Hewlett-Packard 9825A Calculator Operating and Programming

HEWLETT hp PACKARD

Warranty Statement

Hewlett-Packard products are warranted against defects in materials and workmanship. For Hewlett-Packard Desktop Computer Division products sold in the U.S.A. and Canada, this warranty applies for ninety (90) days from date of delivery.* Hewlett-Packard will, at its option, repair or replace equipment which proves to be defective during the warranty period. This warranty includes labor, parts, and surface travel costs, if any. Equipment returned to Hewlett-Packard for repair must be shipped freight prepaid. Repairs necessitated by misuse of the equipment, or by hardware, software, or interfacing not provided by Hewlett-Packard are not covered by this warranty.

NO OTHER WARRANTY IS EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. HEWLETT-PACKARD SHALL NOT BE LIABLE FOR CONSEQUENTIAL DAMAGES.

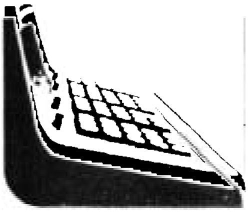
*For other countries, contact your local Sales and Service Office to determine warranty terms.

Operating and Programming

Hewlett-Packard 9825A Calculator

Including Memory Options 001, 002, and 003

9825A Calculator



Hewlett-Packard Desktop Computer Division
3404 East Harmony Road, Fort Collins, Colorado 80525
(For World-wide Sales and Service Offices see back of manual.)

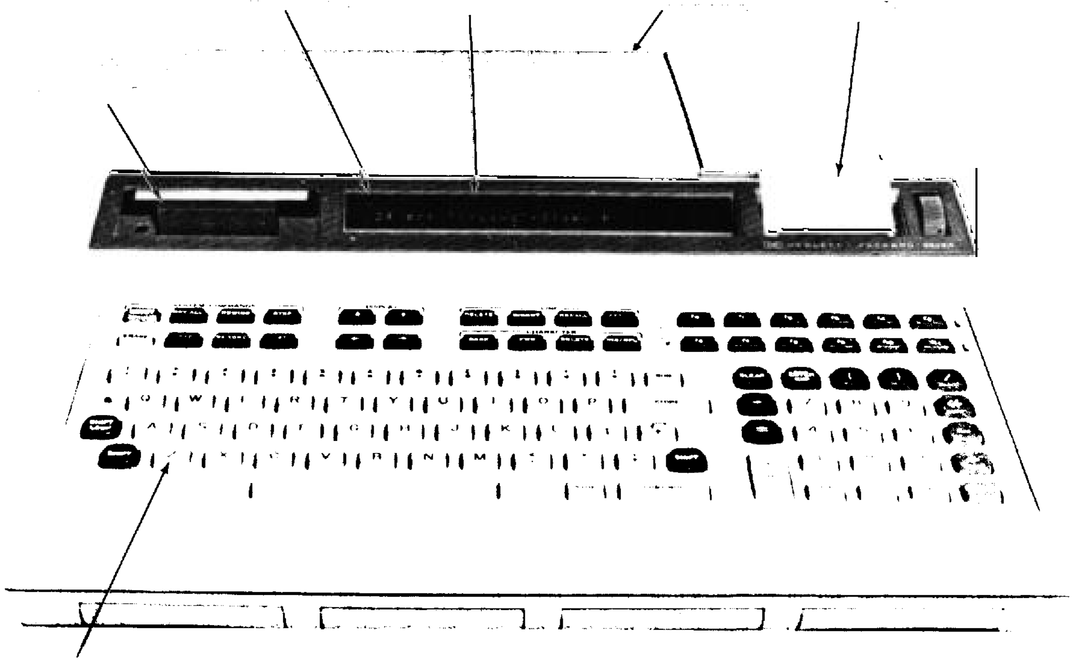
Manual Summary

Table of Contents

Figures

Tables

Overview



- Upper and lower case keyboard similar to a typewriter.

- Abbreviated mnemonics, multi-statement lines, and implied multiplication.

- Allows you to use the calculator while a program is running.

- Three interface slots allow your system to grow. Optional capabilities include DMA, interrupts, and bit-manipulation.

- Two track, high-density, for fast storage and access of programs and data.

- 16-character-wide printer for hardcopy listings of programs, messages, and data.

- 32 characters can be displayed for entering program lines, debugging, and displaying messages.

*Light Emitting Diode

Chapter 1
Owner's Information

This chapter covers the installation of your HP 9825A Calculator, the available accessories, and other information that is important when you first receive your calculator.

Calculator Inspection Procedure

The individual parts of your calculator system were thoroughly inspected before they were shipped to you. All equipment should be in good operating order. Carefully check the calculator, ROMs, peripheral equipment, and other items for any physical damage sustained in transit. Notify HP and file a claim with the carrier if there is any such damage.

Please check to ensure that you have received all of the items which you ordered and that any options specified on your order have been installed in your calculator. Refer to the table on the next page and check that all accessories are present.

If you have any difficulties with your system, if it is not operating properly, or if any items are missing, please contact your nearest HP sales and service office; addresses are supplied at the back of this manual.

Equipment Supplied

The following items are packaged with each HP 9825A Calculator. Peripheral devices and interface cards are packaged separately; each of these has its own manual or operating note and may also have extra items packaged with it.

Equipment Supplied

Description	Quantity	Part No.
Operating and Programming Manual	1	09825-90000
Quick Reference Guide	2	09825-90010
Error Booklet (under paper cover)	1	09825-90015
Blank Tape Cartridge	1	9162-0061
Utility Pac	1	09825-10000
AC Power Cord	1	(see below)
Dust Cover	1	9222-0495
Tape Head Cleaner	1	8500-1251
Special Function Key Overlays (Blank)	5	7120-4802
Spare Fuse (1.5A)	1	2110-0043
Spare Fuse (3A)	1	2110-0003
System Test Cartridge	1	09825-90035
System Test Booklet	1	09825-90031
Printer Paper	3	(see below)
Software Binder	1	9282-0563

All the equipment in the table above can be purchased by ordering HP part number 09825-80000. When ordering paper specify six-roll packs, HP part number 9270-0479.

Power Cords

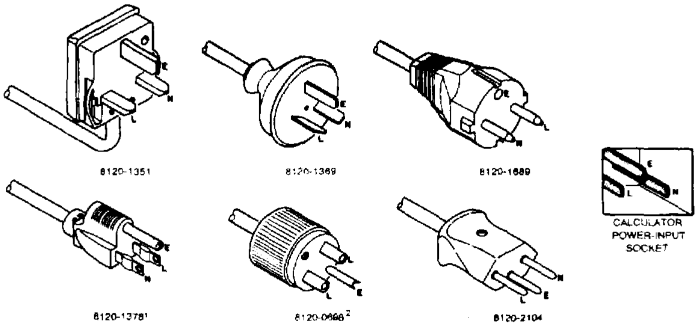
Power cords supplied by HP will have polarities matched to the power-input socket on the calculator, as shown below.

- L = Line or Active Conductor (also called "live" or "hot")
- N = Neutral or Identified Conductor
- E = Earth or Safety Ground

WARNING

IF IT IS NECESSARY TO REPLACE THE POWER CORD, THE REPLACEMENT CORD MUST HAVE THE SAME POLARITY AS THE ORIGINAL. OTHERWISE A SAFETY HAZARD FROM ELECTRICAL SHOCK TO PERSONNEL, WHICH COULD RESULT IN DEATH OR INJURY, MIGHT EXIST. IN ADDITION, THE EQUIPMENT COULD BE SEVERELY DAMAGED IF EVEN A RELATIVELY MINOR INTERNAL FAILURE OCCURED.

Power cords with different plugs are available for the calculator; the part number of each cord is shown below. Each plug has a ground connector. The cord packaged with each calculator depends upon where that calculator is to be delivered. If your calculator has the wrong power cord for your area, please contact your local HP sales and service office.



Power Requirements

The 9825A Calculator has the following power requirements:

- Line Voltage: 100 Vac + 5%, −10%
  120 Vac + 5%, −10%
  220 Vac + 5%, −10%
  240 Vac + 5%, −10%
  } Switch Selectable
- Line Frequency: 48 to 66 Hertz
- Power Consumption: 100V @ 1.7A
  120V @ 1.5A
  220V @ .8A
  240V @ .75A

Grounding Requirements

To protect operating personnel, the National Electrical Manufacturers' Association (NEMA) recommends that the calculator be properly grounded. The calculator is equipped with a three-conductor power cable which, when connected to an appropriate power receptacle, grounds the calculator. To preserve this protection feature, do not operate the calculator from an ac power outlet which has no ground connection.

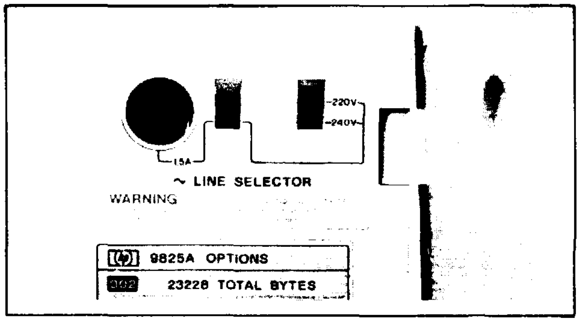
[1] UL and CSA approved for use in the United States of America and Canada with calculators set for either 100 or 120 Vac operation.

[2] UL and CSA approved for use in the United States of America and Canada with calculators set for either 220 or 240 Vac operation.

Fuses

For 100 or 120 Vac operation, use a 3A fuse; for 200 or 220 Vac operation use a 1.5A fuse.

> **WARNING**
>
> TO AVOID THE POSSIBILITY OF SERIOUS INJURY, DISCONNECT THE AC POWER CORD BEFORE REMOVING OR INSTALLING A FUSE.

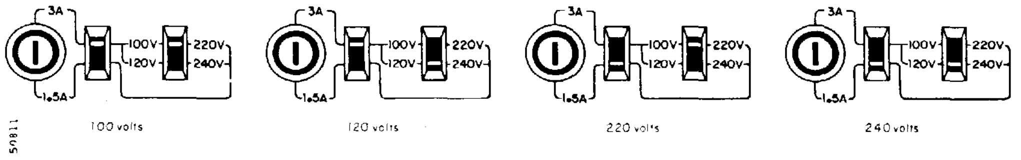


Location of Fuse

The figure shows the location of the fuse under the paper cover. To change the fuse, first disconnect the power cord to the calculator. Then remove the fuse cap by pressing inward while twisting it counterclockwise. Remove the fuse from the cap and insert the correct replacement fuse (either end) into the cap. Finally, put the fuse and cap back into the fuse holder. Press on the cap and twist it clockwise until it locks in place.

Initial Turn-On Instructions

1. With the calculator disconnected from its ac power source, check that the proper calculator fuse has been installed for the voltage in your area (see previous section).
2. Next, ensure that the two voltage selector switches under the paper cover are set for the correct powerline voltage. The figure below shows the correct settings for each nominal line voltage. If it is necessary to alter the setting of either switch, insert the tip of a small screwdriver into the slot on the switch. Slide the switch so that the position of the slot corresponds to the desired voltage, as shown below.



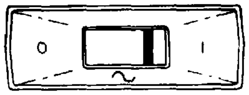
Nominal Line Voltage Settings

3. The operating system module on the right-hand side of the calculator must be inserted so that it is even with the side of the calculator.
4. Install the desired ROM cards and Interface Cards (see ROM installation, page 6 and refer to the appropriate manual for interface installation).

CAUTION

ALWAYS TURN OFF THE CALCULATOR WHEN INSERTING OR REMOVING ROMS AND INTERFACES. FAILURE TO DO SO COULD DAMAGE EQUIPMENT.

5. Connect the power cord to the power input connector on the back of the calculator. Plug the other end of the cord into the ac power outlet.
6. Switch the calculator on using the switch on the right-hand side of the calculator.



Calculator Testing

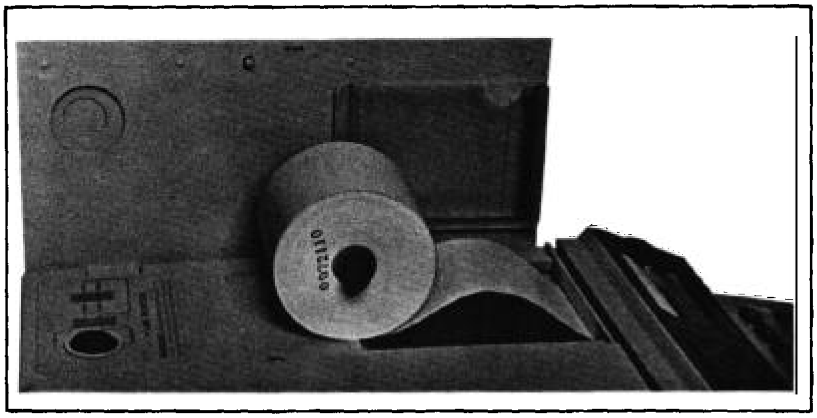
If you wish to test your calculator, or if there is any doubt that your calculator is operating correctly, refer to the System Test Booklet for the calculator test procedure.

Loading Printer Paper

The internal printer uses special heat-sensitive (thermal) paper. When ordering paper, specify the six-roll pack, HP part number 9270-0479.

To load a roll of paper:

1. Lift the paper cover and remove the paper spindle. Discard the old paper core and remove any paper left in the printer using the paper advance wheel.
2. Install the new roll as shown in the following figure.
3. Insert the free end of the paper and advance it through the printer using the paper advance wheel.



Loading Printer Paper

Accessories

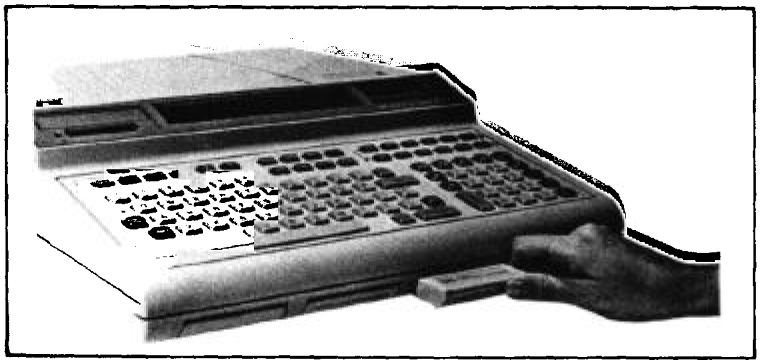
This section contains general ROM information and describes the ROMs, peripherals, and interfaces that are currently available for the 9825A Calculator.

ROM Description

Several ROMs (Read Only Memories) are available for your calculator; each provides additional language capabilities to perform specific tasks, such as plotting, controlling peripherals, or extending the programming capabilities of your calculator. One or more ROMs are packaged in a ROM card.

ROM Installation

A ROM card can be plugged into any one of the four ROM slots on the bottom front of the calculator as shown below



ROM Installation

To install a ROM, first turn off the calculator. Then slide the ROM, with the label right-side-up, through the ROM slot door. Press it in so that it is even with the front of the calculator.

The following ROMs are available in various combinations.

String Variables ROM

This ROM enables the calculator to recognize and operate on letters and words ("strings") in much the same way that it recognizes and operates on numbers. Some of the capabilities which are provided include: single strings and string arrays, numeric value of a string of digits, concatenation, and displaying or printing all special characters.

Advanced Programming ROM

This ROM extends the programming capabilities of the 9825A Calculator. For/next looping, split and integer precision number storage, multiparameter functions and subroutines, and the cross reference statement are the operations provided by the Advanced Programming ROM.

Matrix ROM

The Matrix ROM extends the language to include statements for manipulating matrices and arrays. Addition, subtraction, multiplication, and division of arrays, as well as inversion, transposition, and determinants of matrices are only some of the capabilities provided by this ROM.

9862A Plotter ROM

This ROM enables the 9825A Calculator to control the HP 9862A Plotter. Axes can be drawn and labelled; functions can be plotted; and in the "typewriter" mode, characters can be printed as you type them from the keyboard. More than one 9862A Plotter can be operated at the same time.

General I/O ROM

The General I/O ROM provides basic I/O capability with formatting. Most 9800 series peripherals (not the 9862A Plotter) can be controlled using this ROM. Binary I/O, status checking, and limited control of instruments via the HP-Interface Bus are also provided.

Extended I/O ROM

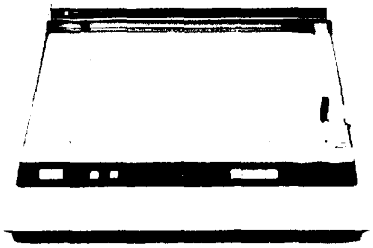
The Extended I/O ROM extends the I/O capability of the calculator by providing complete HP-IB control, bit manipulation and testing, auto-starting, error trapping, and interrupt capabilities.

Calculator Peripherals

Each of these peripherals is available with the necessary interface cables and operating instructions. The General I/O ROM is needed to control all peripherals except the 9862A Plotter.

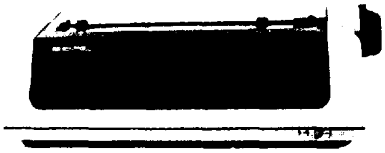
HP 9862A X-Y Plotter

Histograms, pie charts, circuit diagrams, linear, log-log, and polar plots – these are a few of the things you can do with the 9862A Plotter. The 9862A Plotter ROM provides all the instructions needed to control the 9862A Plotter with the 9825A Calculator.

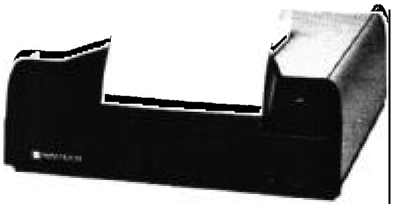


Output Printers

If you need data output in tables, charts, or forms, one of these printers can do the job.



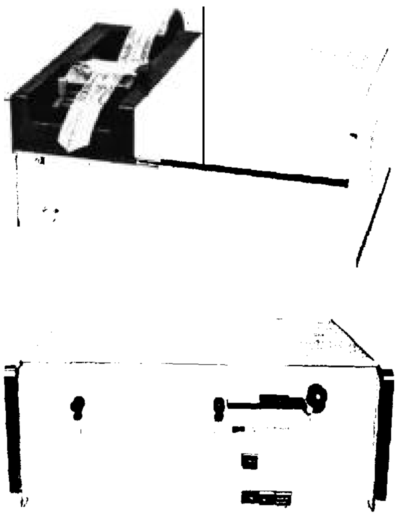
The HP 9871A Character-Impact Printer provides 132-character wide, multiple copy output, and a full 96-character font. The average printing speed is 30 characters per second. This printer includes an additional plotting capability.



The HP 9866B Thermal Printer is a fast (240 lines per minute) line printer. It has a 95-character alphanumeric font with upper and lower case, produces fully-formatted text and tables, and has plotting capability.

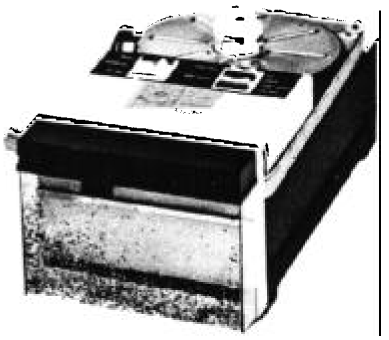
Card Reader

The 9869A Card Reader provides a convenient form of data entry from punched or marked cards. Standard 80-column cards can be read at up to 300 cards per minute.



Paper Tape Readers

Data from analytical instruments, machine tools, and computer terminals can be entered directly into the calculator with one of these paper tape readers. The HP 9863A makes it easy to read data in a wide variety of formats at 20 characters per second. The HP 9883A Tape Reader, designed for high-speed, heavy-volume operations, optically reads tapes at up to 500 characters per second.

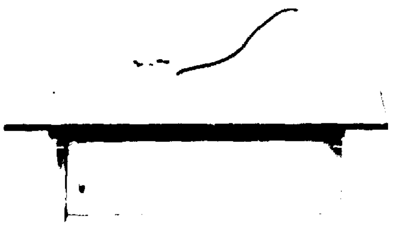


HP 9884A Tape Punch

Add high-speed output to your calculator with a tape punch. This reliable, compact unit punches tape at 75 characters per second.

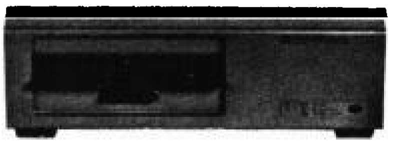
HP 9864A Digitizer

Use this peripheral to read a curve, or any irregular shape, as a series of discrete points and then convert these to a series of digital x-y coordinates. To make entries, simply trace the shape; then the calculator can find dimensions and area of the line or contained shape. With the proper programs, you can directly process graphical data, such as X-rays, blueprints, strip-chart recordings, or cut-and-fill profiles.



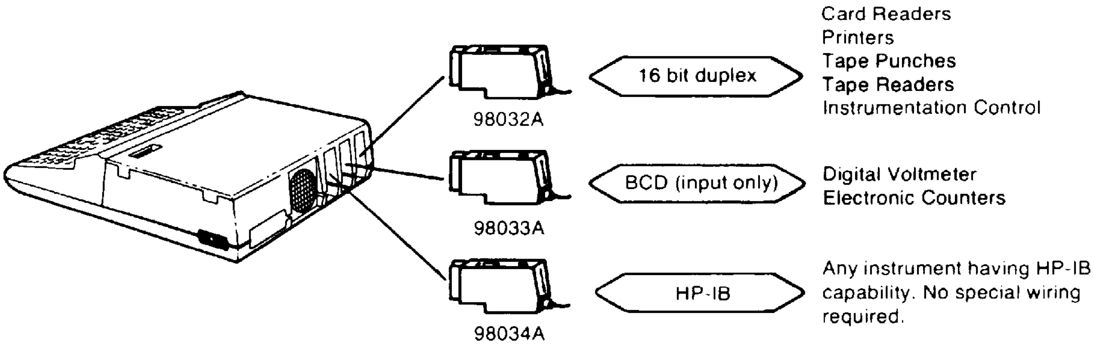
HP 9885 Flexible Disk Drive

The HP 9885 Flexible Disk Drive provides a convenient, reliable, and low-cost method of transferring programs and data to and from the calculator at high speeds. The 9885 is a random access, removable, mass storage device with a capacity of up to 468,480 bytes per disk. The removable, low-cost flexible disks provide unlimited storage to the user. The 9885M (Master) Flexible Disk Drive contains a built-in controller. Up to three 9885S's (Slaves) can be connected to each 9885M.



Interface Kits

Several interfaces are available for the calculator to control and exchange data with a wide variety of equipment other than the peripherals just described. Each interface has general installation and configuration instructions. Typical applications are shown here:



16-Bit Duplex Interface

The HP 98032A Interface is a general purpose card providing a 16-bit parallel, character-serial interface. The interface transfers data between the 9825A Calculator and a peripheral device in full-duplex; that is, the interface can have valid data on the output lines and be inputting data at the same time. The General I/O ROM is needed to control this interface.

BCD Interface

The HP 98033A BCD Interface allows the calculator to take samples from many instruments having parallel, binary-coded-decimal (BCD) output. The interface can input data of up to ten BCD digits, mantissa sign, exponent sign, and overload indication. The General I/O ROM is required to control this interface.

HP Interface Bus

The HP 98034A HP-IB Interface allows up to 14 HP-IB compatible instruments to be connected to the calculator at one time. The General I/O ROM provides simplified control of the interface. The Extended I/O ROM should be used when complete control is desired.

Prerecorded Programs

Tape cartridges containing programs for solving problems from many disciplines are available. A utility program cartridge is supplied with each calculator. For a complete list of prerecorded programs and for pricing information, contact any HP sales office (addresses are provided in the back of this manual).

Service Contracts

When you buy a Hewlett-Packard desk-top calculator, service is an important factor. If you are to get maximum use from your calculator, it must be in good working order. A HP Maintenance Agreement is the best way to keep your calculator in optimum running condition.

Consider these important advantages:

- **Fixed Cost**– The cost is the same regardless of the number of calls, so it is a figure that you can budget.
- **Priority Service**– Your Maintenance Agreement assures that you receive priority treatment, within an agreed upon response time.
- **On-Site Service**– There is no need to package your equipment and return it to HP. Fast and efficient modular replacement at your location saves you both time and money.
- **A Complete Package**– A single charge covers labor, parts, and transportation.
- **Regular Maintenance**– Periodic visits are included, per factory recommendations, to keep your equipment in optimum operating condition.
- **Individualized Agreements**– Each Maintenance Agreement is tailored to your support equipment configuration and your requirements.

After considering these advantages, we are sure you will agree that a Maintenance Agreement is an important and cost-effective investment.

For more information please contact your local HP calculator sales and service office.

Keyboard Magazine

Keyboard is a periodical magazine containing general information about HP calculators and related equipment. It includes articles and programs written by calculator users, description of the latest equipment and prerecorded programs, programming tips, and many other items of general interest to calculator users.

To receive your free subscription to **Keyboard**, merely complete the order form supplied with the calculator.

Chapter 2
General Information

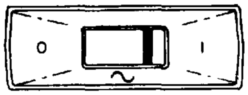
This chapter introduces some of the operating characteristics of the calculator. The keyboard, display, and range are a few of the topics covered.

Before Using the Calculator

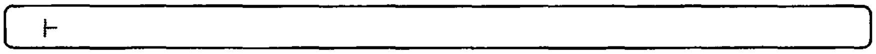
There are a few things you should check each time you turn on the calculator.

If the calculator is turned off:

- Set the power switch on the right-hand side of the calculator to the "1" position:



- When the following display appears, the calculator is ready for use:



If the calculator is turned on and the display is blank:

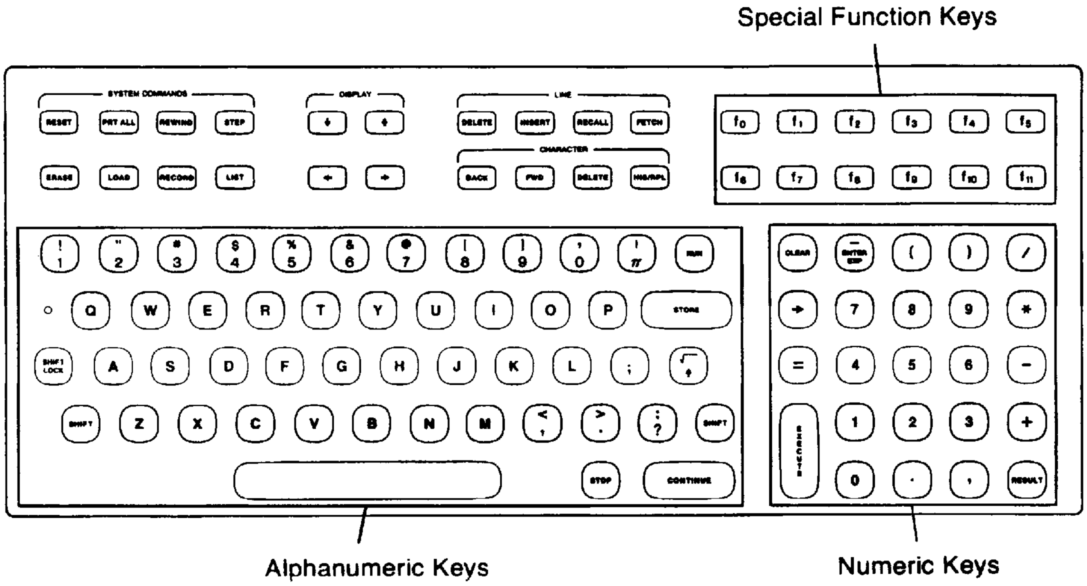
- Press CLEAR or STOP

If the display still remains blank, first check the power connection and fuse as described on pages 3 and 4. Then call your HP sales and service office listed in the back of this manual.

If the calculator is on and the display shows the "lazy T", you can do keyboard operations or arithmetic or you can enter programs and run them.

Description of the 9825A Calculator

The Keyboard



The 9825A Keyboard

The 9825A Keyboard is shown in the figure above. The keyboard is divided into the following functional groups:

- **Alphanumeric Keys** - This area is very much like a standard typewriter keyboard. For instance, to display a capital A, press the shift key and A at the same time; or to display a percent sign, %, press the shift key and %5 at the same time.
- **Numeric Keys** - All the keys needed to enter numbers and do simple arithmetic are located in this block. The numeric keys in the alphanumeric section of the keyboard can also be used to enter numbers. The exponentiation and square root key, √↑, is located in the alphanumeric key section.
- **Special Function Keys** - The keys in the upper right section of the keyboard, namely f_0 through f_{11}, provide additional calculator abilities. These keys are explained in Chapter 4.

The remaining keys provide functions, such as editing and display control, and are explained in Chapter 4.

Keys of the same color have similar functions. For example, all the alphanumeric keys are the same beige color; gold colored keys are control keys used to run programs, store lines, erase programs, etc.

Below are a few more topics related to keyboard operations:

- **Spacing** - In general, spaces are not important. It makes no difference, for example if you key in:

  A+B or A + B

  Both are interpreted the same. Spacing, however, is important when using text (characters within quotes) and when printing and displaying messages.

- **Repetition of Keys** - When a key is held down, its operation is repeated rapidly. This is an especially useful feature with the editing keys.

- **The ⊢ Symbol** - When the display is clear and awaiting inputs, the "lazy T" symbol appears in the leftmost character of the display. This symbol also indicates the end of a stored line.

- **The Run Light** - A small red light in the left end of the display lights when a program is running. For example:

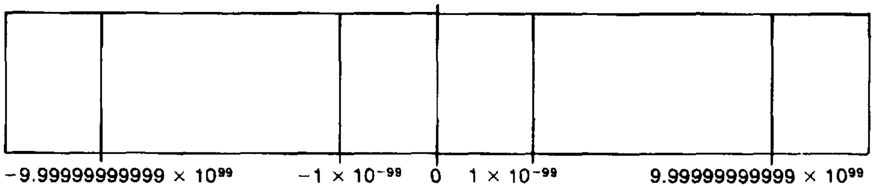
Display and Line Length

The 9825A Calculator has a 5 × 7 dot matrix, 32-character display. Even though only 32 characters can be displayed at one time, up to 80 characters can be keyed into the display. After the 32nd character, additional characters which are keyed in cause the displayed line to shift to the left. After 67 characters are keyed, a beep indicates that only thirteen more characters can be entered. Up to 73 characters can be stored. This includes any spaces or parentheses which the calculator may automatically insert in the line.

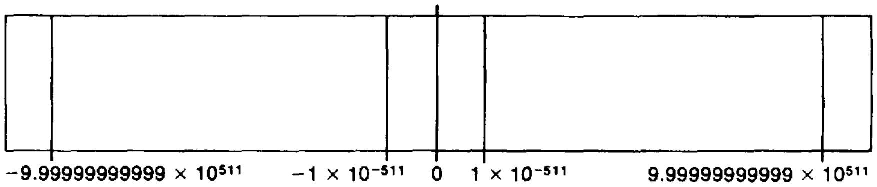
Range

The range of values which can be entered or stored is $-9.99999999999 \times 10^{99}$ through -1×10^{-99}, 0, 1×10^{-99} through $9.99999999999 \times 10^{99}$. However, the range of calculations is from $-9.99999999999 \times 10^{511}$ through -1×10^{-511}, 0, and 1×10^{-511} through $9.99999999999 \times 10^{511}$.

