



Hewlett-Packard 9815A/S 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)
Copyright by Hewlett-Packard Company 1975

Manual Summary

Introduction (page ix)

Briefly introduces the calculator, peripherals, interfaces, and other calculator accessories.

Section 1: KEYBOARD OPERATIONS (page 1)

Describes the HP 9815A strictly as a calculator: Simple and complex arithmetic, number formats, scientific functions, storage registers, etc.

Section 2: PRINTER CONTROL (page 27)

Lists all printer functions and describes how to print formatted alphanumeric messages.

Section 3: PROGRAMMING (page 33)

Shows how to enter and run programs, and describes all programming functions.

Section 4: TAPE CARTRIDGE (page 75)

Discusses the tape cartridge and the tape control keys.

Section 5: SPECIAL FUNCTIONS (page 97)

Shows how to define, run, and record your own functions using the special function keys.

Appendix 1: INSTALLATION PROCEDURE (page 103)

Lists initial turn-on and inspection procedures which should be done immediately after you receive the calculator.

Appendix 2: 9815S SUPPLEMENTAL INFORMATION (page 107)

Explains the minor operating differences between the 9815S and the 9815A.

Appendix 3: SERVICE (page 111)

Describes how to test the calculator and lists periodic maintenance procedures.

Appendix 4: OPERATING LIMITS (page 121)

Lists the calculating range, accuracy, and environmental limits.

TABLE OF CONTENTS

INTRODUCTION

Keyboard Foldout	viii
Your Calculator	ix
Calculator Peripherals	xi
Special Interfaces	xiii
Prerecorded Programs	xiv
Keyboard Magazine	xiv

Section 1: KEYBOARD OPERATIONS

Power On	1
The Keyboard	2
Entering and Clearing Numbers	3
Large and Small Numbers	3
Clearing	4
Simple Arithmetic	4
Serial Arithmetic	5
Last X	6
The Operational Stack	6
Automatic Enter	8
Number Formats	10
Fixed Format	10
Scientific Formats	11
Number Storage	12
Clearing Data Registers	13
Storage and Recall Examples	13
Register Arithmetic	14
Indirect Storage and Recall	15
Indirect Register Arithmetic	16
Additional Data Storage	16
Clearing Numbered Registers	18
Scientific Functions	18
Angular Units	18
Trigonometric Functions	19
Logs and Exponential Functions	20
More Functions of X	21
Conversions	22
Complex Arithmetic	24
Statistical Functions	25

Section 2: PRINTER CONTROL

Printer Functions	27
Manual Paper Advance	28
Printer Paper	28
Printing Alpha	28
The CALL-ALPHA Key	28
Alpha Control Keys	29
Cancelling the Alpha Mode	31

Section 3: PROGRAMMING

Introduction	33
Program Memory	34
Clearing the Memory	34
Program Controls	35
The Program Counter	35
Entering a Program	36
Printout During Program Entry	37
Display During Program Entry	37
Running a Program	37
Example Programs	38
Flowcharting	40
Area of a Polygon	41
Labels	45
Branching	47
IF Instructions	50
Subroutines	53
Nesting Subroutines	54
Example Program	55
FOR-NEXT Loops	57
Nesting FOR-NEXT Loops	60
Flags	61
Editing Programs	63
Program Listing	63
Single Stepping	64
Changing Program Instructions	64
Deleting Instructions	65
Inserting Instructions	66
Clearing the Memory	67
Editing Alpha	67
Programming Hints	69
An Editing Tip	69
Branching Speed	69
Branching Errors	70
Additional Alpha Characters	71
Peripheral CALL Instructions	72

Section 4: TAPE CARTRIDGE

Introduction	75
Tape Instructions	76
The Tape Cartridge	77
Inserting the Cartridge	78
Storing Tapes	78
Cleaning the Tape Head	78
Error Detection	79
Initializing the Tape	80
Tape File Structure	80
Mark Tape	80
Marking a Used Tape	82
Identify File	82

Record and Load Programs	83
Recording Programs	83
Loading Programs	84
Segmented Programs	85
Linking Programs	88
Record and Load Data	88
Record Data	88
Load Data	90
Additional Operations	90
Verify	90
Secure Programs	91
The Autostart Feature	92
Operating Hints	92
Halting the Tape Drive	92
Tape Storage Capacity	92
Master Recordings	94
Re-Marking Used Tapes	94
Recording Special Functions	95

Section 5: SPECIAL FUNCTIONS

Defining Functions	97
Running Functions	98
Key Overlays	99
Nesting Special Functions	100
A Program Directory	100

Appendix 1: INSTALLATION PROCEDURE

Inspection Procedure	103
Equipment Supplied	104
Power Requirements	105
Grounding Requirements	105
Fuses	105
Initial Turn-On	106

Appendix 2: 9815S SUPPLEMENTAL INFORMATION

Introduction	107
Programs	107
Program Storage	107
Branching and Subroutines	108
Editing	108
Data	109
Standard Data Storage	109
Extended Data Storage	109
Split Register Storage	110

Appendix 3: SERVICE

Service Contracts	111
Calculator Tests	111
Cleaning the Calculator	116
Cleaning the Tape Head	116
Loading Paper	117
Peripheral Tests	118
Example Programs	118

Appendix 4: OPERATING LIMITS

Calculating Range	121
Error Messages	121
Accuracy Limits	121
Guard Digits	121
Environmental Limits	122
Tape Specifications	122

Key Index	123
Subject Index	127
Error Messages	(inside-back cover)

FIGURES

Keyboard Foldout	viii
HP 9862A Plotter	xi
HP 9871A Printer	xi
HP 9866A Thermal Printer	xi
HP 9863A Tape Reader	xii
HP 9883A Tape Reader	xii
HP 9884A Tape Punch	xii
HP 9864A Digitizer	xii
Special Interface Applications	xiii
The HP 9815A Keyboard	2
Memory Storage Sketches	16, 17
Alpha Mode Keyboard	28
Flowcharting Symbols	40
Program Form	42
Inserting a Cartridge	75
The HP Tape Cartridge	77
Cleaning the Tape Head	78
Tape File Structure	80
Power Cords	104
The Calculator Fuse	106
Switch Settings for Nominal Powerline Voltages	106
Loading Paper	117
Removing the Printer Window	117

Introduction

The next few pages introduce you first to the HP 9815A Calculator and then to the many accessories for building a complete calculating system. The important features on the calculator keyboard are shown on the facing page.

If you have just received your calculator, or if there's any doubt concerning whether it is set up to operate in your area, please refer to Appendix 1 of this book.

Your Calculator

Arithmetic Operations

Your calculator has four temporary storage locations (working registers) arranged like this ♦

T
Z
Y
X

It's called the "operational stack". Simple arithmetic is done by placing numbers in X and Y and pressing an arithmetic key; the result is left in X. The method is called Reverse Polish (Lukasiewicz) Notation (RPN) and is extremely popular for fast, efficient operation. Section 1, Keyboard Operations, describes how to use the stack and RPN.

The Display and Printer

The 16-character display shows you each number keyed in and each arithmetic result; and you can print the current number in the display whenever you wish. In addition, the display and printer are valuable programming aids. Section 2, Printer Control, lists the many printer functions and shows how to print messages.

Range and Accuracy

The calculating range is from $-9.9999999999 \times 10^{99}$ through $9.9999999999 \times 10^{99}$. Whenever a result exceeds that range the printed message "OVERFLOW" appears. (See the inside-back cover for a complete list of error messages.)

All calculations are computed to 12 places, but the accuracy depends upon the operation. Arithmetic operations (+, −, ×, and ÷) are accurate to within one count in the 12th (least significant) digit.

The Memory

In addition to the four working registers just mentioned, the basic calculator has 10 permanent data-storage registers and a 472-step program memory. A small portion of the program memory is assigned to data storage when the calculator is switched on. (You can easily set up more registers when they are needed).

The program memory can be expanded to 2008 program steps by ordering the calculator with option 001. This option can be added at a later date. The 9815S has a program memory of 3800 steps. There is a field installation kit available for upgrading a 9815A to a 9815S.

Programming

As explained in Section 3, your calculator is completely programmable — that is, you can program all of the keyboard operations, the printer control, the tape-cartridge control, plus a full set of program-control functions.

Tape Cartridge

The miniature tape cartridge used in your calculator can hold a surprising amount of programs and data. As described in Section 4, you can set up each cartridge to fit your needs, or you can load the calculator using the many pre-recorded programs from HP.

Peripheral Control

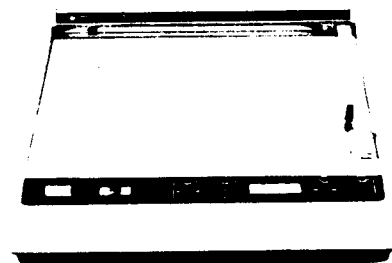
If your calculator has option 002 (two-channel I/O) it can control at least two external peripherals, like those shown on the following pages. Option 002 also enables the calculator to use any two of the special interfaces described on page xiv for controlling measurement devices or special systems. Option 002 can be installed at any time. Two-channel I/O is standard on the 9815S.

Calculator Peripherals

Each of these peripherals is available with all needed interface cables and complete operating instructions. Each peripheral also provides your calculator with a unique set of key sequences, or "call" instructions, enabling full control of the peripheral.

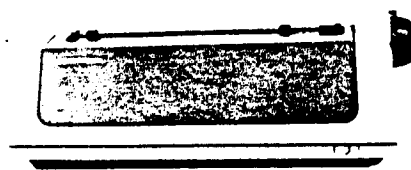
HP 9862A X-Y Plotter

Histograms, pie charts, circuit diagrams, linear, log-log, and polar plots — these are some of the things you can do with the HP 9862A Plotter and your calculator. The plotter can automatically scale your data, label each plot in words and numbers, and draw axes with labels and tic marks. A special feature also allows you to digitize graphical data and store the x-y coordinate points in the calculator.



Output Printers

If you need data output in tables, charts, or on standard forms — almost any format — your calculator and one of these printers will do the job.



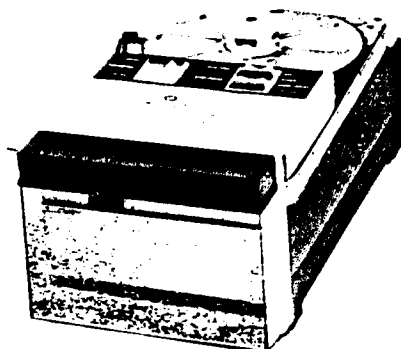
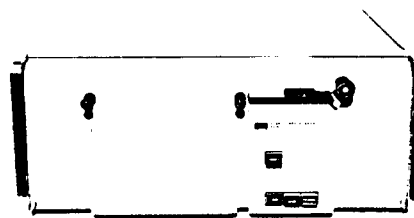
The HP 9871A Output Printer provides up to 15-inch wide, multiple-copy output, uses a full 96-character font, and prints at an average speed of 30 characters per second. The calculator can control character and line spacing, platten motion, tabbing and form feed. A full set of plotting instructions are also available.



