. The integer portion of PI is 3.**Example:** INT -1.9 

-2

LN**log_ex****Function:** Natural or Naperian logarithm**Remarks:** Returns the logarithm to base e (2.718281828 ...) of the numeric argument.**Example:** LN 2 

0.69314718

LOG**log₁₀x****Function:** Common logarithm**Remarks:** Returns the logarithm to base 10 of the numeric argument.**Example:** LOG 1000 

3

***NCR**

$${}_n C_r = n! / r!(n-r)!$$

Function: Combination

Remarks: Enter the values as NCR(n,r).

Example: NCR (6,3)

20

***NPR**

$${}_n P_r = n! / (n-r)!$$

Function: Permutation

Remarks: Enter the values as NPR(n,r).

Example: NPR (6,3)

120

PI

π

Function: PI

Remarks: PI is a numeric pseudovariable that has the value of π . Use of PI is identical to use of the key.

Example: PI

3.141592654

POL

$$(x,y) \rightarrow (r,\theta)$$

Function: Rectangular to polar coordinate conversion

Remarks: Converts numeric arguments of rectangular coordinates to their polar coordinate equivalents.

The first argument indicates the distance from the y-axis and the second the distance from the x-axis. The values converted indicate the distance from the origin and the angle in the polar coordinates, and are assigned to the fixed variables Y and Z, respectively. The angle depends on the mode (DEG, RAD, or GRAD).

Example: DEGREE
POL (8,6)
Z

10 (r = 10)
36.86989765
($\theta \approx 36.9^\circ$)

^ **y^x**

Function: xth power

Remarks: Returns the xth power of the numeric argument. Enter as y ^ x.

Example: 4 ^ 2.5 32

RCP **1/x**

Function: Reciprocal

Remarks: Returns the reciprocal of the numeric argument.

Function: RCP 4 0.25

REC **(r,θ) → (x,y)**

Function: Polar to rectangular coordinate conversion

Remarks: Converts numeric arguments of polar coordinates to their rectangular coordinate equivalents.

The first argument indicates the distance from the origin and the second argument the angle. The angle depends on the mode (DEG, RAD or GRAD). The converted values indicate the distances from the y-axis and the x-axis, and are assigned to the fixed variables Y and Z, respectively.

Example: DEGREE
REC (12,30) 10.39230485 (x ≈ 10.4)
Z 6 (y = 6)

***RND**

Function: Random number

Remarks: See RND and RANDOMIZE in the BASIC COMMAND DICTIONARY.

ROT **^x√y**

Function: xth root

Remarks: Returns the xth root of the argument y. Enter as yROTx.

Example: 7776 ROT 5 6

*SGN

Function: Sign of argument

Remarks: Returns a value based on the sign of the argument.

If $x > 0$, the function returns 1.
If $x < 0$, the function returns -1.
If $x = 0$, the function returns 0.

SIN

$\sin x$

Function: Sine

Remarks: Returns the sine of the angle argument. The value returned depends on the mode (DEG, RAD or GRAD).

Example: DEGREE
SIN 30 0.5

SQR

\sqrt{x}

Function: Square root

Remarks: Returns the square root of the argument.

Example: SQR 3 1.732050808

SQU

x^2

Function: Square

Remarks: Returns the square of the argument.

Example: SQU 4 16

TAN

$\tan x$

Function: Tangent

Remarks: Returns the tangent of the angle argument. The value returned depends on the mode (DEG, RAD or GRAD).

Example: DEGREE
TAN 45 1

TEN

10^x

Function: Antilogarithm

Remarks: Returns the value of 10 (the base of the common log) raised to the value of the numeric argument.

Example: TEN 3  1000

&H

Function: Hexadecimal to decimal conversion

Remarks: Converts a hexadecimal value to its decimal equivalent.

Example: &H F82  3970

Calculation Ranges

Numerical Calculations:

For a calculation involving x, the number x must be within one of the ranges below:

$$-1 \times 10^{100} < x \leq -1 \times 10^{-99} \text{ for negative } x$$

$$10^{-99} \leq x < 10^{100} \text{ for positive } x$$

$$x = 0$$

The displayed value of x is limited by the number of digits that can fit on the display screen.

Functions:

Function	Range of x
sin x cos x tan x	DEG: $ x < 1 \times 10^{10}$ RAD: $ x < \frac{\pi}{180} \times 10^{10}$ GRAD: $ x < \frac{10}{9} \times 10^{10}$ Also, for tan x only: (n=integer) DEG: $ x \neq 90(2n - 1)$ RAD: $ x \neq \frac{\pi}{2}(2n - 1)$ GRAD: $ x \neq 100(2n - 1)$
$\sin^{-1} x$ $\cos^{-1} x$	$-1 \leq x \leq 1$
$\tan^{-1} x$	$ x < 1 \times 10^{100}$
sinh x cosh x tanh x	$-227.9559242 \leq x \leq 230.2585092$
$\sinh^{-1} x$	$ x < 1 \times 10^{50}$
$\cosh^{-1} x$	$1 \leq x < 1 \times 10^{50}$
$\tanh^{-1} x$	$ x < 1$
ln x log x	$1 \times 10^{-99} \leq x < 1 \times 10^{100}$
e^x	$-1 \times 10^{100} < x \leq 230.2585092$
10^x	$-1 \times 10^{100} < x < 100$
$\sqrt[3]{x}$	$ x < 1 \times 10^{100}$
$\frac{1}{x}$	$ x < 1 \times 10^{100}, x \neq 0$
x^2	$ x < 1 \times 10^{50}$
\sqrt{x}	$0 \leq x < 1 \times 10^{100}$
n!	$0 \leq n \leq 69$ (n=integer)
DMS→DEG DEG→DMS	$ x < 1 \times 10^{100}$

Function	Range of x
y^x ($y^x = 10^{x \cdot \log y}$)	<p>when $y > 0$, $-1 \times 10^{100} < x \log y < 100$ when $y = 0$, $x > 0$ when $y < 0$, $\left\{ \begin{array}{l} x = \text{integer or } \frac{1}{x} = \text{odd integer (} x \neq 0) \\ \text{and } -1 \times 10^{100} < x \log y < 100 \end{array} \right.$</p>
$\sqrt[x]{y}$ ($\sqrt[x]{y} = 10^{\frac{1}{x} \log y}$)	<p>when $y > 0$, $-1 \times 10^{100} < \frac{1}{x} \log y < 100$, $x \neq 0$ when $y = 0$, $x > 0$ when $y < 0$, $\left\{ \begin{array}{l} x \text{ or } \frac{1}{x} \text{ must be non-zero integer,} \\ \text{and } -1 \times 10^{100} < \frac{1}{x} \log y < 100 \end{array} \right.$</p>
&H	$0 \leq x \leq 2540BE3FF$ (x in hexadecimal) $FDABF41C01 \leq x \leq FFFFFFFF$
$x, y \rightarrow r, \theta$	$(x^2 + y^2) < 1 \times 10^{100}$ $\left\{ \begin{array}{l} r = \sqrt{x^2 + y^2} \\ \theta = \tan^{-1} \frac{y}{x} \end{array} \right.$ $\frac{y}{x} < 1 \times 10^{100}$
$r, \theta \rightarrow x, y$	$r < 1 \times 10^{100}$ $x = r \cos \theta$ $ r \sin \theta < 1 \times 10^{100}$ $y = r \sin \theta$ $ r \cos \theta < 1 \times 10^{100}$
nPr	$0 \leq r \leq n < 10^{100}$ n, r integers
nCr	$0 \leq r \leq n < 10^{100}$ n, r integers when $n - r < r$, $n - r \leq 69$ when $n - r \geq r$, $r \leq 69$

15. BASIC COMMAND DICTIONARY

This chapter contains an alphabetical listing of the BASIC commands that you can use on the computer.

For simplicity, the following conventions have been adopted in compiling this dictionary:

- expression** Indicates a numeric value, numerical variable or a formula, including numeric values and numerical variables.
- variable** Indicates a numerical variable or string variable, including array variables.
- “string”** Indicates a character string enclosed in quotation marks.
- string variable** Indicates a string variable or string array variable.
- *label** Indicates *label.
(Although both *label and “label” forms may be used with the computer, *label is recommended. *AB and “AB” are recognized as the same labels.)
- []** The parameter in square brackets is optional. The brackets themselves are not part of the command entry.
- ()** Used to enclose parameter values in certain commands. They should be entered as part of the command.
- “ ”** Used to enclose string parameter values in certain commands.
- { A }
 { B }** A or B can be selected.
- P** Program execution is possible.
- D** Direct input operation is possible.
- Abbreviation** Most of the commands can be abbreviated.
The shortest abbreviations allowed are given in this manual.
Example:
 Abbreviation: P. (for PRINT)
 The following abbreviations are also valid:
 PR. PRI. PRIN.

ASC

P
D

FORMAT: ASC { "string"
 string variable }

Abbreviation: AS.
See Also: CHR\$

PURPOSE:

Returns the character code for the first character in the specified string.

REMARKS:

Specify the string as the contents of a string variable in the form X\$ or as an actual string enclosed in quotes, "XXXX". Only the character code of the first character in the string is returned. See Appendix C for character code tables.

EXAMPLE:

```
10: INPUT "ENTER A CHARACTER ";A$
20: N = ASC A$
30: PRINT "THE CHARACTER CODE IS ";N
40: GOTO 10
```

- [10] The user presses a key to enter a character.
- [20] ASC finds the code number for the character.
- [30] The answer is displayed.
- [40] Repeats until the user halts the program by pressing the **BREAK** key.

BEEP

P
D

FORMAT: BEEP expression

Abbreviation: B.
See Also:

PURPOSE:

The BEEP verb is used to produce an audible tone.

REMARKS:

The BEEP verb causes the computer to emit one or more audible tones. The number of beeps is determined by the expression, which must be numeric (positive number less than 65535). 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.

CHR\$

P
D

FORMAT: CHR\$ expression

Abbreviation: CH.

See Also: ASC

PURPOSE:

Returns the character that corresponds to the numeric character code of the parameter.

REMARKS:

See Appendix C for a chart of character codes and their relationship to characters, e.g., CHR\$ 65 is "A".

A hexadecimal number can be specified with "&H" in front of the character code (eg. A\$ = CHR\$ &H5A)

A value greater than 255 generates an error.

EXAMPLE:

```
10: AA$=""
20: INPUT "CODE=" ;A:CLS
30: AA$=AA$+CHR$A
40: LOCATE 7,1:PRINT AA$
50: GOTO 20
```

Displays the characters represented by the codes entered in line 20.

CLEAR

P
D

FORMAT: CLEAR

Abbreviation: CL.

See Also: DIM

PURPOSE:

Erases variables that have been used in the program and resets all preallocated variables to zero or null.

REMARKS:

CLEAR frees memory space that has been used to store simple numeric variables, and array variables secured using the DIM statement. Also use CLEAR at the beginning of a program to clear memory occupied by variables from previously run programs if several programs are in memory.

Do not use the CLEAR command within a FOR...NEXT loop.

EXAMPLE:

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

```
20: CLEAR
```

[20] Frees memory assigned to C() and resets A to zero.

CLOAD

D

FORMAT: 1. CLOAD "filename" 
2. CLOAD 

Abbreviation: CLO.

See Also: CLOAD?, CSAVE

PURPOSE:

Loads a program saved on tape.


REMARKS:

Valid only as direct input in the PRO or RUN mode.

Format 1 clears the memory of an existing program, searches the tape for the program indicated by "filename", and loads the program.

Format 2 clears the memory and loads the first program stored on tape, starting at the current position.

During execution, "*" is displayed at the bottom right after the file name is displayed. After execution, "*" disappears and the prompt (>) will be displayed. When searching for a filename, "*" is not displayed. The same applies to the CLOAD? command.

If the specified filename is not found, the computer continues to search for the filename even after the end of the tape has been reached. Press the  key to stop searching for the filename.

If an error occurs during execution of the CLOAD command, the program being loaded will be invalid.

See tape operation in the section describing peripherals.

EXAMPLE:

CLOAD*



CLOAD "PRO3"***

* Loads the first program found on the tape.