Storage Range



Calculating Range



out of range □ within range □

The extended calculation range is useful for calculations which have intermediate results outside of the storage range, but which have final results within the storage range. For instance:

$$(9.2 \times 10^{23} \times 8.6 \times 10^{80})/(1 \times 10^{24})$$

When the first two values are multiplied their result is:

$$(7.912 \times 10^{104})$$

This intermediate result cannot be stored, but the final result, 7.912×10^{80}, can.

Significant Digits

All numbers are stored internally with 12 significant digits in the mantissa and a two digit exponent. The format used to display or print numbers (such as fxd 2) has no effect on the internal representation of a number.

Memory

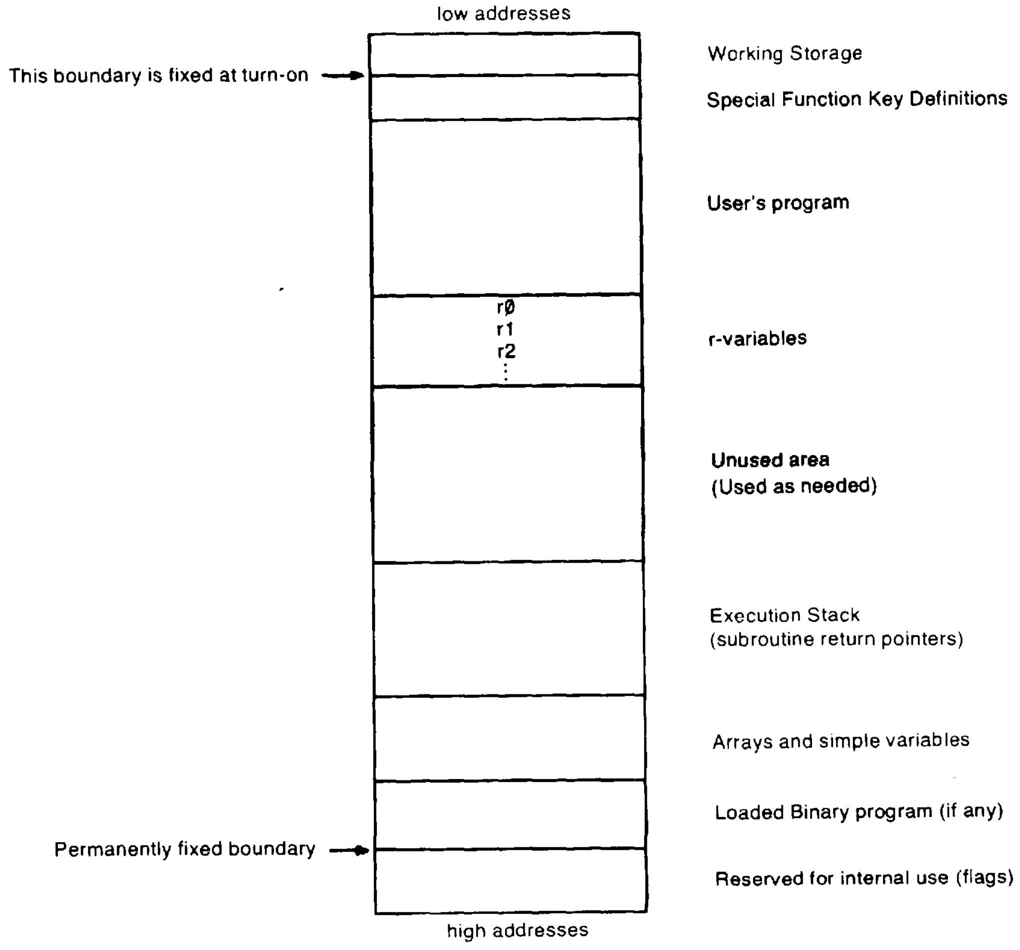
The 9825A Calculator uses two types of memory: Read/Write Memory, and Read Only Memory. Read/Write Memory is used to store programs and data. When you store a program or data, you "Write" into the memory. When you access a line of your program or a data element, you "Read" from memory; thus the term Read/Write.

Read Only Memory differs in that it is permanent. When the calculator is turned off, the contents of the Read/Write memory are lost, whereas the Read Only Memory is unaffected. ROM (for Read Only Memory) cards can be plugged into the ROM slots on the front of the calculator. This makes it possible to expand the language.

Programs and data in Read/Write memory can be saved for future use by recording the information on the tape cartridge.

A small amount of memory is sometimes required by a plug-in ROM. This area is called "working storage".

Read/Write Memory Organization


low addresses
This boundary is fixed at turn-on
Working Storage
Special Function Key Definitions
User's program
rØ
r1
r2
⋮
r-variables
Unused area
(Used as needed)
Execution Stack
(subroutine return pointers)
Arrays and simple variables
Loaded Binary program (if any)
Permanently fixed boundary
Reserved for internal use (flags)
high addresses

Language

The language used by the HP 9825A Calculator is called HPL. The basic programming unit is the statement. Statements are typed using lower case abbreviated mnemonics, such as `prt` for print. Multi-statement lines can be stored by separating statements with semicolons.

Two other characteristics of this language are implied multiplication and the assignment operator. Implied multiplication is a standard algebraic notation, such as 5X. The assignment operator → points to the variable being assigned a value, such as `5 → D`.

More mnemonics can be added to the language by adding ROM cards which plug into the ROM slots on the front of the calculator.

Error Messages

When an error occurs, the calculator beeps and displays an error number. The number references a description that will help pinpoint the cause of the error. For example:

```
error 07
```

Indicates a syntax error.

If an error message is displayed during an attempt to run a program, the program line number where the error occurs will also be displayed. For example:

```
error 17 in 3
```

Indicates that a parameter is out of range in line 3.

Pressing RECALL after some error messages will bring the line containing the error into the display with a flashing cursor indicating the location of the error.

A complete list of the error messages is given in Appendix B, on the inside back cover of this manual, and in the error booklet under the top cover of the calculator.

Chapter 3

Introduction To Keyboard and Programming

This chapter introduces some of the basic concepts of keyboard operation and programming for those who are unfamiliar with the operation of the calculator. Nearly all the operations which you perform from the keyboard can also be programmed. The operations explained in this chapter are covered in more detail elsewhere in the manual.

System Keys

The following keys are used often for keyboard operations and programming.

(CLEAR) Clears the display; the ⊢ symbol remains to show that the calculator is ready for further instructions:

```
⊢
```

(EXECUTE) Performs the operation in the display. For example, to add 2 + 2:

Press: (2)(+)(2)

```
2+2
```

Press: (EXECUTE)

```
4.00
```

(STORE) Stores program lines in the memory. For example, to store a program line:

Type in: (7)(→)(SHIFT)* (A)

```
7→A
```

Press: (STORE)

```
0⊢
```

*The (SHIFT) indicates that the following key is shifted.

This program line will assign the value 7 to the variable A.

RUN Runs the program in memory from line 0.

Keyboard Arithmetic

The six basic arithmetic operations in the 9825A are: addition (+), subtraction (−), multiplication (*), division (/), exponentiation (↑), and square root (√).

To perform a math operation, such as 8 × 2, first you key in the expression as follows:

8 * 2 | 8*2

Then press: EXECUTE | 16.00

To raise a number to a power, such as 8^2, press:

8 ↑ 2 EXECUTE | 64.00

Notice that an operation such as 8^{-2} must appear as: 8↑(-2).

The value which is displayed after pressing the execute key is stored in a location called "result". This value can be used in other calculations. For example:

2 * 3 EXECUTE | 6.00

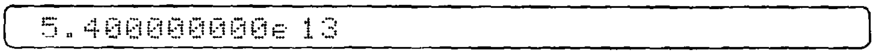
RESULT * 2 EXECUTE | 12.00

RESULT / 4 EXECUTE | 3.00

If you execute an operation involving large numbers, such as:

6000000*9000000

the calculator displays the result in scientific notation, with 9 digits to the right of the decimal point:



```
5.400000000e 13
```

This is because the number is too large for the fixed 2 notation which is set when you switch on the calculator.

Arithmetic Hierarchy

When an expression has more than one arithmetic operation, the order in which the operations take place depends on the following hierarchy:

√	square root	performed first
↑	exponentiation	↓
no operator	implied multiplication	
* /	multiplication and division	
+ −	addition and subtraction	performed last

An expression is scanned from left to right. Each operator is compared to the operator on its right. If the operator to the right has a higher priority, then that operator is compared to the next operator on its right. This continues until an operator of equal or lower priority is encountered. The highest priority operation, or the first of the two equal operations, is performed. Then any lower priority operations on the left are compared to the next operator to the right. If parentheses are encountered, the expression within the parentheses is evaluated before the left-to-right comparison continues. This comparison continues until the entire expression is **evaluated. For example:**

$2+3*\underline{6\uparrow2}(4)/(6-2)\uparrow2$	exponentiation
$2+3*\underline{36(4)}/(6-2)\uparrow2$	implied multiplication
$2+\underline{3*144}/(6-2)\uparrow2$	multiplication
$2+432/\underline{(6-2)}\uparrow2$	evaluate parenthesis
$2+432/\underline{4\uparrow2}$	exponentiation
$2+\underline{432/16}$	division
$\underline{2+27}$	addition
29	result

Variables

A variable is a name of a location where numbers are stored. There are three types of variables: simple variables, r-variables, and array variables. Array variables are discussed on page 37.

Simple Variables

Twenty-six simple variables, named A through Z, are used on the 9825A Calculator. Only the upper case letters can be used for simple variable names.

To assign a value to a variable, the assignment operator is used. For instance, to assign the value 4.5 to N, press:

4 . 5 → SHIFT* N EXECUTE

The number always appears on the left, and the variable appears on the right side of the assignment operation.

Now, N can be used in calculations. For instance, to multiply N by 2, press:

2 * SHIFT N EXECUTE   9.00

N is not changed. New values can be assigned to variables, such as:

SHIFT N + 1 → SHIFT N EXECUTE

r-Variables

r-variables are designated by a lower-case "r" followed by a number (e.g., `r18`). They are useful for one dimensional arrays and can be used in addition to the 26 simple variables.

In the following two examples, the value 12 is assigned to r10. Then the value 20 is assigned to the register designated by the value of r10 (this is called indirect storage).

```
12 → r10        The value 12 is assigned to r10 directly.
20 → rr10       The value 20 is assigned to r12 indirectly.
```

For more information about r-variables, see page 38.

Operating Modes

The calculator can operate in any of three modes: the calculator mode, the program mode, or the live keyboard mode.

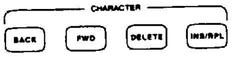
- In calculator mode, no program is running, and the calculator is awaiting inputs or calculating keyboard entries.

*The SHIFT indicates that the following key is shifted.

- In the program mode, a program is running.
- In live keyboard mode, you can perform many calculator operations while a program is running.

Basic Editing

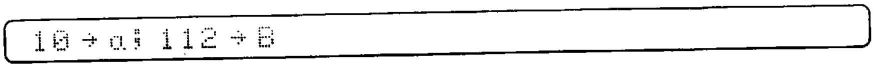
If you make a mistake while entering lines into the display, you can use the character editing keys for changing the line.



For instance, suppose you want to type in this line:

10 → A; 12 → B

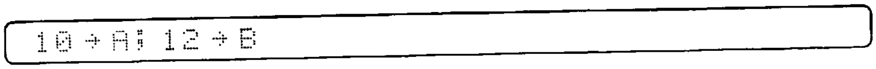
But, instead you type:



```
10→a; 112→B
```

To correct this, simply press BACK until a flashing cursor ■ appears over the "a".

Then type in an A. To delete a "1" in 112, press FWD once and press character DELETE. The resulting display would be:



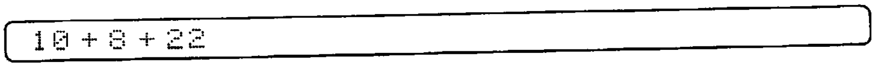
```
10→A; 12→B
```

with a flashing cursor on the "1" of 12. To execute the line, press:

As another example, maybe you want to execute this line:

10 + 18 + 22

But you typed this:



```
10+8+22
```

To insert a one in front of the 8, press the BACK key 4 times. The flashing replace cursor ▓ will be positioned on the 8. Next, press the INS/RPL key. This changes the replace cursor to the insert cursor ◀. Now, type in a 1. The display will be:

```
1▓ + 18 + 22
```

Note that the rest of the line shifted to the right 1 character. The insert cursor ◀ will still be flashing over the 8 indicating that more characters could be inserted if desired. To execute the line, press EXECUTE.

Programming

There are five basic steps in creating a program:

1. Define the problem.
2. Decide how the problem is best solved.
3. Write out the statements for the program.
4. Key the statements into the calculator memory.
5. Debug (correct) and run the program.

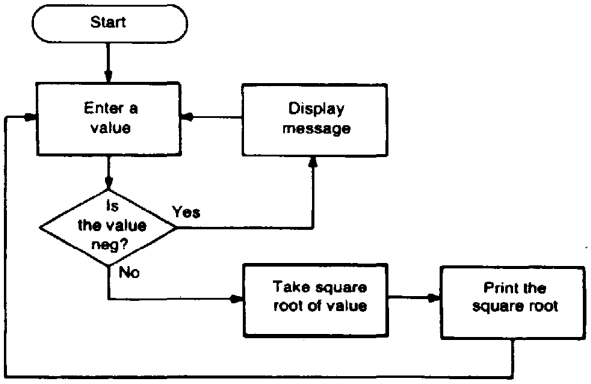
Step 1:

As a simple example, suppose you want to print the square root of each value that you enter. Then, if the value entered is negative, print a message and continue on.

Step 2:

A common method used to solve a problem is flowcharting*. Using a few basic flowcharting symbols, explained at the end of this chapter, we will flowchart the problem.

Flowchart:



*Another method suitable for simple problems is to key in a few statements and try them out.

Step 3:

From the flowchart, write down the statements for the program:

Program	Comments
"start": ent V	V is the value to be entered.
if V<Ø; dsp "neg. V"; gto"start"	Decide if V is negative: if so, display the message and go back to the beginning.
$\sqrt{V}\to S$	S is the square root of the value.
prt S	Print the square root.
gto "start"	Go to "start" for another value.

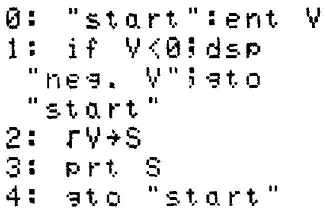
Note that the second line contains three statements separated by semicolons. All of the statements used are discussed later.

Step 4:

The next step is to clear the calculator by executing erase a. Then type in the program exactly as above, one line at a time. Press (STORE) at the end of each line to store that line in the calculator memory. If you make a mistake before you store the line, press (CLEAR) and type the line over.

Step 5:

After the program is stored, press [LIST] (EXECUTE) to get a printed listing. Then, to run the program press (RUN). Each time that V? is displayed, type in a value and press (CONTINUE). The calculator will print the square root of each value.



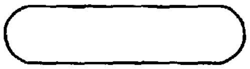
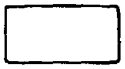
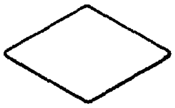
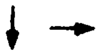
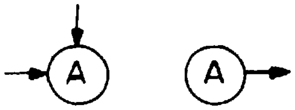
```
0: "start":ent V
1: if V<0;dsp
 "neg. V";gto
 "start"
2: rV→S
3: prt S
4: gto "start"
```

For positive values, the program runs as expected, but if you enter a negative value you won't see the message displayed. This is because the message is displayed for a very short period of time before another display (i.e., V?) appears. Use a **wait** statement after the display statement in line 1. This statement causes the program to pause long enough for you to see the message. To change the program, press: STOP FETCH 1 EXECUTE. Then press the BACK key until it is positioned on the semicolon just before the **gto** statement. Press INS RPL and key in ;wait 500. Press STORE to store the new line at line 1. Then press RUN. Here is a listing of the completed program:

```
0: "start":ent V
1: if V<0;dsp
  "neg. V";wait
  500;gto "start"
2: rV→S
3: prt S
4: gto "start"
```

Since the program is a continuous loop, press STOP to stop the calculator. Then, to do another program key in erase a and press EXECUTE. This clears out the calculator memory.

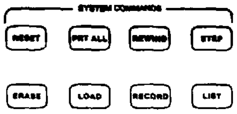
Commonly Used Flowchart Symbols

	Meaning
	Program beginning or end.
	Program segment; usually one statement.
	Decision block indicates that a decision for a branch is made. Usually an **if** statement is used for a decision.
	Flowlines indicate the program flow.
	Connectors indicate that the lines going to or from them are connected.

Chapter 4 The Keyboard

While reading the chapter, refer to the foldout inside the back cover which shows the 9825A Calculator Keyboard. The standard alphanumeric keys are used to enter numbers, commands, and statements. The rest of the keyboard is divided into system command keys, display control keys, line and character editing keys, special function keys, and calculator control keys. All of these keys, except the standard alphanumeric character keys, are explained below.

System Command Keys



RESET Returns the calculator and I/O cards to the power-on state (see table in Appendix D) without erasing programs or variables. RESET is executed immediately when it is pressed; it does not have to be followed by ↓. All calculator activity is halted and the line number of the current location in a program is displayed if a program is running. The reset key should be used to reset the calculator when no other key, such as CLEAR or STOP, will bring the calculator to a ready state.

PRT ALL Sets the print-all mode on or off. When it is pressed once, the word on appears in the display. When it is pressed again, the word off appears in the display. In print-all mode, displayed results, executed lines, and stored lines are printed.

While a program is running in print all mode, all displayed messages and error messages are printed. Print-all mode can be turned on or off while a program is running.

REWIND Automatically rewinds the tape cartridge to its beginning. Other statements and commands can be executed immediately without waiting for the cartridge to completely rewind. If REWIND is pressed while a program is running or while a line is executing from the keyboard, the cartridge rewinds at the end of the current line.

[STEP] Executes a program, one line at a time. Then, the line number of the next line to be executed is displayed. When [STEP] is pressed just after stopping a program, only the line number of the next line to be executed is displayed. The next time [STEP] is pressed, that line is executed.
To step from a specific line, execute a gto X, where X is the line to start stepping from. For example, to begin stepping through your program from line 30, type in gto 30 and press [EXECUTE]. Then use the step key.

[ERASE] This typing aid is used to erase all or part of the Read/Write memory.

[ERASE] [A] [EXECUTE]	Erases the entire calculator memory.
[ERASE] [V] [EXECUTE]	Erases only the variables.
[ERASE] [K] [EXECUTE]	Erases all the special function keys.
[ERASE] [EXECUTE]	Erases programs and variables.
[ERASE] [fn]	Erases the special function key represented by "n".

The table in Appendix D lists things which are affected by the erase command.

[LOAD] This typing aid is used to load programs and data from the tape cartridge. For example to load a program which is on file 3:

[LOAD] [3] [EXECUTE]	Loads the program from file 3 into the calculator.

The display shows ldf (for "load file") when this key is pressed (see the load file statement on page 104).

[RECORD] This typing aid is used to record programs and data on the tape cartridge. Before recording on the tape cartridge, files must be marked (see the mark statement on page 96). In the following example, it is assumed that the file has been marked:

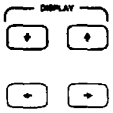
[RECORD] [6] [EXECUTE]	Record the calculator program on file 6 of the tape cartridge.

The display shows rcf (for "record file") when this key is pressed (see the record file statement on page 102).

LIST This typing aid is used to list programs, sections of programs, all special functions keys, or individual special function keys. For example:

LIST EXECUTE	Lists the entire program.
LIST K EXECUTE	Lists all defined special function keys in numerical order.
LIST f_0	Lists special function key, f_0.
LIST 2 0 EXECUTE	Lists the program from line 20 to the end.
LIST 9 , 1 3 EXECUTE	Lists the program from line 9 to 13, inclusive.

Display Control Keys



↓ Brings the line with the next higher-valued line number into the display. If there are no more lines in the program, ↓ clears the display and allows new program lines to be appended to the end of the program.

↑ Brings the line with the next lower-valued line number into the display. If a line number is in the display, ↑ brings that line into the display. If a stop statement is executed from a program, ↑ brings the line following the line with the stop statement into the display. After a program error, ↑ brings the line containing the error into the display for editing.

← Moves the line in the display to the left. This allows all the characters in a line to be moved into the display. Each time it is pressed, the displayed line moves 8 characters.

→ Moves the line in the display to the right for viewing all the characters in a line. Each time this key is pressed the displayed line moves 8 characters.

Editing Keys

There are two sets of editing keys; line editing keys and character editing keys.

Line Editing Keys

LINE

[DELETE] [INSERT] [RECALL] [FETCH]

[FETCH] This typing aid is used to bring program lines into the display and to fetch special function keys. For example:

[FETCH] [2] [0] [EXECUTE]	Brings line 20 into the display.
[FETCH] [f_4]	Accesses special function key f_4. If f_4 is defined, its definition is displayed. Otherwise f4 is displayed.

[DELETE] Deletes the program line in the display from the program. If no program line is in the display, the calculator beeps and the key is ignored. To delete a program line, fetch the line into the display and press [DELETE]. When a line is deleted from a program all subsequent line addresses and all relative and absolute go to and go sub statements are renumbered to reflect the deletion.

This is not the same key as the character delete key explained later. To delete several program lines, the delete (**del**) command can be used. The delete command is explained on page 80.

[INSERT] Inserts a line into a program. The inserted line is inserted before the fetched line. The fetched line and higher line numbers are renumbered. The [FETCH], [↓], or [↑] keys can be used to fetch a line into the display. For example:

To insert the line: ΓA→B between lines 20 and 21:

```
20: A+1→A
21: gto 25
```

Press: [FETCH] [2] [1]

Type in: ΓA→B

Press: [INSERT]

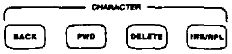
```
20: A+1→A
21: ΓA→B
22: gto 26
```

When a line is inserted into a program, the branching addresses of all relative and absolute go to and go sub statements are adjusted to reflect the insertions as in line 22 above.

RECALL Brings back, into the display, one of the two previous keyboard entries. Pressing RECALL once brings back the most recent keyboard entry. Pressing it twice brings back the previous keyboard entry.

Press RECALL after errors resulting from keyboard operations to recall the line containing the error. For many errors, a flashing cursor indicates the location of an error in the line.

The Character Editing Keys



Lines which are fetched into the display using ↑, ↓, RECALL, or FETCH, and lines which are typed into the display can be edited using the character editing keys.

Two flashing cursors are associated with these keys: the replace cursor █ and the insert cursor ◂.

BACK Moves the flashing replace cursor █, or the flashing insert cursor ◂, from its current position in the line in the display, toward the beginning (left) of the line. If the cursor is not visible, BACK causes the cursor to appear on the right-most character in the line.

FWD Moves the flashing replace cursor █, or the flashing insert cursor ◂, from its current position in the display, towards the last character in the line. For a line which has just been fetched or typed into the display, pressing FWD causes the flashing cursor to appear on the left-most character in the display.

DELETE Deletes individual characters which are under the insert or replace cursor. This is not the same key as the line delete key explained previously.

INS RPL The insert/replace key is used to change the flashing replace cursor to a flashing insert cursor and vice versa. Use the BACK or FWD key to position the cursor in the display. When the insert cursor is flashing, any characters entered from the keyboard are inserted to the left of the cursor and the characters under and to the right of the cursor shift to the right.

When the replace cursor is flashing, any character entered replaces the existing display character at the location of the cursor and the cursor moves to the character on the right.

Calculator Control Keys

RUN STORE SHIFT SHIFT LOCK STOP EXECUTE CONTINUE RESULT CLEAR → | π — ENTER EXP

RUN This key is an immediate execute key which runs the program in the calculator beginning at line zero. All variables, flags, and subroutine return pointers are cleared when RUN is pressed. The run light at the left end of the display indicates a running program.

The table in Appendix D lists things which are affected by pressing RUN.

STORE Stores individual program lines. Also, when a special function key is fetched and defined, STORE is used to store the key's definition. A program line can be a single statement or several statements separated by semicolons. When an error occurs while attempting to store a line, RECALL brings that line back into the display. A flashing cursor usually shows where the error was encountered in the line.

SHIFT and SHIFT LOCK are used to obtain shifted keyboard characters, such as A, #, and Γ. When SHIFT LOCK is pressed, the small light above the key lights. SHIFT LOCK locks the keyboard for shifted characters. Press SHIFT to release shift lock.

STOP Stops the program at the end of the current line. The number of the next program line to be executed is displayed. When STOP is pressed, list, tlist, and wait statements are aborted but the rest of the line is executed. When STOP is pressed in an enter statement, flag 13 is set and the enter statement is terminated.

There is also a stop statement. For details, see pages 48 and 76.

EXECUTE Executes the single or multi-statement line which is in the display. The two most recently executed (or stored) keyboard entries are temporarily stored and can be recalled by pressing RECALL once or twice. The result of a numeric keyboard operation which is not assigned to a variable is stored in Result (see RESULT key). For example:

2 + 2 EXECUTE

```
4.00
```

Pressing EXECUTE displays and stores the result. Pressing the execute key again repeats the same operation.

Although multiple expressions such as:

`2+2;√4` (↓)

are allowed, only the result of the last expression in the line is displayed and stored in Result. In print-all mode, both results are printed.

CONTINUE Automatically resumes a program from where it was stopped. When a line number is in the display (such as after pressing (STOP)) (CONTINUE) resumes the program from that line. However, after pressing (RESET), or after editing the program, the program continues at line 0 when (CONTINUE) is pressed. Pressing (CONTINUE) after an error also causes the program to continue from line 0.

In an enter statement, (CONTINUE) is pressed after entering data. If no data is entered and (CONTINUE) is pressed, the variable maintains its previous value and flag 13 is set. See also the continue command on page 79.

RESULT Accesses the storage location of the result of a numeric keyboard operation which was not assigned to a variable. For example:

Press: (3)(*)(6) `3*6`

Press: (↓) `18.00`

The answer, 18, is also stored in Result and can be used in other operations, such as:

Press: (RESULT)(/)(2) `res/2`

Press: (↓) `9.00`

In a program, values cannot be stored in Result; but the value in Result can be assigned to variables or used in computations.

For example:

`0: 20→res`	This is not allowed.
`1: res+2→A`	This assigns the value of Result +2 to the variable A.

CLEAR Clears the display. If the clear key is pressed during the enter statement, a question mark appears in the display, indicating that an entry is still expected. If this key is pressed after a special function key has been fetched, the key number (e.g., f8) appears in the display.

[→] The assignment operator is used to assign values to variables (this is not the same as the right arrow used for display control.) For example:

Press: [SHIFT] [√] [5] [→] [SHIFT] [X] [EXECUTE] This stores the square root of 5 in X.

[π] To enter the value of π, this key is pressed. The value entered is 3.14159265360.

[ENTER EXP] This key enters a lower case e into the display, representing an exponent of base 10. The unshifted [E] key can be used in place of [ENTER EXP]. For example:

Press: [1] [ENTER EXP] [9] [9] [EXECUTE]

```
1.000000000e 99
```

Press: [4] [E] [2] [3] [EXECUTE]

```
4.000000000e 23
```

Note that there is no difference between pressing [ENTER EXP] and pressing [E].

Special Function Keys

[f0] [f1] [f2] [f3] [f4] [f5]

[f6] [f7] [f8] [f9] [f10] [f11]

There are 12 special function keys, which provide 12 unshifted functions and 12 shifted functions. The special function keys can be used as typing aids, one line immediate execute keys, or as immediate continue keys.

To define a special function key, press [FETCH] and the special function key to be defined. Then enter a line in the display. Press [STORE] to store the definition of the key and to exit key mode. For example:

Press: [FETCH] [f0] f0 is displayed if the key was not previously defined.

Type-in: prt Enters prt in the display.

Press: [STORE] This stores prt under f_0, for use as a typing aid.

If you decide not to define a special function key after fetching one, the [STOP] key can also be used to exit key mode.

To list all of the defined special function keys in numerical order, type in: list k and press [EXECUTE].

To list individual special function keys, press LIST and then the special function key to be listed.

Immediate Execute Special Function Keys

If a line to be stored under a special function key is preceded by an asterisk (*), it is an immediate execute key. This means that when the key is pressed, the contents of the key are appended to the display and the line in the display is executed automatically.

For example:

Press: FETCH ⇧ f11 — Accesses f_{23} (shifted f_{11}).

Type-in: `*prt "π", π` — The asterisk makes this an immediate execute key.

Press: STORE — This stores the line entered in the display under f_{23}.

Whenever ⇧ f11 is pressed and the display is clear, the following is printed:

```
π                    3.14
```

Immediate execute keys are useful for executing selected segments of a program. Using the continue command followed by a line number, you can make several entry points in your programs. For example:

```
f0: *cont 5
f10: *cont 10
```

Each time f0 is pressed, the program continues at line 5, or at line 10 if f10 is pressed.

Immediate Continue Special Function Keys

If a line to be stored as a special function key is preceded by a slash (/), it is an immediate continue key for use with the enter statement. "Immediate continue" means that when the key is pressed, the contents of the key are appended to the display and continue is executed automatically. Immediate continue keys are used to enter often used values in enter statements. For example:

Press: FETCH f_{10} — Fetches special function key f_{10}.

Type-in: `/2.71828182846` — This enters the value of e, the base of the natural logarithms, into the display.

Press: STORE — This stores the line in the display under f_{10}.

Whenever an enter statement is waiting for a value and the f_{10} key is pressed, the approximate value for e (i.e., 2.71828182846) is entered and the program continues (see enter statement in Chapter 5).

Keys with Multiple Statements

By separating statements with semicolons, several statements can be stored under one special function key. As an example, suppose you want to convert inches to centimeters. The following line is stored under special function key f_0.

Press: FETCH f_0

Type-in: `*→R; dsp R, "in. =", 2.54R, "cm."`

Press: STORE

Then key in a number, such as 6, and press f_0. The display will show:

```
 6.00 in. =   15.24 cm.
```

Chapter 5

Programming Instructions

The statements, functions, and operators explained in this chapter are all programmable. Most of these instructions can also be used in calculator mode.

Statements can be programmed or executed. Operators and functions must be part of a statement in order to be programmed. This means that operations, such as **10 + 32** or **√63**, which can be executed from the keyboard, must be part of a statement in order to be programmed. Thus, `10 + 32 → X` or `prt √63` are valid statements.

Syntax Conventions

The instructions explained throughout this manual use the following syntax conventions. A complete list of syntax can be found in Appendix A.

[ ] - items within square brackets are optional.
`dot matrix` - items in dot matrix must appear as shown.
... - three dots indicate that the previous item can be duplicated.

Variables

The calculator uses three types of variables: simple variables, array variables, and r-variables. As variables are allocated, they are initially assigned the value 0. Simple variables, r-variables, and array variable elements require 8 bytes * of memory each.