The HP 9866A Thermal Printer is a fast (250 lines per minute), page-wide printer. It has a 64-character alphanumeric font and produces fully-formatted text and tables with your calculator. In addition, the HP 9866A can list calculator programs in a convenient, four column format.

Paper Tape Readers

Data from analytical instruments, machine tools, and computer terminals goes directly into your calculator with one of our paper tape readers. The HP 9863A makes it easy to read data in a wide variety of formats at 20 characters per second. Our HP 9883A Tape Reader, designed for high-speed, heavy-volume operations, optically reads tapes at up to 300 character per second. Use the 9883A to load programs, too.

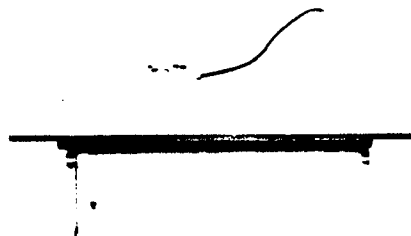


HP 9884A Tape Punch

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

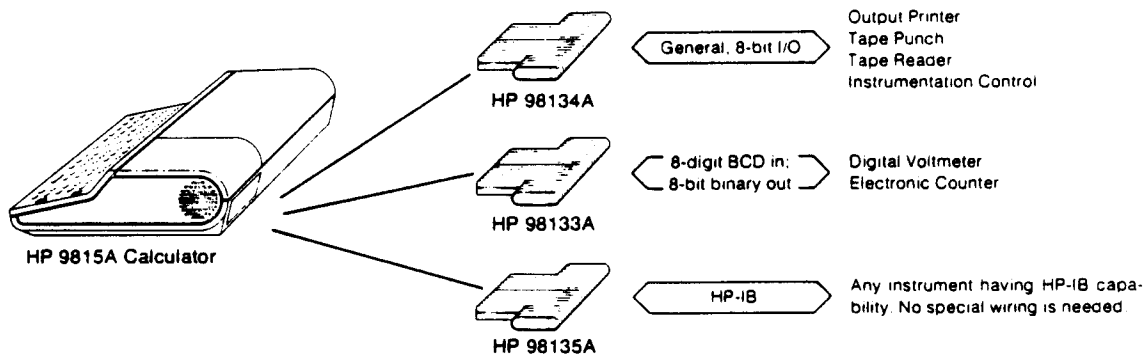
HP 9864A Digitizer

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.



Special Interfaces

The following special interfaces are available for your calculator to control and exchange data with a wide variety of peripheral equipment. Each interface has general installation and operating instructions and provides the calculator with a full set of input/output control instructions. Typical applications are shown here:



Special Interface Applications

General I/O Interface

The HP 98134A Interface is a general-purpose card providing an 8-bit parallel, character-serial interface. The interface transfers data in a "half-duplex" fashion; that is, it can input and output data, but not both at the same time. Buffer-storage for each character is provided, and all lines are compatible with standard TTL levels.

The control sequences provided with the General I/O Interface include: formatted output and free-field data input (standard ASCII coding is used), binary input and output, binary logic operations, formatted program listing, and high-speed program input and output.

BCD Interface

The HP 98133A BCD Interface allows the calculator to take samples and input data reading from one of many instruments having a parallel, binary-coded-decimal (BCD) output. The interface can input data of up to eight BCD digits, and also 8 bits of function, sign, overload, and logic-sense information. Separate 8-bit binary output lines are also available for controlling function, range, etc. on the measurement device.

The control sequences available with the BCD interface are formatted single-sample or split-sample input, two high-speed (burst) sample inputs, and binary output for setting range, function, mode, etc. on the external device. The binary output lines could alternately be used to control a second device. The calculator can input data at the rate of up to 1000 samples per second using a high-speed input instruction.

HP-Interface Bus

The HP 98135A HP-IB Interface permits your calculator to be a controller or the system controller of up to 14 other instruments via the standard HP-IB instrumentation bus. The HP-IB interface is completely prewired, and can be used with any instruments having HP-IB compatibility.

The HP-IB employs a 16-line bus; each instrument on the bus is connected in parallel to all 16 lines. Eight of the lines are used to transmit data and the remaining eight are used for timing and control.

Data is transmitted on the HP-IB as a series of 8-bit characters. Normally, a 7-bit ASCII code is used; the eighth (parity) bit is ignored by the calculator. Data is transferred by means of an interlocked "handshake" technique. This sequence permits asynchronous communication over a wide range of data rates.

For more information on the peripheral devices or interfaces shown on these pages, contact the nearest HP sales and service office.

Prerecorded Programs

Tape cartridges containing programmed solutions to 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 Appendix 3).

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, descriptions 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.

1

Keyboard Operations

This section gives you all the basic instructions needed to solve problems from the HP 9815A keyboard. Later, in Section 3, you'll find that each keyboard operation described here is also programmable.

Power On

Your calculator is shipped fully assembled and ready to operate, but please make these simple checks first:

- Have you just received the calculator? If so, turn to Appendix 1 and follow the turn-on procedure.
- Is the calculator plugged in? If not, plug one end of the power cord into the rear panel; then plug the other end into a suitable power outlet.

Set the two keyboard switches to  and .



Then switch the calculator on .



If the display does not appear as shown here, see Appendix 2, "Service".

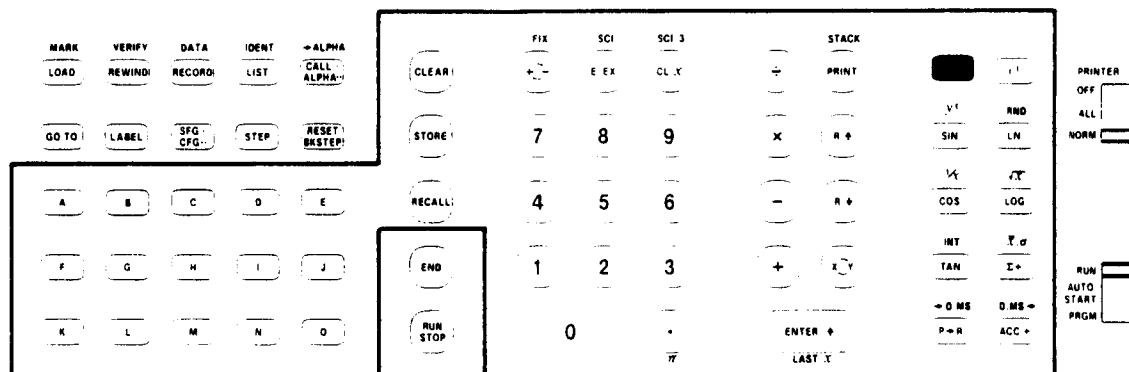
0.00

The Keyboard

A sketch of the keyboard is shown below. All of the keys circled in color are described in this section.

Notice that many of the keys are shown with alternate functions. The  key allows you to do each of those alternate functions. For example, to enter the constant pi (3.1415...), merely press .


Also notice that the functions below keys  through  are not shown here. These are not alternate keyboard functions, but are the programming functions.

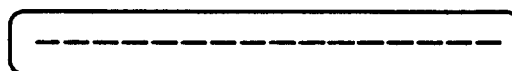


The HP 9815A Keyboard

The switches on the right-hand side of the keyboard control the printer and operating modes. To print each of your keyboard operations automatically, set the Printer switch to ALL. On the other hand, to conserve paper set the switch to NORM or OFF. For now, set the Printer switch to NORM and the Mode switch to RUN.

The calculator prints various messages, regardless of the Printer switch setting. Some messages indicate calculator or peripheral device status, while others tell you of incorrect operations, such as division by zero. A list of error messages is at the back of this manual.

The 16-digit display shows each number you enter and the calculated result. When you wish to print the number currently displayed, just press . Whenever the calculator is either doing a lengthy keyboard operation or running a program, it indicates "I'm busy" by displaying:



Entering & Clearing Numbers

- The calculator has four temporary storage locations arranged as shown on the right.

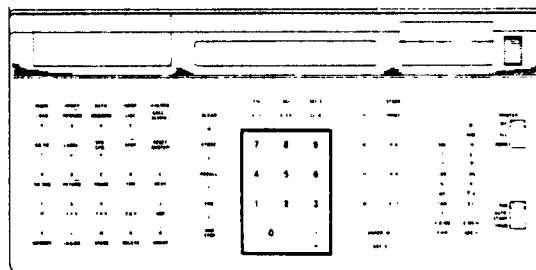
T
Z
Y
X

This is called the "operational stack". Each number keyed in goes into X, the displayed register. To save the number (place it in Y) before keying a second number in, press **ENTER**. To print the current number in X, press **PRINT**.

- To enter a negative number, key in its base value and press **+/-**. Change the sign of the current number in X by pressing **+/-**.
- To key numbers in using scientific notation: key in the base number, press **EE**, and key in the exponent (up to two digits). If the exponent is just one digit, terminate it with a non-numeric key or **.**.
- To clear the stack, press **CLEAR**. To erase just the number in X, press **CLX**.
- To cancel a key sequence which has not been completed, press **RESET**.

The number-entry keys are arranged as on an adding machine. Key in each number from left to right and include a decimal point, if needed. For example, you can key in 98.15 by pressing

9 **8** **.** **1** **5**



Before keying in a second number, save the first one by pressing **ENTER**.

To key in a negative number, press **+/-** after keying in its value. To change the sign of a result, simply press **+/-**.

Large and Small Numbers

You can also enter numbers in scientific notation by using the **EE** (Enter Exponent) key. For instance, to enter 12.5×10^6 , press:

12.5 **EE** **6**

ENTER

12.5 06

12,500,000.00

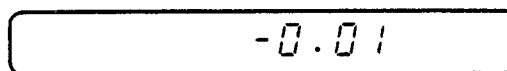
4 Keyboard

And to enter a small negative number, such as -5×10^{-3} , press:

5   3 

















In this case the number you see is rounded to fit the current display format.

Clearing

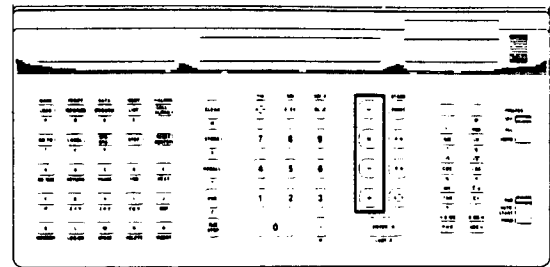
If you make a mistake while keying in a number, you can erase the entire number by pressing  (Clear X). To erase all four working-registers of the operational stack, press .

The  key can be used to clear a key sequence which has not been completed. For example, if you wish to set a scientific number format and press   by mistake, cancel the sequence by pressing . Then press the correct sequence.

