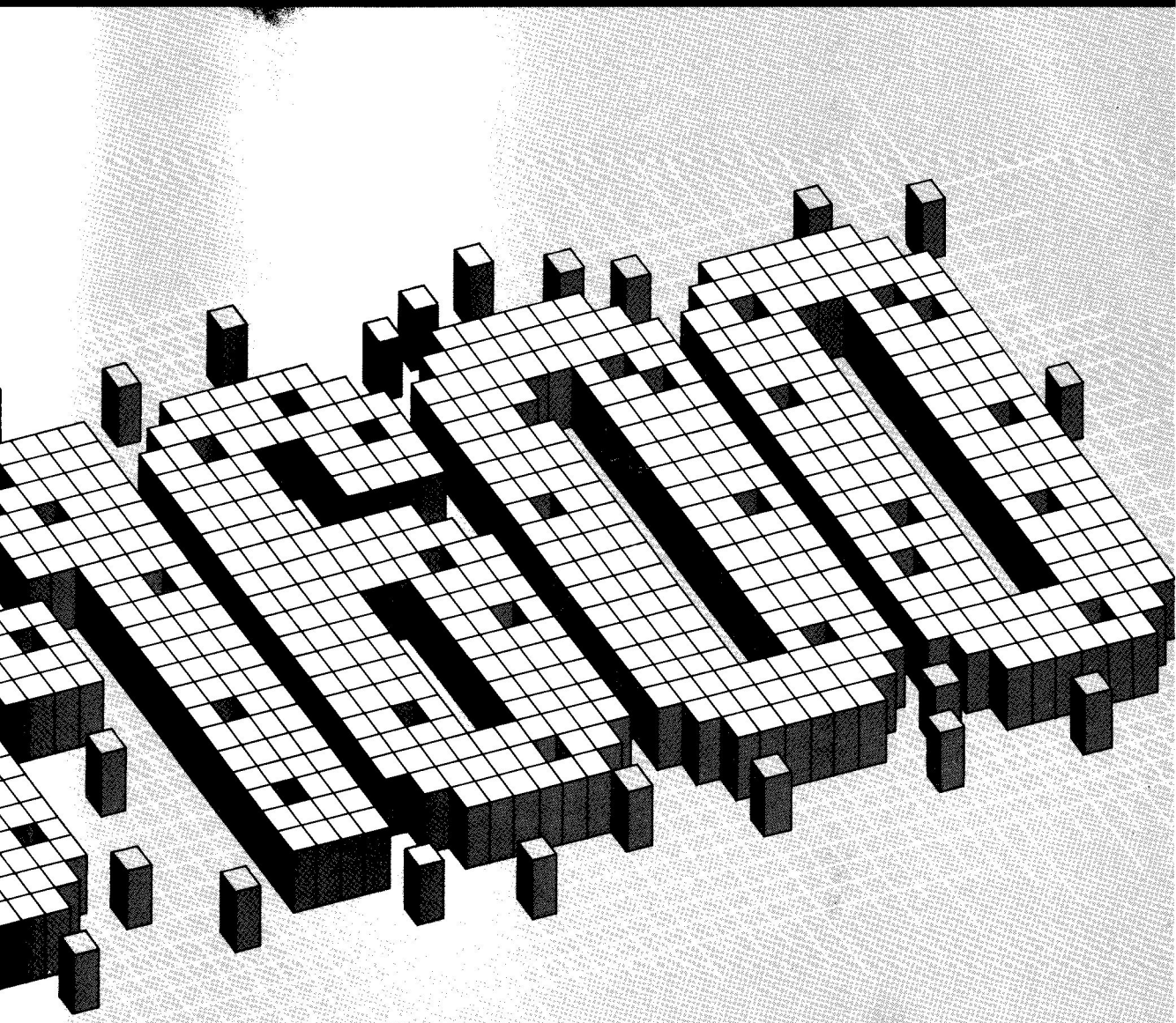
SHARP®

POCKET COMPUTER

MODEL
PC-1600

OPERATION MANUAL



WARNING: THIS EQUIPMENT HAS BEEN CERTIFIED TO COMPLY WITH THE LIMITS FOR A CLASS B COMPUTING DEVICE, PURSUANT TO SUBPART J OF PART 15 OF FCC RULES. ONLY PERIPHERALS (COMPUTER INPUT/OUTPUT DEVICES, TERMINALS, PRINTERS, ETC.) CERTIFIED TO COMPLY WITH THE CLASS B LIMITS MAY BE ATTACHED TO THIS COMPUTER. OPERATION WITH NON-CERTIFIED PERIPHERALS IS LIKELY TO RESULT IN INTERFERENCE TO RADIO AND TV RECEPTION.

This equipment generates and uses radio frequency energy and if not installed and used properly, that is, in strict accordance with the manufacturer's instructions, may cause interference to radio and television reception. It has been type tested and found to comply with the limits for a Class B computing device in accordance with the specifications in Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference in a residential installation. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause interference to radio or television reception which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

Reorient the receiving antenna.
Relocate the computer with respect to the receiver.
Move the computer away from the receiver.
Plug the computer into a different outlet so that computer and receiver are on different branch circuits.

If necessary, the user should consult the dealer or an experienced radio/television technician for additional suggestions. The user may find the following booklet prepared by the Federal Communications Commission helpful:
"How to Identify and Resolve Radio-TV Interference Problems".
This booklet is available from the U.S. Government Printing Office, Washington, D.C., 20402,
Stock No. 004-000-00345-4
BILLING CODE 6712-01-M

A shielded I/F cable is required to insure compliance with FCC regulation for Class B computing equipments.

FOR YOUR RECORDS....

For your assistance in reporting this electronic calculator in case of loss or theft, please record below the model number and serial number which are located on the bottom of the unit.
Please retain this information.

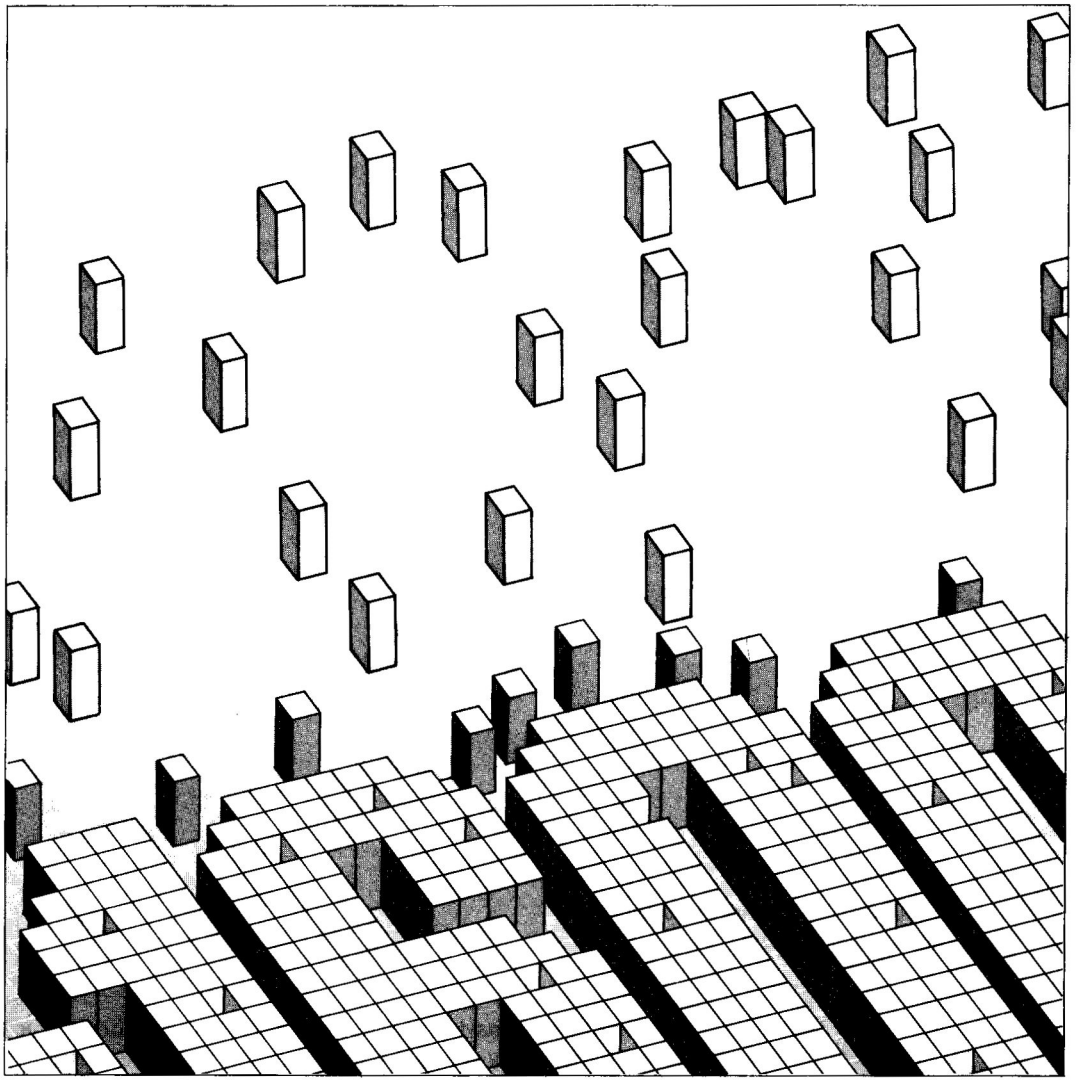
Model Number _____ Serial Number _____

Date of Purchase _____ Place of Purchase _____

POCKET COMPUTER

PC-1600

OPERATION MANUAL



SHARP PC-1600 Pocket Computer
Operation Manual

CONTENTS

INTRODUCTION

Congratulations for selecting the SHARP PC-1600 Pocket Computer. The PC-1600 presents the ultimate in portability and versatility. On its own, the PC-1600 fits easily in the palm of your hand, yet packs the computing power of a computer many times its size. Connected to a full range of specially designed options that are likewise small in size yet provide the functions and features of standard peripherals, your PC-1600 can be expanded into a compact system to meet a variety of needs.

Here are some of the standard features that make your PC-1600 the perfect choice for both experienced users and novices alike:

- Powerful CMOS 8-bit microprocessor, equivalent to Z-80A
- Standard 16K byte random access memory (RAM), expandable to 80K
- 26 column, 4 line LCD with a 5×7 character matrix
- 156×32 dot graphics, all dots addressable
- Real-time clock supports wakeup and alarm features

Your PC-1600 operates on a specially adapted version of BASIC which is contained in the computer memory and lets you write your own programs easily and quickly.

The PC-1600 is battery-powered for convenient on-the-go portability and can also use an AC adapter for computing at home or office.

You can also connect the PC-1600 directly to external I/O devices via the three built-in interface ports: RS-232C and optical serial I/O ports and an analog input port.

The PC-1600 can be used with the following optional accessories:

- CE-1600M Program Module provides 32K RAM expansion with battery backup for storing programs, expanding your memory workspace, or creating a RAM disk.
- CE-1600P Printer with Cassette Interface for four-color printouts of text and graphics allows connection of a cassette recorder for storing programs and data on tape.
- CE-1600F Disk Drive using 2.5″ double-sided Pocket Disks that can store up to 61K per side.

The PC-1600 is also compatible with the lineup of SHARP PC-1500 options that include printer, cassette recorder, RAM modules, interfaces, and BASIC programs.

USING THIS MANUAL

For convenient access to information on your SHARP PC-1600 Pocket Computer, we have arranged this manual in five parts:

PART I, Getting Started, gives a general description of the PC-1600 to get you acquainted with its parts and features. This is definitely a "must read" section.

PART II, Operation, takes you from the initial setup of the PC-1600 including battery installation, turning the computer on and off, and setting the time and date of the built-in real-time clock. It also shows you how to use the edit keys when you type in data, and concludes with a chapter on using the PC-1600 as a calculator.

PART III, Using the Interfaces and Optional I/O Devices, describes the use of the PC-1600's three interface ports and memory module slots. It also provides information on how to use the PC-1600 with its optional printer, floppy disk drive, and cassette recorder. Use of the compatible PC-1500 peripherals is also discussed in the last chapter.

PART IV, Basic Reference Section, describes BASIC, the built-in programming language that operates the PC-1600 and lets you write your own programs and control the optional equipment. This section simply introduces general concepts of BASIC and its use on the PC-1600, but will not try to teach you the "basics of BASIC." We suggest that you refer to one of the many excellent primers available on this popular language. This part includes a complete dictionary of the BASIC commands and statements that can be used on the PC-1600.

PART V, Appendices, includes information on changing the batteries and RAM modules. It also provides various technical information and summaries of error codes, specifications, memory maps, and a BASIC command list.

We suggest that you familiarize yourself with this manual before operating the computer to prevent problems later on. Part I, Getting Started, is especially important as it introduces the features of the PC-1600. Other sections may be checked, then used again in a reference style as necessary.

PART I

GETTING STARTED

We begin this part with an overall description of the PC-1600 after you've unpacked the computer and its accessories and checked that the carton contents are complete.

The individual special features of the PC-1600 are then introduced with a focus on the keyboard and key functions.

Finally, the last section ends with a software overview of the PC-1600's operation and the role of BASIC.

1 Description of the PC-1600

We know that you'd like to start using your new PC-1600 as soon as possible. To use it properly from the beginning and avoid future problems, please take a few minutes to look over the following pages, and your PC-1600, before skipping ahead to any other section of this manual.

Unpacking Notes

After you've unpacked your PC-1600 and its accessories, double-check the contents against the following checklist to make sure that everything was included in the package.

If anything is missing or in any way damaged, contact your SHARP dealer immediately.

Your PC-1600 package should contain:

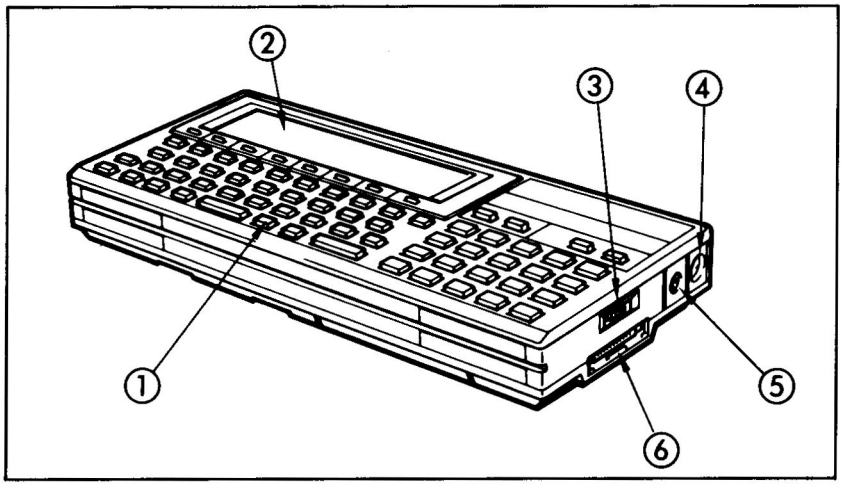
- The SHARP PC-1600 Pocket Computer
- A soft case
- Two keyboard templates
- Four AA size batteries
- This operation manual

When everything has checked out OK, be sure to return the packing materials to the carton and save it for future use. You may at some time be in need of repacking your PC-1600 for shipping or long-term storage.

Hardware Overview

The PC-1600 features a QWERTY style keyboard layout similar to conventional typewriters, a liquid crystal display (LCD) with adjustable contrast, two module slots for memory expansion, and three interface connectors for attaching various optional equipment.

The following pages describe the individual parts of the PC-1600 to acquaint you with their positions and functions.



Front View of the PC-1600

1 **Keyboard** With its standard typewriter-style layout and numeric keypad, the keyboard features a total of 69 keys and incorporates a number of special purpose and programmable function keys.

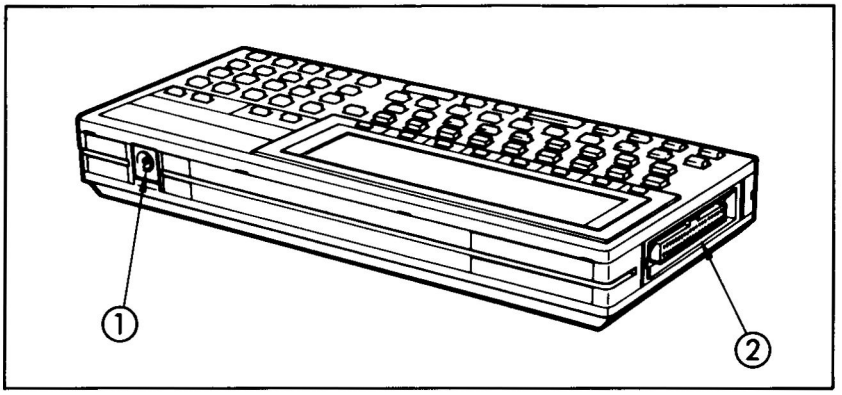
2 **LCD Screen** The PC-1600 display has four lines of 26 columns per line and its contrast is adjustable for comfortable viewing.

3 **LCD Contrast Dial** When the operating position of the computer or lighting conditions change, adjust this dial to lighten or darken the screen for easier viewing.

4 **Optical Serial I/O Port** Use this connector to attach an optical fiber cable from an external device for high-speed, noise-resistant serial communications.

5 **Analog Input Port** This port allows the PC-1600 to receive data from analog sources such as sensors.

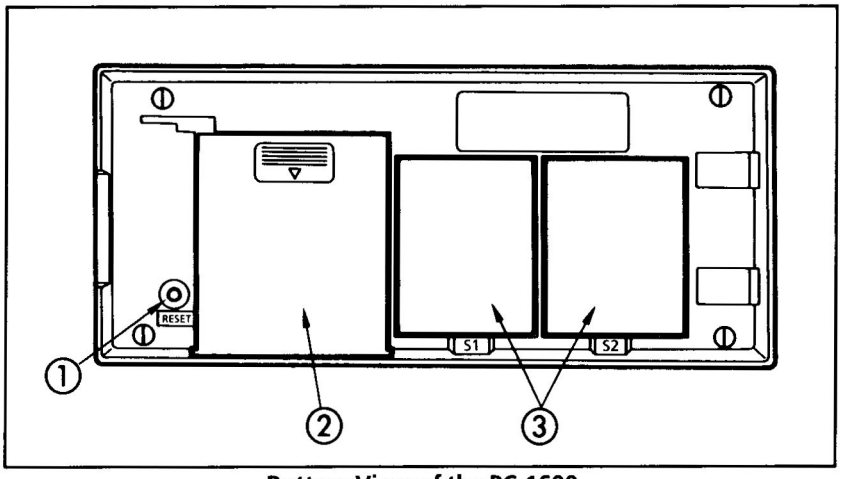
6 **RS-232C Port** Connect a cable to this port from another computer, modem, or printer to send and receive serial data.



Rear View of the PC-1600

1 **AC Adapter Jack** When using the PC-1600 indoors or to conserve the batteries, plug an AC adapter (EA-160 or EA-150) into this connector to allow the computer to receive power from a standard electrical outlet.

2 **System Bus** The bus connector provides a direct link to the computer system for connecting optional printer and interface units and controlling data transmission between them and the computer.



Bottom View of the PC-1600

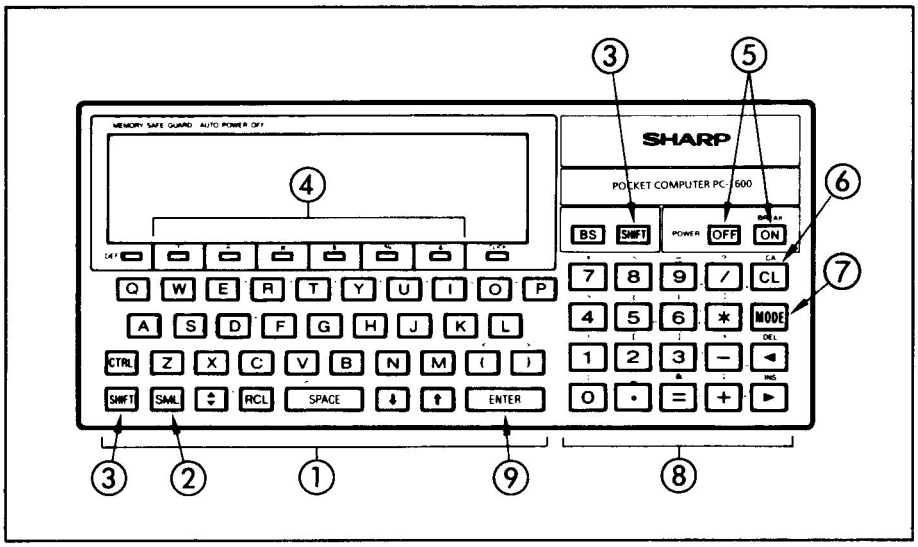
1 **Reset Switch** If the PC-1600 should "lock-up" for some reason during operation, the screen and keyboard will be inoperative. You will be unable to use the OFF and ON keys to restart the computer. Pressing the reset switch clears the PC-1600's memory and restores the computer to the condition after power-on. Use this switch with caution as you may risk the loss of data and programs. See Resetting the Computer in Chapter 3 before using.

2 **Battery Compartment** When the PC-1600 is not connected to AC power with an optional adapter, it is powered by four AA size batteries housed in this compartment. See Chapter 2, Supplying Power to the PC-1600.

3 **Module Slots** Optional RAM expansion modules can be plugged into these two slots to increase the memory workspace, save and load programs, or create a RAM disk.

The PC-1600 Keyboard

This section introduces some important features of the PC-1600 keyboard. Key functions related to BASIC are described later in this manual in Chapter 9, Operating Modes under BASIC, in the Key Operation section, and the use of the CTRL (Control) key in the Edit mode section. Simple screen editing is also described in Chapter 4 on the PC-1600 Display.



The PC-1600 Keyboard

1 **Alphabetic Keys** The alphabetic keys are similar to those found on a standard typewriter and are used for entering programs and data. These keys have an auto-repeat function which continually repeats the character of the key when you hold it pressed down. A BASIC command allows you to turn this function on and off. See KEYSTAT in the Command Dictionary.

2 **Small Key** The SML (Small) key allows you to switch from the computer's default setting of upper case to lower case when you type in alphabetic keys. This also reverses the use of the SHIFT key, as described above. When you've pressed the SML key and are in the Small mode, pressing the SHIFT key before each character entry produces an upper case character.

3 **Shift Keys** For convenience, two SHIFT keys are provided on the PC-1600; both have identical functions. As on a typewriter, pressing the SHIFT key before another gives an alternate character, function or symbol. The PC-1600 normally prints upper case characters on the screen. This is the computer's default setting, unless you specify otherwise. You can produce lower case letters by pressing the SHIFT key before each character key you press. (See the description of the SML (Small) key that follows.) The SHIFT keys also have other special functions when used in combination with other keys in BASIC. These are described in the BASIC Reference Section.

4 **Function Keys** These six keys are at the top of the keyboard, just under the display, and are used for assigning and recalling with fewer keystrokes frequently used commands and keywords. Use of these keys and combinations with other related keys are described in the section on BASIC key functions.

5 **On and Off Keys** These keys are used to turn the computer on and off. While the computer is operating, the ON key functions as the BREAK key which is used to interrupt the execution of a program.

6 **Clear Key** This key deletes the contents of the line the cursor is on and has other features when shifted to the CA key in BASIC. See the key descriptions in the BASIC Reference Section.

7 **Mode Key** This key is used to switch between the RUN and PROGRAM operating modes of the PC-1600. The third mode, RESERVE, is selected when the MODE key is shifted (SHIFT + MODE).

8 **Numeric and Arithmetic Operation Keys** These keys are used when you perform manual calculations and for entering numeric data in programs. The +, −, * and / keys return operations equivalent to addition, subtraction, multiplication, and division. The use of these keys is further discussed in Chapter 5, Calculating on the PC-1600.

9 **Enter Key** Pressing the ENTER key after other keystrokes signals the end of the entry to the computer. The function of this key may be compared to the Return key used on typewriters.

Software Overview

Your PC-1600 pocket computer is based around a version of the Z-80A microprocessor. This microprocessor operates from commands written in machine code. The BASIC programming statements which you will write and input to the PC-1600 are translated internally line by line as your program is executed. This process is carried out by the BASIC Interpreter built into the computer's read only memory (ROM). Except for a few system commands and the internal calculation functions which work directly into machine code subroutines, BASIC is the language with which you communicate your instructions to the PC-1600.

The version of BASIC used in the PC-1600 is SHARP's development of a specially enhanced BASIC for use on pocket computers.

Programs written in BASIC for other computers can usually by adapted with some changes and a little ingenuity to run on the PC-1600. There is a separate section in this manual on compatibility with the version of BASIC used in the PC-1500, including a list of commands which must be changed or modified.

Your PC-1600 will not work as a word processor in the form set up, but the descriptions of the LOAD* and SAVE* statements include hints on how to make use of BASIC program lines to store text rather than programs. In this way the PC-1600 cna be a powerful and compact data base. Of course a text editor program written in BASIC will allow you to edit text as well as store it. There are many good books available with ideas on useful programming projects in addition to the sample programs included with this manual.

For advanced programmers, it is possible to bypass the BASIC Interpreter and write programs directly in machine language. There are a number of commands in the set which allow you to manipulate machine language programs (PEEK, POKE, CALL, BLOAD, BSAVE, etc.). These commands should be used with caution and only by those with a through understanding of the memory allocation of the PC-1600.

PART II

OPERATION

The first chapter in this part starts you off with the practical aspects of using your PC-1600. You'll learn how to get power to the computer, what happens when you turn it on and off, and how to reset it to initialize the system and if trouble occurs.

The next chapter deals with the features of the PC-1600's display and shows you how to set the time and date of the built-in real-time clock.

The final chapter describes how to use the PC-1600 as a calculator and perform simple calculations.

2 Supplying Power to the PC-1600

Your PC-1600 can operate on batteries as well as AC power when an optional AC adaptor is used.

Installing the Batteries

Four AA size, 1.5 volt batteries are included as standard accessories with the PC-1600. Note that these are common dry cell type batteries and are not rechargeable.

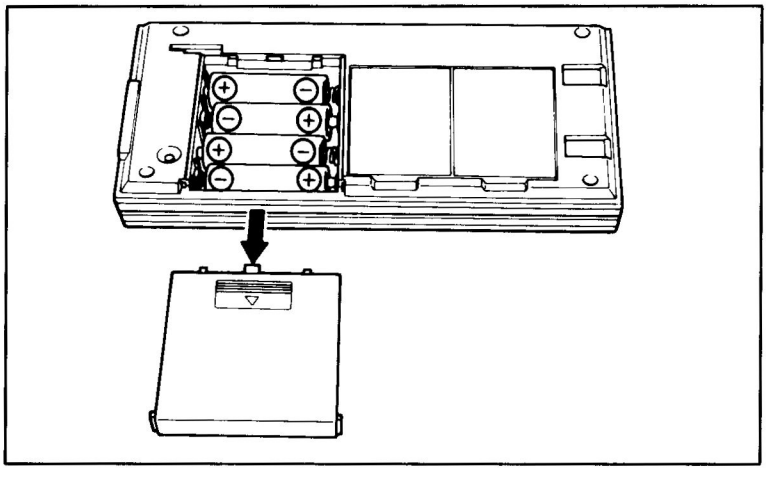
The service life of the batteries depends on how much you use the computer and under what conditions.

Typically, with normal use of 50 minutes display time and 10 minutes processing time per hour, the batteries should last for 25 hours at 20°C (68° F).

To install the batteries:

1 Remove the cover from the battery compartment by sliding it off in the direction of the arrow.

2 Insert the batteries with the +/− polarity as shown in the illustration.



Removing battery cover and installing batteries

3 Replace the battery cover.

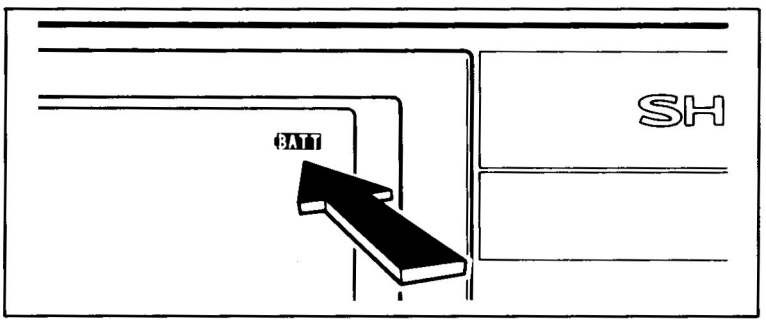
4 Turn to the section on Resetting the Computer in Chapter 3 and follow the ALL RESET steps to initialize the computer.

For information on replacing the batteries see Appendix A.

Low Battery Indicator

A low battery indicator comes on as a symbol BATT in the status line at the top of the PC-1600's LCD if the power of the four batteries is running low.

The BATT symbol may also come on when the computer is connected to a battery-powered peripheral, such as the CE-1600P Printer, and its batteries are getting low.



BATT Low Battery Indicator

If the printer is connected to the PC-1600, when the BATT symbol comes on, you can find out if the low batteries are in the computer or the printer by pressing SHIFT + ON.

- If the BATT symbol is still on after you've done this, **the computer's batteries are low**.

You should replace the batteries in the PC-1600 as soon as possible. See Replacing the Batteries in Appendix A. Caution: battery replacement involves the risk of program and data loss. Be sure to read this section carefully.

You may also choose to use an AC adaptor to power the computer at this time if replacements are not readily available. Use of the adaptor is described in the next section as well as in the pages on battery replacement. If you try to continue using the computer while the BATT indicator is on, the PC-1600 will automatically turn itself off.

○ If the BATT symbol goes out after you've pressed **SHIFT** + **ON**, **the low batteries are in the printer (or other connected peripheral).**

Besides the BATT symbol, your PC-1600 also provides two other types of low battery warnings when options are connected to the computer:

- When the computer is turned on, a screen message, CHECK __, appears followed by a numeric code for the suspect device.
- During program execution, an ERROR code is issued, again with a number identifying the cause of the trouble.

CHECK messages are dealt with in the section on Power On in Chapter 3, and ERROR codes are described for individual I/O devices in PART III, Using the Interfaces and Optional I/O Devices. ERROR codes are also summarized in a reference table in Appendix F.

If the printer is the cause, its built-in NiCad batteries should be recharged. Refer to the instructions in the printer manual for recharging the batteries.

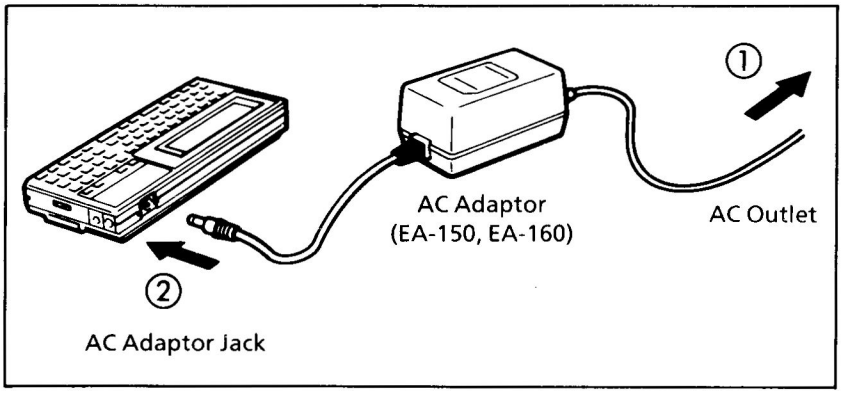
You may double-check that the low batteries are in the printer, not the computer itself, by trying to bring up a CHECK message on the LCD by turning the computer's power off then on again. When the CHECK message appears, see the pages mentioned above.

Connecting to AC Power

Using an AC adaptor connected to the jack on the rear of the PC-1600, the computer can be powered from a standard electrical outlet when indoors or to conserve the batteries.

Two AC adapters are available as individual options or as standard accessories with the following printers:

- EA-160 (with the CE-1600P printer)
- EA-150 (with the CE-150 printer)



Connecting PC-1600 to AC power source via adaptor

To connect the PC-1600 to AC power:

1 Turn off the PC-1600 and any peripheral devices connected to it.

2 First plug the cord from the AC adaptor into the electrical outlet.

3 Then connect the cord from the adaptor to the jack on the back of the PC-1600.

4 Be sure to unplug in reverse when disconnecting the adaptor from the computer. Never unplug the adaptor from the computer jack while the computer is still switched on.

5 Unplug the adaptor from the electrical outlet when not in use.

IMPORTANT: *If the computer's batteries are dead or not installed when the AC adaptor is disconnected from the computer, any data remaining in the computer's memory is lost.*

The AC adaptor is also used to recharge the built-in NiCad batteries in the printers and other devices, and using an AC adaptor is one way of preserving data in the computer's memory when you are replacing batteries. See Replacing the Batteries in Appendix A.

3 Turning the Power On and Off

This chapter describes the methods for turning your PC-1600 on and off. You'll also see samples of the LCD display after power-on under various operating conditions.

The following section mentions the use of RAM expansion modules and other peripheral I/O devices with the PC-1600 computer.

These memory modules and devices are covered in Part III, Using the Interfaces and Optional I/O Devices. A section on Replacing the RAM Modules is included in Appendix B. Also, always refer to the individual manuals accompanying all options available for this computer for specific detailed information.

Power On

You can turn on the power of your PC-1600 in one of three ways:

- Manually, by pressing the **ON** key.
- Use the WAKE$ command in BASIC to start the computer at a specified time and date. (See WAKE$ in the Command Dictionary.)
- Again using the WAKE$ command, program the PC-1600 to turn itself on when a CI signal is received via the computer's RS-232C serial port from an external device such as a modem telephone. (See the WAKE$ command and serial interface specifications.)

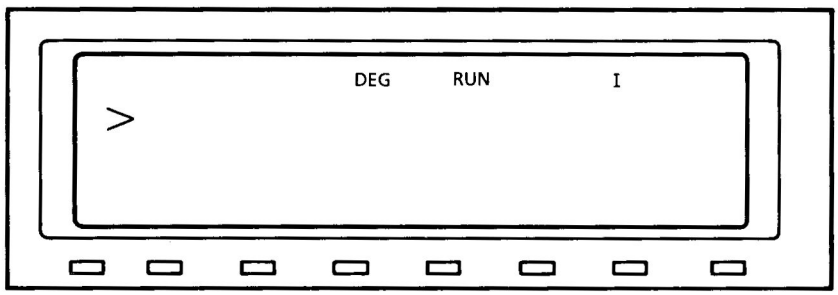
What you see on the screen when you turn on the PC-1600 depends on whether the computer is being used independently, or with optional RAM modules loaded in the expansion slots, or with other I/O devices connected. Your PC-1600 has built-in warning messages and prompts to alert you of low battery conditions and other trouble at power-on.

The following examples show the LCD screen after power-on under these different conditions.

NOTE
When you turn the power on the PC-1600 ON or OFF, the screen momentarily goes black, and may display random patterns.
This is quite normal, and the display will clear almost immediately.

• Using the PC-1600 without options and at normal power-on...

When you press the **ON** key the screen looks like this:



At the top of the display is a status line which shows the operating mode and functions you have selected. When you turn on your PC-1600 the very first time or after an ALL RESET, the status line should show DEG (Degrees), RUN (RUN operating mode), and I (RESERVE Mode I).

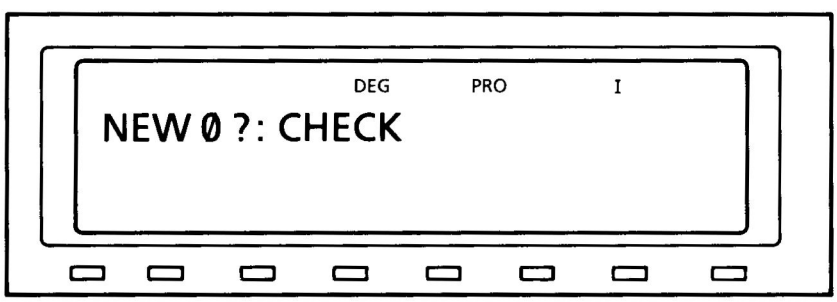
These features are described elsewhere in this manual in the section on the PC-1600 Display and BASIC.

The status line appears every time the computer is turned on and stays lit during operation. The symbols in the line show the settings that were in effect when the computer was last turned off, so they may be different each time you turn on the PC-1600.

The > symbol is called a system prompt and it shows that the computer is waiting for you to type on that line. The prompt is replaced by the first character you enter on the line.

• After installing or replacing a RAM module...

You must turn off the computer before installing or replacing the optional RAM modules. When you turn on the computer again the screen shows:



This message tells you that the computer recognizes the additional memory and is waiting for you to clear the memory area of any possible remaining spurious data. Note that this message comes up only when the memory modules are installed for expansion memory. The message is not displayed if a program module is configured as a RAM disk or program memory. Use of the modules is described in PART III of this manual.

To clear the computer's memory and restore its default, or preset, values, press the **CL** (Clear) key. Then be sure that the operating mode is set to PROgram. Press the **MODE** key if necessary. Then type in N E W Ø and press **ENTER**.

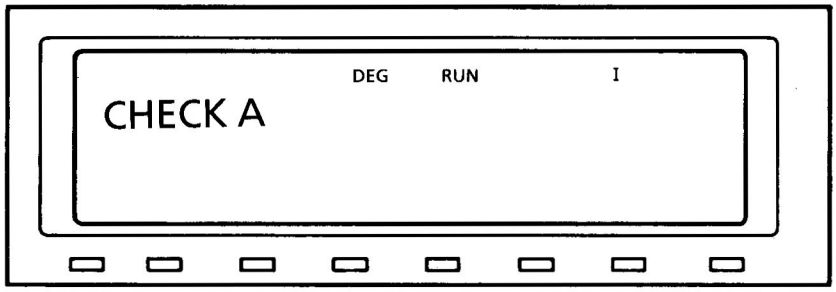
The display should then return to the system prompt > shown in the previous illustration and you can begin to use the computer.

• When RAM modules are installed incorrectly...

There are two expansion slots, S1 and S2, on the underside of the PC-1600 in which you can plug-in optional RAM modules for adding extra memory to the computer or storing programs and data.

These modules are described in detail in PART III, Using the Interfaces and Optional I/O Devices, and in Appendix B for their replacement.

Three of these optional modules, CE-151, CE-155, and CE-159, can be used in Slot 1 only. As both slots are identical in size and shape, if you inadvertently plug one of these modules into Slot 2, the computer will not recognize it. When you then turn on the power, the display shows:



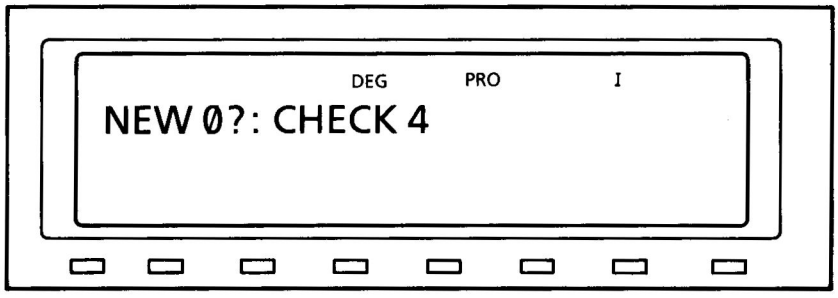
This CHECK message is a special feature of the PC-1600 to alert you that something is wrong and tells you to check the suspect cause of the trouble.

The number following the word CHECK is a code for the problem device. The Ø in this case means Slot 2. A table of these CHECK codes is given in the next section.

You must take action on this CHECK message before continuing to use the computer.

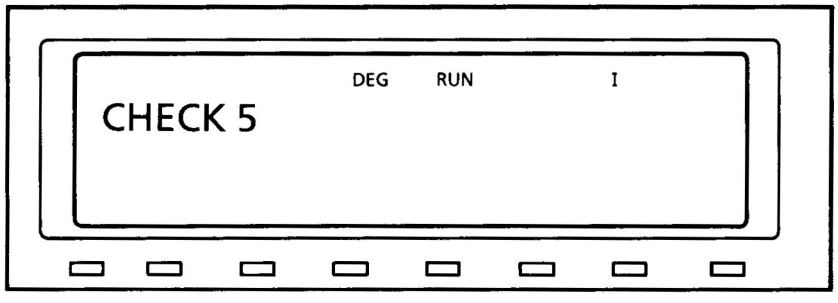
• When I/O devices with low batteries are connected...

Some of the optional peripheral devices that can be connected to your PC-1600 have their own built-in batteries. If the batteries are low in the device connected to the computer, the following messages appear when you turn on the PC-1600:



N E W 0 ? tells you that the computer recognizes a change in memory and is a prompt to clear the memory.

Or, as another possibility,



These CHECK messages tell you that the batteries in the connected device need to be recharged or replaced. Also, as described in the previous section, CHECK A is a warning that the wrong memory module is plugged into Slot 2.

See the following table for the meanings of the CHECK messages illustrated above and the other CHECK codes that might appear. Code numbers missing from the list are reserved for new devices that may be available in the future.

CHECK Code	Connected Device	Cause
4	CE-1600P Printer	Built-in NiCad battery is low or printer hardwere fault.
5	CE-1600F Disk Drive	Connects to the CE-1600P Printer. Built-in NiCad battery in printer is low or drive hardware fault.

6	CE-150 Printer	Built-in NiCad battery is low or printer hardware fault.
8	CE-158 Serial/ Parallel Interface	Built-in NiCad battery is low.
A	Module Slot 2	CE-151 or CE-155 incorrectly installed. Use only Slot 1 for these modules.

Power Off

Your PC-1600 computer has two auto power-off features to switch itself off. The PC-1600's Memory Safe Guard feature preserves any data you have been working on when the computer's power is turned off, both manually and by the auto-off functions alike.

The PC-1600 automatically turns itself off under the following conditions:

- To conserve battery power, the PC-1600 turns itself off when there has been no key input for ten minutes. This will not happen if you are running a program during this time interval. Just press the **ON** key to start again. This function can be disabled with the POWER statement in BASIC.

- If you try to continue operating the computer after the BATT low battery indicator comes on, the PC-1600 switches itself off to prevent processing errors and possible loss of data. When BATT is displayed, you should turn off the computer and replace the batteries or use the optional AC adaptor as soon as possible.

Under normal operating conditions you have two ways to turn off your PC-1600 computer:

- Manually, by pressing the **OFF** key.

- Use the POWER command in a BASIC program. See the BASIC Command Dictionary for details.

Resetting the Computer

There are times when it is necessary to clear the computer's memory, restore its preset or default settings, and return the computer to an operable state.

This action is called resetting the computer and there are two reset levels for your PC-1600: simple and ALL RESET.

Simple reset is performed by pressing the RESET switch on the underside of the computer. ALL RESET is performed by pressing and holding the ON (BREAK) KEY while pressing the RESET switch.

Resetting is necessary under basically two conditions:

- To initialize the computer and prepare it for operation after installing or replacing the batteries.
- To restore the computer to normal operation after it "locks-up".

If the computer is subject to high-level electrical noise or is operated under unstable conditions that could cause processing errors, it may "lock-up" or cease to operate normally.

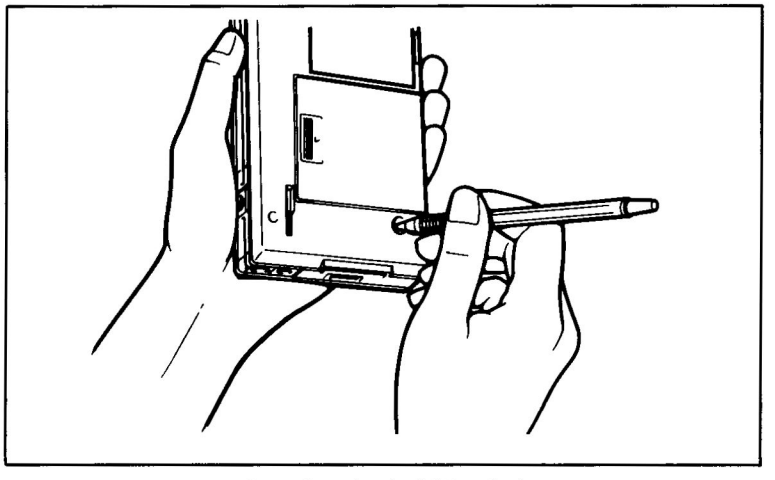
Effects may vary, but the screen could "freeze" or go blank, key operation may produce strange or meaningless characters, or keys may even become completely inoperative. Simply turning the computer off then on again to restart it may not be possible if the OFF and ON keys do not work.

A RESET switch is provided on the bottom of the computer to release this condition. The switch is deeply recessed in the case to prevent accidental use as the reset function is very powerful and should be used with caution.

WARNING: *Programs and data can be destroyed!*

Use a pointed object such as a ballpoint pen to press the RESET switch. Do not use easily broken points such as pencils or pins.

If you have the computer connected to the optional CE-1600P Printer, you do not have to remove the computer to gain access to the RESET switch on its bottom. A similar recessed RESET switch is provided on the underside of the printer case and can be used instead for the steps that follow.



Pressing the RESET switch

The following instructions give the steps and show the appearance of the display when you reset your PC-1600 computer.

If the computer "locks-up", try the simple reset first, as its effect is similar to turning the power off and on, and in most cases, both programs and data should be safely recovered, if not in their entirety. The ALL RESET procedure is **complete** and should only be used as a final measure when all else fails.

ALL RESET is also used to initialize the computer after installing or replacing the batteries.

The following table gives a comparison of the different effects of simple and ALL RESET actions on the computer's system parameters and default settings.

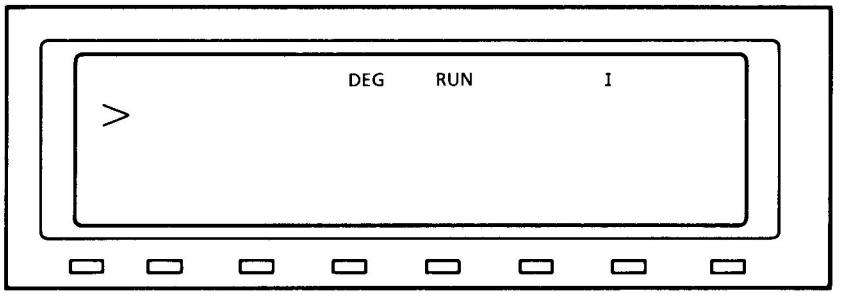
Reset Parameter Summary

	ALL RESET	Simple Reset or Power-on
Auto power-on(WAKE$)	OFF	Setting preserved
Auto power-off	10 min	Setting preserved
TITLE	S0:	Setting preserved
Date/time	1(mo)1(day)0(hr)0(min)0(sec)	Setting preserved
Function keys	Cleared	Setting preserved
Password	Canceled	Setting preserved
RS-232C Serial port	COM1 — 1200, 8, N, 1, X, S COM2 — 38400, 7, E, 2, X, S	Setting preserved
Machine language area	0	Setting preserved
Key click	OFF	Setting preserved
BEEP	ON	Setting preserved
Key repeat	OFF	Setting preserved
Interrupt statements	Disabled	Setting preserved
Mode	RUN DEG I	Preserved or changed after error is generated
BREAK ON/OFF	ON	Setting preserved
LOCK/UNLOCK	UNLOCK	Setting preserved
Mode 0/1	Mode 0	Setting preserved
MAXFILES	0	Setting preserved
Expansion memory module	Memory erased	Memory preserved
Program module	Memory preserved	Memory preserved
RAM disk module	Memory preserved	Memory preserved
Internal RAM	Memory erased	Memory preserved

To perform a simple reset:

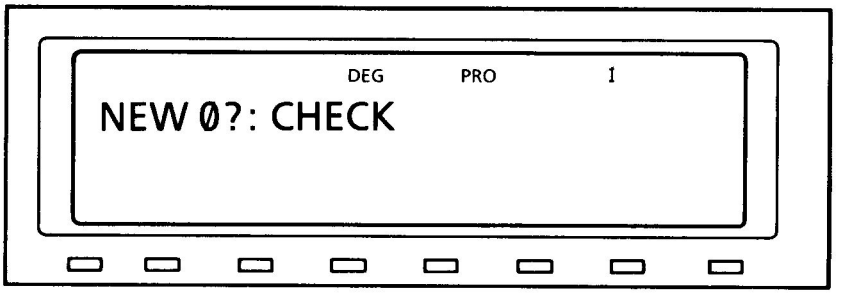
1 Locate the RESET switch on the back of the computer and press it for a couple of seconds.

2 The screen should clear and display the system prompt > as illustrated below:



3 The prompt means that the computer is ready to use and is waiting for your input. If the computer has not cleared, and there is no change to the screen, press the RESET switch again.

4 After you press the RESET switch, sometimes the following screen comes up when the computer is being used with optional I/O devices:



5 To clear the computer from this screen, press the **CL** (Clear) key. Then be sure that the operating mode is set to PROgram. Press the **MODE** key if necessary. Then type in N E W 0 and press **ENTER**.

The display should then return to the system prompt > shown in the previous illustration and you may begin to use the computer again.

To perform an ALL RESET:

Follow these steps after installing or replacing the batteries.

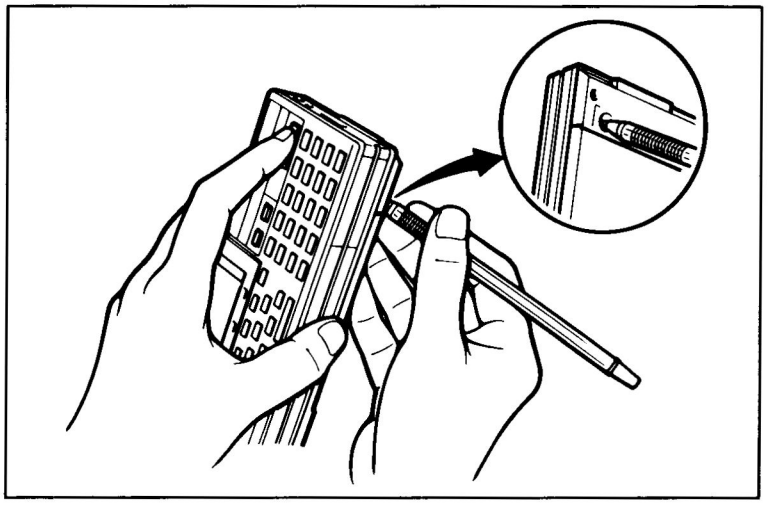
In all other cases, read the following before continuing:

IMPORTANT: *If the computer has locked up during operation, try the simple reset first before going on to these steps. ALL RESET is a complete action which can destroy data and programs. Check the reset parameter table in this section and make sure you can risk this. Data and programs in the computer's internal RAM and expansion memory are completely cleared after an ALL RESET.*

1 Hold the computer so that you have easy access to the ON key at the right of the keyboard. Note that while the computer is on, the ON key functions as the BREAK key.

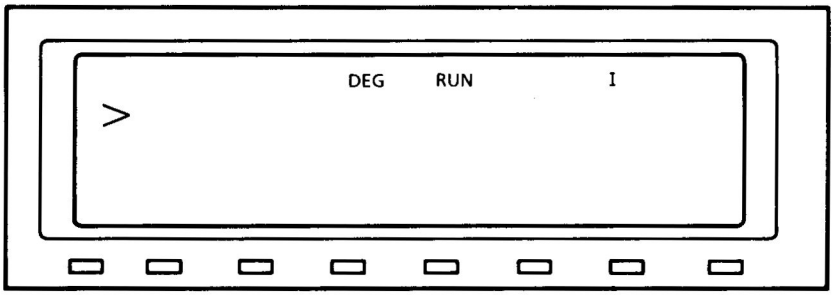
2 Press and hold the ON (BREAK) key, then press the RESET switch on the back of the computer.

3 Release the RESET switch first, then release the ON (BREAK) key as illustrated below.



ALL RESET using RESET switch and ON (BREAK) key

4 The screen should display the system prompt > as illustrated below:



5 The prompt means that the computer is ready to use and is waiting for you to begin input. If the prompt has not appeared, press the RESET switch and **ON** (BREAK) key again, as described in steps 1 to 3.

IMPORTANT: *If you are unable to reset the computer after trying either or both simple and ALL RESET, contact your SHARP dealer immediately.*

4 The PC-1600 Display

The display on your PC-1600 computer is a 26 column × 4 line LCD with adjustable contrast for optimum viewing.

Two screen modes may be specified with a BASIC command. In MODE Ø the screen can be fully used in its 4-line mode displaying 26 characters per line. MODE 1 emulates the 1-line screen of the PC-1500 Pocket Computer when using PRINT and GPRINT statements for compatibility with its programs and data. When input exceeds the physical line in this mode, the screen will scroll up from the bottom line.

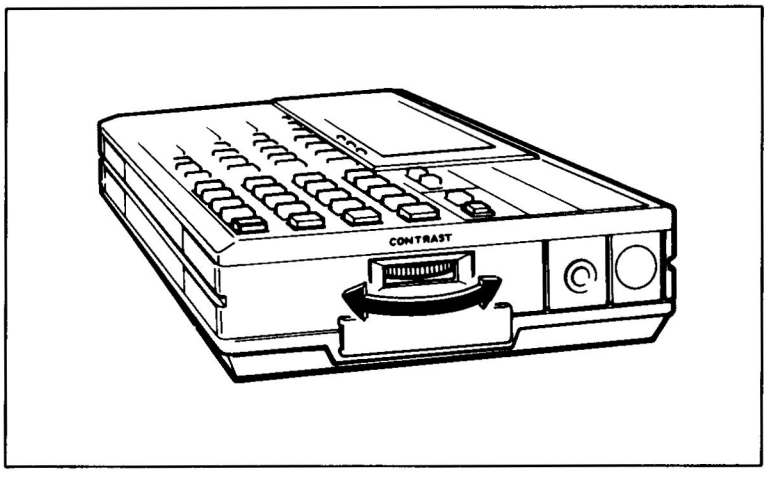
MODE Ø allows the display to be used in a graphics mode in which the entire screen matrix of 156 × 32 dots is available for displaying bit image graphic data.