** Searches the tape for the program "PRO3" and loads it.

CLOAD?

D

FORMAT: 1. CLOAD? "filename" 
2. CLOAD? 

Abbreviation: CLO.?

See Also: CLOAD, CSAVE

PURPOSE:

Compares a program saved on tape with one stored in memory.

REMARKS:

Valid only as direct input in the PRO or RUN mode.

To verify that a program was saved correctly, rewind the tape to the beginning of the program and use the CLOAD? command.

Format 1 searches the tape for the program indicated by "filename" and then compares it to the program stored in memory.

Format 2 compares the program stored in memory with the first program stored on the tape, starting at the current tape position.

When the program on tape does not match the one stored in memory, an error will occur. During execution, "*" is displayed at the bottom right after the file name is displayed. After execution, "*" disappears and the prompt (>) will be displayed.

Note:

When loading a program created with another computer and stored on tape, the program is converted to PC-E220 code. In this case, the CLOAD? command cannot be executed.

For an explanation of tape operation, see the relevant section in the peripherals chapter.

EXAMPLE:

```
CLOAD?*  
CLOAD?*PRO3***
```

* Compares the first program found on the tape with the one in memory.

** Searches the tape for the program "PRO3" and compares it to the one in memory.

CLOSE

P
D

FORMAT: CLOSE [# 1]

Abbreviation: CLOS.

See Also: OPEN

PURPOSE:

Closes a file.

REMARKS:

This commands closes a file on the currently accessed device such as the cassette tape drive. When a file is opened in the OUTPUT mode, the data remaining in memory (the output buffer) is output to the cassette file with an End Of File code before being closed.

A file will be closed in the following cases:

- An END, NEW, or RUN command is executed.
- The power is turned off.
- After editing a program (program entry, correction, deletion, or execution of DELETE command).
- A program is written into, read from, or deleted from the RAM disk.
- The CSAVE, CLOAD, or CLOAD? command is executed.
- The Monitor mode is selected.

Note:

The CLOSE command must be executed whenever serial communications are completed. If a file is left open and the data transfer cable is connected, current will continue to be drawn from the operating batteries in the computer.

CLS

P

FORMAT: CLS

Abbreviation:

See Also: LOCATE

PURPOSE:

Clears the display.

REMARKS:

Clears the display and resets the display start position to (0,0).

CONT

D

FORMAT: CONT 

Abbreviation: C.


See Also: STOP

PURPOSE:

Continues a program that has been temporarily halted.

REMARKS:

Valid only as direct input in the RUN mode.

Enter CONT to continue running a program that has been stopped with the STOP command. Enter CONT at the prompt to continue a program that has been halted using the  key.

EXAMPLE:

CONT

Continues an interrupted program.

CSAVE

P
D

FORMAT: 1. CSAVE "filename"
2. CSAVE
3. CSAVE "filename", "password"
4. CSAVE, "password"

Abbreviation: CS.

See Also: CLOAD, CLOAD?, PASS

PURPOSE:

Saves a program to tape.

REMARKS:

Format 1 writes all program lines in memory to the tape and assigns the indicated filename.

Format 2 writes all program lines in memory to the tape without assigning a filename.

Format 3 writes all program lines in memory to the tape and assigns them a specified filename and password.

Format 4 writes all program lines in memory to the tape without assigning a filename, but assigns a specified password.

Programs saved with a password can be loaded by anyone, but only someone who knows the password can list or modify them. (See PASS command.)

If a program in memory is write-protected, the CSAVE command will be ignored.

Avoid writing different programs with the same filename onto the same side (side A or B) of a tape. This may cause the wrong program to be read. It is recommended that the number on the tape counter be noted when writing a program onto tape.

For an explanation of tape operation, see the relevant section in the peripherals chapter.

EXAMPLE:

CSAVE "PRO3", "SECRET"

Saves all programs in memory to tape under the name "PRO3", protected with the password "SECRET".

DATA

P

FORMAT: DATA list of values

Abbreviation: DA.

See Also: READ, RESTORE

PURPOSE:

Provides values for use by READ.

REMARKS:

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.

A DATA statement may contain any numeric or string values, separated by commas. Enclose string values in quotes. Spaces at the beginning or end of a string should be included in the quotes.

DATA statements have no effect if encountered in the course of regular execution of the program, so they can be inserted wherever appropriate. Many programmers include them after the READ that uses them. If desired, the values in a DATA statement can be read a second time using the RESTORE statement.

EXAMPLE:

```
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
```

[10] Sets up an array.

[40] Loads the values from the DATA statement so that B(1) will be 10,B(2) will be 20,B(3) will be 30, etc.

DEGREE

P
D

FORMAT: DEGREE

Abbreviation: DE.

See Also: GRAD, RADIANT

PURPOSE:

Changes the form of angular values to decimal degrees.

REMARKS:

The computer 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 of all angular values to decimal degree form until GRAD or RADIANT is used. The DMS and DEG functions can be used to convert angles from decimal degree form to degree, minute, second form and vice versa.

EXAMPLE:

```
10: DEGREE
20: X = ASN 1
30: PRINT X
```

[20] X now has a value of 90; i.e., 90 degrees, the arc sine of 1.

DELETE

D

- FORMAT:**
1. DELETE line number
 2. DELETE line number –
 3. DELETE line number – line number
 4. DELETE – line number

Abbreviation: DEL.

See Also: NEW, PASS

PURPOSE:

Deletes specified program lines in memory.

REMARKS:

Valid only as direct input in the PRO mode.

Format 1 deletes only the specified program line.

Format 2 deletes program lines from the line number specified up to the highest program line in memory.

Format 3 deletes all program lines between the first specified line number (lower value) and the second specified line number (higher value).

Format 4 deletes program lines from the lowest line number in memory up to the specified line number.

Using DELETE in the RUN mode generates an error. If a password has been used, the command will not be executed and the prompt will be displayed. Only the digits 0-9 can be in the line numbers. Specifying a line that does not exist generates an error. Specifying a start line number that is greater than the end line number also generates an error.

If the first and second line numbers are omitted, an error will occur.

To delete the whole program, use the NEW command.

EXAMPLE:

```
DELETE 150*  
DELETE 200-**  
DELETE 50-150***  
DELETE -35****
```

* Deletes line 150 only.

** Deletes from line 200 to the highest line number.

*** Deletes all lines between and including line 50 and line 150.

**** Deletes from the lowest line number up to line 35.

DIM

P
D

- FORMAT:**
1. DIM numeric variable name 1 (size) [, numeric variable name 2 (size)]
 2. DIM string variable name 1 (size) *length
[, string variable name 2 (size) *length]
 3. DIM numeric array name 1 (rows, columns)
[, numeric array name 2 (rows, columns)]
 4. DIM string array name 1 (rows, columns) *length
[, string array name 2 (rows, columns) *length]

Abbreviation: D.

See Also: CLEAR, RUN

PURPOSE:

Reserves space for numeric and string array variables.

REMARKS:

DIM must be used to reserve space for any array variable. The size of an array is the number of elements in that array.

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 zeros 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 zeros row and column.

In string arrays, the size of each string element in addition to the number of elements are specified. For example, DIM B\$(3)*12 reserves space for 4 strings that 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.

Integers 0–255 can be used to reserve space for the desired array size (rows, columns), but an error may occur if a variable with the specified size cannot be reserved because of limits in free memory space and conditions of use.

If the size designation includes a decimal point, only the integer part will be recognized (and the fractional part will be ignored).

Example:

X(2.3) recognized as X(2)

Y(0.25) recognized as Y(0)

Array space and size may be declared by a numeric variable or expression:

```
10: INPUT A, B
20: DIM X(A), YY(B-1, A*B)
```

More than one array can be declared using one DIM statement.

Example:

```
DIM V(5), K$(4,3), XB$(5)
```

If an array has been defined, it cannot be defined again. For example, both DIM X(5) and DIM X(3,4) cannot be defined since the variable names are the same.

Numerical array and string array variables are recognized as different arrays; thus, the arrays Z() and Z\$() can both be defined.

Array variables can be cleared (or set undefined) with the CLEAR command. When the program is started using the RUN command, array variables are automatically cleared.

An array can be declared only once and any attempt made to redeclare the array within the program without first CLEARing it will generate errors. Be careful when executing a program using GOTO that the same DIM statement will not be executed again unless CLEAR has been used first.

EXAMPLE:

```
10: DIM B(10)
20: DIM C$(4,4)*10
30: DIM F$(12)
40: DIM H$(4,6)
```

[10] Reserves space for a numeric array with eleven elements.

[20] Reserves space for a two-dimensional string array with five rows and five columns; each string will be a maximum of ten characters.

[30] Reserves space for string variable F\$ with thirteen elements.

[40] Reserves space for a two-dimensional string array with five rows and seven columns; that is, thirty-five elements.

END

P

FORMAT: END

Abbreviation: E.

See Also:

PURPOSE:

Signals the end of a program.

REMARKS:

The program will be terminated when the END statement is executed. Statements after the END statement in the same line cannot be executed. An open file will be closed.

EXAMPLE:

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

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

FILES

D

FORMAT: FILES 



Abbreviation: FI.



See Also: LFILES, SAVE, LOAD



PURPOSE:

Displays names of files on the RAM disk.

REMARKS:

A maximum of four filenames will be displayed at one time, and an ⇒ mark will appear to the left of the filenames. Scroll through the files by pressing the  and  keys to move the ⇒ mark up or down, respectively.

If the  or  key is pressed, the entry prompt (>) is displayed, and the computer waits for the next command.

To display the listing of a program, position the ⇒ mark at the filename of the desired program, then press  + . Listing of contents of programs registered in the TEXT mode cannot be displayed, however.

FOR...NEXT

P

FORMAT: FOR numeric variable = expression 1 TO expression 2
 [STEP expression 3]
 {
 NEXT [numeric variable]

Abbreviation: F. N. STE.

See Also:

PURPOSE:

In combination with NEXT, repeats a series of operations a specified number of times.

REMARKS:

FOR and NEXT are used in pairs to enclose a group of statements that are to be repeated. If the variable following NEXT is omitted, the variable following FOR is assumed. The first time this group of statements is executed the loop variable (the variable named immediately following FOR) is assigned its initial value (expression 1).

When execution reaches the NEXT statement, the loop variable is increased by the STEP value (expression 3) and then this value is tested against the final value (expression 2). If the value of the loop variable is less than or equal to the final value, the enclosed group of statements is executed again, starting with the statement following FOR. If expression 3 for step value is omitted, the increment becomes 1. If the value of the loop variable is greater than the final value, execution continues with the statement that immediately follows NEXT. Because the comparison is made at the end, the statements within a FOR...NEXT loop are always executed at least once.

When the increment is zero, FOR...NEXT will continue in an infinite loop.

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. Write programs so that the program flow does not jump out of a FOR...NEXT loop before the counter reaches the final value. To exit a loop before it has been repeated the specified number of times, set the loop variable higher than the final value.

The group of statements enclosed within one FOR...NEXT loop can include another pair of FOR...NEXT statements that use a different loop variable as long as the enclosed pair is completely enclosed; i.e., if a FOR statement is included within a group of statements, the NEXT paired with it must also be included. FOR...NEXT loops may be "nested" up to five levels deep. Illegally jumping out of an inner loop will generate a nesting error.

Do not use the CLEAR or DIM command within a FOR...NEXT loop.

FRE

P
D

FORMAT: FRE

Abbreviation: FR.

See Also:

PURPOSE:

Returns the free memory available in the program and data area in bytes.

REMARKS:

FRE returns the byte count of the free space (in the area not used by BASIC programs, array variables, simple variables and machine code, and includes the RAM disk and text area) in the program and data area of memory.

Note:

When using this command in the PRO mode, enter PRINT FRE .

GOTO

P
D

FORMAT: GOTO {line number
*label }

Abbreviation: G.

See Also: GOSUB, ON...GOTO, RUN

PURPOSE:

Transfers program control to a specified line number or *label.

REMARKS:

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

Usually, a program is executed sequentially from the lowest line number. However, execution can be transferred to a line with the given line number or *label. Program execution can be started from the specified line by specifying a GOTO statement as direct input in the RUN mode. The transfer destination is specified by entering the line number or *label after the GOTO command.