Simple Arithmetic

- To do arithmetic problems containing two numbers: key in the first number and save it by pressing . Then key in the second number and press the operator key: +, -, ×, or ÷. The result is left in X.
- To do serial calculations (e.g., a series of arithmetic operations linked together), press  to save the first number; then just key in each additional number—the calculator automatically ENTERS (saves) each number keyed in directly after an operator key.
- The last number in X before an operation is done is automatically stored in LAST X. To re-ENTER that number, press  .

For a problem having two numbers and one arithmetic operation, key in the first number and save it in Y by pressing **ENTER +**; then key in the second number and follow it with the operator key (+, -, ×, or ÷). The result is left in X.



For example, to add 12 and 3:

Press

Result

12 **ENTER +** 3 **+**

15.00

The calculator does simple arithmetic for you like this:

- It adds the last number keyed in to the last number saved. ($Y + X \rightarrow X$)
- It subtracts the last number keyed in from the last number saved. ($Y - X \rightarrow X$)
- It multiplies the last number keyed in by the last number saved. ($Y * X \rightarrow X$)
- It divides the last number saved by the last number keyed in. ($Y/X \rightarrow X$)

Serial Arithmetic

A serial calculation is a series of arithmetic operations linked together into one problem. For example:

$$(((3 \times 5) - 3) \div 6) + 7 = 9$$

This problem is easily solved by keying in the numbers and operations from left to right:

3 **ENTER +** 5 **×**

15.00

3 **-**

12.00

6 **÷**

2.00

7 **+**

9.00

As shown, only the first number must be saved by pressing **ENTER +**. Then each number keyed in after an operator key is automatically ENTERED (saved) for you.

Last X

The last number keyed in before an operator key is pressed is automatically stored for you in a location called "Last X". You can recall and use that number by simply pressing



Recalling Last X also performs an automatic ENTER, just like keying in a number after an arithmetic key. Last X is not cleared by **CLEAR** or **CL X**.

As a simple problem using Last X, multiply each of these numbers by .05:

12 82 45

12 **ENTER** .05 **x**

0.60

82 **ENTER** **x**
LAST X

4.10

45 **ENTER** **x**
LAST X

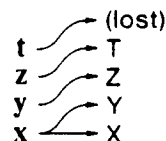
2.25

The Operational Stack

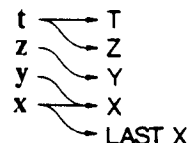
- The four working registers look like this ♦
X is always displayed and can be printed.

T
Z
Y
X

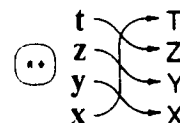
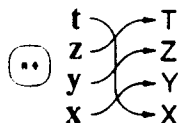
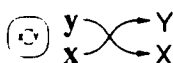
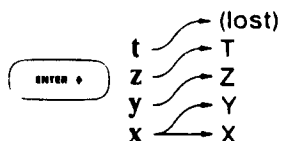
- To clear the stack, press **CLEAR**. To clear only X, press **CL X**.
- Keyed-in numbers go into X. To duplicate X into Y, press **ENTER**. This also moves the Y-value to Z and the Z-value to T (the T-value is lost).



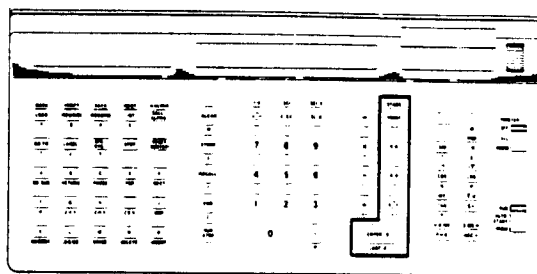
- Pressing an arithmetic key places the result in X and drops the other values down. The former X-value goes to LAST X.



- An automatic ENTER occurs when a number is keyed in following any operator key except **CLEAR**, **CL X**, or **ENTER**.
- To list the numbers in the stack, press **STACK** **PRINT**.
- To move the numbers around in the stack:



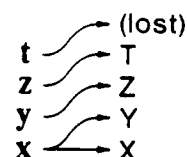
The operations described thus far have used only the bottom half of the stack (X and Y): you simply put the numbers into place, press the required function key, and the result is left in X. Now let's look at the complete stack and its control keys.



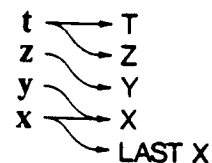
The four working registers are arranged in a "stack", with the X (displayed) register at the bottom:

T
Z
Y
X

You've already seen that each number keyed in goes into X. When you press **ENTER** the number is duplicated in Y. As shown here, pressing **ENTER** also moves the number from Y to Z and moves the number from Z to T (the number in T is lost).



When an arithmetic key is pressed the result is placed in X and the remaining numbers in the stack drop down. The number formerly in X goes to LAST X.



For example, follow the stack contents as we solve: $15/(7 - 4) = ?$

Press

Stack contents at each step



T → 0
Z → 0
Y → 0
X → 0

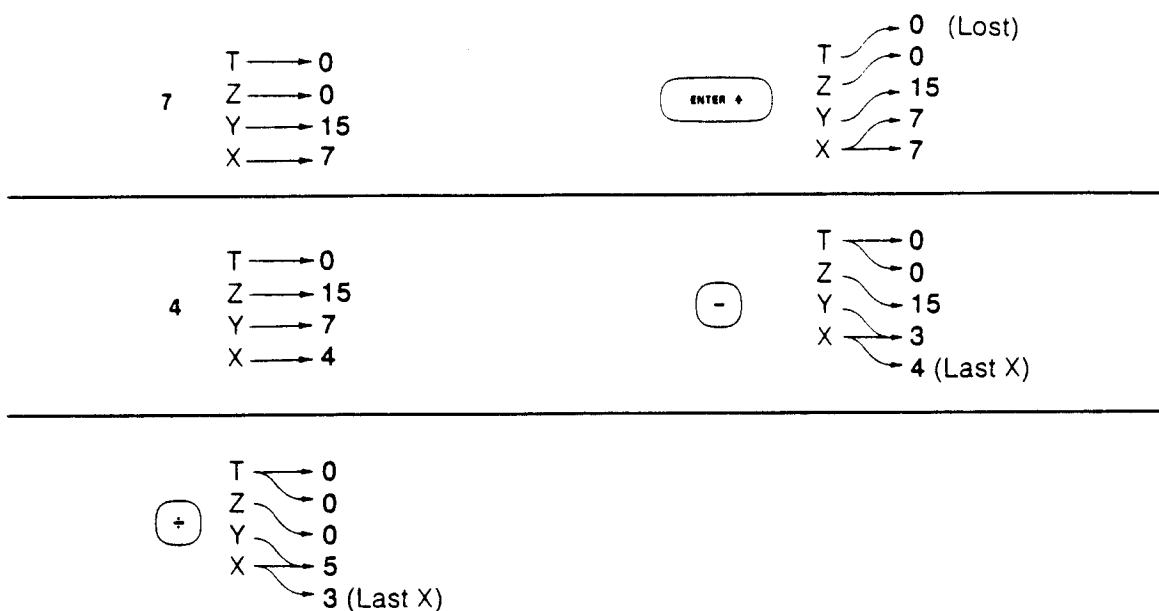
15

T → 0
Z → 0
Y → 0
X → 15



(Lost)
T → 0
Z → 0
Y → 15
X → 15





8 Keyboard



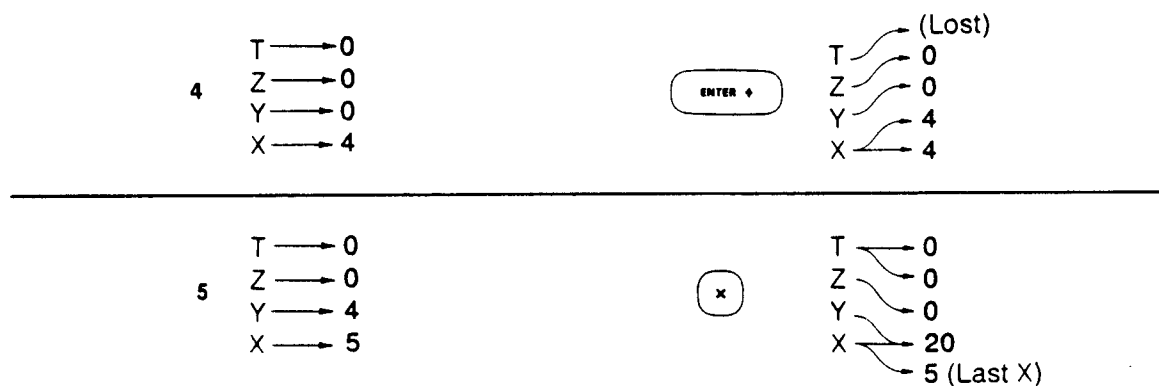
Notice that we entered numbers just as they appear in this problem.

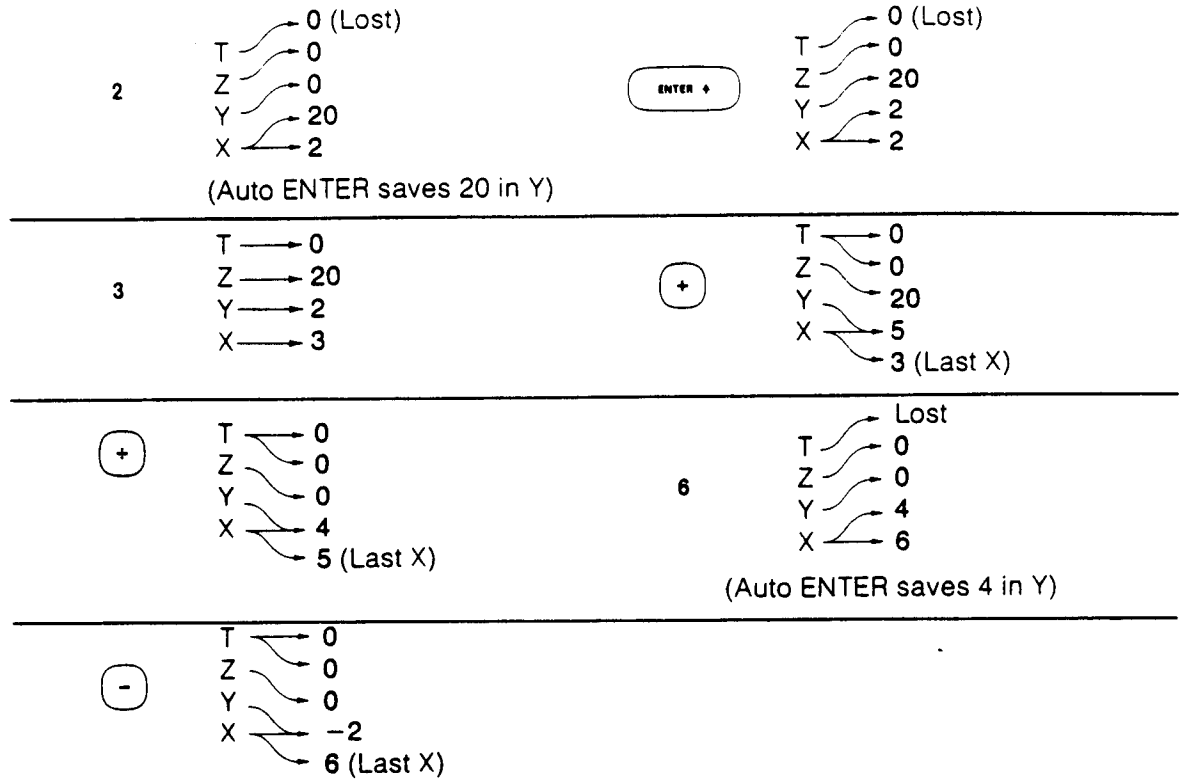
Automatic Enter

As explained earlier, an automatic ENTER is done on each number keyed in directly after an arithmetic key. This always occurs except when the number follows **CLEAR**, **CL X**, or **ENTER +**.