These modes are described in detail in the BASIC Reference Section of this manual.

Adjusting the Display Contrast

When lighting conditions vary or you change the angle of the computer by attaching it to the optional printer, you may need to adjust the display contrast for comfortable viewing.

The contrast adjustment dial is located on the right side of the computer, just above the RS-232C serial port.



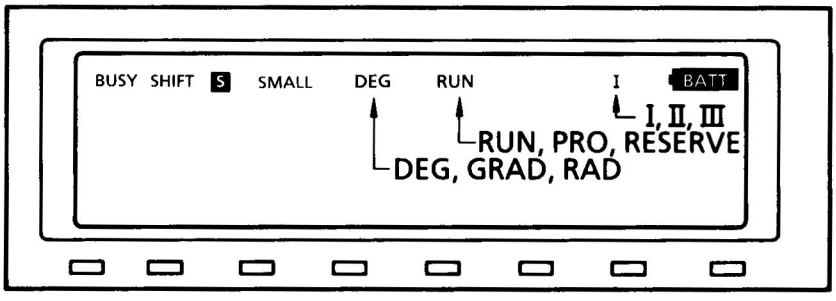
Display Contrast Adjustment Dial

Turn the dial as needed to lighten or darken the screen until you are able to read it easily. Re-adjust the contrast dial when you move the computer or change its angle again.

Status Line Symbols

A special status line is provided at the very top of the PC-1600's LCD, above the first of the four data lines, to display symbols for low battery warning, operating mode and functions you have selected or are currently using.

The status line allows you a quick, at-a-glance check of which functions you've selected, whether you've set them correctly, and what the computer is now doing.



LCD Status Line Symbols

The symbols in the status line will not change, that is the settings remain effective, when you turn the computer back on after turning it off. This allows you to either continue with the previous settings as is, or after checking the status line, make the necessary changes and continue on from that point.

Here is a list of the various symbols that are displayed in the status line and their meanings.

BUSY Displayed while the computer is executing a program or command. Other key entry is not possible at this time.

SHIFT Displayed with each press of the **SHIFT** key. Goes off on the next key-in.

DEG Indicates Degree mode is selected for angular function.

RAD Indicates Radian mode is selected for angular function.

GRAD Indicates Gradient mode is selected for angular function.

RUN	Displayed when computer is in RUN mode for direct calculation or program execution.
SMALL	Displayed while computer is in the Small mode for typing lower-case characters on the screen.
PRO	Displayed when the computer is in PROgram mode for creating, editing or listing programs.
RESERVE	Displayed when the computer is in the RESERVE mode for assigning and editing character strings to the function keys.
DEF	Displayed with each press of the DEF (Define) key. Goes off on the next key-in.
I	Indicates selection of character strings in menu I of RESERVE mode.
II	Indicates selection of character strings in menu II of RESERVE mode.
III	Indicates selection of character strings in menu III of RESERVE mode.
CTRL	Displayed with each press of the CTRL (Control) key. Goes off on the next key-in.
BATT	Indicates low battery in either the PC-1600 computer or a connected peripheral.
S	Indicates selection of keyboard mode II (KB II) for using the international character set template.

Operating Modes

Your PC-1600 has three operating modes: RUN, PROGRAM, and RESERVE. To select either RUN or PROGRAM mode, press the MODE key to toggle, or switch back and forth, between them. The RESERVE mode is selected by pressing SHIFT + MODE.

The symbol for the mode you select is displayed is the status line at the top of the LCD.

You can also set the operating mode using two BASIC commands, LOCK and PASS, when programming on the computer. Look up both of these in the Command Dictionary to find out how they relate to the use of the keys.

The use of these operating modes is described in the BASIC Reference Section. Here are brief descriptions of each mode and when its used:

RUN The RUN mode is used for manual, or direct, calculations on the computer. This allows the PC-1600 to perform as a calculator. This mode is also used to run, or execute, programs created in the PROGRAM mode or loaded into computer memory from an external storage device.

PROGRAM The PROGRAM mode is used for creating, editing, listing programs. To execute the program, you must then switch to the RUN mode.

RESERVE The RESERVE mode is used for creating and editing character strings assigned to the programmable function keys. There are three menu modes, I, II, and III, available in the RESERVE mode.

Regardless of which mode you've selected, when you turn the computer off, then on again, the computer will remain in the same mode set at power-off. Its symbol will be displayed in the status line at the top of the PC-1600's LCD.

Be sure to double-check the mode symbol in the status line if you have difficulty while operating your PC-1600. If you have not set the proper mode for the operation you wish to perform, the computer will return an error code on the screen. As these messages use a numeric code to identify the problem, check the error code list in Appendix F to find out the meaning of the error.

Editing Key Functions

This section briefly describes the use of editing keys for controlling the cursor and making simple corrections in the RUN mode when you are setting the time and date and calculating in the following parts of this chapter.

Advanced functions using other key combinations are covered in the EDIT mode description in the BASIC Reference Section.

Here are the keys and functions you will be using in this chapter:

DEL
◀ Left Arrow Cursor Key

This is one of the two cursor keys for movement on a line. When you press this key, the cursor moves one character position to the left. If this key is pressed and held down, the cursor will continue to move until it reaches the end of the logical line on the left of the screen. Used alone, this key does not erase characters, but passes the cursor under them.

DEL
SHIFT + ◀ Delete Key

When you press this key combination, the character or space on the cursor position is deleted and the space closes up with any characters to the right of the cursor moving left into the space. The SHIFT key is effective for only one keystroke after it. So you must press the SHIFT key for every character you want to delete.

INS
▶ Right Arrow Cursor Key

Pressing this key moves the cursor one character position to the right. If you press and hold this key the cursor will continue to move to the end of the physical line on the screen and will then move down to the beginning of the next line at screen left. As with its counterpart, the Left Arrow Key, this key alone cannot erase characters, but passes the cursor under them.

INS
SHIFT + ▶ Insert Key

When you press this key combination, the character position on the cursor opens to a space with all characters after it moving to the right to accommodate the blank space. As the SHIFT key is effective for only a single keystroke after it, you must press SHIFT + INS ▶ for every space you want to insert.

BS Backspace Key

The BS (Backspace) key moves the cursor to the left one character position, deletes the character or space there, and closes the space by moving any following characters up to the cursor position.

CL Clear Key

Pressing the **CL** (Clear) key deletes the entire line on which the cursor was positioned and returns the system prompt > to the beginning of the line as signal for you to start typing again. The cursor appears after the first character you type.

CTRL + **A** Insert/Overwrite Mode

This Control Key combination switches back and forth between INSERT and OVERWRITE text editing modes. The default setting at power-on is OVERWRITE mode, in which characters from the cursor position onward are deleted as you type in "over" them. When you set INSERT mode, characters typed in will be "inserted" in the line at the position before the cursor. Any characters from the cursor position onward will be moved along to accommodate the inserted characters. Note that the cursor blinks faster when INSERT mode is set.

Here are a few points to keep in mind for troublefree typing and editing:

- The secondary functions that are printed on the computer's case above the keytops themselves are effective only when the **SHIFT** key is pressed first. So if you accidentally forget to press the **SHIFT** key, for instance before INSerting or DELeting, use of these keys will only move the cursor back and forth. Check the status line for the SHIFT symbol to make sure.

- Pressing the **SHIFT** key affects only one keystroke after it. To delete a series of character positions (such as a whole word), you must press the **SHIFT** + **DEL** key combination for each character, or use the **BS** key.

- Once you begin typing, the cursor positions itself at the next position after you stop typing. So the character you are about to type always goes into the position currently occupied by the cursor. Likewise, when you move the cursor about before deleting or inserting, these actions will take place at the cursor's new position.

- For quick correction of short entries in OVERWRITE mode, just reposition the cursor at the unneeded part and type over it. Overtyping only deletes the characters at the newly typed positions. It will not open or close spaces or move the rest of the line away from the portion you are retyping.

- Practicing typing and editing is the best way to become familiar with the keyboard. Use the notes on these pages to refresh your memory when you have difficulty.

Setting the Time and Date

The real-time clock in the PC-1600 allows you to use BASIC commands to start the computer at a specified time and day, sound a beep alarm, or display a message on the screen.

This section will show you how to set the clock to the current time and date. Later, you can display the time whenever it's needed or write programs which rely on the correct time to perform operations.

The BASIC commands that are used for time-related functions are TIME, TIME$, DATE$, WAKE$, and ALARM$. See the BASIC Command Dictionary for full descriptions of these commands.

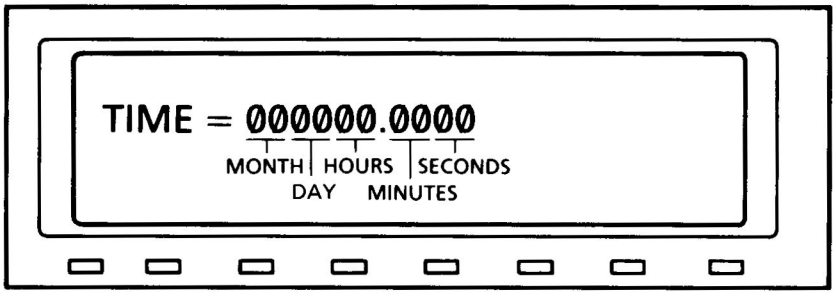
One of these commands, TIME, will be used in the following steps to set the time and date initially. You will also use this command whenever you want to simply see the current time.

To set the current time and date:

1 With the computer turned on, first make sure that the operating mode is set to either RUN or PROgram. Use the **MODE** key if necessary.

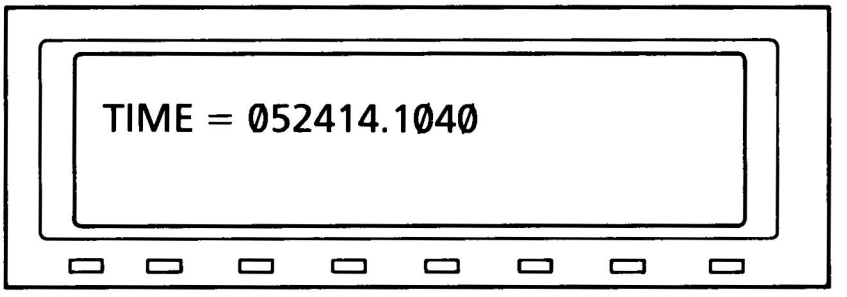
2 When the system prompt > appears type in T I M E =. Do not use spaces or forget the equals (=) sign. The computer is now ready to accept the numbers for the date and time.

3 The format for the numbers corresponding to the date and time is two digits each for month, day, hour, minute, and second. Notice the decimal point between hours and minutes in the illustration.



4 You must include a Ø in the first digit for all numbers from 1 to 9 (thus, type Ø1 to Ø9). Be sure to include the decimal point between the hours and minutes and do not leave any spaces. Use the 24-hour time notation, where midnight is ØØ, noon is 12, 6 pm is 18 and so on.

5 The next illustration shows an example of the time and day set for the month of May (Ø5), on the 24th day (24), at 2 pm (14), 10 minutes (1Ø), and 40 seconds (4Ø). The year cannot be set.

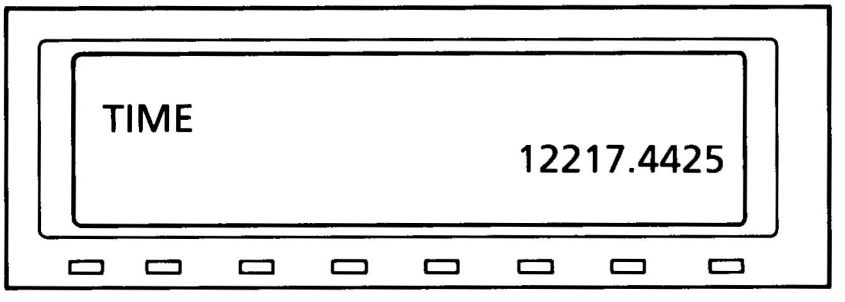


6 Now type in Ø1 to 12 for the month, Ø1 to 31 for the day, ØØ to 23 for the hour, a period to insert a decimal point, ØØ to 59 for the minutes, and lastly ØØ to 59 for the seconds.

7 The computer will set the time you entered to the second when you press the ENTER key. You may find it convenient to synchronize pressing the ENTER key with the time chime tape recording available as a dial-in telephone service. Any other clock or watch should do if you are not concerned about accuracy to the second.

8 Now, when you're ready, press ENTER to set the time and date.

9 Whenever you'd like to see the time, just type in T I M E and press ENTER. No equals sign is necessary. When you do this the screen is a bit different, with the date and time displayed on the next line at the far right of the screen. For other uses of time and the clock, look up the mentioned commands in the Command Dictionary. Here's what the screen looks like when you display the current time and date.



5 Calculating on the PC-1600

This chapter will show you how to perform a variety of calculations on your PC-1600 computer. With its separate numeric keypad, the PC-1600 allows you do simple calculations quickly, as though you were using a portable calculator. And, when the computer is attached to one of the optional printers, CE-1600P or CE-150 (using MODE 1), setting a switch on the printer lets you print out calculation expressions and results as you are working. In this way your PC-1600 computer and printer combination can work much like an adding machine with a printout as well. At advanced levels, too, the function operations in the BASIC programming language built in the computer let you perform calculations within programs.

Check back in the section on Editing Key Functions in Chapter 4 if you make any typing mistakes.

Setting the Mode

Set the RUN mode for using the computer in this chapter. When you are later writing programs to use the calculating functions of the PC-1600 in BASIC you will be setting the PROgram mode. See the BASIC Reference Section for this advanced level of operation.

Check the status line at the top of the screen to make sure that the RUN symbol is on. If necessary, press the **MODE** key to set the RUN mode.

Key Functions

This section introduces the use of the numeric keys and describes their special features and functions.

ENTER ENTER Key

The **ENTER** key, as you've learned from previous sections, is a signal to the computer that an entry sequence is complete and to take action, or execution, on the data it received. In simple calculations, the **ENTER** key is used to mean "equals", that is, to return the result of the expression. Do not press the **=** (Equals) key to perform the calculation as this key's function is reserved for special use in BASIC.

[*] Asterisk Key

The asterisk key returns the function of multiplication, commonly indicated by the arithmetic symbol ×.

[/] Slash Key

The slash key returns the function of division, as otherwise indicated by the arithmetic symbol ÷.

[E] E Key

The character **[E]** key is used to enter numbers in scientific notation with the exponent.

Calculation Examples

Here are a few examples of calculations on the PC-1600. They show the conventional notation of the expressions, the keystrokes needed to enter them in the computer, and the result displayed on the screen. Try these on your PC-1600 to familiarize yourself with the keys and their functions. The last example shows how the built-in BASIC operations work on the computer. After doing these calculations, try a few of your own. Notice that after an equation is executed and the result displayed, the next equation you enter appears on the next line as the screen scrolls up.

Example 1

$2 + 3 \times 4 = ?$

Press: 2 + 3 * 4 [ENTER]

Result: 14

Example 2

$36 \div (1 + 2) = ?$

Press: 36/(1 + 2) [ENTER]

Result: 12

Example 3

$5 \times 10^3 \div (4 \times 10^{-3}) = ?$

Press: 5E3 / 4E − 3 ENTER

Result: 1250000

Example 4

$\sin^2 30° = ?$

Press: DEG. ENTER (SIN 30) ^ 2 ENTER

Result: 0.25

Serial Calculation

As you've noticed from the previous examples, the expression you enter appears on the top line of the screen. When you press the ENTER key, the result is displayed on the right, at the end of the next line.

You can use this result as the first entry of the next calculation by pressing a numeric function key without retyping the result.

The illustration below shows what happens on the screen as you link results in a serial calculation. Notice that you do not need to retype the result, it appears automatically at the beginning of the next line as you enter the numeric functions. When the bottom line of the screen is reached, the screen scrolls up one line to make room for the next expression or result.

Try entering the same numbers in the illustration and watch what happens on your own PC-1600.

When you press these keys... The screen looks like this...

Press:

12 / 6 ENTER

* 5 + 10 ENTER

+ 30 ENTER

Recall Function

In Chapter 4 you learned how to use the right and left arrow keys to move the cursor and edit your entries. There are two other convenient uses of these keys during direct calculation.

The Recall function allows you to redisplay and return to an executed equation to re-enter values or make other changes. This feature can also be used in a slightly different way after an ERROR code is displayed. You'll find out more about ERROR codes in the next section.

You can at any time go back to an equation, after seeing the result, and make some changes in the entries. This may be convenient for trying some simple "what if" changes to see what effect new values have on the outcome of the calculation. Or, when the results produced do not seem proper, perhaps a mistaken value was entered at a point in the equation and needs correction.

After you press the ENTER key to execute a calculation, the result is displayed at the end of the next line. If you then wish to return to the equation itself, press the ▶ (right arrow) key to redisplay the equation with the cursor set at the first character position. Likewise, pressing the ◀ (left arrow) key positions the cursor at the last character.

You may then move the cursor using the ◀ or ▶ keys to the position you wish to change and follow the procedures you learned in the section on Editing Key Functions in Chapter 4.

After you have made changes to the redisplayed equation, press ENTER and execute it again. The screen will scroll up one line and the result will be displayed on the next line.

Error Codes

As the computer tries to execute the equation you entered, it first checks that the expression was entered validly. If your entry is in some way incorrect, the computer will not execute the calculation, but will display an ERROR code on the next line instead of the result.

First, check the list of ERROR codes in Appendix F to find out the meaning, and also check the equation itself.

A typical simple error might be omitting one of a pair of parentheses ( ) in a compound calculation. If an error code is displayed, you must first clear it before continuing to use the computer.

Choose one of these two ways to deal with the error:

- As a variation on the use of the arrow keys in the previous section on the Recall Function, press either ◀ or ▶ to clear the ERROR code and redisplay the equation. This time the cursor will be positioned at the point the error occurred.

  You can then correct the mistake and re-execute the calculation by pressing the ENTER key.

- Or if, after checking the equation, you wish to re-enter it completely, pressing any of the following keys clears both the ERROR code and the equation from the screen and the computer's memory. Remember you will then have to enter the equation again. This may be inconvenient for long or complicated equations, so decide if this is what you want to do. If so, press:

  ON or **CL** or **SHIFT** + **CL**

  Then type in the equation again and execute it by pressing **ENTER**.

BASIC Function Operations

Your PC-1600 features a number of built-in functions which provide a short-cut for calculations in both direct mode and within programs. Two of these functions, Sine and Degree, were briefly introduced in the calculation examples. The list that follows shows all of the functions that are available in this computer's BASIC. For full descriptions of each, see the BASIC Command Dictionary in PART IV of this manual. Also, check the related chapters in the BASIC Reference Section for details on the use of expressions and operators and the evaluation priority for arithmetic expressions.

The following functions are built in your PC-1600's BASIC:

ABS	Absolute value	LN	Natural logarithm
ACS	Arc cosine	LOG	Common logarithm
ASN	Arc sine	PI	Value of pi
ATN	Arc tangent	RND	Random number
COS	Cosine	SGN	Sign
DEG	Decimal/degree	SIN	Sine
DMS	Degree/decimal	SQR	Square root
EXP	Exponent	TAN	Tangent
INT	Integer		

The values entered for the trigonomietric functions can be in degrees, radians or gradient values as set by DEG, RAD or GRAD.

PART III

USING THE INTERFACES AND OPTIONAL I/O DEVICES

The key to the flexibility of your PC-1600 pocket computer is its ability to expand into a compact, full-featured system using a variety of optional I/O devices.

The sections in this part provide an introductory description of individual devices, references to BASIC commands used to operate them, and special notes on connection and error handling.

Chapter 7 is a description of the compatible PC-1500 computer's peripherals that can be used with your PC-1600.

6 Expanding Your PC-1600

This chapter will show you how to build on to your PC-1600 pocket computer using SHARP's many optional devices. You can use these add-ons to create the compact system described on the following pages.

System Overview

Your SHARP PC-1600 computer packs a lot of power and features for its size. It's even more versatile when connected to an array of specially designed optional devices as well as the compatible options of the PC-1500 computer. With the PC-1600 computer at the core, you can build a compact system to handle a variety of data processing, storage, and output needs.

This section of the manual will briefly introduce the options available and mention the key points you need to keep in mind when using the devices. For detailed explanations, always refer to the instruction manuals included with the individual devices.

The computer itself features three standard interface ports: RS-232C serial, optical serial, and analog input. These ports are used to connect the computer directly to such external devices as a printer, modem, or sensor.

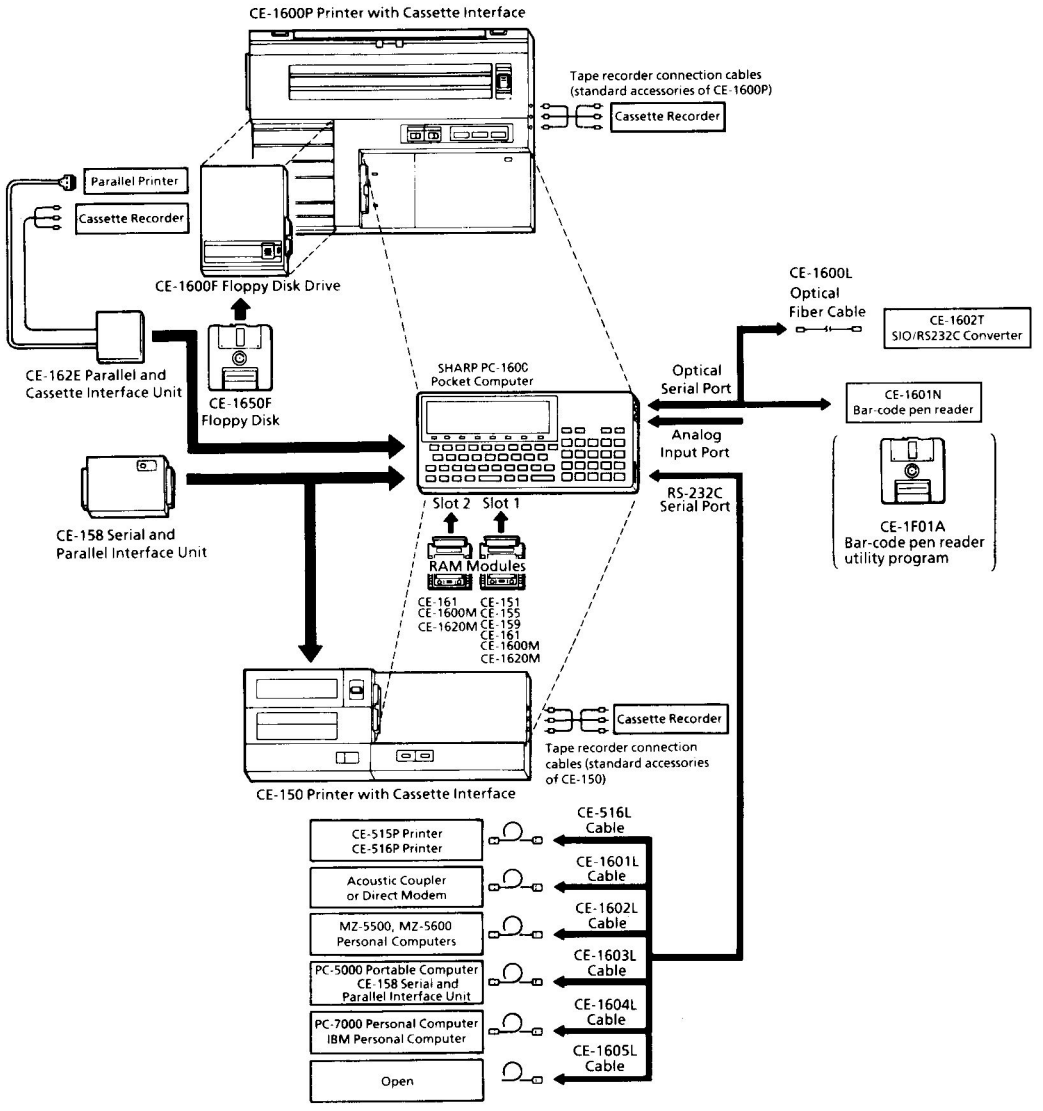
The addition of RAM modules to one or both of the slots on the back of the PC-1600 lets you expand its 16K of internal memory to a maximum of 80K, and with the choice of a program module, you can write and store programs then plug them into the computer as needed.

The PC-1600 computer is designed for a direct slide-in connection to the optional CE-1600P printer for four-color printouts of programs, text, and graphics. The integrated printer and computer unit then provides the base to which other options can be connected.

The CE-1600F Floppy Disk Drive is a useful addition for mass storage of programs and data and mounts right on the printer body to round out the basic configuration of your compact system.

Another key feature of your PC-1600 computer is its compatibility with the SHARP PC-1500 Pocket Computer. This means that it can not only share the PC-1500's BASIC commands and programs, but can also connect to its optional I/O devices. These include a data cassette recorder, RAM modules, printer, and parallel and serial interfaces.

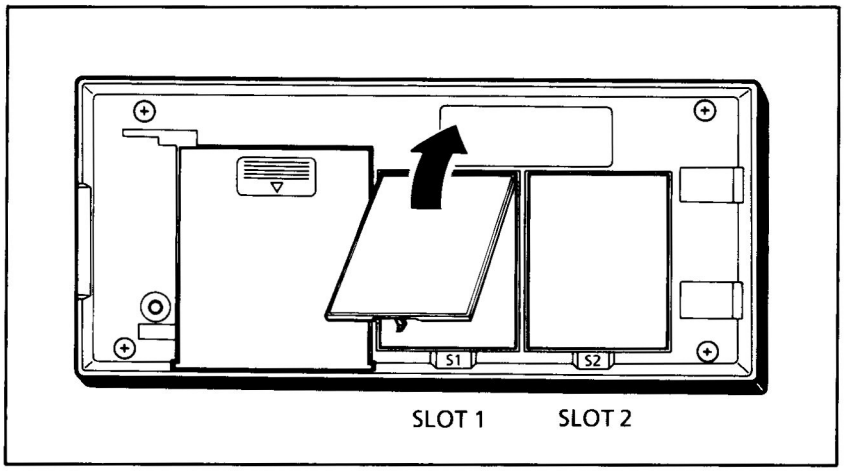
This illustration shows the variety of connections possible for building your pocket computer system. The compatible PC-1500 devices are described in Chapter 7, Using the PC-1500 Peripherals.



SHARP PC-1600 Pocket Computer System Configuration

RAM Module Expansion

You can expand the 16K of internal RAM memory in your PC-1600 by plugging RAM modules into the two slots on the underside of the computer. This section describes the types and use of the modules, gives key points for connection and use, and lists the related BASIC commands and ERROR codes.



Module Slots 1 and 2 on the back of the PC-1600

RAM modules are available in two types: memory modules, which are expansion memory for the computer's built-in RAM, and program modules, which have an internal backup battery and write-protect switch. Program modules allow you to use the memory as program memory to write and store programs that you can plug into the computer as required. One feature of the CE-161 and CE-1600M program modules is that they can be used as a RAM disk to which you can write and read data files as on a floppy disk.

The following table compares the features of the RAM modules available for your PC-1600 computer.

	CE-151	CE-155	CE-159	CE-161	CE-1600M
Module type:					
Memory module	●	●			
Program module			●	●	●
Capacity	4K	8K	8K	16K	32K
Use:					
Expansion memory	●	●	●	●	●
Program memory			●	●	●
RAM disk				●	●
Battery backup			●	●	●
Write-protect switch			●	●	●
Used in SLOT 1/2	1	1	1	1/2	1/2

Key Points

1 **Power Off** Always turn off the computer before installing or replacing RAM modules.
When the computer is turned off, the Memory Safe Guard feature protects the memory contents of RAM expansion modules as well as the computer's internal RAM.

2 **Power On** Never remove a RAM module with the computer on. When you turn on the computer after installing a RAM module for expansion memory, the computer will display a message prompting you to clear and initialize the changed memory area. This message is not displayed for program modules used for program memory or RAM disk. See the Power On section in Chapter 3.

3 **Module Slots** There are two RAM module slots on the back of the computer. Check the table on the previous page to find out which modules are restricted to use in Slot 1. See the Power On section in Chapter 3 for CHECK messages.

4 **Handling Modules** Never touch the connectors on modules or the slot interiors. Static electricity presents a real hazard to computer memory. In carpeted rooms and during dry winter weather, be sure to discharge any static charge by touching a doorknob or similar metal object before handling the modules.

5 **Write-Protect Switch** Program modules have a write-protect switch to prevent changes to the data or programs in memory. Set this switch to off before using a program module for expansion memory.

6 **Battery Backup** Program modules have a built-in lithium cell to preserve the data or program in memory when the module is removed from the computer. Memory modules without this battery backup lose their memory contents when the module is replaced.

7 **INIT Command** RAM modules can be used in three ways: memory modules to expand the user memory area, and program modules as exclusive program memory or RAM disk. Use the INIT command in BASIC to designate the use of the module.

8 **For Reference**:

- Refer to the module's operation manual for information on:
  Detailed instructions for use and care; specifications; battery replacement in program modules.

- See this operation manual for information on:
  Related BASIC commands; power on/off conditions; module slots; replacement notes.

RAM Disk File Commands

The following commands are for using a program module as a RAM disk. Note that these commands are the same as the File commands used for the floppy disk. Note that TITLE, the final command in the list, is used for selecting a program module, but cannot be used when the module is a RAM disk. For full descriptions of each command see the BASIC Command Dictionary in PART IV of this manual.

Command	Description
BLOAD	Loads machine language program from RAM disk into memory.
BSAVE	Saves machine language program from memory to RAM disk.
CLOSE	Terminates access to a file.
COPY	Copies a file to or from RAM disk and external I/O device.
EOF	Returns value indicating end of file during input from a sequential file.
FILES	Displays directory information for specified files.
INIT	Initializes and specifies usage of RAM modules.
INPUT#	Reads items from a sequential file.
KILL	Deletes a file on RAM disk.
LFILES	Lists directory information on specified files on RAM disk to CE-1600P Printer or serial port.
LOAD/LOAD*	Loads a file from RAM disk into memory.
LOC	Returns the number of records read or written since file was opened.
LOF	Returns the size of the file in bytes.
MAXFILES	Specifies maximum number of files that can be open at same time.
NAME	Changes the name of a file.
OPEN	Enables data input and output to and from a file on RAM disk.
PRINT#	Writes data to a sequential file on RAM disk.
SAVE/SAVE*	Saves program or data files to RAM disk.
SET	Sets and removes file protection for files on RAM disk.
TITLE	Selects the program module in slot 1 or 2 for program execution or input. Invalid command when program module is being used as RAM disk.

ERROR Codes

This is a list of the ERROR codes that relate directly to the use of the RAM modules. See also the file and disk ERROR codes in the section on the floppy disk drive. ERROR codes are summarized in Appendix F.

Code	Description
27	Illegal command. Addressed peripheral is not connected.
101	Invalid device name specified with TITLE or NEW statement.
102	Invalid device selection (device not connected).
103	RAM module memory full. INIT parameters cannot be set.

Serial Interfaces

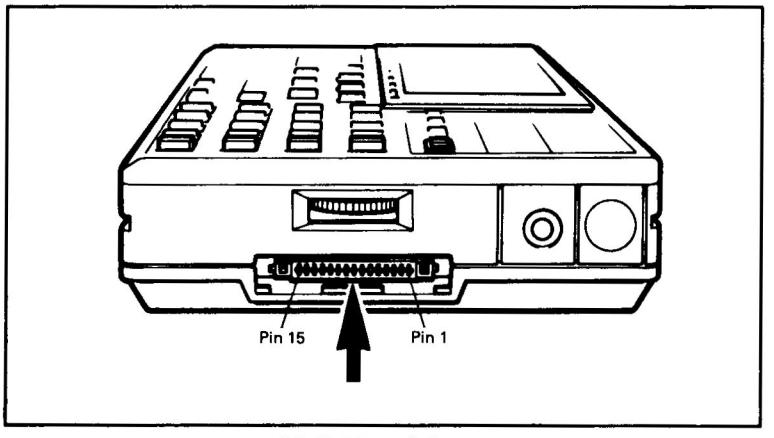
The PC-1600 has two built-in serial interfaces: RS-232C and optical. The following sections describe the use of the serial ports, give lists of specifications and connector pinouts, and summarize the related BASIC commands and ERROR codes.

RS-232C Serial Port

The RS-232C serial port conforms to the EIA standard for asynchronous data transmission between your PC-1600 pocket computer and other devices such as a serial printer, modem, or other computer.

In BASIC commands, the RS-232C port is designated COM1: and cannot be used at the same time as the optical serial port, COM2:.

The RS-232C serial port on your PC-1600 supports the CI signal from a modem telephone for automatically turning on the computer's power or receiving an interrupt.



RS-232C Serial Port

Key Points

1 **Power Off** Always turn off the computer and external device before making cable connections.

2 **Handling** Never touch the connector pins on computer, cables, or external device. Replace connector covers when not in use.

3 **Communication Protocol** Data transmission between this computer and other devices must be done according to compatible parameters. These communication parameters are set using BASIC commands. See the summary list in this section and full descriptions of each command in the BASIC Command Dictionary.

4 **Device Connection** Be sure that a device is connected to the designated port before attempting data transmission. Sending data to a non-existent device will cause the computer to "lock-up". Press the ON (Break) key to reset without losing the file.

5 **For Reference:**

- Refer to the operation manuals of the cable and connected device (other computer, printer, modem) for information on:
  Communication parameters; cable specifications; connector pinouts; data sending and receiving procedures.

- See this operation manual for information on:
  Related BASIC commands; communication protocol and parameters of your PC-1600 computer; RS-232C connector pinouts; ERROR codes.

RS-232C Default Settings

The following list gives the default, or preset, communication parameter settings for the RS-232C port, COM1:. These settings can be changed with two BASIC commands: SETCOM and PCONSOLE. See the BASIC Command Dictionary for detailed instructions on using these commands.

Command	Parameter	Default Setting
SETCOM	Baud rate (bps)	1200
	Word length (5 to 8 bits)	8
	Parity (even, odd, no)	No
	Stop bits (1 or 2)	1
	X-ON/OFF protocol	Yes
	Shift in/out protocol (7 data bits only)	Yes
PCONSOLE	Line length	Unlimited
	End of line code	CR

RS-232C Connector Pinouts

The following table lists the RS-232C pin numbers, signal names and symbols with their descriptions.

Pin No.	Signal	Symbol	Description
2	Send Data (output)	TXD (SD)	Serial data line from the computer.
3	Receive Data (input)	RXD (RD)	Serial data line from external device to the computer.
4	Request to Send (output)	RTS (RS)	Control signal from the computer.
5	Clear to Send (input)	CTS (CS)	Control signal from the external device to the computer.
6	Data Set Ready (input)	DSR (DR)	Control signal from external device to the computer.
7	Signal Ground	SG	Common ground.
8	Carrier Detect (input)	CD	Control signal from external device to the computer.
9	Calling Indicator (input)	CI	Control signal from external device to computer.
10	Logic Voltage (output)	Vcc	When computer is on, logic voltage Vcc (4 to 4.7 V) is output.
14	Data Terminal Ready (output)	DTR (ER)	Control signal from the computer.

NOTES:

1 Output control signals RTS and DTR can be set high or low by a BASIC command. See OUTSTAT in the Command Dictionary.

2 High or low states of input control signals CTS, DSR, CD, and CI can be read using the BASIC command INSTAT. See INSTAT in the Command Dictionary.

3 Input control signal CI can interrupt the computer when it is either on or off. This signal turns the computer on when it is off. See ON PHONE GOSUB, PHONE ON/OFF/STOP, and WAKE$ in the Command Dictionary.

RS-232C I/O Commands

The following commands are for use of the RS-232C port, designated COM1:, for serial data communications. For full descriptions of each command see the BASIC Command Dictionary in PART IV of this manual.

Command	Description
COM$	Returns string of communication parameter values specified in SETCOM statement.
COMn ON/OFF/STOP	Enables or disables received interrupts.
INIT	Sets size of receive buffer.
INSTAT	Returns settings of control signals.
ON COMn GOSUB	Causes execution to jump to specified line of a sub-routine when interrupt is received.
ON PHONE GOSUB	Causes execution to jump to specified line number when input is received from telephone line.
OUTSTAT	Sets high/low states of control signals.
PCONSOLE	Sets line length and EOL code.
PHONE ON/OFF/STOP	Enables and disables interrupts received from telephone modem.
PZONE	Sets print zone for output with LPRINT statement.
RCVSTAT	Sets receive handshake protocol and timeout value.
SETCOM	Sets communication protocol.
SETDEV	Specifies RS-232C port for input and output.
SNDBRK	Sends break codes to halt input transmission.
SNDSTAT	Sets send handshake protocol and timeout value.

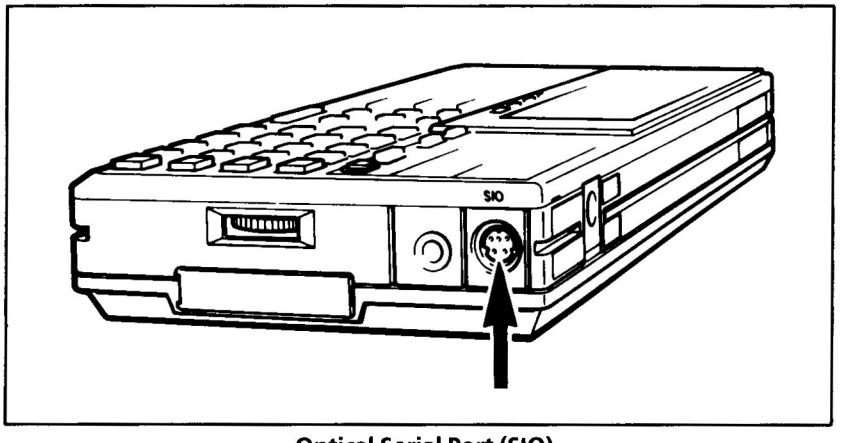
ERROR Codes

This is a list of the ERROR codes that relate directly to the use of both serial interfaces: RS-232C (COM1:) and optical serial (COM2:) ports during data communications. ERROR codes are summarized in Appendix F.

Code	Description
140	Invalid parameters set in SETCOM statement.
141	Size of receive buffer specified in INIT statement is too large (greater than 16383 bytes or available memory).
142	Error during data reception (parity error, overrun error, framing error, receive buffer full error).
143	Timeout error. No response from other party for longer than the time value set in the RCVSTAT or SNDSTAT statements.
144	Serial port specified in SETDEV statement is already open.

Optical Serial Port

Your PC-1600 supports optical serial transmission of data to and from an external device via a 5-pin connector, labeled SIO, on the right side of the computer.



Optical Serial Port (SIO)

This section describes the connector pinouts, gives a related BASIC command summary and lists key points for connection and use.

The use of the optical serial port is similar to the EIA standard RS-232C serial port, but provides distinct advantages. One is the higher transmission speed of up to 38400 baud in half-duplex mode for your PC-1600.

Optical transmission also offers a means for sending and receiving serial data in adverse environments as the optical CE-1600L Optical Fiber Cable provides electrically insulated, noise-free transfer of data.

In BASIC commands, the optical serial port is designated COM2:, and cannot be used at the same time as the RS-232C port, COM1:.

Key Points

1 **Power Off** Always turn off the computer and external device before making cable connections.

2 **Handling** Never touch the connector pins on the computer, cables, or external device. Replace connector covers when not in use. Optical fibers in the cable are damaged when the cable is bent severely, causing data errors and transmission losses. Maintain fully rounded curves when bending can't be avoided.

3 **Communication Protocol** Data transmission between this computer and other devices over an optical serial line must be done according to compatible parameters. These communication parameters are set using BASIC commands. See the summary list in this section and full descriptions of each command in the BASIC Command Dictionary.

4 **Device Connection** Be sure that a device is connected to the designated port before attempting data transmission. Sending data to a non-existent device will cause the computer to "lock-up". Press the ON (BREAK) key to reset without losing the file.

5 **For Reference:**

- Refer to the operation manuals of the cable and connected device (other computer, etc.) for information on:
  Communication parameters; cable specifications; connector pinouts; data sending and receiving procedures.

- See this operation manual for information on:
  Related BASIC commands; communication protocol and parameters of your PC-1600 computer; optical serial connector pinouts; ERROR codes.

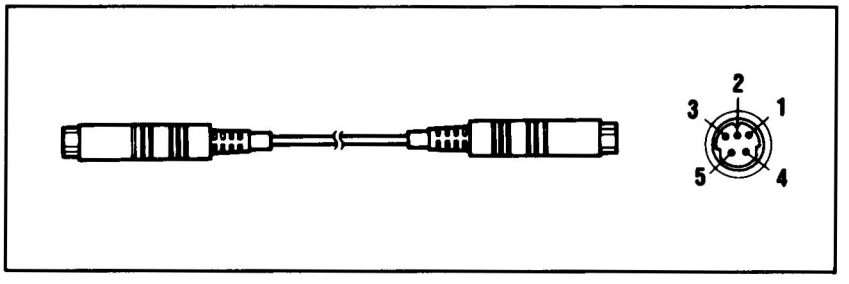
Optical Serial Default Settings

The following list gives the default communication parameter settings for the optical serial port, COM2:. These settings can be changed with two BASIC commands: SETCOM and PCONSOLE. See the BASIC Command Dictionary for detailed instructions on using these commands.

Command	Parameter	Default Setting
SETCOM	Baud rate (bps)	38400
	Word length (5 - 8 bits)	7
	Parity (even, odd, no)	Even
	Stop bits (1 or 2)	2
	X-ON/OFF protocol	Yes
	Shift in/out protocol (7 data bits only)	Yes
PCONSOLE	Line length	Unlimited
	End of line code	CR

Optical Serial Connector Pinouts

The following illustration shows the CE-1600L Optical Fiber Cable and pinouts of the 5-pin connector used for the optical serial I/O port (SIO). The table thereafter lists the pin numbers, signal names and symbols with their descriptions.



CE-1600L Optical Fiber Cable and Connector Pinouts

Pinouts and Signal Descriptions

Pin No.	Signal	Symbol	Description
1	Receive Data (input)	RD	Serial data line from external device to the computer.
2	Ground	GND	Common ground.
3	Send Data (output)	SD	Serial data line from the computer.
4	Logic Voltage (output)	Vcc	Logic voltage Vcc (4 to 5.5 V) is output when the computer is on.
5	Logic Voltage (output)	Vcc	Same as above.

Optical Serial I/O Commands

The following commands are for using the optical serial port, designated COM2:, for serial data communications. For full descriptions of each command see the BASIC Command Dictionary in PART IV of this manual.

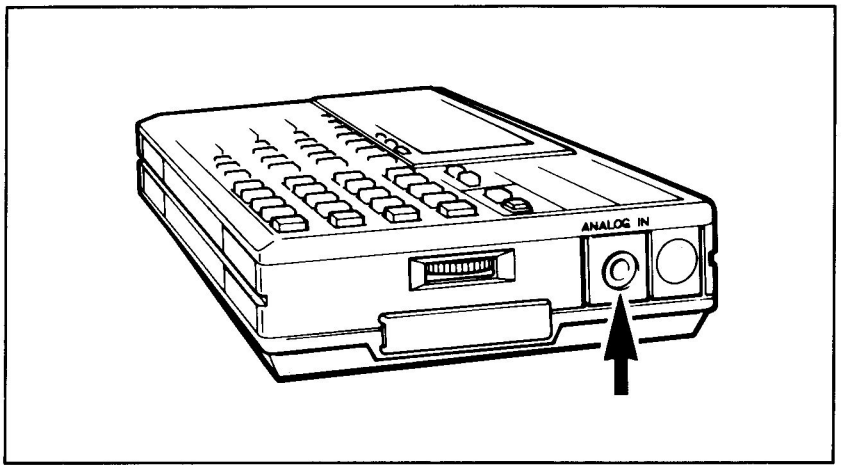
Command	Description
COM$	Returns string of communication parameter values specified in SETCOM statement.
COMn ON/OFF/STOP	Enables and disables received interrupts.
INIT	Sets size of receive buffer.
ON COMn GOSUB	Causes execution to jump to specified line of a sub-routine when interrupt is received.
PCONSOLE	Sets line length and EOL code.
PZONE	Sets print zone for output with LPRINT statement.
RCVSTAT	Sets receive timeout value.
SETCOM	Sets communication protocol.
SETDEV	Specifies optical serial port for input and output.
SNDBRK	Sends break codes to halt input transmission.
SNDSTAT	Sets send timeout value.

ERROR Codes

The list of ERROR codes in the foregoing RS-232C description covers possible errors occurring for both RS-232C and optical serial ports. See this list for ERROR codes when you are using the optical serial I/O port for data communications. ERROR codes are fully summarized in Appendix F.

Analog Input Port

Your PC-1600 computer can receive analog signals from external devices such as sensors via an analog input port, marked ANALOG IN, on the right side of the computer.



Analog Input Port (ANALOG IN)

This section describes the specification of the analog input port and lists related BASIC commands.

Using BASIC commands, you can convert analog signals into the equivalent digital level for recording measurements from analog sources and for initiating execution of subroutines in a program.

Electrical Characteristics

The effective use of the analog input port depends on the combination of the analog signal source device and software control in BASIC (or machine language) for accurately converting to the equivalent digital value. The following table lists the electrical specifications of the analog port.

Analog Input Specifications

Maximum Input Voltage	4 V
Input voltage range	0 to 2.495 V
Converted digital level	0 to 255 (8 bits) (Any input voltage over 2.495 V is converted to 255.)
Conversion error	3% ±2 digits
Input impedance	100 kΩ ±2%

IMPORTANT: *To prevent damage to the computer and analog input port, make sure that the input voltage is maintained within the range of 0 to 4 V.*

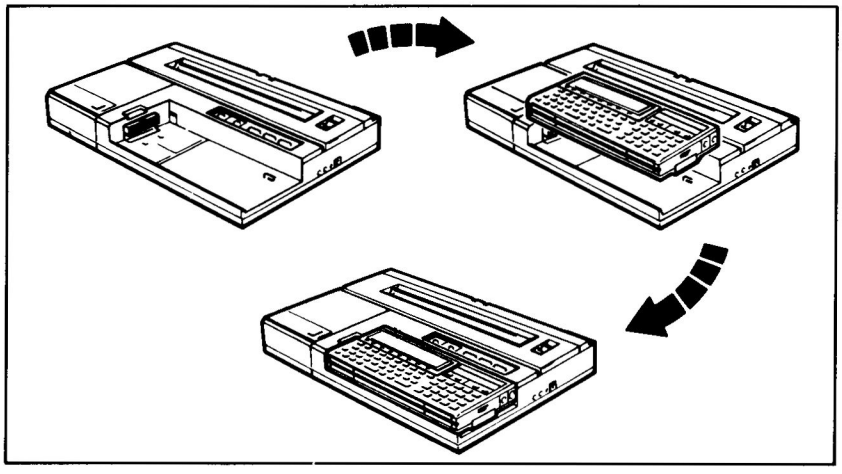
Analog Input Commands

The following commands are for using the analog input port to receive data from an external analog device. For full descriptions of each command see the BASIC Command Dictionary in PART IV of this manual.

Command	Description
ADIN ON/OFF/STOP	Enables and disables received interrupts.
AIN	Returns a value corresponding to input analog voltage level.
ON ADIN GOSUB	Causes execution to jump to a specified line number when input voltage is outside a fixed range.

Printer with Cassette Interface

The CE-1600P Printer provides your pocket computer with the capability of four-color printouts of text and graphics, and is the base upon which you can connect other options.



Connecting to the CE-1600P Printer

This section describes the key points for using the printer and its built-in cassette interface, and lists the related BASIC commands and ERROR codes.

The CE-1600P Printer is designed to accept the PC-1600 computer in a slide-in connection for a smooth integrated profile. The optional CE-1600F Floppy Disk Drive also connects to the printer body in similar fashion to complete your basic

pocket computer system. The direct connection of the printer to the computer's system bus and the connection of the floppy drive eliminate the need for separate cables between devices.

The printer runs on built-in rechargeable NiCad batteries which also provide the power for the floppy disk drive and to the computer, too, if necessary. An AC adaptor, EA-160, comes as a standard accessory for operating the printer from an electrical outlet when indoors, or to conserve the batteries.

Key Points

1 **Power On/Off** The printer does not have on and off switches. When connected to the computer, the printer is powered when the computer's power is turned on and off. Always turn off the computer before connecting or disconnecting it from the printer.

2 **Handling** Never touch the connector pins on the printer or system bus of the computer. Be sure to ground yourself on a metal object when static electricity may be a problem. Replace connector covers when not in use.

3 **Low Batteries** Be sure to use the AC adaptor to charge the printer batteries at initial use. Your PC-1600 computer has three ways to alert you of low batteries: the BATT symbol on the status line of the display, a CHECK message at power-on, and ERROR codes during execution.

4 **RESET Switch** When the computer is connected to the printer, the RESET switch on the underside of the computer is inaccessible. A RESET switch is provided on the back of the printer body for the same purpose. See Resetting the Computer in Chapter 3.

5 **Cassette Interface** The built-in cassette interface allows connection of a data recorder such as a cassette recorder. The remote switch on the printer lets you control power to the tape recorder by a BASIC command. See the next section on Using a Tape Recorder.

6 **For Reference:**

- Refer to the CE-1600P Printer operation manual for information on:
  Details on use and care; pen replacement; charging batteries; paper setting and adjustments; overall specifications and accessory lists.

- See this operation manual for information on:
  Printer related BASIC commands; ERROR codes; setting Graphic and Text modes.

Printer Commands

The following commands are used for writing data to the printer, setting various print format values, and setting print modes. For full descriptions of each command see the BASIC Command Dictionary in PART IV of this manual.

Command	Description
COLOR	Sets pen color.
CSIZE	Sets character size for printing.
GLCURSOR	Moves pen to specified position in Graphic mode.
GRAPH	Sets printer in Graphic mode.
LCURSOR	Moves pen to specified column in Text mode.
LF	Feeds paper a specified number of lines.
LLINE	Draws a line or line segment series between absolute coordinates.
LLIST/LLIST*	Lists program lines to printer.
LPRINT	Outputs data to printer.
PAPER	Sets paper type and vertical print range.
PCONSOLE	Sets print format and EOL code for data to printer.
PITCH	Sets character pitch and line spacing in Text mode.
PZONE	Sets the print zone for output with the LPRINT statement.
RLINE	Draws a line or line segment series between relative coordinates.
ROTATE	Sets print orientation of characters.
SORGN	Sets current pen position as origin.
TAB	Moves pen to specified column in Text mode.
TEST	Runs a four-color print test.
TEXT	Sets printer in Text mode.

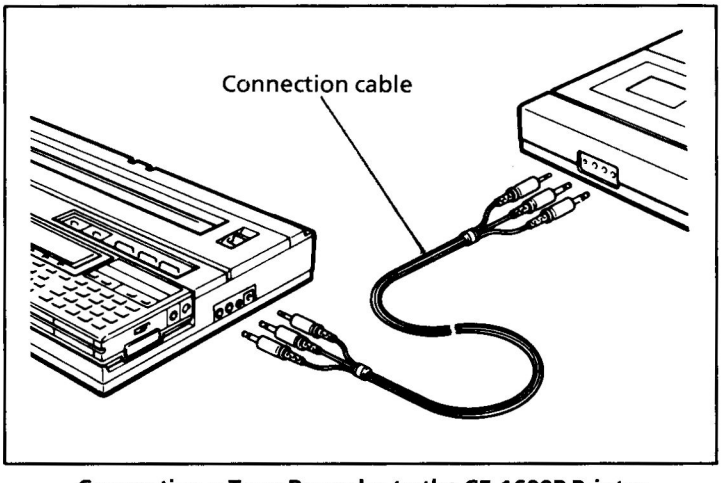
ERROR Codes

This is a list of ERROR codes that relate directly to the use of the printer. For an overall summary of all ERROR codes, see Appendix F.

Code	Description
70	Pen outside range −2048 to 2047 for X or Y.
71	Reverse form feed more than 10.24 cm in Text mode (CE-150 only).
72	Incorrect specification of TAB parameter.
73	Illegal command in current mode. Graphics command used in Text mode or vice versa.
74	Too many parameters in LLINE or RLINE statement.
76	Attempt to LLIST a program line longer than PCONSOLE setting of 17 or below.
78	Low battery, pen not in place, or printer locked. Cannot execute LPRINT or LLINE command.
79	Color signals not output to printer (CE-150 only).
80	Low battery, printer locked up.

Using a Cassette Recorder

You can connect and operate a data recorder, such as a cassette recorder, via the interface on the CE-1600P Printer. This section describes the BASIC commands for writing and reading data to and from tape, and lists the ERROR codes and key points for connection and use. The following illustration shows the cable connection at the printer and recorder.



Connecting a Tape Recorder to the CE-1600P Printer

Key Points

1 **Power Off** Be sure to turn off the computer and recorder before making cable connections.

2 **Cable Connections** The CE-1600P Printer has three jacks on its right side for connecting cables from a tape recorder. The jacks are labeled: REM (Remote) for remote control of the recorder's power, MIC (Microphone) for writing, or recording, data from the computer onto tape, and EAR (Earphone) for reading, or playing back, data from the tape into the computer. The cable that comes with the CE-1600P Printer with Cassette Interface is color-coded for easy connection. The miniplugs that go into the EAR and MIC jacks are the same size. Check to make sure you haven't misplugged them if you have trouble.