Simple Variables

There can be 26 simple variables, named A through Z. A simple variable must appear in upper case. Each simple variable can be assigned one value. For example:

```
0: 12→A            Assigns the value 12 to A.
1: prt A           Prints the value of A on the printer.
```

Array Variables

There can be 26 arrays, named A through Z. Array names are followed by square brackets which enclose the subscripts of the array (e.g., `L[31]`).

*A byte is the basic unit of data in the 9825A. 8 bytes are required to store a number.

Before an array element can be used, the array must be declared in a dimension (**dim**) statement. This reserves memory for the array and initializes all elements in the array to zero. In the dimension statement, each dimension of an array can be specified either by specifying the upper bound, in which case the lower bound is assumed to be one, or by specifying both the lower and upper bounds. For example:

`dim A[4,5]`	Reserves memory for the 20 elements of the two-dimensional array A.
`dim P[-2:1,-2:2]`	Reserves memory for the 20 elements of the two-dimensional array P. (Lower and upper bounds specified.)

An array can have any size and any number of dimensions within the limits of the memory size and line length. The bounds must be between −32767 and 32767.

An individual element of an array is accessed by specifying the subscripts of the element. For example:

`4 → A[1,5,4,6]`	4 is assigned to element 1,5,4,6 of array A.
`3 → P[-2,1]`	3 is assigned to element −2,1 of array P.

Another Example:

`0: dim Q[10,10]`	Reserves memory for 100 elements of array Q.
`1: 3→Q[7,1]`	Q[7,1] is assigned the value 3.
`2: 5→Q`	The value 5 is assigned to the simple variable Q. There is no connection between the simple variable Q and array Q[10,10].
`3: 2→Q[1,Q]`	Q[1,5] is assigned the value 2.

r-Variables

r-variables are specified by a lower case "r" followed by a value or expression. When an r-variable is encountered, memory is reserved for all r-variables with smaller subscripts which have not been allocated. As r-variables are allocated, they are assigned the value 0. Thus if r10 is assigned a value, r0 through r9 are also automatically allocated and assigned the value zero if they have not been previously allocated.

Examples:

`0: 4→r0`	4 is assigned to r-variable 0.
`1: 2→rr0`	2 is assigned to r-variable 4. r0= 4, therefore 2 → r4. This is known as indirect storage.

Variable Allocation

Simple variables and r-variables are allocated when a statement containing either is executed. Array variables must be allocated using a dimension statement.

Before a variable is allocated, three cases are checked:

1. Before a variable is allocated by the dimension statement, a check is made to see if it is already allocated. If so, an error results and execution stops.
2. When a simple variable is referenced in any other statement, a similar check is made as to whether it has been allocated. If not, it is allocated.
3. When an array element is referenced in any other statement, a similar check is made as to whether the array has been dimensioned. If not, an error results.

Within one statement, variables are allocated in the same left-to-right order as they occur in the statement.

Number Formats

Numbers can be displayed or printed in floating-point format (scientific notation) or in fixed-point format. The calculator's internal representation of numbers is unaffected by number formats, therefore, accuracy is not changed.

When the calculator is turned on, RESET is pressed, or `erase a` is executed, the number format is fixed 2 (**fxd 2**), and for very large numbers, the calculator temporarily prints and displays in float 9(**flt 9**).

The Fixed Statement

Syntax:

`fxd` [number of decimal places]

The fixed (**fxd**) statement sets the format for printing or displaying numbers. In fixed-point format, the number of digits to appear to the right of the decimal point is specified. Fixed 0 through fixed 11 can be specified.

To set the number format from floating-point to the current fixed-point setting, `fxd` without parameters is executed.

When a number of the form:

$A = N \times 10^{E}$

where: $1 \leq N < 10$, or $N = 0$

is too large to fit in the fixed-point format, the number format temporarily reverts to the previously set floating-point (float 9 if no other floating-point format has been set) if:

$D + E \geqslant 14$

where: D is the number of decimal places specified in the fixed statement.
E is the exponent of the number.

To illustrate the reversion to a previous float 9 setting, run this program ♦

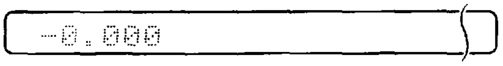
```
0: ent A
1: fxd 0;prt A
2: fxd 1;prt A
3: fxd 2;prt A
4: fxd 3;prt A
5: end
```

If the value `125e10` is entered when `A?` appears in the display, this is printed ♦

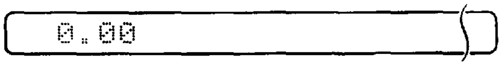
```
  1250000000000
 1250000000000.0
 1.250000000e 12
 1.250000000e 12
```

For numbers too small to fit in the fixed-point format, zeros are printed or displayed for all decimal places, with a minus sign if the number is negative. For example:

```
0: fxd 3;dsp -
 .000125;wait
 2000
1: fxd 2;dsp
 .00204
```



```
-0.000
```



```
0.00
```

Here are some numbers and their output format if `fxd 3` is executed:

Number	Fixed 3 Output
18	18.000
−.000006	-0.000
−2.7532	-2.753
4.5678	4.568
5.3111e3	5311.100
1234567891234.5	1.234567891e 12 (float 9 previously set)

The Float Statement

Syntax:

`flt` [number of decimal places]

The float (**flt**) statement sets floating-point format which is scientific notation. When working with very large or very small numbers, floating-point format is most convenient. Float 0 through float 11 can be specified. To set the number format from fixed-point to the current floating-point setting, `flt` without parameters is executed.

A number output in floating-point format has the form:

−D.D...De−DD

- The left-most non-zero digit of a number is the first digit displayed. If the number is negative, a minus sign precedes this digit; if the number is positive or zero, a space precedes this digit.
- A decimal point follows the first digit; except in **flt 0**.

- Some digits may follow the decimal point; the number of digits is determined by the specified floating-point format (e.g., in **flt 5**, five digits follow the decimal point).
- Then the character e appears, followed by a minus sign or space (for non-negative exponents) and two digits. This is the exponent, representing a positive or negative power of ten. The exponent indicates the direction and the number of places that the decimal point would have to be moved to express the number in fixed-point format.

Here are some numbers as they would appear if `flt 2` is executed:

Number	Float 2 Output
−3.2	-3.20e 00
271	2.71e 02
26.377	2.64e 01
.000004	4.00e-06
2.482e33	2.48e 33

Significant Digits

All numbers are represented internally with 12 significant digits regardless of the number format being used. To illustrate this, execute `fxd 5` then key in the number:

```
123456789.56789
```

then press EXECUTE and note the display:

```
123456789.56700
```

The 13th and 14th digits, 8 and 9, are not stored and zeros are displayed for those digits.

Rounding

A number is rounded before being displayed or printed if there are more digits to the right of the decimal point than the number format allows. The rounding is performed as follows: the first excess digit on the right is checked. If its value is 5 or greater, the digit to the left is incremented (rounded-up) by one; otherwise it is unchanged. The number remains unchanged internally. For instance:

```
0: fxd 2
1: dsp 1.235;
 wait 1000
2: dsp 2.404
3: end
```

```
1.24
```

```
2.40
```

The Display Statement

Syntax:

`dsp` [any combination of text or expressions]

The display (**dsp**) statement displays numbers or text on the calculator display. Commas are used to separate variables or text (e.g., `dsp "No. ", N, B`).

Quotes are used to indicate text. To display quotes within text, it is necessary to press (SHIFT) (" 2) twice for each quote to be displayed. For example:

Type in: `dsp "Say ""Hi"" to her."`

Press: (EXECUTE)

```
Say "Hi" to her.
```

Displayed lines longer than 32 characters can be viewed using the display control keys, (→) and (←).

Numbers and text which are displayed remain in the display until another display operation (such as enter (**ent**) with a prompt) clears it.

The Print Statement

Syntax:

`prt` [any combination of text or expressions]

The print (**prt**) statement is used to print numbers or text on the calculator printer. For example:

```
prt 6                                          6.00
prt "One", 1                  One              1.00
prt "This one"                This one
```

If an expression is to be printed, such as:

`prt 6*7→X`

the expression is evaluated and the equivalent value is printed (and also stored in X in this case).

To print a quote within text press ⇧ " twice for each quote to be printed. For example:

Type in: `prt "Ent ""1"" or ""0"""`

Press: ⇩

```
Ent "1" or "0"
```

Commas are used to separate variables or text. For example:

Type in: `prt "First", 1, "Next", 2`

Press: ⇩

```
First          1.00
Next           2.00
```

When printing lines of text and values, the printout follows this format:

- Text followed by a numeric is printed on the same line if it fits; otherwise the text is printed and the number is printed on the next line.
- Each line of text separated by commas begins on a new line and folds over on successive lines if it is longer than 16 characters.
- Numerics separated by commas are printed one per line unless the format is **flt 10** or **flt 11** which requires two lines each.

When `prt` is specified without parameters, no operation takes place. To space one line, use the space statement on page 47.

The Enter Statement

Syntax:

`ent` [prompt `,`] variable [`,` [prompt `,`] variable...]

The enter (**ent**) statement is used to assign values to variables from the keyboard during a program. The variable can be a simple variable, array variable, or an r-variable. For example:

```
4: ent Q
5: ent A,B[3],
 r11
```

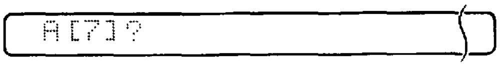
When an enter statement is encountered in a program, key-in a number, variable (such as `r30`), or expression (such as `√5`) and press CONTINUE.

When many items are entered from the keyboard, it is often helpful to have a "prompt" displayed representing the variable being assigned a value. For instance:

```
0: ent "Amount",       Amount
 A
1: ent "Temperat       Temperature
 ure",T
```

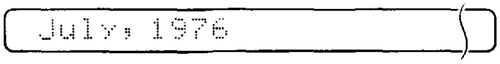
If no prompt is given, the calculator uses the name of the variable as the prompt. For example:

```
3: ent A[7]
```


A[7]?

If a null quote field is given as a prompt, such as `10: ent " ",A` the calculator retains any previously displayed message, unless a print operation is between the display statement and the enter statement. This is useful for variable prompts using the display statement. For example:

```
7: 1976→Y;fxd 0
8: dsp "July,",Y
9: ent "",A
```

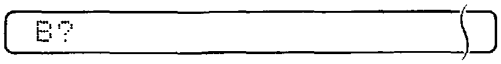

July, 1976

You can calculate values from the keyboard while the program waits in the enter statement. This is done simply by entering the calculation and pressing ↓. If the value to be entered is the result of pressing ↓, press RESULT or RECALL then press CONTINUE. Pressing ↓ immediately before pressing CONTINUE causes a default condition as if CONTINUE were pressed without entering a value.

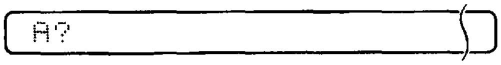
Complex lines can be entered as the response to an enter statement. For instance, run this program:

```
0: ent B
1: ent A
2: prt A
3: end
```

When the display is:


B?

enter a value for B. Then when the display is:


A?

Type in: `20; if B>20; 40`

Then press CONTINUE. If the value that you entered for B is greater than 20, then `40` is printed, otherwise `20` is printed.

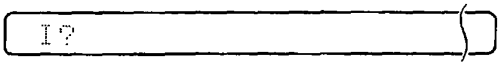
If CONTINUE is pressed without entering a value, the variable maintains its previous value and flag 13 is set. When a value is entered, flag 13 is cleared (see flags on page 59).

To terminate a program during an enter statement, press STOP. The rest of the program line is completed before the calculator stops.

Commands, such as `fetch` or `run`, are not allowed during an enter statement and cause error 03.

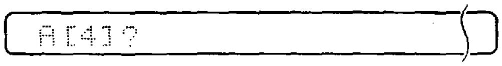
The following example illustrates a unique case using the enter statement. Run the short program:

```
0: dim A[20]
1: 4→I
2: ent I,A[I]
```



I?

Type in: 8



A[4]?

Press: CONTINUE

Notice that the value of I when the enter statement is encountered is used, not the entered value of I. To use the entered value of I as the subscript, use another enter statement. For the above example, change line 2 to:

```
2: ent I;ent
 A[I]
```

Even though you can have one enter statement that enters values for several variables, only one value can be supplied at a time. For example:

```
0: ent A,B
```

type in a value for A when a A? appears in the display and press CONTINUE, then do the same when B? appears in the display.

The Enter Print Statement

Syntax:

enp [prompt ,] variable [, [prompt ,] variable...]

The enter print (**enp**) statement is the same as the enter statement except that prompts and the entered values are printed and displayed as they are encountered.

For example, type in this short program to calculate the area of a circle:

```
0: enp "radius",
 R
1: πRR→A
2: prt "area",A
3: end
```

If 2 is entered for R when the program is run, the printout will be:

```
radius
2
area          12.57
```

The Space Statement

Syntax:

`spc` [number of blank lines]

The space (**spc**) statement causes the printer to output the number of blank lines indicated. The number of lines can be an expression with a range of 0 through 32767. If no parameter is specified, one blank line is output.

Examples:

```
0: spc A+B        Space the number of lines specified by A + B.
1: spc 5          Space 5 lines.
2: spc            Space one line.
```

The Beep Statement

Syntax:

`beep`

The beep statement causes the calculator to output a beep. For example, the calculator normally beeps, displays `error 67`, and stops when the argument of the square root (√) function is negative. In the following short program, the value entered for A is tested. If it is negative, the calculator beeps and displays a message, but the program continues entering values.

```
0: fxd 4
1: "start":ent
 "Argument",A
2: if A<0;gto
 "error"
3: prt √A;gto
 "start"
4: "error":beep
5: dsp "√ of
 neg. no."
6: wait 2000
7: gto "start"
```

The Wait Statement

Syntax:

`wait` number of milliseconds

The wait statement causes a program to pause the specified number of milliseconds (thousandths of a second). The wait statement is often used with display or enter statements to display a message for a specified time. The number of milliseconds can be an expression. The maximum wait is around 33 seconds, which is specified by the value 32767.

Since the wait statement takes time to be executed, small values in the wait statement are actually longer than a millisecond. This becomes evident in a loop which is executed many times.

Examples:

```
wait 2000
```

Pauses for 2 seconds.

```
   2: wait 2*I
```

Pauses for 2*I milliseconds.

In the next example, a display statement is followed by an enter statement. To preserve the first display for one second, the wait statement is used.

```
10: dsp "Please
 enter";wait
 1000
11: ent "value
 of X",X
```

The first display remains one second before the next display.

The Stop Statement

Syntax:

```
stp
```

The stop (**stp**) statement stops program execution at the end of the line in which it is executed. Pressing CONTINUE continues the program at the next program line. STEP can also be used to "step" through the program one line at a time. If any editing is performed after the program stops, CONTINUE and STEP cause the program to continue from line 0.

The stop statement can also be used for debugging. See the section on debugging statements for details.

The End Statement

Syntax:

```
end
```

The end statement causes the program to stop like the stop statement. However, the end statement resets the program line counter to line 0 and resets all subroutine return pointers (see go sub statement). The end statement is usually put at the end of a program. The end statement cannot be executed during an enter statement, nor in live keyboard mode.

Hierarchy

In a statement containing functions, arithmetic operations, relational operations, logical operations, imbedded assignments, or flag operations, there is an order in which the statement is executed. This order is called the hierarchy, which is:

highest priority	functions, flag references, r-variables
	↑ (exponentiation)
	implied multiply
	− (unary minus)
	* / mod
	+ −
	all relational operators (=, >, <, <=, >=, #)
	not
	and
lowest priority	or, xor

An expression is scanned from left to right. Each operator is compared to the operator on its right. If the operator on the right has a higher priority, then that operator is compared to the next operator on its right. This continues until an operator of equal or lower priority is encountered. The highest priority operation, or the first of the two equal operations, is performed. Then any lower priority operations on the left are compared to the next operator to the right. If parentheses are encountered, the expression within the parentheses is evaluated before the left-to-right comparison continues. This comparison continues until the entire expression is evaluated. In the following example, S_1, S_2, S_3... indicate intermediate results:

2A = B + C − D mod E exp (not F)	implied multiplication
S_1 = B + C − D mod E exp (not F)	addition
S_1 = S_2 − D mod E exp (not F)	evaluate parenthesis
S_1 = S_2 − D mod E exp S_3	exp function
S_1 = S_2 − D mod E S_4	implied multiplication
S_1 =S_2 − D mod S_5	mod operator
S_1 = S_2 − S_6	subtraction
S_1 = S_7	equality relation
S_8	final result

Operators

The four groups of mathematical or logical symbols, called operators, are: the assignment operator, arithmetic operators, relational operators, and logical operators.

Assignment Operator

Syntax:

expression → variable

The assignment operator is used to assign values to variables. For example:

`1.4→A`	The value 1.4 is assigned to the variable A.
`3: B→A`	The value of B is assigned to the variable A.

There are other ways to assign values to variables such as the enter (**ent**) statement or the load file (**ldf**) statement.

To assign the same value to many variables, the assignment operator can be used as in this example.

```
32 → A → B → X → r4
```

Multiple assignments can also take the form `(25→A) + 1 →B` **(which is the same as** `25→A; A + 1 →B`). This is called an imbedded assignment.

Arithmetic Operators

There are six arithmetic operators as follows:

+	Add (if unary, no operation)	A + B or +A
−	Subtract (if unary, change sign)	A − B or −A
*	Multiply	A*B
/	Divide	A/B
↑	Exponentiate	A^B
mod	Modulus	A mod B is the remainder of A/B when A and B are integers. A mod B is the same as A −int (A/B)*B.

When A is much larger than B, there is a chance that a value of 0 could be returned for AmodB. This condition can be caught by examining the exponent of A/B when it is represented in floating point notation with one digit to the left of the decimal point. If the exponent is greater than 8, AmodB results in a value of 0.

Besides the * symbol for multiplication, implied multiplication can be used. In the following instances, implied multiplication takes place:

- Two variables together (like AB).
- A variable next to a number (like 5A).
- A variable or number next to a parenthesis [like 5(A + B)].
- A parenthesis next to a parenthesis [like (A + B) (X + Y)].
- A variable, number, or parenthesis preceding a function name (like 32 sinA).

For example:

AB→X	A times B is stored in X.
(5)5→X	5 times 5 is stored in X.
A(B+C)→B	A times the sum B + C is stored in B.
5abs B	5 times the absolute value of B.

Relational Operators

There are six relational operators as shown in the following table.

Symbols	Meaning
=	Equal to.
>	Greater than.
<	Less than.
=> or >=	Greater than or equal to (either form is acceptable).
=< or <=	Less than or equal to (either form is acceptable).
# or < > or ><	Not equal to (either form is acceptable).

The result of a relational operation is either a one (if the relation is true) or a zero (if it is false). Thus if A is less than B, then the relational expression A<=B, is true and results in a value of one. All comparisons are made on 12 significant digits, signs, and exponents.

The relational operators can be used in any statement which allows expressions as arguments. For example:

`A=B → C` Assignment statement. If A and B are equal, a 1 is stored in C; otherwise, a 0 is stored in C.

`if A>B;...` If statement. If A is greater than B, then continue in the line; but if A is less than or equal to B, go to the next line.

`...;jmp A>3` Jump statement. If A is greater than 3, jump 1 line, otherwise jump to the beginning of the line (jmp 0).

`prt A(A>B)+B(A<B)` Print statement. If A is greater than B, the value of A is printed. If A is less than B, then the value of B is printed. If A equals B, then 0 is printed.

Logical Operators

The four logical operators, **and**, **or**, **xor** (exclusive or), and **not** are useful for evaluating Boolean expressions. Any value other than zero (false) is evaluated as true. The result of a logical operation is either zero or one.

Operation	Syntax	Truth Table		
AND	expression `and` expression	A	B	A and B
		F	F	0
		F	T	0
		T	F	0
		T	T	1
OR	expression `or` expression	A	B	A or B
		F	F	0
		F	T	1
		T	F	1
		T	T	1

Operation	Syntax	Truth Table		
Exclusive OR	expression xor expression	A	B	A xor B
		F	F	0
		F	T	1
		T	F	1
		T	T	0
NOT	not expression	A		not A
		F		1
		T		0

For example:

Program:

```
0: .1→A;0→B
1: prt "A and
 B",A and B
2: prt "A or B",
 A or B
3: prt "A xor
 B",A xor B
4: prt "not A",
 not A
5: end
```

Printout:

```
A and B       0.00
A or B        1.00
A xor B       1.00
not A         0.00
```

Math Functions and Statements

The math functions and math statements are explained in this section.

Parentheses must enclose the argument of a function when a "+" or "−" sign precedes the argument. In the examples, parentheses are shown only where they are required.

General Functions

Syntax	Description	Examples (fxd 5)
√ expression	Returns the square root of a non-negative expression. For negative expressions, see the section on math errors on page 57.	√64 = 8.00000 √π = 1.77245
abs expression	Determines the absolute value of the expression.	abs (−3.09) = 3.09000 abs 330.1 = 330.10000

Syntax	Description	Examples (fxd 5)
sgn expression	The sign function returns a −1 for negative expressions, 0 if the expression equals 0, and 1 for a positive expression.	sgn (−18) = −1.00000 sgn 0 = 0.00000 sgn 34 = 1.00000
int expression	Returns the largest integer less than or equal to the expression. This is often referred to as the "floor" integer value of the expression.	int 2.718 = 2.00000 int (−3.24) = −4.00000
frc expression	Gives the fractional part of a number. It is defined by: expression −int expression	frc 2.718 = 0.71800 frc (−3.24) = 0.76000
prnd (expression, rounding specification)	Returns the value of the argument rounded to the power-of-ten position indicated by the rounding specification.	prnd (127.375, −2) = 127.38000 127.375 is rounded to the nearest hundredth (10^{-2})
drnd (expression, number of digits)	The digit round function rounds the argument to the number of digits specified. The leftmost significant digit is digit number 1.	drnd (73.0625, 5) = 73.06300 drnd (−65023, 1) = −70000.00000 drnd (.055, 1) = 0.06000
min (list of expressions and arrays)	Returns the smallest value in the list. An entire array can be specified by substituting an asterisk for the array subscript list (such as B[*]).	0: dim A[3]; 2→A[1] 1: 9→A[2]; 3→A[3] min (A[*]) = 2.00000 min (2, −3, −3, 4) = −3.00000
max (list of expressions and arrays)	Returns the largest value in the list. An entire array can be specified by substituting an asterisk for the array subscript list (such as B[*]).	0: dim A[3]; 2→A[1] 1: 9→A[2]; 3→A[3] max (A[*]) = 9.00000 max (5, 4, −3, 8) = 8.00000
rnd [−] expression	The random number function generates a pseudo-random number greater than or equal to 0 and less than 1. When the argument is positive, the starting seed is π/180 (which is .0174532925200). This seed is initialized when the calculator is turned on, erase a is executed, or RESET is pressed. Each subsequent access to the rnd function with a positive argument uses a seed based on the previous result of the function.	rnd 1 = 0.67822

Syntax	Description	Examples (fxd 5)
	To specify a starting seed other than π/180, use a negative argument. The fractional part of the absolute value of the argument is used as the seed. To obtain a good seed use a number less than 0 and greater than −1. The more non-zero digits in the number, the better. Last digits of 1, 3, 7, or 9 are preferable.	0: wait rnd (−.31317) 1: rnd 1→A Note that the wait statement is used instead of an assignment statement to initialize the starting seed. Line 1 generates a random number based on .31317 instead of π/180.

Logarithmic and Exponential Functions

Syntax	Description	Examples (fxd 5)
ln expression	The natural logarithm function calculates the logarithm (base e) of a positive valued expression.	ln 8001 = 8.98732 ln .0026 = −5.95224
exp expression	The exponential function raises the constant, naperian e, to the power of the computed expression. The range of the argument is approximately from −227.95 through 230.25.	exp 1 = 2.71828 exp (−3) = .04979
log expression	The common logarithm function calculates the logarithm (base 10) of a positive valued expression.	log 305.2 = 2.48458 log .0049 = −2.30980
tn↑ expression	The ten-to-the-power function raises the constant, 10, to the power of the computed expression. The range of the argument is approximately from −99 through 99.999. This function executes faster than: 10↑ expression.	5 tn↑2 = 500.00000 tn↑ (−3) = 0.00100

The math errors and default value associated with the log and ln (natural log) functions are explained in detail on page 57.

Trigonometric Functions and Statements

The angular units: degrees, radians, or grads, are set by statements explained in this section. Degrees are automatically set when the calculator is switched on, erase a is executed, or RESET is pressed.

deg This statement sets degrees for all calculations which involve angles. A degree is 1/360th of a circle.

rad This statement sets radians for all calculations which involve angles. There are 2π radians in a circle.

grad This statement sets grads for all calculations which involve angles. A grad is 1/400th of a circle.

units This statement displays the current angular units.

Syntax	Description	Examples (fxd 5)
sin expression	Determines the sine of the angle represented by the expression in the current angular units.	deg; sin 45 = 0.70711 rad; sin (π/6) = 0.50000 grad; sin (-70) = −0.89101
cos expression	Determines the cosine of the angle represented by the expression in the current angular units.	deg; cos 45 = 0.70711 rad; cos (π/6) = 0.86603 grad; cos (-70) = 0.45399
tan expression	Determines the tangent of the angle represented by the expression in the current angular units.	deg; tan 45 = 1.00000 rad; tan (π/4) = 1.00000 grad; tan 50 = 1.00000
asn expression	Returns the principal value of the arcsine of the expression in the current angular units. The range of the argument is −1 through +1. The range of the result is −π/2 to +π/2 (radians), −90 to +90 (degrees), or −100 to +100 (grads).	deg; asn .8 = 53.13010 rad; asn .8 = 0.92730 grad; asn .8 = 59.03345

Syntax	Description	Examples (fxd 5)
acs expression	Returns the principal value of the arccosine of the expression in the current angular units. The range of the argument is −1 through +1. The range of the result is 0 to π (radians), 0 to 180 (degrees), or 0 to 200 (grads).	deg; acs (−.4) = 113.57818 rad; acs (−.4) = 1.98231 grad; acs (−.4) = 126.19798
atn expression	Calculates the principal value of the arctangent of the expression in the current angular units. The range of the result is −π/2 to +π/2 (radians), −90 to +90 (degrees), or −100 to +100 (grads).	deg; atn 20 = 87.13759 rad; atn 20 = 1.52084 grad; atn 20 = 96.81955

Math Errors

Errors 66 through 77 are displayed when a math error occurs. In this section, the default values of math operations which result in an error are explained. Whenever a math error occurs, flag 15 is set automatically. If you set flag 14, math operations which normally cause an error to be displayed, result in a default value.

When printing, displaying, or storing a default value outside the storage range, the value is converted to an appropriate value of ±9.99999999999e 99.

error 66 Division by zero. The default value is +9.99999999999e 511 if the dividend is positive and −9.99999999999e 511 if the dividend is negative. For example:

−9.5/0 = −9.99999999999e 511

A mod B with B equal to zero. The default value is 0. For example:

32 mod 0 = 0

error 67 Square root of a negative number. The default value is $\sqrt{\text{(abs (argument))}}$. For example:

√ (−36) = 6.

error 68 Tangent of (n×π/2 radians);
Tangent of (n×90 degrees);
Tangent of (n×100 grads);
where n is an odd integer. The default value is 9.99999999999e 511 if n is positive; and −9.99999999999e 511 if n is negative. For example:

rad; tan (−π/2) = −9.99999999999e 511
deg; tan 270 = 9.99999999999e 511
grad; tan 300 = 9.99999999999e 511

error 69 In or log of a negative number. The default is:
In (abs (argument)) or log (abs (argument))
respectively. For example:

ln (−301) = 5.70711
log (−.001) = −3.00000

error 70 In or log of zero. The default value is −9.99999999999e 511. For example:

ln 0 = −9.99999999999e 511
log 0 = −9.99999999999e 511

error 71 asn or acs of a number less than −1 or greater than 1. The default value is
asn (sgn (argument)) or acs (sgn (argument))
respectively. For example (in degrees):

deg; asn (−10) = −90
deg; acs (1.5) = 0

error 72 Negative base to a non-integer power. The default value is
(abs (base)) ↑ (non-integer power) For example:

(−36) ↑ (.5) = 6

error 73 Zero to the zero power (0 ↑ 0). The default value is 1.

error 74 Storage range overflow. The default value is 9.99999999999e 99 or −9.99999999999e 99. For example:

(1e 62) * (1e 38) → A; A will equal 9.99999999999e 99.
(−1e 25) ↑ 5 → B; B will equal −9.93999999999e 99.

error 75 Storage range underflow. The default value is zero. For example:

(1e−66) * (4e−35) → A; A will equal 0

error 76 Calculation range overflow. The default value is 9.99999999999e 511 or −9.99999999999e 511. For example:

(1e 99) ↑ 6 = 9.99999999999e 511
(−1e 99) ↑ 7 = −9.99999999999e 511

error 77 Calculation range underflow. The default value is zero. For example:

(1e−10) ↑ 60 = 0

Flags

Flags are programmable indicators that can have a value of one or zero. When a flag is set, its value is one; when it is cleared, its value is zero. There are 16 flags, numbered 0 through 15. The following flags have special meanings:

Flag 13 - is automatically set when CONTINUE is pressed without entering data in an enter statement or when STOP is pressed in an enter statement. Flag 13 is automatically cleared when data is supplied in an enter statement.

Flag 14 - when flag 14 is set, the calculator ignores math errors such as division by zero and supplies a default value shown in the list beginning on page 57.

Flag 15 - is automatically set whenever a math error occurs, regardless of the setting of flag 14.

The Set Flag Statement

Syntax:

sfg [flag number, ...]

The set flag (**sfg**) statement sets the value of the specified flags to one. The flag number can be a value or an expression. If a non-integer flag number is specified, the value is rounded to an integer. If sfg is executed with no flag number specified, all flags (0 through 15) are set. For example:

sfg 2	Set flag 2.
0: sfg A+1	Set the flag designated by A + 1.
1: sfg 1,X	Set flag 1 and the flag designated by X.

The Clear Flag Statement

Syntax:

```
cfg [flag number, ...]
```

The clear flag (**cfg**) statement clears the specified flags to zero. The flag number can be a value or expression. If a non-integer flag number is specified, the value is rounded to an integer. If `cfg` is executed with no flag numbers specified, all flags (0 through 15) are cleared.

Examples:

`cfg 14`	Clear flag 14.
`3: cfg flg2`	Clear the flag designated by the value of flag 2 (either flag one or flag zero will be cleared).
`4: cfg`	Clears all flags.

The Complement Flag Statement

Syntax:

```
cmf [flag number, ...]
```

The complement flag (**cmf**) statement changes (toggles) the value of the flags specified. If a set flag is complemented, its new value is zero. If a cleared flag is complemented, its new value is one. A value or expression can be given for the flag number. If a non-integer flag number is specified, the value is rounded to an integer. To complement flags 0 through 15, `cmf` is executed without paramenters.