Also, any operation which places a number in X (e.g.,   or  ) causes an automatic ENTER. This feature allows you to easily work serial, chain, and mixed-chain calculations. For example, notice how the automatic ENTER is used in the following problems.

Solve: $((4 \times 5)/(2 + 3)) - 6 = ?$





Now solve: $(12 \times 5) + (11 \times 4) + (10 \times 3) = ?$

12 ENTER 5 ×

60.00

11 ENTER 4 × +

104.00

10 ENTER 3 × +

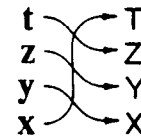
134.00

Here are the other stack control keys:

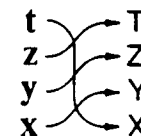
↕ Exchanges the number in X and Y, without affecting Z or T.



↓ Rolls each number in the stack down to the next register.



↑ Rolls each number in the stack up to the next register.




 PRINT Lists the contents of the stack.


(T)	0.00
(Z)	104.00
(Y)	10.00
(X)	3.00

Number Formats


- **FIX**

When the calculator is switched on, the FIX format is automatically set, with a two-digit fraction shown. To set another FIX format, press  (n) ; n can be any number-entry key and specifies the number of decimal places. When a number is too large for the currently-set FIX format, the SCI format is used.

- **SCI**

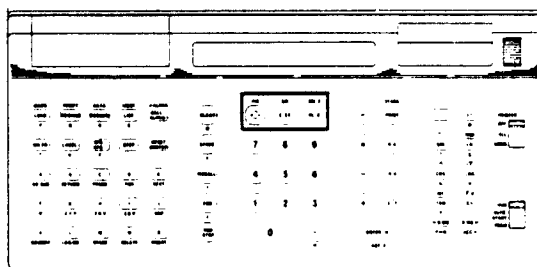
Numbers expressed in standard scientific notation appear as a base number multiplied by a power of 10 (exponent). To set a SCI format, press  (n) ; n specifies the number of decimal places shown.

- **SCI 3**


When a SCI 3 format is set, each number appears in scientific notation, but with an exponent which is a multiple of three (e.g., 3, 6, 9,...). To set a SCI 3 format, press  (n) ;

- Regardless of the format in use, the least-significant digit of each number shown is always rounded. This does not affect internal calculations, but only displayed and printed numbers.

Printed and displayed numbers normally appear in a fixed decimal-point format called "FIX 2". This means that each number is rounded to the nearest hundredth before it's displayed or printed. The number format does not affect calculating accuracy, but only displayed and printed numbers.



Fixed Format

To set another fixed format, press  and any numeric key (0 through 9); the number key indicates how many decimal places to display. For example to see 12.34567 in various fixed formats:

12.34567



FIX 0

12.34567

12



FIX 4

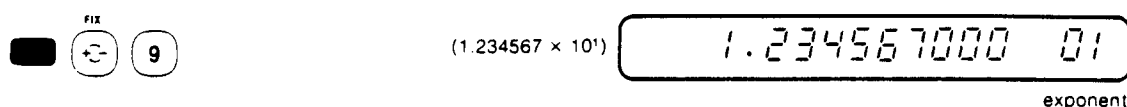
12.3457



FIX 6

12.345670

When a keyed-in number or result is too large for the currently-set format, a scientific format is automatically used.





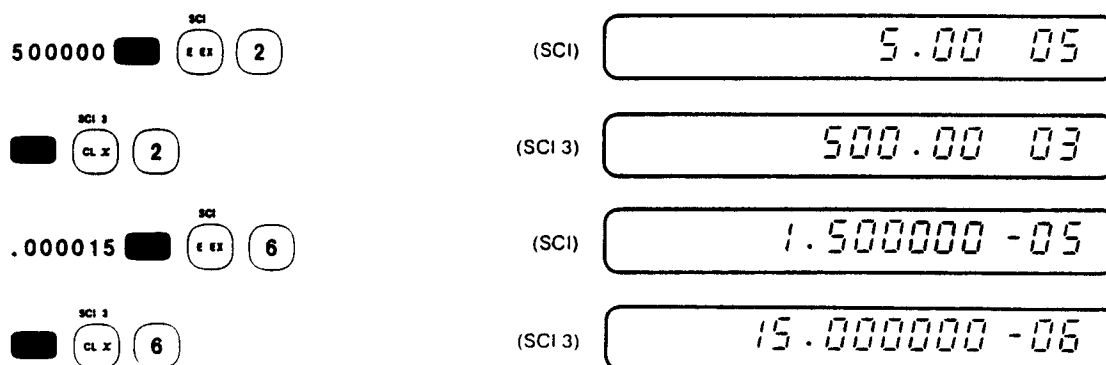
But when a number is too small to be shown in the display, the format does not change. (Of course, the number is still there!)



Scientific Formats

Scientific notation is a shorthand way of expressing very large and very small numbers. It uses a base number which is multiplied by a power of 10. For example 5,000,000 (five million) could be written more easily as 5×10^6 , and .00025 as 2.5×10^{-4} .

Two different scientific formats are available: "standard scientific" (SCI) and "scientific three" (SCI 3). The standard scientific format is set by pressing  and a number key from 0 to 9. The number key indicates how many decimal places are shown. The SCI 3 format is set by pressing  and a number key. When SCI 3 is set, each number has an exponent which is a multiple of three (e.g., 3, 6, 9, ...). Here are some examples:



Remember that the number format affects **only** displayed and printed numbers - all internal calculations are performed to 12 places, and up to 10 most-significant digits are shown.¹

If you wish to actually round the number in X, use the ROUND function explained later.

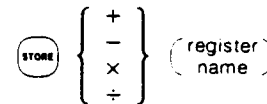
¹For more information, see "Guard Digits" in Appendix 4.

Number Storage

- 20 storage registers, named A through J and 000 through 009, are available when the calculator is switched on. More (or less) numbered registers can be assigned when needed.
- To store the number from X into a register, press **STORE** **(register name)**.
- To recall the number from a register to X, press **RECALL** **(register name)**.

Each number recalled is automatically ENTERED, except after **ENTER**, **CLEAR**, or **CL X**.

- To clear registers A through J, press **STORE** **CLEAR**.
- To do an arithmetic operation directly between X and a register, use this sequence ♦



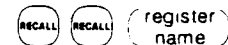
The result is placed in the storage register and X is left unchanged.

- To store or recall numbers indirectly:

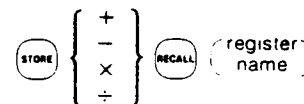
Store X into register indicated by number in specified register:



Recall contents of register indicated by number in specified register:



- To do indirect operations between X and a storage register, use this sequence ♦

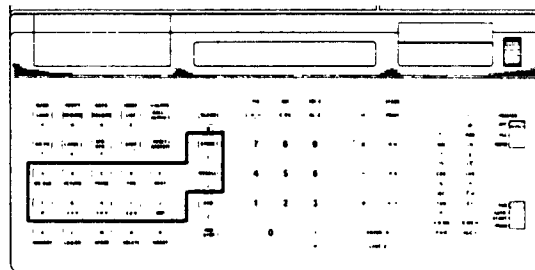


The operation is performed on the register indicated by the number in the specified register. The result is placed in the storage register and X is left unchanged.

- To delete register 000 through 009 and add 80 steps of program memory, press ♦
- To assign more numbered storage registers, place the *total* number of register needed into X and press ♦



The HP 9815A provides you with 20 storage registers for holding constants or intermediate results. Each register can hold one number and is accessed by using the **STORE** and **RECALL** keys, in conjunction with a key indicating the register name: "A" through "J" and "000" through "009". As explained later, more (or less) data registers can easily be assigned when they are needed.



Clearing Data Registers

All data registers are cleared (zeroed) when the calculator is switched on. To clear registers A through J, without affecting anything else, press **STORE** **CLEAR**. To clear numbered registers, either store 0 in each register or use the routine suggested on page 18.

Storage and Recall Examples

First clear all storage registers by switching the calculator off and back on. Then store the constant 4.75 in A:

4.75 **STORE** **A**

4.75

Now use the constant in this problem:

$$8.5 \times 4.75 = ?$$

8.5 **RECALL** **A** **x**

40.38

The example shows that the current number in X is automatically ENTERED (saved in Y) when a number is recalled.

When specifying numbered storage registers, key in the significant digits of the register name, followed by either a non-numeric key or **.**. Or, you can key in the entire three-digit name. Here are more examples.

NOTE

The error message "ILLEGAL ADDRESS" appears when a register number which is not assigned is specified.

First, store another constant (3.66) in register 009:

3.66 **STORE** **9** **.**

3.66

Now use both stored constants (4.75 → A and 3.66 → register 9) to solve:

$$\frac{8 \times 3.66}{4.75} = ?$$

8 **RECALL** 9 **×**

29.28

RECALL A **÷**

6.16

NOTE

If the **STORE** or **RECALL** key is followed directly by an operator key, the calculator will automatically use register 000. To cancel a STORE or RECALL operation which hasn't been completed, press **RESET** **ONSTEP**.

Register Arithmetic

Arithmetic operations can be done directly between the number in X and a storage register. The result is placed in the storage register while the number in X is left unchanged. Here is the general key sequence:

STORE { +
-
×
÷ } (register name)

For example, first store 8 in register C:

8 **STORE** C

8.00

and then increment it by 2:

2 **STORE** + C

2.00

RECALL C

(confirms that C equals 10)

10.00

Now, subtract 4 from C:

4 **STORE** **-** **C**

4.00

RECALL **C**

(confirms that C now equals 6)

6.00

Indirect Storage and Recall

You can do store and recall operations **indirectly** by specifying a register name which contains the number of the actual register to be used. These operations are performed just like direct store and recall, except that **RECALL** is pressed before the intermediate register is specified. Here are the general key sequences:

STORE **RECALL** (register name)

Stores X into the register indicated by the number in the specified register.

RECALL **RECALL** (register name)

Recalls the contents of the register indicated by the number in the specified register.