Example:

GOTO 40 Jumps to line 40
GOTO *AB Jumps to the line with label *AB

If the specified line number or *label does not exist, an error occurs.

If two or more identical *labels are included in a program, program execution transfers to the line with the lowest line number.

EXAMPLE:

```
10: INPUT A$  
20: IF A$ = "Y" 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.

GRAD

P
D

FORMAT: GRAD

Abbreviation: GR.

See Also: DEGREE, RADIAN

PURPOSE:

Changes the form of angular values to gradient form.

REMARKS:

The computer 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 DEGREE or RADIAN is used. Gradient form represents angular measurement in terms of percent gradient, i.e., a 45° angle is a 50 percent gradient.

EXAMPLE:

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

X now has a value of 100, i.e., a 100 gradient, the arc sine of 1.

IF...THEN

P

FORMAT: IF condition THEN

line number
*label
statement

Abbreviation: IF T.

See Also: AND, OR, NOT

PURPOSE:

Conditionally executes a statement at the time the program is run.

REMARKS:

When the condition of the IF statement is true, the statement following THEN is executed; if it is false, the statement following THEN is skipped and the next line is executed.

If THEN is followed by a GOTO statement, either THEN or GOTO may be omitted.

Example:

1) IF A<5 THEN C=A*B:GOTO 50

If A is smaller than 5, then assign the product, A*B, to C and go to line 50.

2) IF B=C+1 GOTO 60

or

IF B=C+1 THEN 60

If B equals C+1, then go to line 60; otherwise go to the next line.

The condition (e.g. A<5) of the IF statement can be any relational expression listed below:

Relational expression	Description
○○ = ××	Equal to
○○ > ××	Greater than
○○ >= ××	Not less than
○○ < ××	Less than
○○ <= ××	Not greater than
○○ <> ××	Not equal to

Note: ○○ and ×× represent expressions (5*4, A, 8, etc.).

More than one relational expression can be linked with the logical operators “*” or “+”. For example:

IF (A>5)*(B>1) THEN

If A is greater than 5 and B is greater than 1, the statement following THEN is executed. Logical operator “AND” may be used in place of “*”. For example:

IF (A>5)+(B>1) THEN

If A is greater than 5 or B is greater than 1, the statement following THEN is executed. Logical operator “OR” may be used in place of “+”.

Using Character Strings in Relational Expressions

The magnitudes of character strings can be compared when used in a relational expression of an IF...THEN statement. The magnitudes of character codes are compared. For example, characters A, B, and C have codes 65, 66, and 67, respectively. So A is smaller than B, and B is smaller than C.

Length of Character Strings in Expressions

The total number of characters used in an expression linking or comparing two or more character strings cannot exceed 255.

EXAMPLE:

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

Note:

Whenever a variable name is to be followed by a statement, be sure to insert a space between, for example:

```
100 IF A=B THEN 200
```

↑ A space is needed.

Pay special attention to this when you use the IF, FOR, ON...GOTO, or ON...GOSUB command.

INKEY\$

P

FORMAT: INKEY\$


Abbreviation: INK.

See Also:

PURPOSE:



Gives the specified variable the value of the key pressed while the INKEY\$ function is executed.



REMARKS:

INKEY\$ is used to respond to the pressing of individual keys without waiting for the  key to end the entry.

See the table below for a list of applicable keys and the characters that are returned.

If no key is pressed when the INKEY\$ command is executed, a null value will be assigned to the variable.

The INKEY\$ command is unable to recognize any key operation made while the  key is pressed or immediately after the  key has been pressed.

 or  key operations also are not recognized. Lowercase alphabetic characters cannot be read.

Note:

If the INKEY\$ command is included in the first line of a program, it may act on the key that is pressed to begin running the program.

EXAMPLE:

```
10: CLS: WAIT 60
20: IF INKEY$ < > " " THEN 20
30: A$=INKEY$
40: IF A$=" " THEN 30
50: PRINT "---"; ASC A$; "---"
60: GOTO 10
```

INKEY\$ Character Code Table

Low \ High	0	1	2	3	4	5	...	8	9	...	F
0		2ndF	SPACE	0		P					
1				1	A	Q			ln		
2	C-CE			2	B	R			log		
3	CONST			3	C	S					
4	↑			4	D	T					
5	↓	CAPS		5	E	U			sin		
6				6	F	V			cos		
7	ANS	BS		7	G	W		1/x	tan		
8	BASIC	RM	(8	H	X		x ²			
9	CAL	M+)	9	I	Y					
A	TAB		x*		J	Z					
B	INS		+	;	K				→DEG		π
C	DEL		,		L				FSE		√
D	←		-	+/-	M				hyp		
E	▶		.		N	y ^x			MDF		
F	◀		+/-		O						

- The  key functions as a Break key.

INPUT

P

FORMAT: 1. INPUT variable [,variable]
2. INPUT "prompt string", variable [[,"prompt string"], variable]
3. INPUT "prompt string"; variable [[,"prompt string"]; variable]

Abbreviation: I.


See Also: INPUT#, INKEY\$, READ, LOCATE

PURPOSE:

Allows entry of one or more values from the keyboard.

REMARKS:

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

Format 1 displays symbol "?" to prompt data entry. If data is entered and the  key is pressed at this prompt, the system assigns the data to the variable and resumes program execution.

If more than one variable is specified, the data prompt is repeated the corresponding number of times.

During data prompt, format 2 displays the character string enclosed by double quotes (" ") as entry guidance. The guidance disappears when data is entered.

Format 3 also displays entry guidance during data prompt, but the entered data appears following the entry guidance, which does not disappear.

Formats 1, 2, and 3 may be concurrently used in one INPUT statement:

```
INPUT "A=";A,B,"C=?",C
```

The type of the variables given in the INPUT statement must match the type of input data. Assign string data to string variables, and numerical data to numerical variables. If "ABC" is entered in response to a numerical entry prompt, the value assigned to variable AB is assigned. This allows you to enter such values as SIN30.

If the start position is specified using the LOCATE statement before executing the INPUT statement, the prompt string or ? will be displayed at the specified location.


EXAMPLE:

```
10: INPUT A  
20: INPUT "A=";A  
30: INPUT "A=",A  
40: INPUT "X=?";X,"Y=?";Y
```

[10] Puts a question mark at the left margin.

[20] Displays "A=" and waits for data to be entered.

[30] Displays "A=". When data is entered, "A=" disappears and the data is displayed starting at the left margin.

[40] Displays "X=?" and waits for the first entry. After  is pressed, "Y=?" is displayed at the left margin.

INPUT#

P

FORMAT: INPUT# 1, variable, variable, ..., variable

Abbreviation: I.#

See Also: OPEN, PRINT#

PURPOSE:

Reads items from sequential files on the SIO (serial I/O) device or cassette tape.

REMARKS:

INPUT# assigns data values that have been transferred from an SIO device or cassette tape to specified variables.

This statement is valid only with an SIO device opened with the file descriptor "COM:" or with a cassette file opened with the file descriptor "CAS:" and the access mode "FOR INPUT".

Specify variables as follows:

- Fixed variables (A, X, B\$, etc.)
- Simple variables (CD, EF\$, etc.)
- Array elements (B(10), C\$(5,5), etc.)
- All array variables (B(*), C\$(*), etc.)

An error will occur if the file contains less data than the number of specified variables. If the file contains more data, the rest of the data will be ignored.

Use a comma (,), space (&H20), CR (&H0D), LF (&H0A) or CR + LF as a delimiter when data are read into numeric variables. Spaces preceding data are ignored.

If data that cannot be translated into numerals is read to a numeric variable, the data is assumed to be zero.

Use a comma (,), CR, LF or CR + LF as a delimiter to read data into character variables. Spaces preceding data are ignored. A data delimiter automatically replaces the 256th character that is output. If a double quotation mark appears at the beginning of data, data is read up to the next double quotation mark. A comma in a character string enclosed by double quotation marks is assumed not to be a delimiter.

Treatment of EOF (End Of File: &H1A) Code:

- An EOF code preceding data causes an error.
- An EOF code in the middle of data is treated as a data delimiter.

Order of Data Input into array variables:

Examples:

For one-dimensional array variables: B (0) → B (1) → B (2) →

For two-dimensional array variables: C (0, 0) → C (0, 1) → C (0, 2) →

C (1, 0) → C (1, 1) → C (1, 2) →

KILL

D

FORMAT: KILL "filename" 

Abbreviation: K.

See Also: SAVE

PURPOSE:

Deletes a file on the RAM disk.

REMARKS:

KILL deletes a file with the specified filename from the RAM disk.

An error occurs if the specified file does not exist.

LEFT\$

P

D

FORMAT: LEFT\$("string",expression)

Abbreviation: LEF.

See Also: MID\$, RIGHT\$

PURPOSE:

Returns the specified number of characters from the left end of a given string.

REMARKS:

LEFT\$ returns the number of characters specified by the expression from the left end of the given string.

For example, if A\$="ABCD", LEFT\$(A\$,3) returns the leftmost 3 characters, "ABC".

EXAMPLE:

```
10: X$="SHARP": WAIT 60
20: FOR N=1 TO 5
30: LET S$=LEFT$(X$,N)
40: PRINT S$
50: NEXT N
```

```
RUN
S
SH
SHA
SHAR
SHARP
```

LEN

P
D

FORMAT: LEN "string"

Abbreviation:

See Also:

PURPOSE:

Returns the number of characters in a string.

REMARKS:

The number of characters in the string includes any blanks or non-printable characters such as control codes or carriage returns.

EXAMPLE:

```
10: INPUT "ENTER A WORD ";A$
20: N=LEN A$
30: PRINT "THE WORD LENGTH IS ";N
40: END
```

```
RUN
ENTER A WORD CHERRY
THE WORD LENGTH IS 6.
```

[10] Prompts for a word. In this example, the user enters "CHERRY".

[20] Finds the length of the word.

[30] Prints out the answer.

LET

P

FORMAT: 1. LET numeric variable = expression
2. LET string variable = string

Abbreviation: LE.

See Also:

PURPOSE:

Used to assign a value to a variable.

REMARKS:

LET assigns the value of the expression to the designated variable. The type of 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.

The LET command may be omitted in all LET statements.

EXAMPLE:

```
10: I=10
20: A=5*I
30: X$=STR$ A
40: IF I >=10 THEN LET Y$=X$+".00"
```

[10] Assigns the value 10 to I.

[20] Assigns the value 50 to A.

[30] Assigns the value 50 to X\$.

[40] Assigns the value 50.00 to Y\$.

LFILES

D

FORMAT: LFILES 

Abbreviation: LF.

See Also: FILES

PURPOSE:

Prints out filenames stored on the RAM disk.

REMARKS:

Prints all the filenames on the RAM disk.

LIST

D

FORMAT: 1. LIST
2. LIST line number
3. LIST *label

Abbreviation: L.

See Also: LLIST, PASS

PURPOSE:

Displays a program.

REMARKS:

Valid only as direct input in the PRO mode.

In format 1, the program is displayed from its first line until the display is full.

In format 2, the program is displayed from the specified line number until the display is full. Use the key to advance to the next line in the list. If the line for the specified number does not exist, the program will be displayed from the line with the next largest number that does exist.

In format 3, the program is displayed from the line with the specified label until the display is full.

If a password has been set, the LIST command is ignored.

An error will occur if a *label that is specified does not exist in the program, or a line number is specified that is greater than the last line number in the program.

EXAMPLE:

LIST 100

Displays line number 100.

LLIST

D

- FORMAT:**
1. LLIST
 2. LLIST $\left\{ \begin{array}{l} \text{line number} \\ *label \end{array} \right\}$
 3. LLIST [line number 1] – [line number 2]

Abbreviation: LL.

See Also: LIST, PASS

PURPOSE:

Prints out a program on the optional CE-126P printer.

REMARKS:

Valid only as direct input in the PRO or RUN mode.

Format 1 prints or sends all programs in memory.

Format 2 prints or sends only the program line for which the number or label is specified.

Format 3 prints or sends the statements from line number 1 through line number 2.

There must be at least two lines between the numbers.

Either line number 1 or line number 2 may be omitted. If line number 1 is omitted, the program listing is printed from the first line through line number 2. If line number 2 is omitted, the program listing is printed from line number 1 through the end of the program.

If the line with the line number given in format 2 does not exist, or the lines with line numbers 1 and 2 given in format 3 do not exist, the nearest larger numbers are assumed.

If a password has been set, the LLIST command is ignored.

EXAMPLE:

LLIST 100 – 200

Prints program listing between line numbers 100 and 200.

LLIST – 200

Prints program listing from the first line through line 200.

LLIST 100 –

Prints program listing from line 100 through the last line.

LOAD

D

FORMAT: LOAD "filename" 





Abbreviation: LO.

See Also: SAVE, FILES

PURPOSE:

Loads a BASIC program on the RAM disk.

REMARKS:

An error will occur if the program area is exceeded as a result of loading a program. In such a case, clear unnecessary variables from the data area. If program filenames have been displayed with the FILES command, the desired program can be simply loaded by selecting it with the  or  key, and pressing  + .

The file extension may be omitted only if it is a ".BAS" extension.

While a program is being loaded, any open file will be closed.

An error will occur if an attempt is made to load a TEXT program.

LOCATE

P

FORMAT: LOCATE expression 1, expression 2

Abbreviation: LOC.

See Also: CLS, INPUT, PRINT

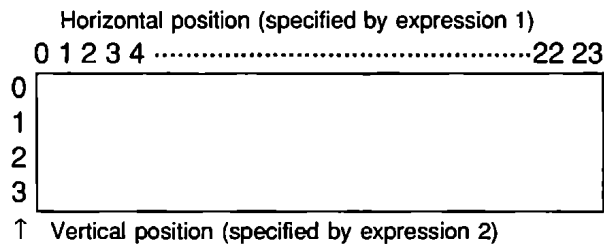
PURPOSE:

Specifies the display start position in column units.

REMARKS:

Specifies the display start position in units of a character position for the contents displayed by the PRINT and INPUT commands.

The display position is specified, as follows:



A position on the display is specified by its horizontal and vertical positions. Expression 1 specifies the horizontal position, and expression 2 specifies the vertical position. The range of expression 1 is 0 to 23, and the range of expression 2 is 0 to 3. An error occurs if the expressions are not specified within these ranges.

EXAMPLE:

```
10: CLS
20: LOCATE 2,1: PRINT "ABCDE"
30: LOCATE 0,2: PRINT "123"
```



Using the LOCATE command allows text to be written to any part of the display without affecting existing text except where characters are directly overwritten. Use the CLS command to clear the entire display.

If the number of characters exceeds the limits of the display, the display will scroll to show all the characters, even if the display start position was specified with the LOCATE command.

LPRINT

P
D

- FORMAT:**
1. LPRINT $\left\{ \begin{array}{l} \text{expression} \\ \text{string} \end{array} \right\} \left[\left[\begin{array}{l} \text{expression} \\ \text{string} \end{array} \right] \right]$
 2. LPRINT $\left\{ \begin{array}{l} \text{expression} \\ \text{string} \end{array} \right\} \left[\left[\begin{array}{l} \text{expression} \\ \text{string} \end{array} \right] \right] [;]$
 3. LPRINT USING "format"; $\left\{ \begin{array}{l} \text{expression} \\ \text{string} \end{array} \right\} \left[\left[\begin{array}{l} \text{expression} \\ \text{string} \end{array} \right] \right] [;]$
 4. LPRINT

Abbreviation: LP.

See Also: PRINT, USING

PURPOSE:

Outputs given data to the printer.

REMARKS:

When a semicolon (;) is placed at the end of the statement, the next LPRINT command in the program will print specified data directly after the data printed by the first LPRINT. A comma (,) cannot be placed at the end of the statement.

If a single item is specified using format 1 for the CE-126P Printer, numerical values are printed from the right margin of the paper, strings from the left.

Use commas (,) to separate expressions or strings in order to divide the 24 columns of each printer line into twelve-column areas. Numerical values or strings are printed in each of these areas; numerical calculations are printed from the right margin of each twelve-column area, strings from the left. For numerical values longer than twelve digits, the extra part of the mantissa is truncated when printed. For strings longer than twelve characters, only the first twelve characters are printed; all other characters will be ignored.

Format 2 consecutively prints out data from the left margin.

When a semicolon (;) is placed at the end of the statement, the next LPRINT command in the program will print data consecutive to the data printed by the first LPRINT.

Format 3 prints out data in the exact format specified in the statement. Either a comma (,) or semicolon (;) may be used as a separator.

For the format for USING, see the USING command.

Format 4 prints only delimiter codes. If the preceding LPRINT statement is terminated with a semicolon (;) with unprinted data left in the buffer, format 4 prints that data.

EXAMPLE:

```
LPRINT A,B  
LPRINT A;B;Z$
```

MID\$

P
D

FORMAT: MID\$(string,N,M)

Abbreviation: MI.

See Also: LEFT\$, RIGHT\$

PURPOSE:

Returns a string of M characters from inside a string starting from the Nth character in the string.

REMARKS:

If N is greater than the number of characters in the string, a null string will be returned. If N is less than 1, an error will occur. M must be in the range of 0 to 255, and N in the range of 1 to 255. Fractions will be truncated.

EXAMPLE:

```
10: Z$="ABCDEFGG"  
20: Y$= MID$(Z$,3,4)  
30: PRINT Y$
```

MON

D

FORMAT: MON 

Abbreviation: MO.

See Also:

PURPOSE:

Selects the machine language Monitor mode.

REMARKS:

Valid only as direct input in the PRO or RUN mode.

MON places the computer in the machine language Monitor mode. (See the explanations of the Machine Language Monitor.)

NEW

D

FORMAT: NEW 

Abbreviation:

See Also: CLEAR, PASS

PURPOSE:

Clears existing programs and data.

REMARKS:

The NEW command clears all programs and data that are currently in memory.
(Programs with passwords cannot be cleared.)

An open file will be closed.

EXAMPLE:

NEW

ON...GOSUB

P

FORMAT: ON expression GOSUB {line number 1
*label 1}, {line number 2
*label 2}, ...

Abbreviation: O. GOS.

See Also: GOSUB, GOTO, ON...GOTO

PURPOSE:

Executes one of a set of subroutines, depending on the value of a control expression.

REMARKS:

When ON...GOSUB is executed, the expression between ON and GOSUB is evaluated and reduced to an integer. If the value of the integer is 1, control is transferred to line number 1 or *label 1 in the list, as in a normal GOSUB. If the expression is 2, control is transferred to line number 2 or *label 2 in the list, and so forth.

Note:

Be sure to place a space just before the GOSUB command. Otherwise it may be regarded as a variable.

If the expression is zero, negative, or larger than the number of line numbers provided in the list, no subroutine will be executed and execution will proceed with the next statement or line of the program.

Use commas (,) to separate line numbers or *labels in the list.

EXAMPLE:

```
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 entry of 1 displays "FIRST"; 2 displays "SECOND"; 3 displays "THIRD". Any other entry does not produce any display.

ON...GOTO

P

FORMAT: ON expression GOTO {line number 1
*label 1}, {line number 2
*label 2}, ...

Abbreviation: O. G.

See Also: GOSUB, GOTO, ON...GOSUB

PURPOSE:

Transfers control to one of a set of locations, depending on the value of a control expression.

REMARKS:

When ON...GOTO 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 line number 1 or *label 1 in the list. If the expression is 2, control is transferred to line number 2 or *label 2 in the list, and so forth.

Note:

Be sure to place a space just before the GOTO command. Otherwise it may be regarded as a variable.

If the expression is zero, negative, or larger than the number of line numbers provided in the list, execution will proceed with the next statement or line of the program.

Use commas (,) to separate line numbers or *labels in the list.

EXAMPLE:

```
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 entry of 1 displays "FIRST"; 2 displays "SECOND"; 3 displays "THIRD". Any other entry does not produce any display.

OPEN

P
D

FORMAT: 1. OPEN "COM:"
2. OPEN "CAS:[filename]" FOR OUTPUT
3. OPEN "CAS:[filename]" FOR INPUT

Abbreviation: OP.
See Also: CLOSE

PURPOSE:

Opens an SIO (serial I/O) device or a cassette tape file for input or output of data.

REMARKS:

Format 1 opens an SIO device for data input/output (opens the SIO circuit). The filename or access mode cannot be specified. Serial communications parameters are set in the TEXT mode. (See page 105.)

Format 2 opens a cassette tape file for output of data to the file. Before executing this command, make sure the cassette recorder is set to Record. When executed, this command writes an information block (header) to the tape. If a filename is specified, it is also recorded in the information block.

Format 3 opens a cassette tape file for input of data from a file. Before executing this command, make sure the cassette recorder is set to Play.

This command searches for the information block in which the specified filename exists, and loads the specified file from the tape. If no filename is specified, the computer loads the file whose filename was first encountered in the information block after the tape was started.

Notes:

- If the specified file is not found, the computer will continue searching after the end of the tape is reached. If this occurs, press the **BREAK** key to stop the search.
- Only one file can be opened at a time. An error will occur if the OPEN command is executed when a file has already been opened.

PASS

D

FORMAT: PASS "character string" 

Abbreviation: PA.

See Also: CSAVE, SAVE, CLOAD, LOAD, DELETE, LIST, NEW, RENUM

PURPOSE:

Sets and cancels passwords.

REMARKS:

Passwords are used to protect programs from being listed or edited by unauthorized users. A password consists of a character string that is no more than eight characters long. The eight characters must be alphanumeric characters or symbols. The character string cannot be a null string.

Once a PASS command has been entered, the programs in memory are protected. A program protected by a password cannot be examined or modified in memory. It cannot be saved to tape or the RAM disk, or listed with LIST or LLIST. Nor is it possible to add or delete program lines. The only way to remove this protection is to execute another PASS command with the same password.

If a password is set in the program to be loaded, that password is also set within the computer. If not, no password is set within the computer.

If PASS is executed when no program is loaded, an error will occur and no password will be set.

A password-protected program is protected against the NEW or DELETE command as well.

EXAMPLE:

PASS "SECRET" 

Establishes the password "SECRET" for the program in memory.

PRINT

P
D

- FORMAT:**
1. PRINT $\left. \begin{array}{l} \text{expression} \\ \text{string} \end{array} \right\} \left[\begin{array}{l} \text{expression} \\ \text{string} \end{array} \right]$
 2. PRINT $\left. \begin{array}{l} \text{expression} \\ \text{string} \end{array} \right\} \left[\begin{array}{l} \text{expression} \\ \text{string} \end{array} \right] [;]$
 3. PRINT USING "format"; $\left. \begin{array}{l} \text{expression} \\ \text{string} \end{array} \right\} \left[\left. \begin{array}{l} \text{expression} \\ \text{string} \end{array} \right\} \right] [;]$
 4. PRINT

Abbreviation: P.

See Also: LPRINT, USING, WAIT, LOCATE

PURPOSE:

Displays information.

REMARKS:

PRINT displays prompt information, results of calculations, etc.

If the start position is specified by the LOCATE statement, the data will be displayed from the specified location.

If a semicolon (;) is at the end of the statement, the contents will be displayed continuously. A comma (,) cannot be placed at the end of the statement.

Format 1 displays as follows:

- 1) For a single item to be displayed:

If the expression is numeric, the value is shown from the right margin of the display. If it is a string, it is shown from the left margin of the display.

EXAMPLE:

10: PRINT 1234

20: PRINT "ABCD"

ABCD	1234.
------	-------

- 2) For two or more items to be displayed (specified with commas):

Divide the twenty-four columns of each line into twelve-column areas. Numeric values are displayed from the right margin of each twelve-column area, and strings from the left margin.

EXAMPLE:

```
10: A = 1234: B = 5/9: C$ = "ABCDE":WAIT 200
20: CLS: PRINT "A=",A
30: CLS: PRINT A,C$,B
```

RUN 

```
A=                               1 2 3 4 .
```

```
                               1 2 3 4 . A B C D E
5 . 5 5 5 5 5 E - 0 1
```

If a numeric value exceeds twelve digits (when the decimal fraction in the exponential display is seven digits or more), the least significant digits are truncated. When a character string exceeds twelve columns, only the first twelve characters (from the left) will be displayed.

Format 2 displays the data continuously from the left margin of the display.

EXAMPLE:

```
10: A = 1234:B = 5/9:C$ = "ABCDE"
20: PRINT "A=";A
30: PRINT "EFGHI";B;C$;A
```

RUN 

```
A = 1 2 3 4 .
E F G H I 5 . 5 5 5 5 5 5 5 6 E - 0 1 A B C D
E 1 2 3 4 .
>
```

Format 3 displays the data by following the specified format.

Refer to the USING command for USING format. Commas (,) and semicolons (;) will be treated as usual. The USING statement is valid also to the next PRINT statement encountered in the program.

Example: PRINT USING "&&&&&&&";"ANSWER=";:PRINT USING "####.##";5/9

Format 4 displays the previously displayed value as is. (Usually, it is used together with the WAIT command to retain the current display.)

EXAMPLE:

```
10: CLS
20: FOR A=0 TO 159
30: PRINT CHR$(A+32);
40: NEXT A
50: WAIT: PRINT
```

The characters displayed between lines 20 and 40 will remain on the display at line 50. (An infinite interval is set.)

PRINT → LPRINT setting

The computer can switch all PRINT commands to function as LPRINT commands.
Connect the printer before executing the following statement:

Setting: PRINT=LPRINT

Resetting: PRINT=PRINT

Resetting can also be performed by:

- 1) executing the RUN command
- 2) pressing the **SHIFT** + **CA** keys
- 3) turning the power off and then on.

Since the RUN command resets the setting, run the program using the GOTO command.

PRINT#

P

FORMAT: PRINT# 1, {expression} [{,}] {expression} [{,}] [{string}] [{;}] [{string}] [{;}]

Abbreviation: P.#

See Also: OPEN, INPUT#

PURPOSE:

Writes values of specified variables to the SIO (serial I/O) device or cassette tape.

REMARKS:

PRINT# is valid only with an SIO device opened with the file descriptor "COM:" or with a cassette file opened with the file descriptor "CAS:" and the access mode "FOR OUTPUT".

When an array variable (one or two dimensions) has been specified in the form of "array name(*)", the entire array is written to the file. Its elements are written in the order of, for example, C\$(0,0), C\$(0,1), C\$(0,2)....C\$(1,0).....C\$(5,5).

It is recommended that the FOR...NEXT statement be used for writing array variable data.

When the respective elements of the array are specified, they must be specified in the form of "B(7)", "C\$(5,6)", etc.

When a character or string element is used, it must not be specified using a semicolon (;):

```
PRINT#1,"ABC"
```

```
PRINT#1,A$
```

If PRINT#1,"ABC";A\$ is executed, no data delimiter will be written and "ABC" and A\$ cannot be distinguished.

A numeric value is recorded in such a form that the sign (a space when it is positive), numeric character string, and a space appear in that form. The recording format is shown below:

- (1) When a comma or semicolon does not follow the data, CR(&H0D) and LF(&H0A) are provided.

Example:

```
PRINT #1, - 1.2
```

-	1	.	2		CR	LF
---	---	---	---	--	----	----

```
PRINT #1, "ABC"
```

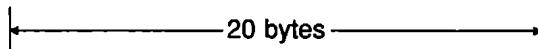
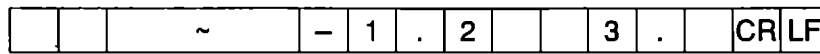
A	B	C	CR	LF
---	---	---	----	----

When data is terminated with a comma (,) or semicolon (;), no CR+LF code will be output.

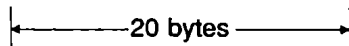
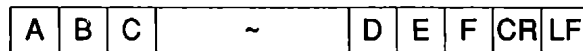
(2) When a comma follows the data, twenty bytes are occupied. A numeric value is right justified and a character string is left justified.

Example:

PRINT #1, - 1.2,3



PRINT #1, "ABC", "DEF"

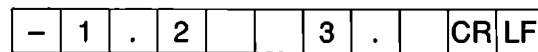


When the character string exceeds twenty bytes, the excess part is written to the next twenty-byte area.

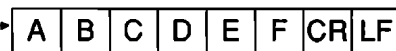
(3) When a semicolon follows the data, it is stored without spaces.

Example:

PRINT #1, - 1.2;3



PRINT #1, "ABC";"DEF"



←----- In this case, character strings "ABC" and "DEF" are not read separately.

When all the variables of an array are specified, each output of a variable is followed by the CR+LF code. For example, PRINT#1, B(*) is identical to PRINT#1, B(0): PRINT#1, B(1): PRINT#1, B(2): and so on.

For exponential values, when the mantissa is output, it is followed by exponent symbol E, the sign, and a two-digit exponent (a one-digit exponent must be preceded by a leading zero).

EXAMPLE:

```
10: OPEN "CAS:DATA" FOR OUTPUT
20: FOR J=0 TO N
30: FOR K=0 TO M
40: PRINT #1,C$(J, K)
50: NEXT K:NEXT J:CLOSE
```

RADIAN

P
D

FORMAT: RADIAN

Abbreviation: RAD.

See Also: DEGREE, GRAD

PURPOSE:

Changes the form of angular values to radians.

REMARKS:

The computer has three forms for representing angular values: 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 of all angular values to radian form until DEGREE or GRAD is used. Radian form represents angles in terms of the length of the arc with respect to the radius, i.e., 360° is 2π radians, since the circumference of a circle is 2π times the radius.

EXAMPLE:

```
10: RADIAN
20: X = ASN 1
30: PRINT X
```

X now has a value of 1.570796327 or $\pi/2$, the arc sine of 1.

RANDOMIZE

P
D

FORMAT: RANDOMIZE

Abbreviation: RA.

See Also: RND

PURPOSE:

Resets the seed for random number generation.

REMARKS:

When random numbers are generated using the RND function, the computer begins with a predetermined "seed" or starting number. RANDOMIZE resets this seed to a new randomly determined value.

The starting seed will be the same each time the computer 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 RANDOMIZE statement can be used to make the seed itself random.

EXAMPLE:

```
10: RANDOMIZE  
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.

READ

P

FORMAT: READ variable, variable, ..., variable

Abbreviation: REA.

See Also: DATA, RESTORE

PURPOSE:

Reads values from a DATA statement and assigns them to variables.

REMARKS:

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 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 using the RESTORE statement.

Note:

The type of data must match the type of variable (numerical or string) to which it is assigned.

EXAMPLE:

```
10: DIM B(10)
20: WAIT 60
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: READ C, D, E$, F$
100: PRINT C,D,E$,F$
110: DATA 3,5,G=,H=
120: END
```

[10] Set up an array.

[40] Loads the values from the first DATA statements into B(). B(1) is 10, B(2) is 20, B(3) is 30, etc.

[90] Loads the values from the final DATA statement. C is 3, D is 5, E\$ is G=, F\$ is H=.

REM(')

P

FORMAT: REM remark or ' remark

Abbreviation:

See Also:

PURPOSE:

Includes comments in a program.

REMARKS:

It is often useful to include explanatory comments in a program. These can provide titles, names of authors, dates of last modification, usage notes, reminders about algorithms, etc. These comments are included using the REM (or apostrophe (')) statement.


The REM (') statement has no effect on program execution and can be included anywhere in the program. Everything following REM (') in that line is treated as a comment.

EXAMPLE:

```
10: ' THIS LINE HAS NO EFFECT
100: REM THIS LINE HAS NO EFFECT EITHER.
```

RENUM

D

FORMAT: RENUM [new line number] [, [old line number] [,increment]] 

Abbreviation: REN.

See Also: DELETE, LIST


PURPOSE:

Renumbers the lines of a program.



REMARKS:

Valid only as direct input in the PRO mode.

The line numbers are changed from old line numbers to new line numbers with a specified increment. If the new line number is not specified, the lines are renumbered starting with line 10. If the increment is not specified, the lines are renumbered with an increment of 10. RENUM updates referenced line numbers in GOTO, ON...GOTO, GOSUB, ON...GOSUB, RESTORE, and (IF)...THEN statements.


If a line number is given in the form of a variable (example: GOTO A) or numerical expression (example: GOTO 2  50), the line will not be renumbered properly. If a line number is given by a variable or expression, temporarily make it a remark (REM), and correct it after executing the RENUM command. It is recommended that you replace such commands with ON...GOTO commands, etc.

If a line number exceeds 65279, an error will be generated. If a specified old line number does not exist, an error will be generated. Changing the execution order generates an error (example: RENUM 15,30 when there are already lines numbered 10, 20, and 30). If a password has been used, an error occurs.

If the display shows "*", pressing the  key will interrupt renumbering. A display of "**" indicates that renumbering cannot be interrupted. Error generation or use of the  key leaves the program unchanged.

EXAMPLE:

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

RENUM 100, 10, 5 

```
100: INPUT "CONTINUE";A$
105: IF A$ = "YES" THEN 100
110: IF A$ = "NO" THEN 125
115: PRINT "ENTER YES OR NO PLEASE!"
120: GOTO 100
125: END
```

RESTORE

P

FORMAT: 1. RESTORE {line number
 *label }
 2. RESTORE

Abbreviation: RES.

See Also: DATA, READ

PURPOSE:

Rereads values in a DATA statement or changes the order in which these values are read.

REMARKS:

In the regular use of READ the computer begins reading with the first value in a DATA statement and proceeds sequentially through the remaining values. Format 1 resets the pointer to the first value of the DATA statement whose line number is equal to the specified line number or *label. Format 2 resets the pointer to the first value of the first DATA statement, so that it can be read again.

EXAMPLE:

```
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
```

[10] Sets up an array.

[50] Assigns the value 20 to each of the elements of B().

RIGHT\$

P
D

FORMAT: RIGHT\$(string,N)

Abbreviation: RI.

See Also: LEFT\$, MID\$

PURPOSE:

Returns N characters from the right end of a string.

REMARKS:

Fractions will be truncated. If N is less than 1, a null string will be returned. If N is greater than the number of characters in the string, the whole string will be returned.

EXAMPLE:

```
5: WAIT 60
10: XX$ = "SHARP COMPUTER"
20: FOR N = 1 TO 14
30: SS$ = RIGHT$(XX$,N)
40: PRINT SS$
50: NEXT N
```

RND

P
D

FORMAT: RND numeric expression

Abbreviation: RN.

See Also: RANDOMIZE

PURPOSE:

Generates a random number.

REMARKS:

If the value of the expression is less than 2 but greater than or equal to zero, the random number is less than 1 and greater than zero. If the expression is an integer greater than or equal to 2, the result is a random number greater than or equal to 1 and less than or equal to the expression. If the expression is greater than or equal to 2 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 that is larger than the expression. (In this case, the generation of the random number changes depending on the value of the decimal portion of the argument.) If the expression is negative, the previously set numeric expression is used to generate the random number.

<u>Argument</u>	<u>Result</u>	
	<u>Lower Limit</u>	<u>Upper Limit</u>
.5	0<	<1
2	1	2
2.5	1	3

The same sequence of random numbers is normally generated because the same "seed" is used each time the computer is turned on. To randomize the seed, see the RANDOMIZE command.

RUN

D

FORMAT: 1. RUN
2. RUN {line number
*label}

Abbreviation: R.
See Also: GOTO

PURPOSE:

Executes a program in memory.

REMARKS:

Format 1 executes a program beginning with the lowest numbered statement in memory.

Format 2 executes a program beginning with a specified line number

An error will occur if the specified line number or *label is not found.

If two or more identical labels exist in a program, the one with the lesser line number will be executed.

EXAMPLE:

RUN 100

Executes the program starting from line 100.

SAVE

D

FORMAT: SAVE "filename"

Abbreviation: SA.

See Also: LOAD, FILES

PURPOSE:

Saves the BASIC program to the RAM disk.

REMARKS:

The SAVE statement names a BASIC program in memory and then writes it to the RAM disk.

A filename is the name given to a program. The desired file can be readily retrieved by the computer if it is given a filename.

A filename may consist of up to eight alphanumeric characters or symbols.

If no extension is specified, .BAS is assumed. The extension can consist of up to three characters.