Examples:

`cmf 1`	Complement flag 1.
`0: cmf X-1`	Complement the flag designated by X−1.
`1: cmf 3,4,5`	Complement flags 3, 4, and 5.

The Flag Function

Syntax:

`flg` flag number

The flag (**flg**) function is used to check the value of a flag. The result of the flag function is zero or one. One indicates a set flag; zero indicates a cleared flag.

Examples:

`4: if flg2;jmp 5`	If flag 2 is set, jump 5 lines.
`5: flg15→A`	If flag 15 is set, 1→A; if flag 15 is cleared, 0→A.

Branching Statements

Branching statements are used to alter the sequential flow of a program. Branching is used for such operations as looping through a section of a program, executing a subroutine program, and branching to different parts of a program based on a decision (**if**) statement. There are three statements used for branching: the go to (**gto**) statement, the jump (**jmp**) statement, and the go sub (**gsb**) statement.

The following three types of branching may be used for both go to and go sub statements:

Absolute Branching	-branch to the specified line number (such as `gto 10`).
Relative Branching	- branch forward or backward in the program the specified number of lines relative to the current line (such as `gsb -3`).
Labelled Branching	-branch to the indicated label. This type of branching is generally the most convenient to use since the programmer doesn't have to know line numbers for a branch (such as `gto "First"`).

Line Renumbering

Line numbers are automatically renumbered when a program line is inserted or deleted. As lines are inserted or deleted in a program, the line numbers of relative or absolute go to or go sub statements are changed as required to reflect the insertion or deletion. The address in the jump statement is not changed. The entire program is checked before any deletion is made. If a line being deleted is the destination of a relative or absolute go to or go sub statement, an error is displayed and no deletion occurs, unless an asterisk (*) is used in the delete command (see page 80).

An error message is not displayed when the line containing a label name in a gto statement is deleted.

If a line becomes too long due to line renumbering, the line number for tha' line will appear followed by a ? when the line is displayed or listed. For example:

```
8: ... ;gto 99
```

Line 8 was stored with 73 characters.

Inserting a line at line 7 causes line 8 to be renumbered such that the branch is to line 100. The line will appear as:

```
9: ?...
```

To view the entire line, delete an appropriate line to recover the original line numbering. The fact that a line is too long to display or list does not affect the operation of the program when the program is run.

More information on line renumbering is in Chapter 6.

Labels

Labels are characters within quotes located either at the beginning of a line, after a go to or go sub statement, or after a run or continue command. Labels at the beginning of a line must be followed by a colon.

Labels are used for branching and for remarks within a program. When used for branching, the label in the go to or go sub statement is compared to the line labels in the program until a match is found. Then, at the end of the line, a branch is made to the line containing the label. The first time a branch is made to a label, the program is scanned beginning at line 0 until a matching label is found. From then on, the branch is directly to the line with that label. When comparing labels for branching, a comparison is made on all characters in the label, including blanks.

Labels are often used to make remarks in a program for documentation purposes.
For example:

```
7: "Area":
8: πR↑2→A
```

Note that a colon must follow a label even if nothing else is in the line.

The Go To Statement

The go to (**gto**) statement causes program control to transfer to the location indicated. When a line contains more than one go to statement, only the last one encountered is executed.

Absolute Go To

Syntax:

`gto` line number

An absolute go to statement is used to branch to the indicated line. The line number must be an integer (such as 5 or 13).

When an absolute or labelled go to statement is executed from the keyboard in calculator mode, the program line counter is set to the specified line number. To view the line, press the [↑] key.

Relative Go To

Syntax:

`gto +` number of lines
`gto –` number of lines

A relative go to statement is used to branch forward (+) or backward (–) the specified number of lines, relative to the current line. The number of lines must be an integer.

Examples:

`20: gto +1`	Go forward 1 line.
`21: gto -3`	Go back 3 lines.

`22: gto +0`
`23: gto -0`
	Go to the beginning of the current line.

Labelled Go To

Syntax:

`gto` label

A labelled go to statement is used to branch to the line with the indicated label (see section on labels). This is the most convenient type of branching since no line numbers have to be considered.

Example:

`gto "Avg."`	Go to the line labelled by "Avg.".

When a labelled go to statement is executed from the keyboard in calculator mode, the program line counter is set to the specified line number. To view the line, press the (↑) key.

Multiple go to statements in a line are useful for N-way branching when used with an if statement. N-way branching is explained on page 68.

The Jump Statement

Syntax:

`jmp` number of lines

The jump (**jmp**) statement allows branching from the current line by the number of lines specified. This statement is similar to the relative go to statement except that the number of lines can be an expression. If the number of lines is positive, the branch is forward in the program. If the number of lines is zero, the branch is to the beginning of the current line. If the number of lines is negative, the branch is backward in the program. If the number of lines is not an integer, then it is rounded to an integer.

The go to statement executes faster than the jump statement. The jump statement can only be at the end of a line, otherwise error 07 is displayed when you try to store or execute the line.

Examples:

`10: jmp 10`	Jump forward 10 lines.
`19: jmp A`	Jump the number of lines designated by the value of A.
`28: jmp (Z=2)2`	Jump forward 2 lines if Z=2; otherwise jump to the beginning of the current line.
`33: ... ;jmp(B+1→B)>20`	Increment B and jump to the next line if B is greater than 20; otherwise jump to the beginning of the current line.

The Go To Subroutine and Return Statements

The go to subroutine (**gsb**) statement allows branching to subroutine portions of a program. Subroutines are useful when the same routine will be executed many times and called from different places in the program. A return pointer is set up when the go sub statement is executed. This pointer points to the next line after the line containing the go sub statement. The return (**ret**) statement returns the program execution to the pointer location. The return statement is the last statement executed in the subroutine and must be the last statement in a line. The depth of subroutine nesting is limited only by the amount of available memory. Each subroutine return pointer requires eight bytes of memory. Subroutines should be entered only by a **gsb** statement and should be exited only by a **ret** statement.

When a line contains more than one go sub statement, only the last one encountered is executed. There are three types of go sub statements: absolute, relative, and labelled.

Absolute Go Sub

Syntax:

gsb line number

An absolute go sub statement is used to go to the subroutine at the specified line number. The line number must be an integer.

Example:

`7: gsb 15`	Go to the subroutine at line 15.
`18: ret`	End subroutine with return statement (program returns to line 8).

Relative Go Sub

Syntax:

gsb + number of lines
gsb − number of lines

A relative go sub statement provides forward (+) or backward (−) subroutine branching the specified number of lines, relative to the current line number. The number of lines must be an integer.

Examples:

```
7: gsb +5
```
Go to the subroutine at line 12.

```
8: gsb -3
```
Go to the subroutine at line 5.

Labelled Go Sub

Syntax:

gsb label

A labelled go sub statement is used to branch to the subroutine at the indicated label. This is the most convenient form of subroutine branching since no line numbers need to be considered.

Example:

```
3: gsb "sub1"
```
Go to the subroutine at the line labelled by "sub1".

Multiple go sub statements in a line are useful for N-way branching when used with the if statement. N-way branching is explained on page 68.

Calculated Gosub Branching

By using the jump statement and the go sub statement together, calculated branching to subroutines is possible. This form of subroutine branching is called the calculated go sub and has the form:

gsb dummy location ; jmp expression

The dummy location can be a line number, + or − a number of lines, or a label, but the calculator branches to the subroutine designated by the computed jump expression. For example:

```
0: ent N
1: gsb "X";jmp N
2: prt "end"
3: "X":end
4: prt "sub1";
 ret
5: prt "sub2";
 ret
6: prt "sub3";
 ret
```

If a 3 is entered for N, the program branches to the subroutine at line 4.

The If Statement

Syntax:

if expression

The if statement is used to branch based on a logical decision. When an if statement is encountered, the expression following it is evaluated. If the computed expression is zero (false), program control resumes at the next program line (unless the preceding statement was a go to or go sub statement as explained later under N-Way Branching). If the computed expression is any other value, it is considered true, and the program continues in the same line. The if statement is most often used with expressions containing relational operators or flags.

Example:

```
0: ent A,B
```

Enter a value for A and B.

```
1: if A=B;gto
 "one"
2: gto "zero"
```

If A=B, go to "one"; otherwise go to 'zero".

```
3: "one":dsp
 "A=B"
4: stp
```

At label "one", display A=B; then stop.

```
5: "zero":dsp
 "A#B"
6: end
```

At label "zero", display A#B; then end the program.

Whenever A and B are equal, A = B is displayed. All other times, A # B is displayed.

The if statement can be used with other statements besides the go to statement used in the above example. The previous example could be shortened to:

```
0: ent A,B
1: if A=B;dsp
 "A=B";stp
2: dsp "A#B"
3: end
```

Note that no go to statements are used.

N-Way Branching

The if statement used with a go to or go sub statement makes it possible to branch to any of several locations. This type of branching is referred to as n-way branching, and has the following forms:

```
gto...; if...; gto...
```

or

```
gsb...; if...; gsb...
```

If the first if statement is false, then the branch is determined by the first go to or go sub statement. If the first if statement is true, the second go to or go sub statement determines the branch. Go to and go sub statements can be mixed in the same line.

When a line contains more than one go to or go sub statement, only the last one encountered is used. An if statement whose expression is zero can abort execution of the remainder of a line (before subsequent go to or go sub statements are encountered).

Example:

```
20: gto 24;if
 X>30;gto 32;if
 X>40;gto "max"
21:
```

If X is less than or equal to 30, the program branches to line 24. If X is greater than 30 and less than or equal to 40, the branch is to line 32. If X is greater than 40, the branch is to the line labelled "max".

The Dimension Statement

Syntax:

`dim` item [, item, ...]

item may be: simple variable
array variable [dimension [, dimension, ...]]

The dimension (dim) statement reserves memory for simple and array variables, and initializes the indicated variables to zero. r-variables can not be dimensioned in a dimension statement.

In the dimension statement, the dimensions of an array can be specified by expressions. For example:

```
0: ent N,I,r2
1: dim A[N,I],           Variables are used to specify dimensions.
 B[r2],C[3,2*N]
```

Variables are allocated in the order that they appear. If a variable is allocated already, an error results. All the variables dimensioned in any one dimension statement are stored in a contiguous block of memory. This is important when recording data.

Dimension statements may appear anywhere in a program but any dimension statement can only be executed once during a program. The number of dimension statements is limited by memory size. The number of dimensions and the size of the dimensions of an array is limited only by memory size and line length. For example:

```
0: dim N[2,2,2,          Reserves 128 array elements.
 2,2,2,2]
1: dim M[1000]           Reserves 1000 array elements.
```

Specifying Bounds for Dimensions

A dimension may be specified by giving lower and upper bounds. The lower bound must be specified before the upper bound. The two are separated by a colon. The bounds must be in the range from −32767 through 32767. For example:

```
0: dim S[-3:0,           Reserves 12 array elements.
 4:6]
```

This statement reserves the same amount of memory as:

```
0: dim X[4,3]            Reserves 12 array elements.
```

The elements of array S are referenced as:

S[−3,4]	S[−3,5]	S[−3,6]
S[−2,4]	S[−2,5]	S[−2,6]
S[−1,4]	S[−1,5]	S[−1,6]
S[0,4]	S[0,5]	S[0,6]

If a lower bound is not specified, as in X[4,3], it is assumed to be 1, the same as X[1:4,1:3].

The Clear Simple Variables Statement

Syntax:

```
csv
```

The clear simple variables (**csv**) statement clears any allocated simple variables to zero. The clear simple variables statement does not de-allocate variables. Therefore, an error results when the following line is executed:

`0: 7→A;csv;dim A`	Not allowed. Cannot allocate A twice.

The List Statement

Syntax:

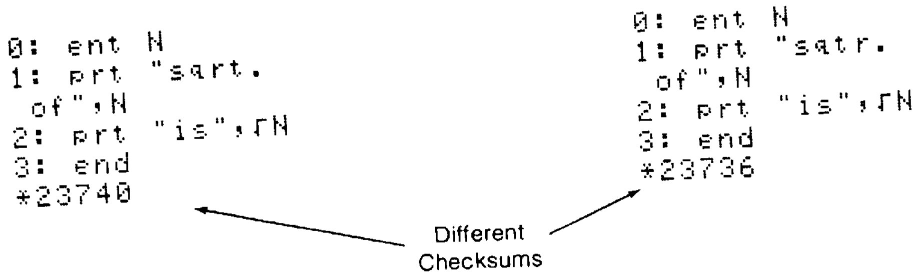
`list` [beginning line number [`,` ending line number]]
`list` special function key
`list k`

The list statement is used to obtain a printed listing of a stored program, section of a program, or special function keys. If no parameter follows the list statement, the entire program is listed. If one line number is specified, the program is listed from that line to the end. If two line numbers are specified, the program segment between the two line numbers is listed. To list all of the special function keys, execute `list k` (for list keys). When list is followed by pressing an individual special function key, then only that key is listed (this is not programmable). The list statement must be the last statement in a line.

Examples:

`0: list`	Lists the entire program.
`list 10,15`	List lines 10 through 15.
`list 4,4`	List line 4.
`list k`	List the special function keys.
`list` f10	List special function key f10 (not programmable).

At the end of a listing, a checksum is printed. This checksum is useful for detecting inte changed or omitted lines and characters. Any difference in the programs generates a differen checksum. In the following two programs, only the characters `rt` in line 1 are interchange Note that the checksums are different. There is no change in checksum from machine t machine, with different memory sizes, nor with different ROMs.



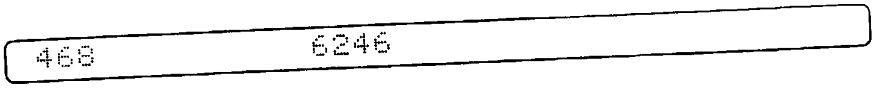
```
0: ent N
1: prt "sqrt.
 of",N
2: prt "is",√N
3: end
*23740
```

```
0: ent N
1: prt "sqtr.
 of",N
2: prt "is",√N
3: end
*23736
```

Different Checksums

Used and Remaining Memory

After a list operation, two numbers are displayed. The first number is the total length of the program in bytes*. This number doesn't include variables, subroutine return pointers, etc. The second number is the unused memory in bytes. For example:



```
 468              6246
```

Program Length Unused Memory

(in bytes)

*A byte is the basic unit of data in the 9825A. 8 bits make up one byte. 8 bytes are required to store a number.

Chapter 6
Debugging

Debugging is the process of refining a program by editing, correcting, and updating. Like programming, it is a creative process. Many operations are involved such as deleting and inserting lines and changing, inserting, and deleting characters. Selective tracing and selective stopping are useful for locating lines which require changes. STEP is useful for going through a program one line at a time. This chapter explains some of the steps in editing a program.

Finding the Problem

The first step in debugging is to find the lines which require changes. This can be done in several ways. One way is to step through a program by pressing STEP once for each line to be executed. Then check the results after each executed program line.

Another way is to use the trace, stop, and normal statements. When program lines are traced, the line number, and variables and flags which are assigned values are printed. This allows you to monitor program activity in individual program lines. Using the stop statement, the program can be stopped whenever a specified program line is encountered. The normal statement is used to terminate tracing and stopping. Stop, trace,and normal statements are explained later.

Fixing the Problem

The next step in debugging is fixing the problem. In many cases, this is as simple as changing one character. Fixing the problem could, however, require rewriting many program lines.

To modify characters within a line, fetch the line by pressing the FETCH key followed by the line number of the line requiring the change. Then press ⏎. The line will appear in the display. Next press either BACK, if the change is closer to the end of the display, or FWD, if the change is closer to the front. Once a flashing cursor is over the location needing correction, you can either insert characters, delete characters, or write over the existing characters. To insert characters, press the INS RPL key. This changes the flashing ▮ to a flashing ◂. Characters that are typed-in are inserted at the left of this cursor. To delete characters, press character DELETE for each character to be deleted. To replace characters, be sure the ▮ cursor is in the display (if the ◂ is in the display, press INS RPL to get ▮) and then enter the necessary characters.

To modify lines within a program, use the FETCH key or the ↑ and ↓ keys to bring the line into the display. To delete the line, press the line DELETE key.

If a line being deleted has a line number referenced by a go to or go sub statement, an error 36 will occur. Either execute the delete command with the optional asterisk (*) parameter (see page 80), or adjust the line reference in the go to or go sub statement accessing that line. In the following example, line 25 is to be deleted; but line 25 is referenced from line 27. Two alternatives are shown.

Program section:

```
25: prt "number"
 ,N
26: if N=27;stp
27: N+1→N;gto 25
```

Alternative 1:

Type in: `del 25,*`

Press: □

Deletes line 25 only. The go to statement in line 27 still addresses line 25.

Alternative 2:

Change line 27 to:

```
27: N+1→N;gto 26
```

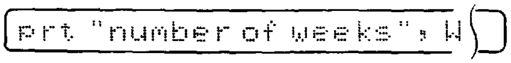
Then fetch line 25 and press line DELETE, or execute `del 25`.

To insert a line, fetch the line that the inserted line is to precede. Then type the new line into the display and press the line INSERT key to store it. All the lines from the fetched line on are automatically renumbered (incremented by one). When a line is inserted, the line references of go to or go sub statements are incremented to reflect the new line numbering. If the line being inserted contains an absolute go to or go sub statement, it is assumed that the line numbers reference the lines before they are renumbered. Thus, if a line inserted before line 30 contains a `gto 45` statement, it will be renumbered to `gto 46`. (The old line 45 is renumbered to line 46.)

In this example, a line is inserted between lines 14 and 15 ➧

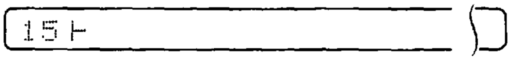
```
14: prt "number
 of days",D
15: gto 19
```

First, fetch line 15, then type the line to be inserted into the display ➧



```
prt "number of weeks", W
```

Then press the line INSERT key. The display will be ➧



```
15 ⊢
```

To see where the line was inserted, execute: `list 14,16`

```
14: prt "number
 of days",D
15: prt "number
 of weeks",W
16: gto 20
```

Note that the line number in the go to statement in line 16 is incremented since old line 19 is now line 20.

The branching address of the jump statement is not affected by adding or deleting lines in a program.

The Debugging Statements

The trace, stop, and normal statements are used for debugging programs. The three statements have dual roles in that their action depends upon whether any parameters are specified.

Operation of Trace, Stop, and Normal

To effectively use the trace, stop, and normal statements, the internal operation should be understood. There is one master flag which enables and disables overall tracing and stopping. In addition, each line has two flags. The trace flag enables and disables tracing of the line. The stop flag enables and disables selective stopping at a line. These flags are unrelated to flags 0 through 15, which are explained in Chapter 5.

The Trace Statement

Syntax:

trc [beginning line number [, ending line number]]

The trace (**trc**) statement sets the master trace flag. If line numbers are specified in the trace statement, then the individual line trace flags are set on the designated lines. One line number specifies that line only and two line numbers specify the block of lines from the beginning line number through the ending line number.

During the execution of the program, a specific line is traced if both the master trace flag and the individual line trace flags are set. When a line is traced, the number of the line is printed as well as information describing any variable assignments and flag operations (involving flags 0 through 15).

The Stop Statement

Syntax:

stp beginning line number [, ending line number]

The stop (**stp**) statement with line numbers sets the master trace flag and stop flags on the designated lines.

Before each program line is executed, the stop flag for that line is checked. If this flag and the master trace flag are set, the program is stopped before the line is executed. The number of the current program line is displayed when the program is stopped. Execution of the program will continue from this line if CONTINUE or STEP is pressed (see description of CONTINUE and STEP keys).

The Normal Statement

Syntax:

nor [beginning line number [, ending line number]]

The normal (**nor**) statement clears the trace and stop flags of the lines specified by the line numbers. If no line numbers are specified, the normal statement clears the master trace flag.

The use of a master trace flag in addition to individual line trace and stop flags makes it easy to enable or disable selective tracing or stopping of parts of a program. This process is shown in the following example.

A 100 line program contains three sections in which critical operations are performed. These sections can be traced by executing the following statements:

```
trc 5,15
trc 40,50
trc 70,85
```

The program is run and the tracing printout indicates that line 45 contains an error. The line is modified and nor is executed to clear the master trace flag. The program is again run, but this time the assignments are not printed. At the conclusion of the program it becomes obvious that the program still contains an error. The three critical sections of the program are again traced by executing trc. This sets the master trace flag so that the lines 5-15, 40-50, and 70-85 are traced (the trace bits are still set on these lines). After the program is totally debugged, the individual line trace flags are cleared by executing nor 0,9999.

The individual line trace and stop flags are not normally stored on the cartridge when a program is recorded by the record file statement. These flags can be recorded on the tape cartridge along with the program by including the optional debug ("DB") parameter in the record file statement. The master trace flag is not recorded. To have the program automatically trace the lines when the program is loaded back into the calculator, put trc in line 0 to set the master flag.

Chapter 7 Commands

There are five commands explained in this chapter. Commands differ from statements in that they can be executed only from the keyboard. Commands cannot be stored as part of a program.

The Run Command

Syntax:

`run` [line number or label]

The run command clears all variables, flags, and subroutine return pointers and then starts program execution. If a line number or label is specified, the program begins execution at the specified line number or label. Since (RUN) is an immediate execute key equivalent to `run 0` ↵, the word `run` must be keyed in to run from a line number or label.

Examples:

`run` ↵	Run beginning at line 0. This is the same as pressing (RUN).
`run 20` ↵	Run, beginning at line 20.
`run "third"` ↵	Run, beginning at the label "third".

The Continue Command

Syntax:

`cont` [line number or label]

The continue key (**cont**) command continues the program without altering variables, flags, or subroutine return pointers. If no line number is specified, then the program continues from the current position of the program line counter. When a line number or label is specified, the program continues at the specified line or label. If the program has been edited or an error has occurred since the program ran, continue without parameters causes execution to begin at line 0. Since (CONTINUE) is an immediate execute key equivalent to `cont 0` ↵, the word `cont` must be keyed in to continue at a line number or label.

Examples:

`cont` (EXECUTE)	Continue from current position of program line counter. This is the same as pressing (CONTINUE).
`cont 3` (EXECUTE)	Continue from line 3.
`cont "loop"` (EXECUTE)	Continue from the label "loop".

The Delete Line Command

Syntax:

`del` beginning line number [`,` ending line number] [`,*`]

The delete (**del**) command is used to delete lines or sections of programs. When one line number is specified , only that line is deleted. When two line numbers are specified, all lines in the block are deleted. To delete an entire program, and leave the variables, `del 0, 9999` can be executed.

Examples:

`del 28` (EXECUTE)	Delete line 28.
`del 13, 20` (EXECUTE)	Delete lines 13 through 20.
`del 18, 9999` (EXECUTE)	Delete program from line 18 to the end. (This does not affect variables.)

An attempt to delete lines that are destinations of relative or absolute go to or go sub statements (except labels) will cause error 36. To delete these lines, the delete command with the optional asterisk parameter can be used. When the asterisk is used, any go to or go sub statements which reference deleted lines are adjusted to reference the first line after the deleted section. For example to delete line 24 in this program segment:

```
22: ent U;if
 U=0;gto 24
23: U+T→T;C+1→C;
 gto 22
24: prt "Avg.
 Usage",T/C
25: prt "Total
 Usage",T
```

Type-in: `del 24,*`

Press: EXECUTE

Press: LIST 2 2 , 2 4 EXECUTE

```
22: ent U;if
 U=0;gto 24
23: U+T→T;C+1→C;
 gto 22
24: prt "Total
 Usage",T
```

The Erase Command

Syntax:

`erase` [`a` or `v` or `k` or special function key]

The erase command is used to erase programs, variables, and special function keys as shown below.

Command	Meaning
`erase`	Erases program and variables.
`erase a`	Erases everything (like switching the calculator off and then on again).
`erase v`	Erases all variables.
`erase k`	Erases all special function keys.
`erase` fn	Erases the indicated special function key.

See Appendix D for things affected by the erase command.

The Fetch Command

Syntax:

fetch [line number or special function key]

The fetch command brings individual program lines into the display. This is useful for editing lines or for viewing individual program lines. Fetching a special function key displays the definition of the key or f followed by the key number if the key is undefined. Executing fetch alone, fetches line 0.

Examples:

fetch 10	Fetch line 10.
fetch fn	Fetch special function key fn.

Chapter 8
Live Keyboard

The calculator's live keyboard mode provides additional power for executing single or multi-statement lines while a program is running. Among other things, you can perform math operations, monitor program activity, and alter program flow in live keyboard mode. Two statements described in this section permit the live keyboard mode to be turned on or off.

How Live Keyboard Works

While a program is running, a live keyboard operation is executed as follows:

- The live keyboard operation is keyed into the display and ↓ is pressed.
- At the end of the current program line, the live keyboard line is executed.
- The live keyboard operation is executed entirely before the program continues.

Live Keyboard Math

Any math operations can be executed from live keyboard. Thus, when a program is running and a few calculations need to be made, key in the operation and press ↓.

Statements in Live Keyboard

Math operations are just a small part of what can be done from live keyboard. If you want a listing of the current program, press LIST ↓.

To check a variable in the program, key in the variable name, such as A or B[4] and press ↓. The current value of the variable will be displayed.

To change a variable from live keyboard, enter the new value and assign it to the variable to be changed. For example to reset a counter such as C + 1 → C to 0, key in 0 → C and press ↓.

Parts of a program can be executed from live keyboard as subroutines using the g statement. For example, the following section of a running program is used to monito variables used in the program:

```
11: "check":prt
 A,B,C,D;ret
```

By executing gsb "check", the values of the variables are printed and control return program.

After a subroutine is finished, control returns to the main program when the return (ret) or (stp) statement is executed or when a stop flag at the beginning of a line is encountered.

Although the special function keys f0 through f23 cannot be defined from live keyboard can be used from live keyboard. In this example, the special function keys are used to alter flow of the running program.

The special function keys are defined as follows:

f0	f1
*1 → F	*2 → F

The program is:

```
0: "Wait":dsp
 "waiting";wait
 100;jmp F
1: gto "first"
2: gto "second"
3: gto "third"
4: "first":prt
 "first";0→F;
 gto "Wait"
5: "second":prt
 "second";0→F;
 gto "Wait"
6: "third":prt
 "third";0→F;
 gto "Wait"
```

When the program is run, waiting is displayed until one of the immediate execute (line preceded by *) special function keys is pressed. Then the program branches to the line where either first, second, or third is printed. Although this is a simple example, it shows one way that special function keys can be used in live keyboard mode.

The Stop Key in Live Keyboard

If STOP is pressed during a live keyboard operation, the live keyboard operation is stopped, but the program continues. Pressing STOP a second time will stop the program.

Live Keyboard Limitations

Operations that modify the stored program or special function keys and operations that directly affect the execution of the program are not allowed in live keyboard mode. These operations include the following:

Mnemonic	Error
Commands:	
run	error 03
cont	error 03
fetch	error 03
erase	error 03
del	error 03
Statements:	
ent	error 13
end	error 09
gto (allowed in a live keyboard subroutine)	error 09
ldp	error 64
ldk	error 64
ldf (program file)	error 64

In addition, the following keys cause a beep and are ignored when pressed in live keyboard mode.

STEP DELETE INSERT RUN STORE CONTINUE

The Display

Lines which are typed in live keyboard mode will disappear from the display if the running program uses the display. The live keyboard line is re-displayed after each keystroke so that the line with the new character added can be seen.

If the running program continually uses the display, the live keyboard lines will not be visible while the line is being typed. In this case, the line that is currently being typed, or the line accessed by [RECALL] can be held in the display by pressing [←] or [→]. These keys will suspend the running program for one second and display the line. If the key is kept depressed, the program will be halted for one second after it is released. After the line is executed, the [←] or [→] key will not re-display the line unless [RECALL] is pressed first. For example, suppose the following program is running in the calculator:

```
0: dsp "Live
 Keyboard";wait
 100
1: sto 0
```

When the following line is typed in live keyboard, it will not be visible:

```
prt √25 → A
```

Press [←] or [→] and the line will be displayed for about one second. When [EXECUTE] is pressed, the line will be executed and 5 will be stored in A and printed.

Results of calculations performed in live keyboard disappear from the display if a running program uses the display. The [←] or [→] keys only hold the live keyboard line in the display and not the result of the execution of a line. The result can be held in the display by appending a wait statement to the end of the line (e.g. `10 + 12; wait 1000`).

A special function key can be defined to preserve the displayed result long enough to be viewed as in this example:

Press: [FETCH] [f0]
Type in: `*; wait 1000`
Press: [STORE]

As you type in a calculation such as `5*6`, press [f0] instead of [EXECUTE]. The result of the calculation will remain in the display for about one second.

The Live Keyboard Enable Statement

Syntax:

```
lke
```

The live keyboard enable (**lke**) statement enables the live keyboard mode. For example:

`31: lke`	Enable live keyboard.

Live keyboard is automatically enabled when the calculator is turned on, `erase a` is executed, or RESET is pressed. To disable live keyboard, the live keyboard disable (lkd) statement is used.

The Live Keyboard Disable Statement

Syntax:

```
lkd
```

The live keyboard disable (**lkd**) statement disables live keyboard mode. For example:

`0: lkd`	The first line of this program disables live keyboard.

To re-enable live keyboard during a program it is necessary to execute the live keyboard enable (lke) statement from the program.

STOP, REWIND, and RESET are the only keys recognized while a program is running with live keyboard disabled.

During cartridge operations, the keyboard is disabled and all keys except RESET are ignored.

Chapter 9
Tape Cartridge Operations

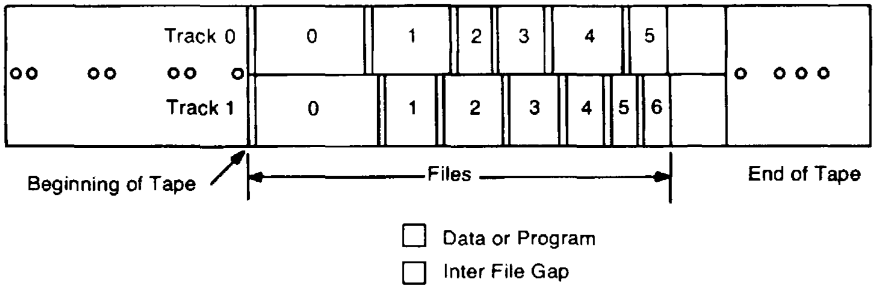
The tape cartridge used with the 9825A Calculator is a high quality, high density, digital storage medium. The structure, care, and use of the tape cartridge are detailed in this chapter.