3 **Remote Control** Make sure that the REMOTE switch on the printer is set ON if you want to be able to turn the recorder's power on and off with the RMT ON/OFF BASIC command.

4 **Cassette Recorders** The output of some cassette recorders is incompatible with the standard system configuration of the PC-1600. This problem is indicated when Error 44 is displayed during an attempt to read data from tape. To enable the PC-1600 to read data from such a cassette recorder, enter the following command:

POKE &F1A9, 0 ENTER

IMPORTANT: *Be very careful when entering the command as you are making changes to system data. If you have mistyped and press* ENTER, *you must perform an ALL RESET of the computer which would clear all memory of any current programs and data.*

To enable the PC-1600 to read another cassette recorder, the standard system configuration by entering the following command:

POKE &F1A9, 128 ENTER

If you attempt to use another cassette recorder without entering the above command, Error 44 will be displayed.

5 For Reference:

- Refer to the operation manual of the CE-1600P Printer, cassette recorder, or other data recorder for information on:
  Recorder cables and connections; specifications, general care and maintenance; operating procedures.

- See this operation manual for information on:
  Using BASIC commands to write and read data to and from tape; ERROR codes.

Cassette Tape I/O Commands

The following is a list of the BASIC commands you can use to read and write data to and from tape, verify the correctness of data recorded on tape, and turn the power to the recorder on and off remotely from the computer. For detailed descriptions of each command, see the BASIC Command Dictionary in PART IV of this manual.

Command	Description
BLOAD	Loads machine language program from tape into memory.
BSAVE	Saves machine language program from memory to tape.
CHAIN	Allows a BASIC program to load and execute another program on tape.
CLOAD	Loads a program or function key usage program into memory from tape (PC-1500 compatible).
CLOAD?	Verifies error-free recording of program on tape (PC-1500 compatible).
CLOADM	Loads machine language program into memory from tape (PC-1500 compatible).
CLOSE	Terminates access to a file.
COPY	Copies a file to or from tape and external I/O device.
CSAVE	Saves entire or part of program from memory to tape (PC-1500 compatible).
CSAVEM	Saves machine language program from memory to tape (PC-1500 compatible).
EOF	Returns value indicating end of file during input of sequential file on tape.
INPUT#	Read items from a sequential file on tape.
LOAD/LOAD*	Loads a file from tape into memory.
MAXFILES	Specifies maximum number of files that can be open at same time.
MERGE	Merges program saved on tape with a program in memory.
OPEN	Enables data input and output to and from a file on tape.

Command	Description
PRINT#	Writes data to a sequential output file on tape.
RMT ON/OFF	Enables and disables remote control of cassette recorder power.
SAVE/SAVE*	Saves program or data files to tape.

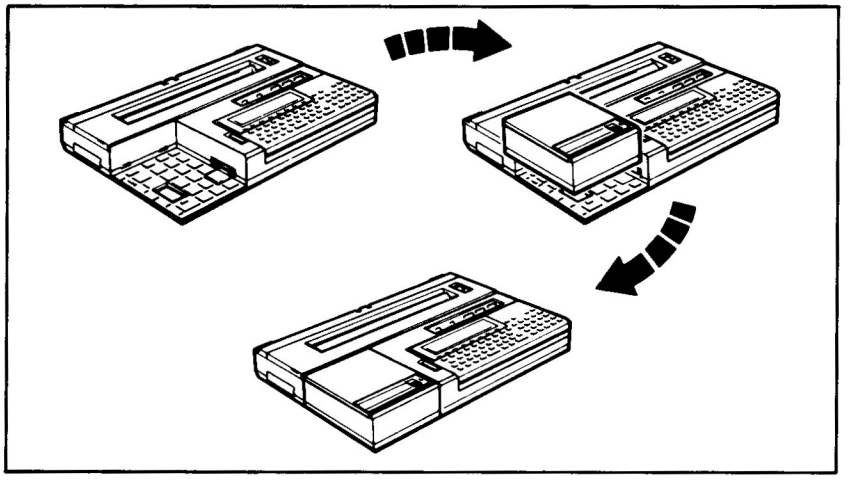
ERROR Codes

This is a list of ERROR codes that relate directly to the use of a tape recorder to store and load data. For full descriptions of all ERROR codes, see Appendix F.

Code	Description
40	Syntax error in tape command.
42	Insufficient memory to load the program.
43	Tape verification error. After executing CLOAD?, the copy in memory does not agree with the original on tape. Possible tape misread. Load and verify again. Type mismatch with INPUT# statement. Variable type and data type do not agree. Try to read string data into numeric variable or vice versa.
44	Incompatible tape recorder output, or tape read error.

Floppy Disk Drive

The addition of the optional CE-1600F Floppy Disk Drive to your computer and printer combination completes the basic setup of your pocket computer system.



Connecting the CE-1600F Floppy Disk Drive to the CE-1600P Printer

This section describes the key points for connection and use of the disk drive and lists the related BASIC commands and ERROR codes.

The double-sided CE-1650F Pocket Disks used in the drive can store up to 61K of data on each side, making them an efficient tool for working with larger amounts of data that limit the use of tape or program RAM modules. Even though the disks are in the small 2.5″ format, their large storage capability and similar BASIC commands make them as easy to use as the larger formats.

The disk drive itself connects directly to the printer, eliminating the need for cables. When both computer and disk drive are connected to the printer, the completed unit has an integrated, all-in-one appearance.

Key Points

1 **Power Off** Be sure to turn off the computer before connecting or disconnecting the disk drive. Never touch the pins on the connectors, and always replace connector covers when not in use.

2 **Disk Formatting** New disks must be formatted before use. The disks are double-sided and each side must be formatted separately. The side of the disk that is facing up is the one being accessed. The disk must be ejected and turned over to access the other side. To format a disk, type in:

INIT "X:" and press **ENTER** .

Formatting the disk takes less than a minute per side. When the green lamp goes off, eject the disk, turn it over and repeat for the other side.

3 **Disk Handling** While the disk is being accessed a green lamp is lit on the top of the drive. Never eject the disk during this time as the spinning of the drive can damage the disk and any data stored on it, as well as the drive mechanism itself. Also, do not eject a disk while a file is open.

4 **Power** The disk drive does not have its own power supply but is powered by the printer's rechargeable batteries. If low batteries cause the BATT warning indicator to light on the LCD's status line, stop using the computer and connect the AC adaptor for recharging or direct use of AC power. Continued use of the drive after the BATT indicator lights may result in errors and loss of data.

5 **Files** The CE-1650F Floppy Disks can store up to 48 files, or 61K, on each side. All files are closed when the power of the computer is turned off, both manually by pressing the **OFF** key, and by sending a POWER statement in BASIC.

6 Backup Disks The version of BASIC used in your SHARP PC-1600 Pocket Computer recognizes two output device names for the floppy disk drive, X: and Y:. The default name is X:, but you may specify either. Using the two device names for the drive, you can easily copy disks containing important programs and data. Note that you must copy files one at a time and specify the file name extension. The wildcard file name extension * cannot be used to specify all of the files on disk for copying. Suppose that you have a disk in the drive and you've specified it with the device name X:. To copy a file on this disk to another disk, type in the following command:

COPY "X:<filename.ext>" TO "Y:<filename.ext>"

When you then press ENTER, this command tells the computer to copy the specified file on the disk in drive X: to a disk in drive Y:. As both X: and Y: device names can be used for the same drive, the computer will copy the file from the current disk into memory, then ask you to insert a new disk with the following prompt:

Set diskette for Y: (or vice versa)

When you see this message, eject the current disk, insert a new disk, and press ENTER. The specified file on the original disk is now copied from memory onto the new backup disk. When a file is completely copied, the drive stops operating, the green lamp goes out, and the system prompt > appears on the screen. Continue copying any or all files remaining on the original disk. When finished, you can eject the backup disk and store it, and continue using the computer. See COPY and FILES in the BASIC Command Dictionary for more information on copying files.

7 For Reference:

- Refer to the operation manuals of the CE-1600P Printer and CE-1600F Floppy Disk Drive for information on:
  Connection of the drive; inserting/ejecting disks; write protection; specifications; general care and operating procedures.

- See this operation manual for information on:
  BASIC commands for writing and reading data on disk; ERROR codes; creating and editing files in BASIC.

Disk I/O Commands

The following is a list of BASIC commands that are used for reading and writing files to and from disk. For detailed descriptions of each command, see the BASIC Command Dictionary in PART IV of this manual.

Command	Description
BLOAD	Loads machine language program from floppy disk into memory.
BSAVE	Saves machine language program from memory to floppy disk.
CLOSE	Terminates access to a file.
COPY	Copies a file to or from floppy disk and external I/O device.
EOF	Returns value indicating end of file during input from a sequential file.
FILES	Displays directory information for specified files.
INIT	Initializes and formats a blank floppy disk.
INPUT#	Reads items from a sequential file.
KILL	Deletes a file on disk.
LFILES	Lists directory information on specified files on disk to printer or serial port.
LOAD/LOAD*	Loads a file from floppy disk into memory.
LOC	Returns the number of records read or written since file was opened.
LOF	Returns the size of the file in bytes.
MAXFILES	Specifies maximum number of files that can be open at same time.
NAME	Changes the name of a file.
OPEN	Enables data input and output to and from a file on floppy disk.
PRINT#	Writes data to a sequential file on floppy disk.
SAVE/SAVE*	Saves program or data files to floppy disk.
SET	Sets and removes file protection for files on floppy disk.

ERROR Codes

This is a list of ERROR codes that relate directly to the use of the disk drive to store and load data. For full descriptions of all ERROR codes, see Appendix F.

Code	Description
150	Too many files specified in MAXFILES statement.
151	File already exists. Use another file name.
152	File not found. Double-check file name specification.
153	Incorrect file number in file read or write statement. Specified file not yet opened.
154	File already open. Close and re-open in new mode.
155	Illegal drive name, or drive not connected.
156	Incorrect parameter specification in SET statement.
157	Illegal file name, or incorrect specification.
158	Command not supported. Illegal command for disk drive.
159	Attempt made to write to a write-protected disk.
160	Disk not in specified drive.
161	Disk not formatted with INIT statement.
162	Disk read or write error.
163	Wrong disk in drive. Disks were changed while file was open.
164	Disk full error.
165	End of file reached with INPUT# statement. All data has been read.
166	Insufficient memory for disk work area. Not enough free space in computer's internal memory for disk drive I/O system.
167	Fatal disk error. Contents destroyed or corrupted.
168	Low battery in printer or disk drive hardware fault.

7 Using PC-1500 Peripherals

The key to the flexibility of your PC-1600 is its compatibility with the SHARP PC-1500 Pocket Computer. This means that it can share most of the PC-1500's wide range of peripheral devices. These options are briefly described on these pages. See your SHARP dealer for availability and operating hints. All optional devices come with their own operation manuals. When using any of these devices with your PC-1600, be sure to check the manuals for details.

In this manual we've already referred to some of these options, RAM modules and cassette recorder. Note that your PC-1600 can emulate the PC-1500 when it is set to MODE 1. For information on this, see Chapter 9 in the BASIC Reference Section.

Also see Appendix H on compatibility of BASIC commands and peripherals between the PC-1600 and PC-1500.

The following is a list of the PC-1500 peripherals you can use with your PC-1600 Pocket Computer.

- CE-150 Printer with Cassette Interface

  A smaller version of the CE-1600P Printer; offers 4-color printing of text and graphics; built-in cassette interface allows connection of two tape recorders.

- CE-151 RAM Module

  4K Memory Module for expanding user memory area.

- CE-152 Cassette Tape Recorder

  Connects to cassette interfaces on CE-1600P or CE-150 Printers, or to CE-162E Cassette and Parallel Interface Unit.

- CE-153 Software keyboard

  Allows assignment of function keys for easier input of repetitive key-ins on a 10 × 14 matrix touch pad.

- CE-155 RAM Module

  8K Memory Module for expanding user memory area.

- CE-158 Serial and Parallel Interface Unit

  Provides RS-232C serial interface for communicating with other external devices such as another computer, modem, or printer. Centronics parallel interface allows connection of a parallel printer.

- CE-159 RAM Module

  8K Program module with write-protect switch and backup battery.

- CE-161 RAM Module

  16K Program Module with write-protect switch and backup battery.

- CE-162E Parallel and Cassette Interface Unit

  Allows connection of parallel printer and data recorder.

- EA-160 AC Adaptor

  Standard accessory with CE-1600P Printer.

- EA-150 AC Adaptor

  Standard accessory with CE-150 printer.

PART IV

BASIC REFERENCE SECTION

This part describes all aspects of the specially adapted version of SHARP BASIC which is the key to the exceptional versatility of the PC-1600 Pocket Computer.

Chapter 8 introduces some programming fundamentals. Chapter 9 gives an overview of the different modes in which the PC-1600 operates under BASIC. Chapter 10 deals with general data handling concepts and available functions.

Chapter 11 deals with more advanced file-handling techniques. Chapter 12 covers serial communication with other devices. Chapter 13 explains some of the simpler debugging techniques.

Chapter 14 lists all the BASIC commands and functions in alphabetical order.

8 Basic Programming Concepts

A program is a logical series of instructions which control the operation of a computer. These instructions must be written in a language that the computer has been designed to understand.

At the most complex level, the PC-1600 understands a language called machine code. However, using its own built-in "interpreter" it is able to converse in a language more understandable to users and convert the instructions into machine code internally and automatically for its own use. It is also possible for advanced users to program the computer directly in machine code.

For most people, however, BASIC will be the language in which programs are written. BASIC comprises about 200 or so individual instructions referred to as commands, statements or functions. The rules of grammar (punctuation and sentence structure) which govern the combining of individual commands are called syntax.

Entering BASIC Commands

There are two modes of entry for many BASIC commands; the direct mode and the indirect or program mode. In the direct mode, commands are typed into the computer and are executed immediately when the ENTER key is pressed. The result of the execution will be immediate. After the command has been executed, the computer is ready to receive another direct command.

This mode is similar to the direct calculation mode of use described in Chapter 5, and is useful for simple, non-repetitive calculations. It is also used for issuing commands to control peripheral devices such as the disk drive. Some commands are not available in direct mode. Also only one command can be entered at a time in this mode. If more than one command is entered before pressing the ENTER key, an error will result [error code 1], or in certain cases, the first command only will be accepted.

In the program mode, commands are put together to form program lines, and each line is allocated a number at its head in the order that the lines are to be executed. These program lines are stored in the computer as a program, but not executed until the user enters RUN, at which time, the computer starts with the lowest numbered program line and executes it. It then continues with the next number in sequence, and so on until there are no more lines left to execute.

The size of a program; that is, the number of program lines, is limited only by the size of your computer's memory. Some commands are not available in program mode.

Here is a simple BASIC program with each line numbered in steps of 10. This is good programming practice as it allows lines to be inserted if needed between the existing lines at a later date.

```
10 PRINT "THIS PROGRAM IS A SIMPLE"
20 PRINT "BASIC PROGRAM TO ILLUSTRATE"
30 PRINT "THE USE OF LINE NUMBERS"
40 LET A = 2
50 LET B = 4
60 LET C = A + B
70 PRINT C
80 END
```

In actual use, the only difference between the direct mode and the indirect or program mode is that in the program mode, all commands and statements must be preceded by line numbers. The computer switches between the two modes automatically depending on whether a line number is present or not.

However, in order to execute a program which has been entered in program mode, the computer must be switched to RUN mode with the MODE key. The computer indicates that it is ready for the next task in direct or program modes by displaying a prompt sign (>) at the start of the line on the screen.

Running a Program

Programs can only be executed with the computer in the RUN mode. There are several methods that can be used to start the execution of a program which is in the computer's memory. The most frequently used method is to enter the RUN command. When this command is entered and the ENTER key is pressed, the computer starts to execute the program line by line from the lowest line number in memory.

For the sample program, execution will start at line 10 and go on to 20, 30, and so on until it reaches the END statement on line 80. Then it will stop. Type in program lines 10 to 80 in the PRO mode, and then switch the mode to RUN and run the program. Notice that during program entry, after you press the ENTER key at the end of each line, the computer inserts a colon after the line number.

Other methods are available to start execution of a program from a particular line or to resume execution after the computer has halted for some reason (see descriptions of the CONT, GOTO and ARUN commands in the Command Dictionary).

Program Labels and the DEF Key:

One other very flexible way to start program execution makes use of the DEF key positioned to the left of the function keys at the top of the keyboard. This key enables a program to be executed starting from a label positioned on the first line of the program. The label must be a single upper-case letter from the keys on the bottom two rows of the keyboard (ASDFGHJKLZXCVBNM) and the = or SPACE keys.

A program line with a label looks like this:

```
10:"A"
```

Notice that the label A is enclosed in quotes and comes immediately after the line number.

A program which starts with a label on the first line can be executed by pressing the DEF key followed by the correct label key. Type in a new line 10 for the sample program. First put the computer in PRO mode again, and then enter:

```
10:"A":PRINT "THIS PROGRAM IS A SIMPLE" ENTER
```

When you press the ENTER key, the new line 10 will replace the existing line 10 automatically. Now switch back to RUN mode and execute the program by pressing DEF + A .

This method of running a program enables several programs to be held in the computer's memory, each with a different label on the first line. A particular program can be run by inputting the correct label. Labels can also be used in many of the BASIC commands to get to particular program lines quickly. Refer to the descriptions of the LIST, DELETE, and MERGE commands in the Command Dictionary for more information on use of the label feature.

Memory Allocation

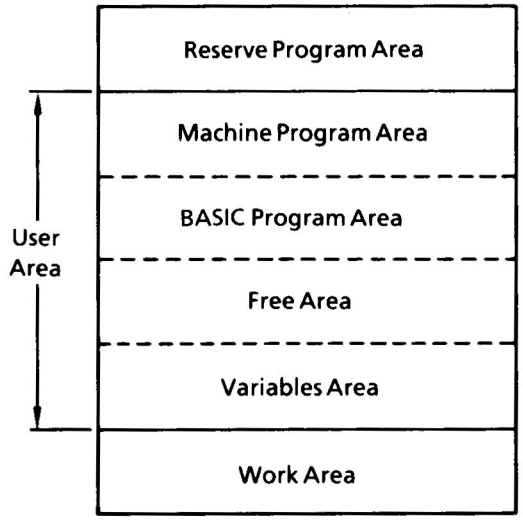
Those areas of the PC-1600's memory which are available to the user for storing and executing BASIC programs can be allocated and cleared by BASIC commands. The relevant commands are:

CLEAR, ERASE, MEM, NEW, STATUS, TITLE

Refer also to the memory maps in Appendix D for allocation of memory.

Available Memory Areas

The built-in RAM (random access memory) in the PC-1600 itself will allocate a maximum of 11834 bytes (approx. 12K) to the user for BASIC (or machine language) programming. There are of course other internal memory areas in the PC-1600, but they are used by the system which controls operation of the PC-1600, the disk drive, and other devices. They are not normally available to the user except in very special applications.



1 User Area

The user area includes space for storage of variables used by a program, and programs which have been loaded into memory from disk or tape, or typed in from the keyboard. With no modules in the expansion slots 1 or 2, the size of the user area is 11834 bytes.

When a RAM module is inserted in either slot 1 or slot 2 to be used as memory expansion, the size of the user area is increased; the internal user area extends out into the memory module. This also happens when a program module is inserted with only part of the module used as memory expansion.

In order to find out how much user area is free at any time before or during operation, use the MEM command. The MEM command only tells you how much free user area is left, so this will be influenced by the size of any programs you have stored in the computer itself. For instance, if you have a 4K memory module in slot 1, then the size of the user area is increased from 12K to 12K + 4K = 16K. But if you load a 3K BASIC program into the computer from disk, there will only be 16K − 3K = 13K of user area free. The MEM command will show the size in bytes as about 13000 bytes or so.

When the CE-1600F Pocket Disk Drive is connected to the Computer, some of the user area is reserved for the disk input/output system. As a result the user area available for programs is reduced to 10810 bytes.

2 Variable Storage Area

It is not possible to calculate exactly the amount which will be free when there is a program in the user area, because some of the memory is used to store variables for the program, and this amount depends on the program itself. However, if you want to clear the area allocated to store these variables (after you have executed a program which uses a lot of arrays for instance), you can do so with the CLEAR command. Refer to the Command Dictionary for details.

3 Machine Language Area

Part of the user area can also be allocated for special machine language programs. This should only affect the advanced user; the amount is allocated with the NEW command. When you type in NEW0 after resetting the computer (see Chapter 3), the user area is completely cleared, and no space is allocated for machine language programs. After this action, the MEM command should show a user area of exactly 11834 bytes.

4 System Work Area

The work area is not accessible to the user. It is the area where the computer actually carries out the processing steps specified in your program.

5. Reserve Program Area

This area is where the programmable function key contents are stored. When the computer is set in the RESERVE mode with SHIFT + MODE, function key setting programs are loaded directly into this special area, or can be saved to a device from this area. The size of the reserve area is fixed.

AUTORUN Files

The AUTORUN function allows BASIC programs which have been saved on a floppy disk on the CE-1600F or on the RAM disk in the PC-1600 to be executed automatically when the power is turned on. In order for a program to be run automatically with this function, it must have been saved with the name AUTORUN.BAS. Refer to Chapter 11-Files, for detailed information on file names.

A program can also be executed automatically using the ARUN command in BASIC (see page 123). The difference in the two functions is that for ARUN, the program must be already in the PC-1600's internal memory, and can have any name, whereas with the AUTORUN function, the program can be on floppy disk or RAM disk and must have the name AUTORUN.BAS.

To set up the PC-1600 to execute an AUTORUN file, follow the steps below:

1. Save the program with the name AUTORUN.BAS on either floppy disk or RAM disk.
2. Put the PC-1600 in RUN mode with the MODE key.
3. Ensure that the **SMALL** and S displays are not lit on the status line. If they are, the AUTORUN function will not work.
4. Turn the power off normally.

When the power is turned on the next time, the screen will display the message:

LOAD"X:AUTORUN.BAS",R (if the file is on floppy disk)

as if you had typed it in from the keybaord, and will then load the file (see LOAD command on page 216) and execute the program automatically. If the file is on RAM disk, then S1: or S2: will be displayed in place of X:

Different AUTORUN files can be stored on both floppy disk and RAM disk at the same time. In this case, an AUTORUN file on floppy disk (drive X: or drive Y:) will be given priority over any AUTORUN file on RAM disk. Also, an AUTORUN file on RAM disk in slot S1: will be given priority over an AUTORUN file on RAM disk in slot S2:

When an AUTORUN file is loaded into the PC-1600's internal memory, it will be loaded into the same memory area or memory expansion area which was selected before the power was turned off.

9 Operating Modes under BASIC

Key Operations

The keys on the computer are laid out similarly to a typewriter or other computers, but there are some special keys and key combinations which are especially useful in BASIC. In addition, certain keys and key functions can be controlled from within a BASIC program.

Alphanumeric Keys

The alphabetic keys are arranged as a normal typewriter keyboard and can be set to auto-repeat; that is the character is generated repeatedly if the key is held down. The auto-repeat function is switched on or off by the KEYSTAT command. This command also sets all the keyboard keys to issue a "click" when pressed if this is required. The numeric keys are arranged as a keypad to the right of the keyboard.

Switch Keys

The switch keys allow the normal keys to perform more than one function. The commonest of these is the SHIFT key, found on most typewriters to allow the same key to give upper and lower case. The function of the SHIFT key on the computer is a little different to this. There are eight switch keys on the computer's keyboard.

MODE — The MODE key switches the computer between the RUN and PROgram modes. The RUN mode is the direct mode for executing BASIC programs and commands. The PROgram mode is the statement mode where programs are entered, listed and edited. In addition, there is a RESERVE mode which is used exclusively for the programmable function keys.

SHIFT + MODE — The computer enters the RESERVE programming mode. In this mode, the contents of the programmable function keys may be set, changed, and edited. Press the MODE key alone to exit the RESERVE mode.

SHIFT

The **SHIFT** key is a toggle; press it once and it remains effective, with SHIFT displayed at the top of the screen, until a second key is pressed. The second key takes on the function written above the key in orange instead of the normal function, and SHIFT is turned off. To use the shifted function of more than one key in succession, the **SHIFT** key must be pressed before each separate key entry.

CTRL

The **CTRL** (control) key is used with the **↑** **↓** **▶** **◀** **A** **D** **E** **F** **H** **R** **X** keys for editing functions (see Edit Mode). It works as a toggle key in the same way as the **SHIFT** key. When pressed, CTRL appears at the top of the screen, and disappears when the second key is pressed.

⇕ (MENU KEY)

The menu key switches between the three RESERVE modes cyclically, in the order I, II, III and then I again. Each of the three reserve modes allows a different set of function key menu strings to be output, giving a total of 18 possible function key strings.

SML

The **SML** (small) key switches between upper case and lower case letters for the keyboard keys. In lower case mode, SMALL is displayed at the top of the screen.

DEF

The **DEF** (define) key has two functions; i) In RUN mode, pressing the **DEF** key starts program execution from a defined label (see Chapter 8) and ii) In any mode, pressing the **DEF** key followed by one of the ten alphabetic keys on the top line of the keyboard (Q,W,E,R,T,Y,U,I,O,P) or followed by one of the six function keys (!,",#,$,%,&) causes the corresponding pre-assigned keyword to be displayed on screen at the cursor position (see page 88).

RCL

The **RCL** (recall) key displays the function key menu for function keys **F1** to **F6** in RESERVE mode. The display clears when the **RCL** key is pressed a second time. The function key menu displayed will be that which corresponds to the current RESERVE mode, I, II or III.

CLICK
KBII

This key switches the key "click" function on and off when pressed after the **SHIFT** key (**SHIFT** + **CLICK**). Used alone, pressing this key selects keyboard II (KBII) for using the characters on the international character template.

Special Keys

Below the keyboard and to the right of the numeric keypad are a variety of keys some of which have special functions in BASIC, or control the cursor movement.

↓

The ↓ key has three functions: i) In the PROgram mode, pressing this key with no program lines on the screen displays the first line of the program currently in memory. Pressing the key again scrolls the screen to display the next program line, and so on through the program. ii) After an execution error, or when the program is halted with BREAK, STOP or INPUT, if the computer is switched to PRO mode, pressing the ↓ key displays the line last executed on the screen for editing. iii) In the RUN mode, when a TRON (trace mode) command has been input and the program is being executed line by line, or after execution is halted by BREAK or STOP, the ↓ key starts execution of the next line.

↑

The ↑ key has four functions: i) In the PRO mode, pressing this key with no lines on the screen displays the last line of the program currently in memory. Pressing the key again scrolls the screen to display the line preceding the one displayed at the top of the screen. ii) In the RUN mode, when execution of a program is halted by an error or with a 'BREAK IN <line #>' display, the line at which execution halted is displayed while this key is held pressed. iii) After an error, or when the program is halted with BREAK, STOP or INPUT, the ↑ key has the same function as the ↓ key. iv) In the RUN mode, when a TRON (trace mode) command has been input and the program is being executed line by line, the ↑ key displays the full contents of the line currently being executed.

SHIFT + ↓

Inputs the π (pi) symbol when in MODE 1. In MODE 0, has the same effect as ↓.

SHIFT + ↑

Inputs the $\sqrt{\ }$ (square root) symbol when in MODE 1. In MODE 0, has the same effect as ↑.

ENTER

The ENTER key is used to signal the end of an entry when a sequence of characters is entered into the computer. Until the ENTER key is pressed, characters are not sent from the keyboard buffer into the computer proper. The computer will wait for the ENTER key to be pressed before starting execution on the entered command or data.

BS Deletes the character immediately to the left of the cursor position and moves the rest of that logical line to the left one character. If the cursor is on the first character in a physical line, the **BS** key will delete the character or space at that position, and move the rest of the line to the left one character. For a logical line, the **BS** key will move the cursor leftwards deleting characters as it moves up to the beginning of that logical line.

POWER OFF Turns off the computer.

POWER ON Turns on the computer. At power on, the status line on the screen shows the modes that were active just prior to the last power off. After power is on, the **ON** key acts as the **BREAK** key.

BREAK Pressing the **BREAK** key interrupts execution of the current program or command. The screen will display the message 'BREAK IN <line #>'. The **BREAK** key can be enabled and disabled with the BREAK ON/OFF statement.

CL The **CL** key deletes the line in which the cursor is positioned from the screen before the **ENTER** key has been pressed. In the RUN mode, if a program is terminated in an error, the **CL** key clears the error message from the screen.

CA The **CA** key (**SHIFT** + **CL**) clears the screen and resets the computer's internal flags to the "ready to execute" state.

▶ The **▶** key has two functions: i) in the PRO mode with the cursor on a program line, it moves the cursor to the right one position. If the cursor is at the end of the logical line, the key has no effect. ii) in the direct calculation mode, pressing the **▶** key after entering an expression for calculation will recall the expression to the screen. If there is a syntax error in the expression and an error message is displayed, pressing this key will recall the expression with the cursor flashing at the error position.

◀ The **◀** key has two functions: i) In the PRO mode with the cursor on a program line, it moves the cursor to the left one position. If the cursor is at the beginning of the logical line, the key has no effect. ii) In the direct calculation mode, pressing the **◀** key has the same effect as the **▶** key.

DEL The **DEL** (delete) key (**SHIFT** + **◀**) deletes the character at the current cursor position and closes up the space.

INS The **INS** (insert) key (**SHIFT** + **▶**) opens up a new space for inserting a single character immediately to the left of the current cursor position and moves the cursor into the space. The **INS** key only allows insertion of a single character each time it is pressed. To insert a series of characters, the **INS** key must be pressed before entering each character.

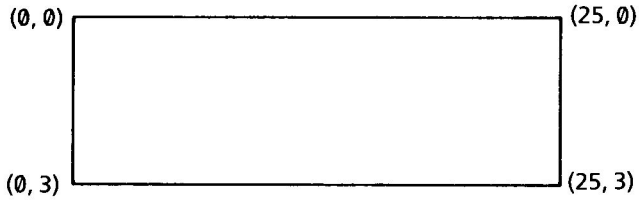
Screen Modes

This section describes the two screen modes for the PC-1600 and the way in which they are used, and explains the coordinate systems used to specify locations of characters and graphics on the two screens.

The two screen modes are referred to as MODE 0 and MODE 1. The use of MODE in connection with screen modes should not be confused with the operating modes (RUN, PRO, RESERVE) of the computer. A full description of how to set the screen mode is given under the MODE command in the Command Dictionary. The default screen mode is MODE 0 (the mode which comes up after resetting the computer with ALL RESET).

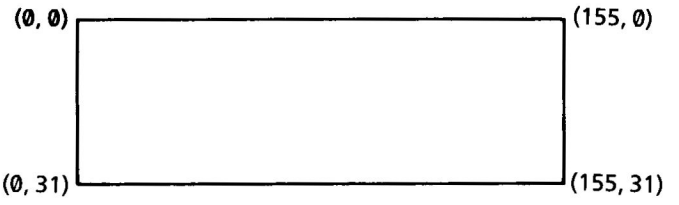
MODE 0 (26 × 4)

In this screen mode, the number of lines available on the screen is set to 4, and the number of columns is set to 26. This gives a maximum of 26 ASCII characters per line. Each character is displayed as a 5 × 7 dot matrix.



Character Coordinates in MODE 0

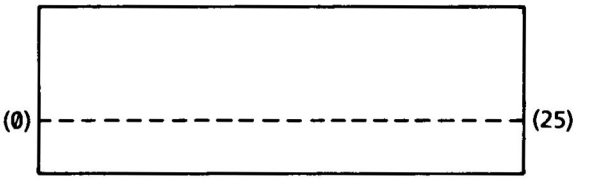
In the graphics mode (see GCURSOR, GPRINT, LINE, POINT, PRESET, and PSET commands), the screen displays bit image data on a 156 × 32 matrix rather than ASCII character codes. For instance, using the PSET statement, a display dot can be turned on or off at any position on the screen by specifying its X coordinate (in the range 0 to 155) and Y coordinate (in the range 0 to 31).



Graphic Screen Coordinates in MODE 0 and MODE 1

MODE 1 (26 × 1)

In this screen mode, the number of lines on the screen is set to 1 and the number of columns is set to 26. This mode is basically to allow compatibility with the Sharp PC-1500 display. The active line is the bottom line on the screen. The other three lines are not active for the PRINT and GPRINT commands, but in PRO mode, or when the INPUT statement is used, the screen scrolls up a line at a time. In this mode, the Pi symbol (ASCII code &5D) and the square root symbol (ASCII code &5B) are displayed as "π" and "$\sqrt{\ }$" as on the PC-1500. The valid character set is also the same as for the PC-1500. In MODE 1, the screen displays bit image data on a 156 × 32 matrix as in MODE 0.



Character Coordinates in MODE 1

The graphics statements in BASIC allow a wide variety of customized displays to be programmed by the user. The POINT and PSET statements allow actual dot-by-dot patterns to be displayed on the screen from a program. The following screen graphics commands are available: GCURSOR, GPRINT, LINE, POINT, PRESET, and PSET.

Edit Mode

A line of BASIC instructions can be up to 80 characters long. However, as the computer's screen can only accommodate 26 characters per line, a BASIC line can occupy more than one line on the screen.

To distinguish between a line in BASIC and a line on the screen the terms "logical line" and "physical line" are used. A logical line is a line of BASIC instructions beginning with a line number and containing up to 80 valid characters. A physical line is a line on the screen containing up to 26 characters. One logical line may therefore extend over several physical lines. The editing functions in BASIC do not distinguish between physical lines.

Program lines can only be edited in PROgram mode. There are several methods which can be used to edit lines of a BASIC program after they have been typed in.

The simplest is just to retype the entire logical line using the same line number. Using the LIST command you can move to a line easily and use the edit functions to position the cursor on the line and then make changes. The computer automatically replaces the old line with the new one when the ENTER key is pressed. If you do not press ENTER your changes will not be stored, and the line will be unchanged in memory.

After execution halts in an error code, or after BREAK, STOP or INPUT, pressing the MODE key will switch to PRO mode. Then pressing the ↓ or ↑ key will display the last line executed with the cursor at the break position ready for editing. Pressing the ENTER key in this state will bring subsequent program lines up on the screen. The screen can be scrolled for lines not displayed using the ↑ and ↓ keys as normal. The state is cancelled by pressing SHIFT + CL.

The CTRL key used together with the following normal BASIC keys allow you to enter and edit lines of a BASIC program easily and to go straight to lines in a long program without having to go line by line through a long listing. The GOTO command can be used with a label to allow editing of only one of a number of programs in memory concurrently. Then the LIST, RENUM and DELETE commands allow manipulation of the selected program lines.

CTRL + ↑ — Pressing this key combination clears the screen and displays the program's first line with the cursor at the beginning of that line.

CTRL + ↓ — Pressing this key combination clears the screen and displays the programs's last line with the cursor at the beginning of that line.

CTRL + ▶ — Pressing this key combination moves the cursor to the position following the last character on the current logical line.

CTRL + ◀ — Pressing this key combination moves the cursor to the beginning of the logical line on which it is currently positioned.

CTRL + **A** Pressing this key combination toggles between INSERT and OVERWRITE when editing a program line. In INSERT mode, characters input from the keyboard are inserted in the line in front of the cursor and characters are moved to the right. In OVERWRITE mode characters input from the keyboard overwrite the characters at the cursor position and the line length is unchanged. The default when turning on the power is OVERWRITE mode. The cursor blinks at a faster rate when INSERT is toggled on.

CTRL + **D** Pressing this key combination deletes all characters in front of the cursor position to the head of the line, including the line number.

CTRL + **E** Pressing this key combination deletes all characters from the current cursor position to the end of that logical line.

CTRL + **F** Pressing this key combination moves the cursor forward to the first character in the next word on the current line.

CTRL + **H** Pressing this key combination deletes the character immediately to the left of the cursor and moves the remainder of the logical line to the left one character position. If the keys are pressed with the cursor on the first character of a logical line, they will delete the character or space at that position and move the rest of the line to the left one character. For a logical line, pressing the keys moves the cursor leftwards along that line, erasing characters as it goes up to the beginning of that logical line, and running over physical lines where necessary. The **BS** (backspace) key has the same effect.

CTRL + **R** Pressing this key combination turns the key repeat function on and off in the same way as the KEYSTAT command.

CTRL + **X** Pressing this key combination deletes all characters on the logical line on which the cursor is positioned. The **CL** key has the same effect.

Reserve Mode

In RESERVE mode, the computer allows access to a special area of memory which stores the character strings allocated to the six function keys located at the top of the keyboard directly under the screen, and labelled ! " # $ % &.

The character string output when one of the keys is pressed may be changed or modified by the user. The strings can also be stored on a device in the same way as a normal program and loaded into the computer at any time to set up the function keys in a certain way for a particular application.

Selecting RESERVE Mode

Press SHIFT + MODE. The status line at the top of the screen will display RESERVE. You can exit RESERVE mode by pressing the MODE key once.

With RESERVE displayed on the status line it is possible to select three separate RESERVE modes: RESERVE mode I, RESERVE mode II, and RESERVE mode III. Mode I, II or III is selected with the ⇕ (Menu Key) and displayed on the status line next to the RESERVE display. The Menu key switches between the three RESERVE modes cyclically in the order I, II, III and then I again. Each of the three RESERVE modes allows a different set of function key strings to be output, giving a total of 18 possible function key settings.

Setting the Function Key Character Strings

1 First select RESERVE mode with SHIFT + MODE. Type in NEW from the keyboard and press ENTER. This will clear all existing strings in RESERVE memory for all three modes. If you want to preserve the current contents of the function keys, or only to modify certain keys, you should omit this step.

2 Select mode I to set up the contents of function keys F1 to F6.

3 Press key F1 and type in the character string you wish to be output to the screen when F1 is pressed during normal operation. All alphabetic, numeric characters and symbols can be used. Then press ENTER.

4 Repeat this to set the character strings for function keys F2 to F6.

5 Select mode II by pressing the Menu key ⇕ once.

6 Repeat six more settings for the six function keys in mode II (steps 3 and 4).

7 Select mode III with the Menu key and repeat the six settings.

You should now have 18 character strings assigned to the function keys in modes I, II and III.

Possible settings for the keys may include BASIC command words and function names, or even actual calculation formulae.

Sample function key settings:

KEY	Mode I	Mode II	Mode III
F1	SIN	FOR	PRINT
F2	COS	TO	LIST
F3	TAN	STEP	AUTO
F4	LOG	NEXT	LOAD
F5	EXP	GOTO	SAVE
F6	INT	RUN	LLIST

The maximum length of the character string allocated to one function key must be within 110 bytes. A BASIC command or a function (e.g., RUN, PRINT, SIN, COS) takes up 2 bytes. A number, letter or symbol takes up 1 byte. So one function key can store a phrase much longer than just a single command. For instance; F1 could be set to hold the following statement in BASIC:

F1 FOR I = 1 TO 100 STEP 5

You can choose your settings from the commands and statements you most frequently use in programming, and speed up program entry time considerably.

Recalling the Function Key Character Strings

The function key settings held in RESERVE memory for modes I, II or III can be recalled in the RUN or PRO modes during direct input or program writing. Select the mode required with the ⇕ key, and press the appropriate function key. The contents will be output to the screen at the current cursor position. For instance, pressing F1 with the mode set to III in the above example will output PRINT to the screen.

The memory storage area used to store function key character strings in mode II is also used to hold the optional message for an ALARM$ command. If the alarm is set with a message, that message will be displayed instead of the function key string, and will overwrite the string in memory.

Function Key Menus

As the contents of function keys are not displayed until the key is pressed, the PC-1600 incorporates a feature which will display a single-line menu on the screen for the function key modes I, II and III. The menu is displayed by pressing the RCL (recall) key. The menu can, for instance, indicate the contents of each key in abbreviated form. The menu is stored in the computer as a 26-character string, the sole function of which is to give the user an indication of what the function keys will output when pressed.

Creating a Menu:

1 First set the computer in RESERVE mode with SHIFT + MODE.

2 Select the mode for which you want to input the menu using the Menu ⇕ key.

3 Type in the menu string within 26 characters, enclosed in quotation marks (") and press ENTER.

The menu will be stored in part of the reserve memory for future recall. Remember, the maximum length of the menu is 26 characters (the width of the screen). In addition, no one menu item should be longer than 4 characters if you want the menu item to locate correctly above the function key it refers to.

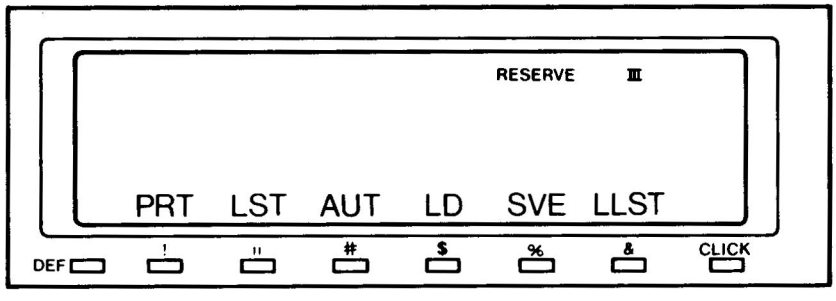
A sample menu for the function key settings in mode III given above could be:

PRT LST AUT LD SVE LLST
← 26 Characters →

Displaying the Menu:

The sample menu can be displayed in RUN or PRO modes by selecting mode III with the Menu key and pressing the RCL key. Press the RCL key a second time and the menu is removed. In the same way, the menus for modes I and II can be displayed when needed to remind the user of the function key settings.

The sample menu for mode III will appear on the PC-1600's screen in the following position in relation to the function keys:



Saving and Loading Function Key Programs:

The strings assigned to function keys can be saved and loaded from cassette tape in the same way as a normal BASIC program. First set the computer in RESERVE mode with SHIFT + MODE. Then use the CSAVE or CLOAD command as if you were saving or loading a BASIC program. Floppy disk and RAM disk cannot be used to store function key settings.

IMPORTANT: *When loading a function key program, it is important to double-check that the computer is set to RESERVE mode; if in PRO mode, the program will be loaded into the normal BASIC program area and will corrupt any BASIC programs already existing there.*

Pre-assigned Keywords

A few of the most frequently used BASIC command keywords have been permanently assigned to single alphabetic keys in the top row of the keyboard (QWERTYUIOP). These keywords may be retrieved, in any mode, by pressing the DEF key followed by the alphabetic key. The keywords available and their corresponding alphabetic keys are:

Key	Keyword
Q	INPUT
W	PRINT
E	USING
R	GOTO
T	GOSUB
Y	RETURN
U	CSAVE
I	CLOAD
O	MERGE
P	LIST

The CSAVE, CLOAD and MERGE keywords can only be used when a cassette recorder is connected via the CE-1600P Printer with Cassette Interface Unit.

These keywords are engraved on the template supplied with your computer. Place it over the keyboard keys for easy reference.

In addition the DEF key followed by a function key will retrieve six more keywords as below:

DEF + ! → RUN ENTER
DEF + " → AUTO
DEF + # → LOAD"
DEF + $ → SAVE"
DEF + % → FILES"
DEF + & → "COM1:"

10 Data Representation

Data inside the computer is handled in units of eight binary digits. One eight-bit unit is called a byte. In the binary number system, it is possible to represent all numbers from 0 to 255 using eight bits. For example, the number 0 is represented as 00000000, the number 17 is represented as 00010001, and the number 255 is represented as 11111111.

Normally, to simplify the writing of these strings of eight bits in tables and charts, they are converted to their equivalent hexadecimal value. The hexadecimal representation is always shorter than the binary representation of the same byte. To distinguish between decimal and hexadecimal, an "&" is placed before a hexadecimal number. So &27 means 27 in the hexadecimal number system.

Types of Data

Text Data and Character Sets

In order to use this eight-bit system to represent alphabetic characters and symbols such as punctuation marks or brackets, a standard code is used, with each character represented by an eight-bit binary number.

There are three main standard codes in use at present, ASCII code (USA), EBCDIC code (Europe) and JIS code (Japan). As far as the main alphabetic and other commonly used characters are concerned, these three standard codes are almost 100% interchangeable. There are some differences in graphic symbols between them, and in some special symbols (the currency symbol for example). The PC-1600 uses a character code set which is similar to the IBM PC.

There are some differences between the character set used by the PC-1500 and the PC-1600. This means that some of the display characters in the PC-1600 change between screen MODE 0 and MODE 1 (see Screen Modes). The differences are in the characters represented by &5B (square root sign), &5D (Pi sign), and some of the bracket types. See the tables in Appendix C.

Character Strings

Strings of characters up to 80 characters long can be handled as single units by the computer. They are handled in a different way to numerical data, and numerical calculations cannot be performed with them, although they can be manipulated to some extent; they can be added to make longer strings and parts of strings can be extracted.

Numeric Data

All numeric data is converted to binary form for storage or calculation in the computer. In addition, the computer will accept numbers input in either normal decimal notation, or in floating point notation. In special cases, input and output can be specified in hexadecimal form (see HEX$ command description).

The computer uses floating point representation for numerical calculations; a number is represented internally in exponential form in 12 digits times a power of 10. But the value is rounded to 10 digits for output. Thus the computer inputs and outputs numbers in the range $\pm$ 9.999999999 $\times$ 10^{-99} to $\pm$ 9.999999999 $\times$ 10^{+99}. This rounding gives an operational error of $\pm$ 1 in the 10th digit (or strictly, $\pm$ 1 in the mantissa).

Constants

String Constants

A string constant is any sequence of alphanumeric characters enclosed in quotation marks. The following examples are all valid string constants:

"SHARP PC-1600"
"ABCDEF"
"The hand-held computer revolution"
"1.256 to 84.4"

The computer will not distinguish between numeric characters and alphabetic characters if they are enclosed in quotes. A pair of quotation marks in succession " " designates a 'null string' of zero length. This has certain special uses in programming (see INKEY$).

The maximum length of a string constant is 30 characters.

Numeric Constants

Numeric constants can be positive or negative numbers. There are four types of numeric constants:

1 Integer constants

These are whole numbers in the range −32768 to +32767. They can be preceded by a + or − sign. If no sign is present, a positive integer is assumed.

e.g. 123, +2, −57, 1024

2 Fixed point constants

These are positive or negative numbers which include a decimal fraction. They can be preceded by a + or − sign. If no sign is present, a positive number is assumed.

e.g. 12.7, +1.345, 0.222232, −67.9888

3 Floating point constants

These are positive or negative numbers represented in exponential form. They consist of an integer or fixed point constant followed by the letter E and an exponent. Either the fixed point part or the exponent part can be positive or negative. If no sign is given, that part is assumed to be positive.

e.g. 2.43E9, +1.99E−3, +50.23E+2

4 Hexadecimal constants

These are constants represented in the hexadecimal number system. To distinguish them from decimal integer constants, they are preceded by the ampersand symbol "&".

e.g. &23, &8000, &2B, &FFFF

Hexadecimal constants can be in the range &0 (zero decimal) to &FFFF (65535 decimal). Negative or fractional numbers are illegal.

Variables

Variables are defined areas in the computer's memory which have been assigned names by the user to hold values during execution of a program. The values held in variables can be set by the user, or written into the variable by the program itself.

There are two types of variables used with SHARP BASIC. They are: numeric variables and string variables. Numeric variables are used to store numbers to be used in calculations or the results of calculations. They are handled by the computer internally in floating point form, but may be output in either fixed or

floating point form depending on the type of calculation performed; BASIC automatically converts from one type to the other when necessary. Numeric variables can be entered into the computer either in fixed point or floating point form.

e.g. A = 12.6043, VA = 1.2345436687E−3

String variables are used to store strings of valid ASCII characters.

In addition to assigning a name for a variable, certain variables must also be assigned specific amounts of memory space according to the size of the data to be stored. This is done with the BASIC statement, DIM. A numeric variable has the value zero until some value is stored in it. A string variable similarly has the value null, or empty string.

Variable Names

Numeric variable names may consist of up to 2 characters. The first character can be any upper-case letter from A to Z. The second character can be any upper-case letter from A to Z or any number from 0 to 9.

The name may contain subsequent characters, but they will be ignored when identifying the variable. Thus TOTAL and TOP will be taken as the same variable name by BASIC. But note that A and AB are different variables.

e.g. A, AB, C1, D9, @(3), RE(6), X2(I)

String variable names follow the rules for numeric variables but in addition must have the "$" symbol as the last character.

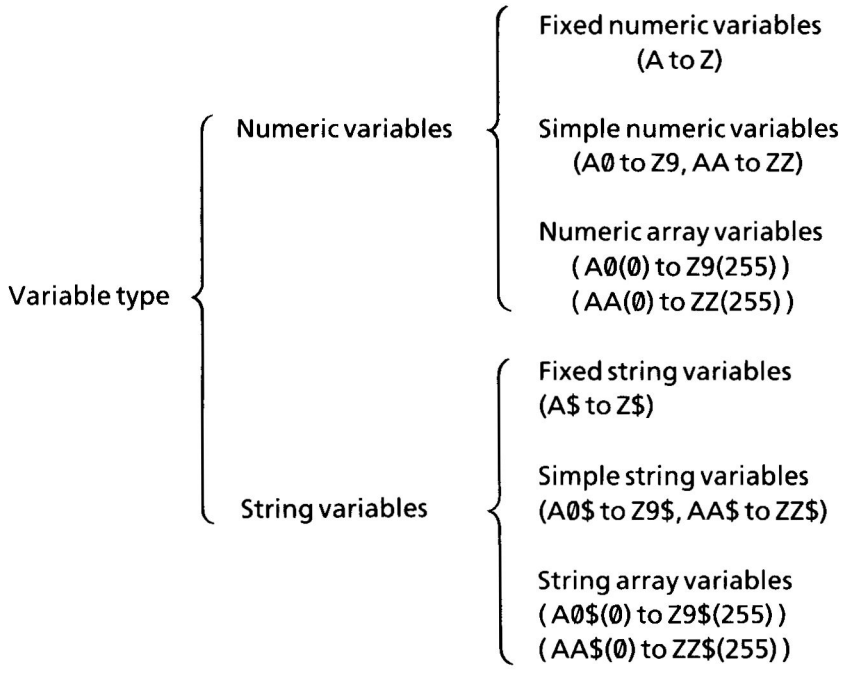
e.g. CC$, B$, D4$, NO$(4), TR$(12)

Variable Types

In Sharp BASIC, variables are divided into three types for allocation of memory. These three types are:

i) fixed variables
ii) simple variables
iii) array variables

Certain letters are reserved for specific variable types. The table shows these three variable types with variable name allocations for Sharp BASIC:



1 Fixed Numeric Variables

The storage locations for both numeric and fixed string variables are situated in a specially allocated area of memory (see Memory Maps-Appendix D).

These locations are always available to store the maximum number of fixed variables that can be named. The fixed variable area can be used as a one-dimensional array by using the @ symbol for the name, and a subscript from 1 to 26. So @(1) is stored in location A and @(26) is stored in location Z.

Fixed variables can be cleared from memory using the CLEAR statement. Simple and array variables are also cleared at the same time. To clear only simple and array variables and to leave fixed variables as they are, use the ERASE statement.

2 Simple Numeric Variables

Simple numeric variables are stored in part of the BASIC program area, and the size of the area allocated changes with the number of variables used.

The more variables used, the less memory left for the program; the larger the program, the less area available to store simple variables. If the program is very large, there may in extreme cases be insufficient space left to store simple variables generated by the program. The solution in this case would be to expand the computer's memory area by inserting a RAM module (refer to Chapter 6).

The first character of simple variable names must be an upper-case letter.

3 Numeric Array Variables

An array (or matrix) is a group of variables referred to by a common name. Each member of the array is referred to by a subscript after the array name which must be an integer.

e.g. A(2) is the second member of the array A(X).
A(15) is the fifteenth member.

An array can be one-dimensional, as in the above example, or two-dimensional, in which case two subscripts are needed to refer to a particular member of that array:

e.g. X(3,4), AB(4,12)

The same array name cannot be used for a 1- and 2-dimensional array at the same time.

A one-dimensional array is a list, and a two-dimensional array is a table. As an array is a list or table of members, the total number of members in any particular array is important to know, in order to determine how much space needs to be reserved to hold all the members. The DIM statement is used whenever arrays are used in a program to define or dimension the maximum number of members that the array is going to have.

Here are all the members of the array A(X,Y):

	column 0	column 1	column 2	column 3	column Y
row 0	A(0,0)	A(0,1)	A(0,2)	A(0,3)	A(0,Y)
row 1	A(1,0)	A(1,1)	A(1,2)	A(1,3)	A(1,Y)
row 2	A(2,0)	A(2,1)	A(2,2)	A(2,3)	A(2,Y)
row 3	A(3,0)	A(3,1)	A(3,2)	A(3,3)	A(3,Y)
.....					
row X	A(X,0)	A(X,1)	A(X,2)	A(X,3)	A(X,Y)

Note that in the computer, the first row is row 0 and the first column is column 0. You can see that an array whose largest member is A(3,3) is a two-dimensional array (a table in fact) with 16 members, each having its own unique array name. So space must be reserved in memory for each of the sixteen possible members. Refer to the DIM statement in the Command Dictionary for a detailed description.

Array names can be assigned in the same way as simple variable names, but must be followed by the subscript(s), which may itself be a variable. This enables a simple looping program to be used to automatically handle all the members in an array in order (see programming example for READ..DATA statement in Command Dictionary).

A(X,Y), KA(J), B1(J,K) are all array variables which can represent any member in the array. A(1,3), KA(6), B1(5,7) are all particular members of the above arrays.

Numbers stored as members of an numeric array can be either in fixed point or floating point form, as with simple numeric variables.

The maximum number of members in an array is 65535. For a two dimensional array this is 255 columns × 255 rows.

Memory space reserved for the members of an array can be cleared by the ERASE statement, which will leave all fixed variables as they are. The maximum size for any array is 64K bytes of memory.