An existing file will be erased if the same filename is specified, but an error occurs if the existing file is a TEXT file.

The SAVE command will be ignored when there is no program loaded or the program is password protected.

STOP

P

FORMAT: STOP

Abbreviation: S.
See Also: CONT

PURPOSE:

Halts execution of a program for diagnostic purposes.

REMARKS:

When STOP is encountered in program execution, execution halts and a message such as "BREAK IN 200" is displayed 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 to examine the state of variables. Execution may be restarted with the CONT command. Pressing the key executes the program line by line.

EXAMPLE:

10: STOP

Causes "BREAK IN 10" to appear on the display.

STR\$

P
D

FORMAT: STR\$ expression

Abbreviation: STR.

See Also: VAL

PURPOSE:

Converts numeric data into string data.

REMARKS:

The STR\$ function changes numeric data to a string. The string will be composed of the same digits as the original number. The STR\$ function has the opposite effect of the VAL function.

If the numeric data is negative, the string will be preceded by a minus (-) sign.

EXAMPLE:

```
:  
:  
110: N=N*3  
120: A$=STR$ N  
130: B$=LEFT$ (A$,3)  
140: M=VAL B$  
:
```

[110] Program performs calculations on numeric variable N.

[120] The numeric variable N is converted to the string variable A\$. String variables are easier to manipulate than numerics. In this example, suppose that the first 3 digits of the number are required. Having converted the number to a string, we can use any of the string manipulation commands: LEFT\$, RIGHT\$, MID\$.

[130] Stores only the first 3 digits (characters) of the number into string variable B\$.

[140] The first 3 digits are reconverted into a numeric variable for processing by the program as a number.

TROFF

P
D

FORMAT: TROFF

Abbreviation: TROF.
See Also: TRON

PURPOSE:
Cancels trace (TRON) mode.

REMARKS:
Execution of TROFF restores normal execution of the program.

EXAMPLE:
See TRON.

TRON

P
D

FORMAT: TRON

Abbreviation: TR.
See Also: TROFF

PURPOSE:
Enables the trace mode.

REMARKS:
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. To stop trace execution, press the **BREAK** key or execute the STOP command. After trace execution is stopped, the computer waits for the down arrow key to be pressed before moving on to the next statement. The trace mode continues until TROFF is executed, or the **SHIFT** + **CA** keys are pressed.

EXAMPLE:
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, and 30.

USING

P
D

FORMAT: 1. USING format string
2. USING

Abbreviation: U.

See Also: LPRINT, PRINT

PURPOSE:

Controls the format of displayed or printed output.

REMARKS:

USING can be used by itself or as a clause within a PRINT or LPRINT statement. USING establishes a specified format for output that is used for all output that follows until changed by another USING.

#: Right justified numeric field character.

Length of integer field: 2 to 11 (including sign)

If a value is shorter than the specified numeric field, the extra portion of the field is filled with spaces. An error occurs if a value is longer than the field.

If a numeric field with a length of twelve or more digits is specified, it is regarded to be eleven digits long.

Length of decimal field: 0 to twelve (0 to 9 for exponential numbers)

If a value is shorter than the specified field, zeros appear in the extra portion of the field. If the former is longer than the latter, the extra digits are truncated.

.: Decimal point (delimiter for integer and decimal parts)

^: Used to indicate that numbers should be displayed in scientific notation.

With this notation, the length of the mantissa field is always 2 (1 digit and the sign), without regard to the specified length of the integer field. If the given length of the decimal field is nine or more digits, the length of the decimal field of the mantissa is also nine digits.

&: Left justified alphanumeric field

If a string is shorter than the specified field, spaces appear in the extra portion of the field. If the former is longer than the latter, the extra characters are dropped.

(1) USING"###"

Prints the sign and two integer digits.

(2) USING"###."

Prints the sign, two integer digits, and a decimal point.

(3) USING"###.##"

Prints the sign, two integer digits, a decimal point, and two decimal places.

(4) USING"##.##^"

Prints numerical data in exponential form with up to two decimal places.

Spaces for one integer digit and the sign are automatically reserved for the mantissa, and for two integer digits, the capital E, and the sign for the exponent.

- (5) USING"&&&&&"
Prints a string of six characters.
- (6) USING"###&&&"
Prints a string adjacent to a numeric value.
- (7) USING
Format 2 clears formatting.

Formatting is also cleared by executing the RUN command, pressing **SHIFT** + **CA**, or turning the computer off and then on.

EXAMPLE:

```
10: B=-10:C=10.7703
20: PRINT USING "&&&###" ; "B=" ; B ; "_C=" ;; PRINT USING "###.###"; C
```

VAL

P
D

FORMAT: VAL string

Abbreviation: V.
See Also: STR\$

PURPOSE:

Converts a string of numeric characters into a decimal value.

REMARKS:

The VAL function converts a character string, which may include the hex number designator (&H), numbers (0–9), a sign (+, –), and exponential symbols (E), into a numeric value.

If the string is in decimal notation, it must be composed of the characters 0 to 9, with an optional decimal point and sign. In this form, VAL is the opposite of the STR\$ function.

If illegal characters are included, conversion is performed up to the first occurrence of an illegal character.

EXAMPLE:

```
A=VAL"-120"   Assigns -120 to variable A.
B=VAL"3.2*4=" Assigns 3.2 to variable B.
C=VAL"&H64"   Assigns 100 to variable C.
```

WAIT

P
D

FORMAT: 1. WAIT expression
2. WAIT

Abbreviation: W.
See Also: PRINT


PURPOSE:

Controls the length of time that displayed information is shown before program execution continues.

REMARKS:

Format 1 specifies the time in which execution of the PRINT command halts. The program temporarily halts for the specified time interval, then automatically restarts.

The value of the expression may be set to any value from 0 to 65535. A value of 1 as the expression corresponds to an interval of approx. 1/64 sec. When the computer is turned on or the RUN command is executed, WAIT0 (a wait time of zero) is assumed.

The WAIT command is valid for all the PRINT commands used in the program. To set an infinite interval, use format 2. If format 2 is used, the  key must be pressed to resume program execution.

Note:

In general, the WAIT command is not available on personal computers. On PCs, the FOR...NEXT statement is used for wait time control, for example:

```
50: FOR J=1 TO 500:NEXT J
```

EXAMPLE:

```
10: WAIT 64
```

Causes PRINT to wait about 1 second.

APPENDICES

CE-T801 Data Transfer Cable	A
Error Messages	B
Character Code Chart	C
Key Functions in BASIC	D
Troubleshooting	E
Memory Maps	F
Specifications	G
Using Programs from Other SHARP Computers	H
Care of the PC-E220	I

APPENDIX A

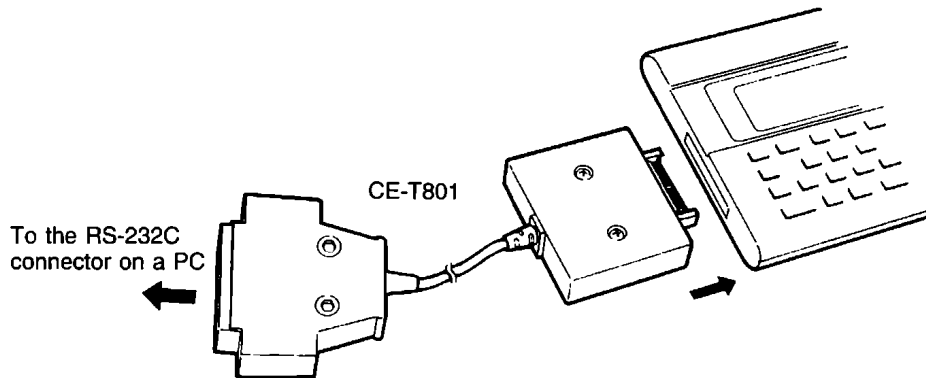
CE-T801 DATA TRANSFER CABLE

The CE-T801 is an optional serial communications cable designed to allow your PC-E220 to communicate with a personal computer or other serial device. It lets you transfer TEXT or machine code programs to and from your PC using the TEXT mode's SIO function or the Monitor's SIO commands.

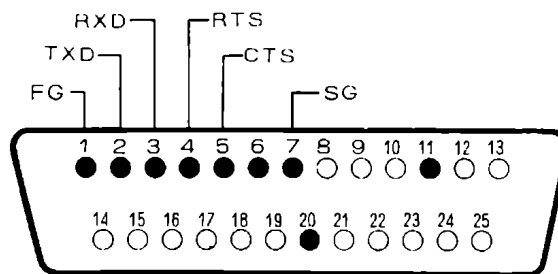
The CE-T801 contains a level converter that converts the RS-232C signal level to the signal level of the PC-E220. For serial communications with the PC-E220, a PC normally uses an RS-232C serial communications port. Because the serial interface of the PC-E220 does not conform to the RS-232C standard, it will be damaged if RS-232C signals are directly applied to it. The CE-T801 contains a level matching circuit to match the two signal levels, thus enabling serial communications.

Caution:

Do not touch the CE-T801 cable's connector pins with your bare hand. Static discharge from your body may damage the internal circuitry.



CE-T801 connector pin assignments (DB-25(W))



Pins 6 and 20 are connected.

Signal Description

RS-232C signal			Direction of signal viewed from the PC-E220	Description
Pin No.	Signal name	Symbol		
1	Frame ground	FG	—	
2	Send Data	TXD (SD)	Input	Data signal transferred to the PC-E220
3	Receive Data	RXD (RD)	Output (Note 2)	Data signal transferred from the PC-E220
4	Request To Send	RTS (RS)	Input	The PC transmits data when this line is set to High, and stops data transmission when it is set to Low.
5	Clear To Send	CTS (CS)	Output (Note 2)	Set to High when the PC-E220 is ready to receive data; set to Low when it is not ready.
7	Signal Ground	SG	—	Used to match the reference potential level between the input and output devices.
11	—	NC	—	Not used.

Notes:

1. The CE-T801 is designed to feed output signals (pins 2 and 4) from the PC to input pins of the PC-E220, and output signals from the PC-E220 to input pins (pins 3 and 5) on the PC. The cable can thus directly connect to an RS-232C connector on the PC.
2. The status of PC-E220 output signals are indefinite in any but the following cases:
 - (1) SIO is opened in the BASIC mode.
 - (2) R or W command is executed in the Monitor mode.
 - (3) Data is transferred through the SIO in the TEXT mode.

Communicating with a Personal Computer

For data transfer from a personal computer, a work area of approx. 300 bytes is required.

APPENDIX B

ERROR MESSAGES

When an error occurs, one of the error codes listed below will be displayed. For errors that occur during program execution, the error code is followed by the number of the line in which the error occurred.

Error code	Meaning
10	Invalid expressions or statements have been used.
12	An attempt was made to execute a command that is illegal as a program execution or direct input operation. The mode for PRO or RUN was selected incorrectly.
13	The CONT statement was executed illegally.
14	An attempt was made to designate a password for a program that does not exist.
20	The calculated result exceeds the calculation range.
21	An attempt was made to divide by zero.
22	An illegal operation was attempted.
30	An attempt was made to declare an array variable name which is already declared.
31	The array variable name was specified without the DIM statement.
32	Array was addressed illegally (array subscript exceeds the size of the array specified in the DIM statement)
33	The specified value exceeds the allowable range.
40	The specified line number or label does not exist.
41	The line number was specified illegally.
43	Invalid RENUM statement. (An old line number that does not exist was specified or line execution order was changed.)
44	The ending line number was specified with a number less than the starting line number in a statement such as LLIST or DELETE.
50	The levels of nesting in the GOSUB or FOR statement exceeds the allowable range.
51	An attempt was made to execute the RETURN statement without calling the subroutine.
52	The FOR statement is missing for the NEXT statement.
53	The DATA statement is missing for the READ statement.
54	The allowable number of data buffers (8) or function buffers (16) was exceeded.