Specifications

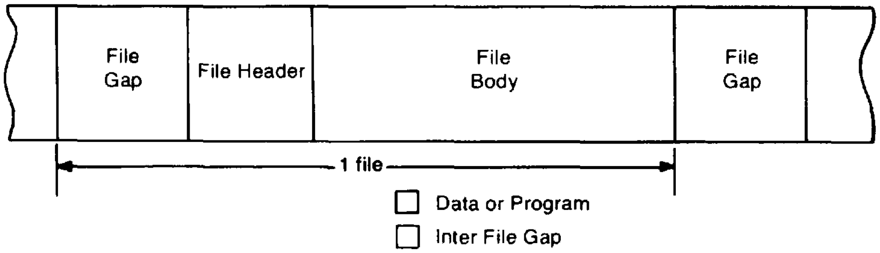
Typical data transfer rate (the rate at which information is loaded from or recorded on the tape cartridge)	2750 bytes per second
Typical access rate (the rate at which information passes over the tape head when searching for a file)	14300 bytes per second
Typical rewind time (from end to end)	19 seconds
Typical erase time (one entire track)	40 seconds
Usable tape length (typical)	42.67 meters (140 ft.)
Number of tracks	2

Tape Structure

The structure of the tape is diagrammed below:



An individual file has the following format:



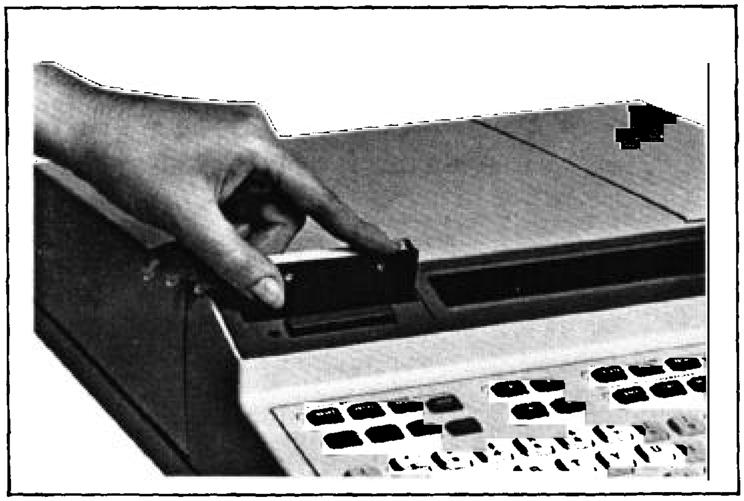
Tape Cartridge

The tape cartridge, shown below, is used to store programs, data, and the defined special function keys.

To record on the tape cartridge, the record slide tab must be in the rightmost position, that is, in the direction of the arrow (as shown).

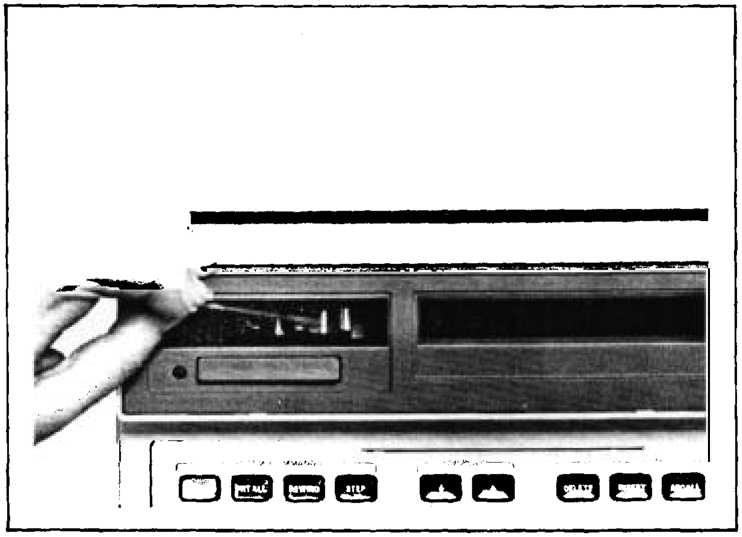
Inserting the Cartridge

Insert the tape cartridge so that the label on the cartridge faces the back of the calculator as shown.



Inserting the Tape Cartridge

Tape Care



Cleaning the Tape Head and Capstan

Dirt and dust are by far the greatest cause of cartridge related-errors. Several basic precautions can reduce such problems substantially.

- Clean the tape head and capstan (drive wheel) of the tape transport after at least every eight hours of use, or more frequently in dirty environments.
- Rewind the cartridge after each use.
- Keep the tape transport door clean.
- Keep the cartridge in the plastic container supplied with it.

Two other factors can affect the reliability of the tape cartridge. Strong magnetic fields can erase data and programs stored on the cartridge. Physical damage to the tape, such as wrinkled or folded tape can also cause record and load problems. A back-up copy should be maintained for critical programs or data on a separate tape cartridge.

Refer to Appendix F for information on tape error recovery.

The Rewind Statement

Syntax:

```
rew
```

The rewind (**rew**) statement is used to rewind the tape cartridge to its beginning. This statement has the same function as REWIND. Operations which do not use the tape cartridge can take place while the tape rewinds. To stop a tape while it is rewinding, press the tape cartridge ejection bar. The rewind statement must be executed before marking a new tape (see page 99).

The Track Statement

Syntax:

`trk` track number

The (**trk**) statement sets track 0 or track 1 of the tape cartridge. When the track statement is executed, any following cartridge operations are performed on that track. Track 0 is automatically set whenever the machine is switched on, RESET is pressed, or `erase a` is executed. The track does not change when the cartridge is removed nor when RUN is pressed.

The track number can be an expression with a value of 0 or 1, only.

CAUTION

THE TRACK IS AUTOMATICALLY SET TO 0 WHEN RESET IS PRESSED, `erase a` IS EXECUTED, OR WHEN THE CALCULATOR IS SWITCHED ON. UNLESS A SUBSEQUENT TRACK STATEMENT SPECIFIES TRACK 1, CARTRIDGE OPERATIONS WILL BE PERFORMED ON TRACK 0. IF YOU ARE UNAWARE OF THIS, YOU COULD LOSE IMPORTANT PROGRAMS OR DATA.

The Identify File Statement

Syntax:

`idf` [file number [, file type [, current file size [, absolute file size [, track number]]]]]

The identify file (**idf**) statement is used to load the contents of the current file header into the return variables specified. After the identify file statement is done, the tape is positioned in front of the file just identified. Thus, the tape is positioned for easy loading or recording of the identified file.

All five of the parameters are optional return variables. That means that a value is returned to the variable specified when the statement is executed. If one variable is specified, as in: `idf A`, then only the file number is returned. Two variables must be specified to get the file type; three variables to get the current file size in bytes; four variables to get the absolute file size in bytes; and five variables to get the track number. The return variables can be any variable type.

The file type can be one of the following:

0	null* file
1	binary program
2	numeric data
3	string or string and numerics (String Variables ROM required)
4	memory file (from record memory statement)
5	key file
6	program file

*A null file has an absolute size of zero.

The tape position becomes unknown when a tape cartridge is inserted into the tape drive, the track is changed, RESET is pressed, or `erase a` is executed. If the tape position is unknown such as after switching tracks, at least one return variable must be specified or error 45 will occur.

Example:

`idf A, B, C, D, E`	Identify the current file and return the file number, file type, current file size, absolute file size, and track number to A, B, C, D, and E, respectively.
`idf A, A, A`	Return the current file size to A.

The Find File Statement

Syntax:

`fdf` [file number]

The find file (**fdf**) statement is used to find the specified file on the current track of the tape cartridge. The tape is positioned at the beginning of the file specified. The file number can be an expression. A find file statement without parameters finds file 0. Other statements can be executed while the find file statement is executing.

> **NOTE**
>
> If a file number which does not exist is specified, the *next* cartridge statement executed (except find file or rewind) will result in error 65.

Examples of the find file statement:

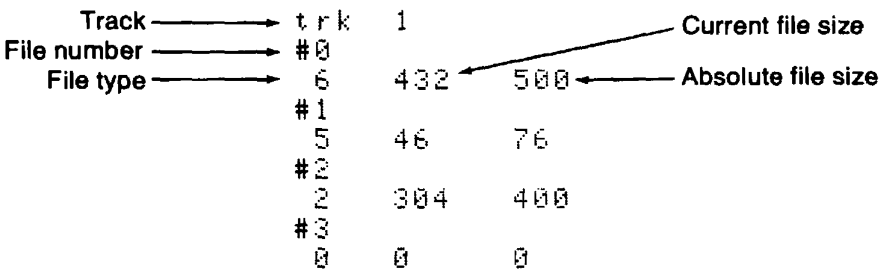
`fdf 8`	Find file 8.
`4: fdf A[3]`	Find the file specified by the value of A [3].

The Tape List Statement

Syntax:

```
tlist
```

The tape list (**tlist**) statement is used to identify the files on the tape cartridge. Starting from the tape's current position, the track, file number, file type, current file size in bytes, and absolute file size are printed as shown below.



```
trk  1
#0
  6  432   500
#1
  5  46    76
#2
  2  304   400
#3
  0  0     0
```

The file type can be one of the following:

0 null* file
1 binary program
2 numeric data
3 string or mixed string an numeric data (String Variables ROM must be present or loading this file will give error S0).
4 memory file (from record memory statement)
5 key file
6 program file

*A null file has an absolute size of zero.

If (STOP) is pressed while tlist is being executed, the tlist will terminate. Otherwise it will halt when the last file (null file) is reached.

A convenient way to determine the current track setting is to execute "tlist" then press. Alternately, use the identify file statement as in: `idf T, T, T, T, T; T(`

[illegible]

Syntax:

`mrk` number of files , file size in bytes [, return variable]

The mark (**mrk**) statement reserves file space on the tape cartridge. A file must be reserved before a program or data can be recorded. One file more than the number of files specified is marked. This file is the null file and is used as the starting point when marking more files. The null file has an absolute size of zero.

The file size is specified in bytes. If an odd number of bytes is specified, one more byte is automatically marked. For example, if 111 bytes are specified, 112 bytes are marked.

In order to mark files, the position of the tape must be known. If the position is unknown, execute a find file, or rewind statement to position the tape where you are going to start marking. Executing a mark statement where the first two parameters are zero (e.g., mrk 0,0) is a special case and is explained in Appendix F.

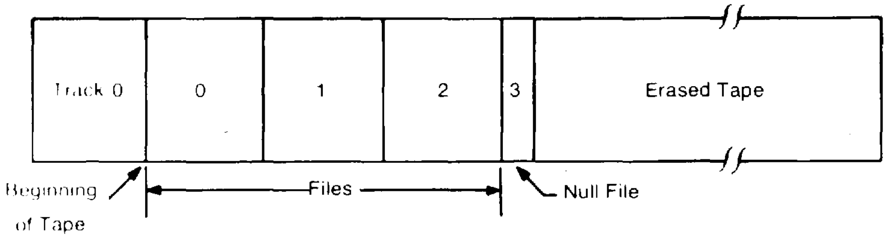
The number of files and the file size can both be expressions. If a return variable is specified, the file number of the last usable file marked is stored in it. If the value of the return variable is positive, all the files specified were marked. If the value is negative, an end-of-tape (eot) condition occurred before all the requested files were marked. In either case, the absolute value of the return parameter is the last usable file marked. The null file is one file beyond.

Example:

A tape is to be re-marked for 3 files with a length of 320 bytes each on track 0. The following short program performs this operation.

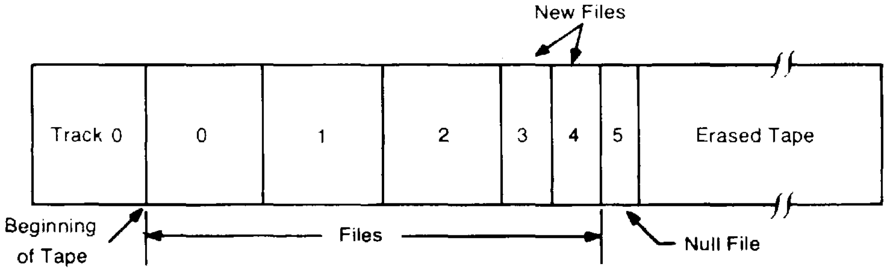
```
0: rew              Rewind the cartridge.
1: trk 0            Set to track 0.
2: mrk 3,320,X      Mark 3 files, 320 bytes long.
3: ert X+1          Erase the rest of track 0.
4: end              End the program.
```

The tape will be positioned at the beginning of file 3 and the resulting tape structure will be:



Then, 2 files with a length of 80 bytes are to be marked. Execute: `mrk 2,80`

New tape structure:



To mark 2 files, 300 bytes long beginning at file 4, execute: `fdf 4; mrk 2,300`

Determining Size to Mark a File

Program Files

When marking a file for a program which is currently in the calculator, execute `list -1`. The number in the left-hand portion of the display is exactly the number of bytes needed to record the program. It is advisable to mark the file larger to accommodate any future program changes.

Data Files

Data files require 8 bytes for each data element to be recorded. For example, to record data which is stored in the variables A and B, mark a file 16 bytes long.

.Special Function Key Files

Special function key files require 1 byte for each character under the keys, plus 2 bytes for each defined key. If the number of bytes for each key is odd, add one byte. The sum for all keys is the minimum size to mark the file.

Memory Files

For a memory file (using record memory statement), mark the file for the size of your calculator's memory as listed below:

8192	bytes for standard calculator
16384	bytes for Option 001
24576	bytes for Option 002
32766	bytes for Option 003

Tape Capacity

Table of Typical Storage Capacities

File size (bytes)	Typical number of files per track	Bytes per track
50	827	41350
100	656	65600
250	404	101000
500	239	119500
750	170	127500
1000	131	131000
2500	56	140000
5000	28	140000
7500	19	142500
10000	14	140000

Due to the overhead required by each file, the number of bytes per track is not the same for different size files.

Tape Capacity Calculations

The number of files which can be stored on the tape cartridge depends on the size of the file. Using the following calculations, the number of files that can be stored on the tape cartridge can be calculated.

$L = 1.278 + .209 \text{ int } (A/256 + .999) + .0105A$

where: A = absolute file size in bytes.
L = length of the file in inches.

a) For **typical** capacity per track:
Number of files per track = int (1665/L)

b) For **minimum** capacity per track:
Number of files per track = (1498/L)

The following program can be used to mark more files and calculate the percentage of a track used.

```
0: rew;fxd 0
1: ent "Track,
 0 or 1?",T;trk
 T
2: ent "New tape
 ? Yes=1 No=0",N
3: if N;gto "Mar
 k"
4: 0→F→L
5: fdf F;idf A,
 A,A,A
6: if A=0;gto
 "Mark"
7: gsb "L"
8: Q+L→L;F+1→F;
 gto 5
9: "Mark":prt
 "% of Tape"
10: prt "   Mark
 ed",L/1665*100→
 r0
11: if r0>=100;
 prt "All Marked
 ";gto "out"
12: ent "Mark
 more files?
 Yes=1 No=0",M
```

```
13: if M=0;gto
 "out"
14: ent "Length
 of files",A;
 gsb "L"
15: dsp "Number
 of files",A,
 "long?"
16: ent "",I;I*
 Q+L→r1
17: if (r1/1665→
 Z)>=1;prt "Too
 long...","That
 is %",Z*100;
 gto "out"
18: r1→L;mrk I,
 A,R;if R<0;dsp
 "100% Marked";
 gto "out"
19: gto "Mark"
20: "out":dsp
 "That's it";end
21: "L":1.278+
 .209int(A/256+
 .999)+.0105A→Q;
 ret
```

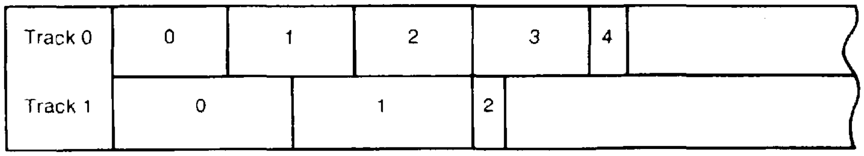
Marking New Tapes

Since no files are marked on a new tape, the rewind statement must be used to position the tape before marking files. For example, to mark 4 files 200 bytes long on a new tape, execute the following:

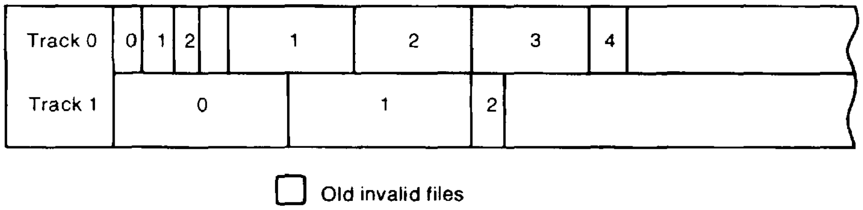
```
rew
mrk4, 200
```

Marking Used Tapes

When re-marking a used tape, it is possible that some old files may remain on the tape. These files can be accessed accidentally by changing tracks. For example, suppose track 0 has 4 files of 1000 bytes and track 1 has 2 files of 1500 bytes:

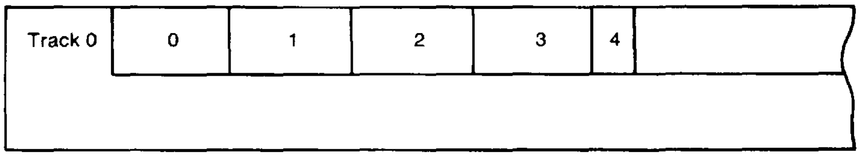


Then track 0 is re-marked from the beginning to contain 2 files of 200 bytes each:

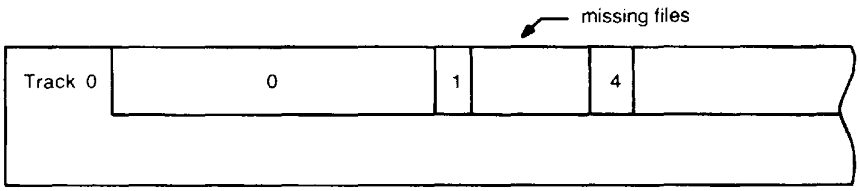


If the tape is positioned at file 1 on track 1, and `trk 0` is executed, the tape will be positioned in an old section of tape. Accessing file 1 on track 0 will result in using **old** file 1.

With slightly different conditions, it is possible to have missing files rather than multiple files. For example, suppose that track 0 has 4 files of 1000 bytes each:



If the tape is rewound and `mrk 1, 3000` is executed, the tape would have a gap of missing files:



To remove the old files, use the erase tape (ert) statement. For the first example:

`rew`	Rewind the tape on track 0.
`mrk 2, 200, A`	Mark 2 files, 200 bytes each.
`ert A + 1`	Erase the tape starting at the null file.

CAUTION

WHEN MARKING OVER A PREVIOUSLY MARKED TAPE, USE THE ERASE TAPE STATEMENT TO REMOVE OLD FILES.

The Erase Tape Statement

Syntax:

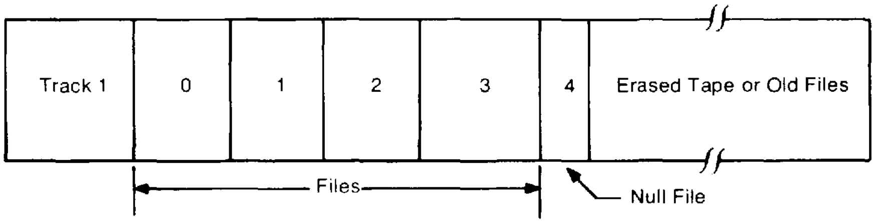
```
ert file number
```

The erase tape (**ert**) statement is used to erase everything on the current track starting from the file number specified. It is usually executed after a mark statement (see Marking Used Tapes on page 99). The erase tape statement:

1. Positions the tape in front of the file specified.
2. Marks that file as a null file.
3. Then, erases the track from the null file to the end of the track.
4. Finally the tape is positioned in the file gap in front of the null file.

The file number can be an expression.

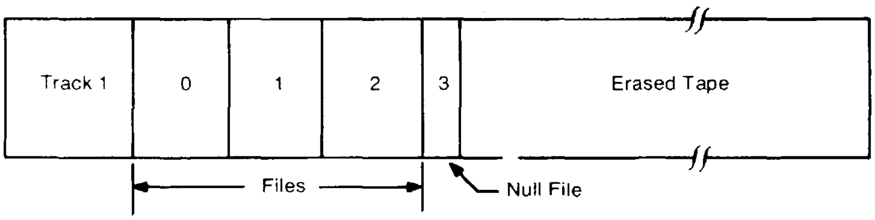
For example, a tape has the following structure on track 1:



To erase everything on track 1 starting at, and including file 3, the following program is used:

```
0: trk 1
1: ert 3
2: end
```

After running this program, the tape's structure is:



Track 0 is not altered.

The Record File Statement

The record file statement is used to store both data and programs. The syntax for each is explained below.

Recording Programs

Syntax:

rcf [file number [, beginning line number [, ending line number]] [, "SE" or "DB"]]

To record a program or a section of a program the record file (rcf) statement is used. If no file number is specified, the file is assumed to be file zero. If no line numbers are specified, the entire program is recorded on the specified file. If the beginning line number is specified, then the program from that line number to the end is recorded. If both line numbers are specified, that program section is recorded from the first line number to the second line number, inclusive.

The file number and ending line number parameters can both be constants, variables, or expressions. The beginning line number can be a constant, or expression (such as 1A), but must not be a variable. Using a variable as in rcf 1,A records the value of "A" as data. To record the program beginning at the line whose value is A, use rcf 1,1A.

If "SE" (for secure) follows at the end of the statement, the program is secured when stored on tape. When the secured program is loaded back into the calculator, the program cannot be listed or displayed, but can be re-recorded on a tape cartridge.

When "DB" (for debug) follows at the end of the statement, any trace or stop flags are recorded with the program (see page 77).

The tape file must be marked before recording a program. The file size must be greater than or equal to the size of the program being recorded.

Example:

`7: rcf 8,3`	Record the program on file 8, starting at line 3 through the end.

Recording Data

Syntax:

rcf file number , data list

The record file (rcf) statement is used to record data when this syntax is used. The data list can consist of simple variables, array variables, or r-variables. r-variables are stored in a different area in memory which is not contiguous with array or simple variables (see page 17). Due to this, r-variables cannot be mixed with simple or array variables in the record file statement.

To record an entire array, the array name is followed by an asterisk in brackets. For example:

`rcf 2, S[*]`	Record the entire array S on file 2.

Simple and array variables must appear in the data list in the same order as allocated. If the variables appear in a dimension statement, then they must appear in the same order in the record file statement.

Example:

`0: dim A[10,10]`	The array A is allocated 100 elements (800 bytes).
`1: 0→X`	The variable X is allocated 8 bytes.
`2: X+1→X`	Doesn't affect memory allocated to X.
`3: 1→I`	The variable I is allocated 8 bytes.

`4: rcf 5,A[*],X,`
`I`
	The array A, and variables X and I are recorded in the same order as allocated (contiguously) on file 5 (total of 102 numbers or 816 bytes).

If one r-variable is specified in the data list, all r-variables from r0 to that r-variable are recorded. If two r-variables are specified, all r-variables from the first through the second are recorded.

Considerations for Recording Data

When recording data on the tape cartridge, the variables being recorded must be listed in the same order as they are allocated in memory. For example:

```
0: ent A
1: 2*A→B
2: dim C,X,Y,Z
   •
   •
   •
15: rcf A,B,C,X,
 Y,Z
```

In the program, the variables A and B are allocated outside a dimension statement. Variables C, X, Y, and Z are allocated in a dimension statement. But, if B were allocated before A in the program, line 15 would cause error 56 to be displayed since the variables must be listed in the same order as they are allocated. Because lines are not necessarily executed in numerical order, it is sometimes difficult to know the order in which variables are allocated. For this reason, when a group of simple or array variables is to be recorded on a single file, it is recommended that they all be allocated in one dimension statement.

The Load Program Statement

Syntax:

ldp [file number [, line number$_1$ [, line number$_2$]]]

The load program (**ldp**) statement is used to load a program from the specified file on the current track and **run** it automatically. The automatic run implies that all variables are erased, all subroutine return pointers are cleared, and all flags (0 through 15) are cleared.

When the file number only is given, the program is loaded from the file, beginning at line zero, and the program automatically **runs** from line zero. If the file number and the first line number are specified, the program is loaded from that file, beginning at the specified line number and **runs** from that line number. When all three parameters are specified, the program is loaded from the specified file number beginning at the first specified line number and is **run** beginning at the second specified line number. If no parameters are specified, zeros are assumed for all three. All three parameters can be expressions.

If a program is loaded at the end of an existing program, go to and go sub branching line numbers are not renumbered.

The load program statement can only be stored as the last statement in a line. This statement is not allowed in live keyboard mode nor during an enter statement.

Examples:

`ldp 2`	Loads the program from file 2 beginning at line 0 and runs from line 0.
`ldp 8,2`	Loads the program from file 8 beginning at line 2 and runs from line 2.
`ldp 16,3,0`	Loads the program from file 16 beginning at line 3 and runs from line 0.

The Load File Statement

The load file (**ldf**) statement is used to load both data and program files into the calculator memory.

CAUTION

THE LDF STATEMENT LOADS THE PROGRAM OR VARIABLE AREA OF MEMORY DEPENDING ON THE FILE TYPE ACCESSED. BUT THE LDP AND RCF STATEMENTS LOAD OR RECORD A SPECIFIED PART OF MEMORY DEPENDING ON THE STATEMENT. THUS, WITH THE LDF STATEMENT IT IS POSSIBLE TO ACCIDENTALLY LOAD A PROGRAM WHEN THE INTENT WAS TO LOAD VARIABLES OR VICE VERSA.

Loading Programs

Syntax:

ldf [file number [, line number$_1$ [, line number$_2$]]]

The load file (**ldf**) statement loads programs from the specified file on the current track into the calculator memory.

This statement is like the load program (ldp) statement except that ldf can be used to continue a program, while the ldp statement causes the program to run.

From the Keyboard

This statement is executed from the keyboard as follows: When no parameters are given, the program on file zero is loaded, beginning at line 0. If the file number is given, that file is loaded beginning at line 0. If the file number and a line number are specified, then that file is loaded beginning at the specified line number. When all three parameters are given, the specified file is loaded beginning at the first line number, and the program automatically **continues** at the second line number (all variables are preserved whereas ldp destroys the old variables; see continue command page 79).

If a program is loaded at the end of an existing program, go to and go sub branching line numbers are not renumbered.

In a Program

The ldf statement is executed in a program as follows: When no parameters are specified, the program on file zero is loaded beginning at line zero and continues at line zero. If the file number is specified, then the program is loaded from the specified file beginning at line zero and continues at line zero. When the file number and a line number are given, the specified file is loaded beginning at the specified line number and the program continues from that line number. When all three parameters are given, the statement is executed the same as from the keyboard. That is, a "continue" is performed from the second line number. All three parameters can be expressions.

This statement is not allowed in live keyboard mode nor during an enter statement to load a program file. However, the ldf statement can be used to load a data file in live keyboard.

Example:

```
ldf 1
```

Load file 1 beginning at line 0 (executed from keyboard).

```
13: ldf 2
```

Load file 2 beginning at line 0 and **continue** from line 0.

Linking Programs

Programs too long to store in the calculator memory can be segmented and stored in separate files on the tape cartridge. Each segment can be loaded as needed by the program, and, using the ldf statement, variables, flags, and subroutine return pointers can be preserved for each segment.

In the following basic example, three segments are used. Each segment is loaded as it is needed by the program. The first segment loads the second and the second loads the third.

Program Segment on file 0 ⧫

```
0: prt "file 0";
 π→A
1: ldf 1
```

Program Segment on file 1 ⧫

```
0: prt "file 1";
 prt A
1: ldf 2
```

Program Segment on file 2 ⧫

```
0: prt "file 2"
1: end
```

Press: LOAD 0 EXECUTE RUN

```
file 0
file 1
              3.1416
file 2
```

Loading Data

Syntax:

```
ldf [file number [, data list]]
```

The load file (ldf) statement loads data from the specified file on the current track. The data list contains the names of variables separated by commas. r-variables cannot be in the same load file statement with simple and array variables.

If no list is specified, data begins filling the r-variables from r0 until all the data has been loaded. If one r-variable is specified, then the data begins filling r-variables from that r-variable until all the data has been loaded into higher r-variables. If two r-variables are specified, the data starts filling from the first location specified (lower r-variable) to the second, higher, r-variable. If there is more data than available or specified r-variables, no data is loaded.

When simple or array variables are specified, data begins filling the first variable until all variables have assigned values. If there is more data than variables, no data is loaded. If there is less data than variables, the data is loaded until all data is used. Variables must be contiguous.

Examples:

`ldf 4, r2, r10`	Load r2 through r10 from data file 4.
`ldf r12, A, B[*]`	Load the data file designated by r12 into the variable A and array B.

Array and r-variable Recording

Array variables are recorded in the opposite order of r-variables. Thus, if r-variables are recorded, then loaded back into an array, they will be in the opposite order. For example:

```
0: 1→r1;2→r2;
 3→r3;4→r4;5→r5
1: rcf 0,r5
2: dim A[6]
3: ldf 0,A[*]
4: 1→I
5: prt A[I];jmp
 (I+1→I)>6
6: end
```

r5– A[1]	5.0000
r4– A[2]	4.0000
r3– A[3]	3.0000
r2– A[4]	2.0000
r1– A[5]	1.0000
r0– A[6]	0.0000

In line 1, r-variables 0 through 5 are recorded on file 0. Then in line 3, the array A is loaded from file 0. A[6] is loaded first; A[1] is loaded last.

The Record Keys Statement

Syntax:

`rck` [file number]

The record keys (**rck**) statement is used to record all the special function keys on the specified file on the current track. If the file number is omitted, file zero is assumed. The file number can be an expression. The specified file must be marked before the record keys statement is executed.

Examples:

`rck 2`	Record the special function keys on file 2.
`3: rck A[12]`	Record the special function keys on the file designated by the 12th element of array A.

The Load Keys Statement

Syntax:

`ldk` [file number]

The load keys (**ldk**) statement is used to load the special function keys exactly as they were recorded from the specified file on the current track. If the file number is omitted, file zero is assumed. The file number can be an expression. Executing the load keys statement from the keyboard causes subroutine return pointers to be reset and causes the program counter to reset to line zero.

This statement is not allowed in live keyboard nor during an enter statement.

Example:

`ldk 4`	Load the special function keys from file 4.

The Record Memory Statement

Syntax:

`rcm` [file number]

The record memory (**rcm**) statement records the entire read-write memory (program, data, keys, pointers, etc.) on the specified file on the current track of the tape cartridge.

number is omitted, file 0 is assumed. The file number can be an expression.

ence of a binary program in memory is a special case. The information supplied with nary program explains the calculator operation when the record memory statement is ed.

[file number]

d memory (**ldm**) statement is used to load a previously recorded memory file. When the eration is complete, the calculator is in the same state it was in when memory was ed. If the file number is omitted, file 0 is loaded.

gram was running when the record memory statement was executed, that program will e with the next statement after the record memory statement when the load memory ent is executed.

ord memory and load memory statements can be executed from live keyboard or from al function key. The record memory statement can be used to "freeze" the state of the n without interrupting the running program. These statements can be especially useful in where frequent power interruptions occur.

file number can be an expression.

[file number]

ad binary program (**ldb**) statement loads binary programs* into the calculator's read/ memory from the specified file on the current track of the tape cartridge. Binary prog- can be loaded over other binary programs of equal or greater length at any time.

file number is specified, file 0 is assumed. The file number can be an expression.

mple:

Load the binary program from file 2.