For example, first store 3 in register B:

3 **STORE** **B**

3.00

Now indirectly store 25 in register 003:

25 **STORE** **RECALL** **B**

25.00

Now clear X and verify the contents of register 003:

CL X

0.00

RECALL **RECALL** **B**

(contents of register 003)

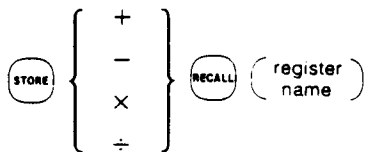
25.00

NOTE

The absolute integer value in the intermediate register is used as the indirect register number; so the sign and any fractional part of the value are ignored.

Indirect Register Arithmetic

The register arithmetic and indirect storage operations just described can be combined. Here is the general key sequence:



In each case, the arithmetic operation is performed on X and the contents of the register indicated by the number in the register named. The result is placed in the storage register and X is left unchanged.

For example, to add 2 to the register indicated by the number in register E:



And to subtract 6 from the register indicated by the number in register 003:

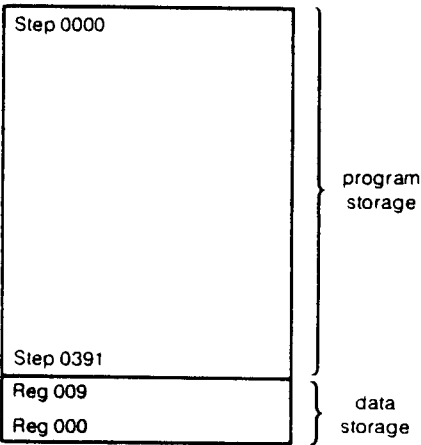


As with the indirect storage operations, you'll probably find indirect register arithmetic most often used in programs, to increment either data values or event counters.

Additional Data Storage

When you require more than the 10 numbered storage registers available when the calculator is switched on, you can easily assign up to 58 numbered registers.¹ This is done by assigning more of the program memory as data storage.

When the calculator is switched on, it automatically assigns 80 program steps for data storage and identifies the area as registers 000 through 009. Here is a simplified sketch of that memory ↴

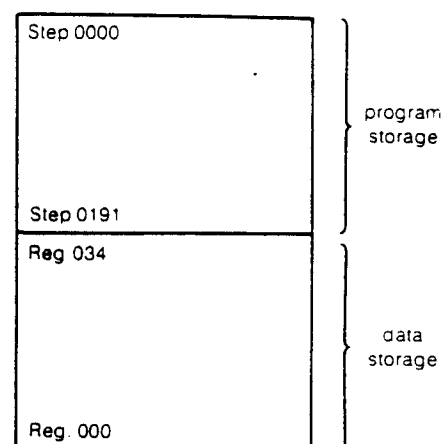


¹Up to 250 numbered registers (000 through 249) can be assigned when option 001 is installed. The 9815S can directly access up to 256 numbered registers.

If you need, say, 25 more numbered registers (i.e., 35 in all), they can be assigned by pressing:

35  

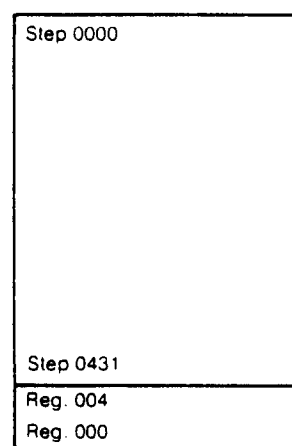
Here's the new memory configuration ♦
Notice that the program memory lost 8 steps for each data register assigned.



Additional storage registers remain assigned until you either change the assignment or switch the calculator off. For example, if most of those additional registers were no longer needed and you wished to load a program requiring most of the memory, you could change the assignment to, say, 5 numbered registers (you still have registers A through J) by pressing:

5  

Notice that registers 005 through 034 are deleted and the memory gains 240 steps, but the contents of registers 000 through 004 remain unchanged.







If there is not enough program memory available to accept any given register assignment, the printer will indicate "MEMORY OVERFLOW" and the assignment will be ignored.

Clearing Numbered Registers

The last example hints at an easy way to clear a block of numbered storage registers: first delete the block of registers to be cleared; then reassign them.

For example, assume that 25 registers are assigned and you wish to clear registers 010 through 024. Press this sequence:

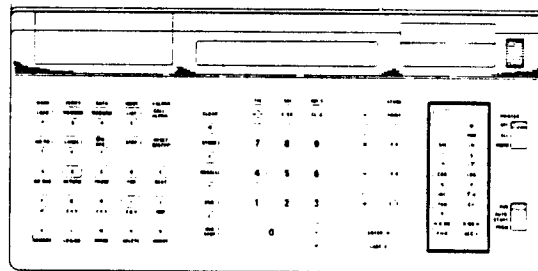
10   (Assign registers 000 through 009)







25   (Assign registers 000 through 024)

Now registers 010 through 024 are clear, while the data in registers 000 through 009 remain unchanged!

Scientific Functions

In addition to four-function arithmetic, there are 24 scientific functions on the keyboard. All of these functions are grouped into one key-block, as shown on the right. Most functions operate on the current number in the X register.





Each scientific function key has primary, secondary, and inverse functions. To perform the primary function, merely press the appropriate key. To do the secondary function, prefix the appropriate key with  (e.g., press   to find the reciprocal of X). To use the inverse function, prefix the key with  (e.g., pressing   takes the arc cosine of X).


Angular Units

Operations involving angles, such as trig functions and angular conversions, can be done by expressing the angles in any of three units: decimal degrees, radians, or metric grads (360 degrees = 2π radians = 400 grads). Degrees is automatically set when the calculator is switched on.

Here's how to specify the desired units:










 **2** to specify radians.

 **3** to specify grads.

 **1** to specify degrees.

Trigonometric Functions

The trig functions are summarized in the next table. You are invited to familiarize yourself with the functions by working the example problems which follow.

Function ¹	Keys	Result
Sine		Sine X → X
Arc sine	 	Arc sine X → X (principal value)
Cosine		Cosine X → X
Arc cosine	 	Arc cosine X → X (principal value)
Tangent		Tangent X → X
Arc tangent	 	Arc tangent X → X (principal value)

¹Be sure to set the desired units before working any trig problems.


Find the sine of 30°.



After switching the calculator on, press:

30 

0.50

Now find the arc cosine of -1, in radians:

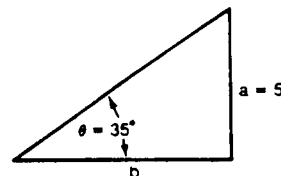
 **2** (set radians units)

1  

3.14



Find the area of this right triangle ♦

Since: $\text{Area} = \frac{ab}{2}$ and $b = \frac{a}{\tan \theta}$



We can use the formula: $A = \frac{a^2}{2 \tan \theta}$

 **1** (set degrees units)

5  

35  2 



a^2

25.00

$2 \tan \theta$

1.40

Area

17.85

Logs and Exponential Functions

Function	Keys	Results
Natural Log ($\log e$)	$\boxed{\text{LN}}$	$\ln X \rightarrow X$
Natural Antilog (e^x)	$\boxed{1^x} \boxed{\text{LN}}$	$e^x \rightarrow X$
Common Log (\log_{10})	$\boxed{\text{LOG}}$	$\log X \rightarrow X$
Common Antilog (10^x)	$\boxed{1^x} \boxed{\text{LOG}}$	$10^x \rightarrow X$
Exponential (Y^x)	$\boxed{\text{Y}^x} \boxed{\text{SN}}$	$Y^x \rightarrow X^*$

*The number in Y is lost after the operation.

Display the constant e to nine places.

$\boxed{\text{FIX}} \boxed{\text{9}} \boxed{1} \boxed{1^x} \boxed{\text{LN}}$

2.718281828

Now find $\sqrt[5]{32}$ using natural logs:

32 $\boxed{\text{LN}}$ $\boxed{\text{ENTER} \rightarrow}$

3.465735903

5 $\boxed{\div}$

0.693147181

$\boxed{1^x} \boxed{\text{LN}}$

2.000000000

If you invest \$1000 at a 10% interest rate per period, how much would you have after six periods? Using the Y^x function, evaluate the following:

$$\text{principal} \times (1 + \text{interest rate})^{\text{Periods}} = 1000 \times (1.10)^6 = ?$$

$\boxed{\text{FIX}} \boxed{\text{2}}$

1.10 $\boxed{\text{ENTER} \rightarrow}$

6 $\boxed{\text{Y}^x} \boxed{\text{SN}}$

(1.10⁶)

1.77

1000 $\boxed{\times}$

(principal & interest)

1,771.56

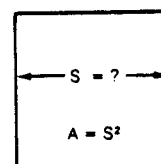
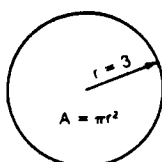
More Functions of X

As with the trig and log functions, each of these functions places the result in the X register.

Function	Keys	Result
Square root	\sqrt{x} LOG	$\sqrt{X} \rightarrow X$
Reciprocal	$\frac{1}{x}$ COS	$1/X \rightarrow X$
Integer	INT TAN	Integer $X \rightarrow X$
Round	RND LN	Y rounded to X places $\rightarrow X^*$
Pi	π	$3.14159265360 \rightarrow X$

*The original numbers in X & Y are lost after this operation.

What size square has the same area as a circle whose radius is 3? (Hint: the square root of a circle's area gives the side of a square having an equivalent area.)



3 ENTER \rightarrow X

r^2

π X

πr^2

\sqrt{x} LOG

S

Calculate $\frac{1}{1/5 + 1/7}$ and display the result to four places.

$\frac{1}{x}$ 4

$1/5$

5 $\frac{1}{x}$ COS

$1/7$

7 $\frac{1}{x}$ COS +

$\frac{1}{x}$ COS

Round 5840 to the nearest 100 (10^2).

5840  2 

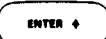
5,800.0000

Now round the answer to 10^4 .

4 

10,000.0000

To round a fractional value, enter a negative number into X. For instance, round 5.5555 to the nearest thousandth (10^{-3}).

5.5555 

5.5555

3 

5.5560

Now find the integer portion of the result.

5.0000

The INTEGER function can also be used with negative numbers.

5.6  

-5.0000

Conversions

Keyboard functions are available to convert rectangular (x,y) coordinates to their equivalent polar form (r = magnitude, θ = angle). Another conversion function allows you to convert decimal angles to degrees, minutes and seconds. Each function is listed on the next page.

Function	Input Values	Keys	Results
Polar \rightarrow Rectangular	$r \rightarrow X$ $\theta \rightarrow Y^1$	$\boxed{P \rightarrow R}$	x coordinate $\rightarrow X$ y coordinate $\rightarrow Y$
Rectangular \rightarrow Polar	x coordinate $\rightarrow X$ y coordinate $\rightarrow Y$	$\boxed{r^1}$ $\boxed{P \rightarrow R}$	$r \rightarrow X$ (magnitude) $\theta \rightarrow Y$ (angle) ¹
Decimal Angle \rightarrow D.MS ³	Decimal $\theta^2 \rightarrow X$	$\boxed{\rightarrow 0.MS}$ $\boxed{P \rightarrow R}$	D.MS ³ $\rightarrow X$
DMS \rightarrow Decimal Angle	D.MS ³ $\rightarrow X$	$\boxed{\rightarrow 0.MS}$ $\boxed{ACC \rightarrow}$	Decimal $\theta \rightarrow X^1$

¹Angles are entered or returned in currently-set units.