Error code	Meaning
55	The length of the entered string exceeds 255 bytes. The line exceeds 255 bytes.
60	The size of program or variable exceeds the memory capacity.
70	Characters cannot be printed in the format specified in the USING statement.
71	The format specified in the USING statement is illegal.
80	Check sum error on the cassette tape or read-in error in the SIO.
81	Time-out error in cassette interface or SIO (specified wait time was exceeded during program or data I/O).
82	Data mismatch found during verify operation with the cassette tape or other media.
83	The type of variable specified in the INPUT# statement does not match the type of data read.
84	Printer-related error.
85	A device had not been opened when data transfer to or from it was attempted with the PRINT# or INPUT# command.
86	Attempt was made to open a device when another device was already in use.
87	Attempt was made to read more data after all data in the file had been read.
90	Attempt was made to assign characters to a numeric variable or numerals to a string variable, or a string variable was specified in a function that can only have numerals as the variable, such as SIN A\$.
91	A fixed variable to which numerals have been assigned was used as a string variable, or that to which characters have been assigned was used as a numeric variable.
92	Password mismatch.
93	Attempt was made to manually execute the MON command when a password has been set.
94	Specified file was not found.
95	Invalid filename.
96	Attempt was made to read a TEXT file in BASIC mode, or a BASIC file in TEXT mode.
97	The number of files exceeded 255.

APPENDIX C

CHARACTER CODE CHART

The character code chart shows the characters and their character codes used by the CHR\$ and ASC commands. Each character code consists of two hex characters (or eight binary bits). The most significant hex character (four bits) is shown along the top of the chart and the least significant hex character (four bits) is shown along the left side of the chart. If no character is shown, the entered character is illegal on the computer.

For example, the character "A" is hex 41 or decimal 65 or binary 01000001. The character "P" is decimal 80 or hex 50 or binary 01010000.

The character codes are represented as follows:

Examples:

Code for *

Hexadecimal &H2A

Decimal 42 (32 + 10)

Code for P

Hexadecimal &H50

Decimal 80

Note:

When printing on the CE-126P, codes &H00-&H1F and &H7F-&HFF are spaces.

Most Significant Four Bits

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Least Significant Four Bits	0	NL	space	0	@	P	'	p	space	Π	space	π	è	∫	ñ	c	
	1		!	1	A	Q	a	q	A	P	α	ρ	0	0	l	P	
	2		"	2	B	R	b	r	B	Σ	β	σ	1	1	x	e	
	3		#	3	C	S	c	s	Γ	T	γ	τ	2	2		o	
	4		\$	4	D	T	d	t	Δ	Υ	δ	υ	3	3	▶	◀	
	5		%	5	E	U	e	u	E	Φ	ε	φ	4	4	≤	▶	
	6		&	6	F	V	f	v	Z	X	ζ	χ	5	5	≥	°	
	7		'	7	G	W	g	w	H	Ψ	η	ψ	6	6	↕	■	
	8		(8	H	X	h	x	Θ	Ω	θ	ω	7	7	[ε	
	9)	9	I	Y	i	y	I	N	ι	ι	8	8]	⋮	
	A		*	:	J	Z	j	z	K	n	κ	φ	9	9	.	^	
	B		+	;	K	[k	{	Λ	η	λ	ȳ	-	Å	√	→	
	C		,	<	L	\	l		M	μ	μ	≠	+	ª	≠	↔	
	D		—	=	M]	m	}	N	ħ	v	°C	∞	b	À	↑	
	E		.	>	N	^	n	~	Ξ	∂	ξ	1/2	±	e	B	↓	
	F		/	?	O	_	o		O	ö	o	x̄	≠	m	e	↵	

Codes &H01-&H1F and &H7F are spaces.

APPENDIX D

KEY FUNCTIONS IN BASIC

BREAK
ON

- Use to turn the power on when the power has been turned off by the Auto OFF function.
- Pressing this key during program execution functions as a **BREAK** key and causes program execution to be interrupted.
- When pressed during direct input operation, execution of CLOAD will be interrupted.
- In the STAT mode, the key is used to return to the submenu.
- In the TEXT and ASMBL modes, the key is used to return to the Main Menu or Menu.

SHIFT
2nd F

- Either the yellow key marked "SHIFT" or "2nd F" must be pressed (and held for "SHIFT") to use a key's second function (indicated immediately above the key).

Example: **SHIFT** + **Y** → & is entered.
(**2nd F** **Y**)

C•CE

- Use to clear the contents of the entry and the display.
- Use to reset after an error.

SHIFT + **CA**

- Not only clears the display contents, but resets the computer to its initial state.
 - Reset: the WAIT timer.
 - the display format (USING format).
 - the TRON state (TROFF).
 - PRINT=LPRINT.
 - an error condition.

BASIC

- Use to change the operational submode from another mode to RUN or from PRO to RUN.

ANS

- Use to recall the last answer.

TAB

- Use to advance the cursor to the next tab position. In the RUN or PRO mode, the key advances the cursor in seven-column increments. In the Text Editor mode, it first advances the cursor eight columns, then six columns, and seven columns thereafter.

CONST

- Use to set a constant and operators for constant calculations ("CONST" indicator appears on the display). **2nd F** **CONST** (**SHIFT** + **CONST**) displays the currently set constant.

SHIFT + **<**
SHIFT + **>**

- Use when entering logical operations in IF statements.
- Use to specify relational expressions.

- ;
 - Use to provide multi-display (two or more values displayed at a time).
 - Use to separate commands and variables.

- SHIFT + :
 - Use to separate more than one statement defined on a single line.

- ,
 - Use to provide multi-display (two or more values displayed at a time).

- ▶
 - Shifts the cursor to the right.
 - Executes playback instructions.
 - Calls the cursor if it is not displayed while the contents are displayed.
 - Clears an error in direct input operation.

- ◀
 - Shifts the cursor to the left.
 - Otherwise the same as the ▶ key.

- INS
 - Inserts a single space at the current cursor position.

- DEL
 - Deletes the character at the cursor position.

- BS
 - Deletes the character to the left of the cursor.



- ↵
 - Enters a program line into the computer.
 - Use when writing programs.
 - Requests manual calculation or direct execution of a command statement by the computer.
 - Use to restart a program temporarily stopped by an INPUT command.

- SHIFT + P-HP
 - Use to set print or non-print mode when the optional printer is connected.

- CAPS
 - Toggles the uppercase mode (the CAPS symbol appears on the display).
 - When the CAPS symbol is displayed, letters will be entered in uppercase. If CAPS is pressed, the CAPS symbol disappears and lowercase characters will be entered.

- SHIFT + &
 - Use to indicate a hexadecimal value.

The ↓ and ↑ keys have the functions listed in the following table, depending on the mode, as well as the state of the computer:

Mode	State		
RUN	Program being executed	Not functional	
	Interrupted by the STOP command or the BREAK key	Executes the next line and stops.	Hold down to display program line being executed or already executed.
	Error condition during program execution	Not functional	Hold down to display error-producing line.
	Trace mode ON	Executes program in Trace mode.	Hold down to display program line being executed or already executed.
PRO	(When the mode is changed to PRO mode and program lines are not displayed)		
	Program is temporarily interrupted	Displays the line interrupted.	Same as at left
	Error condition	Displays the line with the error.	Same as at left
	Other condition	Displays the first line.	Displays the last line.
	(When the program lines are displayed)		
	Displays the next program line.	Displays the preceding program line.	

Note:

If no key is pressed in the key entry request mode for approximately eleven minutes, the power is automatically turned off (Auto OFF function).

APPENDIX E

TROUBLESHOOTING

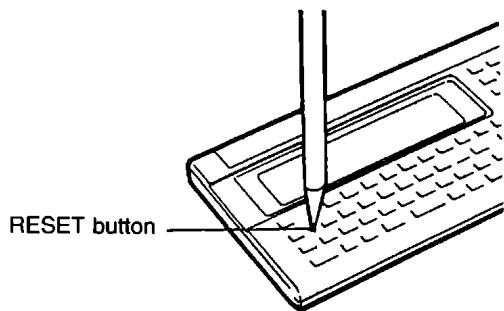
This appendix provides you with some hints on what to do when your computer does not do what you expect it to. You should try each of the following suggestions, one at a time, until you have corrected the problem.

1. If the display is too light or too dark:
 - adjust the contrast control.
2. If the power does not come on (nothing is displayed):
 - the batteries may be exhausted. Replace the batteries.
 - the memory protect switch may be set to position B. Check that the switch is set to position A.
3. If the power does not turn off:
 - the computer is running a program using a command that takes a long time, such as CSAVE or CLOAD. Press the **BREAK** key to interrupt program execution, then the **OFF** key.

If the power will still not turn off, perform the following operation 4.
4. If the computer does not operate properly,
 - a peripheral device may have been connected or disconnected while the power was on, or there may have been an error during program execution, or the computer may have been subjected to strong electrical interference or shocks during use.

Perform one of the following operations:

Reset Operation
Press the RESET button with a ball-point pen or any other appropriate device.



Use only a ball-point pen or similar device to press the RESET button. Do not use a mechanical pencil with the lead exposed or a device with a sharp point, such as a sewing needle.

When you release the RESET button, the following message will appear. If any other message appears, press the RESET button again. (You will be prompted to make sure you want to clear the entire contents of memory.)

```
MEMORY CLEAR O.K.? (Y/N)
```

- If you don't want to clear the program or data stored in memory, press the N key (or any key other than Y). The display will return to the opening screen in the RUN mode.

- If an error occurs again when the program is run, do the following:

- Clearing All Contents of Memory

When the above message is displayed, press the Y key. The entire contents of memory will be cleared and the following message will flash on the display (this shows that the computer has been initialized and the entire contents of memory cleared):

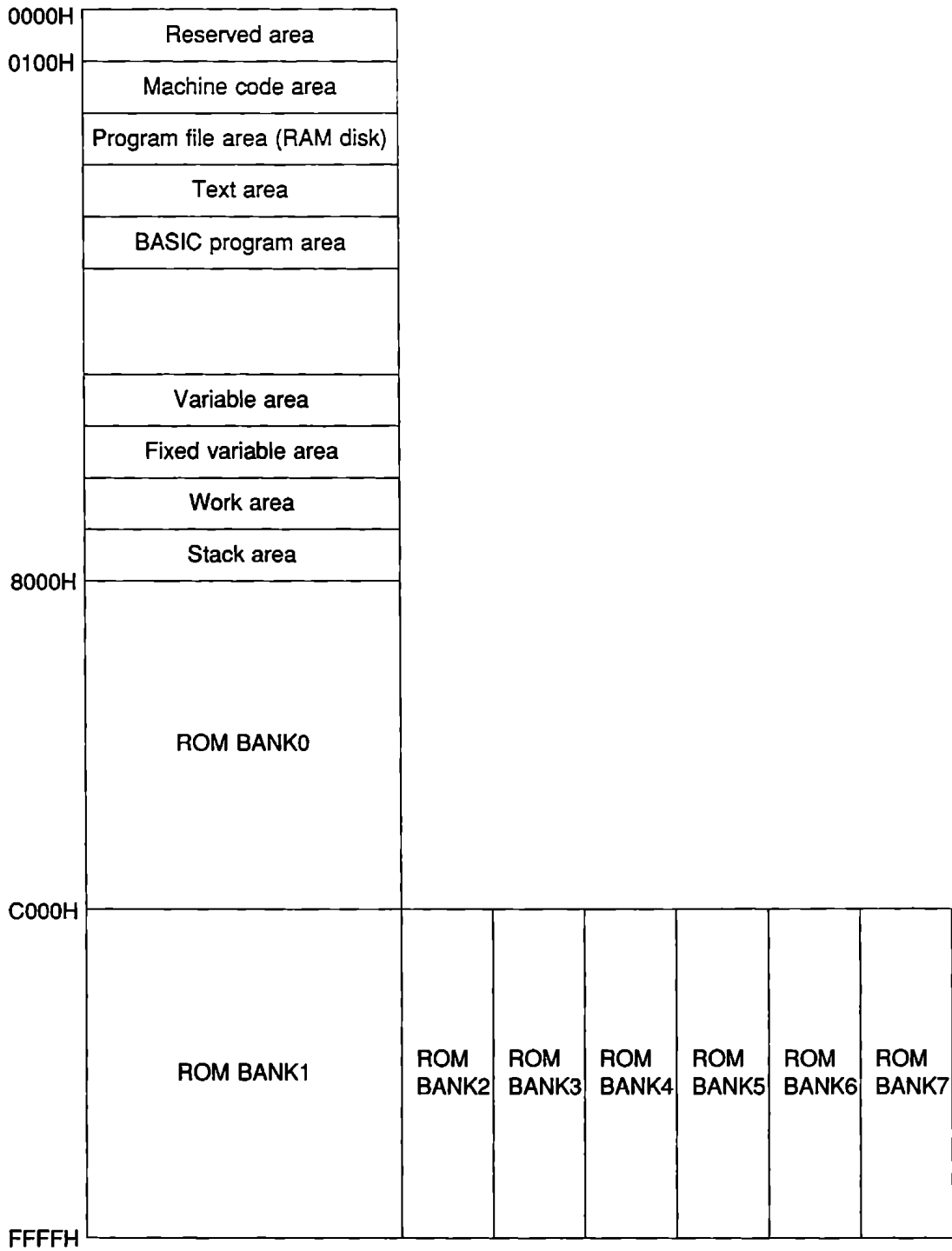


Press any key to return to the opening screen in the RUN mode.