* program is a machine language program which cannot be listed or displayed.

Since binary programs occupy a special place in memory, certain rules must be followed when loading them:

1. Any binary program can be loaded at any time (from the keyboard or a running program) provided there is room in memory for it and no simple or array variables are allocated.
2. Once simple or array variables are allocated, a binary program cannot be loaded unless space has been allocated for it by a previous binary program load operation.

The following procedure is suggested: Before any simple or array variables are referenced, load the largest binary program file that will be needed. Then any variables can be allocated and other binary programs can be loaded without concern about room for the binary program.

File Verification

File verification is used to compare a tape file against the calculator memory to detect recording errors without losing the information in memory. If you get a verify error (error 44), try re-recording the file. Repeated verify errors on a file may indicate damaged tape.

File verification requires a stronger tape signal than load; thus, it increases confidence that a file will load properly at a later time.

When the calculator is turned on, `erase a` is executed, or RESET is pressed, the calculator automatically verifies files on all record operations. Two statements are used to control automatic verification.

The Auto-Verify Disable Statement

Syntax:

```
avd
```

The auto-verify disable (avd) statement turns off file verification. For example:

```
12: avd                                Turn-off automatic file verification.
```

The Auto-Verify Enable Statement

Syntax:

The auto verify enable (**ave**) statement turns on automatic file verification. After ave is executed, all record operations are automatically followed by a verify. When the calculator is turned on, [illegible] is pressed, or erase a is executed, automatic file verification is again enabled.

Example:

Turn on Automatic file verification.

The Verify Statement

Syntax:

[return variable]

The verify (**vfy**) statement is used to compare a tape file with the calculator memory. If the calculator memory is identical to the tape file, the value of the return variable is 0 after the operation. If the two are different, the return variable is one. If no return variable is specified and if the memory and tape file are not identical, error 44 occurs. The return variable can be either a simple variable, array variable, or r-variable.

The verify statement can follow any record operation except the record memory (rcm) statement. The record memory statement followed by vfy will result in error 44. Memory files can only be verified using automatic verification.

With the verify statement, you can selectively verify files. This can be useful to save time when recording many files. Another important use is recovery from verify errors using the return variable parameter.

This statement does not alter the calculator memory.

The Set Select Code Statement

The set select code (**ssc**) statement is provided for possible future access to peripherals and should not be used at the present time. Select code 1 is used by the internal cartridge.

Appendix A
Syntax

Two sets of syntax are given in this appendix. The first set is the informal syntax used throughout the manual. The second is a language description syntax of HPL.

Informal Syntax

Syntax	Name	Page
abs expression	Absolute value function	53
acs expression	Arccosine function	57
[expression] + expression	Addition operator (or unary plus)	50
expression and expression	AND operator	52
asn expression	Arcsine function	56
expression → variable	Assignment operator	50
atn expression	Arctangent function	57
avd	Auto-verify disable statement	110
ave	Auto-verify enable statement	111
beep	Beep statement	47
cfg [flag number , ...]	Clear flag statement	60
cmf [flag number , ...]	Complement flag statement	60
cont [line number or label]	Continue command	79
cos expression	Cosine function	56
csv	Clear simple variables statement	70
deg	Degrees statement	56
del beginning line number [, ending line number][, *]	Delete command	80
dim item[, item , ...] where item is: simple variable array variable [dimension [, dimension , ...]]	Dimension statement	68
expression / expression	Division operator	50
drnd (expression , number of digits)	Digit rounding function	54

Syntax	Name	Page
dsp [any combination of text or expressions]	Display statement	[illegible]
end	End statement	4
enp [prompt ,] variable [, [prompt ,] variable...]	Enter print statement	[illegible]
ent [prompt ,] variable[, [prompt ,] variable...]	Enter statement	[illegible]
expression = expression	Equal to operator	51
erase [a or v or k or special function key]	Erase command	83
ert file number	Erase tape statement	101
expression xor expression	Exclusive OR operator	53
exp expression	Exponential function	57
expression ↑ expression	Exponentiation operator	50
fdf [file number]	Find file statement	94
fetch [line number or special function key]	Fetch command	83
flg flag number	Flag function	61
flt [number of decimal places]	Float statement	41
frc expression	Fraction function	54
fxd [number of decimal places]	Fixed statement	40
grad	Grads statement	56
expression > expression	Greater than operator	51
expression >= expression	Greater than or equal to operator	51
expression => expression	Greater than or equal to operator	51
gsb line number	Absolute go sub statement	65
gsb label	Labelled go sub statement	66
gsb + number of lines	Relative go sub statement	65
gsb − number of lines	Relative go sub statement	65
gto line number	Absolute go to statement	63
gto label	Labelled go to statement	63
gto + number of lines	Relative go to statement	63
gto − number of lines	Relative go to statement	63
idf [file number [, file type [, current file size [, absolute file size [, track number]]]]]	Identify file statement	93
if expression	If statement	67
int expression	Integer function	54
jmp number of lines	Jump statement	64
ldb [file number]	Load binary program statement	109

Syntax	Name	Page
ldf [file number [, data list]]	Load file statement (Data file)	107
ldf [file number [, line number$_1$ [, line number$_2$]]]	Load file statement (Program file)	105
ldk [file number]	Load keys statement	108
ldm [file number]	Load memory statement	109
ldp [file number [, line number$_1$ [, line number$_2$]]]	Load program statement	104
expression < expression	Less than operator	51
expression <= expression	Less than or equal to operator	51
expression =< expression	Less than or equal to operator	51
list k	List keys statement	70
list special function key	List special function key command	70
list [beginning line number [, ending line number]]	List statement	70
lkd	Live keyboard disable statement	87
lke	Live keyboard enable statement	87
ln expression	Natural logarithm function	55
log expression	Common logarithm function	55
max (list of expressions and arrays)	Maximum value function	54
min (list of expressions and arrays)	Minimum value function	54
expression mod expression	Modulus operator	50
mrk number of files , file size [, return variable]	Mark statement	96
expression * expression	Multiplication operator (explicit)	50
(expression) (expression)	Multiplication operator (implicit; no mnenomic)	51
nor [beginning line number [, ending line number]]	Normal statement	76
expression # expression	Not equal to operator	51
expression <> expression	Not equal to operator	51
expression >< expression	Not equal to operator	51
not expression	NOT operator	53
expression or expression	OR operator	52
prnd (expression , rounding specification)	Power-of-ten rounding function	54
prt [any combination of text or expressions]	Print statement	43
rad	Radians statement	56
rcf file number , data list	Record file statement (Data file)	102
rcf [file number [, beginning line number [, ending line number]] [, "SE" or "DB"]]	Record file statement (Program file)	102

Syntax	Name	Page
rck [file number]	Record keys statement	108
rcm [file number]	Record memory statement	108
ret	Return statement	65
rew	Rewind statement	92
rnd [-] expression	Random number function	54
run [line number or label]	Run command	79
sfg [flag number , ...]	Set flag statement	59
sgn expression	Sign function	54
sin expression	Sine function	56
spc [number of blank lines]	Space statement	47
√ expression	Square root function	53
ssc select code	Set select code statement	111
stp	Stop statement	48
stp beginning line number [, ending line number]	Stop statement (for debugging)	76
[expression] - expression	Subtraction operator (or unary minus)	50
tan expression	Tangent function	56
tlist	Tape list statement	95
tn↑	Ten-to-the-power function	55
trc [beginning line number [, ending line number]]	Trace statement	76
trk track number	Track statement	92
units	Units statement	56
vfy [return variable]	Verify statement	111
wait number of milliseconds	Wait statement	47

LEGEND

: := "is defined as..."

⟨ ⟩ enclose an element of HPL

```
⟨Line⟩              : :=  ⟨Statements⟩
                          Label : ⟨Statements⟩

⟨Statements⟩        : :=  End-of-line
                          ⟨First⟩ End-of-line
                          ⟨Second⟩ End-of-line

⟨First⟩             : :=  ⟨Type 1⟩
                          ⟨First⟩ ; ⟨Type 1⟩

⟨Second⟩            : :=  ⟨Type 2⟩
                          ⟨First⟩ ; ⟨Type 2⟩

⟨Type 1⟩            : :=  ⟨Assignment⟩
                          ⟨Group 1⟩
                          ⟨Group 2⟩ ⟨Value⟩
                          ⟨Group 3⟩
                          ⟨Group 3⟩ ⟨Value⟩
                          ⟨Group 4⟩
                          ⟨Group 4⟩ ⟨Expression list⟩
                          ⟨Group 5⟩
                          ⟨Group 5⟩ ⟨I/O list⟩
                          ⟨Group 6⟩
                          ⟨Group 6⟩ ⟨Value⟩
                          ⟨Group 6⟩ ⟨Value⟩ , ⟨Value⟩
                          ⟨Group 7⟩
                          ⟨Group 7⟩ ⟨Cartridge I/O list⟩
                          ⟨Group 8⟩ Integer
                          ⟨Group 8⟩ Label
                          ⟨Group 9⟩ ⟨ent list⟩
                          dim ⟨dim list⟩
                          mrk ⟨mrk list⟩
                          vfy
                          vfy ⟨Variable⟩
                          idf
                          idf ⟨Variable list⟩
```

```
⟨Group 1⟩          ::= avd
                       ave
                       beep
                       csv
                       deg
                       end
                       grad
                       listk
                       lkd
                       lke
                       rad
                       rew
                       tlist
                       units
⟨Group 2⟩          ::= ert
                       if
                       ssc
                       trk
                       wait
⟨Group 3⟩          ::= fdf
                       flt
                       fxd
                       ldb
                       ldk
                       ldm
                       rck
                       rcm
                       spc
⟨Group 4⟩          ::= cfg
                       cmf
                       sfg
⟨Group 5⟩          ::= dsp
                       prt
⟨Group 6⟩          ::= nor
                       stp
                       trc
⟨Group 7⟩          ::= ldf
                       rcf
```

```
⟨Group 8⟩            ::=  gsb
                          gsb +
                          gsb −
                          gto
                          gto +
                          gto −

⟨Group 9⟩            ::=  enp
                          ent

⟨Type 2⟩             ::=  jmp ⟨Value⟩
                          ret
                          ⟨Group 10⟩
                          ⟨Group 10⟩ ⟨Expression list⟩

⟨Group 10⟩           ::=  ldp
                          list

⟨Value⟩              ::=  ⟨Expression⟩
                          ⟨Assignment⟩

⟨Assignment⟩         ::=  ⟨Expression⟩ → ⟨Variable⟩
                          ⟨Assignment⟩ → ⟨Variable⟩
⟨Expression⟩         ::=  ⟨Conjunction⟩
                          ⟨Expression⟩ or ⟨Conjunction⟩
                          ⟨Expression⟩ xor ⟨Conjunction⟩
⟨Conjunction⟩        ::=  ⟨Denial⟩
                          ⟨Conjunction⟩ and ⟨Denial⟩

⟨Denial⟩             ::=  ⟨Relation⟩
                          not ⟨Relation⟩

⟨Relation⟩           ::=  ⟨Sum⟩ ⟨Relational operator⟩ ⟨Sum⟩
                     ::=  ⟨Sum⟩
⟨Relational Op.⟩     ::=  #
                          < >
                          > <
                          < =
                          = <
                          > =
                          = >
                          =
                          <
                          >
```

```
⟨Sum⟩              ::= ⟨Term⟩
                       ⟨Sum⟩ + ⟨Term⟩
                       ⟨Sum⟩ − ⟨Term⟩
⟨Term⟩             ::= ⟨Negation⟩
                       ⟨Term⟩ * ⟨Negation⟩
                       ⟨Term⟩ / ⟨Negation⟩
                       ⟨Term⟩ mod ⟨Negation⟩
⟨Negation⟩         ::= ⟨Factor⟩
                       + ⟨Factor⟩
                       − ⟨Factor⟩
⟨Factor⟩           ::= ⟨Exponentation⟩
                       ⟨Factor⟩ I* ⟨Exponentiation⟩(I* means Implied multiply)

⟨Exponentiation⟩   ::= ⟨Primary⟩
                       ⟨Exponentiation⟩ ↑ ⟨Primary⟩

⟨Primary⟩          ::= ⟨Variable⟩
                       number
                       π
                       res
                       ( ⟨Value⟩ )
                       ⟨Function⟩ ⟨Primary⟩
                       drnd ( ⟨Value⟩ , ⟨Value⟩ )
                       prnd ( ⟨Value⟩ , ⟨Value⟩ )
                       min ( ⟨Cartridge I/O list⟩ )
                       max ( ⟨Cartridge I/O list⟩ )
⟨Variable⟩         ::= Simple variable
                       Array variable [ ⟨Expression list⟩ ]
                       r ⟨Primary⟩
⟨Function⟩             abs
                   ::= acs
                       asn
                       atn
                       cos
                       exp
                       frc
                       flg
                       int
                       ln
                       rnd
                       sgn
                       log
                       sin
                       √‾
                       tan
                       tn↑
```

```
⟨Variable list⟩          : := ⟨Variable⟩
                              ⟨Variable list⟩ , ⟨Variable⟩

⟨Expression list⟩        : := ⟨Value⟩
                              ⟨Expression list⟩ , ⟨Value⟩

⟨dim list⟩               : := ⟨dim item⟩
                              ⟨dim list⟩ , ⟨dim item⟩

⟨dim item⟩               : := Simple variable
                              Array variable [ ⟨Bounds list⟩ ]

⟨Bounds list⟩            : := ⟨Bounds⟩
                              ⟨Bounds list⟩ , ⟨Bounds⟩

⟨Bounds⟩                 : := ⟨Value⟩
                              ⟨Value⟩ : ⟨Value⟩

⟨ent list⟩               : := ⟨Variable⟩
                              ⟨ent list⟩ , ⟨Variable⟩
                              Prompt , ⟨Variable⟩
                              ⟨ent list⟩ , Prompt , ⟨Variable⟩

⟨mrk list⟩               : := ⟨Value⟩ , ⟨Value⟩
                              ⟨Value⟩ , ⟨Value⟩ , ⟨Variable⟩

⟨Cartridge I/O list⟩     : := ⟨Value⟩
                              ⟨Cartridge I/O list⟩ , ⟨Value⟩
                              ⟨Cartridge I/O list⟩ , Array Variable [*]

⟨I/O list⟩               : := ⟨I/O item⟩
                              ⟨I/O list⟩ , ⟨I/O item⟩

⟨I/O item⟩               : := ⟨Value⟩
                              "Text"
```

Appendix B
Error Messages

An error in a program sets the program line counter to line 0, but does not affect variables, or subroutine return pointers. Pressing CONTINUE will continue the program from line 0. Execute the continue command with a line number to continue at any desired line (such as: `cont 50`).

Mainframe Error Messages

`error 00` System Error

- Memory error occurred.
- Operating system error.
- Component failure; service required. Contact your nearest HP sales and service office listed in the back of this manual.

`error 01` Unexpected peripheral interrupt. Occurs only when a peripheral is being used. Press RESET to recover.

`error 02` Unterminated text. The line of text must have an ending quote.

`error 03` Mnemonic is unknown. This error is usually caused by typing errors, such as `go to` instead of `gto`; or by attempting to execute a command in live-keyboard mode or during an enter statement.[2]

`error 04` System is secured. This error is generally caused by trying to list or fetch lines in a secured program.

`error 05` Operation not allowed – line cannot be stored or executed with line number. This can be caused by pressing EXECUTE, STORE, or line INSERT with a fetched line in the display.

`error 06` Syntax error in number. For example, executing:

`1 e 104`

`error 07` Syntax error in input line. For example:

`gto prt 5`

[2] See 9885 Disk ROM error messages also.

[*] These errors give a cursor when the recall key is pressed, showing location of error.

error 08 Internal representation of the line is too long (gives cursor sometimes).

error 09 The gto, gsb, or end statement is not allowed in the present context. For example, attempting to execute an end statement during an enter statement. [1]

error 10* The gto or gsb statement requires an integer. For example:
gto 23.4 is not allowed

error 11 Integer out of range or integer required. Must be between −32768 and +32767. For example:
spc 50000 Integer out of range.
del A Integer constant required.

error 12 The line cannot be stored. It can only be executed. Expressions such as $\sqrt{32}$ or flg A or 2*B must be part of a statement to be stored. $\sqrt{32}$→A or prt flg A or jmp 2*B are statements and can be stored. For example:
2 + 2 This is acceptable.
2 + 2 STORE This is not allowed (can't store expression).

error 13 Enter (ent) statement is not allowed in present context. For example, ent X is not allowed from the keyboard, only from a program.

error 14 Program structure destroyed. This can be caused by pressing RESET while a program is being modified or loaded. It is advisable to record data then execute erase a to recover.

error 15 Printer out of paper or printer failure.

error 16 The String Variables ROM is not present for the string comparison, or the argument in a relational comparison is not allowed. For example, if the String Variables ROM is not in the calculator: if "B"<"A" results in error 16.

error 17 Parameter is out of range. For example, the following are not allowed.
wait -5
fxd 15

error 18 Incorrect parameter. For example:
erase z

error 19 Bad line number. For example:
del 10,5

[1] See Advanced Programming ROM error messages also.

*These errors give a cursor when the recall key is pressed, showing the location of error.

error 20 A ROM or binary program is missing. As a result, the line cannot be reconstructed. This error usually occurs when fetch or list is executed or when ↑ or ↓ is pressed. The number to the right of the error number in the display indicates the missing ROM. In the program mode, the ROM number is not displayed, but replaced by line number.

ROM Number in Display	ROM
1	Binary Program
6	String Variables
8	Extended I/O
9	Advanced Programming
10	Matrix
11	9862A Plotter
12	General I/O
15	9885M Disk ROM

error 21 Line is too long to store. This can occur when blanks or parentheses are automatically added. For example, parentheses are automatically added when storing the line: `tan 2→A` which will appear in a listing as: `tan (2)→A`.

error 22 Improper dimension specification. For example, this error occurs when the lower bound of a subscript is greater than the upper bound. Also, if the String Variables ROM is not in the calculator and a string is dimensioned this error results.

error 23 The simple variable has already been allocated. For example:

`2→X; dim A[5],X`

error 24 The array has already been dimensioned. For example:

`dim A[4],B[5],A[6]`

error 25 Dimensions of array disagree with subscripts. For example:

`dim X[2,7]; 1→X[5]`

error 26 Subscript of array element is out of bounds.[1] For example:

`dim A[12]; 2→A[58]`

error 27 Undefined array. All arrays must appear in a dimension (dim) statement before being used elsewhere.

error 28 The return (ret) statement has no matching gsb statement.

error 29 Cannot execute the line because a ROM or binary program is missing. For example, executing the plt statement without a Plotter ROM.[2]

[1] See Advanced Programming ROM error messages also.

[2] [illegible] messages also

error 30 Special function key has not been defined.[2]

error 31 Non-existent program line. For example, gto 9 in a 5 line program.

error 32 Improper data type. A number is required.[1]

error 33 Data types don't match in an assignment statement.

error 34 Display overflow due to pressing a special function key. Only 80 characters can be entered into the display.

error 35 Improper flag referenced (there is no such flag). For example: sfg 18

error 36 Attempt to delete the destination of a gto or gsb statement. Operation not performed.

error 37 Display buffer overflow caused by display (dsp) statement.

error 38 Insufficient memory for subroutine return pointer.[1]

error 39 Insufficient memory for variable allocation or binary program. No allocation takes place.

error 40 Insufficient memory for operation. For example, attempting to store a line with insufficient memory available.[1]

error 41 No cartridge in the tape transport.

error 42 Tape cartridge is write protected. Slide the record tab to the other position for recording, marking, or erasing.

error 43 Unexpected Beginning-of-Tape (BOT) or End-of-Tape (EOT) marker encountered; or a tape transport failure.

error 44 Verify has failed (see File Verification page 110).

error 45 Attempted execution of idf statement without parameters when tape position is unknown, or attempted execution of mrk statement when the tape position is unknown.

error 46 Read error of file body. The partition containing the error is lost (see Appendix F).

error 47 Read error of file head (see Appendix F).

error 48 The end of tape was encountered before the specified number of files were marked.

error 49 File is too small.

[1] See Advanced Programming ROM error messages also.
[2] See 9885 Disk ROM error messages also.

error 50 The ldf statement for a program file must be the last statement in the line.[2]

error 51 A ROM or binary program is present but was not when the memory was recorded. Remove the ROM indicated by the number to the right of the error number in the display, and re-execute the load memory (ldm) statement. When in the program mode, the line number is given instead of the ROM number.[2]

ROM Number	
in Display	ROM
1	Binary Program
6	String Variables
8	Extended I/O
9	Advanced Programming
10	Matrix
11	9862A Plotter
12	General I/O
15	9885M Disk ROM

error 52 The ROM or binary program indicated by the number to the right of the error number was present when the memory was recorded but is now missing. Insert the indicated ROM and re-execute the load memory (ldm) statement.[2]

ROM Number	
in Display	ROM
1	Binary Program
6	String Variables
8	Extended I/O
9	Advanced Programming
10	Matrix
11	9862A Plotter
12	General I/O
15	9885M Disk ROM

error 53 Negative parameter in cartridge statement. For example:

```
mrk -12,300
trk -1
```

error 54 Binary program to be loaded is larger than present binary program and variables have been allocated.

error 55 Illegal or missing parameter in one of the cartridge statements.

error 56 Data list is not contiguous in memory for one of the cartridge statements.

[2] See 9885 Disk ROM error messages also.

error 57 Improper file type. For instance, this can occur when trying to load a program from a data file or key file.

error 58 Invalid parameter in rcf statement; "SE" or "DB" expected.

error 59 Attempt to record a program, or special function keys which do not exist.

error 60 Attempt to load an empty file or the null file (type = 0).

error 61 The line referenced in an ldf or ldp statement does not exist. If the line containing the ldf or ldp statement has been overlaid by the load operation, the line number in the display may be incorrect.

error 62 Specified memory space is smaller than cartridge file size.

error 63 Cartridge load operation would overlay subroutine return address in program; load not executed.[2]

error 64 Attempt to execute ldk, ldf (program file), or ldp during live keyboard or enter statement.[2]

error 65 File not found, or the file specified in the previous find file (fdf) statement does not exist.

For errors 66 through 77, the default value is used and no error is displayed if you set flag 14.

error 66 Division by zero. The default value is + or − 9.99999999999e 511. A mod B with B equal to zero. The default value is 0.

error 67 Square root of a negative number. The default value is $\sqrt{\text{(abs (argument))}}$.

error 68 Tan (n*π/2 radians);
Tan (n*90 degrees);
Tan (n*100 grads);
where n is an odd integer.
The default value is +9.99999999999e 511, for n > 0.
The default value is −9.99999999999e 511, for n < 0.

error 69 ln or log of a negative number. The default value is ln (abs (argument)) or log (abs (argument)).

error 70 ln or log of zero. The default value is −9.99999999999e 511.

error 71 asn or acs of number less than −1 or greater than +1. The default value is asn (sgn (argument)) or acs (sgn (argument)).

error 72 Negative base to a non-integer power. The default value is (abs (base)) ↑ (non-integer power).

[2] See 9885 Disk ROM error messages also.

error 73 Zero to the zero power (0↑0). The default value is 1.

error 74 Storage range overflow. The default value is + or−9.99999999999e 99.

error 75 Storage range underflow. The default value is 0.

error 76 Calculation range overflow. The default value is + or −9.99999999999e 511.

error 77 Calculation range underflow. The default value is 0.

Advanced Programming ROM Error Messages

error A0 Relational operator in for statement not allowed. No closing apostrophe in subprogram name.

error A1 A for statement has no matching next statement.

error A2 A next statement encountered without a previous for statement.

error A3 Non-numeric parameter passed as a p-number.

error A4 No return parameter for a function subprogram.

error A5 No functions or subroutines running. Improper p-number reference.

error A6 Attempt to allocate local p-numbers from the keyboard.

error A7 Wrong number of parameters in fts, stf, fti, or itf function. Parameter for stf or itf must be a string (not a numeric). Parameter for stf or itf contains too few characters.

error A8 Overflow or underflow in fts function or overflow in fti function.

error A9 String Variables ROM missing for stf or itf functions.

These mainframe errors have additional meaning when the AP ROM is installed.

error 09 Attempt to execute a next statement from keyboard while for/next loop with same variable is executed in program or from program while for/next loop with same variable is executed from keyboard. Attempt to call a function or subroutine from keyboard.

error 26 A p-number reference is negative.

error 32 Non-numeric value in for statement or non-numeric parameter in fts or fti function.

error 38 Memory overflow during function or subroutine call.

error 40 Memory overflow while using for statement or while allocating local p-numbers.

Extended I/O ROM Error Messages

error E0

- Extended I/O operation executed when a General I/O ROM is not installed.
- HP-IB Error under interrupt: When an HP-IB interrupts with status clear and the ERR bit in the status byte is set, select code 0 is logged in. At the end-of-line service routine, this error is issued.

error E1 Wrong Number of Parameters:

- Bit manipulation functions do not have 2 parameters.
- The on err statement does not have a label.
- The oni statement has less than 2 parameters.
- The polc or rqs statement has less than 2 parameters.
- The tfr statement has less than 2 parameters.
- The cmd statement with bus address has no second parameters.
- The equ or dev statement has an odd number of parameters.
- New buffer allocation with less than 3 parameters.

error E2 Improper Buffer, Device or Equate Table Usage:

- Attempt to add a name in a buffer device or equate table list when that name already exists.
- Buffer, device, or equate name is a null string.
- Attempt to declare multiple listeners with one of the entries not addressing a 98034A Card, or not all on the same HP-IB.
- Read status of multiple listeners.
- Multiple listeners name list ends in a comma.
- Attempt to read to, or write from, a busy buffer.
- Entry in buffer, device, or equate table not found.

error E3 Wrong Parameter Type:

- Parameter of ctbl statement is not a string variable.
- Numeric parameter found when string parameter expected.
- String parameter found when numeric parameter expected.
- Mask parameter in bit function has more than 16 characters.

- Null string found for required string parameter.

error E4 Timeout Error: Specified time ran out without response from peripheral.

error E5 Buffer Overflow or Underflow:

- Attempt to read from an empty buffer or write to a full buffer.
- Attempt to transfer to or from an empty buffer.

error E6 Parameter Overflow:

- Decimal parameter not in range of from −32768 thru 32767 with flag 14 clear.
- Octal parameter not in range of from 0 thru 177777 with flag 14 clear.
- Octal representation contains an 8 or a 9.
- Extended bus address not in range of from 0 thru 31 decimal.
- Buffer type not in the range of from 0 thru 4.
- Negative parameter for buffer size specification.
- Allocating a string as a buffer: After taking 16 characters for working storage, no room left in the string for buffer area.
- Abort byte in eir statement, interrupt enable byte in eir, or character parameter in tfr statement is more than 8 bits; i.e., not in range of from 0 thru 255 decimal or from 0 thru 377 octal.

error E7 Parity Failure: Parity bit of character read does not match specified parity type 1, 2, or 3.

error E8 Improper Interrupt Procedure:

- Attempt to execute an iret statement that is not in a running program, or when no interrupt service routine is active.
- A new program was loaded after an interrupt occured and before the end-of-line service branch, and the service routine was overlayed.
- A new program was loaded from an interrupt service routine and the intercepted line (destination of the iret statement) was overlayed.
- Attempt to transfer a DMA (type 4) buffer with a 98034A HP-IB Interface.
- Attempt to address a select code or a buffer that has not completed the transfer operation. Attempt to read or write with a busy buffer or select code.

error E9 Illegal HP-IB Operation:

- Attempt to address the HP-IB while calculator is not active controller.

- Illegal HP-IB command sequence.
- Attempt to request service on an HP-IB when calculator is active controller.

The Extended I/O ROM adds these meanings to General I/O error messages G4 and G9:

error G4 Improper Select Code:

- Select code parameter of an eir or oni statement is not in range of from 2 thru 15.
- Parameter of an iof or ios statement is not in range of from 0 thru 15.
- Attempt to declare a device name for select code 0 or 1.
- Transfer statement source and destination parameters specify two buffers or two peripherals, rather than one buffer and one peripheral.
- HP-IB control statement used with non-HPIB select code or buffer.
- HP-IB control statement select code specifies bus when only addressed device allowed or addressed device when only bus allowed.

error G9 Improper Hardware Configuration: HP-IB bus functions addressed to non-HP-IB interface card or empty slot.

General I/O ROM Error Messages

error G1 Incorrect format numbers:

- Format number in format statement not in range of 0≤n≤9.
- Referenced format number not executed.

error G2 Referenced format statement has an error:

- Incorrect format spec.
- Numeric overflow in format statement.

error G3 Incorrect I/O parameters:

- Parameter not number or string.
- Negative parameter with f z numeric spec.
- Numeric parameter with c edit spec.
- Binary parameter not in range of −32768≤n≤32767.
- More than one parameter for read binary or read status function.
- Missing parameter or a non-numeric parameter for write control statement.

error G4 Incorrect select code:

- Select code is non-numeric or greater than 4 digits.
- Select code is greater than 2 digits for read status.
- Select code is not in range from 0 through 16.
- Select code 1 allowed only for read status.
- HP-IB address code not in range from 0 through 31.
- Read from select code 0 not allowed.

error G5 Incorrect read parameter:

- Constant in read list.
- String not filled by read operation.
- Numeric parameter references c format spec.

error G6 Incorrect parameter in conversion statement:

- More than 20 parameters.
- Odd number of parameters.
- Non-numeric parameter.
- Parameter not in range 0≤n≤127.

error G7 Unacceptable input data:

- More than one decimal point or "E" read.
- 511 characters read without a LF.
- "E" with no leading digit.
- More than 158 numeric characters read.

error G8 Peripheral device down:

- Incorrect status bits.
- STOP cancelled operation.

error G9 Interface hardware problem:

- Improper HP-IB operation.
- Empty I/O slot.
- Select code does not match interface card (e.g., wrt 711 when a 98032A is set to 7, or wrt 6 when 98034A is set to 6).
- Write Control addressed to a 98034A HP-IB Card.

Matrix ROM Error Messages

error M1* Syntax error.

error M2 Improper dimensions. Array dimensions are incompatible with each other or are incompatible with the stated operation.

error M3 Improper redimension specification: New number of dimensions must equal original number; new size cannot exceed original size.

error M4* Operation not allowed. An array which appears to the left of → cannot also appear on the right.

error M5 Matrix cannot be inverted. Computed determinant equals 0.

9862A Plotter ROM Error Messages

error P1 Wrong state. Statements executed out of order. (See Appendix in ROM manual).

error P2 Wrong number of parameters.

error P3 Wrong type of parameters. Parameters for a label statement must be expressions, text, or string variables.(String Variables ROM is required to use strings.)

error P4 Scale out of range. Maximum value is less than or equal to the minimum value.

error P5 Integer out of range. Pen control parameter is out of the range −32768 to +32767 or the select code is not 0 nor in the range of 2 through 15.

error P6 Character size out of range. Width or Height in letter statement is zero or there is an integer overflow in csize calculations or results.

error P7 Not used.

error P8 Axes origin off scale. X, Y specified for axis statement doesn't fall on plotter surface.