²Angle is entered in currently-set units; the result is in degrees.

³The format is DDDDD.MMSS, where D = degrees, M = minutes and S = seconds. An error message results for decimal angles above 99999.99999 decimal degrees (or equivalent in radians or grads) or 99999.59599 in D.MS format.

Convert the rectangular coordinates 4, 3 (x,y) to polar form, with the angle expressed in degrees.

$\boxed{\rightarrow 0.MS}$ $\boxed{1}$ (set degrees)

$\boxed{\rightarrow 0.MS}$ \boxed{FIX} $\boxed{2}$ (set FIX 2 format)

3 $\boxed{ENTER \rightarrow}$ 4 $\boxed{r^1}$ $\boxed{P \rightarrow R}$

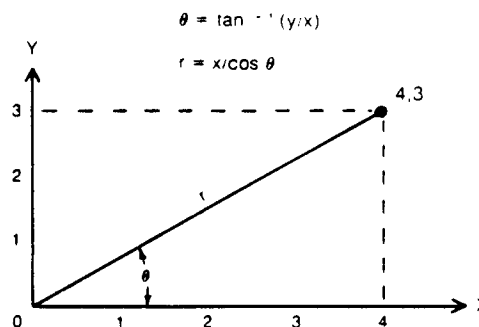
$\boxed{\rightarrow 0.MS}$

r (magnitude)

5.00

θ (angle)

36.87



Convert the polar coordinates $r = 8$, $\theta = 120^\circ$ to rectangular form.

120 $\boxed{ENTER \rightarrow}$ 8 $\boxed{P \rightarrow R}$

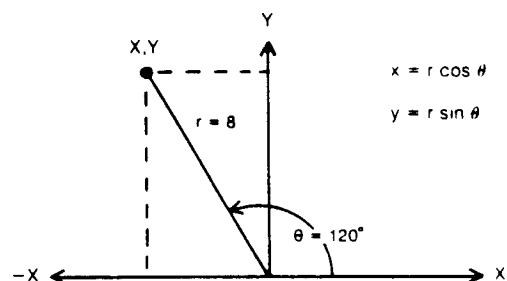
$\boxed{\rightarrow 0.MS}$

x coordinate

-4.00

y coordinate

6.93



Convert $\pi/7$ radians to degrees, minutes and seconds form.

2

(set radians)

•

7

÷

π

↔ D.MS

P→R

25° 42' 51"

0.45

25.4251

Notice that the "FIX 4" format is automatically set to display the D.MS form.

Now add these two angles: 15° 10' 47" and 7° 49' 33". They must first be converted to decimal degrees before adding, and then converting back to D.MS form.

1

(set degrees)

15.1047

↔ D.MS

ACC +

7.4933

↔ D.MS

ACC +

+

(decimal degrees)

23.0056

↔ D.MS

P→R

(23° 00' 20")

23.0020

Complex Arithmetic

The

ACC +

 key allows you to work complex arithmetic using two variables. The

ACC +

 key operates using (simultaneously) register sets X & I and Y & J. Here are the functions:

Function	Keys	Results
Accumulate +	<div>ACC +</div>	I + X → I J + Y → J
Accumulate -	<div>r1</div> <div>ACC +</div>	I - X → I J - Y → J
Recall Total	<div>RECALL</div> <div>ACC +</div>	I → X (Totals are J → Y automatically ENTERED)

Solve this example problem:

$$(2x + 3y) + (4x + 5y) - (3x - 6y)$$

PRINTER
OFF
ALL
NONE

Set

STORE

CLEAR

(registers I & J must be initially cleared)

FIX

2

3

ENTER +

2

ACC +

(store first term)

5

ENTER +

4

ACC +

(add second term)

6

ENTER +

3

ACC +

(add third term)

RECALL

ACC +

PRINT

PRINT

CLRA+J
FIX 2
3.00
ENTER↑
2.00
ACC+
5.00
ENTER↑
4.00
ACC+
-6.00
ENTER↑
3.00
ACC-
R ACC+
PRINT
(x)
3.00
X≠Y
PRINT
(y)
14.00

Statistical Functions

Two important stat functions are available on the keyboard. They use storage registers C, D, and E; so be sure to clear the registers before doing a stat operation.

Function	Keys	Results
Summation	<div>Σ+</div>	<div>X + C → C</div> <div>X² + D → D</div> <div>number of entries → E</div>
Delete Entry	<div><div>I¹</div><div>Σ+</div></div>	<div>C - X → C</div> <div>D - X² → D</div> <div>number of entries - 1 → E</div>
Basic Stat	<div><div></div><div><div>Σ,σ</div><div>Σ+</div></div></div>	<div>mean (\bar{x}) → X</div> <div>std. deviation (σ) → Y</div>

Find the average (mean) of these 10 numbers:

62 84 47 58 68 60 62 59 71 73

Set

PRINTER
OFF ☐
ALL ☐
NORM ☐

STORE

CLEAR

62

$\Sigma+$

 84

$\Sigma+$

47

$\Sigma+$

 58

$\Sigma+$

68

$\Sigma+$

 60

$\Sigma+$

62

$\Sigma+$

 59

$\Sigma+$

71

$\Sigma+$

 73

$\Sigma+$

\bar{X}, σ

$\Sigma+$

PRINT

$\times \div$

PRINT

CLRA+J
SUM+ 62.00
SUM+ 84.00
SUM+ 47.00
SUM+ 58.00
SUM+ 68.00
SUM+ 60.00
SUM+ 62.00
SUM+ 59.00
SUM+ 71.00
SUM+ 73.00
SUM+
MN&SD
PRINT
mean (x) 64.40
X \neq Y
PRINT
standard deviation (σ) 10.10

Since we have not cleared the storage registers yet, we can add more data items, say 49 and 73, and find the new mean.

49

$\Sigma+$

 7.

$\Sigma+$

\bar{X}, σ

$\Sigma+$

(new mean)

58.33

That new mean seems too low; let's check the total number of entries.

RECALL

E

(n is correct)

12.00

Oops! That last entry should have been 73, not 7. Now delete the wrong entry, add the correct one, and compute basic stat again.

7

\div

$\Sigma+$

73

$\Sigma+$

\bar{X}, σ


$\Sigma+$

(correct mean)














63.83

2

Printer Control










The preceding section showed you how the built-in printer complements the display, allowing you to record the current number in X by pressing . Here's a brief review of all the printer functions already described:

Printer Functions

Function	Keys	Result	For more details see:
Print X		Print number in X.	
Set printer (and display) format	     	Numbers appear in FIXED (n) decimal point format. Standard scientific format is set. Special scientific format is set.	page 10
Print stack	 	Print contents of the stack.	page 9
Print all	   	Print each keyboard operation.	page 119
Error messages	—	Automatic printout of keyboard errors and status conditions.	(inside back cover)

In addition to all of those functions, the printer gives you alphanumeric capability and valuable programming aids. Each of the programming functions is described further in Section 3.

Additional Printer Functions

Function	Operation	Result
Print alphanumeric labels and messages	  alpha keys 	Alpha message printed.
Print a number within the alpha message		Number in X included in alpha message.
Program entry	   	List program steps as they are keyed in.
List program		List program memory.

Manual Paper Advance



Use the thumb wheel next to the printer to manually advance the printer paper. Paper advance is also controlled when printing alpha, as explained next, and by a programmable instruction called SPACE.




Printer Paper

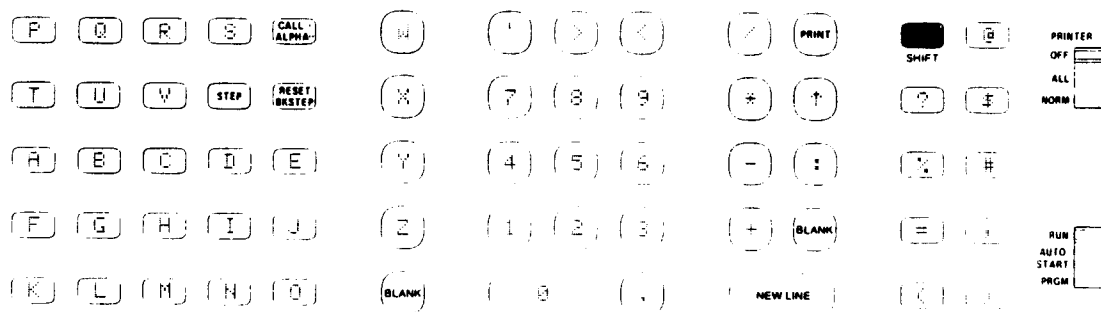
Three rolls of thermal printing paper are supplied with each calculator. The message "OUT OF PAPER" appears just before the printer runs out of paper. To load another roll just remove the old paper core and then follow the diagram under the calculator top-cover. A complete procedure is in Appendix 3.

Printing Alpha

The Key

The Alpha mode is accessed by using the  key. This key is also used to access control of external devices, such as a plotter. As shown on the key, pressing  once "calls" an external device, while pressing the key twice sets the Alpha mode.

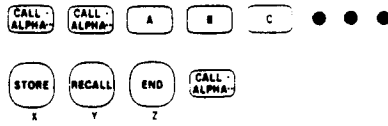
After   has been pressed, the word "ALPHA" in the display indicates that the mode is set (the numbers in the stack are not altered). Now press the required keys to form the alpha message; the available alpha characters and their corresponding keys are shown below. After the last alpha character is keyed in, terminate the mode and print the message by pressing  again. If your message is longer than 15 characters, the line will be printed automatically as the 16th character is keyed in.



Alpha Mode Keyboard

For example, to print the alphabet:

Press



ABCDEFGHIJKLMNOPQRSTUVWXYZ
QRSTUVWXYZ

Notice that the first line was printed automatically as the 16th character (P) was keyed in.

Alpha Control Keys

These control keys are available when the Alpha mode is set:

NEW LINE prints all preceding characters and advances the paper one line. Successive NEW LINE instructions advance the paper, one line at a time.

BLANK Inserts a blank character space in the message.

NOTE

STEP and **RESET** are not alpha control keys. Pressing either key inserts a blank character space in a keyboard alpha message.

SHIFT Shifts the keys **A** through **N** to their alternate European characters. To reset the primary characters, press **SHIFT** again. The other alpha keys are not affected by the **SHIFT** key.

PRINT Prints the current number in X within the alpha message. The number appears right-justified on the line (unless alpha characters follow the number) and in the current number format.