4 Fixed String Variables

Fixed string variable names obey the rules for fixed numeric variables, but must have the $ symbol as the second character.

e.g. A$, C$, P$, @$(2)

A fixed string variable can be up to 16 characters long.

5 Simple String Variables

Simple string variable names obey the rules for simple numeric variable names, but they must have the $ symbol as their last character.

e.g. AB$, C7$, Z9$, XX$

Simple string variables can be up to 16 characters long.

6 String Array Variables

String array variable names obey the rules for numeric arrays but they must have the $ symbol as the last character.

e.g. A1$(3), KS$(5,5), C9$(J), JH$(P,Q)

In addition, when declaring the number of members in string arrays with the DIM statement, the maximum length of each member of the array must be declared to

reserve sufficient memory space. If the length is not declared, sufficient memory is reserved for 16 characters in each member. For an array with many members, this can sometimes lead to a lot of memory being reserved and never used. This would be the case for example if the members of a string array were never more than 3 characters long. If there were 25 members in the array (e.g.AB(4,4)) and each member had a maximum length of 3 characters, not declaring the string length would result in unused memory being reserved to hold 16 − 3 = 13 characters for each member × 25 members = 325 unused character spaces!!

The maximum number of members in a string array is 65535. For a two dimensional array this is 255 columns × 255 rows.

Memory space reserved for the members of an array can be cleared by the ERASE statement.

Expressions and Operators

In a program, any combination of variables, numeric constants and string constants is called an expression. An expression may also contain variables or constants alone.

Operators are the symbols which indicate mathematical or logical operations to be carried out on expressions or parts of expressions. There are four basic categories of operations which can be carried out in BASIC. They are: (1) arithmetic operations, (2) relational operations, (3) logical operations, and (4) functional operations.

Arithmetic Operations

Operations are carried out in a fixed order of precedence which is determined by BASIC itself. In a single arithmetic expression, the order of precedence is as follows:

Operator	Operation performed	Example
^	Exponentiation	X^3
−	Negation (changing sign)	−X
*, /	Multiplication, division	X*2, X/Y
MOD	Modulus arithmetic	A MOD B
\	Integer division	A\B
+, −	Addition, subtraction	X + A, Y − B

The above arithmetic operations take precedence over relational or logical operations.

The order in which arithmetic expressions are evaluated can be changed by enclosing parts of the expression in parentheses ( ) as with the normal rules of algebra. In this case, those parts of the expressions enclosed in the parentheses will be evaluated before the others. Normal algebraic expressions are given below with the equivalent BASIC expression:

Normal algebraic form	BASIC expression
x^2	x ^ 2
$(x + 3)^4$	(x + 3) ^ 4
$2(x + y)$	2 * (x + y)
b^{x^2}	b ^ x ^ 2

NOTE:
- Consecutive exponential operations in the same expression are performed from right to left:

$$\text{e.g. } 3 \text{ ^ } 4 \text{ ^ } 2 \rightarrow 3^{4^2}$$

- When powers and negation appear in the same expression, evaluation is from right to left:

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

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

Relational Operations

Relational operators allow two values to be compared. The result of the comparison is given as either "true" (1) or "false" (0). This result can be used in a program to make a decision on the subsequent program flow using IF..THEN..ELSE and IF..GOTO type statements.

The following relational operators can be used:

Operator	Operation performed	Example
=	Equality	A = B
<>	Inequality	X <> Y
<	Less than	A < B
>	Greater than	B > A
<=	Less than or equal to	Y <= 3
>=	Greater than or equal to	X >= 1

NOTE:
- The "=" sign is used in both comparison expressions and in simple assignment expressions with the LET statement. However, the meaning is not the same in the two cases.

- Comparisons can be made between two numeric expressions or variables (i.e., X and Y) or between two string expressions or variables (i.e., A$ and B$). But numeric expressions cannot be compared with string expressions.

- Two character strings are compared character by character from left to right on the basis of their character code values (see Appendix C). If both strings contain the same characters in the same order and the strings are of equal length, they are equal.

  For strings of equal length but with a mismatch found between a pair of characters, the character code of the first pair of mismatching characters is compared, and the string containing the mismatching character which has the largest code value is regarded as greater. If a short string is compared with a long string and all characters in the short string match those so far tested in the long string, the longer string is regarded as greater.

  A space in a character string is compared as a character on the basis of its character code value. The shortest string possible is the "null" string, with character code value 0.

- Arithmetic operators take priority over comparisons in order of evaluation.

Logical Operations

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..ELSE and IF..GOTO statements.

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 listed here:

AND

X	Y	X AND Y
1	1	1
1	0	0
0	1	0
0	0	0

OR

X	Y	X OR Y
1	1	1
1	0	1
0	1	1
0	0	0

NOT

X	NOT X
1	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:

41 AND 27 →
equals
9

41 = 101001
27 = 011011 AND
← 001001

41 OR 27 →
equals
59

41 = 101001
27 = 011011 OR
← 111011

NOT 3 →
equals
−4 (two's complement form)

3 = 0000000000000011 NOT
← 1111111111111100

NOT X can generally be calculated by the equation NOT X = −(X + 1).

Functional Operations

Functions are operations which calculate a specific value from a single operand by built-in BASIC routines. The functions are generally described together with the BASIC commands in the Command Dictionary. The following functions are built into Sharp BASIC:

ABS	Absolute value	LN	Natural logarithm
ACS	Arc cosine	LOG	Common logarithm
ASN	Arc sine	PI	Value of π
ATN	Arc tangent	RND	Random number
COS	Cosine	SGN	Sign
DEG	Decimal/degree	SIN	Sine
DMS	Degree/decimal	SQR	Square root
EXP	Exponent	TAN	Tangent
INT	Integer		

Arguments can be specified or returned in degrees, radians, or as gradient values by selecting the mode with the DEGREE, RADIAN and GRAD commands.

In calculations involving functions and arithmetic operations both in Basic programs and in calculator mode, the priority of execution is as follows:

Exponentiation has priority over functions. Thus in some expressions, parentheses must be used to ensure correct evaluation.

Example:

Normal algebraic form	PC-1600 expression
$\sin^2 30°$	(SIN 30)^2
$(\sin 30°)^2$	SIN 30^2
$\cos^4(A+B)$	(COS(A+B))^4

In addition to the mathematical and trigonometric functions, there are a range of string functions which can be used to perform a variety of operations on character strings. Refer to the command descriptions for details.

11 Files

A file is a set of data records which can be stored on an external device (such as a disk drive). The data records can be read into the memory work area for processing, and written back to the file as needed under a single identifier: the file name.

A file can contain text data in the form of character codes representing the characters in the lines of a BASIC program, or binary data. Files can be stored in RAM disk, cassette tape or the floppy disk drives; and input and output between different devices. Files can also be sent through the serial I/O ports to and from other computers.

All the files on the PC-1600 are sequential files, which means that data is written and read from the file sequentially from the first item through to the last item. Random access to data in the middle of a file is not possible on the PC-1600.

File Descriptors

Files are identified by a "descriptor". The file descriptor specifies a device name (d:), a file name, and an extension (.ext) in the following format:

d:filename.ext

Device Name

The device name specifies on which device the file is to be stored for a new file, or on which device the file exists for an existing file. The following device names can be used. Each device name must be followed by a colon:

Name	Device
S1:	Memory module in slot 1
S2:	Memory module in slot 2
COM1:	RS-232C serial port
COM2:	Optical serial port
COM:	Currently accessed serial port
CAS:	Cassette tape recorder connected to CE-1600P Plotter/Printer Unit
X:, Y:	2.5-inch optional floppy disk drive

File Name

Every file must have a name for it to be saved to a device (the only exception is for files stored on cassette tape in PC-1500 compatible mode with the CSAVE command). The file name may be any name up to 8 characters long and including the following characters:

A to Z, 0 to 9, #, $, %, &, ', ( ), − , ^, { }, @

A file can only be assigned one name.

Extension

A file extension is an additional way of identifying the type of file (eg, BASIC program file, machine language program file, text file).

An extension consists of up to three characters added to the end of the file name and separated by a period. The user may specify any valid extension to a file when saving it. For instance, all program files relating to financial calculations could have the extension .FIN . Or the extension .BIN could be used for all machine language binary files. It is an easy way to identify file contents when viewing the directory of a disk.

When a BASIC program file is saved to a device with the SAVE command, the PC-1600 automatically assigns the three-letter extension .BAS to the file name to identify it as a BASIC program. When the file is re-loaded into memory with the LOAD command, the .BAS extension need not be specified; the computer assumes that the extension is .BAS if none is given.

When the FILES or LFILES commands are used to list the directory of the files on a device, BASIC program files will appear on the directory listing with the .BAS extension attached unless some other extension has been specified by the user when the file was saved. The .BAS extension must always be specified when using the COPY command.

File Storage & Retrieval

Files can be stored on cassette tape, floppy disk, and RAM disk.

Cassette Tape Files

Files are stored on cassette tape sequentially; that is the files are held on the tape in the order in which they were saved to the tape. This means that to get to the

fifth file stored on the tape, it is necessary to wind on past the first four files from the beginning of the tape.

Each file on the tape has a label at the front bearing the file name. If you use the LOAD command to load a file from tape, the computer controls the tape recorder to wind through the tape until it reaches the label bearing the file name which was specified in the LOAD command, and loads the file. This is a relatively slow way of finding a file compared with floppy disk or RAM disk.

The following commands are available to create, access or update files on cassette tape:

BLOAD, BSAVE, CHAIN, CLOAD, CLOAD?, CLOAD M, CLOSE, CSAVE, CSAVE M, INPUT#, LOAD, MAXFILES, MERGE, OPEN, PRINT#, RMT ON/OFF, SAVE

The CLOAD, CLOAD?, CLOAD M, CSAVE and CSAVE M commands are for downward compatibility with the PC-1500 Pocket Computer, and have no special function for the PC-1600.

The number of files that can be stored on a single cassette tape is determined only by the length of the tape and the size of the files themselves.

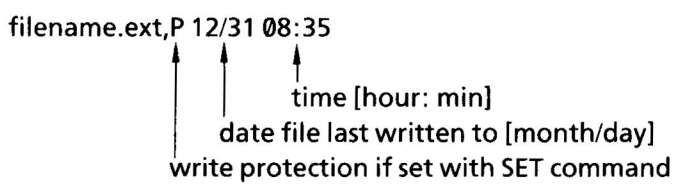
Disk Files

Files are stored on floppy disk or RAM disk in small sectors which can be positioned anywhere on the disk (or in the RAM). The computer is able to access any point on the disk (or in the RAM) directly and immediately. This means that any file stored on a floppy disk or RAM disk can be loaded into the computer's memory virtually instantaneously.

Directories

In order to know where the file is on the disk, the computer keeps a directory of all file names and the size of each file, along with other information needed for access.

The directory can be displayed on the screen for either a floppy disk or a RAM module by using the FILES command or printed out on the printer by using the LFILES command. (This function is not available for the cassette tape, which does not keep a directory of files stored). The directory displays information on the file or files specified in the following format:



The following commands are available to create, access or update files on floppy disk or RAM disk:

BLOAD, BSAVE, CLOSE, COPY, FILES, INIT, INPUT#, KILL, LFILES, LOAD, MAXFILES, NAME, OPEN, PRINT#, SAVE, SET

Number of Files per Device

The number of files that can be stored on a floppy disk or RAM disk is determined by the size of the directory block; that is the number of file names that the directory can hold. The maximum number of file names that the floppy disk directory can hold is 48. The maximum number of files that the CE-1600M optional RAM disk module will hold is 48.

IMPORTANT: *CE-159 Program modules can only be used to store single programs but they are not in the form of a file with a file name. They are not loaded with the LOAD command. After the module has been inserted into one of the slots at the rear of the computer, that slot is made active by using the TITLE command. The program in that module is then executed by entering RUN as if it had been loaded into the computer's internal memory.*

Ambiguous File Names – "Wildcards"

A group of files can be specified by using an ambiguous file name which is satisfied by all the file names in the group. This is done using "wildcard" characters to substitute for certain characters in the ambiguous file name. The wildcard characters are the question mark ?, which can replace any number of single characters in a file name, and the asterisk ✱, which can replace any complete file name and/or any complete extension.

Example:

Ambiguous file name	File names satisfying the ambiguous name
?ET	SET, MET, RET, @ET, 4ET
?PT?	RPT1, RPT2, SPTM, (PT), –PTS
?????	Any five-character file name
???1.B??	REP1.BAS, DAT1.BIN, LET1.BKK
✱	Any file name up to 8 characters
✱ . ✱	Any file name with any extension

Specifying an ambiguous file name with "wildcard" characters when using the FILES or LFILES commands, will result in all files which satisfy that ambiguous name being listed on the screen or printer. The ✱ . ✱ form will list all files in that

device's directory to the screen or printer. The description of the FILES command in the Command Dictionary also explains the use of the "wildcard" characters * and ? in specifying file names. The wildcard ? may also be used in specifying the alarm time with ALARM$.

File Protection

Individual files stored on floppy disk or RAM disk may be protected using the SET command in BASIC. When protected, a file cannot be written to or destroyed. See the SET command for full details of types of protection.

In addition, files may be protected by setting the hardware protection for the whole device. For the floppy disk, this consists of a write-protect switch for each side of the disk which protects all the contents of the side from erasure or editing. For the RAM disk, the contents may be protected by setting the write-protect switch on the front of the module to ON. This must be done with the computer switched OFF. The contents are then protected from erasure or editing.

The contents of the computer's internal memory can be protected from unauthorized access by setting a password with the PASS command. The contents of memory cannot be erased, edited or listed out unless the user enters the correct password.

Creating a File

There are four steps in creating a file which must be followed in the correct sequence:

1 Execute a MAXFILES statement to assign the number of files to be opened.

2 Execute an OPEN statement to assign a file name and corresponding file number and open the file in the OUTPUT mode.

3 Write data to the file using the PRINT# statement.

4 Execute a CLOSE statement to close the file. This must be done before the file can be reopened for input in the INPUT mode.

Example:

The following short program creates a file on floppy disk of name and address data which can form the basis of a simple computerized address book.

```
 5 MAXFILES = 1                          Assign one file to be opened.
10 OPEN "X:ADDRESS" FOR OUTPUT AS #1     Open the file for output.
20 INPUT "ENTER NAME?";N$                Read in data entered from
30 IF N$ = "END"THEN 100                      the keyboard on name,
40 INPUT "ENTER CITY?";C$                     city, and tel number.
50 INPUT "ENTER TEL. NUMBER?";T$
60 PRINT #1,N$;",";C$;",";T$            Write the data to file #1.
70 PRINT                                 Cursor to next line.
80 GOTO 20                               Enter next data item.
100 CLOSE #1                             Close the file.
110 END
```

Enter the program into the PC-1600 and type RUN. The following prompt will appear on the screen:

ENTER NAME?

Type in the name of a person whose location and telephone number you want to store, and press **ENTER**. The following prompt will appear:

ENTER CITY?

Type in the city and press **ENTER**. The next prompt will appear:

ENTER TEL. NUMBER?

Type in the telephone number and press **ENTER**. The first prompt will appear again for you to enter the three data items on the second person. Use the following data to create a sample ADDRESS file:

M.JONES	NEW YORK	212-758-0354
T.SMITH	LONDON	01-634-4431
M.BERRY	NEW YORK	212-432-0012
N.ITO	TOKYO	03-927-1345
S.SHARP	NEW YORK	212-124-5364
P.PETERS	LONDON	01-433-0056

When you have entered all the data, enter END in response to the ENTER NAME? prompt. This will end the program and close the data file.

If you look at the directory of the floppy disk with the FILES command using the ambiguous file name * . * , you will see that the file "ADDRESS" has been added to the directory by this program. This file contains all the data entered to the program in the order in which it was entered.

IMPORTANT: *If you run the program again, the data file ADDRESS will be reopened for output in line 10. The process of reopening the file for output will erase all the data that was stored there previously and leave an empty file called ADDRESS ready to be written to!*

The only way to add new data to existing data in a file is by opening it in the APPEND mode. A separate program must be created to do this with the file opened by the following statement:

```
OPEN "X:ADDRESS" FOR APPEND AS #1
```

With a sequential file, data can only be added to the end of the file. This means that items cannot be inserted into alphabetical position using APPEND. However, this can be done by writing another program which reads all the data from the file into the computer, adds the new item in the correct alphabetical position, and then writes all the data items back to the file in the new sequence. APPEND can only be used for files on floppy disk or in a RAM disk module.

Accessing a File

There are four steps in accessing a file which must be followed in the correct sequence:

1 Execute a MAXFILES statement to assign the number of files to be opened.

2 Execute an OPEN statement to open the file "X:ADDRESS" in the INPUT mode.

3 Read data from the file into variables using the INPUT# statement.

4 Execute a CLOSE statement to close the file. This must be done before the file can be reopened in the OUTPUT or APPEND mode.

Each time the file is opened, data will be read from the beginning of the file. Each time an INPUT# statement is executed, the next data item in the file will be read into the program. When the end of the file is reached, if there is no check in the program to stop reading from the file, error code 165 will be displayed.

Example:

The following program will read the data stored in the ADDRESS file created with the previous program and display on the screen all people living in NEW YORK:

```
 5 MAXFILES = 1                          Assign number of files to be opened.
10 OPEN "X:ADDRESS" FOR INPUT AS #1      Open the file for input.
20 PRINT "NEW YORK"                      Print city and headings for printout.
```

```
30 PRINT:PRINT "NAME","TEL. NUMBER"
40 IF EOF(1) THEN 100          Test for end of file.
50 INPUT #1,N$,C$,T$           Read item from file.
60 IF C$ = "NEW YORK" THEN 80  Compare city.
70 GOTO 40                     Back to read next item.
80 PRINT N$,T$                 Print name and number.
90 GOTO 40                     Back to read next item.
100 CLOSE #1                   Close file.
110 END
```

Enter the program into the PC-1600 and type RUN. The screen will show the following:

```
NEW YORK

NAME       TEL. NUMBER
M.JONES    212-758-0354
M.BERRY    212-432-0012
S.SHARP    212-124-5364
```

The comparison in line 60 for the "NEW YORK" string requires that the spelling and spacing of the string be exactly the same, and it must be in upper case.

This program can quite easily be modified to allow the user to choose which city should be searched, by adding and replacing the following lines:

```
15 INPUT "WHICH CITY";S$
20 PRINT S$
60 IF C$ = S$ THEN 80
```

Updating a File

As described previously, with a sequential file, data can only be added or appended to the end of the file if the file has been opened in the APPEND mode. To insert items into the existing list, a more sophisticated program is required to read in all the records, update them in the PC-1600's memory, and then write them to the file in their new order.

Changing line 10 in the first program to the following allows additional name data to be added to the end of the existing ADDRESS file:

```
10 OPEN "X:ADDRESS" FOR APPEND AS #1
```

The DSKF, EOF, LOC, LOF, and MAXFILES commands are available for more advanced file handling programs to check file sizes when files are being created to ensure that device full errors and other file errors do not occur.

12 Access to Serial Ports

The PC-1600's two serial communications ports make it possible to connect the PC-1600 to other computers and devices which support RS-232C interface access, or have a compatible fiber optic port. Both the RS-232C port and the optical port are handled by BASIC as sequential input/output devices, identified by the device names COM1: and COM2:. COM: specifies the currently accessed port.

Specifying the Port

The two ports are opened for data communication by the SETDEV command. After specifying COM1: or COM2: with SETDEV, that port is used for all input and output for the BASIC communication commands until another SETDEV command is executed. The following commands are available to control the two serial ports:

COMn ON/OFF/STOP, PCONSOLE, INIT, INSTAT, ON COMn GOSUB, ON PHONE GOSUB, OUTSTAT, PHONE ON/OFF/STOP, RCVSTAT, SETCOM, SETDEV, SNDBRK, SNDSTAT, PZONE

In addition, certain printer commands can be used to send output to one of the serial ports. When power is turned on, output from the LLIST, LFILES and LPRINT commands is normally directed to the printer. The SETDEV command has options which direct output from these commands to the selected port. SETDEV is also used to direct output back to the printer, effectively closing both serial ports. The current settings for the SETDEV command can be displayed using the DEV$ function. The SAVE and LOAD commands are available to send or receive complete files via the serial ports.

Protocol Options

Before sending or receiving data through a serial port, the communications protocol which governs the states of the control signals must be set according to the device with which the PC-1600 is communicating. The handshake protocol is set with the SNDSTAT command for sending data, and the RCVSTAT command for receiving data. For sending, the states of all the control signals can be set individually with the OUTSTAT command.

Setting requires detailed knowledge of control signals used for data transmission; however, the command leaves the PC-1600 set up for simple data transmission if OUTSTAT parameters are not specified. The current receive protocol settings can be displayed using the INSTAT command.

Communications Parameters

When communicating with another computer or device, the speed and mode of data transmission must agree for both devices. The settable parameters are:

1 transmission speed (baud rate) set in the range 50 to 38400 baud
2 word length set in the range 5 to 8 bits
3 parity set as even, odd or no parity
4 number of stop bits set as 1 or 2
5 X-ON/OFF protocol set on or off
6 shift in/out protocol set on or off

These parameters are set with the SETCOM command. The current communication parameter settings can be displayed with the COM$ function.

When using the SAVE, LOAD, BSAVE, BLOAD commands, or when using COPY to transfer files between a RAM disk and a communications port, the maximum settable baud rate is 9600 baud for the RS-232C port (COM1:) and 38,400 baud for the optical port (COM2:). When using the INPUT, INPUT#, PRINT#, LLIST and LPRINT commands, the baud rate for both ports is limited to a maximum of 4,800 baud.

Receive Buffer

Data which is received through either of the two serial ports is first stored in the receive buffer. The size of this buffer can be set by the user with the INIT command to best suit the type of data being received.

Output to a Serial Port

Commands and functions which can be used for output to either of the serial ports are:

CHR$, LFILES, LLIST, LPRINT/LPRINT USING, OPEN, PRINT#/PRINT# USING, PZONE, SAVE

Sending Programs and Data

Data generated by a program or program listings can be sent directly to a serial port by the following procedure:

1 Open the port using the SETDEV statement with the PO option to direct output from the LPRINT, LLIST and LFILES commands to that port.

2 Use the PZONE command to set the format for the output of LPRINT commands.

3 Use the PCONSOLE command to set the appropriate line length and end of line code.

This allows a program file to be sent to another computer. First load the file into memory with the LOAD command. Then use the SETDEV command to direct output to the RS-232C serial port. Finally type LLIST to list out the program lines one by one to the serial port.

The port can also be opened as a file and data items written sequentially to the port with the PRINT# statement by the following procedure:

1 Assign the number of files to be opened with the MAXFILES command.

2 Open the port for output and assign it a number with the OPEN command.

3 Use the PRINT# or PRINT# USING commands in a program to write data items to the port as if it were a normal file.

4 Close the port with a CLOSE statement as with a normal file.

Sending Program Files

An existing file on floppy disk, RAM disk or tape can be sent to a serial port directly using the SAVE command to save the file to the port in the same way as for floppy disk or RAM disk.

Sending Control Codes

The CHR$ function can be used to send special character codes to the port in the same way as they are sent to a printer. For instance, the ASCII control code 4 is used by many computers to signify end of text (EOT). This code can be sent to the serial port by the following line of BASIC, after the port has been opened for output with SETDEV.

```
10 LPRINT CHR$(4)
```

Input from a Serial Port

Commands and functions which can be used for input from either of the serial ports are:

INIT, INPUT#, LOAD, OPEN, PCONSOLE, RXD$, SNDBRK

Receiving Data

Data received at one of the serial ports can be read into a program as data by the following procedure:

1 Assign the number of files to be opened with the MAXFILES command.

2 Set the size of the receive buffer as needed with the INIT command.

3 Open the port for INPUT as a file with the OPEN command.

4 Read in data from the port using an INPUT# statement as with a normal file.

5 Close the port with the CLOSE statement.

Receiving Files

A file can be received through a serial port and stored in memory using the LOAD command in the same way as loading a file from floppy disk or RAM disk.

13 Debugging

It is virtually impossible to write a perfect program the first time. There will be typing errors on entering the program and possibly errors in syntax (grammatical errors) in the way that you have specified your BASIC statements. Even careful checking after entry cannot catch all errors, especially errors in the logical design of a program. Debugging refers to the process of eliminating the various types of errors systematically until the program runs as designed.

Syntax Errors

When you execute your program for the first time, syntax and most other errors are detected line by line as the program is executed. Execution will stop at the first error which occurs in the program and the appropriate error code and the number of the line containing the error will be displayed on the screen.

You should refer to the list of error codes and their meanings in Appendix F. If the error is a syntax error (error code 1), type in the command LIST<line number> to list out the line containing the error. Or alternatively, press the CL key to clear the error code, then the MODE key to change to PRO mode, and then the ↓ key. The line will be displayed with the cursor at the error. Then use the editing keys to correct the error and type RUN again to re-execute the program. The computer will stop execution on the next error which is found. Repeat this procedure each time execution is halted by an error code 1.

Other types of errors must be dealt with according to type. Some may be errors in your program design, others more easily solved; for instance, specifying a FOR..TO statement and forgetting to put in the matching NEXT statement. Practice will teach you the best systematic approach to debugging your programs beyond the syntax stage.

Errors in syntax and typing errors can be corrected using the editing functions described in the Edit Mode section in Chapter 9. You will find that after practice, lines can be corrected with the minimum of retyping using these keys.

Trace Mode

The TRON (trace mode on) and TROFF (trace mode off) commands can be used to follow execution of a program line by line.

When the trace mode is turned on by execution of the TRON command, the computer executes one line and halts execution for 0.5 seconds with the line

number displayed on the right of the screen before continuing to execute the next line. This mode is useful for tracing the flow of your program; making sure that the program is jumping to the correct places at the right time, and so on.

The trace mode can also be set to execute a single line and wait for the Down Arrow key ↓ before continuing. This gives more time to observe the flow of program execution. The trace mode continues until a TROFF command is executed. Refer to the Command Dictionary description for more details.

STOP commands can be strategically placed in your program at the debugging stage to get the computer to stop at a certain point, and to print out interim results as part of a check. Execution can then be resumed again at the same point by using the CONT command.

Error Processing Routines

It is possible to prevent error messages stopping program execution if they are due to foreseeable situations. For instance, if a user enters the wrong type of data in response to an INPUT statement in your program. This is achieved by making use of the ON ERROR GOTO command together with the ERL and ERN functions.

ON ERROR GOTO is placed at the beginning of the program and specifies where the first line of the error processing routine is located. After this statement is executed, any error occurring in the program will cause execution to jump to the first line of the error processing routine, which is designed either to correct the error or allow the user of the program to input the correct data; to have a second try, without allowing the program to be stopped by an error code.

The ERL and ERN functions are used within the routine to determine what type of error has occurred and on what line in the program. The error processing routine can resume program execution at the relevent point with the RESUME statement, or terminates execution with a suitable message to the user.

Refer to the description of the ON ERROR GOTO command for an example of a simple error processing routine.

14 BASIC Command Dictionary

The Command Dictionary contains all the valid BASIC commands and functions for the PC-1600. Some commands are included for complete compatibility with the PC-1500 model in MODE 1, and are not so useful in the PC-1600 mode.

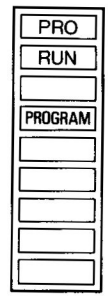
The following symbols are used on the command pages to identify the modes in which the command can be used, and the peripheral devices necessary.

PRO	Can be entered directly in PRO mode.
RUN	Can be entered directly in RUN mode.
RESERVE	Used for function key programming in RESERVE mode.
PROGRAM	Can be entered as a program line.
	Floppy disk command. The CE-1600F Pocket Disk Drive must be connected.
	Cassette tape command. A cassette tape recorder must be connected via the CE-1600P cassette interface.
	Printer command. The PC-1600 must be installed in the CE-1600P Printer and Cassette Interface Unit, (or in the CE-150).
	Communications command. For use with one of the two serial ports. A suitable serial device must be connected to the specified port.
RAM	RAM disk command. The CE-1600M or CE-161 Program module must be installed and initialized in Slot 1 or Slot 2.

The following symbols are used in the command format descriptions:

[ ]	The parameter included in the square brackets is optional. The brackets themselves are not part of the command entry.
< >	Used to indicate an actual parameter value to be entered. The brackets themselves are not part of the command entry.
( )	Used to enclose parameter values in certain commands, and should be entered as part of the command.

ABS



FORMAT: 1. ABS(X)

Abbreviation: AB.
See Also:

PURPOSE: Returns the absolute value of the number X.

REMARKS: The number X may be any numeric expression.

EXAMPLE:

```
5:WAIT 70
10:PRINT "        NUMBER       ABSOLUTE"
20:FOR I=-2TO 2
30:PRINT I,ABS (I)
40:NEXT I
50:END

>RUN
        NUMBER        ABSOLUTE
            -2               2
            -1               1
             0               0
             1               1
             2               2
>
```

PRO
RUN
PROGRAM

ACS

FORMAT: 1. ACS(X)

Abbreviation: AC.
See Also: ASN, ATN, COS

PURPOSE: Returns the arc cosine of X.

REMARKS: This function returns the inverse cosine of the expression X in degrees, radians, or as a gradient value depending on which mode the computer is set to with the DEGREE, RADIAN, or GRAD command. The value of X is limited to $-1 \leqq X \leqq 1$.

In DEG mode, ACS(X) is returned in the range 0° to 180°. In RAD mode, ACS(X) is returned in the range 0 to π radians. In GRAD mode, ACS(X) is returned in the range 0 to 200.

EXAMPLE:

```
10:DEGREE
20:PRINT "ARC COS OF 0.5 IS ";ACS (0.5)
30:PRINT "ARC COS OF 0 IS ";ACS (0)
40:END

>RUN
ARC COS OF 0.5 IS  60
ARC COS OF 0 IS  90
>
```

ADIN ON/OFF/STOP

FORMAT: 1. ADIN ON
2. ADIN OFF
3. ADIN STOP

Abbreviation: AD.
See Also: AIN, ON ADIN GOSUB

PURPOSE: Enables or disables interrupts received through the analog input jack.

REMARKS: Interrupts are received through the analog input jack in the form of a certain voltage level being reached. See the description of the AIN statement for details on the applicable voltage levels.

ADIN ON enables interrupts to be received through the analog jack. The ON ADIN GOSUB statement can then be used to branch execution to the relevant interrupt-processing routine. ADIN OFF disables interrupts from the analog input jack. ADIN STOP disables interrupts from the analog input jack but registers the most recent of any interrupt requests which are received. If ADIN ON is executed subsequently, that interrupt will be processed immediately. The default value is ADIN STOP.

EXAMPLE: See ON ADIN GOSUB.

PRO
RUN
PROGRAM

AIN

FORMAT: 1. AIN

Abbreviation: AI.
See Also: ADIN ON/OFF/STOP, ON ADIN GOSUB

PURPOSE: Returns a value corresponding to the analog voltage level input.

REMARKS: AIN is a system variable which holds the result of converting the current analog voltage input level to a digital level in the range 0 to 255. Refer to Chapter 6 for details on the allowable voltage input levels for the analog input.

EXAMPLE:

```
>PRINT AIN
43
>
```

Prints out the current analog voltage expressed as a digital value.

ALARM$

PRO
RUN
PROGRAM

FORMAT: 1. ALARM$ = "MM/DD/HH/mm[;<message string>]"
2. ALARM$ = " "
3. ALARM$

Abbreviation: AL.
See Also: TIME$, DATE$, POWER

PURPOSE: Sets the alarm time and an optional message string.

REMARKS: ALARM$ = "MM/DD/HH/mm" sets the alarm time, where:

MM is the month	(01 to 12)	Separated by a slash (/).
DD is the day	(01 to 31)	
HH is the hour	(00 to 23)	
mm is the minute	(00 to 59)	

When the specified alarm time is reached, the computer's internal alarm will sound with repeated beeps for 1 second duration. Wildcard characters "?" may be used in the specification of the month and day to enable monthly or daily alarm calls to be set. For example, "??/??/13/30" will set the alarm daily at 13:30.

The optional <message string> may be up to 26 characters long. The message will be displayed on the screen when the alarm time is reached. ALARM$ = " " clears and resets the alarm time and optional message string. ALARM$ used as a variable returns the currently set alarm time.

NOTE:
The message string specified will overwrite the contents of the function key character string for mode II. Refer to Chapter 9, Reserve Mode for details.

EXAMPLE:

```
>ALARM$="12/25/08/00:HAPPY CHRISTMAS!"
```

This will print out the message "HAPPY CHRISTMAS!" at 8am on Christmas day.

AREAD

FORMAT: 1. <label>[:]AREAD<variable name>

Abbreviation: A.
See Also: RUN

PURPOSE: Allows an item displayed on the screen to be read into a program variable.

REMARKS: AREAD can only be used immediately following the label on the first line of a program and is only effective if the program execution was started by pressing the DEF key followed by the label. In all other cases, AREAD will be ignored.

When effective, AREAD will take numerics or characters displayed on the screen at the time it is executed and assign them to the variable specified in <variable name>. The variable name must correspond to the data type: a numeric variable for numerics and a string variable for characters. If the display contains numerics, up to 10 digits and 2 exponents can be read into the variable. If the display contains characters, the number of characters which can be read in is equal to the length of the string variable as declared in the DIM statement. The default is 16 characters. If only the BASIC's prompt sign (>) is displayed when AREAD is executed, the contents of <variable name> will be cleared to 0.

EXAMPLE:

```
10:"A" :AREAD N
:
:
:
```

[10] When the program is run, the numeric (if any) currently displayed on the screen will be read into the program variable "N".

ARUN

FORMAT: 1. ARUN

Abbreviation: ARU.
See Also: RUN

PURPOSE: Sets the computer to start program execution automatically on power on.

REMARKS: When ARUN is included as the first program statement (i.e., with the lowest line number in the program) at power on, the program will execute as if a RUN command had been entered from the keyboard. The computer must have been turned off in RUN mode. It is ignored if it is not the first statement.

If you have a program module installed in one of the slots on the PC-1600, when ARUN is executed at power on the PC-1600 will search for any ARUN command in any programs which exist in memory and execute that command. The search includes programs in program modules in slot 1 or slot 2, and the search order is:

slot 2 → slot 1 → internal RAM

Refer to page 75; AUTORUN Files for additional information on autoexecution of files.

NOTE:
If any peripheral devices are changed while the power is off, including printer, cassette tape recorder, floppy disk drive and memory modules, the computer may return an error code to say that peripheral devices are not ready and not execute the program. Start execution with the RUN command in this event. Executing ARUN at power on does not clear variables or parameters like the RUN command does. Use a CLEAR statement in the program to do this.

EXAMPLE:

```
10:ARUN :CLS
20:PRINT "WELCOME TO THE WORLD OF"
30:PRINT "THE PC-1600"
40:PRINT "THE TIME IS NOW ";TIME$
50:PRINT "YOU HAVE ";MEM;" BYTES FREE"
60:END
```

The program runs automatically on POWER ON.

PRO
RUN
PROGRAM

ASC

FORMAT: 1. ASC(<string variable>)
2. ASC("<string>")

Abbreviation:
See Also: CHR$

PURPOSE: Returns the ASCII code value for the first character in the specified string.

REMARKS: The string can be specified as the contents of a string variable in the form X$ or as an actual string enclosed in quotes, "XXXX". Only the value of the first character in the string is returned regardless of the length. For character code tables, refer to Appendix C.

EXAMPLE:

```
10:INPUT "ENTER A CHARACTER ";A$
20:N=ASC (A$)
30:PRINT "THE ASCII CODE IS ";N
40:GOTO 10
50 END
```

[10] The user hits a key to enter any character.
[20] ASC finds the code number for this character.
[30] Prints out the answer.
[40] Repeats until the user halts the program by hitting the break key.

ASN

PRO
RUN
PROGRAM

FORMAT: 1. ASN(X)

Abbreviation: AS.

See Also: ACS, ATN, SIN

PURPOSE: Returns the arc sine of X.

REMARKS: This function returns the inverse sine of the expression X in degrees, radians, or as a gradient value depending on which mode the computer is set to with the DEGREE, RADIAN, or GRAD command. The value of X is limited to: $-1 \leqq X \leqq 1$.

In DEG mode, ASN(X) is returned in the range $-90°$ to $90°$. In RAD mode, ASN(X) is returned in the range $-\pi/2$ to $\pi/2$ radians. In GRAD mode, ASN(X) is returned in the range -100 to 100.

EXAMPLE:

```
5:WAIT 60
10:CLS
20:DEGREE
30:PRINT "ARC SIN","ANGLE"
40:FOR H=0TO 10
50:X=H/10
60:DX=ASN X
70:PRINT X,DX
80:NEXT H
```

PRO
RUN
PROGRAM

ATN

FORMAT: 1. ATN(X)

Abbreviation: AT.

See Also: ACS, ASN, TAN

PURPOSE: Returns the arc tangent of X.

REMARKS: This function returns the inverse tangent of the expression X in degrees, radians, or as a gradient value depending on which mode the computer is set to with the DEGREE, RADIAN, or GRAD command.

In DEG mode, ATN(X) is returned in the range $-90°$ to $90°$. In RAD mode, ATN(X) is returned in the range $-\pi/2$ to $\pi/2$ radians. In GRAD mode, ATN(X) is returned in the range -100 to 100.

EXAMPLE:

```
10:CLS
20:RADIAN
25:PRINT "TANGENT","ANGLE"
30:FOR T=0TO 20
40:RT=ATN (T)
50:PRINT T,RT
60:NEXT T
```

AUTO

PRO

FORMAT: 1. AUTO
2. AUTO<line #>
3. AUTO<line #>,<increment>
4. AUTO,<increment>

Abbreviation: AU.
See Also: RENUM

PURPOSE: Turns on automatic program line number generation for easier keying in of the program in PRO mode.

REMARKS: After AUTO has been entered, each time the **ENTER** key is pressed after keying in a program line, the next input line appears with the cursor positioned ready for input after the new line number. If a statement already exists in memory with that line number, the existing line is displayed.

Entering AUTO with no parameters generates line numbers starting at 10 in increments of 10. Entering AUTO<line #> generates line numbers from the specified number in increments of 10. Entering AUTO<line #>,<increment> generates line numbers from the specified number in the specified increment.

To cancel the AUTO command, press the **BREAK** key or the **CL** key, or press the **ENTER** key twice.

EXAMPLE:

```
>AUTO
```

Generates line numbers 10, 20, 30, 40 ....

```
>AUTO 100
```

Generates line numbers 100, 110, 120, 130 ....

```
>AUTO 400,20
```

Generates line numbers 400, 420, 440, 460 ....

PRO
RUN
PROGRAM

BEEP

FORMAT: 1. BEEP<number>
2. BEEP<number>[,<tone>[,<duration>]]

Abbreviation: B.
See Also: BEEP ON/OFF

PURPOSE: Generates beeps of the specified tone and duration through the computer's internal speaker.

REMARKS: <number> specifies the number of times the beep will sound in the range 0 to 65535.

The <tone> option specifies the rising tone (frequency) of the beep in the range 255 to 0. 255 corresponds to a frequency of approximately 230 Hz, and 0 corresponds to a frequency of approximately 7 kHz. The default value for <tone> gives a frequency of about 4 kHz.

The <duration> option specifies the duration of the beep. The beep duration setting varies with the <tone> parameter. The same <duration> value will appear relatively longer for lower frequencies. The default value is 160.

EXAMPLE:

```
10:FOR I=1TO 3
20:FOR J=50TO 250STEP 50
30:BEEP I,J
40:NEXT J
50:NEXT I
60:END
```

[10] This outer loop is used to change the number of beeps from 1 to 3.
[20] The inner loop counter is used to change the tone.
[30] The BEEP statement is executed 15 times. Notice how the duration of the beep lengthens as the tone drops even though the <duration> default value is used every time.

BEEP ON/OFF

PRO
RUN
PROGRAM

FORMAT: 1. BEEP ON
2. BEEP OFF

Abbreviation: B.
See Also: BEEP

PURPOSE: Disables or re-enables the BEEP command.

REMARKS: BEEP OFF disables the BEEP command. It also disables the monitoring tones produced when the computer is reading a cassette tape file.

BEEP ON re-enables the BEEP command after it has been turned off with BEEP OFF.

EXAMPLE:

```
10:BEEP 4
20:BEEP OFF
30:BEEP 4
40:BEEP ON
50:END
```

[10] Beep 4 times.
[20] Until BEEP ON is executed, all subsequent BEEP statements will now be disabled.
[30] When the program is run only the 4 beeps of line 10 are heard.
[40] As a general rule, it is always a good idea to reset defaults for software features. Here, on termination of the program, BEEP is re-enabled.

PRO
RUN
PROGRAM
RAM

BLOAD

FORMAT: 1. BLOAD "<d:filename>"[,#<mem bank>,<address>]

Abbreviation: BL.
See Also: BSAVE, CLOAD, NEW, SET

PURPOSE: Loads a machine language program into memory from floppy disk, RAM disk module, or tape.

REMARKS: <d: file name> specifies the device and file name. The following device names can be specified:

S1:, S2:	RAM disk modules
X:, Y:	Floppy disk drive
COM1:, COM2:	RS-232C and optical serial I/O ports
CAS:	Cassette tape

<mem bank> specifies a memory bank from 0 to 7 as the destination for the machine language program.

<address> specifies an address in the specified memory bank in hexadecimal. It is the address from which storage of the program will begin. If these two parameters are not specified, the program will be loaded back to the address in the memory bank from which it was previously saved to the device.

If the program was originally saved with an auto-start address specified (see BSAVE command), the program will be loaded and executed from that address automatically. Refer to Appendix D for details on memory banks and addresses.

EXAMPLE:

```
>BLOAD"X:RXOUT"
```

Loads the machine language program RXOUT on floppy disk drive X: into the same memory location where it was saved from.

BREAK ON/OFF

FORMAT: 1. BREAK ON
2. BREAK OFF

Abbreviation: BR.
See Also: CONT

PURPOSE:

Used to disable or re-enable the BREAK key.

REMARKS: BREAK OFF disables the BREAK key. This means that a program cannot be interrupted from the keyboard during execution. BREAK ON re-enables the BREAK key. During execution of the program, if the BREAK key is pressed, execution is interrupted, and the message "BREAK IN <line #>" is displayed on the screen. Execution can be resumed with the CONT command.

As a general principle, it is a good idea to set BREAK OFF at the start of long programs with little screen display or printer output; usually calculation programs. But BREAK ON should be set at the end of the program to re-enable the BREAK key for normal use. If a program becomes trapped in an infinite loop with the BREAK key set to off, the only way to stop execution is to press the RESET button. So it is best to ensure that programs are thoroughly debugged before incorporating the BREAK OFF statement.

EXAMPLE:

```
>BREAK OFF
>RUN
>BREAK ON
```

RUN starts the execution of a lengthy program. So as to avoid any interruption of the program by hitting the break key, BREAK OFF is keyed in prior to running the program. On program termination, BREAK ON is keyed in to re-enable the break key. Alternatively, BREAK OFF and BREAK ON could be included in the program itself near the beginning and end respectively. This way avoids having to remember to key in BREAK ON on program termination.

PRO
RUN
PROGRAM
RAM

BSAVE

FORMAT: 1. BSAVE "<d:filename>",#<mem bank>,<start address>,
<end address>[,<auto-start address>]

Abbreviation: BS.
See Also: BLOAD, CSAVE M

PURPOSE: Saves a machine language program to a floppy disk, RAM disk module, or tape.

REMARKS: <d:file name> specifies the device and file name. The following device names can be specified:

S1:, S2:	RAM disks
X:, Y:	Floppy disk drive
COM1:, COM2:	RS-232C and optical serial I/O ports
CAS:	Cassette tape

If the specified device has been write protected, the command will generate an error code.

<mem bank> specifies a memory bank from 0 to 7 where the machine language program is stored.

<start address> specifies the lower address in the specified memory bank where the program is stored.

<end address> specifies the upper address in the specified memory bank where the program is stored.

<auto-start address> specifies the address from where auto-execution should start after the program has been loaded to the same memory location with the BLOAD command. When no <auto-start address> is specified, the default value &FFFF turns off the auto start function.

Refer to Appendix D for details on memory banks and addresses.

Unlike the SAVE command no file extension name is automatically added to the file name.

EXAMPLE:

```
>BSAVE"S1:SORT",#1,&8000,&8AFF
```

Saves machine language program in memory bank 1 from address **&8000** to address &8AFF to the RAM disk in slot 1 with file name SORT.

PRO
RUN
PROGRAM

CALL

FORMAT: 1. CALL [#<mem bank>,]<address>[,<variable name>]

Abbreviation: CA.
See Also: NEW, POKE, XPOKE

PURPOSE: Used to call and execute a machine language program from a BASIC program.

REMARKS: The CALL statement transfers control to a machine language program or subroutine stored in memory. When the program is called, a single variable value may be passed from the current BASIC program to the machine language program, and when execution of the machine language program is finished, the same single variable will be passed back to the calling program. The B register and the DE register pair are used for this purpose. The machine language program must have been written into memory using the POKE or XPOKE statements before it can be called.

<mem bank> specifies a memory bank from 0 to 7 where the machine language program is stored. If this parameter is not specified, the default memory bank is bank 0.

<address> specifies the lower address in the specified memory bank where the machine language program is stored in the range 0 to 65535 (&0 to &FFFF).

<variable name> specifies the variable whose value is to be passed to the machine language program on entry, and passed back to the BASIC calling program on exit if the carry flag is raised. If <variable name> specifies a numeric variable, the value must be an integer in the range −32768 to 32767. The value is passed to the DE register pair on entry, and the contents of the DE register pair are passed back to the calling program as a BCD value with the same variable name on exit. If <variable name> specifies a string variable, the DE register pair hold the start address of the string location, and register B holds the string length.

EXAMPLE:

```
:
:
400:CALL #3,&8000,X
410:PRINT "THE VALUE OF X RETURNED"
420:PRINT "FROM THE MACHINE LANGUAGE IS ";X
:
:
```

[400] Control passes to the machine language held in memory bank 3 with execution starting in memory location &8000.
[410–420] The CALL passed across the numeric variable "X" and the returned value is printed out here.

CHAIN

FORMAT: 1. CHAIN
2. CHAIN ["<filename>"][,<line #>]

Abbreviation: CHA.
See Also: CSAVE, MERGE

PURPOSE: Makes it possible for one BASIC program to load and execute a program on cassette tape in PC-1500 mode.

REMARKS: The program to be chained must be on tape.

CHAIN loads the first program on tape into memory and executes it from the first line number. CHAIN <line #> loads the first program on tape into memory and executes it from the specified line number. CHAIN "<filename>" searches the tape for the program called <filename>, loads it into memory and executes it from the first line number. If optional <line #> is also specified, execution starts from that line number.

Using the CHAIN statement allows programs which are larger than the memory capacity to be divided into sub-programs which will fit into memory, to be loaded and executed sequentially. If a password has been set with the PASS command CHAIN will generate an error.

EXAMPLE:

```
>RUN

10:REM THIS IS PROG1
20:
30:
:
:
400:CHAIN "PROG2"

10:REM THIS IS PROG2
20:
30:
:
:
200:END
>
```

RUN starts execution of PROG1 that is currently in memory. The final line [400] of the program CHAINs to the continuing program PROG2 saved on cassette tape. PROG2 is loaded from tape into memory and execution continues.

CHR$

PRO
RUN
PROGRAM

FORMAT: 1. CHR$(<integer expression>)
2. CHR$(<integer>)

Abbreviation: CH.
See Also: ASC

PURPOSE: Returns the character whose ASCII code equals the value of the <integer expression> or the value of the <integer>.

REMARKS: The CHR$ function can be used to send special control characters to terminal equipment or through a serial port, or to print out graphics characters which are not on the keyboard from a BASIC program. For character code tables, refer to Appendix C.

EXAMPLE:

```
10:FOR X=33TO 126
20:PRINT CHR$(X);
30:NEXT X
40:END
```

[20] Prints out the ASCII code table characters on the screen.

PRO
RUN
PROGRAM

CLEAR

FORMAT: 1. CLEAR

Abbreviation: CL.
See Also: DIM, ERASE, TITLE

PURPOSE: Erases all variables from memory including fixed variables.

REMARKS: CLEAR frees up the space used by all variables including the fixed variables A–Z, A$–Z$, or @(1)–@(26) which do not need a DIM statement, since they are permanently assigned, and resets numeric values to zero and string variables to null strings (ASCII code 0).

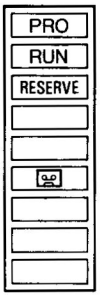
The CLEAR command can also be used within a program. It is used to recover the space occupied in memory to store variables when they are no longer needed; for instance, when the variables used in the first part of a program are not needed in the second part and available memory space is limited. CLEAR may also be used at the beginning of a program when several programs are resident in memory and you want to free the memory space used by execution of prior programs.

EXAMPLE:

```
5:WAIT 30
10:DIM C(5)
20:FOR N=1TO 5
30:READ A:LET C(N)=A:PRINT C(N)
40:NEXT N
50:DATA 10,20,30,40,50
60:CLEAR
70:PRINT A
```

[5] Sets wait time to display printout
[10] Dimensions array C(N).
[20–40] Reads the numbers from the data line into the elements of C(N) and prints out.
[60] Frees up the space allocated to C(1) to C(5) and resets A to 1.
[70] Prints the cleared value for A.

CLOAD



FORMAT: 1. CLOAD ["<filename>"][,R]

Abbreviation: CLO.
See Also: CLOAD?, CSAVE, MERGE

PURPOSE: Used to load a program or a function key usage program saved on cassette tape with CSAVE into memory in PC-1500 mode.

REMARKS: CLOAD on its own clears all current program lines in memory and loads the first program from the tape starting from the current tape position. CLOAD "<filename>" clears all current program lines in memory, searches the tape for the program with the specified name, and loads it into memory.

In PRO and RUN modes, the program is loaded into normal user memory, while in RESERVE mode, the function key usage program is loaded into reserve memory. Care should be taken to ensure that programs are not loaded into the wrong memory by mistake, as the current contents will be destroyed.

If the R option is specified, the file will be loaded into memory and executed automatically as if RUN had been entered. If the file does not contain a BASIC program, an execution error will result.

If the password has been set with the PASS command, CLOAD will generate an error.

CLOAD will only load programs which were saved to tape using the CSAVE command.

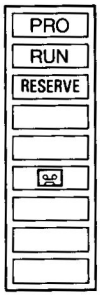
EXAMPLE:

```
>CLOAD
```

Loads into memory the first program found on tape starting from the current tape head position.

```
>CLOAD"PROG01"
```

Searches the tape for program "PROG01" starting from the current tape head position and loads it into memory.



CLOAD?

FORMAT: 1. CLOAD? ["<filename>"]

Abbreviation: CLO.?
See Also: CLOAD, CSAVE, MERGE

PURPOSE: Used to verify that a program has been recorded onto cassette tape with no errors in PC-1500 mode.

REMARKS: The verified program may be either the program used in RUN and PRO modes or, in RESERVE mode, the function key usage program.

When using the CLOAD? command by itself, the tape must be manually rewound to the start of the program before entering the command. The program will then be re-read from the tape and compared block by block with the version in memory. If there are any errors, an error code will be generated. If there are no errors, the (>) prompt sign will reappear.

CLOAD? "<filename>" will search the tape for the program specified and verify it against the contents of memory.

EXAMPLE:

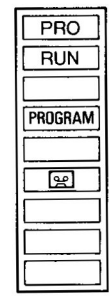
```
>CLOAD
>CLOAD?
```

Loads and verifies the first program found on tape starting from the current tape head position.

```
>CLOAD"PROG01"
>CLOAD?"PROG01"
```

Searches the tape for program "PROG01" starting from the current tape head position and loads and verifies it.

CLOAD M



FORMAT: 1. CLOAD M ["<filename>"][;#<mem bank>,<address>]

Abbreviation: CLO. M
See Also: BLOAD, CALL, CSAVE M, NEW

PURPOSE: Used to load a machine language program from cassette tape into memory in PC-1500 mode.

REMARKS: Machine language programs are loaded into a different area of memory to normal BASIC programs, and the format used to store them on tape is different. A machine language program cannot be loaded with the normal CLOAD command.

<mem bank> specifies a memory bank from 0 to 7 as the destination for the machine language program.

<address> specifies an address in the specified memory bank in hexadecimal. It is the address from which storage of the program will begin. If these two parameters are not specified, the program will be loaded back to the address in the memory bank from which it was previously saved to the device. If the program was originally saved with an auto-start address specified (see CSAVE M command), the program will be loaded and executed from that address automatically.

EXAMPLE:

```
>CLOADM"MAC1"
```

Loads machine code program MAC1 from tape into the same memory bank and location that it was saved from with CSAVE M. Memory bank and location information is held with the machine language program on tape.

PRO
RUN
PROGRAM
RAM

CLOSE

FORMAT: 1. CLOSE
2. CLOSE #<file #>[,#<file #> .....]

Abbreviation: CLOS.
See Also: END, OPEN

PURPOSE: Terminates access to a file on the currently accessed device.

REMARKS: CLOSE specified alone closes all open files. <file #> is the number under which the file was opened with the OPEN statement.

Once opened, a file must be closed before it can be re-opened for a different purpose (i.e.; input, output or append usage). If an attempt is made to open a file which is already open, an error code will be generated.

All files are automatically closed upon execution of the END, NEW, RUN and LOAD commands, when the program is edited or when the computer is switched OFF.

EXAMPLE:

```
5:MAXFILES =2
10:OPEN "X:PAYMENT"FOR INPUT AS #1
20:OPEN "CAS:UPDATE"FOR OUTPUT AS #2
:
:
:
400:CLOSE #1,#2
```

[400] Closes both files after processing.