If the error message PLT DOWN occurs, check all plotter connections and be sure the plotter is switched on and the CHART HOLD key is activated.

*These errors give a cursor when the recall key is pressed, showing location of error.

String Variables ROM Error Messages

error S0 Invalid set of strings in data list of load file (ldf) statement.

error S1 Improper argument for string function or string variable.

error S2 More parameters than expected for string function or string variable.

error S3 Accessing or assigning to non-contiguous string. num function of null string.

error S4 Trying to find the value of non-numeric string or null string. Exponent too large. Exponent format invalid (e.g., 1e + +5).

error S5 Invalid destination type for string assignment.

error S6 Parameter is zero, negative, exceeded dimensioned size. Invalid sequence of parameters for string variable.

error S7 String not yet allocated.

error S8 String previously allocated.

error S9 Maximum string length exceeded; additional string length must be specified in dim statement.

9885 Disk ROM Error Messages

Hardware Errors

error d0 Firmware/driver out of synchronization. More than six defective tracks in a row. (Press RESET)

error d1 All drives in system not powered.

error d2 Door opened while disk is being accessed.

error d3 Disk not in drive or no such drive number.

error d4 Write not allowed to protected disk.

error d5 Record header error. (Use Error Recovery Routine)

error d6 Track not found. (Use Error Recovery Routine.)

error d7 Data checkword error. (Use Error Recovery Routine)

error d8 Hardware failure. (Press RESET)

error d9 Verify error due to drive problem. Marginal data. (Reprint data)

Software Errors

error D0 Improper argument.

error D1 Argument out of range.

error D2 Improper file size (negative, 0 or >32767).

error D3 Invalid file name.

error D4 File not found.

error D5 Duplicate file name.

error D6 Wrong file type.

error D7 Directory overflow.

error D8 Insufficient storage space on disk.

error D9 Verify error due to cable, calculator or drive problem. Bad data (Reprint data.)

error F0 File overflow when read or print executed.

error F1 Bootstraps not found. (Reload bootstraps)

error F2 String read but wrong data type encountered.

error F3 Attempt to read data item but type doesn't match.

error F4 Availability table overflow. (Repack)

error F5 Attempt on end branch from other than running program.

error F6 Unassigned data file pointer.

error F7 Disk is down so line cannot be reconstructed.

error F8 Disk is down and STOP pressed.

error F9 System error. (Save files individually and reinitialize)

These mainframe errors take on additional meaning when the Disk ROM is installed.

error 03 Mnemonic not found because disk may be down.

error 29 Line can't be executed because ROM (usually String) is missing.

error 31 Line not found.

error 50 Get or chain should be last statement in a line.

error 51 ROM now installed which wasn't when savem was executed.

error 52 ROM now missing which wasn't when savem was executed.

error 63 Disk load operation would overlay gsb return address so load not executed.

error 64 get, chain or getk not allowed from live keyboard mode or during an ent statement.

These errors may result during the binary Initialization and Error Recovery Routines.

error B0 Wrong syntax, argument out of range or variable not properly dimensioned.

error B1 More than six defective tracks on the disk.

error B2 Verify error. Boots on the disk not identical to boots on the cartridge.

error B3 dtrk, tinit or ltrk not allowed because error information lost or error not d5, d6, d7 or d9.

error B4 Attempt to access record for error correction which isn't part of data file.

error B5 Improper string length (inconsistent with length given in header).

error B6 Not enough space in calculator buffer for data item or item can't be placed in this part of buffer.

error B7 Missing disk or String ROM.

error B8 Track still bad after tinit.

Appendix C
Programming Hints

There are usually several ways to write a program or section of a program to perform a specific job, and the programmer is often faced with the choice of which of several methods to use. Usually the goal is to save program space and execution time and at the same time maintain readability. However, these goals are sometimes conflicting and the programmer must decide which is the overriding concern.

This appendix is not intended to discuss programming techniques in general but to describe a collection of hints for the programmer who wishes to save space or time. While by no means complete, this list describes some of the trade-offs which are "machine dependent" and therefore not necessarily obvious.

In most cases, the time savings are small and are not observable unless the statement is executed thousands of times. The space savings usually only amount to a few bytes. To check the amount of space used by a statement, execute `list -1` after storing the statement.

Method A	Method B	Method Requiring Less Program Storage	Method With Faster Execution Time
Simple Variables	r-variables	A	A
r-variables	one-dimensional array variables	Same	A
Multiple statements per line	One statement per line	A	A
gto +5	gto 5	Same	Same
gto −5	gto 5	Same	Same
gto "5" (one or two character label)	gto 5	Same	Same
gto +5	jmp 5 (Note 1)	B	A
$\sqrt{X}$	X↑.5	A	A
XX	X↑2 (Note 2)	A	Same

Method A	Method B	Method Requiring Less Program Storage	Method With Faster Execution Time
implied multiply	explicit multiply	Same	Same
π	3.14159...	A	A
if flg 2=1	if flg 2	B	B
if flg 2=0	if not flg 2	B	B
if A#0	if A	B	B
if (A<B) or (B<C)	if (A<B) + (B<C)	Same	A
if (A<B) and (B<C)	if (A<B) * (B<C)	Same	A
J+5→K; K−3→L	(J+5→K)−3→L	B	B
J+1→J; if J<5	if (J+1→J)<5	B	B
Specify lower bounds for array dimensions.	Use default lower bounds.	B	Same
Use simple variable as a flag (as 1→A).	flag	(Note 3)	B
Using both tracks alternately.	Using one track at a time, sequentially.	Same	A

Note 1: For computed branching, only jump statement can be used.

Note 2: X↑Y is done by repeated multiplication if Y is an integer.

Note 3: If only one test is made, the flag method takes less room. If two tests are made, both methods are the same. For more than two tests, the simple variable method takes less room.

Appendix D
Calculator Status Conditions

The following table shows the calculator status conditions when the indicated operations are performed. For details about the status condition of modes, variables, etc., see the appropriate section in the manual.

Operation

	Erase all or Power on	Reset	Erase	Run	Continue after editing	Continue
Variables	R	X	R	R	X	X
Flags 0 through 15	R	X	R	R	X	X
Result	R	X	X	X	X	X
Binary program	R	X	X	X	X	X
Subroutine return pointers	R	R	R	R	R	X
Print-all mode	R	R	X	X	X	X
Verify mode	R	R	X	X	X	X
Live keyboard mode	R	R	X	X	X	X
Secure mode	R	X	R	X	X	X
Cassette select code	R	R	X	X	X	X
Cassette track	R	R	X	X	X	X
Angular units for trig functions	R	R	X	X	X	X
Fixed/Float setting	R	R	X	X	X	X
Random number seed	R	R	X	X	X	X
Trace mode	R	R	X	X	X	X

R = Restored to power-on value
X = Unchanged

Extended I/O Status Conditions

The following table shows status conditions for various Extended I/O operations and modes. Notice that the Erase, Erase All-Power on, and Run columns from the previous table are combined into one column here. R = restored to power-on state; X = unchanged.

	Calculator Operation			
Extended I/O ROM Operation or Mode	Power On Erase Erase All Run	Reset	Continue (after edit)	Continue (after Stop)
Conversion and parity tables	R	X	X	X
Binary mode (reset to decimal)	R	X	X	X
I/O buffer area	R	X	X	X
Service name list	R	X	X	X
Equate name list	R	X	X	X
Buffer select code for tfr	R	R	R	X
Interrupt parameters	R	R	R	X
Error recovery routine	R	R	R	X
Timeout routine	R	X	X	X

Appendix E
9825A and 9820A/9821A Compatibility

In general, any program which is used with the HP 9820A/9821A Calculators can be entered into the HP 9825A Calculator with only minor changes, such as changing E (enter exponent) and statement mnemonics to lower case.

The following is a list of subtle differences between the 9820A/9821A and the 9825A Calculators. The list is divided into two sections; those differences which occur when **entering** a program and those which occur when **running** the program.

Entering Programs

- A line label must be followed by a colon. The 9820A/9821A requires a semicolon.
- Parentheses must be used to indicate which relational operator to apply first:

  `if A = B = C;` must be entered as: `if (A = B) = C;` on the 9825A (not the same as `if A = B and B = C`).
- Storing a line with an end statement does not delete higher numbered lines in memory on the 9825A.
- `√-A` is not allowed on the 9825A; use `√(-A)`.
- The enter (ent) statement is syntax checked by the 9825A Calculator. The items in an enter statement must be text or variables. Expressions, such as `ent AA → B` are not allowed on the 9825A; the equivalent on the 9825A would be `ent A; AA → B`.
- A string of unary operators such as `---X` is not allowed on the 9825A.
- File sizes in the mark statement are given in bytes instead of registers. Therefore, `MRK 1, X` in the 9821A becomes `mrk 1, 8X` in the 9825A.
- Linking programs is done differently on the 9825A; `GTO X; LDF Y` becomes `ldf Y, X` on the 9825A.
- `A+B → C+X → Y` on the 9820A/9821A must be typed as `(A+B → C) + X → Y` on the 9825A.

- E-6 on the 9820A/9821A must be written 1e-6.
- The TBL function of the 9820A/9821A Math Block has been replaced as follows:

9820A/9821A Function	9825A Replacement
TBL0	units
TBL1	deg
TBL2	rad
TBL3	grad
TBL4	no replacement
TBL5	csv
TBL6	cfg

Running Programs

- Relational comparisons are made to 12 significant digits on the 9825A. The 9820A/9821A rounds to 10 significant digits and then compares. The 9825A equivalent of: `if X = Y;` is: `if drnd (X,10) = drnd(Y,10)`.
- Floating point numbers are rounded on the 9825A instead of truncated as on the 9820A/9821A when an integer value is required.

  Some implications of this are shown in these examples for the 9825A:

1. r(4.9) refers to r5
2. jmp 2.9 is the same as jmp 3.
3. sfg 5.95 is the same as sfg 6 (and similarly for cfg, flg, and cmf).

- A gto or gsb to a label requires an exact match in the 9825A instead of a match on the last 4 characters as on the 9820A/9821A.
- The 9820A/9821A returned a 0 for 0 ↑ 0. On the 9825A, 0 ↑ 0 results in error 73 (default is 1).
- A number, expression, or statement are valid replies on the 9825A. However, if `ent A` is the enter statement and a statement such as `10→Z` is entered, flag 13 is set and A retains its previous value. For example, no value is entered by the enter statement and flag 13 is set in the following.

`prt X`	Print statement.
`Z→X`	Assignment statement.

These are valid entries:

`A+7π` Expression.

`(Z→K)` Imbedded assignment.

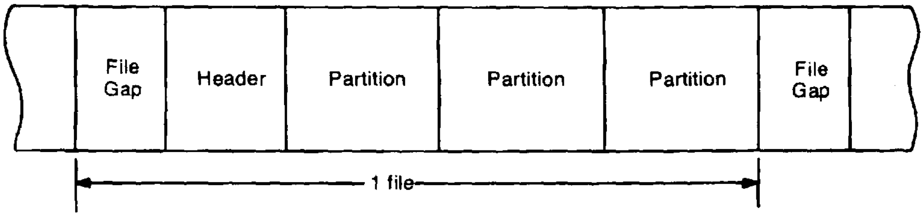
- Flag 13 is cleared when a number or expression is supplied during an enter statement.
- On the 9820A/9821A Calculators, if the run program key is pressed without entering a **value for:** `ent R(X+1→X)`, **the value for** X would not be incremented and RX would not be modified. On the 9825A, the expression, `(X+1→X)`, is executed even if no value is entered.
- The 9825A's integer (int) function is defined as the largest whole number less than or equal to the argument. The 9820A/9821A definition is the largest whole number less than or equal to the absolute value of the argument, with the sign of the result being the same as the sign of the argument.
- On the 9825A, if an error occurs during the execution of a statement, the entire line is aborted. On the 9820A/9821A the rest of the statements in the line are performed.
- Implied storage to Z is replaced by implied storage to result (res). Z is no longer different from other simple variables. A statement with implied storage cannot be stored - a variable must be given explicitly. A program can access the value of result (res), but the value in res cannot be altered by the program.
- Branching to the line which is numbered one higher than the last line of the program no longer treats that line as if it were an end statement.
- Flags are not cleared by the end statement on the 9825A.
- The stop (stp) statement does not destroy subroutine return information.
- On the 9825A, the identify file (idf) statement always positions the tape **before** the header of the identified file. Thus, repeated idf statements do not advance the tape. Also an idf statement followed by a mrk statement marks the identified file (any information on the identified file will be lost).

Appendix F Tape Cartridge Errors

File Body Read Error

If a file body read error (error 46) occurs, first clean the tape head and drive wheel as explained on page 92. Then, execute the statement which caused the error again. If an error still occurs, the next step depends on the type of file being loaded.

It will be informative at this point to explain something about the file structure of the tape. A file is made up of one or more "partitions". This structure makes it possible to recover portions of a file even though a loading error has occured. Error 46 indicates that one or more partitions may be erroneous.



Loading A Program File

If error 46 occurs while loading a program file, one or more program lines may be lost. The place where this error occured is indicated by a line of asterisks (*) inserted in the program at the point where the program lines are missing. These lines can be replaced by referring to a previous listing.

Note that go to and go sub statement addresses are not adjusted during this editing. Thus, it may be necessary to re-adjust the go to and go sub addresses after inserting the lost lines.

Loading a Data File

If error 46 occurs while loading numeric data, the partition in question is marked by a single number replaced by ?.???????????e 00 (in float 11 format). A partition in a numeric data file always contains 32 numbers. With one entry replaced by ?.???????????e 00, there are 31 numbers remaining which may be incorrect. To determine the bounds of the affected partition:

- For r-variables, tre 31 higher numbered r-variables may be incorrect.
- For simple and array variables, determine the order in which the variables in question were allocated (see dimension statement). From the element that is replaced by ?.???????????e 00, go from right to left in the parameter list of the dimension statement. For an array in the list, the first element in the lost partition will have the largest subscripts. Decreasing the leftmost subscript first for an array reveals the missing values. For example, a partition is lost and the dimension statement was:

```
0: dim A,B,C,
 D[3,10]
```

The value D[3,10] contains question marks. All questionable values can be accessed in this order:

→

D[3,10], D[2,10], D[1,10], D[3,9], D[2,9],
D[1,9], D[3,8], D[2,8], D[1,8], D[3,7],
D[2,7], D[1,7], D[3,6], D[2,6], D[1,6],
D[3,5], D[2,5], D[1,5], D[3,4], D[2,4],
D[1,4], D[3,3], D[2,3], D[1,3], D[3,2],
D[2,2], D[1,2], D[3,1], D[2,1], D[1,1],
C, B

File Header Read Error

If a file head read error (error 47) occurs, proceed as follows:

1. Clean the tape head and drive wheel as explained in the section on "Tape Care" on page 91. This may solve the current read error and prevent future read errors.
2. Execute the statement that caused the error again.

> CAUTION
>
> RE-MARKING A FILE HEADER IS A "LAST RESORT" OPERATION, SINCE ALL INFORMATION ON A FILE WITH A RE-MARKED HEADER IS LOST AND THAT FILE CAN NO LONGER BE USED. HOWEVER, THIS DOES PERMIT YOU TO ACCESS FILES BEYOND THE BAD FILE.

3. If, after steps 1 and 2, the error still occurs, re-mark the tape-file header.

To re-mark the head of file N (file which cannot be loaded), execute:

fdf N-1	Positions the tape.
mrk 0,0	Re-marks file header of file N.

For file 0, execute:

`rew`	Positions the tape.
`mrk 0,0`	Re-marks file header of file 0.

After the file header has been re-marked the absolute size of the file is 2 bytes.

Conditioning the Tape

Repeated operations over a short length of tape (usually less than 4000 bytes or 5 ft.) can cause slack. (Extreme changes in temperature can also cause this.) The outer layer of tape can slip and rub on the cartridge, causing damage to the tape. If operation continues, the tape may jam and be ruined.

> NOTE
>
> This condition is most likely to occur if exclusive use is made of one file or two adjacent files near the beginning or end of tape.

If a particular application requires such operation, this slack can be prevented by moving the tape periodically 15 feet or more toward midtape. For example, for a tape with 80 files where only files 0 and 1 are used, execute the following program segment after every 200 operations on file 0 or 1:

```
18: fdf 80
19: rew
```

Tape Life

The tape cartridge does not have an infinite life span. Many factors increase wear and decrease life. A high resistance to turning and continuous use for long periods of time (½ to 3 hours) both result in increased temperature in the cartridge. High humidity, high temperature (above 45°C, 113°F for the cartridge itself) and a high duty cycle (percent of the time the tape is accessed during the total time the 9825A is used) all increase wear.

Several things start happening to the cartridge which are danger signs:

- The tape begins to wear out and lose information.
- The capstan develops dark bumps due to slippage.
- The cartridge can stall, causing the capstan to wear a flat spot on the drive pulley.
- The cartridge sounds rattly, rather that making a constant hum when the tape moves.
- Errors 43 (indicating tape transport failure), 46 and 47 occur more frequently.

If any of these occur, replace the cartridge at once. If you continue to use it, you could lose all the information on the tape and damage the drive itself.

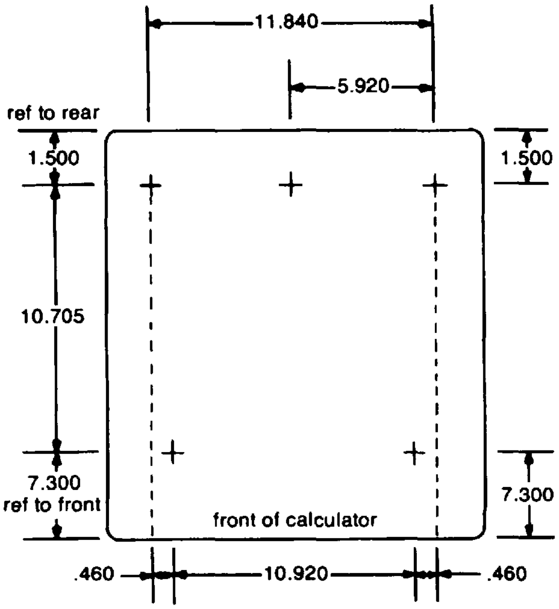
CAUTION

NEVER OVERRIDE `error 43` IN YOUR PROGRAMS. BY OVERRIDING A TRANSPORT ERROR, YOU CAN EASILY DAMAGE THE TRANSPORT AND BE FORCED TO REPLACE IT.

Appendix G
Table Mounting

Your calculator can be mounted to the top of a desk or table by following these steps:

1. Drill 5 holes in the top of your desk or table to accommodate #6-32 (National Coarse) screws according to the diagram below.
2. Remove the Phillips head #6-32NC screws that hold the rubber feet to the bottom of the calculator.
3. Use screws that are ½ inch longer than the thickness of the table top. This ½ inch allows for the thickness of the rubber feet and the hole for the screw in the bottom of the calculator.



HEWLETT PACKARD

SALES & SERVICE OFFICES

AFRICA, ASIA, AUSTRALIA

ANGOLA
Telectra
Empresa Técnica de
Equipamentos
Eléctricos, S.A.R.L.
R. Barbosa Rodrigues, 42-1 DT
Caixa Postal, 6487
Luanda
Tel 35515/6
Cable TELECTRA Luanda

AUSTRALIA
Hewlett-Packard Australia
Pty. Ltd
31-41 Joseph Street
Blackburn, Victoria 3130
P.O. Box 36
Doncaster East, Victoria 3109
Tel. 89-6351
Telex 31-024
Cable HEWPARD Melbourne

Hewlett-Packard Australia
Pty. Ltd
31 Bridge Street
Pymble
New South Wales, 2073
Tel 449-6566
Telex 21561
Cable: HEWPARD Sydney

Hewlett-Packard Australia
Pty. Ltd
153 Greenhill Road
Parkside, S.A., 5063
Tel. 272-5911
Telex: 82536 ADEL
Cable: HEWPARD ADELAID

Hewlett-Packard Australia
Pty Ltd.
141 Stirling Highway
Nedlands, W.A. 6009
Tel: 86-5455
Telex: 93859 PERTH
Cable HEWPARD PERTH

Hewlett-Packard Australia
Pty. Ltd.
121 Wollongong Street
Fyshwick, A.C.T. 2609
Tel 95-2733
Telex 62650 Canberra
Cable: HEWPARD CANBERRA

Hewlett Packard Australia
Pty Ltd
5th Floor
Teachers Union Building
495-499 Boundary Street
Spring Hill, 4000 Queensland
Tel 229-1544
Cable. HEWPARD Brisbane

GUAM
Medical/Pocket Calculators Only
Guam Medical Supply, Inc.
Jay Ease Building, Room 210
P.O. Box 8947
Tamuning 96911
Tel 646-4513
Cable. EARMED Guam

HONG KONG
Schmidt & Co.(Hong Kong) Ltd.
P.O. Box 297
Connaught Centre
39th Floor
Connaught Road, Central
Hong Kong
Tel. H-255291-5
Telex 74766 SCHMC HX
Cable. SCHMIDTCO Hong Kong

INDIA
Blue Star Ltd
Kasturi Buildings
Jamshedji Tata Rd
Bombay 400 020
Tel 29 50 21
Telex: 001-2156
Cable BLUEFROST

Blue Star Ltd
Sahas
414/2 Vir Savarkar Marg
Prabhadevi
Bombay 400 025
Tel 45 78 87
Telex 011-4093
Cable. FROSTBLUE

Blue Star Ltd
Band Box House
Prabhadevi
Bombay 400 025
Tel 45 73 01
Telex 011-3751
Cable: BLUESTAR

Blue Star Ltd
7 Hare Street
P.O. Box 506
Calcutta 700 001
Tel 23-0131
Telex 021-7655
Cable BLUESTAR

Blue Star Ltd
7th & 8th Floor
Bhandari House
91 Nehru Place
New Delhi 110024
Tel 634770 & 635166
Telex: 031-2463
Cable: BLUESTAR

Blue Star Ltd
Blue Star House
11/11A Magarath Road
Bangalore 560 025
Tel: 55668
Telex 043-430
Cable: BLUESTAR

Blue Star Ltd.
Meenakshi Mandiran
xxx/1678 Mahatma Gandhi Rd
Cochin 682 016
Tel: 32069,32161,32282
Telex 0885-514
Cable BLUESTAR

Blue Star Ltd.
1-1-117/1
Sarojini Devi Road
Secunderabad 500 003
Tel: 70126, 70127
Cable: BLUEFROST
Telex 015-459

Blue Star Ltd
2/34 Kodambakkam High Road
Madras 600034
Tel 82056
Telex. 041-379
Cable. BLUESTAR

INDONESIA
BERCA Indonesia P.T
P.O. Box 496/Jkt
JLN•Abdul Muis 62
Jakarta
Tel 40369. 49886.49255.356038
JKT 42895
Cable BERCACON

BERCA Indonesia P.t
63 JL. Raya Gubeng
Surabaya
Tel 44309

ISRAEL
Electronics & Engineering Div
of Motorola Israel Ltd.
17, Kremenetski Street
P.O. Box 25016
Tel-Aviv
Tel 38973
Telex 33569
Cable BASTEL Tel-Aviv

JAPAN
Yokogawa-Hewlett-Packard Ltd
Ohashi Building
59-1 Yoyogi 1-Chome
Shibuya-ku, **Tokyo** 151
Tel 03-370-2281/92
Telex 232-2024YHP MARKET
TOK 23-724
Cable: YHPMARKET

Yokogawa-Hewlett-Packard Ltd.
Chuo Bldg., 4th Floor
4-20, Nishinakajima 5-chome
Yodogawa-ku, Osaka-shi
Osaka,532
Tel 06-304-6021

Yokogawa-Hewlett-Packard Ltd.
Nakamo Building
24 Kami Sasajima-cho
Nakamura-ku, **Nagoya**, 450
Tel: (052) 571-5171

Yokogawa-Hewlett-Packard Ltd.
Tanigawa Building
2-24-1 Tsuruya-cho
Kanagawa-ku
Yokohama, 221
Tel: 045-312-1252
Telex: 382-3204 YHP YOK

Yokogawa-Hewlett-Packard Ltd
Mito Mitsu Building
105, Chome-1,San-no-maru
Mito, Ibaragi 310
Tel: 0292-25-7470

Yokogawa-Hewlett-Packard Ltd.
Inoue Building
1348-3, Asahi-cho, 1-chome
Atsugi, Kanagawa 243
Tel: 0462-24-0452

Yokogawa-Hewlett-Packard Ltd
Kumagaya Asahi
Hachijuni Building
4th Floor
3-4, Tsukuba
Kumagaya, Saitama 360
Tel 0485-24-6563

KENYA
Technical Engineering
Services(E.A.)Ltd.
P.O. Box 18311
Nairobi
Tel: 557726/556762
Cable PROTON

Medical Only
International Aeradio(E.A.)Ltd.
P.O. Box 19012
Nairobi Airport
Nairobi
Tel 336055/56
Telex 22201/22301
Cable INTAERIO Nairobi

KOREA
Samsung Electronics Co., Ltd
20th Fl. Dongbang Bldg. 250, 2-KA
C.P.O. Box 2775
Taepyung-Ro, Chung-Ku
Seoul
Tel: (23) 6811
Telex 22575
Cable: ELEKSTAR Seoul

MALAYSIA
Teknik Mutu Sdn. Bhd.
2 Lorong 13/6A
Section 13
Petaling Jaya, **Selangor**
Tel: 54994/54916
Telex MA 37605

Protel Engineering
P.O. Box 1917
Lot 259, Satok Road
Kuching, **Sarawak**
Tel: 2400
Cable PROTEL ENG

MOZAMBIQUE
A.N. Goncalves, Lta.
162, 1 Apt. 14 Av. D. Luis
Caixa Postal 107
Lourenco Marques
Tel 27091, 27114
Telex. 6-203 NEGON Mo
Cable NEGON

NEW ZEALAND
Hewlett-Packard (N.Z.) Ltd.
P.O. Box 9443
Courtenay Place
Wellington
Tel 877-199
Cable. HEWPACK Wellington

Hewlett-Packard (N.Z.) Ltd
Pakuranga Professional Centre
267 Pakuranga Highway
Box 51092
Pakuranga
Tel 569-651
Cable HEWPACK,Auckland

Analytical/Medical Only
Medical Supplies N.Z. Ltd
Scientific Division
79 Carlton Gore Rd., Newmarket
P.O. Box 1234
Auckland
Tel 75-289
Cable DENTAL Auckland

Analytical/Medical Only
Medical Supplies N.Z. Ltd.
P.O. Box 1994
147-161 Tory St
Wellington
Tel 850-799
Telex 3858
Cable DENTAL, Wellington

Analytical/Medical Only
Medical Supplies N.Z. Ltd
P.O. Box 309
239 Stanmore Road
Christchurch
Tel 892-019
Cable DENTAL, Christchurch

Analytical/Medical Only
Medical Supplies N.Z. Ltd
303 Great King Street
P.O. Box 233
Dunedin
Tel 88-817
Cable DENTAL, Dunedin

NIGERIA
The Electronics
Instrumentations Ltd
N68/770 Oyo Road
Oluseun House
P.M.B. 5402
Ibadan
Tel 61577
Telex 31231 TEIL Nigeria
Cable THETEIL Ibadan

The Electronics Instrumentations Ltd
144 Agege Motor Road, Mushin
P.O. Box 6645
Lagos
Cable THETEIL Lagos

PAKISTAN
Mushko & Company, Ltd
Dosman Chambers
Abdullah Haroon Road
Karachi-3
Tel 511027, 512927
Telex 2894
Cable COOPERATOR Karachi

Mushko & Company, Ltd
38B, Satellite Town
Rawalpindi
Tel 41924
Cable: FEMUS Rawalpindi

PHILIPPINES
The Online Advanced
Systems Corporation
Rico House
Amorsolo cor. Herrera Str
Legaspi Village, Makati
Metro **Manila**
Tel 85-35-81, 85-34-91
Telex 3274 ONLINE

RHODESIA
Field Technical Sales
45 Kelvin Road North
P.O. Box 3458
Salisbury
Tel 705231 (5 lines)
Telex RH 4122

SINGAPORE
Hewlett-Packard Singapore
(Pte.) Ltd
1150 Depot Road
Alexandra P.O. Box 58
Singapore 4
Tel 270-2355
Telex. HPSG RS 21486
Cable HEWPACK, Singapore

SOUTH AFRICA
Hewlett-Packard South Africa
(Pty.), Ltd
Private Bag Wendywood
Sandton, Transvaal 2144
Hewlett-Packard Centre
Daphne Street, Wendywood,
Sandton, Transvaal 2144
Tel: 802-10408
Telex 8-4782
Cable HEWPACK JOHANNESBURG

Service Department
Hewlett-Packard South Africa
(Pty.), Ltd
P.O. Box 39325
Gramley, Sandton, 2018
451 Wynberg Extension 3,
Sandton, 2001
Tel 636-8188/9
Telex 8-2391

Hewlett-Packard South Africa
(Pty.), Ltd
P.O. Box 120
Howard Place, Cape Province, 7450
Pine Park Centre, Forest Drive,
Pinelands, Cape Province, 7405
Tel 53-7955 thu 9
Telex 57-0006

Service Department
Hewlett-Packard South Africa
(Pty.), Ltd
P.O. Box 37099
Overport, Durban 4067
Braby House
641 Ridge Road
Durban, 4001
Tel 88-7478
Telex 6-7954

TAIWAN
Hewlett-Packard Far East Ltd.,
Taiwan Branch
39 Chung Hsiao West Road
Sec. 1, 7th Floor
Taipei
Tel 3819160-4
Cable HEWPACK TAIPEI

Hewlett-Packard Far East Ltd
Taiwan Branch
68-2, Chung Cheng 3rd. Road
Kaohsiung
Tel (07) 242318-Kaohsiung

Analytical Only
San Kwang Instruments Co., Ltd.
No. 20, Yung Sui Road
Taipei
Tel 3715171-4 (5 lines)
Telex 22894 SANKWANG
Cable: SANKWANG TAIPEI

TANZANIA
Medical Only
International Aeradio (E.A.), Ltd
P.O. Box 861
Dar es Salaam
Tel 21251 Ext. 265
Telex 41030

THAILAND
UNIMESA Co., Ltd.
Elcom Research Building
2538 Sukumvit Ave
Bangkok
Tel: 3932387, 3930338
Cable UNIMESA Bangkok

UGANDA
Medical Only
International Aeradio(E.A.), Ltd.
P.O. Box 2577
Kampala
Tel 54388
Cable INTAERIO Kampala

ZAMBIA
R.J. Tilbury (Zambia) Ltd.
P.O. Box 2792
Lusaka
Tel 73793
Cable: ARJAYTEE, Lusaka

OTHER AREAS NOT LISTED, CONTACT
Hewlett-Packard Intercontinental
3200 Hillview Ave
Palo Alto, California 94304
Tel (415) 493-1501
TWX 910-373-1267
Cable HEWPACK Palo Alto