For example, suppose you wish to label and print these three numbers as they appear in the stack:


Z	12.00	(N)
Y	123.45	(S)
X	1,234.56	(Total)





Press

  (place "N" in X)


     

N= 12.00

 (place "S" in X)

S= 123.45

 (replace "total" in X)

TOTAL = \$1234.56

Notice that each number appears right-justified on the line, and that the Alpha mode must be terminated before the numbers in the stack can be moved.

The number will always appear on the same line as the message provided that there is space for it. When the space is too small, however, the entire number is printed on the next line. Here is how the last line of the previous example would appear using a larger number ♦

TOTAL = \$
12345.67

Alpha characters can also appear on both sides of the printed number, as shown in the next example. In this case, notice that both the number and the characters following it appear right-justified on the same line.

To print "LENGTH = 2.54CM",

2.54  

LENGTH = 2.54CM

But when the number becomes too long to fit in the space available, first the trailing character(s) are printed on the next line ♦

LENGTH =1234.56C
M

And finally the number and its trailing characters all appear on the next line ♦

LENGTH =
123456.78CM

You'll find more examples using printer alpha in Sections 3 and 4.

Cancelling the Alpha Mode

If you make an error while doing a keyboard alpha message, the Alpha mode can be cancelled without printing the message by switching the mode switch to PRGM and back to RUN. This operation does not affect the numbers in the stack or the program memory.

3

Programming

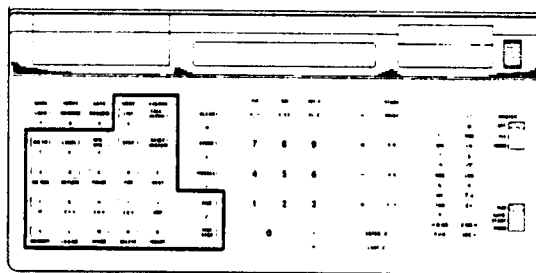
Introduction

Up until now, this manual has shown how the HP 9815A works for you from the keyboard: you enter numbers and press operator keys — the calculator does each operation instantly and leaves the result in the display. But you also have a more powerful feature than is available on most calculators: complete programmability!

Simply stated, programmability means that you can enter each of those key sequences that you've been doing from the keyboard as program instructions in the memory. Then tell the calculator to run them for you whenever you wish. So each key sequence need be pressed only once and the calculator will do it from then on.

Complete programmability means that every key sequence previously described can be programmed. It also means that there is a full set of program-control functions, such as: sub-routine branching and labelling, "IF" instructions for making computed decisions, and FOR-NEXT instructions for repeating program segments automatically.

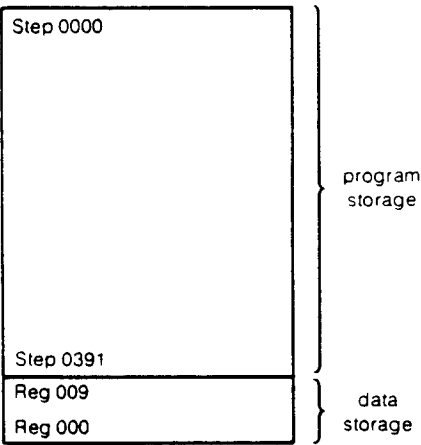
The program-control functions are grouped into one area of the keyboard, as shown on the right. The functions shown below keys **A** through **O** are automatically available when the mode switch is set to PRGM. All of those keys are programmable except for the editing keys:



Program Memory

The basic program memory contains 472 steps, numbered from 0000 through 0471. Each step holds one program instruction consisting of either a single keystroke (e.g., + , PRINT , SIN) or a combined key sequence (e.g., STORE A , v SIN). Appropriate key sequences are combined automatically as you enter the program. In some cases, combined key sequences cannot be stored into one step (e.g., FIX 3 , STORE 5 .), so an additional step is used. Examples of this appear in the first example program.

As shown on the right, part of the program memory is assigned to data storage when the calculator is switched on. If you wish to assign those bottom 80 steps to program storage, the 10 data registers can be deleted by pressing CL X STORE . For more details see "Additional Data Storage" in Section 1.



The basic memory can be expanded to 2008 program steps by ordering calculator option 001, or to 3800 steps by ordering a 9815S. The extra memory is normally installed at the factory. However, if you wish to have it installed at a later date, contact your HP Sales and Service office for details.

Clearing the Memory

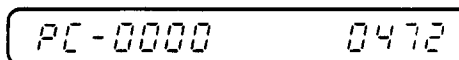
It's recommended that you clear the program memory before entering each new program. To erase the program memory, without altering existing data storage or the stack, switch to the Program mode and press P N . To clear the entire calculator memory (program and data storage), switch the calculator off.

Here is a quick preview of the general program controls and keys to be described next:

Program Controls



Set to enter program steps. The display format is:



current step address

steps to end of memory



Set to list each step as it's entered.

Program Counter

An internal mechanism controlling the current step displayed, printed, or executed.



Press to set the program counter to the indicated address. Programmed GOTO instructions send program control to the indicated step address.



Press to set the program counter to step 0000. When executed in a program, END also halts the program, and clears all program flags and subroutine nesting levels.



Press to start or stop program execution. When programmed, this instruction halts the program.



Press to print a list of program steps from the current step address to an END instruction.



Use to stop the program for about 1/8 second and display the current number in X.





Use to advance the printer paper.







A No Operation instruction is used to provide a blank area in the program for later use. The entire program memory is filled with NOP's when the calculator is switched on or the memory is erased.

The Program Counter


The program counter is an internal mechanism that determines which program step is displayed, printed, or executed. Many programming keys and instructions control the operation of the counter, allowing you to enter, edit, run, and record programs.

The program counter can be set to any desired step when the Run mode is set by pressing  and the number (or "address"). As when specifying numeric register names, you need not key in the entire address, just key in the significant digits and follow it with a non-numeric key or . For example, to set the program counter to step 0025:


Set 

Press either  25  or  0025



current step address




steps to end of program memory

Now set 





The display indicates the current step address and the number of steps available for programming to the end of memory. This latter number reflects any steps assigned to data storage or a peripheral interface.¹

To manually move the counter forward or back while you are entering a program, press  or . The counter advances automatically as you enter program steps.

The program counter is automatically set to step 0000 whenever you either press  (in the Run mode), or switch the calculator on, or erase the program memory.

Entering a Program

The general procedure to enter a program is:


1. Set  and press  (beginning step address). When the first step is to be step 0000, you can press  instead of using .

2. Set  



3. As each step is entered, the printer lists the step and the program counter increments to the next step address.
4. The last step of each program should be END.

¹A small portion of the program memory is automatically assigned to each interface plugged in when the calculator is switched on. See each interface operating manual for details.

Printout During Program Entry

The printer automatically lists each step address and instruction when its switch is set to NORM. A portion of a program listing, which calculates $2 + 5$ and prints the result, is shown on the right. When the Program mode is set, the printer also lists each step as the counter is advanced with .

```
0018 2
0019 ENTER+
0020 5
0021 +
0022 PRINT
```

To list any portion of a program, set the program counter to the first step and press . To stop the listing, press . The listing stops automatically after an END instruction is printed.

Display During Program Entry

As you enter a program, the display normally shows you the current (program counter) step address and the number of steps still available for programming ♦

PC-0025 0447



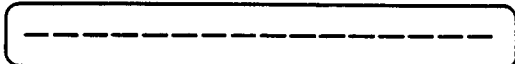
But when you are entering a STORE or RECALL instruction, the display looks like this ♦


PC-0025 REC-000

A numbered register name appears on the right as it's keyed in. Other special displays appear when branching instructions are being entered; see "Branching" and "Labels".

Running a Program

After the program has been entered:

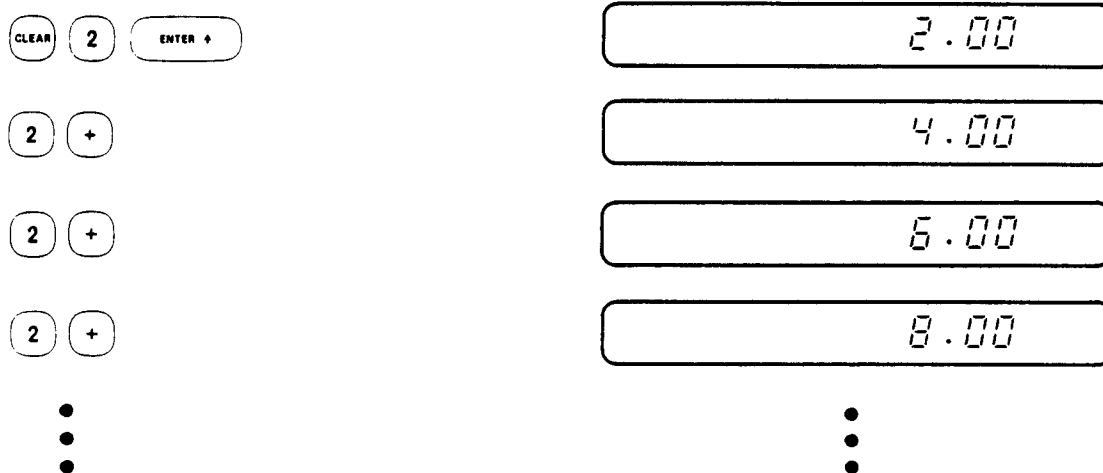
1. Set .
2. Set the program counter to the first step of the program.
3. Press  Display 

The program now runs automatically until it encounters either a STOP instruction (e.g., to wait for keyboard entries) or an END. While the program is running, the display appears as shown above, unless PAUSE instructions are used to display the current number in X. To manually stop the program, press .

Example Programs


First let's enter and run a simple program to count by two.

To find the key sequence needed to count by two from the keyboard:



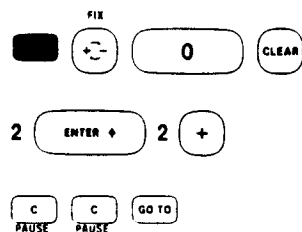
It looks like we can key in the basic sequence once and have the calculator repeat it by using a GOTO instruction (more uses for GOTO are described later). Also, let's set the number format to "FIX 0" and add two PAUSE instructions so we can view each result.

Here's how to enter the program:

Set  and press **END** to set the counter to step 0000.

Set  

Press



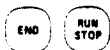
```
0000 FIX      0
0002 CLEAR
0003 2
0004 ENTER↑
0005 2
0006 +
0007 PAUSE
0008 PAUSE
```

Look at the listing and decide which step the program should return, in order to repeat the counting sequence. Then enter that step address to complete the GOTO instruction:



```
0009 GOTO 0005
0011 END
```

To see if the program runs:



4

6

8

10

⋮

Since the END instruction is not executed in this program, the program will run until you stop it.