CLS

PRO
RUN
PROGRAM

FORMAT: 1. CLS

Abbreviation:
See Also:

PURPOSE: Clears the screen.

REMARKS: CLS clears all lines of the screen and positions the cursor at the top left "home" position (0,0) in MODE 0.

EXAMPLE:

```
>CLS
```

Clears the screen and repositions the cursor to the home position.

COLOR

FORMAT: 1. COLOR <number>

Abbreviation: COL.
See Also:

PURPOSE: Sets the pen color for the printer.

REMARKS: Four colors can be specified with the COLOR statement:

0 — black
1 — blue
2 — green
3 — red

At power on, the default color is black (0).

EXAMPLE:

```
>COLOR 3
```

Selects the red pen.

COM$

FORMAT: 1. COM$ "COMn:"

Abbreviation: COM.
See Also: SETCOM

PURPOSE: Returns a string containing the values set for the communication parameters for the port specified in the last SETCOM command.

REMARKS: The string contains the parameters in the same order as specified in the SETCOM command:
, <WL>, <PR>, <ST>, <XO>, <SI>.

COM$ "COM1:" returns the values for the RS-232C serial port.
COM$ "COM2:" returns the values for the optical serial I/O port.
COM$ "COM:" returns the values for the currently opened port.

EXAMPLE:

```
10:SETCOM "COM1:",300,8,N,1,X,S
20:PRINT COM$"COM1:"
```

[10] Sets the parameters for the RS-232C serial interface.
[20] Prints out the string containing the parameters set in line 10.

PRO
RUN
PROGRAM

COMn ON/OFF/STOP

FORMAT: 1. COMn ON
2. COMn OFF
3. COMn STOP

Abbreviation:
See Also: ON COMn GOSUB, SETCOM

PURPOSE: Enables or disables interrupts received through one of the communication ports.

REMARKS: Parameter n specifies the RS-232C port (n=1) or the optical I/O port (n=2).

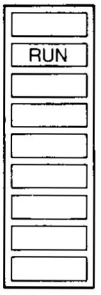
COMn ON enables interrupts to be received through the specified port. The ON COMn GOSUB statement can then be used to branch execution to the relevant interrupt-processing routine. COMn OFF disables interrupts from the specified port. COMn STOP disables interrupts from the specified port but registers the most recent of any interrupt requests which are received. If COMn ON is executed subsequently, that interrupt will be processed immediately. The default value is COMn STOP.

EXAMPLE:

```
10:COM1 ON
20:
30:
40:COM1 OFF
```

[10] Enables interrupts to the RS-232C port.
[40] Disables interrupts to the RS-232C port.

CONT



FORMAT: 1. CONT

Abbreviation: C.
See Also: RESUME, RUN, STOP, WAIT

PURPOSE: Resumes execution of a program which has been interrupted by a STOP statement, a PRINT statement or by the BREAK key.

REMARKS: This command is often used during program debugging together with STOP. When execution is interrupted by STOP, or by pressing the BREAK key, entering CONT at the keyboard will resume program execution at the point at which it was terminated, provided no program changes are made during the interruption. (GOTO <line #> can be keyed in to resume execution at a different line number.) Note that CONT will not resume execution after END or when execution is interrupted by an error condition. For this, see RESUME.

EXAMPLE:

```
10:PRINT "PROGRAM HALTS HERE"
20:STOP
30:PRINT "AFTER STOP"
40:PRINT "GOODBYE"
50:END

>RUN
PROGRAM HALTS HERE

 BREAK IN 20
>
>CONT
AFTER STOP
GOODBYE
>
```

After STOP terminates the program, CONT is used to continue running the program from the line following the one where the break occurred.

PRO
RUN
PROGRAM
RAM

COPY

FORMAT: 1. COPY "<d1:filename1.ext>" TO "<[d2:]filename2.ext>"

Abbreviation: COP.
See Also: SET

PURPOSE: Copies a file from one device to another device.

The COPY command can be used to copy files between devices. <d1:filename1.ext> specify the device and the file name to copy from (source). <d2:filename2.ext> specify the device and the file name to copy to (destination). The file extension must always be given when copying a file.

If the destination device is not specified, the source device is used.

The file will not be copied if filename2 already exists on destination device.

The COPY command can also be used to change the floppy disk drive name. Initially the drive name is set to X: However, the PC-1600 can address drive name Y: This allows backup copies of your disks to be made with the single drive. To copy one disk to another, refer to the Floppy Disk Drive section in Chapter 6.

		Copy TO device name						
		X:	Y:	S1:	S2:	CAS:	COM1:	COM2:
	X:	○	○	○	○	○	○	○
	Y:	○	○	○	○	○	○	○
	S1:	○	○	○	○	○	○	○
	S2:	○	○	○	○	○	○	○
	CAS:	○	○	○	○	×	×	×
	COM1:	○	○	○	○	×	×	×
	COM2:	○	○	○	○	×	×	×

○ indicates copying is possible.
× indicates copying is not possible.

EXAMPLE:

```
>COPY"S1:RICH.BAS" TO "S2:RICH.BAS"
```

Copies the file RICH.BAS from the RAM disk module in expansion slot 1 to the same file name in the RAM disk module in expansion slot 2.

```
>COPY"X:HAC.BAS" TO "HACOPY.BAS"
```

Copies the file HAC.BAS on floppy disk to the new file HACOPY.BAS on the same floppy disk.

COS

PRO
RUN
PROGRAM

FORMAT: 1. COS(X)

Abbreviation:
See Also: ACS, SIN, TAN

PURPOSE: Returns the cosine of the angle X.

REMARKS: This function returns the cosine of the angle X, where X is expressed in degrees, radians or as a gradient value, depending on which mode the computer is set to with the DEGREE, RADIAN or GRAD command.

EXAMPLE:

```
10:DEGREE
20:PRINT "COS OF 60 IS ";COS (60)
30:PRINT "COS OF 90 IS ";COS (90)
40:END

>RUN
COS OF 60 IS  0.5
COS OF 90 IS  0
>
```

PRO
RUN
RESERVE
PROGRAM

CSAVE

FORMAT: 1. CSAVE
2. CSAVE ["<filename>"][,A][;<line #>,<line #>]

Abbreviation: CS.
See Also: CLOAD, CLOAD?, LLIST, MERGE

PURPOSE: Saves a program or part of a program held in memory onto cassette tape in PC-1500 mode.

REMARKS: The program saved will be the normal program if in RUN or PRO mode, or the function key usage program if in RESERVE mode.

CSAVE alone saves the whole program starting from the current tape position, and with no specified program name.

CSAVE "<filename>" saves the program under the specified file name. When the [,A] option is specified, the file is saved in ASCII format; otherwise it is saved in binary format. The <line #>,<line #> option allows specification of any sequence of line numbers in the same way as for the LLIST command.

If a password has been set with the PASS command, the program cannot be saved with CSAVE until the password is cleared.

When loading a program which was saved to tape with the CSAVE command, the CLOAD command must be used.

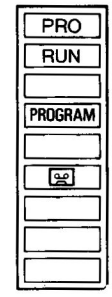
The CSAVE command cannot be used with the CE-1600P Printer/Cassette Interface Unit. It is only for use in PC-1500 mode with the CE-150 and CE-162E Units.

EXAMPLE:

```
>CSAVE"PROG01";200,380
```

Saves lines 200 to 380 of the current program in memory to tape under the filename "PROG01".

CSAVE M



FORMAT: 1. CSAVE M "<filename>"; #<mem bank>, <start address>, <end address> [, <auto-start address>]

Abbreviation: CS. M
See Also: CLOAD M, CALL, BSAVE

PURPOSE: Saves a machine language program held in memory onto cassette tape in PC-1500 mode.

REMARKS: <mem bank> specifies a memory bank from 0 to 7 where the machine language program is stored. <start address> specifies the lower address in the specified memory bank where the program is stored. <end address> specifies the upper address in the specified memory bank where the program is stored. <auto-start address> is optional and specifies the address from where execution should start after reloading. The default value is &FFFF, which turns off the auto-start function.

EXAMPLE:

```
>CSAVEM"MAC1";#2,&8000,&8AFF
```

Saves the machine language program "MAC1" to tape that is currently held in memory bank 2 from location &8000 to &8AFF (32768 to 35583). This memory bank and location information is stored with the program itself on tape. If CLOAD M is used later to reload the program into memory without specifying these values explicitly, the values on tape will be read and used automatically.

PRO
RUN
PROGRAM

CSIZE

FORMAT: 1. CSIZE <size>

Abbreviation: CSI.
See Also: PCONSOLE

PURPOSE: Sets the size of the characters printed out by the printer.

REMARKS: The <size> parameter specifies the character size in the range 1 to 9. The character size set for the value of the <size> parameter is tabulated below:

<size> setting	1	2	3	4	5	6	7	8	9
Chars/line	160	80	53	40	32	26	22	20	17
Height (mm)	1.2	2.4	3.6	4.8	6.0	7.2	8.4	9.6	10.8
Width (mm)	0.8	1.6	2.4	3.2	4.0	4.8	5.6	6.4	7.2

NOTE:
The character sizes corresponding to the value of the <size> parameter given in the above table are with the line length set to infinity with the PCONSOLE statement, <length> =0.

If character size has been set to any value greater than 2 with CSIZE, executing a LLIST command will reset the character size to 2.

EXAMPLE:

```
>CSIZE 2
```

Sets the printer character size to size setting 2.

CURSOR

PRO
RUN
PROGRAM

FORMAT: 1. CURSOR <column>[,<line>]

Abbreviation: CU.
See Also: PRINT

PURPOSE: Positions the cursor at any position on the screen.

REMARKS: The CURSOR command positions the cursor at a specified column on a specified line. The column can be specified in the range 0 (left end) to 25 (right end), and the line number can be specified in the range 0 (top line) to 3 (bottom line). The default is the current line that the cursor is on.

EXAMPLE:

```
10:WAIT 50
20:FOR N=1TO 6
30:READ A$
40:CURSOR 12,1
50:PRINT A$
60:NEXT N
70:END
80:DATA "H","E","L","L","O","!"
```

This program will print the message HELLO! in the middle of the screen with each subsequent character overwriting the previous one.

[10] WAIT adds a short delay to the printing of the characters so that the sequence is easy to read.
[20] Loops through the 6 characters.
[30] Reads the next character.
[40] Moves the cursor to character position 12 along the second line from the top of the screen.
[50] Prints the character and overwrites any existing one.

DATA

FORMAT: 1. DATA <list of constants>

Abbreviation: DA.
See Also: READ, RESTORE

PURPOSE: Lists data items to be read in by the READ statement.

REMARKS: A DATA statement may contain any numeric or string constants separated by commas. String constants must be enclosed in quotes.

Once the data items on a data line have been read, they cannot be read again until a RESTORE statement has been executed.

Data statements are not executable, and so can be put anywhere in a program, often as the last lines where they can be easily found for future modification. A program may contain any number of data lines, and they will be read by READ statements in order of line number. A single READ statement need not correspond to a single DATA statement.

EXAMPLE:

```
10:FOR J=1TO 4
20:READ A$,B
30:PRINT A$,B
40:NEXT J
50:END
60:DATA "MICHAEL:MIKE",23,"DAVID",38,"WENDY",-24,"   BRIAN",34

>RUN
MICHAEL:MIKE 23
DAVID         38
WENDY        -24
   BRIAN      34
>
```

DATE$

PRO
RUN
PROGRAM

FORMAT: 1. DATE$="MM/DD"
2. DATE$

Abbreviation: DATE.
See Also: TIME$, ALARM$

PURPOSE: DATE$ is a system variable which contains the date of the computer's built-in real-time clock.

REMARKS: In the first format, as a statement, DATE$ sets the value of the real time clock. MM is a two digit number between 01 and 12 indicating the month. DD is a two digit number between 01 and 31 indicating the day.

In the second format, DATE$ returns the date of the real-time clock in MM/DD format to the program as a string variable. The day variable moves forward one day when the internal time held in the TIME$ variable changes from 23:59:59 to 00:00:00.

EXAMPLE:

```
>DATE$="12/25"
```

Sets the date of the real time clock to 25th December.

```
:
:
:
450:PRINT "TODAY'S DATE IS ";DATE$
:
:
:
```

[450] Prints out the current date from the real time clock.

PRO
RUN
PROGRAM

DEG

FORMAT: 1. DEG <dd.mmssrr>

Abbreviation:
See Also: DMS

PURPOSE: For an angle specified in degrees, minutes and seconds, converts the value to decimal degrees.

REMARKS: The angle must be expressed in the format dd.mmssrr where dd is degrees (range not limited), mm is minutes in the range 00 to 59, ss isseconds in the range 00 to 59, and rr is the remainder in the range 00 to 99. Note that the degree part is separated from the rest by a decimal point. DEG will return the angle in degrees and decimal degrees to 10 significant digits.

EXAMPLE:

```
10:X=DEG 50.300000
20:PRINT X
30:END

>RUN
                          50.5
>
```

DEGREE

PRO
RUN
PROGRAM

FORMAT: 1. DEGREE

Abbreviation: DE.
See Also: RADIAN, GRAD

PURPOSE: Sets the COMPUTER to degree mode.

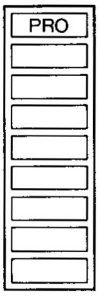
REMARKS: In degree mode, DEG is displayed on the top line of the screen. Input to the arguments of SIN, COS and TAN must be in degrees. Values returned by the ASN, ACS and ATN functions will be in decimal degrees.

EXAMPLE:

```
10:DEGREE
20:PRINT "TRIG FUNCTIONS NOW IN DEGREES"
30:PRINT ASN(0.5),ASN(1.0)
40:PRINT ACS(0.5),ACS(1.0)
50:PRINT ATN(0.5),ATN(1.0)
60:END
```

Try running this program and compare the results with those for GRAD and RADIAN.

[10] Select DEGREE mode.
[30–50] Print out some sample values.



DELETE

FORMAT: 1. DELETE <line #>
2. DELETE <line #>,
3. DELETE <line #>, <line #>
4. DELETE ,<line #>

Abbreviation: DEL.
See Also: NEW

PURPOSE: Deletes specified program lines in memory.

REMARKS: DELETE <line #> deletes only the specified program line. DELETE <line #>, deletes program lines from the line number specified up to the highest program line in memory. DELETE <line #>, <line #> deletes all program lines between the first specified line number (lower value) and the second specified line number (higher value). DELETE ,<line #> deletes program lines from the lowest line number in memory up to the specified line number.

To delete the whole program, use the NEW command.

EXAMPLE:

```
>DELETE 150
```

Deletes line 150 only.

```
>DELETE 200,
```

Deletes from line 200 to the highest line number.

```
>DELETE 50,150
```

Deletes all lines between line 50 and line 150.

```
>DELETE ,35
```

Deletes from lowest line number up to line 35.

DIM

PRO
RUN
PROGRAM

FORMAT: 1. DIM numeric variable name (<size>)
2. DIM string variable name (<size>)[*<length>]
3. DIM numeric array name (<rows>,<columns>)
4. DIM string array name (<rows>,<columns>)[*<length>]

Abbreviation: D.
See Also: CLEAR, ERASE

PURPOSE: Used to reserve space in memory for numeric and string array variables, and to specify the number of subscripts in each array.

REMARKS: Except for single-value variables A–Z, A$–Z$, @(1)–@(26), and @$(1)–@$(26), all array variables must be declared with a DIM statement before use in a program, to allocate sufficient storage space.

DIM numeric variable name (<size>) specifies the number of elements in a one-dimensional numeric variable in the range 0 to 255.

DIM string variable name (<size>) [*<length>] specifies the number of elements in a one-dimensional string variable in the range 0 to 255. The maximum number of characters in each element is specified with the optional *<length> parameter in the range 1 to 80. If length is not specified, the default is 16 characters.

DIM numeric array name (<rows>,<columns>) specifies the number of rows and columns in a two-dimensional numeric array in the range 0 to 255.

DIM string array name (<rows>,<columns>)[*<length>] specifies the number of rows and columns in a two-dimensional string array in the range 0 to 255. The optional *<length> parameter specifies the maximum number of characters in each element in the range 0 to 80.

The minimum value of any subscript is zero, which means that the number of rows or columns in an array is one greater than the number in the DIM statement. So, for example, DIM A (2,3) declares an array with 3 rows and 4 columns:

	Col. 1	Col. 2	Col. 3	Col. 4
Row 1	A(0,0)	A(0,1)	A(0,2)	A(0,3)
Row 2	A(1,0)	A(1,1)	A(1,2)	A(1,3)
Row 3	A(2,0)	A(2,1)	A(2,2)	A(2,3)

That is a total of 12 elements.

Once the size of an array has been DIMensioned, it cannot be redimensioned until it has been reset with a CLEAR, NEW or RUN statement or selectively reset with the ERASE statement. Initially, the elements in a numerical array are set to zero and elements in a string array are set to null strings.

Error codes are generated when a program references an array undeclared in a DIM statement, when a DIM statement declares an array already declared in a previous DIM statement, and when a subscript that exceeds the value set in the DIM statement is referenced.

EXAMPLE:

```
10:DIM C(13)

20:DIM F$(10)

30:DIM H(4,6)

40:DIM G$(7,5)*25
```

[10] Specifies a numeric variable C with 14 elements.
[20] Specifies a string variable F$ with 11 elements.
[30] Specifies a 5-row by 7-column numeric array of 35 elements.
[40] Specifies an 8-row by 6-column string array of 48 elements, each 25 characters long.

DMS

PRO
RUN
PROGRAM

FORMAT: 1. DMS <angle>

Abbreviation: DM.
See Also: DEG

PURPOSE: For an angle specified in decimal degrees, converts the value to degrees, minutes and seconds.

REMARKS: The angle is returned in the format dd.mmssrr where dd is degrees (range not limited), mm is minutes in the range 00 to 59, ss is seconds in the range 00 to 59, and rr is the remainder in the range 00 to 9. Note that the degree part is separated from the rest by a decimal point. <angle> must be entered in degrees as a decimal dd. with a decimal point.

EXAMPLE:

```
10:X=DMS 50.5
20:PRINT X
30:END

>RUN
                          50.3
>
```

PRO
RUN
PROGRAM
RAM

DSKF

FORMAT: 1. DSKF "d:"

Abbreviation: DS.
See Also:

PURPOSE: Returns the amount of free space on the specified device.

REMARKS: The following device names can be specified:

S1:	Device in expansion slot 1
S2:	Device in expansion slot 2
X:,Y:	Floppy disk drive

The amount of free space on the device available for storage of files or programs is returned as an integer in bytes.

EXAMPLE:

```
>DSKF"S1:"
```

Returns the available space in bytes in the module in expansion slot 1 (e.g., RAM disk module).

END

FORMAT: 1. END

Abbreviation: E.
See Also: STOP

PURPOSE: Stops execution of the current program, closes all files, and the serial I/O interface if it is open.

REMARKS: The END statement need not be on the last line of the program. Especially with programs containing subroutines, the END statement may be placed before the subroutine modules, so that execution does not continue into the subroutine when the main part has been executed. If END is omitted, execution terminates when there are no more program lines to execute, and files will be closed.

EXAMPLE:

```
10:GOSUB 50
20:PRINT "AFTER THE SUBROUTINE"
30:PRINT "PROGRAM HALTS ON LINE 40"
40:END
50:PRINT "SUBROUTINE ON LINE 50"
60:RETURN
```

PRO
RUN
PROGRAM
RAM

EOF

FORMAT: 1. EOF(<file number>)

Abbreviation: EO.
See Also:

PURPOSE: Returns a value which indicates when the end of a file has been reached during input from a sequential file.

REMARKS: <file number> is the number under which the file was opened with the OPEN command.

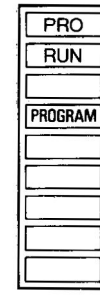
The value returned will be **0** if the end of the file has not been reached, and 1 if the end of the file has been reached.

EXAMPLE:

```
10: IF EOF(1) THEN 100
```

[10] Jumps to line **100** when end of file #1 is reached.

ERASE



FORMAT: 1. ERASE <list of variable names>

Abbreviation: ERA.
See Also: CLEAR

PURPOSE: Erases specified variables and arrays.

REMARKS: ERASE erases numeric and string variables and arrays, but does not erase fixed variables A–Z, A$–Z$, @(1) to @(26), and @$(1) to @$(26). It is not possible to erase individual elements of an array; the whole array is erased together. An array is erased by specifying the array name followed by two brackets ( ).

<list of variable names> specifies the variable names, separated by commas.

EXAMPLE:

```
10:ERASE AB,Z$()
```

PRO
RUN
PROGRAM

ERL

FORMAT: 1. ERL

Abbreviation:
See Also: ERN, ON ERROR GOTO, RESUME

PURPOSE: Returns the line number at which an error occurred during execution.

REMARKS: The ERL function is used with the ERN function and the ON ERROR GOTO statement in error processing routines to control program flow when an error occurs. A line number is only set in ERL if the error occurred during program execution.

EXAMPLE: See ERN.

ERN

PRO
RUN
PROGRAM

FORMAT: 1. ERN

Abbreviation:
See Also: ERL, ON ERROR GOTO, RESUME

PURPOSE: Returns the error code number of the last execution error.

REMARKS: The ERN function is used with the ERL function and the ON ERROR GOTO statement in error processing routines to control program flow when an error occurs.

EXAMPLE:

```
10:ON ERROR GOTO 100
20:FOR N=1 TO 20
30:READ A
40:PRINT A
50:NEXT N
60:END
100:IF ERL =30AND ERN =4THEN PRINT "YOU HAVEN'T GOT A DATA LINE"
110:STOP
```

PRO
RUN
PROGRAM

EXP

FORMAT: 1. EXP(X)

Abbreviation: EX.
See Also: LN

PURPOSE: Returns the value of the exponential function e raised to the power of X.

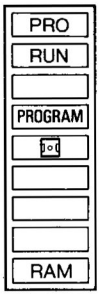
REMARKS: The expression X must be in the range +230.2585092 to −227.9559242. For values below this range, 0 is returned. To raise another number base to a power, use the "∧" function. The computer holds the value of e as 2.718281828.

EXAMPLE:

```
>PRINT EXP(10)
                22026.46579
>
```

Prints out the value of e raised to the power 10.

FILES



FORMAT: 1. FILES "<d:>"
2. FILES "<d:filename>"
3. FILES "<d:ambiguous filename>"

Abbreviation: FI.
See Also: LFILES, SET

PURPOSE: Displays directory information for the specified files on floppy disk or RAM disk.

REMARKS: The FILES command displays the file name, .BAS extension, "P" protection (see SET command), and creation date and time. d: specifies the device.

1. If no filename is specified, all files on the specified device are listed to the screen one entry at a time. To scroll to the next entry, press the ↓ key. To abort the listing, press any key but the SHIFT, CTRL, DEF or SML keys.

2. If a single file name is specified, directory information is displayed for that file.

3. An ambiguous file name can be specified to list directory information on groups of files with common name forms. There are two wildcard characters for this purpose. The asterisk "*" stands for any number of characters (including 0) in the file name. The question mark "?" stands for a single character in a file name. Examples of how these wildcard characters are used are shown below:

Filename specification	Files satisfying the specification
TEST?	TEST, TESTS, TEST1, TESTA
T??T	TEST, TEXT, TORT, TXYT
S?MPLE	SIMPLE, SAMPLE, S2MPLE, S0MPLE
A?????	ABCDEF, APPEND, APPLES, A12345
R*	RATES, R1, RETURNS, RAND2, R, ROBERT

EXAMPLE:

```
>FILES"X:"
```

Lists all files on floppy disk drive X: on the screen.

```
>FILES"S1:DATA"
```

Displays the file named DATA held in the RAM disk module in slot 1 on the screen.

```
>FILES"S2:???1"
```

Lists all files held in the RAM disk module in slot 2 on the screen that have 4-letter names ending in 1.

FOR..NEXT

FORMAT: 1. FOR <counter> = <initial value> TO <final value>
[STEP <increment>]
NEXT <counter>
2. FOR <counter> = <expression1> TO <expression2>
[STEP <expression3>]
NEXT <counter>

Abbreviation: F. ~ N.
See Also:

PURPOSE: Allows the execution of program lines between FOR and NEXT to be repeated a specified number of times.

REMARKS: The FOR..NEXT statement causes program execution to loop through the same part of the program a specified number of times. <counter> is a counter which keeps a record of the number of loops that have been made. In format 1, the <final value> and <increment> are integer numerical values in the range −32768 to +32767 (the <increment> must be non-zero). In format 2, they can be in the form of expressions, the values of which are used as the initial and final counter values. Combinations of these 2 formats are allowed. Fractional values will be truncated. The program lines following FOR are executed until the NEXT statement is encountered, and then the counter is incremented by the amount specified in <increment>, or by 1 if no increment is specified.

Next, the value of the counter is compared with <final value>. If the counter is less than or equal to <final value>, execution jumps back to the statement following the FOR statement, and the sequence is repeated. When the value of the counter exceeds <final value>, execution continues with the statement following the NEXT statement. <increment> can be a positive or negative numerical value in the range −32768 to +32767. If negative, <final value> should be less than <initial value>. If <final value> is less than <initial value> for a positive <increment>, or the reverse for a negative <increment>, the FOR..NEXT loop will be executed once only and execution will continue with the line following the NEXT statement.

FOR..NEXT loops may be nested; that is, one FOR..NEXT loop may be contained inside another loop. This, in combination with the second format of the statement using expressions for the counter specification, makes the FOR..NEXT statement possibly the most important and flexible of all the BASIC commands. FOR..NEXT statements must be paired. If a NEXT statement is encountered before the corresponding FOR statement, or if loops are nested beyond the capacity of the computer, an error code will be generated.

NOTE:
The following commands cannot be used in a FOR...NEXT loop:
ERASE, MAXFILES, INIT "COMn:", <buffer size>

EXAMPLE:

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

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

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

[5] Sets wait time for display printout.
[10] Prints out the numbers 1, 2, 3, 4, 5.
[40] Counts down from 10 to 0 in units of 1.
[70] Calculates and prints out the value of factorial N for the numbers from 1 to 10.

GCURSOR

PRO
RUN
PROGRAM

FORMAT: 1. GCURSOR X[,Y]

Abbreviation: GC.
See Also:

PURPOSE: Positions the graphics cursor at the point (X,Y) on the screen.

REMARKS: In graphics mode, the screen is treated as a 156 × 32 matrix of dots. The graphics cursor can be positioned at any dot within these coordinate values, or even outside these values, although in that case, the result will not be visible. After positioning the graphics cursor, the GPRINT statement may be used to output to that location on the screen.

The X coordinate is normally in the range 0 to 155. The Y coordinate is normally in the range 0 to 31. Both coordinates must, however, be in the range −32768 to 32767. If the Y coordinate is omitted, the currently selected Y coordinate will be assumed.

NOTE:
In MODE 1 (PC-1500 mode), the Y parameter is meaningless and should not be specified.

EXAMPLE:

```
:
:
300:X=5
310:Y=20
320:GOSUB 600
:
:
:
590:END
600:GCURSOR X,Y
610:GPRINT 255;255;255;255
620:RETURN
```

GCURSOR is useful for positioning the screen cursor prior to drawing some symbol or set pattern.

[300–320] Fixes the screen location for the symbol to be drawn and calls the subroutine to draw the symbol.
[600] Positions the cursor to the XY values set in the main body of the program.
[610] Prints the symbol. Here, the symbol is a simple block character 8 dots high by 4 dots wide.

GLCURSOR

FORMAT: 1. GLCURSOR (X,Y)

Abbreviation: GL.
See Also: LCURSOR

PURPOSE: In graphics mode, moves the printer pen to the specified position.

REMARKS: The GLCURSOR command is similar to the LCURSOR command, but moves the pen to a graphics coordinate given in the form X,Y where X and Y are the horizontal and vertical distances from the current origin in the range −2048 to 2047. The pen is lifted during the movement.

EXAMPLE: See LLINE.

GOSUB..RETURN

FORMAT: 1. GOSUB <line #/label>
:
RETURN

Abbreviation: GOS. RE.
See Also: GOTO, ON..GOSUB, ON..GOTO

PURPOSE: The GOSUB and RETURN statements are used to jump to and return from BASIC subroutines.

REMARKS: A subroutine is a group of program lines or a program 'module' designed to do a specific function repeatedly at different times within the program. Instead of repeating those lines each time the function is needed, the GOSUB statement causes execution to jump to the line on which the subroutine starts specified by line number or label. This is typically at the end of the main program, after the END statement, where it will not be encountered during normal execution. The last statement in the subroutine must be a RETURN statement. The statements in the subroutine are executed and when the RETURN statement is encountered, execution jumps back to the line in the main program following the GOSUB line. The same subroutine may be called any number of times from different GOSUBs, and one subroutine may be called from within another.

EXAMPLE:

```
5:WAIT 150
10:GOSUB 70
20:PRINT "RETURNED FROM MAIN SUBROUTINE"
30:GOSUB 70
40:PRINT "RETURNED THE SECOND TIME!"
50:END
60:REM SUBROUTINE STARTS HERE AFTER MAIN PROGRAM
70:PRINT "MAIN SUBROUTINE ON LINE 70"
80:GOSUB 100
90:RETURN
100:PRINT "NESTED SUBROUTINE ON LINE 100"
110:PRINT "RETURNING TO MAIN SUBROUTINE"
120:RETURN
```

```
>RUN
MAIN SUBROUTINE ON LINE 70
NESTED SUBROUTINE ON LINE 100
RETURNING TO MAIN SUBROUTINE
RETURNED FROM MAIN SUBROUTINE
MAIN SUBROUTINE ON LINE 70
NESTED SUBROUTINE ON LINE 100
RETURNING TO MAIN SUBROUTINE
RETURNED THE SECOND TIME!
>
```

[5] Sets wait time for display printout.
[10] Calls the main subroutine for the first time.
[30] Calls the main subroutine for the second time.
[70–90] The main subroutine called by the main program.
[100–120] The nested subroutine called by the main subroutine.

GOTO

FORMAT: 1. GOTO <line #/label>

Abbreviation: G.
See Also: GOSUB..RETURN, ON..GOTO

PURPOSE: Transfers execution to the specified line number unconditionally.

REMARKS: The GOTO statement is used to make unconditional jumps from one point in a program to another. Unlike the GOSUB..RETURN statement, the GOTO statement does not 'remember' the location from which execution jumped. If the statement on the line specified is not executable (i.e., a DATA statement or a REM line) execution starts on the first executable line after that. If a non-existent line is specified, an error code is generated. GOTO can also be used to continue execution after a BREAK has occurred (see CONT). GOTO alone starts execution from the first line in the program.

EXAMPLE:

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

Prints YES if a Y is entered from the keyboard, and prints NO if anything else is entered.

PRO
RUN
PROGRAM

GPRINT

FORMAT: 1. GPRINT [SET/OR/XOR,] <bit-image value>;<bit-image value>;.....
2. GPRINT [SET/OR/XOR,] "<hex. bit-image string>"
3. GPRINT

Abbreviation: GP.
See Also: GCURSOR

PURPOSE: Draws graphics patterns on the screen.

REMARKS: GPRINT displays a column or series of columns of bit-image data on the screen from the current position of the graphics cursor. The bit image data may be specified in three ways; as a list of decimal values or hexadecimal values separated by semicolons, or as a single hexadecimal string. In each of these, each data item is converted to the equivalent 8-bit binary number, and each of the eight bits is used to either turn a dot on (dark) or off (light). So one item of bit-image data sets the eight dots in a single column on the screen whose height is equal to one character line. A list or string of bit-image data items will set the dots along a line on the screen whose height is the same as a line in text mode:

Example 1:

x – bit 1 = 1
x – bit 2 = 1
x – bit 3 = 1
x – bit 4 = 1
x – bit 5 = 1
x – bit 6 = 1
x – bit 7 = 1
x – bit 8 = 1

bit 8 bit 1
↓ ↓
Bit image data = 11111111 in binary
= 255 decimal
= FF hexadecimal

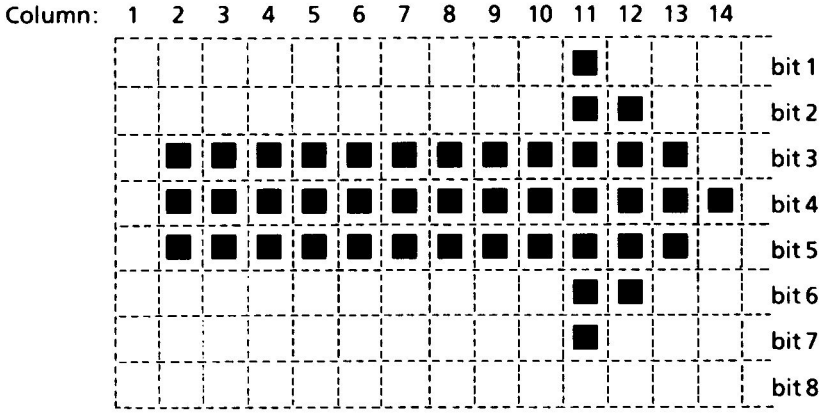
Example 2:

– bit 1 = 0
x – bit 2 = 1
x – bit 3 = 1
– bit 4 = 0
x – bit 5 = 1
– bit 6 = 0
x – bit 7 = 1
x – bit 8 = 1

bit 8 bit 1
↓ ↓
Bit image data = 11010110 in binary
= 214 decimal
= D6 hexadecimal

If the value of the bit-image data item is given in hexadecimal, it must be preceded by an "&" sign (e.g., &D6, &FF) except where specified as a hexadecimal string in format 3. In this form, the range is from &00 to &FF. If the string contains an odd number of characters, the last character in the string will be ignored.

Example 3:



The graphic symbol above can be displayed on the screen by outputting 14 columns of bit-image data as follows:

Column 1 contains no dots, so the data is 00000000 = &0.
Bit image data is 00011100 for columns 2 to 10 = &1C
01111111 for columns 11 = &7F
00111110 for columns 12 = &3E
00011100 for columns 13 = &1C
and 00001000 for columns 14 = &08
So the following statement will print the large graphics arrow to the screen:

```
GPRINT &0; &1C; &1C; &1C; &1C; &1C; &1C; &1C; &1C; &7F;
&3E; &1C; &08
```

The following statements specify the same dot patterns in formats 1 and 2:

```
GPRINT 16;40;18;253;18;40;16              (Format 1 decimal)
GPRINT &10;&28;&12;&FD;&12;&28;&10        (Format 1 hexadecimal)
GPRINT "102812FD122810"                   (Format 2 hex. string)
```

The SET, OR and XOR parameters are specified as follows:

SET — The dot pattern is set as specified in the bit-image.
OR — The dot settings specified in the bit-image will be OR-ed with the existing dot settings at that position on the screen (see description of OR operator).
XOR — The dot setting specified in the bit-image will be exclusive OR-ed with the existing dot settings at that position on the screen.

The SET option is the default.

GPRINT alone will move the graphics cursor down a line without clearing any existing display on that line.

EXAMPLE:

```
10:CLS
20:FOR I=1TO 77
30:GPRINT 255;214;
40:NEXT I
50:END
```

[10] Clears the screen.
[20] GPRINT defines two 8-bit image values and each time line 30 is executed, an 8 by 2 (height X width) area is displayed on the screen. The screen width is 156 bits or dots, thus running this program will fill a single line on the screen except for the last 2 8-bit columns. Notice that line 20 ends in a semicolon. This is important to stop carriage returns from being added after each GPRINT execution that would make the pattern be printed on 77 different lines.
[30] The same patterns described above in the command examples 1 and 2 are repeated 77 times.

GRAD

FORMAT: 1. GRAD

Abbreviation: GR.
See Also: DEGREE, RADIAN

PURPOSE: Sets the computer to gradient mode.

REMARKS: In gradient mode, GRAD is displayed on the top line of the screen. Input to the arguments of SIN, COS and TAN must be as a gradient value. Values returned by the ASN, ACS and ATN functions will be gradient values.

EXAMPLE:

```
10:GRAD
20:PRINT "TRIG FUNCTIONS NOW IN GRADIENT VALUES"
30:PRINT ASN(0.5),ASN(1.0)
40:PRINT ACS(0.5),ACS(1.0)
50:PRINT ATN(0.5),ATN(1.0)
60:END
```

Try running this program and compare the results with those for DEGREE and RADIAN.

[10] Selects GRAD mode.
[30–50] Prints out some sample values.

GRAPH

FORMAT: 1. GRAPH

Abbreviation: GRAP.
See Also: TEXT

PURPOSE: Sets the printer in graphics mode.

REMARKS: When GRAPH is executed, the PAPER command print range parameters <limit from> and <limit to> are reset to their default values.

EXAMPLE:

```
>GRAPH
```

The computer selects GRAPH mode for the printer.

HEX$

PRO
RUN
PROGRAM

FORMAT: 1. HEX$(X)
2. HEX$(&X)

Abbreviation: H.
See Also: VAL

PURPOSE: Returns a character string representing the value of a decimal expression X, or a hexadecimal expression &X.

REMARKS: The expression X specified must evaluate to a decimal number in the range 0 to 65535. The character string returned will be in the range &0 to &FFFF. Non-integer numbers will be rounded to the nearest integer before returning the string. To convert a hexadecimal number to its equivalent string, use "&" before the value.

The expression X (or &X) must be a simple variable or a number; it cannot be an algebraic expression containing an arithmetic operator or trigonometric function. To evaluate a complex expression, for instance HEX$ (SIN A*B), first evaluate SIN A*B, and assign the result to a simple variable, say C, then convert C to a string with HEX$(C).

EXAMPLE:

```
10:PRINT "DECIMAL TO HEX CONVERSION"
20:INPUT "TYPE IN DECIMAL NUMBER ",X
30:IF X>65535THEN 100
40:IF X<0THEN 110
50:D$=HEX$(X)
60:PRINT "HEX EQUIVALENT OF ";X;" IS ";D$
70:INPUT "ANOTHER? ENTER Y OR N ",A$
80:IF A$="Y"THEN 20
90:GOTO 120
100:PRINT "NUMBER IS TOO LARGE":GOTO 20
110:PRINT "NUMBER MUST BE +VE":GOTO 20
120:END
```

IF..THEN

FORMAT: 1. IF <condition> THEN <line #/label/statement>
2. IF <condition> THEN <line #/label/statement>
ELSE <line #/label/statement>

Abbreviation: IF ~ T. ~ EL.
See Also:

PURPOSE: Changes the flow of program execution according to the results of the specified condition at the time the program is run.

REMARKS: The IF..THEN pair allow decisions to be made within a BASIC program. The THEN clause following the IF condition is executed if the result of the condition is true. If false, the THEN clause is ignored and the ELSE clause is executed. If there is no ELSE clause, execution continues with the next line. <condition> can be a simple equality such as X=1 or a more complex logical expression. The THEN or ELSE clause can specify any BASIC statement (usually a GOTO jump or assignment).

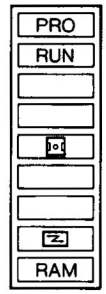
IF..THEN..ELSE statements may be nested on a program line up to the maximum program line length (80 characters).

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
```

The program continues to ask Continue? as long as YES is entered; it stops if NO is entered and complains otherwise.

INIT



FORMAT: 1. INIT "Sn:" , {"F" / "M" / "P"}
2. INIT "X:"
3. INIT "COMn:", <buffer size>

Abbreviation: INI.
See Also: SETCOM, SETDEV, TITLE

PURPOSE:
1. Initializes the modules in expansion slots 1 or 2 and specifies the usage.
2. Initializes and formats a blank floppy disk.
3. Sets the size of the receive buffer for the two serial ports.

REMARKS: Format 1:
Sn: initializes the RAM disk module in expansion slot S1: (slot 1) or S2: (slot 2).

This format is only effective in MODE 0
The module in the specified expansion slot can be initialized as a RAM disk, program memory, or memory expansion as follows:

Parameter	Usage
"F"	Formats the module as a RAM disk. This allows programs and files to be stored in the same way as on floppy disk. See Chapter 11; Files for more details on storing files on RAM disk.
"M"	Initializes the module as an expansion of the PC-1600's internal user area. This enables larger sized programs to be run.
"P"	Initializes the module as a program storage area. A single program can be written to the module and will be preserved there by the back-up battery even when the module is removed from the computer.

The INIT command cannot be executed on a module which already holds programs or files, or one with the write-protect switch set to ON.

NOTE:
The INIT command cannot be executed in the following cases:

1) On a RAM module which already holds programs or files. First clear the files with the KILL command, or delete programs by executing NEW on that module.

2) On a RAM module with the write-protect switch set ON. First set the write-protect switch OFF.

3) On a program module which has been selected with the TITLE command.

Refer to page 43; RAM Module Expansion, for a list of RAM modules available.

When initializing a CE-159 (8K) module, all of the 8K of memory must either be allocated as program memory or as expansion memory. Even if you specify only 4K or 6K of the total memory for read-only use with the select switch on the module, the remaining memory cannot be used as expansion memory.

Format 2:
INIT "X:" will initialize and format a blank floppy disk. All new disks must be formatted with this command before use. If a disk has some information stored on it, executing the INIT command will effectively erase ALL the contents.

Format 3:
The size of the receive buffer for a serial port can be set in the range 80 to 16383 bytes, or set to a minimum size of 40 bytes by specifying 0 as the buffer size. The following ports can be specified with COMn:

COM1:	RS-232C serial port
COM2:	Optical serial port
COM:	Port currently selected with the SETDEV statement.

The value of <buffer size> is reset to 0 (40 bytes) at power on or after a reset operation.

If files are being exchanged between two computers which are hard-wired together (that is with no modem or telephone line involved), a large receive buffer will speed up file transfer. On the other hand, if the communications functions are not being used, setting the size of the buffer to a minimum (0=40 bytes) will free up memory space for other purposes. A typical average setting for the receive buffer size is 256 bytes.

If there is insufficient memory available for the specified buffer size, or if a file is currently open under the APPEND option of the OPEN statement, an error code is generated.

EXAMPLE:

```
>INIT"S1:","F"
```

Initializes the module in slot 1 as a RAM disk module.

```
>INIT"X:"
```

Initializes the floppy disk in the floppy disk drive by formatting it. Any existing data on the floppy is lost.

PROGRAM

INKEY$

FORMAT: 1. <string variable>=INKEY$
2. <string variable>=INKEY$(0)
3. <string variable>=INKEY$(1)

Abbreviation: INK.
See Also:

PURPOSE: Checks the keyboard buffer during program execution, and reads a single key code into the program.

REMARKS: INKEY$ and INKEY$(0) have the same meaning. INKEY$, or INKEY$(0), reads the latest key character from the keyboard buffer and assigns it to <string variable>. If no key has been pressed, INKEY$ returns a null string (ASCII code 0). If the keyboard buffer contains several key entries, INKEY$(1) can be used to read from the oldest. The character read from the keyboard buffer is not displayed as a character on the screen.

Only the characters listed in the table can be read in with the INKEY$ command:

INKEY$ Character Table

High / Low	0	1	2	3	4	5	6
0			SPACE	0		P	
1	SHIFT	F1		1	A	Q	
2	SML	F2		2	B	R	
3	CTRL	F3		3	C	S	
4	KBD/CLICK	F4		4	D	T	
5	BS	F5		5	E	U	
6		F6		6	F	V	
7				7	G	W	
8	◀	CL	(	8	H	X	
9	◆	RCL	)	9	I	Y	
A	↓		*		J	Z	
B	↑	DEF	+		K		
C	▶				L		
D	ENTER		−	=	M		
E	ON		·		N		
F	OFF	MODE	/		O		

EXAMPLE:

```
:
:
300:A$=INKEY$
310:IF A$=""THEN 300
320:IF A$="Y"THEN 500
330:GOTO 300
:
:
500:PRINT "YES........."
```

This part of the program is similar to using the INPUT statement. The program is waiting for the user to hit a key. The difference is that unlike INPUT, the INKEY$ statement does not use the screen for input. The program keeps cycling until a key is hit.

[300] Reads a character from the keyboard buffer and assigns it to the string variable A$.
[310] Keeps checking the keyboard buffer until a key is input.
[320] Jumps to line 500 if "Y" is input.
[330] Jumps back to line 300 if anything apart from "Y" is input.

PRO
RUN
PROGRAM

INP

FORMAT: 1. INP (<port address>)

Abbreviation:
See Also: OUT

PURPOSE: Returns a value corresponding to the input to the Z-80A microprocessor's port directly.

REMARKS: INP returns the value of a bit pattern entered directly to a machine input port.

<port address> specifies the machine input port as a 16-bit address in the range 0 to 65535 (&0 to &FFFF) and returns one byte of data from the port address.

EXAMPLE:

```
:
:
300:A=INP (20)
:
:
```

Returns the value input to machine port address 20 (&14).

INPUT

FORMAT: 1. INPUT ["<message string>" { ; , }]<list of variables>

2. INPUT "<message string>" { ; , } <variable>,"<message string>" { ; , } <variable>,...

Abbreviation: I.
See Also:

PURPOSE: Allows values to be input from the keyboard and entered in program variables.

REMARKS: When INPUT is encountered, execution pauses and displays the message string, if any. Data can then be entered at the keyboard, or through a serial I/O port if so designated with SETDEV, and will be input and assigned to the variable name(s) specified in <list of variables>. Variables in <list of variables> must be separated by commas. The values entered at the keyboard must be entered one by one separated by ENTER.

If there is no message string, a question mark is displayed to prompt the user to enter data. If the message string is followed by a semicolon, no question mark is displayed, and the cursor is positioned after the last character in the message string. If the message string is followed by a comma, the cursor moves to the front of the line below the message string and waits for data entry.

Data can also be input from the RS-232C port with the INPUT command in the following format:

INPUT <variable>[,<variable>] ....

No message string can be specified and no question mark is displayed.

When using the INPUT command to input data from the RS-232C serial port attached to the CE-158 Serial Interface Unit, only fixed and simple variables are allowed. Array variables cannot be used.

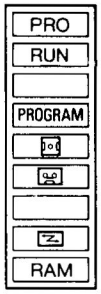
EXAMPLE:

```
10:PRINT "VOLUME OF SOLID"
20:INPUT "ENTER L,B,H ",L,B,H
30:V=L*B*H
40:PRINT "VOLUME IS ";V
50:END

>RUN
VOLUME OF SOLID
ENTER L,B,H
3 ENTER
4 ENTER
5 ENTER
VOLUME IS 60
>
```

[20] Asks the user to input three numeric values. The message here will appear on the screen when the program is run. In the RUN example, the user enters 3, 4 and 5 as the values.
[30–40] Calculates the volume and prints out the answer.

INPUT#



FORMAT: 1. INPUT#<file #>,<variable list>
2. INPUT#["<filename>";]<variable list>

Abbreviation: I.#
See Also: DIM, INPUT, OPEN, PRINT#, SET

PURPOSE: Used to read items from a sequential file on floppy disk, RAM disk module or cassette tape.

REMARKS: 1. Disk or RAM disk module
<file #> is the number allocated to the file when it was opened for input with the OPEN statement. An attempt to input items from a file which has not been opened will generate an error code.

2. Cassette tape files
The default value when "<filename>" is not specified for the cassette tape is the first file from the current position of the tape head.

<variable list> specifies the names of the variables into which the data items will be written when the INPUT# statement is executed. The list can contain fixed variables, simple variables and array variables delimited by commas. The order in which the variables are specified in the list must correspond to the order in which the data occurs in the file; each variable type must have a matching data type. String variables must also be of sufficient length for the data items to be read in. If arrays are read in with the INPUT# statement, the necessary memory space must have been declared in the program with DIM.

When numeric data items in a sequential data file are read into numeric variables, a comma, a space (&20) or CR+LF (&0D0A) will be interpreted as a data delimiter. Any extra spaces preceding data items are ignored.

When character strings in a sequential data file are read into string variables, a comma, or CR+LF (&0D0A) will be interpreted as a data delimiter. Spaces preceding character strings are ignored. If a single pair of quotation marks (") are used inside a character string, the quotation marks and all subsequent characters in that data item will be ignored. Character string data which include commas can be read in by enclosing the whole string in a pair of quotation marks. The comma will then be read in as part of the data item.

If the number of data items being read in from the file is less than the number of variables in the list, execution will halt in the waiting state. Terminate this state by pressing the BREAK key. If the number of data items in the file is more than the number of variables in the list, excess data items will not be read.

In format 2 for cassette tape files, array variables must be specified in the form A(*), using an asterisk for the element. Single elements of an array cannot be specified. This form cannot be used in format 1.

EXAMPLE: See OPEN.

INSTAT

PRO
RUN
PROGRAM

FORMAT: 1. INSTAT "COM1:"

Abbreviation: INSTA.
See Also: OUTSTAT

PURPOSE: Returns the settings of the control signals for the RS-232C serial port.

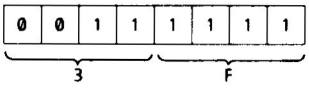
REMARKS: The settings of the control signals are returned in the form of an 8-bit binary number. Each bit represents the value of a control signal as follows:

Bit	Value	Signal condition
1	0	DTR is high
	1	DTR is low
2	0	RTS is high
	1	RTS is low
3	0	CTS is high
	1	CTS is low
4	0	CD is high
	1	CD is low
5	0	DSR is high
	1	DSR is low
6	0	CI is high
	1	CI is low
7	always 0	
8	always 0	

EXAMPLE:

```
>INSTAT"COM1:"
 63
```

Returns the settings of the control signals for the RS-232C serial port. Here, 63 decimal refers to 3F hex and the following bit pattern:



All six signals are thus set low (bit 1 is the leftmost bit), the default condition.

PRO
RUN
PROGRAM

INSTR

FORMAT: 1. INSTR([<col>,]X$,Y$)
2. INSTR([<col>,]"<string>","<char>")

Abbreviation: INS.
See Also:

PURPOSE: Searches string X$ to find the first occurrence of character Y$ and returns the position as an integer.

REMARKS: If Y$ cannot be found, or if X$ is a null string, the value 0 is returned. In format 2, the actual character is specified with an actual string, both enclosed in quotes. <col> is the starting column for the search. The default is column 1.

EXAMPLE:

```
10:INPUT "ENTER A WORD ";A$
20:N=INSTR (A$,"A")
30:IF N=0THEN 60
40:PRINT "THE FIRST `A` WAS LETTER #";N
50:GOTO 70
60:PRINT "NO LETTER `A` IN THE WORD"
70:END

>RUN
ENTER A WORD DATA
THE FIRST LETTER `A` WAS LETTER # 2
>
```

This program searches for 'A' in a word.

[10] The user keys in any word.
[20] The INSTR statement looks for any 'A's in the word.
[40–50] The location of the first 'A' is printed out.
[60] No 'A' was found in the word.

INT

FORMAT: 1. INT(X)

Abbreviation:
See Also:

PURPOSE: Truncates the decimal part of a real number and returns an integer (whole number).

REMARKS: The value X is rounded down to an integer. The integer result is always less than or equal to X regardless of whether X is positive or negative. For negative values, the absolute integer result will thus be greater than or equal to the absolute value.

EXAMPLE:

```
5:WAIT 60
10:A=-3.3:PRINT A,INT(A)
20:B=-1.9:PRINT B,INT(B)
30:C=-0.5:PRINT C,INT(C)
40:D=0.2:PRINT D,INT(D)
50:E=1.6:PRINT E,INT(E)
60:F=3:PRINT F,INT(F)

>RUN
-3.3          -4
-1.9          -2
-0.5          -1
 0.2           0
 1.6           1
 3             3
>
```

PRO
RUN
PROGRAM

KBUFF$

FORMAT: 1. KBUFF$=<string of characters>

Abbreviation: KB.
See Also: INKEY$

PURPOSE: Writes a string of characters into the keyboard buffer.

REMARKS: The KBUFF$ command can be used to run the computer in "batch" mode. System commands can be entered into a command file in the order in which they are to be executed, and written into the keyboard buffer with KBUFF$. This causes the commands to be executed as if they had been entered from the keyboard directly.

Command files or batch files can be used to execute a series of commands for a certain task automatically. For instance, a command file can be written to clear memory, load a specific program, execute the program, and write the results back to a file.

The character string can be up to 32 characters long. The string will overwrite any existing contents of the buffer.

EXAMPLE:

```
:
:
200:KBUFF$ ="45"+CHR$ (&0D)
210:INPUT A
220:PRINT "A IS";A
:
:
```

[200] Inserts 45 into the keyboard buffer as if the user had typed in the number at program run time.
[210] INPUT waits for the user to key in a number but this has been done already by use of the KBUFF$ command. Notice that there is no carriage return entered with the KBUFF$ string. When the program is run, 45 will appear automatically on the screen but the user will still have to hit the enter key. See CHR$ and the ASCII code tables for inserting non-printable key characters into string variables.
[220] Prints out the value of A.

KEY ON/OFF/STOP

PRO
RUN
PROGRAM

FORMAT: 1. KEY(<key #>)ON
2. KEY(<key #>)OFF
3. KEY(<key #>)STOP

Abbreviation:
See Also: ON KEY GOSUB

PURPOSE: Enables or disables the specified function key used to cause program branching at run time.

REMARKS: This statement controls the action of the ON KEY GOSUB statement.

KEY(<key #>) OFF disables one of the function keys F1 to F6 specified by <key #> in the range 1 to 6. Pressing that key will have no effect on an ON KEY GOSUB statement after this statement has been executed. KEY(<key #>) ON enables a function key previously disabled with the KEY(<key #>) OFF/STOP command. These two commands can be issued from within a program to prevent program interruptions.

KEY(<key #>) STOP disables a function key. However, if a KEY ON statement is executed at a later time, if the function key has been pressed during the period in which KEY OFF was in effect, execution will immediately jump according to the ON KEY GOSUB statement. The default value is KEY STOP.

A key which is assigned in a ON KEY GOSUB statement cannot be used as a normal function key to enter predetermined strings or commands.

EXAMPLE:

```
10:KEY(3) OFF
:
:
:
390:KEY(3) ON
```

[10] Disables the F3 function key during program execution.
[390] Re-enables the F3 function key for normal use.

PRO
RUN
PROGRAM

KEYSTAT

FORMAT: 1. KEYSTAT,[<parameter 1>][,<parameter 2>]

Abbreviation: KE.
See Also: INKEY$

PURPOSE: For key input control, specifies the standard input device, switches the key repeat function on/off, and switches the key click on/off.

REMARKS: <parameter 1> switches the key repeat function on/off:

0 — key repeat OFF
1 — key repeat ON

<parameter 2> switches the key click on/off:

0 — key click OFF
1 — key click ON

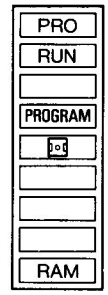
At power on, the default setting always preserves the last set values. After an ALL RESET, the default setting is KEYSTAT, **0**,**0**.

EXAMPLE:

```
>KEYSTAT,1,1
```

Switches both the key repeat function and the key click ON.

KILL



FORMAT: 1. KILL "<d:filename>"

Abbreviation: K.
See Also: SAVE, SET

PURPOSE: Used to delete a file on floppy disk or RAM disk module.

REMARKS: <d:filename> specifies the device and filename. If the file is a BASIC file, the .BAS extension must be specified with the file name. If the file has been protected with the "P" attribute of the SET statement, or if the RAM module's write protect switch has been set on, an error code is generated. A file which is currently OPEN cannot be deleted until it has been CLOSED.

EXAMPLE:

```
>KILL"S1:OLDATA"
>
```

Deletes the file OLDATA held in the RAM disk module in expansion slot 1.

LCURSOR

FORMAT: 1. LCURSOR <column>

Abbreviation: LC.
See Also: GLCURSOR, PCONSOLE, TAB

PURPOSE: In TEXT mode, moves the printer pen to the specified column.

REMARKS: The LCURSOR command is similar to the GLCURSOR command, but moves the pen to the specified print column in the range Øto maximum line length set with the current PCONSOLE statement.

EXAMPLE:

```
10:LCURSOR 40
20:LPRINT "MIDDLE"
30:LPRINT "LEFT"
40:END
```

This program will only run in TEXT mode.

[10] Moves the pen carriage to the middle of the paper.
[20] This message will start printing from the current printer position in the middle of the page.
[30] After the last LPRINT, the printer head moves back to the left. Thus, this line prints in the normal left justified position.

LEFT$

PRO
RUN
PROGRAM

FORMAT: 1. LEFT$(X$,N)
2. LEFT$("<string>",N)

Abbreviation: LEF.
See Also: MID$, RIGHT$

PURPOSE: Returns N characters from the left end of any string, X$.

REMARKS: The value of N must be in the range 0 to 80. Fractions will be rounded down (truncated). If N is less than 1, a null string is returned. If N is greater than the number of characters in X$, the whole string is returned.

EXAMPLE:

```
10:X$="SHARP"
20:FOR N=1TO 6
30:LET S$=LEFT$ (X$,N)
40:PRINT S$
50:NEXT N

>RUN
S
SH
SHA
SHAR
SHARP
SHARP
>
```

PRO
RUN
PROGRAM

LEN

FORMAT: 1. LEN(X$)
2. LEN("<string>")

Abbreviation:
See Also:

PURPOSE: Returns the number of characters in string X$.

REMARKS: The number of characters in the string includes any blanks or non-printing 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] Inputs a word. In this example, the user enters "CHERRY".
[20] Finds the length of the word.
[30] Prints out the answer.

LET

PRO
RUN
PROGRAM

FORMAT: 1. [LET]<variable>=<expression>
[,<variable>=<expression>].....

Abbreviation: LE.
See Also:

PURPOSE: Assigns a value to a variable

REMARKS: The word [LET] is optional except in statements included in a THEN or ELSE clause in the IF..THEN..ELSE statement. The type of expression must match the type of variable. Only numeric expressions can be assigned to numeric variables and only string expressions can be assigned to string variables.

EXAMPLE:

```
10:LET X$="FIRST STRING"
20:Y$="SECOND STRING"
30:PRINT X$:PRINT Y$
40:LET A=6
50:B=A+1
60:LET R$=LEFT$ (X$,A)
70:S$=LEFT$ (Y$,B)
80:PRINT R$;S$
90:END
```

LF

FORMAT: 1. LF [<lines>]

Abbreviation:
See Also: CSIZE, PITCH, PAPER

PURPOSE: Feeds the paper in the printer the specified number of lines.

REMARKS: In TEXT mode, LF on its own feeds the paper one line each time it is executed. <lines> specifies the number of lines in the range set with the current PAPER statement. If <lines> is positive, the paper will be fed forward, and if negative, the paper will be fed in reverse.

The line height effective for the LF command depends on the character size selected with CSIZE.

EXAMPLE:

```
>LF 3
```

Advances the paper three lines.

LFILES

PRO
RUN
PROGRAM
RAM

FORMAT: 1. LFILES "<d:>"
2. LFILES "<d:filename>"
3. LFILES "<d:ambiguous filename>"

Abbreviation: LF.
See Also: FILES, SETDEV

REMARKS: Lists directory information on the files specified for the device specified to the printer or to one of the serial ports.

REMARKS: Options for the LFILES command are specified in the same way as the FILES command but the directory information is sent to the specified serial port or printer instead of to the screen.

The port used will be the port specified in the currently active SETDEV command, either COM1: (RS-232C port) or COM2: (optical SIO) port. If SETDEV has not been used, the default device is the printer.

On the CE-1600P Printer/Cassette Interface Unit, LFILES will print in character size 2, regardless of the CSIZE setting.

EXAMPLE: See FILES.

PRO
RUN
PROGRAM

LINE

FORMAT: 1. LINE [(X1,Y1)] – (X2,Y2)[,<dot toggle>] [,<pattern>] {[,B] / [,BF]}

Abbreviation: LIN.
See Also: LLINE

PURPOSE: Draws a line between two points on the graphics screen.

REMARKS: The LINE command draws a line from the point specified by the coordinates X1,Y1, or from the current position of the graphics cursor if X1 and Y1 are not specified, to the point specified by the coordinates X2,Y2. The coordinates are relative to the home position at top left of the screen (0,0). The optional <dot toggle> parameter changes the dot pattern specified in <pattern> as follows:

<dot toggle>	Effect
S	Specifies that bits set to 1 in the <pattern> parameter set dots ON (black), and bit set to 0 set dots OFF.
R	Specifies that bits set to 1 in the <pattern> parameter set dots OFF. Bits set to 0 in the <pattern> parameter have no effect on dots on the screen. This is the reverse of the S option and can be used to draw a line in inverse video on a dark screen (all dots ON).
X	Inverts the state of all the dots along the line.

The default toggle is S.

The optional <pattern> parameter specifies the dot pattern used to draw the line as a number in the range 0 to 65535. The decimal value is then converted to a 16-bit binary number, each of the bits representing the presence (1) or absence (0) of a dot at that point. The value may be specified as a hexadecimal number preceded by the "&" sign in the range &0000 to &FFFF.

Example 1: <pattern> = 65535 decimal or &FFFF hex.
Binary value = 1111111111111111
Line pattern: xxxxxxxxxxxxxxxx — this is a solid line.

Example 2: <pattern> = 43690 decimal or &AAAA hex.
Binary value = 1010101010101010
Line pattern: x x x x x x x x — this is a dotted line.

Example 3: <pattern> = 26214 decimal or &6666 hex.
Binary value = 0110011001100110
Line pattern: xx xx xx xx — this is a dashed line.

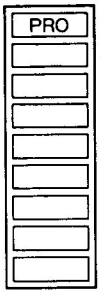
The default pattern is a solid line. The B option draws a box on the line as diagonal. The BF option draws the box and fills it in.

NOTE:
The X and Y coordinates for the LINE command can be specified in the range −32768 to 32767. But only coordinates in the range 0 to 155 for X and 0 to 31 for Y will display on the screen.

EXAMPLE:

```
10:CLS
20:FOR N=10TO 100STEP 30
30:M=N+20
40:LINE (N,10)-(M,20),,,BF
50:NEXT N
60 END
```

[10] Clears the screen.
[20] Loop to repeat the drawing of each box. Four boxes will appear in a row across the screen.
[30] Draws a filled rectangular box.



LIST

FORMAT: 1. LIST
2. LIST <line #>

Abbreviation: L.
See Also: LLIST

PURPOSE: Lists all or part of a program in memory on the display.

REMARKS: Executing LIST without specifying a line number will list out one screenful of the current program starting at the lowest line number. The ▄ key will scroll up the next line in the listing.

LIST <line #> displays the specified program line on the screen. If the line number specified is larger than any existing line number, an error code will be generated.

EXAMPLE:

```
>LIST
```

Displays program lines in order.

```
>LIST 100
```

Displays program line 100.

LLINE

PRO
RUN
PROGRAM

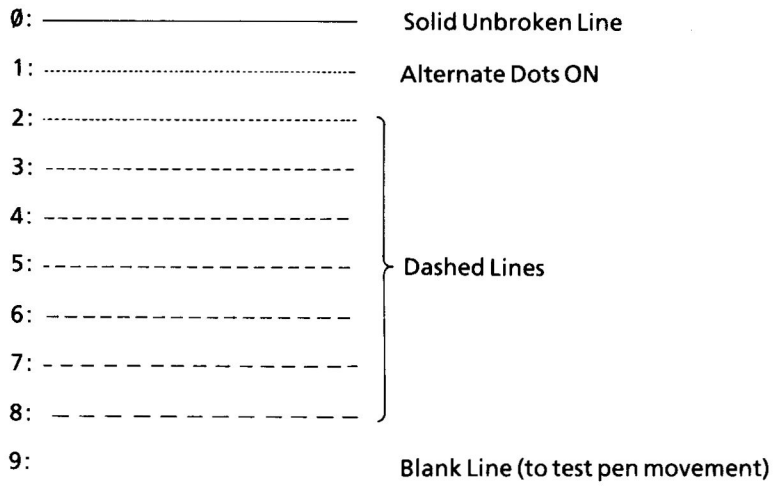
FORMAT: 1. LLINE [(X1,Y1)] – (X2, Y2)[– (X3,Y3)..], [<type>], [<color>][,B]

Abbreviation: LLIN.
See Also: COLOR, RLINE, SORGN

PURPOSE: Draws a line or a series of line segments between points specified in absolute coordinates on the printer.

REMARKS: In graphics mode, the LLINE command draws a line segment from the point specified by the coordinates X1,Y1 or from the current pen position if X1 and Y1 are not specified, to the point specified by the coordinates X2,Y2. X and Y must be in the range –2048 to 2047. The coordinates are absolute; that is they are relative to the point set as origin (0,0) with the current SORGN statement. Five subsequent contiguous line segments can be specified.

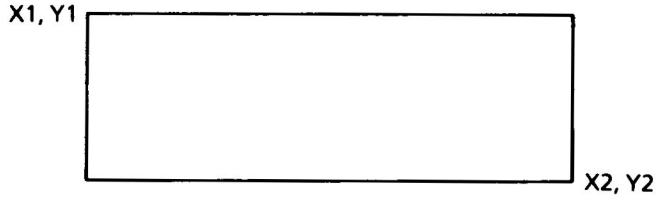
The optional <type> parameter specifies the line type in the range 0 to 9.



The optional <color> parameter specifies the pen color in the range 0 to 3 (see COLOR command). If <type> and <color> are not specified, the currently set values will be used. When executing LLINE or RLINE immediately following an LPRINT statement in graphics mode, the <type> parameter must be specified; the default line type does not work in these cases. This also applies to programs transferred from the PC-1500 involving the PC-1500's LINE statement.

The B option specifies a rectangle to be drawn on the diagonal specified by X1,Y1 and X2,Y2 as below:

For the command LLINE (X1,Y1) – (X2,Y2),,,B



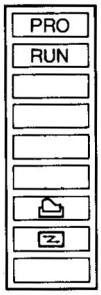
EXAMPLE:

```
10:GRAPH
20:GLCURSOR (40,40)
30:SORGN
40:LLINE -(100,0)-(0,100)-(0,0)
50:TEXT
60:END
```

This will draw a triangle away from the corner of the page.

[10] Sets the machine to graph mode.
[20] Moves the printer pen to the specified position away from the corner of the page.
[30] Resets the origin using the current pen position as position 0,0. All subsequent drawing will use this coordinate system.
[40] Draws the triangle in three movements ending back at the origin.
[50] Now that the drawing is finished, returns to text mode.

LLIST / LLIST*



FORMAT: 1. LLIST[*]
2. LLIST[*]<line #>
3. LLIST[*]<line #>,<line #>
4. LLIST[*]<line #>,
5. LLIST[*]<,line #>

Abbreviation: LL.
See Also: LIST

PURPOSE: Lists program lines in memory to the optional printer or to the serial I/O interface if it has been OPENed.

REMARKS: The LLIST command is used in the same way as LIST, but output is directed to the printer or serial I/O interface instead of the screen. It is more flexible than the LIST command.

To the printer:
LLIST with no parameter lists out the whole program.
LLIST <line #> lists out only the specified program line.
LLIST <line #>,<line #> lists out program lines starting at the first line number specified and finishing at the second line number specified.
LLIST <line #>, lists out program lines from the line number specified up to the end of the program.
LLIST ,<line #> lists out program lines up to and including the specified line number.

When using LLIST to send lines to the printer, the printer output is affected by the <line length> set by the current PCONSOLE command. If <line length> is set to 18 or more, lines will be printed and continued on the next line when too long. If <line length> is set to 17 or 16, ERROR 76 will display if the line to be printed is longer than <line length> and there will be no printout.

To the Serial I/O Interface:
If the serial I/O interface has been opened for output with SETDEV using the PO parameter, the LLIST command will send program line output to the interface instead of the printer.

Each line in the program will contain the number of characters specified in the PCONSOLE command. Each line will be terminated with either a CR, LF or LF+CR as specified in the PCONSOLE command.

If a PASSword has been set, LLIST is ignored. LLIST* only lists those comment lines starting with an apostrophe ('). Line number and the apostrophe will be suppressed. Comments not starting at the head of a line and all REM comments will be ignored. This is a useful way of listing out just the descriptive information of a program held in the comment statements. If the end (80th column) of a comment line ends in a semicolon, any comment appearing on the next line will not be printed on a new line in the listing but will continue the previous line.

When using LLIST* to the serial port, program lines with 3-digit line numbers (e.g., 100 and upwards) will be truncated somewhat in the listing to the port due to limits on buffer space. For complete listings, therefore, keep line numbers in 2-digits only (up to 99).

If the current character size selected by CSIZE is 1, the program will be listed out in character size 1. For all other character sizes (2 to 9) the listing will be in character size 2. LLIST/LLIST* automatically selects text mode before listing out.

EXAMPLE:

```
>LLIST
```

Lists the whole program.

```
>LLIST 100,200
```

Lists program lines 100 to 200 inclusive.

```
>LLIST 150,
```

Lists from line 150 to the end of the program.

```
>LLIST ,40
```

Lists the program from the beginning up to and including line 40.

LN

PRO
RUN
PROGRAM

FORMAT: 1. LN(X)

Abbreviation:
See Also: EXP

PURPOSE: Returns the natural logarithm (to base e) of the expressions X.

REMARKS: The value specified for X must be greater than zero. To find the antilog of a number, use the EXP function.

EXAMPLE:

```
10:CLS
20:INPUT "TYPE IN A NUMBER ",X
30:PRINT "THE NATURAL LOGARITHM OF ";X;" IS ";LN(X)
40:INPUT "USE AGAIN? Y/N ",A$
50:IF A$="Y"THEN 20
60:END
```

PRO
RUN
PROGRAM
RAM

LOAD / LOAD*

FORMAT: 1. LOAD"<d:filename>"[,R]
2. LOAD*"<d:filename>"

Abbreviation: LOA.
See Also: CHAIN, LLIST*, MERGE, REM, RUN, SAVE

PURPOSE: Loads a file into memory from a floppy disk drive, RAM disk module, tape, or one of the serial ports.

REMARKS: <d:filename> specifies the device name and file name of the file to be loaded. The following device names can be specified.

S1:, S2:	Module slots 1 and 2
X:, Y:	Floppy disk drive
COMn:	Serial port
CAS:	Cassette tape

The specified file name must exist on the device.

If the R option is specified, after a file has been loaded into memory with LOAD, it will be executed automatically as if RUN had been entered. The R option is also used with AUTORUN files (see page 75) to allow a file to be loaded and executed automatically. If the file does not contain a BASIC program, an execution error will result. Any files which are open when LOAD is executed on another file will be closed, except when the R option is specified.

Placing an asterisk (*) after LOAD in format 2 prefixes each line with a line number and an apostrophe (') after loading into memory. This has the effect of making each line a BASIC comment line (see the REM statement). The lines are numbered from 10 in steps of 10. Using this LOAD* form allows ASCII text files to be loaded into memory as BASIC comment lines. This is the only way that raw text data can be handled by the PC-1600 at file level. The LLIST* command can then be used to list out the file. For instance, a file containing the following text data will be loaded in the form shown with the LOAD* option.

When loading a file from the serial port, a received CR+ LF (&ØDØA) will be interpreted as an end-of-line code, and &1A as the end-of-file code.

```
10 ' The LOAD* statement enables text data
20 ' to be loaded into the computer at
30 ' BASIC level and displayed as
40 ' information on the screen. This can
50 ' be used for simple memos, or address
60 ' lists, etc.
```

The LOAD* option has no use with BASIC program files, which already contain line numbers.

EXAMPLE:

```
>LOAD"BIOCALC",R
```

Loads the program in the current memory device held under the name "BIOCALC" into user memory and starts program execution.

PRO
RUN
PROGRAM
RAM

LOC

FORMAT: 1. LOC(<file number>)

Abbreviation:
See Also:

PURPOSE: Returns the number of records which have been read or written since the file was opened.

REMARKS: LOC can be used only for files on floppy disk, or in a module in slot 1 or slot 2. A LOC statement can be used to control program flow according to the number of records which have been accessed by a program since the file was opened. One record equals 256 bytes.

EXAMPLE:

```
5:MAXFILES =1
10:OPEN "X:FILE01"FOR INPUT AS #1
20:IF EOF (1)THEN 50
30:INPUT #1,N
40:GOTO 20
50:M=LOC (1)
60:PRINT "THE FILE HAS ";M;" RECORDS"
70:CLOSE #1
80:END
```

This program checks the number of records in a file on floppy disk.

[5] Allocates space for 1 file in memory.
[10] Opens the file FILE01 for input.
[20] Checks if the end of file has been reached.
[30] Inputs the next record. The file consists of just numeric records.
[40] Go and read the next record.
[50–60] The end of the file has been reached and LOC holds the number of records read
[70] Close the file.

LOCK / UNLOCK

PRO
RUN
PROGRAM

FORMAT: 1. LOCK
2. UNLOCK

Abbreviation: LOC. UN.
See Also:

PURPOSE: Disables and re-enables use of the MODE key.

REMARKS: LOCK disables the MODE key preventing accidental mode changes being made. The computer is locked in the current mode (PRO or RUN). It cannot be locked in RESERVE mode. UNLOCK re-enables the MODE key for normal use.

EXAMPLE:

```
10:LOCK
:
:
:
:
90:UNLOCK
100:END
```

[10] Disables the MODE key at the start of the program
[90] Re-enables the MODE key at the end.

PRO
RUN
PROGRAM
RAM

LOF

FORMAT: 1. LOF (<file number>)

Abbreviation:
See Also: DSKF

PURPOSE: Returns the size of a file in bytes.

REMARKS: LOF returns the size of a file on floppy disk (X:,Y:) or in modules in expansion slots 1 (S1:) or 2 (S2:) in bytes after it has been opened during program execution. <file number> is the number under which the file was opened with the OPEN command.

EXAMPLE:

```
5:MAXFILES =1
10:OPEN "X:FILE01"FOR INPUT AS #1
20:N=LOF (1)
30:PRINT "FILE01 FILE SIZE IN BYTES IS ";N
40:CLOSE #1
50:END
```

This program checks the file size for file FILE01 on floppy disk.

[5] Allocates space for 1 file in memory.
[10] Opens the file FILE01 for input.
[20–30] Finds the size of the file and prints out the value.
[40] Closes the file.

LOG

PRO
RUN
PROGRAM

FORMAT: 1. LOG(X)

Abbreviation: LO.
See Also:

PURPOSE: Returns the common logarithm (to base 10) of the expression X.

REMARKS: To obtain a logarithm to a base other than 10, use the following conversion formula: Log to base B of X = LOG(X)/LOG(B).

To obtain the antilog of a common logarithm, raise 10 to the power of the logarithm using the " ∧ " operator.

EXAMPLE:

```
>PRINT LOG(2)
           3.010299957E-01
>
```

LPRINT / LPRINT USING

FORMAT: 1. LPRINT [<list of expressions>][;]
2. LPRINT USING <format string>;<list of expressions>
3. LPRINT TAB <column>;<expression>;TAB<column>;
<expression>;TAB <.....

Abbreviation: LP.
See Also: PCONSOLE, PRINT, PZONE, TAB

PURPOSE: Used to output data to the printer or to a serial port.

REMARKS: The LPRINT and LPRINT USING statements are used in the same way as the PRINT and PRINT USING statements, but they direct the output to the printer (or to one of the serial I/O ports) instead of the screen.

LPRINT used with no parameters specified outputs a blank line to the printer; this has the same effect as a line feed. Several can be used in succession to feed paper through the printer.

LPRINT <list of expressions> prints out the values of the expressions in <list of expressions>. The expressions in the list can be numeric or string variables, or actual strings if enclosed in quotes. The exact print format depends on the delimiter used between the items in the <list of expressions>. If a semicolon (;) is used to delimit items in the list, the items are printed out immediately following one another. If a comma (,) is used to delimit data items, after printing one item, the pen moves to the next print zone differently for numeric and string data. For numeric data, if the pen is at the start of a print zone, the item is printed right-justified in that zone. If the pen is not at the start of the print zone, and for string data items, the item is left-justified in the next zone. If the print position is specified with the TAB command, printing starts from the specified tab position in all cases.

If the list of expressions is terminated by a semicolon, the next output to the printer will be continued on the same line; if the list is terminated without a semicolon, the next output will be on a new line.

The USING <format string> option is specified in the same way as for the PRINT USING statement. Refer to that description for a detailed explanation on how to specify format strings.

If the device specified in the current SETDEV statement is one of the serial ports, the LPRINT statement will output to that port instead of the printer.

When LPRINT is specified in graphics mode, lines are not terminated with CR and LF.

The TAB option specifies the actual column in which the next item is to be printed. If the TAB column is larger than the printer width set in the current PCONSOLE statement, an error code will be generated.

If the device specified in the current SETDEV statement is one of the serial ports, the LPRINT statement will output to that port instead of the printer.

When LPRINT is specified in graphics mode, lines are not terminated with CR and LF.

EXAMPLE: See PRINT USING and PZONE.

PRO
RUN
PROGRAM
RAM

MAXFILES

FORMAT: 1. MAXFILES = <number of files>

Abbreviation: MA.
See Also: CLOSE, OPEN

PURPOSE: Specifies the maximum number of files that can be open at the same time.

REMARKS: All files must be closed at the time the MAXFILES command is executed. <number of files> is **0** immediately after power on and so MAXFILES must be specified before any file can be OPENed. The maximum <number of files> is 15. If there is insufficient memory to allocate the specified number of files, an error will be generated. The size of each file allocated with MAXFILES is 313 bytes. This size will be increased as needed in a program.

EXAMPLE:

```
>MAXFILES =3
```

Allows up to three files to be concurrently open.

MEM

PRO
RUN
RESERVE
PROGRAM

FORMAT: 1. MEM

Abbreviation: M.
See Also: STATUS

PURPOSE: Returns the amount of unused memory space in the user area.

REMARKS: The amount of memory space unused is shown in bytes, including the variables storage area (same as STATUS Ø).

EXAMPLE:

```
>MEM
```

PRO
RUN

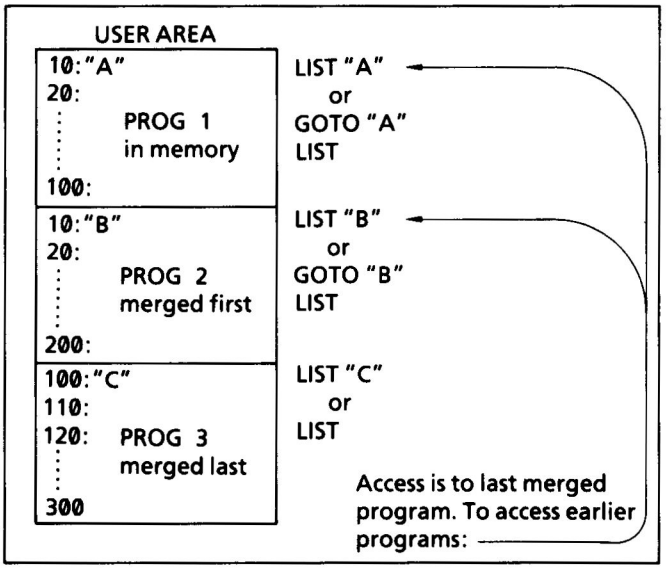
MERGE

FORMAT: 1. MERGE
2. MERGE "<filename>"

Abbreviation: MER.
See Also: CLOAD

PURPOSE: Used to load a program saved on cassette tape with another program currently in memory in PC-1500 mode.

REMARKS: Programs merged into memory may have the same line numbers as existing programs. Each program is allocated an exclusive area in memory. Only GOTO<label> or LIST<label> will move between programs. LIST will only list the last MERGEd program. To list the other programs use the <label> option with LIST. This means that all programs MERGEd into memory should have labels on their first lines.



In addition, care must be taken when MERGEing programs containing READ and DATA statements. In order to ensure that the READ statement accesses the correct DATA statements after merging, preface the DATA line with a <label>. Then insert a line before the READ statement with RESTORE <label> on it, as shown in the example below:

```
10 RESTORE "A"
20 READ X,Y,Z
30 PRINT X;Y;Z
40 "A": DATA 3,4,5
```

MERGE alone merges the next program on tape from the current tape position into memory. <filename> specifies which file held on tape is to be MERGEd.

EXAMPLE:

```
>MERGE "PROFIT"
```

Searches for program "PROFIT" on tape starting from the current tape position and loads it to the current user memory.

PRO
RUN
PROGRAM

MID$

FORMAT: 1. MID$(X$,N,M)
2. MID$("<string>",N,M)

Abbreviation: MI.
See Also: LEFT$, RIGHT$

PURPOSE: Returns a string of M characters from inside string X$ starting from the Nth character in string X$.

REMARKS: If N is less than 1 or greater than the number of characters in X$, a null string is returned. M must be in the range 0 to 80 and N in the range 1 to 80. Fractions will be rounded down (truncated).

EXAMPLE:

```
10:Z$="ABCDEFG"
20:LET Y$=MID$ (Z$,3,4)
30:PRINT Y$

>RUN
CDEF
>
```

MOD

FORMAT: 1. <n>MOD<m>

Abbreviation:
See Also: INT

PURPOSE: Returns the remainder of the division of <n> by <m> where n and m are decimal integers. If n or m are not integers, they will be rounded to the nearest integer before the division.

EXAMPLE:

```
10:INPUT "THE VALUE OF n ";N
20:INPUT "THE VALUE OF m ";M
30:A=N MOD M
40:PRINT "THE REMAINDER IS ";A
50:GOTO 10
60:END
```

[10–20] Input the parameters.
[30] Calculate the remainder.
[40] Print out the answer.
[50] Repeat the process. The user must hit the break key to stop the program.

PRO
RUN

MODE

FORMAT: 1. MODE [0]
2. MODE 1

Abbreviation: MO.
See Also:

PURPOSE: Selects the screen mode.

REMARKS: The MODE command cannot be specified on a program line; it must be entered in direct mode.

If MODE or MODE 0 is specified, all four lines on the screen are active, and the computer can be used with the CE-1600P and other peripherals for the PC-1600. The character set for the PC-1600 is selected.

MODE 1 selects PC-1500 mode. In this mode, only the bottom line on the screen is active, but the screen scrolls up line by line when in the PRO mode for program entry or editing or when using the INPUT command. The length of the line on the screen is fixed at 26 characters, and the COMPUTER can be used with PC-1500 peripherals. The character set for the PC-1500 is also selected. See Appendix C for character code tables in the two modes. It is not possible to run programs which were created on a PC-1500 in MODE0; they must be run in MODE1.

In MODE 0, PRINT statements display data on consecutive lines of the screen until the screen is full. Then the screen scroll up one line at a time. Items longer than one screen line (26 characters) are wrapped onto the next line.

In MODE 1, PRINT statements display data on the bottom line of the screen. Lines with more than 26 characters are truncated, and the truncated part is lost. Subsequent PRINT statements overwrite the existing display each time. The screen cannot be scrolled, except when entering or editing a program or with an INPUT statement.

NOTE:
After changing modes, in either mode, the prompt sign will be displayed at the left of the screen in RUN mode, and the cursor will be displayed in the home position (0, 0) without clearing the screen in PRO and RESERVE modes. In PRO mode, all four lines of the screen are active in both MODE 0 and MODE 1.

EXAMPLE:

```
>MODE
```

Selects PC-1600 mode.

```
>MODE 1
```

Selects PC-1500 mode.

PRO
RUN
PROGRAM
RAM

NAME

FORMAT: 1. NAME "<d:oldfilename>" AS "<d:newfilename>"

Abbreviation: NA.
See Also: COPY, FILES

PURPOSE: Changes the name of a file on floppy disk or RAM disk.

REMARKS: <d:oldfilename> must be the name of a file which exists on the drive specified with d:
<d:newfilename> must be a file name which is not assigned to any other file on drive d:

The same drive name must be specified for the renamed file.

File renaming is not effective in the following cases:

(1) The floppy disk in the drive specified in d: has the "write protect" notch covered.
(2) The file specified with <d:oldfilename> has been write protected with the P option of the SET command.
(3) The file specified is currently open.
(4) The write-protect switch is set to ON on the RAM disk module.

EXAMPLE:

```
>NAME "X:OLDNAM"AS "NEWNAM"
```

Changes the file name OLDNAM on floppy disk to NEWNAM.

NEW

PRO
RESERVE
RAM

FORMAT: 1. NEW
2. NEW "Sn:"[,<address>]
3. NEW <address>
4. NEW Ø

Abbreviation:
See Also: DELETE, STATUS, TITLE

PURPOSE: Deletes all program lines currently in memory, and/or assigns memory space for machine code programs.

REMARKS: If used in the RESERVE mode, it deletes all function key assignments from the reserve memory. If a program is keyed into memory before clearing the existing program lines, existing program lines with line numbers different from those keyed in will be retained as they are, causing severe programming problems! Using NEW first will prevent this.

PC-1600 mode (MODE Ø)
The computer's accessible memory area is referred to as the "User Area". The user area in the PC-1600 is 11834 bytes in size, and includes the BASIC program area, variables storage area, and free area. Any memory allocated for machine language programs with the NEW command reduces the effective size of the user area. The addresses used are relative addresses from the top of memory (Ø).

NEW clears all BASIC and machine programs in the current memory that was specified using TITLE and leaves the area allocated for machine language programs at the previously set value.

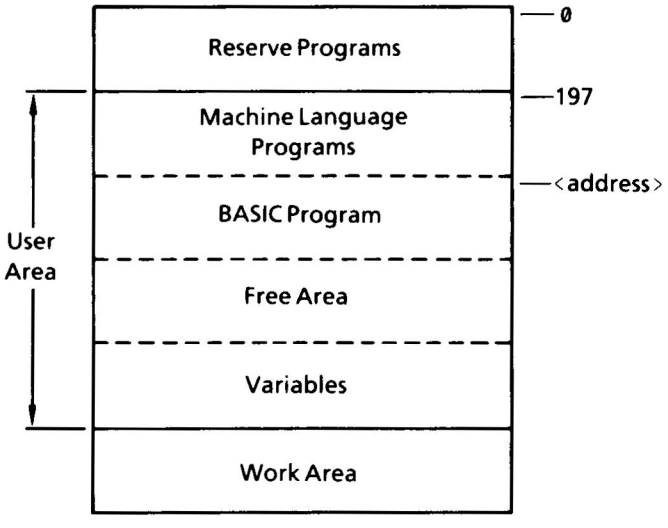
NEW "Sn:" clears all programs in the memory area designated with Sn, where:

SØ: is the computer's main memory area.
S1: is the program module currently in slot 1.
S2: is the program module currently in slot 2.

and leaves the area allocated for machine language programs at the previously set value.

NEW "Sn:",<address> clears all programs in the memory area designated with Sn and allocates the upper address of the machine language program area to <address>. The lower address is 197. The upper address is 65535.

Internal RAM allocation:



NEW 0 clears all BASIC and machine language programs in the memory area specified by TITLE command and sets the area allocated for machine language program to 0 (address = 197).

PC-1500 mode (MODE = 1)
NEW clears all programs in the user area and leaves the area allocated for machine language programs at the previously set value.

NEW <address> clears all programs in the user area and allocates the upper address of the machine language program area to <address>. The lower address is 197. NEW 0 clears all programs inthe user area and sets the area allocated for machine language programs to 0 (address = 197).

EXAMPLE:

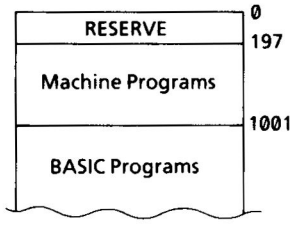
```
>LIST
10:REM THIS IS A
20:REM THREE LINE PROGRAM
30:END
>

>NEW
>LIST
>
```

NEW clears the memory and the program is lost.

```
>NEW 1001
```

This also clears the current memory but reserves addresses 197 to 1000 for machine language programs.



PROGRAM

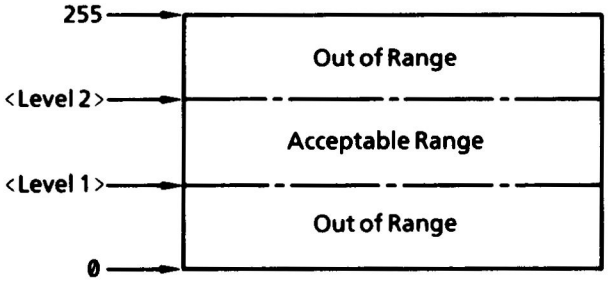
ON ADIN GOSUB

FORMAT: 1. ON ADIN (<level 1>,<level 2>) GOSUB <line #/label>

Abbreviation: O. AD. GOS.
See Also: ADIN ON/OFF/STOP, AIN, RETI

PURPOSE: Causes execution to jump to the specified line number when the voltage input to the analog input jack deviates from a set range.

REMARKS: <level 1> and <level 2> specify the range between 0 and 255 outside of which the input will generate an interrupt as follows:



0 ≤ Level 1 < Level 2 ≤ 255

Refer to Part III Chapter 6 for details on input levels.

Input of an interrupt to the ON ADIN GOSUB statement may be controlled by the ADIN ON/OFF/STOP statement. If an ADIN ON statement is not specified in the program after the ON ADIN GOSUB statement, the default setting is ADIN STOP, and this will be effective after execution of the statement, disabling all interrupts.

The subroutine referenced in this statement must end in a RETI statement.

The maximum number of interrupts sllowed in one program is 8.

NOTE:
Piror to the execution of this command, you must execute the following command:
POKE & F12C, (PEEK & F12C) OR 1

EXAMPLE:

```
10:POKE &F12C,(PEEK &F12C)OR 1
20:ON ADIN (103,206)GOSUB 500
:
:
:
:
490:END
500:REM JACK INTERRUPT ROUTINE
:
:
:
600:RETI
```

[20] Defines where control jumps to on receiving a jack input signal outside the specified range.

[30–480] The main program. When the interrupt occurs during execution of these lines control jumps immediately to the start of the routine on line 500.

[500–600] The jack input interrupt routine. Action is taken according to the interrupt. The routine ends in the RETI statement. Its effect is equivalent to RETURN in a normal routine.

ON COMn GOSUB

FORMAT: 1. ON COMn GOSUB < line #/label >

Abbreviation: O. COM GOS.
See Also: COMn ON/OFF/STOP, RETI

PURPOSE: Causes execution to jump to the specified line of a subroutine when an interrupt is received at the specified port.

REMARKS: Parameter n specifies either the RS-232C port (n = 1) or the optical serial I/O port (n = 2). Input of an interrupt to the ON COMn GOSUB statement may be controlled by the COMn ON/OFF/STOP statement. If a COMn ON statement is not specified in the program after the ON COMn GOSUB statement, the default setting is COMn STOP, and this will be effective after execution of the statement, disabling all interrupts.

The subroutine reference in this statement must end in a RETI statement. The maximum number of interrupts in a program is 8.

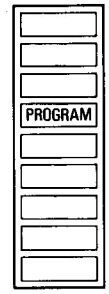
NOTE:
See Part III Chapter 6 for details on which control signals are used for interrupts to the two serial ports.

EXAMPLE:

```
20:ON COM1 GOSUB 500
:
:
:
:
490:END
500:REM INTERRUPT ROUTINE FOR RS-232C PORT
:
:
:
600:RETI
```

[20] Defines where control jumps to on receiving a COM1 interrupt.
[20–480] The main program. If any COM1 interrupt is received during execution of these lines control jumps immediately to the start of the routine on line 500.
[500–600] The COM1 or RS-232C port interrupt routine. Action is taken according to the COM1 interrupt. The routine ends in the RETI statement. Its effect is equivalent to RETURN in a normal routine.

ON ERROR GOTO



FORMAT: 1. ON ERROR GOTO <line #/label>

Abbreviation: O. ER. G.
See Also: RESUME, ERL, ERN

PURPOSE: Causes execution to jump to the specified line (usually the first line of an error processing routine) when an error occurs.

REMARKS: Error processing routines can be written by the user to evaluate what type of error has occurred (for instance, a syntax error or an error in data format), or to check for occurrence of a certain type of error, and then to resume execution at a different point in the program depending on the error type. If there is an error in the error processing routine itself, BASIC will return to the ON ERROR GOTO statement, display the error code and terminate execution.

The error processing routine must end in RESUME, STOP or END. There is no limit on the number of ON ERROR GOTO statements in a program. When an error occurs, control is passed according to the specification of the last ON ERROR GOTO statement that was executed. ON ERROR GOTO 0 switches back to normal BASIC error handling. The effect of ON ERROR GOTO is released by RUN and END execution, or All Clear operation with the SHIFT + CL keys but not on start of execution with GOTO or DEF.

EXAMPLE:

```
5:ON ERROR GOTO 100
10:SAVE "X:DEMO"
20:PRINT "OK"
30:END
:
:
100:IF ERN =160THEN PRINT "NO DISK IN DRIVE X:"
110:PRINT "PLEASE INSERT AND PRESS Y"
120:INPUT A$
130:IF A$="Y"THEN RESUME
140:STOP
```

[100–140] Error processing routine to check for Error 160.

ON..GOSUB / ON..GOTO

FORMAT: 1. ON <numeric expression> GOTO <list of line #s/labels>
2. ON <numeric expression> GOSUB <list of line #s/labels>

Abbreviation: O. GOS./O.G.
See Also: GOSUB...RETURN, GOTO

PURPOSE: Causes execution to jump unconditionally to one of several program lines depending on the simple value of an expression at execution time.

REMARKS: The value of <numeric expression> at the time of execution determines which line number out of the <list of line #s> execution will jump to. If the value of <numeric expression> is 1, execution will jump to the first line number in the list; if it is 2, execution will jump to the second line number in the list, and so on. If the value of <numeric expression> is larger than the number of items in <list of line #s> or less than 1, control passes to the next program line.

With the ON..GOSUB form, each of the line numbers in the list must be the first line of a subroutine.

EXAMPLE:

```
10:INPUT "NUMBER (1-3)?",N
20:ON N GOTO 30,50,70
30:PRINT "IGNITION"
40:GOTO 80
50:PRINT "BOOST"
60:GOTO 80
70:PRINT "ORBIT"
80:END
```

[10] Inputs a number from 1 to 3. Notice that here there is no check that the value entered at the keyboard lies within this range. It would be wise to include an IF statement immediately following this line to ensure this.
[20] Control now branches to one of three lines depending on the value of N.
[30] Branches here when N is 1.
[50] Branches here when N is 2.
[70] Branches here when N is 3.

```
10:INPUT "NUMBER (1-3)?",N
20:ON N GOSUB 100,200,300
:
90:END
100:REM FIRST SUBROUTINE
:
190:RETURN
200:REM SECOND SUBROUTINE
:
290:RETURN
300:REM THIRD SUBROUTINE
:
390:RETURN
```

The logic of using ON GOSUB is the same as for ON GOTO except that control passes to the start of a subroutine.

ON KEY GOSUB

FORMAT: 1. ON KEY GOSUB <list of line #s/labels>

Abbreviation: O. KEY GOS.
See Also: KEY ON/OFF/STOP, RETI

PURPOSE: Causes program execution to branch when one of the function keys F1 to F6 is pressed during run time.

REMARKS: Which of the six function keys is pressed determines which line number out of the <list of line #s/labels> execution will jump to. If F1 is pressed, execution will jump to the subroutine on the line number listed first in the <line #s/labels>. If F6 is pressed, execution will branch to the subroutine listed sixth. If the <list of line #s/labels> contains more than six items, the subroutines on the line numbers in the excess items will never be accessed by this statement. If the <list of line #s/labels> contains less than six items, pressing a function key which has no corresponding line number will have no effect on program execution flow.

Each subroutine must be terminated in a RETI statement to return execution to the statement immediately after the ON KEY GOSUB statement. Use of the function keys as run time interrupts is only active while the program is running. Execution of the RUN command or END statement clears all ON KEY GOSUB function key allocations and returns key functions to original settings. If a KEY ON statement is not specified in the program after an ON KEY GOSUB statement, the default setting is KEY STOP, and this will be effective after execution of the statement, disabling all interrupts.

EXAMPLE:

```
20:ON KEY  GOSUB 400,500,600
30:KEY(1) ON
40:KEY(2) ON
50:KEY(3) ON
:
390:END
400:REM INTERRUPT ROUTINE FOR KEY 1
:
490:RETI
500:REM INTERRUPT ROUTINE FOR KEY 2
:
590:RETI
600:REM INTERRUPT ROUTINE FOR KEY 3
:
690:RETI
```

[20]–[50] Defines where control jumps to when a key is pressed.
[60–390] The main program. If any key interrupt is received during execution of these lines control jumps immediately to the start of the corresponding routine.
[400–690] The key interrupt routines. Action is taken according to the key interrupt. The routines end in the RETI statement. Its effect is equivalent to RETURN in a normal routine.

ON PHONE GOSUB

FORMAT: 1. ON PHONE GOSUB<line #/label>

Abbreviation: O. PH. GOS.
See Also: PHONE ON/OFF/STOP, RETI

PURPOSE: Causes execution to jump to the specified line number when there is an input to the computer from a telephone modem via the RS-232C port.

REMARKS: Input of an interrupt to the ON PHONE GOSUB statement may be controlled by the PHONE ON/OFF/STOP statement. If a PHONE ON statement is not specified in the program after an ON PHONE GOSUB statement, the default setting is PHONE STOP, and this will be effective after execution of the statement, disabling all interrupts.

The subroutine referenced in this statement must end in a RETI statement. The maximum number of interrupts in one program is 8.

NOTE:
The actual signal polled by the computer with this statement is the CI signal on pin 9 of the RS-232C interface. This signal is supported by some types of modems. See Part III Chapter 6 for details on the control signals for the RS-232C serial port.

EXAMPLE:

```
20:ON PHONE GOSUB 500
:
:
:
490:END
500:REM PHONE INTERRUPT ROUTINE
510:PRINT "YOU HAVE A CALL"
:
:
:
600:RETI
```

[20] Defines where control jumps to on receiving a phone interrupt.
[30–480] The main program. If any phone interrupt is received during execution of these lines control jumps immediately to the start of the routine on line 500.
[500–600] The phone interrupt routine. Action is taken according to the phone interrupt. The routine ends in the RETI statement. Its effect is equivalent to RETURN in a normal routine.

ON TIME$ GOSUB

FORMAT: 1. ON TIME$="MM/DD/HH/mm" GOSUB <line #/label>

Abbreviation: O. TI. GOS.
See Also: RETI, TIME$, TIME$ ON/OFF/STOP

PURPOSE: Causes execution to jump to a subroutine when the specified time is reached.

REMARKS: The built-in real time clock provides the input against which the time set in the TIME$="MM/DD/HH/mm" parameter is compared. When the set time is reached, execution jumps to the specified first line of a subroutine. The subroutine must be terminated on the last line with the RETI statement.

For the format in which the time is specified, refer to the description of the TIME$ statement.

The ON TIME$ GOSUB statement can be disabled and re-enabled with the TIME$ ON/OFF/STOP statement. If a TIME$ ON statement is not specified in the program after the ON TIME$ GOSUB statement, the default value is TIME$ STOP, and this will be effective after execution of the statement, disabling all interrupts.

The maximum number of interrupts in a program is 8.

EXAMPLE:

```
10:ON TIME$="05/12/15/30" GOSUB 500
```

[10] Jumps to interrupt subroutine on line 500 at the specified time and date during execution.

PRO
RUN
PROGRAM
RAM

OPEN

FORMAT: 1. OPEN "<d:filename>" FOR { INPUT / OUTPUT / APPEND } AS #<file #>

Abbreviation: OP.
See Also: CLOSE, INPUT#, MAXFILES, PRINT#

PURPOSE: Enables input and output of data to and from a file on floppy disk or other device.

REMARKS: This command assigns a number to a data file by which to specify the file in subsequent read or write statements (see descriptions of INPUT# and PRINT#). After execution of this command, the file is said to be "open".

A file can be opened in three modes; INPUT, OUTPUT, and APPEND. In INPUT mode, data is read from the file with the INPUT# statement. In OUTPUT mode, data is written to the file with the PRINT# statement. In APPEND mode, data is added to the end of the existing file data with the PRINT# statement.

OUTPUT mode cannot be used to add data to an existing file; only to write data to a new file. If a file already exists under the same name on the device, it will be completely overwritten.

<d:filename> specifies the device name followed by the filename. The value of <file #> is any number between one and the maximum number of files set with the current MAXFILES command.

A file cannot be opened for input and output at the same time; it must first be closed, and then reopened. A file cannot be opened in the following cases:

(1) If OUTPUT mode is specified for a file which has been write-protected with the P option of the SET command.
(2) If the file to be opened is on a floppy disk which has the "write-protect" notch covered.
(3) If the file to be opened is on a RAM disk module with the "write-protect" switch set to ON.
(4) If COM1: or COM2: (serial I/O ports) or CAS: (cassette tape) is specified for the device in the APPEND mode.

EXAMPLE:

```
5:MAXFILES =1
10:OPEN "X:DATA"FOR OUTPUT AS #1
20:FOR J=1TO 5
30:PRINT #1,J
40:NEXT J
50:CLOSE #1
60:OPEN "X:DATA" FOR INPUT AS #1
70:IF EOF (1)THEN 110
80:INPUT #1,J
90:PRINT J
100:GOTO 70
110:REM THE END OF THE DATA FILE HAS BEEN REACHED
120:CLOSE #1
130:END
```

[5] Allocate space for 1 file in memory.
[10] Create a new file on disk.
[30] Write the numbers 1 to 5 to the file consecutively.
[60] Open the file again.
[70] Test for end of file.
[80] Read in and print the numbers 1 to 5.

PRO
RUN
PROGRAM

OUT

FORMAT: 1. OUT <port address>, <list of expressions>

Abbreviation:
See Also: INP

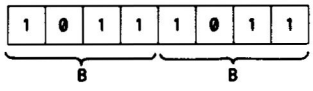
PURPOSE: Outputs a bit pattern directly to the Z-80A microprocessor's output port.

REMARKS: OUT outputs the binary bit pattern for the values of the expressions in <list of expressions> directly to the specified machine output port. <port address> specifies the machine output port as a 16-bit address in the range 0 to 65535 (&0 to &FFFF). <list of expressions> specifies a list of one or more expressions, separated by commas, whose values in the range 0 to 255 (&0 to &FF) are sent to consecutive ports starting from the specified port address.

EXAMPLE:

```
>OUT 80,187
```

Outputs the bit code for 187(&BB) to port address 80(&50). Bit pattern &BB appears as:



OUTSTAT

FORMAT: 1. OUTSTAT "COM1:"[,<setting>]

Abbreviation: OU.
See Also: INSTAT

PURPOSE: Sets the states of the control signals for the RS-232C serial port.

REMARKS: OUTSTAT sets the states of the RTS (Request To Send) and DTR (Data Terminal Ready) signals for the RS-232C port as follows:

<setting>	RTS	DTR
0	High	High
1	High	Low
2	Low	High
3	Low	Low

If <setting> is not specified, both the RTS and DTR signals remain high while a serial port command is being executed or while the port is receiving data, but are low at other times. When the rate of incoming data results in the receive buffer becoming full, RTS changes to low to temporarily halt data transmission and allow the buffer to empty.

EXAMPLE:

```
>OUTSTAT "COM1:",2
```

For the RS-232C port, sets RTS low and DTR high.

PRO
RUN
PROGRAM

PAPER

FORMAT: 1. PAPER <type>[,<limit from>][,<limit to>]

Abbreviation: PAP.
See Also: GRAPH, TEXT

PURPOSE: Sets the type of paper and the print range in the vertical direction for the printer.

REMARKS: The paper type is specified as:

C for cut sheets
R for roll paper

The two optional print limit parameters <limit from> and <limit to> set the printable area in the Y direction (vertical) in units of 0.2mm. Pressing the [↑] key on the printer, turning power on or executing TEXT, GRAPH or PAPER set the print limits in the PAPER statement from the current paper and pen positions. At power on, the last set values are effective. After an ALL RESET, <type> is reset to R.

<limit from> specifies the limit of the print range in the reverse direction in the range 30 to 2047. The default value is 30 (6mm) for cut sheet and 999 (199.8mm) for roll paper.
<limit to> specifies the limit of the print range in the forward direction in the range 30 to 2047. The default value is 1354 (272.8mm) for cut sheet and 999 (199.8mm) for roll paper.

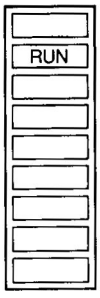
The <limit from> and <limit to> parameters are reset to their default values when the GRAPH or TEXT commands are executed. In addition, in TEXT mode, the default for <limit to> becomes infinite when roll paper is specified.

NOTE:
In graphics mode, an instruction to move the pen outside the range specified in the PAPER statement draws the reflection of the projected path at the print limit setting.

EXAMPLE:

```
>PAPER C
```

Sets the printer to use standard A4 sheets. The default <limit from> and <limit to> values will be used.



PASS

FORMAT: 1. PASS "<string>"

Abbreviation: PAS.
See Also: CLOAD, CSAVE

PURPOSE: Sets and cancels the passwords used to restrict access to programs.

REMARKS: The PASS command is used to restrict the computer to operating just in RUN mode to protect programs from unauthorized access. A password consists of a character string enclosed in quotes of up to 8 characters consisting of any combination of the keyboard characters plus the numbers 0 to 9, with the exception of the " symbol.

Once a password has been set, the COMPUTER cannot be changed to PRO or RESERVE modes. The following operations are not effective and lines cannot be added or deleted: LIST, LLIST, (C)SAVE, (C)LOAD, NEW, TITLE, MERGE, CHAIN, and the use of the ↑ and ↓ keys.

If there are several programs in memory when the password is set, all programs in memory are protected. The only way to remove protection is to execute another PASS command with the same password.

The command cannot be used if there is no program in memory. To change the password, first enter the command with the old password, then reenter the command with the new password.

EXAMPLE:

```
>PASS "SECRET"
```

Sets the password "SECRET" for all programs in memory.

PAUSE

PRO
RUN
PROGRAM

FORMAT: 1. PAUSE [<list of expressions>] {[;] / [,]}

2. PAUSE USING <format string>;<list of expressions>

Abbreviation: PA.
See Also: PRINT, WAIT

PURPOSE: Displays data on the screen for a fixed length of time.

REMARKS: The PAUSE statement is identical to the PRINT statement except that when used in MODE 1 the data is displayed on the screen for a fixed period of time (0.85 seconds) and is then scrolled. See the description of the PRINT statement. Use of the PAUSE command is similar to using a WAIT command preceding a PRINT command.

In MODE 1, the <list of expressions> can contain a maximum of 2 items if the separator is a comma.

EXAMPLE:

```
10:PAUSE "READ THIS QUICKLY!"
20:PRINT "NOW ITS GONE"