CANADA

ALBERTA
Hewlett-Packard (Canada) Ltd
11620A - 168th Street
EdmontonT5M 3T9
Tel: (403) 452-3670
TWX: 610-831-2431

Hewlett-Packard (Canada) Ltd
210 7220 Fisher St. S.E.
Calgary T2H 2H8
Tel. (403) 253-2713
Twx. 610-821-6141

BRITISH COLUMBIA
Hewlett-Packard (Canada) Ltd
837 E. Cordova Street
Vancouver V6A 3R2
Tel: (604) 254-0531
TWX: 610-922-5059

MANITOBA
Hewlett-Packard (Canada) Ltd
513 Century St.
St. James
Winnipeg R3H 0L8
Tel: (204) 786-7581
TWX 610-671-3531

NOVA SCOTIA
Hewlett-Packard (Canada) Ltd.
800 Windmill Road
Dartmouth B3B 1L1
Tel: (902) 469-7820
TWX 610-271-4482 HFX

ONTARIO
Hewlett-Packard (Canada) Ltd
1020 Morrison Dr
Ottawa K2H 8K7
Tel. (613) 820-6483
TWX 610-563-1636

Hewlett-Packard (Canada) Ltd
6877 Goreway Drive
Mississauga L4V 1M8
Tel (416) 678-9430
TWX 610-492-4246

QUEBEC
Hewlett-Packard (Canada) Ltd
275 Hymus Blvd
Pointe Claire H9R 1G7
Tel (514) 697-4232
TWX 610-422-3022
TLX 05-821521 HPCL

FOR CANADIAN AREAS NOT LISTED:
Contact Hewlett-Packard (Canada)
Ltd. in Mississauga

CENTRAL AND SOUTH AMERICA

ARGENTINA
Hewlett-Packard Argentina
S.A.
Av. Leandro N. Alem 822 - 12
1001**Buenos Aires**
Tel 31-6063,4,5,6 and 7
Telex 122443 AR CIGY
Cable HEWPACK ARG

BOLIVIA
Casa Kavlin S.A.
Calle Potosi 1130
P.O. Box 500
La Paz
Tel 41530,53221
Telex CWC BX 5298 ITT 3560082
Cable KAVLIN

BRAZIL
Hewlett-Packard do Brasil
I.e.C. Ltda
Avenida Rio Negro 980
Alphaville
06400**Barueri** SP
Tel 429-3222

Hewlett-Packard do Brasil
I.e.C. Ltda
Rua Padre Chagas, 32
90000-**Pórto Alegre**-RS
Tel (0512) 22-2998, 22-5621
Cable HEWPACK Pôrto Alegre

Hewlett-Packard do Brasil
I.E.C. Ltda
Rua Siqueira Campos, 53
Copacabana
20000-**Rio de Janeiro**
Tel 257-80-94-DDD (021)
Telex 391-212-1905 HEWP-BR
Cable HEWPACK
Rio de Janeiro

CHILE
Calcagni y Metcalfe Ltda
Alameda 580-01 807
Casilla 2118
Santiago, 1
Tel 398613
Telex 3520001 CALMET
Cable CALMET Santiago

COLOMBIA
Instrumentación
Henrik A. Langebaek & Kier S.A.
Carrera 7 No. 48-75
Apartado Aéreo 6287
Bogotá, I.D.E.
Tel 69-88-77
Cable AARIS Bogotá
Telex 044-400

COSTA RICA
Científica Costarricense S.A.
Avenida 2. Calle 5
San Pedro de Montes de Oca
Apartado 10159
San Jose
Tel 24-38-20 24-08-19
Telex 2367 GALGUR CR
Cable GALGUR

ECUADOR
Calculators Only
Computadoras y Equipos
Electrónicos
P.O. Box 6423 CCI
Eloy Alfaro #1824,3 Piso
Quito
Tel 453482
Telex 02-2113 Sagita Ed
Cable Sagita-Quito

EL SALVADOR
Instrumentacion y Procesamiento
Electronico de el Salvador
Bulevar de los Heroes 11-48
San Salvador
Tel 252787

GUATEMALA
IPESA
Avenida La Reforma 3-48
Zona 9
Guatemala City
Tel 63627, 64786
Telex 4192 Teletro Gu

MEXICO
Hewlett-Packard Mexicana,
S.A. de C.V.
Av. Periférico Sur No. 6501
Tepepan, Xochimilco
Mexico 23, D.F.
Tel 905-676-4600

Hewlett-Packard Mexicana,
S.A. de C.V.
Ave. Constitución No. 2184
Monterrey, N.L.
Tel 48-71-32, 48-71-84
Telex 038-410

NICARAGUA
Roberto Terán G.
Apartado Postal 689
Edificio Terán
Managua
Tel 25114, 23412,23454
Cable ROTERAN Managua

PANAMA
Electrónico Balboa, S.A.
P.O. Box 4929
Calle Samuel Lewis
Cuidad de Panama
Tel 64-2700
Telex 3483103 Curunda,
Canal Zone
Cable ELECTRON Panama

PERU
Compañía Electro Médica S.A.
Los Flamencos 145
San Isidro Casilla 1030
Lima 1
Tel 41-4325
Cable ELMEO Lima

PUERTO RICO
Hewlett-Packard Inter-Americas
Puerto Rico Branch Office
Calle 272,
No. 203 Urb. Country Club
Carolina 00924
Tel (809) 762-7255
Telex 345 0514

URUGUAY
Pablo Ferrando S.A.
Comercial e Industrial
Avenida Italia 2877
Casilla de Correo 370
Montevideo
Tel 40-3102
Cable RADIUM Montevideo

VENEZUELA
Hewlett-Packard de Venezuela
C.A.
P.O. Box 50933
Caracas 105
Los Ruices Norte
3a Transversal
Edificio Segre
Caracas 107
Tel 35-00-11 (20 lines)
Telex 25146 HEWPACK
Cable HEWPACK Caracas

FOR AREAS NOT LISTED, CONTACT:
Hewlett-Packard
Inter-Americas
3200 Hillview Ave
Palo Alto, California 94304
Tel (415) 493-1501
TWX 910-373-1260
Cable HEWPACK Palo Alto
Telex 034-8300, 034-8493

EUROPE, NORTH AFRICA AND MIDDLE EAST

AUSTRIA
Hewlett-Packard Ges.m.b.H.
Handelskai 52
P.O. box 7
A-1205 Vienna
Tel: (0222) 351621 to 27
cable: HEWPAK Vienna
Telex: 75923 hewpak a

BELGIUM
Hewlett-Packard Benelux
S.A./N.V.
Avenue de Col-Vert, 1,
(Groenkraaglaan)
B-1170 Brussels
Tel: (02) 672 22 40
Cable: PALOBEN Brussels
Telex: 23 494 paloben bru

CYPRUS
Kypronics
19, Gregorios & Xenopoulos Rd.
P.O. Box 1152
CY-Nicosia
Tel: 45628/29
Cable: KYPRONICS PANDEHIS
Telex: 3018

CZECHOSLOVAKIA
Vyvojova a Provozni Zakladna
Vyzkumnych Ustavu v Bechovicich
CSSR-25097 Bechovice u Prahy
Tel: 89 93 41
Telex: 121333

Institute of Medical Bionics
Vyskumny Ustav Lekarskej Bioniky
Jedlova 6
CS-88346 Bratislava-Kramare
Tel: 44-551/45-541

DDR
Entwicklungslabor der TU Dresden
Forschungsinstitut Meinsberg
DDR-7305
Waldheim/Meinsberg
Tel: 37 667
Telex: 112145

Export Contact AG Zuerich
Guenther Forgber
Schlegelstrasse 15
1040 Berlin
Tel: 42-74-12
Telex: 111889

DENMARK
Hewlett-packard A/S
Datavej 52
DK-3460 Birkerød
Tel: (02) 81 66 40
Cable: HEWPACK AS
Telex: 37409 hpas dk

Hewlett-Packard A/S
Navervej 1
DK-8600 Silkeborg
Tel: (06) 82 71 66
Telex: 37409 hpas dk
Cable: HEWPACK AS

FINLAND
Hewlett-Packard OY
Nahkahousuntie 5
P.O. Box 6
SF-00211 Helsinki 21
Tel: (90) 6923031
Cable: HEWPACKOY Helsinki
Telex: 12-1563 HEWPA SF

FRANCE
Hewlett-Packard France
Quartier de Courtaboeuf
Boite Postale No. 6
F-91401 Orsay Cédex
Tel: (1) 907 78 25
Cable: HEWPACK Orsay
Telex: 600048

Hewlett-Packard France
Agency Régionale
"Le Saquin"
Chemin des Mouilles
B.P. 162
F-69130 Ecully
Tel: (78) 33 81 25,
Cable: HEWPACK Ecuiy
Telex: 31 06 17

Hewlett-Packard France
Agence Régionale
Péricentre de la Cépière
Chemin de la Cépière, 20
F-31300 Toulouse-Le Mirail
Tel:(61) 40 11 12
Cable: HEWPACK 51957
Telex: 510957

Hewlett-Packard France
Agence Régionale
Aéroport principal de
Marseille-Marignane
F-13721 Marignane
Tel: (91) 89 12 36
Cable: HEWPACK MARGN
Telex: 410770

Hewlett-Packard France
Agence Régionale
63, Avenue de Rochester
B.P. 1124
F-35014 Rennes Cédex
Tel: (99) 36 33 21
Cable: HEWPACK 74912
Telex: 740912

Hewlett-Packard France
Agence Régionale
74, Allée de la Robertsau
F-67000 Strasbourg
Tel: (88) 35 23 20/21
Telex: 890141
Cable: HEWPACK STRBG

Hewlett-Packard France
Agence Régionale
Centre Vauban
201, rue Colbert
Entrée A2
F-59000 Lille
Tel: (20) 51 44 14
Telex: 820744

Hewlett-Packard France
Centre d' Affaires Paris-Nord
Bâtiment Ampère
Rue de La Commune de Paris
B.P. 300
F-93153 Le Blanc Mesnil Cédex
Tel: (01) 931 88 50

GERMAN FEDERAL REPUBLIC
Hewlett-Packard GmbH
Vertriebszentrale Frankfurt
Bernerstrasse 117
Postfach 560 140
D-6000 Frankfurt 56
Tel: (0611) 50 04-1
Cable: HEWPACKSA Frankfurt
Tel: (0611) 50 04-1
Cable: HEWPACKSA Frankfurt
Telex: 04 13249 hpffmd

Hewlett-Packard GmbH
Technisches Buero Böblingen
Herrenbergerstrasse 110
D-7030 Böblingen, Württemberg
Tel: (07031) 667-1
Cable: HEPAK Böblingen
Telex: 07265739 bbn

Hewlett-Packard GmbH
Technisches Buero Düsseldorf
Emanuel-Leutze-Str.1 (Seestern)
D-4000 Düsseldorf 11
Tel: (0211) 59711
Telex: 085/86 533 hpdd d

Hewlett-Packard GmbH
Technisches Buero Hamburg
Wendenstrasse 23
D-2000 Hamburg 1
Tel: (040) 24 13 93
Cable: HEWPACKSA Hamburg
Telex: 21 63 032 hphh d

Hewlett-Packard GmbH
Technisches Buero Hannover
Am Grossmarkt 6
D-3000 Hannover 91
Tel: (0511) 46 60 01
Telex: 092 3259

Hewlett-Packard GmbH
Werk Groetzingen
Ohmstrasse 6
D-7500 Karlsruke 41
Tel: (0721) 69 40 06
Telex: 07-825707

Hewlett-Packard GmbH
Technisches Buero Nuremberg
Neumeyer Str. 90
D-8500 Nuremberg
Tel: (0911) 56 30 83/85
Telex: 0623 860

Hewlett-Packard GmbH
Technisches Buero Munchen
Unterhachinger Strasse 28
ISAR Center
D-8012 Ottobrunn
Tel: (089) 601 30 61/7
Cable: HEWPACKSA München
Telex: 0524985

Hewlett-Packard GmbH
Technisches Buero Berlin
Keith Strasse 2-4
D-1000 Berlin 30
Tel: (030) 24 90 86
Telex: 18 3405 hpbln d

GREECE
Kostas Karayannis
08, Omirou Street
GR-Athens 133
Tel: 3237731
Cable: RAKAR Athens
Telex: 21 59 62 rkar gr

Analytical Only
"INTECO" G. Papathanassiou & Co.
Marni 17
GR - Athens 103
Tel: 522 1915
Cable: INTEKNIKA Athens
Telex: 21 5329 INTE GR

Medical Only
Technomed Hellas Ltd.
52, Skoufa Street
GR - Athens 135
Tel: 362 6972, 363 3830
Cable:ETALAK athens
Telex: 21-4693 ETAL GR

HUNGARY
MTA
Müszerügyi és Méréstechnikai
Szolgalata
Lenin Krt. 67
1391 Budapest VI
Tel: 42 03 38
Telex: 22 51 14

ICELAND
Medical Only
Elding Trading Company Inc.
Hafnarhvoli - Tryggvatotu
IS-Reykjavik
Tel: 1 58 20
Cable: ELDING Reykjavik

IRAN
Hewlett-Packard Iran Ltd.
No. 13, Fourteenth St.
Miremad Avenue
P.O. Box 41/2419
IR-Tehran
Tel: 851082-7
Telex: 21 25 74 khrm ir

IRAQ
Hewlett-Packard Trading Co.
4/1/8 Mansoor City
Baghdad
Tel: 5517827
Telex: 2455 hepairaq ik
Cable: HEWPACDAD,
Baghdad Iraq

IRELAND
Hewlett-Packard Ltd.
King Street Lane
GB-Winnersh,Wokingham
Berks, RG11 5AR
Tel: (0734) 78 47 74
Telex: 847178/848179

ITALY
Hewlett-Packard Italiana S.p.A.
Via Amerigo Vespucci 2
Casella postale 3645
I-20100 Milano
Tel: (2) 6251 (10 lines)
Cable: HEWPACKIT Milano
Telex: 32046

Hewlett-Packard Italiana S.p.A.
Via Pietro Maroncelli 40
(ang. Via Visentin)
I-35100 Padova
Tel: (49) 66 48 88
Telex: 41612 Hewpacki

Medical only
Hewlett-Packard Italiana S.p.A.
Via d'Aghiardi, 7
I-56100 Pisa
Tel: (050) 2 32 04
Telex: 32046 via Milano

Hewlett-Packard Italiana S.p.A.
Via G. Armellini 10
I-00143 Roma
Tel: (06) 54 69 61
Telex: 61514
Cable: HEWPACKIT Roma

Hewlett-Packard Italiana S.p.A.
Corso Giovanni Lanza
I-1031 Torino
Tel: (011) 682245/659308

Medical/Calculators Only
Hewlett-Packard Italiana S.p.A.
Via Principe Nicola 43 G/C
I-95126 Catania
Tel:(095) 37 05 04

Hewlett-Packard Italiana S.p.A.
Via Amerigo Vespucci, 9
I-80142 Napoli
Tel: (081) 33 77 11

Hewlett-Packard Italiana S.p.A.
Via E. Masi, 9/B
I-40137 Bologna
Tel: (051) 30 78 87

KUWAIT
Al-Khaldiya Trading &
Contracting Co.
P.O. Box 830-Saft
Kuwait
Tel: 424910-411726
Telex: 2481 areeg kt
Cable: VISCOUNT

LUXEMBURG
Hewlett-Packard Benelux
S.A./N.V.
Avenue du Col-Vert, 1,
(Groenkraaglaan)
B-1170 Brussels
Tel: (02) 672 22 40
Cable: PALOBEN Brussels
Telex: 23 494

MOROCCO
Gerep
190, Blvd. Brahim Roudani
Casablanca
Tel: 25-16-76/25-90-99
Cable: Gerep-Casa
Telex: 23739

NETHERLANDS
Hewlett-Packard Benelux N.V.
Van Heuven Goedhartlaan 121
P.O. Box 667
NL-1134 Amstelveen
Tel: (020) 47 20 21
Cable: PALOBEN Amsterdam
Telex: 13 216 hepa nl

NORWAY
Hewlett-Packard Norge A/S
Nesveien 13
Box 149
N-1344 Haslum
Tel: (02) 53 83 60
Telex: 16621 hpnas n

POLAND
Biuro Informacji Technicznej
Hewlett-Packard
U1 Stawki 2, 6P
00-950 Warszawa
Tel: 395962/395187
Telex: 81 24 53 hepa pl

UNIPAN
Zaklad Doswiadczalny
Budowy Aparatury Naukowej
U1. Krajowej Rady Narodowej 51/55
00-800 Warszawa
Tel: 36190
Telex: 81 46 48

Zaklady Naprawcze Sprzetu
Medycznego
Plac Komuny Paryskiej 6
90-007 Lódź
Tel: 334-41, 337-83

PORTUGAL
Telectra-Empresa Técnica de
Equipamentos Eléctricos S.a.r.l.
Rua Rodrigo da Fonseca 103
P.O. Box 2531
P-Lisbon 1
Tel: (19) 68 60 72
Cable: TELECTRA Lisbon
Telex: 12598

Medical only
Mundinter
Intercambio Mundial de Comércio
S.a.r.l.
Av.A.A.de Aquiar 138
P.O. Box 2761
P - Lisbon
Tel: (19) 53 21 31/7
Cable: INTERCAMBIO Lisbon

RUMANIA
Hewlett-Packard Reprezentanta
Bd.N. Balcescu 16
Bucharest
Tel: 158023/138885
Telex: 10440

I.I.R.U.C.
Intreprinderea Pentru
Intretinerea
Si Repararea Utilajelor de Calcul
B-dul prof. Dimitrie Pompei 6
Bucharest-Sectorul 2
Tel: 12 64 30
Telex: 11716

SAUDI ARABIA
Modern Electronic Establishment
King Abdul Aziz str.(Head office)
P.O. Box 1228
Jeddah
Tel: 31173-332201
Cable: ELECTRA

P.O. Box 2728 (Service center)
Riyadh
Tel: 62596-66232
Cable: RADUFCO

SPAIN
Hewlett-Packard Española, S.A.
Jerez, Calle 3
E-Madrid 16
Tel:(1) 458 26 00 (10 lines)
Telex: 23515 hpe

Hewlett-Packard Española, S.A.
Milanesado 21-23
E-Barcelona 17
Tel: (3) 203 6200 (5 lines)
Telex: 52603 hpbe e

Hewlett-Packard Española, S.A.
Av Ramón y Cajal, 1
Edificio Sevilla, planta 9°,
E-Sevilla 5
Tel: 64 44 54/58

Hewlett-Packard Española S.A.
Edificio Albia II 7° B
E-Bilbao-1
Tel: 23 83 06/23 82 06

Calculators Only
Hewlett-Packard Española S.A.
Gran Via Fernando El Catdlico, 67
E-Valencia-8
Tel: 326 67 28/326 85 55

SWEDEN
Hewlett-Packard Sverige AB
Enighetsvägen 1-3
Fack
S-161 20 Bromma 20
Tel: (08) 730 05 50
Cable: MEASUREMENTS
Stockholm
Telex: 10721

Hewlett-Packard Sverige AB
Ostra Vintergatan 22
S-702 40 Orebro
Tel: (019) 14 07 20

Hewlett-Packard Sverige AB
Frötallisgatan 30
S-421 32 Vastra Frölunda
Tel: (031) 49 09 50
Telex: 10721 Via Bromma Office

SWITZERLAND
Hewlett-Packard (Schweiz) AG
Zürcherstrasse 20
P.O. Box 307
CH-8952 Schlieren-Zurich
Tel: (01) 730 52 40/730 18 21
Cable: HPAG CH
Telex: 53933 hpag ch

Hewlett-Packard (schweiz) AG
Château Bloc 19
CH-1219 Le Lignon-Geneva
Tel: (022) 96 03 22
Cable: HEWPACKAG Geneva
Telex: 27 333 hpag ch

SYRIA
Medical/Calculator only
Sawah & Co.
Place Azmé
B.P. 2308
SYR-Damascus
Tel: 16367, 19697, 14268
Cable: SAWAH, Damascus

TURKEY
Telekom Engineering Bureau
P.O. Box 437
Beyoglu
TR-Istanbul
Tel: 49 40 40
Cable: TELEMATION Istanbul
Telex: 23609

Medical only
E.M.A.
Muhendislik Kollektif Sirketi
Adakale Sokak 41/6
TR-Ankara
Tel: 175622

Analytical only
Yilmaz Ozyurek
Milli Mudafaa Cad No. 16/6
Kizilay
TR-Ankara
Tel: 25 03 09
Telex: 42576 ozek tr

UNITED KINGDOM
Hewlett-Packard Ltd.
King Street Lane
GB-Winnersh, Wokingham
Berks. RG11 5AR
Tel: (0734) 78 47 74
Cable: Hewpie London
Telex:847178/9

Hewlett-Packard Ltd.
Trafalgar House,
Navigation Road
Altrincham
Cheshire WA14 1NU
Tel: (061) 928 6422
Telex: 668068

Hewlett-Packard Ltd.
Lygon Court
Hereward Rise
Dudley Road
Halesowen,
West Midlands B62 8SD
Tel: (021) 550 9911
Telex: 339105

Hewlett-Packard Ltd.
Wedge House
799, London Road
GB-Thornton Heath
Surrey CR4 6XL
Tel: (01) 684 0103/8
Telex: 946825

Hewlett-Packard Ltd.
c/o Makro
South Service wholesale Centre
Wear Industrial Estate
Washington
GB-New Town, County Durham
Tel: Washington 464001 ext. 57/58

Hewlett-Packard Ltd
10, Wesley St.
GB-Castleford
West Yorkshire WF10 1AE
Tel: (09775) 50402
Telex: 557355

Hewlett-Packard Ltd
1, Wallace Way
GB-Hitchin
Herts
Tel: (0462) 52824/56704
Telex: 825981

Hewlett-Packard Ltd
2C, Avonbeg Industrial Estate
Long Mile Road
Dublin 12
Tel: Dublin 509458
Telex: 30439

USSR
Hewlett-Packard
Representative Office USSR
Pokrovsky Boulevard 4/17-KW 12
Moscow 101000
Tel:294-2024
Telex: 7825 hewpak su

YUGOSLAVIA
Iskra-standard/Hewlett-Packard
Miklosiceva 38/VII
61000 Ljubljana
Telb. 31 58 79/32 16 74
Telex: 31583

SOCIALIST COUNTRIES NOT SHOWN PLEASE CONTACT:
Hewlett-Packard Ges.m.b.H
P.O. Box 7
A-1205 Vienna, Austria
Tel: (0222) 35 16 21 to 27
Cable: HEWPAK Vienna
Telex: 75923 hewpak a

MEDITERRANEAN AND MIDDLE EAST COUNTRIES NOT SHOWN PLEASE CONTACT:
Hewlett-Packard S.A.
Mediterranean and Middle
East Operations
35, Kolokotroni Street
Platia Kefallariou
GR-Kifissia-Athens, Greece
Tel: 8080337/359/429
Telex: 21-6588
Cable: HEWPACKSA Athens

FOR OTHER AREAS NOT LISTED CONTACT
Hewlett-Packard S.A.
7, rue du Bois-du-Lan
P.O. Box
CH-1217 Meyrin 2 - Geneva
Switzerland
Tel: (022) 82 70 00
Cable: HEWPACKSA Geneva
Telex: 2 24 86

UNITED STATES

ALABAMA
8290 Whitesburg Dr., S.E.
P.O. Box 4207
Huntsville 35802
Tel: (205) 881-4591

Medical Only
228 W. Valley Ave.,
Room 220
Birmingham 35209
Tel: (205) 942-2081/2

ARIZONA
2336 E. Magnolia St.
Phoenix 85034
Tel: (602) 244-1361

2424 East Aragon Rd.
Tucson 85706
Tel: (602) 294-3148

*ARKANSAS
Medical Service Only
P.O. Box 5646
Brady Station
Little Rock 72215
Tel: (501) 376-1844

CALIFORNIA
1430 East Orangethorpe Ave
Fullerton 92631
Tel: (714) 870-1000

3939 Lankershim Boulevard
North Hollywood 91604
Tel: (213) 877-1282
TWX: 910-499-2671

5400 West Rosecrans Blvd
P.O. Box 92105
World Way Postal Center
Los Angeles 90009
Tel: (213) 970-7500

*Los Angeles
Tel: (213) 776-7500

3003 Scott Boulevard
Santa Clara 95050
Tel: (408) 249-7000
TWX: 910-338-0518

*Ridgecrest
Tel: (714) 446-6165

646 W. North Market Blvd
Sacramento 95834
Tel: (916) 929-7222

9606 Aero Drive
P.O. Box 23333
San Diego 92123
Tel: (714) 279-3200

COLORADO
5600 South Ulster Parkway
Englewood 80110
Tel: (303) 771-3455

CONNECTICUT
12 Lunar Drive
New Haven 06525
Tel: (203) 389-6551
TWX: 710-465-2029

FLORIDA
P.O. Box 24210
2806 W. Oakland Park Blvd.
Ft. Lauderdale 33311
Tel: (305) 731-2020

*Jacksonville
Medical Service only
Tel: (904) 398-0663

P.O. Box 13910
6177 Lake Ellenor Dr
Orlando 32809
Tel: (305) 859-2900

P.O. Box 12826
Pensacola 32575
Tel: (904) 476-8422

GEORGIA
P.O. Box 105005
Atlanta 30348
Tel: (404) 955-1500
TWX 810-766-4890

Medical Service Only
*Augusta 30903
Tel: (404) 736-0592

P.O. Box 2103
Warner Robins 31098
Tel: (912) 922-0449

HAWAII
2875 So. King Street
Honolulu 96814
Tel: (808) 955-4455
Telex: 723-705

ILLINOIS
5201 Tollview Dr.
Rolling meadows 60008
Tel: (312) 255-9800
TWX: 910-687-2260

INDIANA
7301 North Shadeland Ave
Indianapolis46250
Tel: (317)842-1000
TWX: 810-260-1797

IOWA
2415 Heinz Road
Iowa City 52240
Tel: (319) 338-9466

KENTUCKY
Medical Only
Atkinson Square
3901 Atkinson Dr.,
Suite 407 Atkinson Square
Louisville 40218
Tel: (502) 456-1573

LOUISIANA
P.O. Box 840
3229-39 Williams Boulevard
Kenner 70063
Tel: (504) 443-6201

MARYLAND
6707 Whitestone Road
Baltimore 21207
Tel: (301) 944-5400
TWX: 710-862-9157

2 Choke Cherry Road
Rockville 20850
Tel: (301) 948-6370
TWX: 710-828-9684

MASSACHUSETTS
32 Hartwell Ave
Lexington 02173
Tel: (617) 861-8960
TWX: 710-326-6904

MICHIGAN
23855 Research Drive
Farmington Hills 48024
Tel: (313) 476-6400

724 West Centre Ave.
Kalamazoo 49002
Tel: (606) 323-8362

MINNESOTA
2400 N. Prior Ave
St. Paul 55113
Tel: (612) 636-0700

MISSISSIPPI
*Jackson
Medical Service only
Tel: (601) 982-9363

MISSOURI
11131 Colorado Ave.
Kansas City 64137
Tel: (816) 763-8000
TWX: 910-771-2087

1024 Executive Parkway
St. Louis 63141
Tel: (314) 878-0200

NEBRASKA
Medical Only
7171 Mercy Road
Suite 110
Omaha 68106
Tel: (402) 392-0948

NEW JERSEY
W. 120 Century Rd.
Paramus 07652
Tel: (201) 265-5000
TWX: 710-990-4951

Crystal Brook Professional
Building
Eatontown 07724
Tel:(201) 542-1384

NEW MEXICO
P.O. Box 11634
Station E
11300 Lomas Blvd., N.E.
Albuquerque 87123
Tel: (505) 292-1330
TWX: 910-989-1185

156 Wyatt Drive
Las Cruces 88001
Tel: (505) 526-2484
TWX: 910-9983-0550

NEW YORK
6 Automation Lane
Computer Park
Albany 12205
Tel: (518) 458-1550

201 South Avenue
Poughkeepsie 12601
Tel: (914) 454-7330
TWX: 510-253-5981

650 Perinton Hill Office Park
Fairport 14450
Tel: (716) 223-9950

5858 East Molloy Road
Syracuse 13211
Tel: (315) 454-2486
TWX: 710-541-0482

1 Crossways Park West
Woodbury 11797
Tel: (516) 921-0300
TWX: 710-990-4951

NORTH CAROLINA
P.O. Box 5188
1923 North Main Street
High Point 27262
Tel: (919) 885-8101

OHIO
16500 Sprague Road
Cleveland 44130
Tel: (216) 243-7300
TWX: 810-423-9430

330 Progress Rd
Dayton 45449
Tel: (513) 859-8202

1041 Kingsmill Parkway
Columbus 43229
Tel: (614) 436-1041

OKLAHOMA
P.O. Box 32008
Oklahoma City 73132
Tel: (405) 721-0200

OREGON
17890 SW Lower Boones
Ferry Road
Tualatin 97062
Tel: (503) 620-3350

PENNSYLVANIA
111 Zeta Drive
Pittsburgh 15238
Tel: (412) 782-0400

1021 8th Avenue
King of Prussia Industrial Park
King of Prussia 19406
Tel: (215) 265-7000
TWX: 510-660-2670

SOUTH CAROLINA
6941-0 N. Trenholm Road
Columbia 29260
Tel: (803) 782-6493

TENNESSEE
*Knoxville
Medical Service only
Tel: (615) 523-5022

3027 Vanguard Dr
Director's Plaza
Memphis 38131
Tel: (901) 346-8370

Nashville
Medical Service only
Tel: (615) 244-5448

TEXAS
P.O. Box 1270
201 E. Arapaho Rd
Richardson 75080
Tel: (214) 231-6101

10535 Harwin Dr.
Houston 77036
Tel: (713) 776-6400

205 Billy Mitchell Road
San Antonio 78226
Tel: (512) 434-8241

UTAH
2160 South 3270 West Street
Salt Lake City 84119
Tel: (801) 972-4711

VIRGINIA
P.O. Box 12778
No. 7 Koger Exec. Center
Suite 212
Norfolk 23502
Tel:(804) 461-4025/6

P.O. Box 9669
2914 Hungary Springs Road
Richmond 23228
Tel: (804) 285-3431

WASHINGTON
Bellefield Office Pk.
1203-114th Ave. S.E.
Bellevue 98004
Tel: (206) 454-3971
TWX: 910-443-2446

*WEST VIRGINIA
Medical/Analytical Only
Charleston
Tel: (304) 345-1640

WISCONSIN
9004 West Lincoln Ave
West Allis 53227
Tel: (414) 541-0550

FOR U.S. AREAS NOT LISTED:
Contact the regional office
nearest you: Atlanta, Georgia
North Hollywood, California
Rockville, Maryland, Rolling Meadows
Illinois. Their complete
addresses are listed above.

*Service Only

1/78

Subject Index

HEWLETT hp PACKARD

PART NO. 09825-90000
MICROFICHE NO. 09825-99000

PRINTED IN U.S.A.
October 1, 1976