APPENDIX F

MEMORY MAPS

Memory Map



APPENDIX G

SPECIFICATIONS

Processor:	8-bit CMOS CPU (Z80A equivalent)	
Memory capacity:	System internal	2.1K bytes approx.
	Fixed variable area	208 bytes
	Program/data area	30435 bytes
Stack:	Subroutine stack:	10 buffers
	Function stack:	16 buffers
	FOR-NEXT stack:	5 buffers
	Data stack:	8 buffers
Operators:	Addition, subtraction, multiplication, division, trigonometric and inverse trigonometric functions, logarithmic and exponential functions, angle conversion, square and square root, power, sign, absolute, integer, coordinate conversion, pi, etc.	
Numeric precision:	10 digits (mantissa) + 2 digits (exponent)	
Editing features:	Cursor left and right, line up and down, character insert, character delete Text editor, Z80 machine language monitor	
Memory protection:	Battery backup	
Interface capability:	11-pin (for cassette interface, printer, SIO device)	
Display:	4-line, 24-column liquid crystal display with 5 × 7 dot matrix.	
Power supply:	For computer operation: 6.0 Vdc Type-AA dry cell battery (R06) × 4	
	For memory backup: 3.0 Vdc Lithium battery (CR2032) × 1	
Power consumption:	0.37 W	
	Approximately 80 hours of continuous operation under normal conditions (based on 10 minutes of operation or program execution and 50 minutes of display per hour at a temperature of 20°C/68°F).	
	Note: When the computer is used for serial communications through the optional CE-T801 Data Transfer Cable, the number of hours the unit can be operated continuously will drop to approx. 48 hours (when used for 2 min. of communications, 8 min. of calculation or program execution, and 50 min. of display per hour at an ambient temperature of 20°C/68°F).	

The operating time may vary slightly depending on usage and the type of battery used.

Operating temperature: 0° – 40°C (32° – 104°F)

Dimensions: 215(W) × 100(D) × 18(H) mm
8-15/32"(W) × 3-15/16"(D) × 23/32"(H)

Weight: 300 g (0.66 lb.) (with batteries)

Accessories: Hard cover, four AA batteries, one lithium battery, and Operation Manual.

Options: Printer/Cassette Interface (CE-126P)
Data Transfer Cable (CE-T801)

APPENDIX H

USING PROGRAMS FROM OTHER SHARP COMPUTERS

Programs written for the following SHARP PC series computers can be run on the PC-E220 computer with slight modifications:


PC-1403(H), PC-1460, PC-1425

The PC-E220's WAIT command has an initial default value of zero. This means that the display will not be temporarily frozen when the PRINT command is executed. If you wish to display more than four lines at a time, insert a WAIT command in the transferred program to temporarily freeze the display (see "WAIT" command). Since the PC-E220 has 24 display columns, the transferred program may require modifications to fit the PC-E220's display width. Programs containing commands or characters not defined on the PC-E220 will also require modification.

- **Memory Capacity**

A program written on another computer requires a different amount of memory on the PC-E220.

- **User-Defined Keys**



The PC-E220 has no user-definable keys for program execution. As an alternative for user-defined keys, enter GOTO *label  instead. Use *label rather than "label" in the GOTO statement.

- **Line Numbers**

Variables or expressions cannot be used for line numbers specified in the GOTO, GOSUB, RESTORE, or THEN statement.

- **Cassette Tape Recorder**

When loading a program from another computer into the PC-E220, use one of the following commands:

CLOAD@"filename"  or CLOAD@

If the filename is omitted, the program first encountered after the tape is started will be loaded. The PC-E220 automatically converts loaded program codes into PC-E220 program codes as it reads each line. An error will occur if a converted program line exceeds 255 bytes. An error will also occur if the size of the loaded program is too large for the PC-E220's program area. If this happens, clear unnecessary variables from the program area, then reload the program. For code conversion, a work area of approx. 300 bytes is required, in addition to the program space.

When inputting/outputting data from/to cassette tape, the OPEN command must be executed.

Example:

```
20 PRINT #F,G → 10 OPEN "CAS:" FOR OUTPUT
                 20 PRINT #1,F,G
                 30 CLOSE #1
```

- **Array Variables**

If array A() is used in a program written on another computer without first being declared with a DIM statement, the array must be declared at the beginning of the program. Fixed variable space is not shared with array variable A().

- **Miscellaneous**

1. For logical operations, the PC-E220 returns value -1 for true, and 0 for false.
2. It is not allowed to use expressions in the DATA statement, such as DATA SIN 30+2, CHR\$66. Use the statement in the form of, for example, DATA 2.5, B.

APPENDIX I

CARE OF THE PC-E220

To ensure trouble-free operation of your computer, note the following:

- Always handle the computer carefully, as the liquid crystal display is made of glass.
- Keep the computer away from extreme temperatures, as well as moisture and dust. During warm weather, temperatures in vehicles left in direct sunlight will rise sharply. Prolonged exposure to high temperature may damage 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 the computer is subjected to strong static electricity or external interference, it may "hang up" (all keys will become inoperative). If this happens, press the RESET button. (See TROUBLESHOOTING)
- Keep this manual for future reference.

COMMAND INDEX

ASC	148	REM (')	196
BEEP	148	RENUM	197
CHR\$	149	RESTORE	198
CLEAR	150	RIGHT\$	199
CLOAD	151	RND	200
CLOAD?	152	RUN	201
CLOSE	153	SAVE	202
CLS	154	STOP	203
CONT	154	STR\$	204
CSAVE	155	TROFF	205
DATA	156	TRON	205
DEGREE	157	USING	206
DELETE	158	VAL	207
DIM	159	WAIT	208
END	161		
FILES	162		
FOR...NEXT	163		
FRE	164		
GOSUB...RETURN	165		
GOTO	166		
GRAD	167		
IF...THEN	168		
INKEY\$	170		
INPUT	172		
INPUT#	173		
KILL	174		
LEFT\$	174		
LEN	175		
LET	176		
LFILES	176		
LIST	177		
LLIST	178		
LOAD	179		
LOCATE	180		
LPRINT	181		
MID\$	182		
MON	182		
NEW	183		
ON...GOSUB	184		
ON...GOTO	185		
OPEN	186		
PASS	187		
PRINT	188		
PRINT#	191		
RADIAN	193		
RANDOMIZE	194		
READ	195		

INDEX

A

- ASCII code 98, 109
- ASMBL mode 9, 121
- Array variables 77
- Assembler 118
 - assembling 126
 - errors 134
 - list 129
 - pseudo instructions 130
 - reserving area 120
 - source program coding and editing 122
- Auto OFF 8

B

- BASIC
 - commands 85
 - concepts and terms 74
 - converter 9
 - mode 9
 - program operation 73
 - reference 135
 - statements 84
- Basic operations 26
- BATT indicator 11, 12
- Batteries
 - handling 15
 - operating 7, 12
 - memory backup 7, 14
 - replacement 12
- BUSY indicator 10

C

- CAL mode 9, 22
- Calculations
 - errors 71
 - length 66
 - ranges 145
 - scientific 27, 66, 136
 - serial 62
 - statistical 50
- CAPS indicator 10
- Care of computer 226
- Cassette recorder
 - interface 16, 17
 - specifications 17

- Character codes 215
- Command reference
 - BASIC 135
 - machine language 112
- Complex numbers 48
- Communication parameters 105
- CONST indicator 10
- Constants
 - calculations 63
 - physical 43
 - string 74
- Contrast 11
- Conversions
 - angle/time 29, 68
 - metric 45
 - polar/rectangular 30, 68
 - TEXT/BASIC 109
- Cursor 10

D

- Data transfer cable 16, 210
- Debugging 94
- Decimal places 24, 31
- DEG indicator 10
- Degree 25, 66
- Devices, peripheral 16
- Direct calculation 69
- Direct command 86, 147
- Direct input 21
- Display 6, 10
- Display mode 24

E

- E indicator 11
- 11-pin connector 6, 16
- Engineer software (ENG) 9, 34
 - list 35
- Error messages 117, 212
- Errors 62, 71, 134
- Expressions 80
 - logical 82
 - relational 81
 - string 81
- Extensions of filenames 80

F

Factorization 34
 formulas 36
Filenames 80
Files, program 80
Fixed variables 76
Formulas
 factorization 36
 integration 40
 trigonometric 38
Free memory (FRE) 3

G

GRAD indicator 10
Gradient 25, 66
Greek alphabet 42

H

Hardware 6, 8
Hexadecimal numbers 74

I

I/O, serial 16, 104
 command table 16
Initializing the computer 2
Integration 34
 formulas 40

K

Key functions 216
Keyboard 6

L

Labels 85, 124
Last answer recall 65
LCD screen 6, 10
 contrast dial 6, 11
Line numbers 84
Logical expressions 82

M

M indicator 11
Machine language monitor 111
 command reference 112
 error messages 117
Memory
 available 3
 calculations using 27
 maps 221

Metric conversion 45
Modify function 32
Monitor mode 111

N

Names, file 80
Numeric operators 26

O

Object program 118, 122
Operation modes 9
 selection 9
Operator precedence 32, 70, 83
Operators
 logical 82
 numeric 26, 80
 relational 69, 81

P

Parentheses 31, 33, 83
Peripheral devices 16
Physical constants 43
Precedence 32, 70, 83
PRINT indicator 11
Printer/cassette interface 16, 19
Printing, direct input 71
Priority levels 32, 70, 83
PRO mode 9, 73, 86
 indicator 11
Program
 entering 86
 execution 92
 files 80
 from other computers 224
 listing 87, 90
 storing 92
Programming 84, 86
Prompt 10, 113
Protective cover ii
Pseudo instructions 130

R

RAD indicator 10
Radians 25, 66
RAM disk i, 80, 221
Relational operators 69, 81
RESET
 button 7
 operation 2, 219

RUN mode 9, 11, 58
 calculations in 58
 indicator 11

S

Scientific
 calculations 27, 66, 136
 notation (SCI) 24
Second function (2nd F)
 indicator 10
 key 9
Serial calculations 62
Serial I/O 98, 104
Shift key 9
Single-variable statistics 50
Source program 118, 123
Specifications 222
STAT mode 11, 50
 indicator 11
Statement 84, 123
Statistics
 calculations 50, 54
 data deletion 51
 data entry 51, 55
 printing 54
 single-variable 50
 two-variable 54
Status line 10
Storing programs 92

T

TAB key 100
Tape
 notes 19
 recording 17
TEXT mode 98
 communication parameters 105
 editor 99
 functions 98
 indicator 11
TEXT/BASIC conversion 109
Trace mode 94
Trigonometric calculations 28, 66, 67
 formulas 38
Troubleshooting 219
Turning on 8

V

Variables 75
 fixed 76
 numeric 76
 simple 77
 string 76
 types 76
 using in calculations 64
 using in programs 89

W

Wild card 102

SHARP CORPORATION

OSAKA, JAPAN

1991 (C) SHARP CORPORATION
PRINTED IN JAPAN/IMPRIMÉ AU JAPON

1E1KS(TINSE1204ECZZ)2