>MODE 1
>RUN
```

[10] Prints out the message which is overwritten in less than a second.

PRO
RUN
PROGRAM

PCONSOLE

FORMAT: 1. PCONSOLE "LPT1:",[<line length>],[<EOL code>],[<offset>]
2. PCONSOLE "COMn:",[<line length>],[<EOL code>]

Abbreviation: PCONS.
See Also:

PURPOSE: 1. Sets the print format and end of line code for the printer.
2. Sets the line length and end of line code for communication through the serial ports.

REMARKS: 1. In the first format, PCONSOLE "LPT1:" sets the format for the printer connected to the currently active printer port. <line length> sets the length of each line printed to match the number of columns printable by the printer currently being used. If set to 0, the line length is unlimited. Otherwise set in the range 16 to 255. <EOL code> sets the function of any CR code (&0D) encountered in data sent to the printer by an LPRINT command according to the table below. Note, however, that for the CE-1600P Printer/Cassette Interface Unit, the <EOL code> setting is not valid; all CRs in the data are converted to CR+LFs.

<EOL code>	end of line code
0	CR (&0D)
1	LF (&0A)
2	CR + LF

Any other value will generate an error code. <offset> sets the page offset for the printer; that is, the number of blanks columns printed at the head of each line. The setting must be less than or equal to the value of <line length> −4. The default values are all 0 (infinite line length, CR, no offset).

NOTE:
When the number of characters printed out with LPRINT moves the pen less than 4mm (ie when characters are very small as in CSIZE 1), the pen will not return to the left margin after a line feed. To avoid this problem, when printing in this situation, pad the print statement with blanks so that the pen is moved at least 4mm when printing. e.g. LPRINT "A "

2. In the second format PCONSOLE "COMn:" sets the line length and end of line code when sending a file through either of the two serial ports.

COM1: RS-232C serial port.
COM2: Optical serial port.
COM: Port currently selected with the SETDEV statement.

<line length> sets the length of each line of text sent to the port delimited by the end of line code as set in <EOL code>. If set to 0, the text sent to the port is not delimited into lines. Otherwise set in the range 16 to 255.
<EOL code> sets the end of line code which is sent to the port each time a carriage return code in encountered. The settings are 0,1 or 2 as in format 1. When data is sent to the serial port with the LLIST, LPRINT, or LFILES commands, all CRs encountered in the data are converted as specified by the <EOL code> setting. However, for files transferred via the serial port with the SAVE, LOAD, PRINT# and INPUT# commands, the EOL code is fixed as CR+LF. For data received through the serial port with the INPUT command, all CRs and CR+LFs encountered in the received data are interpreted as data delimiters.

The currently set values for all parameters in the PCONSOLE statement are retained until changed by another PCONSOLE statement.

If the character size is changed using CSIZE, the line length specified by PCONSOLE will change in the same proportion but the offset will remain unchanged.

EXAMPLE:

```
>PCONSOLE "LPT1:",42,2,3
```

Sets the maximum printer line length to 42 characters, and terminates each line in a carriage return and line feed (CR + LF). Sets the left margin at position 4, i.e., the first 3 columns are blank and printing begins at column 4.

PRO
RUN
PROGRAM

PEEK

FORMAT: 1. PEEK (<address>)
2. PEEK #(<mem bank>,<address>)

Abbreviation: PE.
See Also: POKE, XPEEK, XPOKE

PURPOSE: Returns one byte of data from the specified address in memory.

REMARKS: <address> specifies the memory address in the range 0 to 65535 (&0 to &FFFF). <mem bank> specifies a memory bank in the range 0 to 7 in which the required data byte is stored. Using format 1 with an address in the range &C000 to &FFFF reads from the internal RAM held in bank 0. For reading all other memory areas, use format 2 where the memory bank is explicitly stated. For further details, refer to the memory map in the Appendices.

EXAMPLE:

```
>PEEK#(0,100)
                                17
>
```

Returns the data in memory address 100 (&64) in memory bank 0 . The current value is 17.

PHONE ON/OFF/STOP

FORMAT: 1. PHONE ON
2. PHONE OFF
3. PHONE STOP

Abbreviation: PH.
See Also: ON PHONE GOSUB

PURPOSE: Enables or disables interrupts received through the RS-232C port via a telephone modem.

REMARKS: PHONE ON enables interrupts to be received through the RS-232C port. The ON PHONE GOSUB statement can then be used to branch execution to the relevant interrupt-processing routine.

PHONE OFF disables interrupts from the RS-232C port.
PHONE STOP disables interrupts from the RS-232C port but registers the most recent of any interrupt requests which are received. If PHONE ON is executed subsequently, that interrupt will be processed immediately. The default value is PHONE STOP.

EXAMPLE:

```
10:ON PHONE GOSUB 250
20:PHONE ON
:
40 PHONE OFF
```

PRO
RUN
PROGRAM

PITCH

FORMAT: 1. PITCH [<char.pitch>][,<line spacing>]

Abbreviation: PI.
See Also: CSIZE

PURPOSE: Sets the character pitch and line spacing for the printer in text mode.

REMARKS: The spacings set by the PITCH command are as follows:

<char.pitch> parameter	Actual char.pitch
default 4–240	6 × CSIZE <size> × 0.2 mm <char. pitch> × 0.2 mm

<line spacing> parameter	Actual line spacing
default 4–255	12 × CSIZE <size> × 0.2 mm <line spacing> × 0.2 mm

The PITCH settings are returned to the default values after execution of TEXT, GRAPH, CSIZE, LLIST, TEST and PCONSOLE. The PITCH command can be used in graphic mode, but the <line spacing> parameter is ignored.

EXAMPLE:

```
10:PITCH 30,48
20:LPRINT "FIRST LINE"
30:LPRINT "SECOND LINE"
40:LPRINT "THIRD LINE"
50:END
```

[10] Sets the character pitch for the printer to 30 and the line spacing to two.
[20–40] These lines will be printed on the printer to the above specification. Try changing the PITCH parameters and see the effect is has.

POINT

PRO
RUN
PROGRAM

FORMAT: 1. POINT (X,Y)
2. POINT (Xcol)

Abbreviation: POI.
See Also: GPRINT, PRESET, PSET

PURPOSE: Returns the setting of the dot or bit-image column at the specified position on the screen.

REMARKS: In format 1, the function returns a 1 or 0 indicating whether the dot at the specified position (X,Y) is set (1) or not set (0). The values for X are normally in the range 0 to 155 (one line on the graphics screen) and the values for Y in the range 0 to 31 (screen height).

In format 2, the function returns a decimal number in the range 0 to 255 indicating the value of the bit-image data in the 8-dot column at the specified horizontal location (Xcol) on the current line as follows:

POINT (25) will return the value in decimal of the 8-bit image data in the 25th dot column from the left of the screen in the current line. If the value returned is, say 98, this converts to 01100010 inbinary. Then the 8-dot column is:

	−0
x	−1
	−0
	−0
	−0
x	−1
x	−1
	−0

Refer to the description of GPRINT for details on bit-image representation.

NOTE:
Although the X and Y coordinate values must be in the range 155 × 31 in order to refer to a point on the real screen, they can be specified in the range −32768 to 32767 without generating an error. The number returned in this case will be 0 for both formats.

EXAMPLE:

```
:
:
300:N=POINT (77,15)
310:IF N=1GOTO 500
:
:
```

[300] Checks the dot in the middle of the screen at location 77, 15 to see if it is on (black).
[310] Control jumps to line 500 if the dot was on. The POINT statement can be used in this way to detect patterns on the screen and to use this information to make the program branch.

POKE

PRO
RUN
PROGRAM

FORMAT: 1. POKE [#<mem bank>,]< address>,<integer list>

Abbreviation: PO.
See Also: PEEK, XPEEK, XPOKE

PURPOSE: Writes one byte or a series of data bytes into the specified address(es) in memory.

REMARKS: <address> specifies the memory address in the range 0 to 65535 (&0 to &FFFF).

<mem bank> specifies a memory bank in the range 0 to 7 in which the required data byte is to be stored. If no memory bank is specified, the default memory bank is that in which the program file header label for the program currently being executed is held.

<integer list> specifies a list of one or more data bytes as integers in the range 0 to 255 (&0 to &FF) separated by commas, which are written into consecutive memory addresses starting from the specified address. If there is insufficient memory free to hold all the data in the list, an error code is generated. For further details, refer to the memory map in the appendices.

EXAMPLE:

```
>POKE #2,&FF00,255,255
>
```

Starting in memory address &FF00 (65280) of memory bank 2 the two values, 255 and 255 are written to memory. 255 is &FF. Thus the result will be to set 16 bits of memory to "1". Do not experiment with this command unless you have studied the memory map and you know where you are writing to. This example is only a demonstration and will cause problems if run! However, an ALL RESET will recover from errors resulting from bad memory mapping.

PRO
RUN
PROGRAM

POWER

FORMAT: 1. POWER OFF
2. POWER AOFF [(n)]

Abbreviation: POW.
See Also: ALARM$, ARUN, WAKE$

PURPOSE: Sets the auto power off function and/or allows power to be turned off from within a program.

REMARKS: POWER OFF turns off the COMPUTER immediately, either entered directly or from within a program.

POWER AOFF (n) If n = 0 or is omitted, auto power off is set to 10 minutes after the last keyboard input or processing step. If n = 1, auto power off is disabled. The command has no effect for values other than 0 or 1. The default setting after an ALL RESET is auto power off after 10 minutes and auto power on from a telephone modem or at a specified time (see the WAKE$ command).

EXAMPLE:

```
5:WAIT
10:PRINT "IF THERE IS NO MORE WORK, I'LL TAKE A REST"
20:POWER OFF

>POWER AOFF 0
>
```

Sets auto power off to 10 minutes.

PRESET

PRO
RUN
PROGRAM

FORMAT: 1. PRESET (X,Y)

Abbreviation: PRE.
See Also: PSET, LINE

PURPOSE: Resets the dot at the specified coordinate on the screen.

REMARKS: The PRESET command resets (turns OFF) the dot at the position specified by the coordinates (X,Y) in the range of the graphics screen (155 × 31).

NOTE:
Although the X and Y coordinate values must be in the range 155 × 31 in order to reset a point on the real screen, they can be specified in the range − 32768 to 32767 without generating an error.

EXAMPLE: See PSET.

PRO
RUN
PROGRAM

PRINT

FORMAT: 1. PRINT [<list of expressions>] $\left\{\begin{matrix}[;]\\[,]\end{matrix}\right\}$

Abbreviation: P.
See Also: MODE, PRINT USING, WAIT

PURPOSE: Outputs data to the screen.

REMARKS: The positions in which the data items from the PRINT statement are displayed on the screen are determined by the delimiting punctuation used to separate the expressions in <list of expressions>. Either the semicolon or the comma may be used, but the effect is different. In MODE 1, the <list of expressions> can contain a maximum of 2 items if the separator is a comma.

BASIC divides the display screen up into zones consisting of 13 spaces each. When items in a print list are delimited by commas, items are displayed according to these fixed zones.

PRINT <list of expressions> displays on the screen the values of the expressions in <list of expressions>. The expressions in the list can be numeric or string variables, or actual strings if enclosed in quotes. The exact display format depends on the delimiter used between the items in the <list of expressions>. If a semicolon (;) is used to delimit items in the list, the items are displayed immediately following one another. If a comma (,) is used, each item is displayed positioned at the front of a fixed-length print zone of 13 spaces. Numeric data is right-justified and string data left-justified. This means in effect that data is displayed every 13 spaces, equivalent to tabbing 13 columns. In MODE 1 the PRINT statement halts execution and waits for an ENTER before continuing with the program. In MODE 0, the screen will continue to display whatever is currently on the screen for the time defined in the last WAIT statement before continuing. See the description of the WAIT command for details.

Unlike the LPRINT statement, the TAB option is not available in a PRINT statement, but data can be displayed in predetermined positions on the screen using the CURSOR statement, or in special formats using the USING option (see description of PRINT USING).

If the <list of expressions> is terminated in a semicolon, the next output to the screen will be continued on the same line; if the list is terminated without a semicolon, the next output will be on a new line.

Using the PRINT command alone outputs a blank line to the screen; equivalent to a line feed on the printer.

EXAMPLE:

```
10:A=1
20:B=22
30:C=333
40:PRINT
50:PRINT "THE VALUES: ";A;B;C
60:PRINT "THE VALUES: ",A,B,C
70:END

>RUN

THE VALUES:  1  22  333
THE VALUES:                1
             22           333
>
```

[40] PRINT on its own prints a blank line.
[50] Uses semicolons to separate the values. A single space precedes the printing of each value.
[60] Uses commas to separate the values. Values are printed in fixed print zones.

PRO
RUN
PROGRAM
RAM

PRINT# / PRINT# USING

FORMAT: 1. PRINT# <file # >, <list of expressions>
2. PRINT# <file # >, USING <format string>; <list of expressions>
3. PRINT# ["<filename>";] <list of variables>

Abbreviation: P.#/P.#U.
See Also: INPUT#, OPEN, PRINT USING

PURPOSE: Used to write data to a sequential output file on floppy disk, RAM disk module, or cassette tape.

REMARKS: 1&2. Floppy disk or RAM disk module
<file # > is the number allocated to the file when it was opened for output with the OPEN statement. An attempt to write to a file which has not been opened will generate an error code. USING <format string> specification is described under the PRINT USING statement.

If items in <list of expressions> are delimited by semicolons (;), data items are written to the file with no blanks between items. If items are delimited by commas (,), data items are written to the file with a blank between each item. The delimiting comma must be written to the file as a data item itself, enclosed in quotes. See the sample program in the section on creating a file in Chapter 11.

3. Cassette tape
When <"filename" > is not specified for the cassette tape, data is written from the current position of the tape head and the file is given no filename. In format 3, only the comma can be used as a separator in <list of variables>.

<list of variables> can contain numeric or string variables. In both cases care must be taken to use the correct delimiters when writing to a file or there may be problems when the data is read in from the file later with the INPUT# statement. Array variables cannot specify individual elements; the whole array must be specified in the form A(*).

EXAMPLE: See OPEN and PRINT.

PRINT USING / USING

PRO
RUN
PROGRAM

FORMAT: 1. PRINT USING <format string>;<list of expressions>
2. USING <format string>

Abbreviation: P.U./U.
See Also: PRINT, LPRINT USING

PURPOSE: Prints string or numeric data using the format specified in <format string>.

REMARKS: <format string> consists of special characters enclosed in quotes which determine the size and format of the field in which the expressions are displayed. <list of expressions> consists of the numeric or string expressions which are to be displayed. Each expression in the <list of expressions> must be delimited from the next by a semicolon.

The characters used to make up <format string> are different when the expression in <list of expressions> is a numeric expression from when it is a character string. For numeric expressions, the number sign (#) is used to compose the format string; for character string expressions, the ampersand sign (&) is used.

There are eight possible formats, illustrated in the following eight examples:

(1) "###"	Integer format:
43	A two digit integer or a two digit
−57	integer with − sign (a + sign will not be displayed).
(2) "###."	Fixed point (integer) format:
98.	A two digit number (or two digits and −
−43.	sign) followed by the decimal point.
(3) "###.##"	Fixed point format:
64.29	Two digit integer part (or two digits and −
−13.44	sign) plus two digit fractional part.
(4) "##.##^"	Floating point format:
3.11E05	Fixed point mantissa plus the exponent
−4.33E−02	character E plus a two digit exponent (or single digit plus − sign). The exponent character plus the exponent itself will always occupy a minimum of 4 columns.

(5) "###,###."	Integer thousands format:
34,567.	A five digit integer (or five digits plus −sign)
−22,444.	with a comma at the thousands position and decimal point.
(6) "+ ###"	Signed integer format:
+ 988	Three digit integer preceded by + or − sign.
(7) "*####"	Filled integer format:
2345	An up to four digit integer (or four digits
**22	and − sign) with leading spaces filled with
*230	asterisks followed by the decimal point.
−***1	If negative, the − sign will appear in the first space, before any asterisk.
(8) "&&&&&&"	Character string format:
ABCDEF	A string of up to six characters, left justified.
GHI	

The total number of numeric format characters (# or *) which can be placed in one format string is 11 in formats (1), (2), (3), (4), (6), and (7) and 14 in format (5). The PC-1600 outputs data to 10 significant digits if no format string is specified (see Chapter 10; Types of Data). The USING statement can be used on its own. In this form, the <format string> specified in that USING statement remains effective for all subsequent PRINT statements until another USING statement is encountered.

To clear the currently active <format string>, execute a PRINT USING or USING statement with no format string specified.

EXAMPLE:

```
10:A=1
20:B=22
30:C=33
40:USING "###.##"
50:PRINT "THE VALUES ARE"
60:PRINT A;B;C
70:END

>RUN
THE VALUES ARE
  1.00 22.00 33.00
>
```

[40] Defines the format for printing numerics; two digits plus sign before the decimal and two after.
[50] Prints out on the screen the values according to the specified format.

PSET

PRO
RUN
PROGRAM

FORMAT: 1. PSET (X,Y) [,<function code>]

Abbreviation: PS.
See Also: PRESET, LINE

PURPOSE: Sets or resets the dot at the specified coordinates on the screen.

REMARKS: The PSET command sets (turns ON) the dot at the coordinates specified by (X,Y) in the range of the graphics screen (155 × 31). The <function code> option is specified as an upper case X. When specified, PSET inverts the state at the specified coordinates. If the dot is ON, it is turned OFF, and vice versa.

NOTE:
Although the X and Y coordinate values must be in the range 155 × 31 to set a point on the real screen, they can be specified in the range − 32768 to 32767 without generating an error.

EXAMPLE:

```
10:FOR Y=10TO 20STEP 2
20:FOR X=10TO 100STEP 10
30:PSET (X,Y)
40:NEXT X
50:NEXT Y
60:FOR Y=10TO 20STEP 2
70:FOR X=10TO 100STEP 10
80:PRESET (X,Y)
90:NEXT X
100:NEXT Y
110:END
```

This will draw a grid of dots across the screen and then remove them.

[10] Sets the loop for the rows.
[20] Sets the loop for the columns.
[30] Each time this statement is executed, a single dot on the screen is turned on (black).
[60–100] Repeats the process of lines 10–50 turning off each dot.

PZONE

FORMAT: 1. PZONE "COMn:",<width>
2. PZONE "LPT1:",<width>

Abbreviation: PZ.
See Also: LPRINT

PURPOSE: Sets the print zone when outputting to the printer or to a serial port with the LPRINT statement.

REMARKS: In BASIC, printout with the LPRINT statement is divided up into zones consisting of a fixed number of spaces each. Items printed out with the LPRINT command using the comma as the delimiter are printed in these zones, with numeric items right-justified and string data left-justified in each zone. The PZONE statement specifies the width of these zones.

Data can also be sent to a serial port with the LPRINT statement, in which case, PZONE specifies the zoning of the items sent to the port in exactly the same way as for the printer.

Either serial port can be specified:

COM1: RS-232C serial port
COM2: Optical serial port
COM: Port currently selected with the last SETDEV statement

For COM1: the <width> range is 8 to 255.
For LPT1: the <width> range is 8 to 80.

The default setting for <width> at power on is 20.

EXAMPLE:

```
10:A=1
20:B=22
30:C=333
40:PZONE "LPT1:",10
50:LPRINT A,B,C
60:END


>RUN
          1           22          333     (on the printer)
>
```

[40] Sets the zone widths for the printer to 10.
[50] Prints on the printer the three values in the columns specified by PZONE.

PRO
RUN
PROGRAM

RADIAN

FORMAT: 1. RADIAN

Abbreviation: RAD.
See Also: DEGREE, GRAD

PURPOSE: Sets the computer to radian mode.

REMARKS: In radian mode, RADIAN is displayed on the top line of the screen. Input to the arguments of SIN, COS and TAN must be in radians. Values returned by the ASN, ACS and ATN functions will be in radians.

EXAMPLE:

```
10:RADIAN
20:PRINT "TRIG FUNCTIONS NOW IN RADIANS"
30:PRINT ASN(0.5),ASN(1.0)
40:PRINT ACS(0.5),ACS(1.0)
50:PRINT ATN(0.5),ATN(1.0)
60:END
```

Try running this program and compare the results with those for DEGREE and GRAD.

[10] Select RADIAN mode.
[30–50] Print out some sample values.

RANDOM

FORMAT: 1. RANDOM

Abbreviation: RA.
See Also: RND

PURPOSE: Changes the series of random numbers to be generated by the RND function.

REMARKS: Using RANDOM at the beginning of a program causes the computer to change the random number generating sequence each time the power is turned ON.

EXAMPLE:

```
>RANDOM
>RUN
```

Changes the series of random numbers that will be generated by RND in the program before the program is run.

PRO
RUN
PROGRAM

RCVSTAT

FORMAT: 1. RCVSTAT "COMn:", <protocol> ,[<timeout>]

Abbreviation: RC.
See Also: INSTAT, SNDSTAT

PURPOSE: Sets the receive handshake protocol for the RS-232C port and the timeout value for the serial ports.

REMARKS: Either serial port can be specified:

COM1: RS-232C serial port
COM2: Optical serial port
COM: Port currently selected with last SETDEV statement

The RCVSTAT command sets up the signal conditions for the CTS (Clear To Send), CD (Carrier Detect) and DSR (Data Set Ready) signals for receiving data through the RS-232C port. If the conditions are met, data can be received through the port. This is known as "handshaking". The signal states in which data will be received are set by the value of the decimal number set for the <protocol> parameter. This number is converted internally to an 8-bit binary number and bits 3, 4, and 5 set the conditions for the control signal for handshaking as shown in the following table. Note that bits 1, 2, 6, 7, and 8 are not used; set them to 0.

Bit	Value	Signal condition
1 (LSB)	Not used	
2	Not used	
3	0	CTS must be high
	1	CTS state doesn't matter
4	0	CD must be high
	1	CD state doesn't matter
5	0	DSR must be high
	1	DSR state doesn't matter
6	Not used	
7	Not used	
8 (MSB)	Not used	

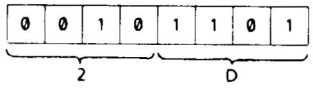
The <protocol> option has no meaning for the optical serial port.

Timeout is the length of time that the COMPUTER will wait for input to a port before deciding that the device being waited for is not ready or not available. The optional <timeout> parameter sets the timeout in the range 0 to 255, in units of 0.5 seconds. 0 timeout disables the timeout setting (infinite timeout) and is the default.

EXAMPLE:

```
>RCVSTAT"COM1:",45
```

Sets up the signal conditions for the RS-232C serial port specifying that RTS and DSR must be high for data to be received. The timeout value is indefinite. 45 in decimal is 2D hex and represents the following bit pattern:



Notice that the most significant, or leftmost bit, is bit 8 while the least significant, or rightmost bit, is bit 1.

READ .. DATA

FORMAT: 1. READ < list of variables >
:
DATA < list of constants >

Abbreviation: READ. DA.
See Also: DATA, RESTORE

PURPOSE: Reads values from a DATA statement and assigns them to variables.

REMARKS: The READ statement must always be accompanied by at least one DATA statement. Each item in the < list of constants > is assigned to each item in the < list of variables > on a one-to-one basis, though a single READ statement need not correspond with a single DATA statement.

Items in the < list of constants > are separated by commas. The type of variable and the type of constant must match for each item read in. See the DATA statement. After DATA items have been read once, they cannot be re-read until a RESTORE statement has been executed.

If the number of variables in < list of variables > is greater than the number of items in < list of constants >, an error code will be generated. If the number of constants is greater than the number of variables, the excess constants on the data line will remain unread.

Refer to the MERGE command for details of using the READ and DATA commands when merging programs.

On first execution of a program, if there is no RESTORE command before the DATA statement specifying where to find the first data line, the PC-1600 will search for the data statement in memory areas in the following order:

S2: (program memory) → S1: (program memory) → S0

However, if program execution is interrupted and restarted with a GOTO command or with the DEF key, if a READ statement has already been executed before the interruption, when it encounters the next READ statement, the PC-1600 will go to the next DATA line in the program to read data.

EXAMPLE:

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

>RUN
10
20
30
40
50
60
70
80
90
100
>
```

[5] Sets the wait time for screen display.
[10] Reserve space for a one-dimensional array.
[20] Read in the values from the DATA statements, B(1) is 10, B(2) is 20, B(3) is 30, etc., and print out the values.

REM

FORMAT: 1. REM<remark line>
2. '<remark line>

Abbreviation:
See Also: LIST, LLIST

PURPOSE: Used to include comments in a program.

REMARKS: An apostrophe (') SHIFT + 1 may be used in place of the REM keyword. Remark lines are ignored by BASIC during program execution, but are listed out exactly as they are written when the program is LISTed. Comments about what the program is doing are extremely useful to other people who may wish to use or modify the program. If a GOTO or other such jump statement jumps to a REM line, execution will resume with the first subsequent line containing an executable statement. Remarks can be appended to lines already containing executable statements by using a colon and then REM, followed by the remark. No other statements can be placed after the REM statement on the same line.

EXAMPLE:

```
10:REM THIS LINE HAS NO EFFECT
20:DIM A$(3,5):REM RESERVING SPACE FOR A 3 X 5 ARRAY
30:'THIS LINE HAS NO EFFECT EITHER
```

RENUM

PRO

FORMAT: 1. RENUM [<new line #>],[< old line #>],[<increment>]

Abbreviation: REN.
See Also: DELETE, EDIT, LIST

PURPOSE: Renumbers the lines of a program.

REMARKS: The line numbers are changed from <old line #> to <new line #> in the specified <increment>. If <new line #> is not specified, the lines are renumbered starting from 10 in increments of 10. A RENUM command which would produce line numbers higher than 65279 will result in a error.

All line numbers referred to in GOTO, GOSUB and other branching statements are renumbered accordingly. If the line number referred to by a branching statement cannot be found in the program prior to renumbering, the program is not renumbered and "undefined in <line #>" is displayed. If a GOTO statement is specified with an algebraic expression (e.g., GOTO A*5), an attempt to renumber the program will result in an error.

The RENUM command cannot be executed on programs which were loaded in MODE 1. First save the program in ASCII format with the A option, and then load the program again in MODE 0.

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
>LIST
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

FORMAT: 1. RESTORE
2. RESTORE <line #/label>

Abbreviation: RES.
See Also: READ, DATA

PURPOSE: Allows DATA statements to be re-read starting from the line specified.

REMARKS: Normally, the READ statement reads items from DATA lines beginning with the first item on the lowest numbered DATA line. It continues in sequence for each subsequent READ statement in the program. The current read position is remembered by a "pointer" stored in memory. RESTORE resets the pointer to the first item in the first data line in the program. RESTORE<line #/label> resets the pointer to the first item on the line number specified. If the specified line number does not contain a DATA statement, the pointer is reset to the first data item on the first data line following the specified line number.

EXAMPLE:

```
10:DIM A1$(3),A2$(3)
20:FOR N=1TO 3
30:READ A1$(N)
40:PRINT A1$(N)
50:NEXT N
60:RESTORE
70:FOR N=1TO 3
80:READ A2$(N)
90:PRINT A2$(N)
100:NEXT N
110:END
120:DATA "PLUM","PEACH","NECTARINE"
```

```
>RUN
PLUM
PEACH
NECTARINE
PLUM
PEACH
NECTARINE
>
```

[10] Dimensions the two string arrays.
[20–50] Reads the three data strings into the three element array A1$ and prints them out.
[60] Restores all data statements so that they can be read again.
[70–100] Reads the same data strings into A2$, a second three element array, and prints them out. The two arrays now hold identical data.

RESUME

FORMAT: 1. RESUME [<line #>]
2. RESUME NEXT

Abbreviation: RESU.
See Also: ON ERROR GOTO, ERL, ERN

PURPOSE: Used to return to normal execution of a program after execution has jumped to the user's error processing routine on meeting an error.

REMARKS: The RESUME command has the same effect for an error processing routine as the RETURN statement has for a subroutine.

RESUME <line #> resumes execution at the statement having the specified line number. RESUME resumes execution at the statement which caused the error. RESUME NEXT resumes execution at the line following the line containing the error. The RESUME statement cannot be used anywhere in a program except in an error processing routine.

EXAMPLE: See ON ERROR GOTO.

RETI

FORMAT: 1. RETI

Abbreviation:

See Also: ON ADIN GOSUB, ON COMn GOSUB, ON KEY GOSUB, ON PHONE GOSUB, ON TIME$ GOSUB

PURPOSE: Returns program execution from an interrupt subroutine.

REMARKS: RETI is used in the same way as RETURN to return to the line following the one which specified a subroutine call. The difference is that RETI is used with subroutines activated by interrupts: ON ADIN GOSUB, ON COMn GOSUB, ON KEY GOSUB, ON PHONE GOSUB and ON TIME$ GOSUB. When RETI is executed, the last interrupt which occurred, if any, while the interrupt processing routine itself was being executed is processed.

EXAMPLE:

```
10:ON PHONE GOSUB 100
:
:
100:PRINT "YOU HAVE A CALL"
110:RETI
```

PRO
RUN
PROGRAM

RIGHT$

FORMAT: 1. RIGHT$(X$,N)
2. RIGHT$("<string>",N)

Abbreviation: RI.
See Also: LEFT$, MID$

PURPOSE: Returns the N characters from the right end of any string X$.

REMARKS: The value of N must be in the range 0 to 80. Fractions will be rounded down (truncated). If N is less than 1, a null string is returned. If N is greather than the number of characters in X$, the whole string is returned.

EXAMPLE:

```
5:WAIT 32
10:X$="SHARP PC1600"
20:FOR N=1TO 13
30:LET S$=RIGHT$ (X$,N)
40:PRINT S$
50:NEXT N

>RUN
0
00
600
1600
C1600
PC1600
 PC1600
P PC1600
RP PC1600
ARP PC1600
HARP PC1600
SHARP PC1600
SHARP PC1600
>
```

RLINE

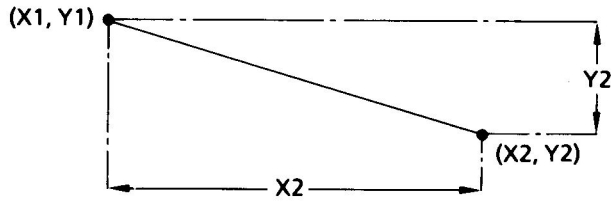
PRO
RUN
PROGRAM

FORMAT: 1. RLINE [(X1,Y1)]–(X2,Y2)[–(X3,Y3)..],[<type>],[<color>][,B]

Abbreviation: RL.
See Also: COLOR, LLINE

PURPOSE: Draws a line or a series of line segments between points specified in relative coordinates on the printer.

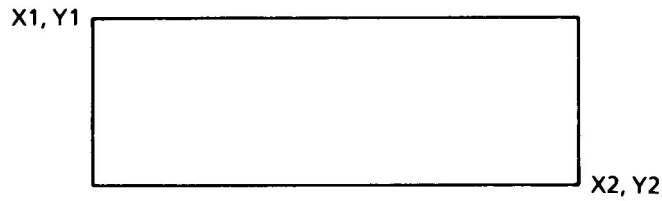
REMARKS: In graphics mode, the RLINE command draws a line segment from the point specified by the coordinates X1, Y1 or from the current pen position if X1 and Y1 are not specified, to the point specified by the relative coordinates X2,Y2. Unlike the LLINE command, the coordinates in the RLINE command are all relative to the last pen position as origin; i.e., the coordinates (X2,Y2) are measured from the point (X1,Y1) as origin.



Up to five subsequent contiguous line segments can be specified with the coordinates of each new point being measured from the last point as origin.

The optional <type> parameter specifies the line type in the range 0 to 9. (See LLINE command). The optional <color> parameter specifies the pen color in the range 0 to 3 (see COLOR command). If <type> and <color> are not specified, the currently set values will be used. The B option specifies a rectangle to be drawn on the diagonal specified by X1, Y1 and X2, Y2 as below:

For the command RLINE (X1,Y1)–(X2,Y2),,,B



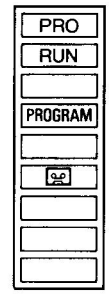
EXAMPLE:

```
5:PAPER R
10:GRAPH
20:GLCURSOR (40,40)
30:RLINE -(100,0)-(-100,100)-(0,-100)
40:TEXT
50:END
```

This will draw a triangle away from the corner of the page. The program produces the same result as the example for LLINE.

[10] Sets the machine to graph mode.
[20] Moves the printer pen to the specified position away from the corner of the page.
[30] Draws the triangle in three movements ending back at the starting point. Notice that unlike LLINE example, for RLINE all pen movements are relative to the last pen position.
[40] Now that the drawing is finished, return to text mode.

RMT ON/OFF



FORMAT: 1. RMT ON
2. RMT OFF

Abbreviation: RM.
See Also:

PURPOSE: Enables and disables remote control of power to the cassette recorder.

REMARKS: RMT ON allows the power to the tape recorder to be turned on by the RMT signal from the COMPUTER during a tape I/O operation, and off otherwise.

RMT OFF disables remote control of power to the tape recorder; the power will remain on whether or not there is a tape I/O operation in progress.

The remote function is operational only when the REMOTE ON/OFF switch on the CE-1600P unit is ON.

EXAMPLE:

```
>RMT ON
```

Enables computer control of the tape recorder.

```
>RMT OFF
```

Disables computer control of the tape recorder.

PRO
RUN
PROGRAM

RND

FORMAT: 1. RND (X)

Abbreviation: RN.
See Also: RANDOM

PURPOSE: Returns a random number in the range specified by the number X.

REMARKS: The computer generates a random number by its built-in random number generator. The same sequence of random numbers is generated each time a program containing RND is executed after power is turned ON.

If X is less than 0, the random number generator repeats the same sequence of random numbers each time RND is executed. If X is a decimal greater than 0 and less than 1, a random number is generated in the range 0 to 1 (but not including 1). If X is greater than 1, a random number is generated in the integer range 1 to X.

All random numbers are generated to 10 significant digits.

EXAMPLE:

```
10:FOR I=1TO 3
20:FOR J=1TO 10
30:R=RND (9)
40:PRINT R;
50:NEXT J
60:PRINT
70:NEXT I
80:END
```

```
>RUN
6 4 2 5 6 8 2 7 6 8
5 5 7 7 1 2 6 5 3 6
3 1 5 7 3 4 5 7 4 2
>

>RUN
2 9 1 1 2 3 3 3 3 5
8 7 4 5 7 5 8 1 4 8
2 8 2 5 9 1 9 4 7 6
>
```

[10] Outer loop for each line of random numbers.
[20] Inner loop for generating 10 random numbers.
[30] Generates a random number from 1 to 9 inclusive.

Here the program is run twice. Notice that a different sequence of random numbers are generated each time.

PRO
RUN
PROGRAM

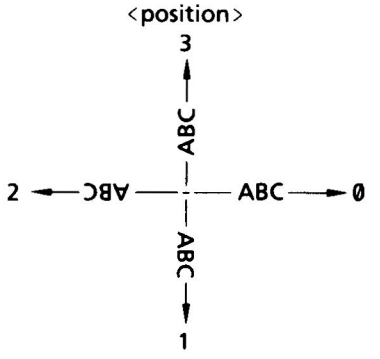
ROTATE

FORMAT: 1. ROTATE <position>

Abbreviation: RO.
See Also:

PURPOSE: Sets the orientation of the printing of each character.

REMARKS: <position> sets the orientation of the print characters as a number from 0 to 3. The effect of <position> on the printout is shown in the two diagrams below:



Defaults are the values set with the last ROTATE command.

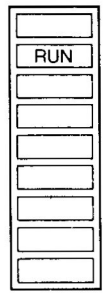
EXAMPLE:

```
5:GRAPH
10:GLCURSOR (400,0)
20:FOR P=0TO 3
30:ROTATE P
40:LPRINT " ABC ";
50:NEXT P
60:END
```

Running this program will demonstrate all the 4 possible combinations for printer ROTATE values.

[5] Sets graphic mode.
[10] Move the printer head to the middle of the paper.
[20] Loops through the 4 possible character positions.
[30] ROTATE according to the loop values.
[40] Print to the printer the simple string.
[50] Repeat the loop.

RUN



FORMAT: 1. RUN
2. RUN <line #/label>

Abbreviation: R.
See Also: CONT, GOTO, LOAD, MERGE

PURPOSE: Starts execution of a program held in memory.

REMARKS: RUN starts execution of the program in memory starting from the lowest numbered statement.
RUN <line #/label> starts execution from the line number or label specified. Executing a program with the RUN command clears all variables and arrays and resets the data pointer on a DATA line to the start.

EXAMPLE:

```
>RUN
```

Starts program execution from lowest line.

```
>RUN 100
```

Starts program execution from line 100.

```
>RUN "F"
```

Starts program execution from line with label "F".

RXD$

FORMAT: 1. RXD$

Abbreviation:
See Also:

PURPOSE: Returns the data, if any, being input to the currently selected serial port at the time of execution.

REMARKS: The data byte being input to the port currently selected with the SETDEV command is returned as a hexadecimal string. If there is no data currently being input, two bytes of data, &20 + &20 are returned. If at the time of execution there is a communication error, three bytes of data, &3F + &20 + &20 are returned.

EXAMPLE:

```
>PRINT RXD$
```

Prints out the data byte currently input to the active serial port.

SAVE / SAVE*

PRO
RUN
PROGRAM
RAM

FORMAT: 1. SAVE "<d:filename>"[,A]
2. SAVE* "<d:filename>"

Abbreviation: S.
See Also: LOAD, MERGE

PURPOSE: Used to save programs or data files to floppy disk or RAM disk module, tape, or one of the serial ports.

REMARKS: SAVE saves the file on the specified device under the specified file name. If the A option is specified, the file is saved in ASCII format, otherwise it is saved in compressed binary format. <d:filename> specifies the device and filename.

SAVE* only saves comment lines (prefixed by REM or an apostrophe ('); see the REM statement) under the specified filename. Other lines are ignored. This gives a simple method of storing text in a file for future display on the screen. See description of LOAD* statement for details on storing and retrieving text.

If no extension is specified for the filename, SAVE will automatically attach the extension .BAS.

If the specified file name already exists, the existing file will be overwritten, unless protected by the P option of the SET command, or unless the disk or module is write-protected.

Files can also be sent to serial port COM2: with the SAVE command using the A option (in ASCII format). Any CR+LF codes (&ØDØA) encountered in the file will be interpreted as end-of-line codes, and &1A as the end-of-file code.

EXAMPLE:

```
>SAVE"X:CHESS"
```

Saves the program in memory to the floppy disk under the name CHESS.

```
>SAVE*"S2:INFO"
```

Saves just the comment lines of the program in memory to the RAM disk module in expansion slot 2. The file is given the name INFO.

PRO
RUN
PROGRAM
RAM

SET

FORMAT: 1. SET "<d:filename>","P"
2. SET "<d:filename>"," "

Abbreviation:
See Also:

PURPOSE: Sets and removes file protection for files on floppy disk and RAM disk.

REMARKS: The SET command is used to set parameters to protect a file against unauthorized access or accidental deletion. <d:filename> specifies the device and filename.

After the P option, write-protect, has been set, the file is protected as follows:

(1) No data can be written to the file. The file cannot be opened in APPEND or OUTPUT mode.
(2) The file cannot be erased with the KILL command.
(3) The file cannot be renamed with the NAME command.

To release the P protection level on a file, execute the SET command for that file name with a space in place of the "P" parameter (format 2).

EXAMPLE:

```
>SET "X:PAYRUN","P"
>
```

Protects the program PAYRUN held on floppy disk from being written to, erased or renamed.

SETCOM

PRO
RUN
PROGRAM

FORMAT: 1. SETCOM "COMn:",[
],[<WL>],[<PR>],[<ST>],[<XO>],[<SI>]

Abbreviation: SETC.
See Also: COM$, SETDEV

PURPOSE: Sets the communication protocol for the serial ports
REMARKS: Either serial port can be specified:

COM1: RS-232C serial port
COM2: Optical serial port
COM: Port currently selected with the last SETDEV statement

The communication protocol is set as a string of six options, separated by commas. The options are:

 Baud rate: 50 to 38400 baud
<WL> Word length: 5–8 bits
<PR> Parity: E–Even parity
O–Odd parity
N–No parity
<ST> Stop Bits: 1 or 2
<XO> X-ON/OFF protocol: X–Yes
N–No
<SI> Shift in/out protocol
For 7 data bits only: S–Yes
N–No

The default setting for the RS-232C serial port is 1200,8,N,1,X,S. The default setting for the optical serial port is 38400, 7, E, 2, X, S.

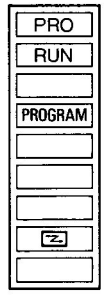
NOTE:
When the SAVE, LOAD, BSAVE or BLOAD commands are being used to transfer files through a port, the word length must be set to 8 bits, and shift IN/OUT protocol must be set to N. However, when saving an ASCII file to a port with SAVE, or when loading an ASCII file from a port with LOAD, or when using PRINT # or INPUT # with a port, communication parameter settings are unrestricted.

EXAMPLE:

```
>PRINT COM$"COM1:"
300,7,N,1,N,N
>SETCOM "COM1:",,,E
>PRINT COM$"COM1:"
300,7,E,1,N,N
>
```

The first line returns the current communication setting of the RS-232C serial port. SETCOM is used to change the parity setting from no parity to even parity. PRINT COM$ is then keyed in to check that the change has been made. Notice that the third parameter is now E instead of N.

SETDEV



FORMAT: 1. SETDEV "COMn:" [,PO] [,KI]

Abbreviation: SE.
See Also:

PURPOSE: Specifies one of the serial ports for input and output with certain BASIC commands.

REMARKS: The SETDEV command basically opens a port for input and output. In addition, two optional parameters can be specified to define the input or output mode. The ports are specified as follows:

COM1:	RS-232C serial port
COM2:	Optical serial port
COM:	Currently opened port

The two optional parameters are specified as follows (in any order):

PO	Directs output from LPRINT, LLIST and LFILES commands to the specified port.
KI	Uses the specified port for input of data with an INPUT command.

Specifying SETDEV with no parameters releases current device settings and reverts to the printer for output and keyboard for input.

EXAMPLE:

```
>SETDEV "COM1:",PO

>SETDEV
```

PRO
RUN
PROGRAM

SGN

FORMAT: 1. SGN(X)

Abbreviation: SG.
See Also:

PURPOSE: Returns the sign of expression X.

REMARKS: X can be any numeric expression. SGN(X) returns either 1, 0 or −1 depending on the sign of expression X as shown below:

Value of X	Value returned by SGN(X)
X>0	1
X=0	0
X<0	−1

EXAMPLE:

```
5:WAIT 100
10:FOR N=-3TO 3
20:PRINT N,SGN (N)
30:NEXT N
40:END

>RUN
             -3            -1
             -2            -1
             -1            -1
              0             0
              1             1
              2             1
              3             1
>
```

SIN

FORMAT: 1. SIN(X)

Abbreviation: SI.
See Also: ASN, COS, TAN

PURPOSE: Returns the sine of the angle X.

REMARKS: This function returns the sine of the angle X, where X is expressed in degrees, radians or as a gradient value, depending on which mode the computer is set to with the DEGREE, RADIAN or GRAD commands.

EXAMPLE:

```
10:DEGREE
20:PRINT "SIN OF 30 IS ";SIN (30)
30:PRINT "SIN OF 90 IS ";SIN (90)
40:END


>RUN
SIN OF 30 IS  0.5
SIN OF 90 IS  1
>
```

PRO
RUN
PROGRAM

SNDBRK

FORMAT: 1. SNDBRK "COMn:",<number>

Abbreviation: SNDB.
See Also:

PURPOSE: Sends a specified number of break codes to the specified serial port to halt input transmission from another computer.

REMARKS: The ports are specified as follows:

COM1:	RS-232C serial port
COM2:	Optical serial port
COM:	Port specified by the last SETDEV command

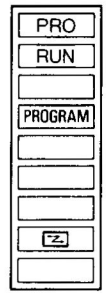
<number> specifies the number of break characters to be sent as a continuous string to the port in the range 1 to 255.

EXAMPLE:

```
>SNDBRK "COM1:",20
```

This will send the break character 20 times to the RS-232C port to stop data transmission. It is often advisable to send more than one break character as the other computer may not successfully receive the first one.

SNDSTAT



FORMAT: 1. SNDSTAT "COMn:",<protocol>,[<timeout>]

Abbreviation: SN.
See Also: RCVSTAT

PURPOSE: Sets the send handshake protocol for the RS-232C port and the timeout value for both serial ports.

REMARKS: Either serial port can be specified:

COM1: RS-232C serial port
COM2: Optical serial port
COM: Port currently selected with the last SETDEV statement

The signal states for CTS (Clear To Send), CD (Carrier Detect) and DSR (Data Set Ready) which enable data transmission are set by the value of the decimal number (0 to 255) set for the <protocol> parameter. The number is converted to an 8-bit binary number internally and bits 3, 4, and 5 set the conditions for the control signal for handshaking to be performed as shown in the following table. Note that bits 1, 2, 6, 7, and 8 are not used; set them to 0.

Bit	Value	Signal condition
1 (LSB)	Not used	
2	Not used	
3	0	CTS high
	1	CTS state doesn't matter
4	0	CD high
	1	CD state doesn't matter
5	0	DSR high
	1	DSR state doesn't matter
6	Not used	

7	Not used
8 (MSB)	Not used

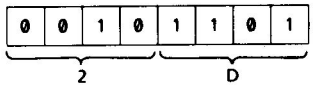
The <protocol> option has no meaning for the optical serial port.

Timeout is the length of time that the COMPUTER will wait before deciding that the device being waited for is not ready or not available. The optional <timeout> parameter sets the timeout in the range 0 to 255, in units of 0.5 seconds. 0 timeout disables the timeout setting (infinite timeout) and is the default.

EXAMPLE:

```
>SNDSTAT "COM1:",45
```

Sets up the signal conditions for the RS-232C serial port specifying that RTS and DSR must be high for data to be sent. The timeout value is infinite. 45 in decimal is 2D hex and represents the following bit pattern:



Notice that the most significant, or leftmost bit, is bit 8 while the least significant, or rightmost bit, is bit 1.

SORGN

FORMAT: 1. SORGN

Abbreviation: SO.
See Also: LLINE, GLCURSOR

PURPOSE: Sets the current pen position as the origin.

REMARKS: In graphics mode, SORGN sets the current position of the pen as the point (0,0) for all following graphic commands. The default position for the origin is the left home position for X and the current paper position set with the printer [↑] and [↓] keys for Y.

EXAMPLE: See LLINE.

PRO
RUN
PROGRAM

SQR

FORMAT: 1. SQR(X)

Abbreviation: SQ.
See Also:

PURPOSE: Returns the square root of expression X.

REMARKS: If the expression evaluates to a negative number, SQR(X) generates an error code.

EXAMPLE:

```
>PRINT SQR(5)
              2.236067977
>
```

STATUS

PRO
RUN
PROGRAM
RAM

FORMAT: 1. STATUS <value>

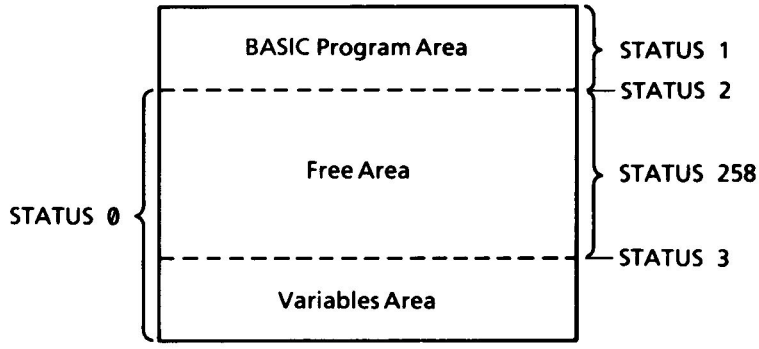
Abbreviation: STA.
See Also: MEM, NEW

PURPOSE: Returns the amount of free space and other information for the various memory areas.

REMARKS: <value> specifies the memory area and information returned as follows:

<value>	Information returned
0	Amount of free memory in the user area in bytes (free area + variables area).
1	Size in bytes of currently loaded program(s).
2	Lower address in memory of the free user area.
3	Last address in free user area + 1.
4 to 255	Last line number executed for the current program.
256	Bank number in which lower address of free user area exists.
257	If an expansion RAM is installed, returns the bank number of the last bank in the free user area in the expansion RAM. If no expansion RAM is installed, returns 0.
258	Amount of unallocated memory in the user area in bytes (free area).
259	Amount of unused memory in program module in slot 1 in bytes.
260	Amount of unused memory in program module in slot 2 in bytes.

Memory Map: Internal RAM User Area

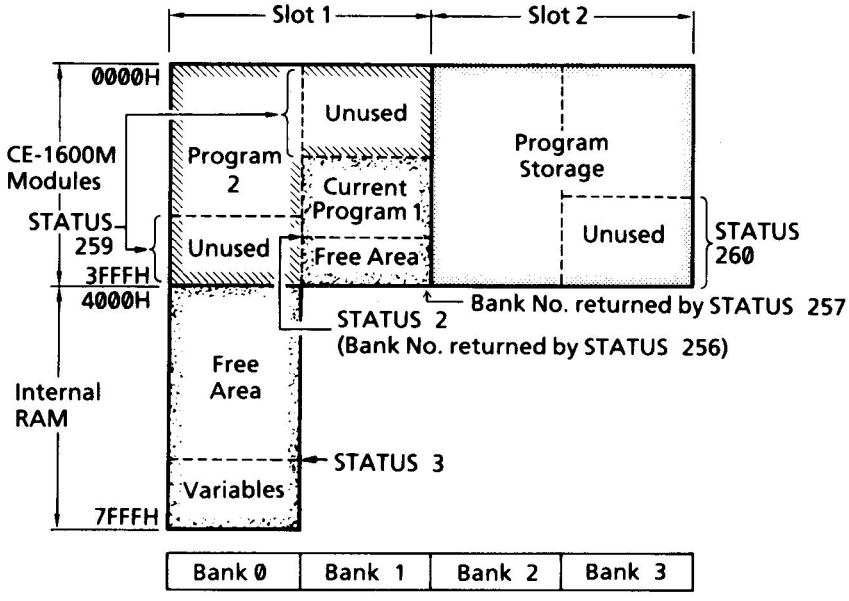


Memory Map: Internal RAM and Expansion Slots

The following memory map represents memory allocation with:

(1) A CE-1600M module in slot 1 with part only allocated for program storage; the remaining space is allocated for expansion of internal RAM (free area).

(2) A CE-1600M module in slot 2 with the space allocated only to program storage.



EXAMPLE:

```
>STATUS 0
                              10789
```

The available free memory is 10789 bytes.

STOP

FORMAT: 1. STOP

Abbreviation: ST.
See Also: CONT, END

PURPOSE: Used to halt execution, usually during debugging.

REMARKS: The STOP statement is usually inserted into a program at the debugging stage to halt execution to allow the contents of variables to be examined or changed by direct input. The following message is displayed:

BREAK IN <line #>

Execution may be restarted using the CONT command provided no modifications are made to the program during the interruption. Unlike the END statement, files are not closed when a STOP statement is executed.

EXAMPLE:

```
10:FOR N=1 TO 50
20:LET S=N*5
25:STOP
30:LINE (0,0)-(N,S)
40:NEXT N

>RUN
BREAK IN 25
:
:
PRINT S
                              5
>CONT
:
```

[25] Debugging line to look at the value of S.

STR$

PRO
RUN
PROGRAM

FORMAT: 1. STR$ (<numeric value>)

Abbreviation: STR.
See Also: VAL

PURPOSE: Converts numeric data into string data.

REMARKS: The STR$ function changes an integer number into a string. The string will be composed of the same digits as the original number, but it is treated as a string of SHARP ASCII characters in subsequent processing. The STR$ function has the opposite effect to the VAL function.

If the numeric data is negative, the string will be preceded by a – sign. If <numeric value> is too large to be held in the string variable, it is represented in floating point notation.

EXAMPLE:

```
:
:
110:N=N*3
120:A$=STR$ (N)
130:B$=LEFT$ (A$,1)
140:M=VAL (B$)
:
:
```

[110] The program performs some calculations on the numeric variable N.
[120] The numeric variable N is converted to the string variable A$. String variables are much easier to manipulate than numerics. In this example, imagine that the first digit of the number is required. Maybe it is a code for some further process. Having converted the number to a string we can use any of the string manipulation commands: INSTR, LEFT$, RIGHT$, MID$.
[130] This stores just the first digit of the number, or character as it is now treated by the program, in the string variable B$.
[140] The single digit is reconverted into a numeric variable so that it can be processed by the program as a number.

PRO
RUN
PROGRAM

TAB

FORMAT: 1. TAB <column>

Abbreviation:
See Also: LCURSOR, LPRINT

PURPOSE: In text mode, moves the printer pen to the specified column.

REMARKS: The TAB command is similar to the LCURSOR command; it moves the pen to the specified print column in the range 0 to maximum line length set with the current PCONSOLE statement. The print column locations change with the character size set by the CSIZE command. TAB can also be used within an LPRINT statement.

EXAMPLE:

```
5:TEXT
10:TAB 40
20:LPRINT "MIDDLE"
30:LPRINT "LEFT"
40:END
```

This program will only run in TEXT mode.

[10] Moves the pen carriage to the middle of the paper.
[20] This message will start printing from the current printer position in the middle of the page.
[30] After the last LPRINT, the printer head moves back to the left. Thus, this line prints in the normal left justified position.

TAN

PRO
RUN
PROGRAM

FORMAT: 1. TAN(X)

Abbreviation: TA.
See Also: ATN, COS, SIN

PURPOSE: Returns the tangent of the angle X.

REMARKS: This function returns the tangent of the angle X, where X is expressed in degrees, radians or as a gradient value, depending on which mode the computer is set to with the DEGREE, RADIAN or GRAD command. An error code is generated if X is 90°.

EXAMPLE:

```
10:DEGREE
20:PRINT "TAN OF 0 IS ";TAN (0)
30:PRINT "TAN OF 45 IS ";TAN (45)
40:END

>RUN
TAN OF 0 IS  0
TAN OF 45 IS  1
>
```

TEST

FORMAT: 1. TEST

Abbreviation: TE.
See Also:

PURPOSE: Runs a test on the printer in all colors.

REMARKS: TEST issues commands to the printer to draw four boxes in the four available colors in order of color number (0 to 3): black → blue → green → red. The computer is then set in text mode. To interrupt the test, press the BREAK key.

EXAMPLE:

```
>TEST
```

Starts the printer test to draw the 4 boxes.

TEXT

FORMAT: 1. TEXT

Abbreviation: TEX.
See Also: GRAPH

PURPOSE: Sets the printer to text mode.

REMARKS: The character size is set to 2 in text mode, and the origin is set at the left position of the pen.

Executing the TEXT command resets the PAPER command <limit from> and <limit to> parameters to their default values.

EXAMPLE:

```
>TEXT
```

The computer selects TEXT mode for the printer.

PRO
RUN
PROGRAM

TIME

FORMAT: 1. TIME
2. TIME = MMDDHH.mmss

Abbreviation:
See Also: TIME$

PURPOSE: Sets and returns the date and time for the computer's built-in clock.

REMARKS: TIME returns the current time in the format MMDDHH.mmss where MM is a two digit number between 01 and 12 indicating the month, DD is a two digit number between 01 and 31 indicating the day, HH is a two digit number between 00 and 23 indicating the hour, mm is a two digit number between 00 and 59 indicating the minutes, and ss is a two digit number between 00 and 59 indicating the seconds. The clock is unaffected by power off. It does not account for leap years.

TIME = MMDDHH.mmss sets the current time and date of the built-in clock in the same format as above.

EXAMPLE:

```
>TIME=031220.3215
```

Sets the current time and date to 8 : 32 pm and 15 seconds on March 12th.

TIME$

PRO
RUN
PROGRAM

FORMAT: 1. TIME$ = "HH:mm:ss"
2. TIME$

Abbreviation: TI.
See Also: DATE$, TIME$ ON/OFF/STOP, ON TIME$ GOSUB

PURPOSE: TIME$ is a system variable which contains the time of the computer's built-in real-time clock.

REMARKS: In the first format, as a statement, TIME$ sets the value of the real time clock. HH is a two digit number between 00 and 23 indicating the hour. mm is a two digit number between 00 and 59 indicating the minutes. ss is a two digit number between 00 and 59 indicating the seconds. Colons are used to separate the three entries.

In the second format, TIME$ returns the time of the real-time clock in HH:mm:ss format to the program as a string variable.

EXAMPLE:

```
>TIME$="09:15:00"
:
:
>PRINT TIME$
09:15:06
```

Sets the time and then prints out the time.

PRO
RUN
PROGRAM

TIME$ ON/OFF/STOP

FORMAT: 1. TIME$ ON
2. TIME$ OFF
3. TIME$ STOP

Abbreviation: TI.
See Also: TIME$, ON TIME$ GOSUB

PURPOSE: Disables or re-enables the ON TIME$ GOSUB statement.

REMARKS: TIME$ OFF disables all ON TIME$ GOSUB statements so that the subroutine calls are never made.

TIME$ ON re-enables all ON TIME$ GOSUB statements.

TIME$ STOP also disables all ON TIME$ GOSUB statements but when TIME$ ON is subsequently executed, if the time set in TIME$ = of an ON TIME$ GOSUB statement has been reached during the time that the TIME$ STOP statement was effective, a jump will be made to the subroutine immediately. The default value is TIME$ STOP.

EXAMPLE: See ON TIME$ GOSUB.

TITLE

PRO
RUN
RAM

FORMAT: 1. TITLE "S0:"
2. TITLE "S1:"
3. TITLE "S2:"
4. TITLE ?

Abbreviation: TIT.

See Also:

PURPOSE: Selects the program module in slot 1 or slot 2.

REMARKS: Up to two modules can be mounted in the slots in the back of the computer. This command selects which slot is currently selected.

TITLE: "S0:" selects the computer's internal RAM. This is the default value after an ALL RESET.
TITLE "S1:" selects the module in slot 1.
TITLE "S2:" selects the module in slot 2.
TITLE ? returns the number (0,1 or 2) of the currently-selected module.

If the module slot contains a RAM disk module instead of a program module, an error code message is displayed.

NOTE:
In some situations after executing TITLE during a halt in execution in a main memory program, the ↑ and ↓ keys will not list the program in the specified module. In this case, use LIST instead, or complete execution of the main memory program first.

EXAMPLE:

```
>TITLE "S0:"
>
```

Selects the main computer memory for usage after using a memory module in one of the slots.

PRO
RUN
PROGRAM

TRON / TROFF

FORMAT: 1. TRON
2. TROFF

Abbreviation: TR./TROF.
See Also: CONT, STOP

PURPOSE: Used to set or cancel the debugging program trace.

REMARKS: Tracing provides assistance in debugging programs. When the trace is on, the computer halts for 0.5 seconds after executing a program line and the line number is displayed on the right hand side of the screen.

If the ↓ key is pressed while the line number is displayed, the computer switches to single step execution, and the ↓ key must be pressed each time to execute the next line. Holding the ↓ key pressed causes execution to continue from line to line with no display of line numbers. The statement which has just been executed can be scanned along its length by pressing the ▶ or ◀ keys. Pressing the ↑ key returns to continuous execution (0.5 sec wait between lines).

When execution is halted at a PRINT or INPUT statement, pressing the ENTER key will resume execution.

If execution is interrupted by a STOP statement or by pressing the BREAK key, continuous trace execution can be resumed by pressing the ↑ key, or single step trace execution by pressing the ↓ key.

The trace can also be turned on and off from within a program. After TRON, the trace remains on for all subsequent execution until TROFF is executed.

EXAMPLE:

```
10:FOR I=1 TO 3
20:PRINT "I=";I
30:NEXT I
40:END

>TRON
>RUN
                              10
I=1                           20
                              30
I=2                           20
                              30
I=3                           20
                              30
>TROFF
>
```

Running this program after turning on the trace will display line numbers on the screen as the lines are executed. TROFF turns off the trace so that line numbers will not be shown in subsequent running of programs.

PRO
RUN
PROGRAM

VAL

FORMAT: 1. VAL(X$)
2. VAL("<string>")

Abbreviation: V.
See Also: STR$

PURPOSE: Converts a string of numeric characters into a decimal value.

REMARKS: The VAL function is the reverse of the STR$ function. It changes a string composed of numeric characters into a numeric value that can be used as a number in subsequent processing.

If the string is in decimal, it must be composed of the characters 0–9, with optional decimal point and sign. In this form. VAL is the opposite of the STR$ function.
If illegal characters are included in a string for conversion, the PC-1600 converts all characters up the first illegal one.

EXAMPLE:

```
10:INPUT "CYCLE FREQUENCY ";A$
15:IF ASC (A$)<48OR ASC (A$)>57THEN 100
20:F=VAL (A$)
30:PRINT F
40:STOP
:
:
100:PRINT "MUST BE A NUMBER":GOTO 10
```

[10] This inputs a string, which is to be converted to a numeric value.
[20] The string is converted to its numeric equivalent. Inputting a number as a string gives the programmer a chance to check the input to make sure it is of the correct type or in the correct range within the program itself.

WAIT

PRO
RUN
PROGRAM

FORMAT: 1. WAIT <duration> [,P]
2. WAIT <duration> [,S]
3. WAIT

Abbreviation: W.
See Also: GPRINT, LINE, PRINT

PURPOSE: Specifies the time for which execution is halted after execution of a PRINT, GPRINT or LINE statement.

REMARKS: WAIT is used to display data on the screen for a fixed time before continuing execution.

WAIT <duration> causes execution to halt after a PRINT, GPRINT or LINE statement for a duration specified in the range 0 to 65535, measured in units of approx. 1/64 of a second. Thus WAIT 64 halts execution with the result of the statement displayed on the screen for about 1 second.

The P and S options can only be specified in MODE 0 (PC-1600 mode).

The P option causes execution to wait after the PRINT, GPRINT or LINE statement for the specified duration. The S option specifies that the WAIT duration be applied to scrolling information displayed on the screen; that is, after one screenful of data (4 lines) has been printed out, the screen will not start to scroll up until after the duration specified. The default setting is P.

Specifying WAIT alone sets the wait duration to infinity; the computer will halt execution until the ENTER key is pressed.

When execution is started with a RUN command, the defaults are as follows:

0 in MODE 0 (PC-1600 mode); no pause in execution following a PRINT statement.

INFINITY in MODE 1 (PC-1500 mode); execution halted until the ENTER key is pressed.

EXAMPLE:

```
20:WAIT 200
30:PRINT "READ THIS QUICKLY!"
40:WAIT
50:PRINT "NOW ITS GONE"
60:END

>MODE 1
>RUN
```

[20] Sets the time (here about 3 seconds) that execution should halt for after each PRINT.
[30] Prints out the message which is overwritten by line 50 in about three seconds.
[40] Resets WAIT conditions to default values.
[50] The second message overwrites the first and stays on the screen (default value for WAIT).

WAKE$

PRO
RUN
PROGRAM

FORMAT: 1. WAKE$(0) = "<time>:<command string>"
2. WAKE$(1) = "<command string>"
3. WAKE$(0) = ""
4. WAKE$(1) = ""

Abbreviation: WAK.
See Also: KBUFF$

PURPOSE: Sets the wake time and command string for auto power-on.

REMARKS: In the first format, WAKE$ allows the computer to be programmed to turn on at a specified time and to execute a command on turning on. In conjunction with the KBUFF$ command, this allows automatic execution of batch files at a specified time. The <time> is specified in MM/DD/HH/mm format, using slashes.

In the second format, WAKE$ allows the computer to be programmed to turn on and execute a command when the CI signal line (pin 9) from the RS-232C interface goes "high", and execute the specified <command string>. See Part III Chapter 6 on the RS-232C interface.

The <command string> line must be terminated by the hexadecimal code for the enter key; that is, CHR$ (&0D) as in the example:

WAKE$ (0) = "10/03/08/00; RUN" + CHR$ (&0D)

Maximum command length is 26 characters. Formats 3 and 4 release the specifications of formats 1 and 2 respectively.

EXAMPLE:

```
>WAKE$(0)="12/25/07/30:RUN"+CHR$ (&0D)

10:PRINT "GOOD MORNING AND....."
20:PRINT "...MERRY CHRISTMAS!..."
:
:
```

WAKE$ will power up the computer on Christmas day at 7:30 am and will execute the current program in memory. Here, the program will print the greeting and continue.

PRO
RUN
PROGRAM

XCALL

FORMAT: 1. XCALL <address>[,<variable name>]

Abbreviation: XC.
See Also: NEW, POKE, XPOKE

PURPOSE: Used to call a machine language program.

REMARKS: The XCALL statement is basically the same as the CALL statement and transfers control to a machine language program or subroutine stored in memory.

The difference is that it is compatible with the command set in the PC-1500. When the program is called, a single variable value may be passed from the current BASIC program to the machine language program, and when execution of the machine language program is finished, the same single variable will be passed back to the calling program. The A register and the X register are used for this purpose. The variable to be passed must have been assigned before XCALL is executed. The machine language program must have been written into memory using the XPOKE statement before it can be called.

<address> specifies the lower address in the current memory bank where the machine language program is stored.

<variable name> specifies the variable whose value (integer) is to be passed to the machine language program on entry, and passed back to the BASIC calling program on exit if the carry flag is raised. It must be either a simple numeric or fixed numeric variable in the range −32768 to 32767. The value is passed to the X register, and the contents of the X register is passed back to the calling program with the same variable name on exit. If a two-character variable name is used, it must have been declared with the DIM statement or an error will result.

If <variable name> specifies a string variable, the X register holds the start address of the string location, and register A holds the string length.

EXAMPLE:

```
:
:
400:XCALL 57405,X
410:PRINT "THE VALUE OF X RETURNED"
420:PRINT "FROM THE MACHINE LANGUAGE IS ";X
:
:
```

[400] Control passes to the machine language held in memory with execution starting in memory location 80.
[410–420] XCALL passed across the numeric variable X and the returned value is printed out here.

PRO
RUN
PROGRAM

XPEEK

FORMAT: 1. XPEEK <address>
2. XPEEK #<address>

Abbreviation: XP.
See Also: PEEK, POKE, XPOKE

PURPOSE: Returns one byte of data from the specified address in memory.

REMARKS: The XPEEK command is basically identical to the PEEK command, but the format is compatible with the PC-1500 command set.

<address> specifies an address in memory area 0 (ME 0) to be accessed.

#<address> specifies an address in memory area 1 (ME 1) to be accessed. The memory bank used is the currently accessed bank.

For further details, refer to the memory map in the Appendices.

EXAMPLE:

```
>XPEEK100
                              37
>
```

Returns the data in memory address 100 (&64). The current value is 37.

XPOKE

PRO
RUN
PROGRAM

FORMAT: 1. XPOKE <start address>,<integer list>
2. XPOKE #<start address>,<integer list>

Abbreviation: XPO.
See Also: PEEK, POKE, XPEEK

PURPOSE: Writes one or more data bytes into memory starting from the specified address.

REMARKS: The XPOKE command is basically identical to the POKE command, but the format is compatible with the PC-1500 command set.

<start address> specifies the address where the first byte will be written in memory area 0 (ME 0). The following bytes are written to subsequent addresses.
#<start address> specifies the address in memory area 1 (ME 1).

<integer list> specifies the list of integer values in the range 0 to 255 (&0 to &FF) to be written into consecutive addresses from the specified address. The items in the list are separated by commas.

The memory bank used is the currently accessed bank.

For further details, refer to the memory map in the Appendices.

EXAMPLE:

```
>XPOKE100,255,255
>
```

Starting in memory address 100 (&64) the two values, 255 are written to memory. 255 is &FF. Thus the result will be to set 16 bits of memory to "1". Do not experiment with this command unless you have studied the memory map and you know where you are writing to. This example is only a demonstration and will cause problems if run! However, an ALL RESET will recover from errors resulting from bad memory mapping.

PART V

APPENDICES

A. REPLACING THE BATTERIES
B. REPLACING THE RAM MODULES
C. CHARACTER CODE TABLES
D. MEMORY MAPS
E. MACHINE LANGUAGE PROGRAMS
F. ERROR CODES FOR THE PC-1600
G. BASIC COMMAND LIST
H. COMPATIBILITY WITH PC-1500 MODEL AND PERIPHERALS
I. CARE & TROUBLESHOOTING
J. SPECIFICATIONS

Appendix A
Replacing the Batteries

Your PC-1600 has two ways to warn you that its batteries are getting low:

- A low battery indicator lights on the status line at the top of the PC-1600's display as the symbol BATT. The BATT symbol may also come on if the batteries in the printer are low when it is connected to the computer. See the section on BATT.

- During execution, the computer may return an ERROR code on the screen as a low battery warning. Note that ERROR codes may also come up for low batteries in connected devices such as the printer. See the summary of ERROR codes in Appendix F and the ERROR code lists in the sections on optional devices in Chapter 6.

Note that the Memory Safe Guard feature that preserves any data in the computer's memory when you turn off the power is not effective when the batteries are dead or replaced. This includes both the computer's internal memory and that in memory modules when installed. Program modules are protected against this loss of data by their built-in backup batteries.

If you have no data currently in memory that needs saving, turn to the section on Installing the Batteries. Replace the batteries and note the instructions there, then go to the Resetting the Computer section and follow the ALL RESET procedure to initialize the computer and clear the memory.

If you do have data in memory that must be preserved, there are two ways to prevent loss of the data when you replace the batteries:

- Connect an AC adaptor to the computer to maintain the power supply while replacing the batteries. Follow these steps:

    1 Turn off the computer's power.
    2 Connect the AC adaptor first to the outlet, then to the computer.
    3 Replace the batteries.
    4 Unplug the AC adaptor.
    5 Turn on the computer and resume operation. Data and programs in memory are intact. No need to reset the computer.

The second procedure is somewhat more complicated.

- Save the data or program in memory to floppy disk, tape, or RAM disk before replacing the batteries. After installing new batteries, load the data or program back into memory to continue working. Follow these steps:

    1 If an output device is not connected, turn off the computer's power.
    2 Connect the optional disk drive, cassette recorder, or program (RAM disk) module.
    3 Turn on the computer and use the SAVE commands (see BASIC Command Dictionary) to write the data from memory to one of the devices mentioned.
    4 Turn off the power and replace the batteries.
    5 Turn to the section on Resetting the Computer and follow the procedure to initialize the computer.
    6 Use the LOAD commands to load the data back into the computer's memory from the device to continue operation.

Appendix B
Replacing the RAM Modules

There are two types of RAM modules: program modules with a built-in backup battery to preserve the contents of the memory when removed, and memory modules that are not backed up and are used as additions to the computer's internal user memory area.

Program modules don't require special treatment for replacement and can be removed and installed as needed.

Data in memory modules is lost when they are removed from the computer.

After you install a memory module and turn on the computer, the message NEW 0 ?: C H E C K comes on the screen to tell you that the computer recognizes a change in the amount of memory with the addition of the module. To clear the entire memory, both the computer's internal RAM and the RAM module, press CL and make sure that the mode is set to PROgram. Press the MODE key if a change is necessary. Then type in N E W 0 and press ENTER. This message is not displayed nor is clearing needed when the installed module is a program module. Note that the NEW command clears the contents of memory, but will not clear RESERVE mode function key strings if executed in PRO mode. Therefore, after installing a module, the RESERVE area may contain uncleared function key strings. Clear these by executing NEW in the RESERVE mode.

Until specified with the TITLE command, program modules are independent of the internal user RAM when installed.

You can use the MEM statement in BASIC to display the amount of free user memory after installing a RAM module or at any other time. Type in M E M and press ENTER. A figure will be displayed on the right end of the next line showing the number of bytes of free user memory available. For instance, without modules in the slots, entering M E M returns 11834, the number of bytes available in the computer's internal RAM without data.

When a memory expansion module is installed, BASIC text is written into expansion memory areas in the order S2: → S1: → Internal RAM.

Appendix C
Character Code Tables

Mode 0 Character Code Table

In Mode 0 the PC-1600 character set includes graphic and Greek symbols and international characters in addition to the normal upper and lower case letters, numbers and symbols. This character set is similar to the IBM PC character set.

Hex	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL			0	@	P	`	p	Ç	É	á	▒	└	╨	α	≡
1			!	1	A	Q	a	q	ü	æ	í	▒	┴	╤	ß	±
2			"	2	B	R	b	r	é	Æ	ó	▒	┬	╥	Γ	≥
3			#	3	C	S	c	s	â	ô	ú	│	├	╙	π	≤
4			$	4	D	T	d	t	ä	ö	ñ	┤	─	╘	Σ	⌠
5			%	5	E	U	e	u	à	ò	Ñ	╡	┼	╒	σ	⌡
6			&	6	F	V	f	v	å	û	ª	╢	╞	╓	μ	÷
7	BEL		'	7	G	W	g	w	ç	ù	º	╖	╟	╫	τ	≈
8	BS		(	8	H	X	h	x	ê	ÿ	¿	╕	╚	╪	Φ	°
9	HT		)	9	I	Y	i	y	ë	Ö	⌐	╣	╔	┘	θ	•
A	LF		*	:	J	Z	j	z	è	Ü	¬	║	╩	┌	Ω	·
B	VT		+	;	K	[	k	{	ï	¢	½	╗	╦	▒	δ	√
C	FF		,	<	L	\	l	¦	î	£	¼	╝	╠	▒	∞	n
D	CR		-	=	M	]	m	}	ì	¥	¡	╜	═	▒	ø	2
E	SO		.	>	N	^	n	~	Ä	₧	«	╛	╬	▒	ε	■
F	SI		/	?	O	_	o	▒	Å	ƒ	»	┐	╧	▒	∩	

PC-1600 Character Code Table

To look up the hexadecimal code corresponding to a character in the table, write down the number at the head of the column in which the character appears followed by the number on the left of the row in which the character is. The hexadecimal code for "a" is therefore &61.

Any character in the table can be printed out to the screen using the CHR$ (<character code>) function in BASIC. Refer to CHR$ in the Command Dictionary for details.

International Character Set

The international characters in the character code table from code value &80 to code value &A8 are assigned to the alphabetic keys on the keyboard on pressing the KBII button to the right of the six function keys. A template is provided with the international characters shown above the corresponding key positions.

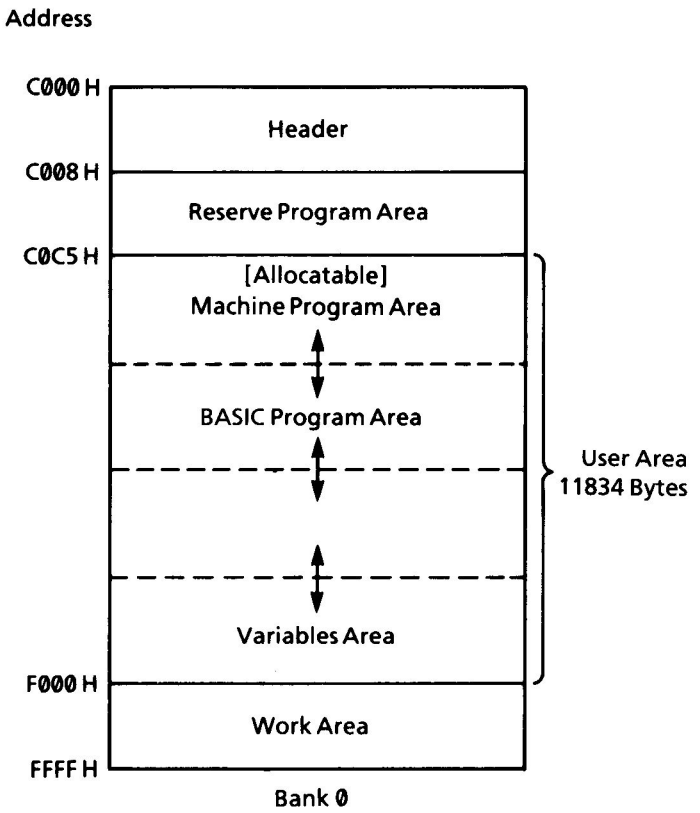
Mode 1 Character Code Table

In Mode 1, the PC-1600 character set is modified to make it compatible with the more limited character set in the PC-1500. The table below gives the hexadecimal value of the character position, and the corresponding character for Mode 1.

HEX CODE	27	5B	5C	5D	5E	5F	60	7B	7C	7D	7E	7F
MODE 0	'	[	\	]	^	_	`	{	¦	}	~	▒
MODE 1		√	¥	π	^						~	

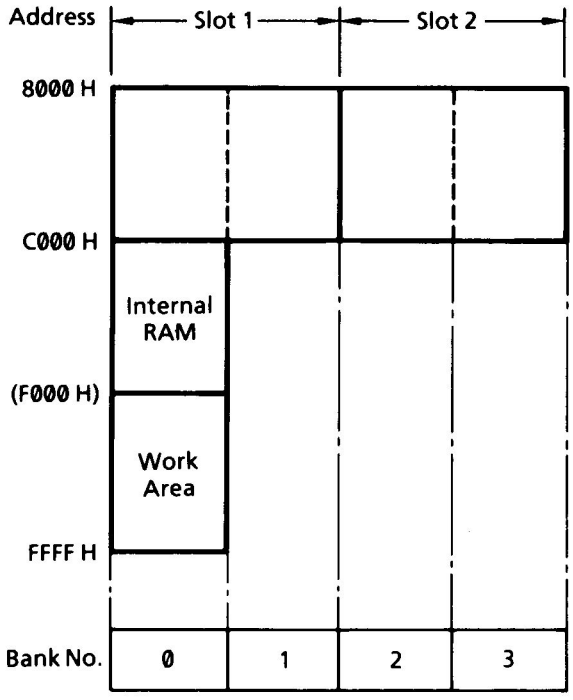
Appendix D
Memory Maps

The PC-1600 can address 8 memory banks (bank 0 to bank 7). The first four banks are allocated to RAM. Banks 4 to 6 hold internal system ROMs and peripheral memory. Bank 7 is unused, but is addressable.

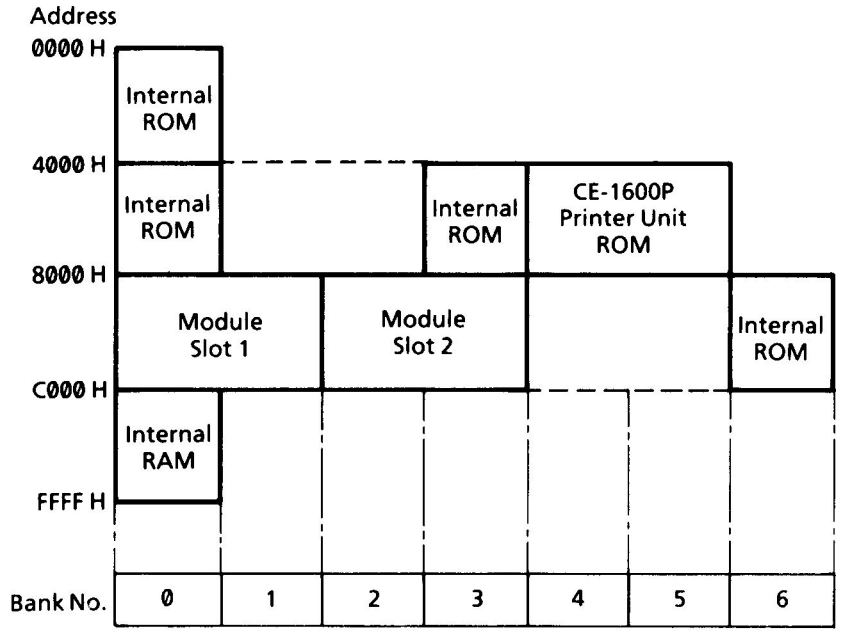


Internal RAM [Viewed from Z-80A Processor]

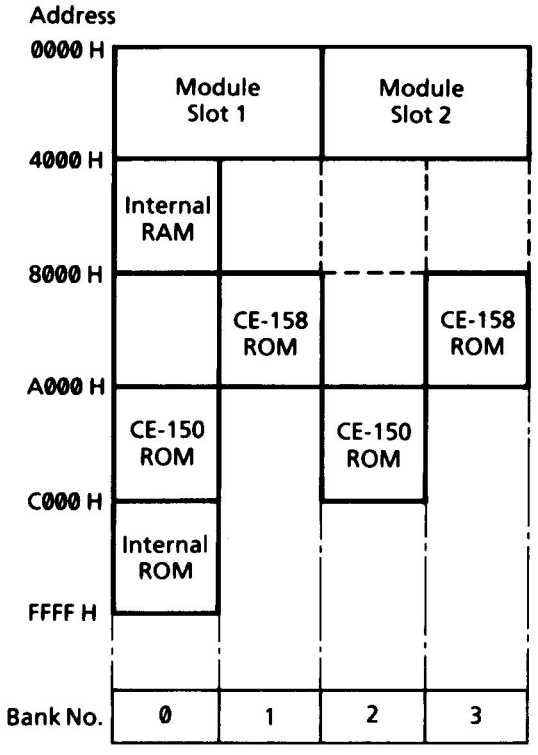
With no modules in the expansion slots, the internal RAM has 11834 bytes of user area. The above memory map shows some of the user area allocated for machine language programs. This machine program area can be set to 0 with the NEW Command.



Internal RAM and Expansion Slots (banks 0 to 3)
[Viewed from the Z-80A Processor]



Overall Memory [Viewed from Z-80A processor]



Overall Memory [Viewed from LH-5803 Sub-Processor]

Appendix E
Machine Language Programs

The PC-1600 can be programmed directly in Z-80A Assembler code by the advanced programmer.

BASIC Machine Language Related Commands

BASIC supports a number of commands to load, save, access and call machine language routines, or to control the machine I/O ports directly. Some of them address the main Z-80A processor, and some address the LH-5803 sub-processor. They are:

BLOAD, BSAVE, CALL, CLOADM, CSAVEM, INP, OUT, PEEK, POKE, XCALL, XPEEK, XPEEK#, XPOKE, XPOKE#

Refer to the Command Dictionary for detailed descriptions.

Memory Allocation for Machine Language Programs

Internal RAM can be allocated for machine language programs with the NEW command, which sets the lower address of the BASIC program area. The user can also access system utilities in other memory areas, or the peripheral device ROMs, but this needs a detailed knowledge of the memory allocation of the PC-1600 above that covered in this manual. Refer to the memory maps in Appendix D.

Compatibility with the PC-1500

The following table lists the PC-1600 machine language related commands which are different from the PC-1500 command set:

COMMAND NAME		FUNCTION
PC-1600	PC-1500	
XCALL	CALL	Runs machine language program for LH-5801/3 sub-processor.
CALL		Runs machine language program in PC-1600's main processor (Z-80A).
XPOKE	POKE	Writes data to LH-5801/3 memory space.
POKE		Writes data to main Z-80A processor memory space.
XPEEK	PEEK	Reads data from LH-5801/3 memory area.
PEEK		Reads data from main Z-80A processor memory area.
XPOKE#	POKE#	Sends data byte to LH-5801/3 machine I/O port.
XPEEK#	PEEK#	Returns data byte from LH-5801/3 machine I/O port.

Appendix F

Error Codes for the PC-1600

Error Codes	Description

PROGRAM ERRORS

1	Syntax error. Statement does not conform to the rules of Sharp BASIC. Program command used in direct mode or vice versa when using with PC-1500 peripherals (CE-150, CE-158, CE-162E) in Mode 1. Line number out of range. Line number in program exceeds 65279.
2	NEXT without FOR. A NEXT statement was encountered without a corresponding FOR statement. This could be due to improperly nested FOR/NEXT loops, the variable in a NEXT statement not being the same as the variable specified in the FOR statement, or more than one NEXT statement being specified for a single FOR statement.
4	READ without DATA. A READ statement was encountered, but there are no DATA statements in the program. Out of data. There are no more data items to read on the DATA line.
5	Array variable declared twice. The array variable being declared now has been declared previously.
6	Undeclared array variable. The array variable encountered in the program has not been declared with a DIM statement.
7	Type mismatch. The variable type does not match the data type. An attempt made to assign a numeric value to a string variable or vice versa.
8	More than 2 dimensions declared for an array variable.
9	Subscript out of range. The subscript specified in a statement is out of the range declared in the DIM statement.

Error Codes	Description
10	Memory variable area full. No room to store any more variables.
11	Undefined line number in GOTO, ON..GOSUB or ON..GOTO statements. The line number specified for the jump does not exist.
12	Incorrect format string specified in USING statement.
13	Memory full. Insufficient user memory available for program size or for storage of string variables in program.
14	Stack full. FOR/NEXT loops nested too deeply in the program. Stack full during computation of a function.
15	Stack full. Subroutines nested too deeply in the program. String length longer than 80 bytes.
16	Exponent greater than 99 found.
17	Illegal data types in arithmetic statement. Attempt to perform arithmetic operations on a string variable.
18	Incorrect specification of parameters in a function.
19	Parameter out of range.
20	Incorrect specification of fixed variable name.
21	Numeric expression used instead of variable.
22	Memory full. Can't load program.
23	Incorrect specification of time in TIME string.
24	Attempt made to save, list or edit a program on which a password has been set with the PASS command.
25	Invalid address specified with NEW statement.
26	Invalid mode. Cannot execute the command in current mode.

Error Codes	Description
27	Illegal command. The peripheral addressed by this command is not connected.
28	An INPUT or AREAD statement has encountered illegal string data which duplicates a command name or function name.
30	Line number out of range. Line number in program exceeds 65539 when PC-1600 is used in Mode 1 with PC-1500 peripherals.
32	Cursor position set with GCURSOR prevents display of data entered in response to current INPUT statement.
35	PC-1500 peripheral not connected.
36	Unable to display data as specified in format string in USING statement.
37	Overflow. Result of computation exceeds 9.999999999E99.
38	Division by zero. Zero used as a divisor because a variable or expression was zero at that point in the program. Undefined variable used as divisor.
39	Illegal function call. A negative number specified as the argument of a SQR or LN function; a decimal argument specified instead of an integer in certain functions.
100	Renumbering unsuccessful with RENUM command.
101	Invalid device name specified with TITLE or NEW statements.
102	Invalid device selection (device not connected).
103	Memory full for RAM module. INIT parameters cannot be set.
104	More than eight software interrupts specified in program (ON GOSUB statements).

Error Codes	Description
105	Too many software interrupt levels.
106	RETI statement found with no corresponding ON..GOSUB statement.
107	RESUME statement found with no corresponding ON ERROR GOSUB statement.
108	Password cannot be cleared.
109	Illegal variable name in ERASE statement.
110	Cannot set MODE 1 (PC-1500 mode). Command invalid in MODE 0. Commands associated with PC-1500 peripherals will only work in MODE 1.
111	Invalid mode on PC-1600.
112	Line too long. During progam editing, length of logical line exceeds 80 characters.
131	String data preceded by + or − sign.

CASSETTE TAPE ERRORS

40	Syntax error in tape command.
42	Insufficient memory available. Not enough memory space to load the program.
43	Tape verification error. After executing CLOAD?, the copy in memory does not agree with the original on tape. Possible tape misread. Load and verify again. Type mismatch with INPUT# statement. Variable type and data type do not agree. Attempt to read string data into a numeric variable or vice versa.
44	Tape error. Tape recorder incompatible; tape read error; recorder settings need adjustment (tone and volume).

Error Codes	Description

PRINTER ERRORS

70	Pen outside range −2048 to 2047 for X or Y.
71	Reverse form feed more than 10.24 cm in TEXT mode. (Only for CE-150 Printer & Cassette Interface.)
72	Incorrect specification of TAB parameter.
73	Illegal command in current mode. Graphics command used in TEXT mode or vice versa.
74	Too many parameters in LLINE or RLINE statement.
76	Line output to printer is longer than line length set with PCONSOLE, or result of a numerical computation too long to fit on one line in TEXT mode.
78	Cannot execute LPRINT or LLINE command. Pen not in place or printer locked due to low battery voltage.
79	Color signals not output to printer. (Only for CE-150 Printer & Cassette Interface.)
80	Low battery. Printer locked up due to low battery voltage.

SERIAL I/O PORTS [COM1: and COM2:]

140	Invalid parameters set in SETCOM statement.
141	Size of receive buffer specified in INIT statement is too large (greater than 16383 bytes or greater than available memory).
142	Error in data reception through a serial port (parity error, overrun error, framing error, receive buffer full error).
143	Timeout error. No response from other party for longer than the timeout value set in the RCVSTAT statements.
144	Serial port specified in SETDEV statement is already open.

Error Codes	Description

FILE ERRORS

150	Too many files specified in MAXFILES statement.
151	File already exists. Use another file name.
152	File not found. Check that file name specified is correct. Disk full error when saving to disk with SAVE command.
153	Incorrect file number specification in file read or write statement. File with specified file number has not been opened.
154	File already open. Close it first and reopen in the new mode.
155	Illegal drive name, or drive not connected.
156	Incorrect parameter specification in SET statement.
157	Illegal file name, or incorrect specification.
158	Command not supported. Illegal command for disk drive.
159	Attempt made to write to a disk which is write-protected.
160	Disk not inserted in specified drive.
161	Disk has not been formatted with the INIT statement.
162	Disk read or write error.
163	Wrong disk in drive. Disks have been changed while a file was open.
164	Disk full error.
165	End of file reached with INPUT# statement. All data has been read.
166	Insufficient memory space for disk internal work area. Not enough free space in the PC-1600's memory area for the disk drive I/O system.

Error Codes	Description
167	Fatal disk error. Contents have been destroyed or otherwise corrupted.
168	CE-1600F Disk Drive not functioning correctly. Low battery or hardware fault.

Appendix G

Basic Commands List

Command Name	Description
ABS	Returns absolute value of X
ACS	Returns arc cosine of X
ADIN ON/OFF/STOP	Enables/disables analog interrupts
AIN	Returns analog input level
ALARM$	Sets alarm time and message
AREAD	Reads a variable from the screen
ARUN	Starts program execution automatically
ASC	Returns the character code for a string
ASN	Returns arc sine of X
ATN	Returns arc tangent of X
AUTO	Turns on automatic line numbering
BEEP	Generates sound through internal speaker
BEEP ON/OFF	Enables/disables sound generation
BLOAD	Loads a machine language program from disk or tape
BREAK ON/OFF	Enables/disables BREAK key
BSAVE	Saves a machine language program to disk or tape
CALL	Calls a machine language program
CHAIN	Loads and executes a BASIC program on tape
CHR$	Returns a character from its ASCII code
CLEAR	Erases all variables in memory
CLOAD	Loads a BASIC program from tape (PC-1500 mode)
CLOAD?	Verifies loading from tape (PC-1500 mode)
CLOAD M	Loads a machine language program from tape (PC-1500 mode)
CLOSE	Closes a device file
CLS	Clears the display screen
COLOR	Sets the pen color for the printer
COM$	Returns the communication parameters
COMn ON/OFF/STOP	Enables/disables communication interrupts
CONT	Resumes execution after STOP or BREAK
COPY	Copies a file on disk or tape
COS	Returns the cosine of X
CSAVE	Saves a BASIC program to tape (PC-1500 mode)
CSAVE M	Saves a machine language program to tape (PC-1500 mode)
CSIZE	Sets the size of the printer characters
CURSOR	Positions the cursor on the screen
DATA	Lists data items for a READ statement
DATE$	Returns the date
DEG	Converts degrees to decimal
DEGREE	Sets the computer in DEGREE mode

Command Name	Description
DELETE	Deletes program lines
DIM	Reserves memory space for variables/arrays
DMS	Converts decimal degrees to deg/min/sec
DSKF	Returns free space on a disk device
END	Stops execution of current program
EOF	Returns value to show end of file
ERASE	Erases specified variables and arrays
ERL	Returns line number of an error
ERN	Returns error code number for an error
EXP	Returns exponential e raised to power X
FILES	Displays directory information for disk
FOR..NEXT	Allows repeated execution of program lines
GCURSOR	Positions graphics cursor on screen
GLCURSOR	Positions printer pen in graphics mode
GOSUB..RETURN	Jumps to a subroutine
GOTO	Unconditional jump to a line number
GPRINT	Draws bit-image graphics on display screen
GRAD	Sets computer in GRADIENT mode
GRAPH	Sets printer in graphics mode
HEX$	Returns hexadecimal string for a number
IF..THEN	Conditional jump
INIT	Initializes module, disk; sets receive buffer
INKEY$	Reads character from keyboard buffer
INP	Returns data from microprocessor port
INPUT	Inputs data from keyboard to a program
INPUT#	Reads records from a file
INSTAT	Returns control signal states for serial port
INSTR	Searches for a character in a string
INT	Truncates decimal part of a number
KBUFF$	Writes characters into keyboard buffer
KEY ON/OFF/STOP	Enables/disables function keys
KEYSTAT	Sets key repeat, key click functions
KILL	Erases a file on disk
LCURSOR	Moves printer pen to specified position
LEFT$	Returns characters from left end of string
LEN	Returns number of characters in string
LET	Assigns a value to a variable
LF	Feeds paper in printer
LFILES	Prints out directory information to printer
LINE	Draws line between points on screen
LIST	Lists a program to the screen
LLINE	Draws line between points on printer
LLIST	Lists a program to the printer
LN	Returns the natural logarithm of X

Command Name	Description
LOAD	Loads a file from disk or tape to memory
LOC	Returns number of records accessed in a file
LOCK/UNLOCK	Enables/disables MODE key
LOF	Returns size of a file on disk
LOG	Returns the common logarithm (base 10) of X
LPRINT	Outputs data to the printer or serial port
MAXFILES	Sets maximum number of files for a program
MEM	Returns unused memory space in user area
MERGE	Merges program on tape into memory
MID$	Returns a string from inside another string
MOD	Returns remainder of division
MODE	Selects screen mode for PC-1500 compatibility
NAME	Renames a file on disk
NEW	Clears memory/assigns space for machine code
ON ADIN GOSUB	Jumps for an analog interrupt
ON COMn GOSUB	Jumps for a serial port interrupt
ON ERROR GOTO	Jumps to error processing routine on error
ON..GOSUB/ON..GOTO	Multiple conditional jump
ON KEY GOSUB	Jumps for a function key input
ON PHONE GOSUB	Jumps for a telephone modem input
ON TIME$ GOSUB	Jumps at specified time
OPEN	Opens a file for access
OUT	Writes data to a microprocessor port
OUTSTAT	Sets control signal states for serial ports
PAPER	Sets paper type and vertical print range
PASS	Sets/releases password
PAUSE	Displays data on screen for fixed time
PCONSOLE	Sets print format/EOL code for printer/ports
PEEK	Returns data byte from memory (PC-1600 mode)
PHONE ON/OFF/STOP	Enables/disables RS-232C interrupts
PITCH	Sets character pitch/line spacing for printer
POINT	Returns dot setting at a point on the screen
POKE	Writes data byte to memory (PC-1600 mode)
POWER	Sets auto power off
PRESET	Resets a dot at a point on the screen
PRINT	Outputs data to the display screen
PRINT#	Writes data to a file
PSET	Sets/resets a dot at a point on the screen
PZONE	Sets print zone for printer or serial port
RADIAN	Sets the computer in RADIAN mode
RANDOM	Initiates random number generation
RCVSTAT	Sets receive protocol/timeout for serial port
READ..DATA	Reads data into program from DATA line
REM	Used to include comments in a program

Command Name	Description
RENUM	Renumbers program lines
RESTORE	Used to re-read data from DATA lines
RESUME	Resumes execution after an error routine
RETI	Returns from an interrupt subroutine
RIGHT$	Returns characters from right end of string
RLINE	Draws line in relative coordinates on printer
RMT ON/OFF	Enables/disables remote control of tape
RND	Generates a random number
ROTATE	Sets print orientation and print head travel
RUN	Starts execution of a program
RXD$	Returns current data from serial port
SAVE	Saves a BASIC program to disk or tape
SET	Sets write protection on a disk file
SETCOM	Sets communication protocol for serial ports
SETDEV	Selects a serial port for output
SGN	Returns the sign of expression X
SIN	Returns the sine of X
SNDBRK	Sends break characters to a serial port
SNDSTAT	Sets send protocol/timeout for serial port
SORGN	Sets current pen position as origin
SQR	Returns the square root of X
STATUS	Returns amount of free space for memory areas
STOP	Halts execution during program debugging
STR$	Converts numeric data into string data
TAB	Moves printer pen to specific column
TAN	Returns the tangent of X
TEST	Runs a test on the printer
TEXT	Sets the printer in TEXT mode
TIME	Sets/returns the time of the built-in clock
TIME$	Returns time of built-in clock as a string
TIME$ ON/OFF/STOP	Enables/disables clock interrupts
TITLE	Selects computer's memory area
TRON/TROFF	Sets/cancels debugging program trace
VAL	Converts numeric strings to decimal values
WAIT	Sets wait time after a PRINT statement
WAKE$	Sets time/command string for auto power on
XCALL	Calls machine language program (PC-1500 mode)
XPEEK/XPEEK#	Returns data byte from memory (PC-1500 mode)
XPOKE/XPOKE#	Writes data byte to memory (PC-1500 mode)

Appendix H

Compatibility with PC-1500 Model Peripherals

BASIC programs written on the PC-1500 can in general be run on the PC-1600 in MODE 1. The majority of the general BASIC command names are the same between the two models; but there are differences in default settings for the printer paper width which must be corrected for programs which output to the printer. The PC-1600 commands associated with floppy disk and RAM disk files, and serial port interrupt commands are not supported on the PC-1500. Programs for the PC-1500 stored on tape can be loaded and run on the PC-1600. However, there are a few points which should be kept in mind, especially with regard to commands which access memory directly. The following command names are different between the two models:

Equivalent Command Names

PC-1600	PC-1500
TAB	LCURSOR
LLINE	LINE
LINE	no equivalent
XCALL	CALL
CALL	no equivalent
XPOKE	POKE
POKE	no equivalent
XPEEK	PEEK
PEEK	no equivalent
XPOKE#	POKE#
XPEEK#	PEEK#

Note: Refer to Appendix E for details on the machine language related commands and their compatibility.

It is not possible to specify TIME = 0 on the PC-1600, although this is used in many PC-1500 programs to reset a time counter.

Emulating the PC-1500

When the PC-1600 is set in MODE 1 with the MODE command, it emulates the PC-1500; only the bottom line on the screen is active, and the screen is controlled as for the PC-1500. The character set also changes from the PC-1600 character set to that used in the PC-1500; the symbols corresponding to certain codes change. Refer to Appendix C for the character code tables.

Typing a PC-1500 BASIC Program into the PC-1600

1 Type in the program in the PRO mode as normal, making sure that the LINE command is changed to the LLINE command wherever it occurs.

2 Save the program on any device, but it is wise to include a comment line at the head of the program to remind you that this is a PC-1500 program.

Programs Stored on Cassette Tape

PC-1500 programs stored on tape via the PC-1500's CE-150 printer/cassette interface unit can be loaded into the PC-1600 in MODE 1 with no modifications using the CLOAD command.

However, PC-1600 programs stored on tape via the PC-1600's CE-1600P printer with cassette interface unit cannot be loaded into the PC-1500. This is because the tape format is different in some respects, and memory allocation in the two computers is different.

Running a PC-1500 BASIC Program on the PC-1600

1 Ensure that there is a module in slot 1 or slot 2. This is to make sure that memory space is available as needed. One of the following types should be used, and the size should be less than 16K bytes: CE-151, CE-155, CE-159, CE-161. If there is no module in either slot, an error code will be displayed.

2 Set the PC-1600 in MODE 1 with the MODE command.

3 Run the program in the normal way. Results will be displayed on the screen exactly as on the PC-1500.

Printout to the CE-1600P Printer Unit

All the PC-1500's printer commands will work on the CE-1600P Printer, but because of the difference in paper width between the CE-150 and the CE-1600P printer units, if you want the output format to be the same as it was on the CE-150 printer, you will need to set the paper width. Type in the following commands in direct mode, or add them to the front of your program:

```
CSIZE 2
PZONE "LPT1:",18
PCONSOLE "LPT1:",18,0,0
```

Programs stored on tape using the PC-1500 with the CE-150 interface unit can be loaded into the PC-1600 via the CE-1600P interface unit and will run with no problems. But if the original program contains LCURSOR statements, when listing out on the CE-1600P, LCURSOR will list as the ~ symbol. To get round this problem, you can edit the program on the PC-1600 by replacing LCURSOR with the TAB function.

Using the PC-1600 with the CE-150 Printer/Cassette Interface

The PC-1600 can be mounted on the CE-150 Printer/Cassette Interface Unit, or with the CE-158 and CE-162E Interface Units for the PC-1500. Set in MODE 1, the computer will function exactly like the PC-1500. However, when PC-1600 programs are listed out via these devices, the PC-1600 command names will be printed out as their equivalent PC-1500 commands. This is because the internal software built into these units contains only the PC-1500 command names. This means, for instance that any LLIST commands in a program will print out on the program listing as LIST. When printing to the CE-150 Printer, the PRINT switch must be set to the • side. Manual calculations cannot be printed out.

PC-1500A Compatibility

The area in internal RAM (&7C00 to &7FFF) which is free user area in the PC-1500A is not available to the user in the PC-1600; it is used by the system. For this reason the PC-1500A model is not compatible with the PC-1600.

Compatibility with Early Model PC-1500

In some earlier model PC-1500's the value held in the FOR... NEXT counter gives a result 1 greater than on the PC-1600 after execution of a FOR... NEXT loop. For example, in the following loop:

```
10 FOR K=1 TO 10
20 NEXT K
30 PRINT K
```

The value of K printed will be 11 on the PC-1600, but will be 10 on early model PC-1500's.

Also, in the following example:

```
10 S=0
20 FOR K=1 TO 10 STEP 4
30 S=S+1
40 NEXT K
50 PRINT S
```

The value of S will be 3 on the PC-1600 and 4 on early model PC-1500's.

The early model PC-1500 also evaluates the IF... THEN statement differently. The value of the expression in the IF... THEN statement is evaluated as TRUE or FALSE according to the following table.

MODEL	Evaluated as	
	TRUE	FALSE
PC-1600 PC-1500	$\neq 0$	0
Early model PC-1500	>0	<0

To determine if a PC-1500 is one of these early models, look into address &C5C0 using the PEEK Command: PEEK &C5C0. If the value returned is 6, then you have an early model PC-1500. If the value is not 6 then there is no problem with FOR... NEXT loops.

Appendix I
Care & Troubleshooting

Your SHARP PC-1600 Pocket Computer is a precision device which deserves the best care at all times. Throughout this manual we've provided a number of notes and cautions anticipating where you might be likely to have difficulty. These points have covered a wide range of topics during operation, handling and programming.

This section provides some hints on what you need to do to keep your PC-1600 operating properly — and what to do when it doesn't.

The following is a list of key points on general care for your PC-1600.

1 **Display** The LCD screen on your PC-1600 is made of glass. It's fragile and can crack if mistreated. Be sure to keep the computer in its protective soft case when you're not using it. Be especially careful not to strike the screen with cable connectors and accessories when using peripherals.

2 **Work and Storage Areas** Avoid places of extreme temperature and humidity changes for both use and storage. The heat from prolonged exposure to direct sunlight can severely damage the computer. Moisture is also an enemy in any form. During dry winter months, be careful of static charge buildup. Never touch module pins or connectors.

3 **Cleaning** Use only a soft, dry cloth to clean the computer and peripherals. Never use soaps, commercial cleaners or any liquids.

4 **Battery Leakage** Always remove the batteries before long-term storage of the computer to prevent damage from leaks and chemical reactions.

5 **Service** When your PC-1600 is in need of repair, be sure to contact your SHARP dealer or service center. Unauthorized service or alteration of the computer may cause additional problems.

Your PC-1600 responds in a number of ways to directly prevent problems or to alert you to possible trouble with such features as ERROR codes, CHECK messages, auto power-off, Memory Safe Guard, and low battery warning. Be sure to check this operation manual and the manuals of any options you might be using when trouble occurs.

The following checklist gives you a guideline for working out the most common problems. Also see Chapter 13 on BASIC debugging when you have difficulty with programs.

We suggest that you try to solve a problem on your own and leave dealer contact as a last measure. You may find that you inadvertently caused the problem, and correcting it yourself means learning more about your SHARP PC-1600.

Troubleshooting Hints

If this happens ...	Do this ...
Computer was turned on, but screen is blank	1 Press **OFF** and **ON** again to double-check that power is still on. 2 Check for low battery indicator. 3 If you're using AC adapter, is it properly plugged in? 4 Adjust contrast control dial.
Display is OK, but keys are ineffective.	1 Press **CL** to clear. 2 Turn computer off and on again. 3 Press RESET switch alone for a simple reset. 4 Press **ON** key and RESET switch combination for ALL RESET.
Typed-in calculation is displayed in BASIC statement format with (:) after first number.	Press **MODE** key to change from PROgram mode to RUN mode for direct calculation.

Appendix J
Specifications

Microprocessors	
Main CPU	SC7852 (Z-80A compatible CMOS; 3.58 MHz)
Slave CPU	LH5803 (PC-1500 compatible; 1.3 MHz)
Sub CPU	LU57813P (307.2 kHz)
Display	Liquid Crystal Display; adjustable contrast
Alphanumeric characters	26 columns × 4 lines
Graphics	156 × 32 dots
Keyboard	63 alphanumeric keys; 6 function keys
Memory	
ROM	96K bytes
RAM	16K bytes (expandable to 80K bytes) User area – 11834 bytes
Expansion RAM	2 slots at rear for expansion modules
Interfaces	RS-232C serial I/O, optical serial I/O, analog input
Features	Low battery display Built-in real-time clock Alarm and auto power on/off functions External interrupts supported Communications functions
Power requirements	6 V ⎓ (DC) Four dry type batteries (SUM-3, AA or R6) AC adaptor (EA-160 or EA-150)
Power consumption	0.48 W
Battery	25 hours at 20°C (68°F): 10 mins processing time, 50 mins display time per hour.
Dimensions	195 (W) × 86 (D) × 25.5 (H) mm 7 11/16″ (W) × 3 3/8″ (D) × 1″ (H)
Weight	Approx. 390 g (0.86 lbs) with batteries
Operating temperature	0°C to 40°C (32°F to 104°F)
Accessories	Soft case Two keyboard templates Four batteries Operation manual

INDEX

— A —

— B —

— C —

— D —

— E —

— F —

— G —

— H —

— I —

— J —

— K —

— L —

— M —

— N —

— O —

— P —

— Q —

— R —

— S —

— T —

— U —

— V —

— W —

— X —

— Y —

— Z —

MODEL PC-1600 OPTIONAL BOARD AND PERIPHERALS
LIMITED WARRANTY

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

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

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

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

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

SERVICE CENTER ADDRESS

SHARP ELECTRONICS CORPORATION
Sharp Plaza
Mahwah, New Jersey 07430-2135
(201) 512-0055

SHARP ELECTRONICS CORPORATION
Sharp Plaza, 20600 South Alameda St.,
Carson, California 90810
(213) 637-9488

SHARP ELECTRONICS CORPORATION
725 Old Norcross Road
Lawrenceville, GA 30245
(404) 995-0717

SHARP ELECTRONICS CORPORATION
1205 Executive Drive
East Richardson, Texas, 75081
(214) 234-1136

SHARP ELECTRONICS CORPORATION
1300 Naperville Drive
Romeoville, Illinois 60441
(708) 759-8555

To order Supplies or Accessories, contact your local SHARP Dealer/Retailer or in U.S.A. only contact THE SHARP ACCESSORIES AND SUPPLY CENTER at 1 (800) 642-2122.

REGIONAL SALES OFFICES AND DISTRIBUTION CENTERS.

Eatern	Sharp Plaza, Mahwah. New Jersey 07430-2135	Phone: (201) 529-8200
Midwest	1300 Naperville Drive, Romeoville Illinois 60441	Phone: (708) 759-8555
Western	Sharp Plaza, 20600 South Alameda Street Carson, California 90810	Phone: (213) 637-9488

SHARP

SHARP ELECTRONICS CORPORATION

Sharp Plaza, Mahwah. New Jersey 07430-2135

SHARP CORPORATION

OSAKA, JAPAN

PRINTED IN JAPAN/IMPRIMÉ AU JAPON

1L 0.5-1 (TINSE1029ECZZ) ⑦