If the program did not run correctly, first check it by listing it:



If your listing does not compare with this one ♦ enter the program again.

```
0000 FIX 0
0002 CLEAR
0003 2
0004 ENTER↑
0005 2
0006 +
0007 PAUSE
0008 PAUSE
0009 GOTO 0005
0011 END
```

There are four basic steps in creating a program:




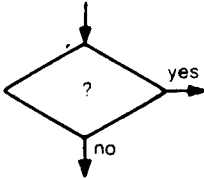
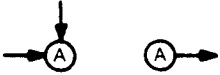
1. Define the problem (we wanted the calculator to count by two);
2. Decide how the problem is best solved (we tried the key sequence from the keyboard);
3. Write out the steps for the program;
4. Enter the steps into the calculator memory.

Since our counting program was quite simple, we combined steps 3 and 4. But for most programs, steps 2 through 4 will not be as easy. One method that will help you decide how the problem is best solved is **flowcharting**. Once you have a solution, use the program pad supplied with your calculator to write the program steps.

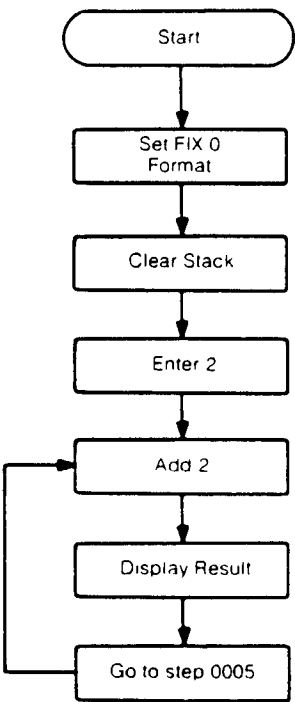
Flowcharting

A flowchart is a diagram representing each program operation. Usually, each symbol in the flowchart indicates one complete operation, rather than just one step or instruction.

Some general flowcharting symbols and their meanings are shown below.

Symbol	Meaning
	The terminator represents the starting or ending point in a program.
	The notation within a function block indicates one portion of the program.
	Flowlines indicate the direction of flow in the chart.
	The notation within a decision block indicates what computed test is being made. One flowline enters the diamond and two or three lines exit it. Two exit flowlines are usually labeled "yes" or "no". The IF and FLAG instructions described later are used for making computed tests.
	When it's inconvenient to draw a flowline between two or more points, connectors can be used. In this case, the "A" connectors indicate that the lines going to and from them are connected.

Here is a flowchart of the preceding example program:



Area of a Polygon

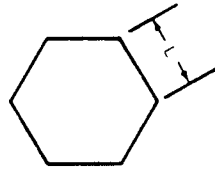
Now let's write a program to compute the area of any regular polygon, given its number of sides and length of a side. Use this equation:

$$A = \frac{1}{4}NL^2 \cot \frac{180}{N}$$

When: N = number of sides

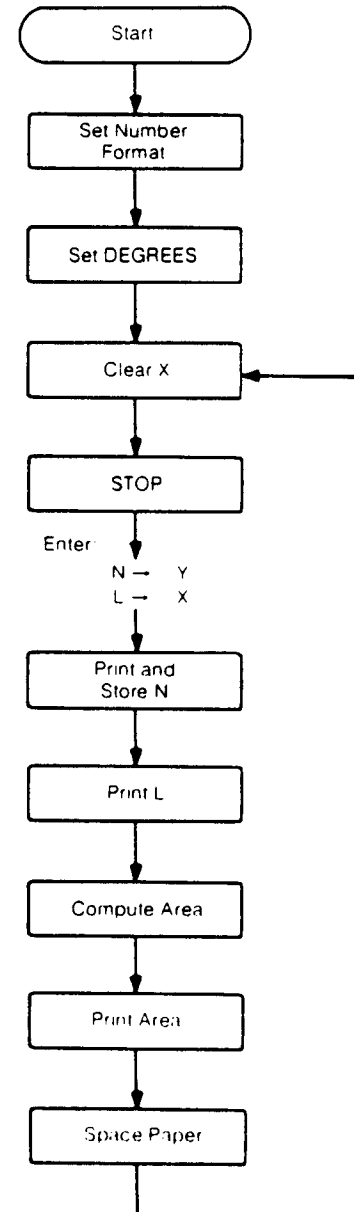
L = length of each side

$$\cot \theta = \frac{1}{\tan \theta}$$



Here's a flowchart and explanation showing one method to write the program:

1. Set desired number format.
2. Set degrees units.
3. Clear X before number entries.
4. Stop for number entries.
5. Print and store N.
6. Print L.
7. Compute area.
8. Label and print answer.
9. Space paper so answer is above printer window.



Here's a list of the program steps written using the program pad:

HP 9815A PROGRAM FORM

PROGRAM TITLE Area of a polygon PAGE 1 OF 1 PAGES

MEMORY SIZE ☒ Standard 472 steps DATE Mar 17, 1976 PROGRAMMER K Massey
☐ Expanded 2008 steps



PERIPHERALS ☐ HP 9862A Plotter (Uses 72 steps.) ☐ HP 9871A Printer (Uses 40 steps.) ☐ Other (Uses 8 steps.) ☐ Other (Uses 8 steps.) TOTAL REGISTERS SET

STEP #	INSTRUCTION	REMARKS	STEP #	INSTRUCTION	REMARKS
0	NOP	} space for no. format set degrees	50		
1	NOP		51		
2	DEGS		52		
3	CLEAR	← enter N→Y, L→X here	53		
4	STOP		54		
5	X=Y		55		
6	PRINT	} print N and L store N	56		
7	STO A		57		
8	X=Y		58		
9	PRINT	} compute area	59		
10	ENTER		60		
11	X		61		
12	X=Y		62		
13	÷		63		
14	÷		64		
15	X		65		
16	↓		66		
17	B		67		
18	O		68		
19	RCL A	} label and print ans.	69		
20	÷		70		
21	TAN		71		
22	½		72		
23	X		73		
24	PRINT α		74		
25			75		
26	A		76		
27	R		77		
28	E		78		
29	A	} space paper up repeat program	79		
30	(BLANK)		80		
31	=		81		
32	PRINT		82		
33	END α		83		
34	SPACE		84		
35	SPACE		85		
36	SPACE		86		
37	SPACE		87		
38	GOTO 3		88		
39		89			
40	END	90			
41		91			
42		92			
43		93			
44		94			
45		95			
46		96			
47		97			
48		98			
49		99			

NOTE: The following instructions require two program steps. All others require only one step.

- All storage instructions involving numbered registers or indirect storage.
- All branch instructions, except computed branches.
- All display formatting instructions.
- Any instruction beginning with CALL ALPHA.
- All labels.

To enter the program, first erase the program memory.

Set  and 













































Press


 

PC-0000

nnnn

Now enter the instructions:

Set 

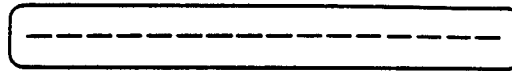
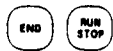
```

0000 NOP
0001 NOP
0002 DEGS
0003 CLEAR
0004 STOP
0005 X≠Y
0006 PRINT
0007 STO A
0008 X≠Y
0009 PRINT
0010 ENTER↑
0011 *
0012 X≠Y
0013 4
0014 ÷
0015 *
0016 1
0017 8
0018 0
0019 RCL A
0020 ÷
0021 TAN
0022 1/X
0023 *
0024 PRNTα
0026 A
0027 R
0028 E
0029 A
0030
0031 =
0032 PRINT
0033 ENDα
0034 SPACE
0035 SPACE
0036 SPACE
0037 SPACE
0038 GOTO 0000
0040 END

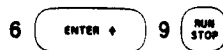
```

44 Programming

To start the program:



Now enter values for N and L into Y and X ,
respectively:



```

        6.00      (N)
        9.00      (L)
AREA =    210.44
  
```

The calculator is now ready to accept another set of values for N and L. Here are some examples:

```

        4.00
        8.00
AREA =    64.00
  
```

```

        18.00
        9.00
AREA =   2067.18
  
```

The NOP instructions at steps 0000-0001 provide space for you to enter another number format, if needed. For example, to enter a "FIX 5" instruction at step 0000,

Press



Set



Press



```
0000 FIX    5
```


Set



Now list the program and note that the new instruction replaced the two NOPs.

If your program does not run as shown here, first compare your program listing with the sample shown. When you find the error, set the program counter to that step address, set the Program mode, and re-enter the program from that point on. Later in this section "Editing Programs" shows how to insert and delete program steps.

Labels

- 115 label names are available for naming programs or subroutines: they are labels A through O and labels 00 through 99. When keying in one-digit numeric labels, be sure to follow each with either a non-numeric key or .
- Each label requires two program steps.
- Labels A through O are also used with special functions (see Section 5).
- If a branching instruction specifies a non-existent label name, "LABEL NOT FOUND" is printed and the program is halted at the branching instruction.
- When the same label name appears more than once in the memory, any branch to the label goes to the one closest to the beginning of the memory.


A label is a programmable aid used to name a location in the program memory. The label instruction is entered immediately before the program segment to which it refers. Then the program counter can be set to the label location by using any appropriate branching instruction, such as a GOTO.

Using labels provides a method of addressing program segments which is completely independent of step addresses. So, when step addresses change due to program editing, label addresses need not be changed.

A time-saving technique often used when entering and debugging programs is to use labels wherever possible for branching. Then, as steps are inserted or deleted to complete the program, the branching addresses stay valid even though step addresses change. Once the program is running, the labels can be removed and the branching instructions can be changed to use step addresses.

NOTE

Since labels A through O are also used for special function names, it's recommended that they not be used in programs where special functions are also in use. Otherwise, pressing a special function key could start the program running at the label location, rather than run the intended special functions.

A label name is entered by pressing  and the desired name. As when keying in numeric register names, you need not key in the entire name (see below). Each label requires two program steps.

For example, if you are entering a program and wish to have "label 01" located at step 0050:

LABEL 1 • or LABEL 01

```
0049  •
0050  LBL
----- 01
0052  •
```

And to enter "label C" at step 150:

LABEL C

```
0149  •
0150  LBL
-----  C
0152  •
```

The dashed line is included with each label to help you find the label in a long program listing.

Here are two labelled segments at the end of a program; label 05 calculates $(A \times B)/(A + B)$ and label 06 calculates $.7071(X)$.

0100	LBL		0111	LBL	
----	05		----	06	
0102	STO	B	0113	.	
0103	X=Y		0114	7	
0104	STO	A	0115	0	
0105	*		0116	7	
0106	RCL	A	0117	1	
0107	RCL	B	0118	*	
0108	+		0119	FIX	4
0109	÷		0121	END	
0110	NOP				

Program execution can branch to either label 05 or label 06. If the program branches to label 05, however, both labels will be executed. For example, to have the program branch from step 0055 to label 05:

GO TO LABEL 5 •

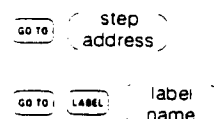
```
0054  •
0055  GOTO    L05
0057  •
```

Once the labelled segments are entered, you can also run them from the keyboard. To run label 06 enter a value in X and press GO TO LABEL 6 RUN STOP .

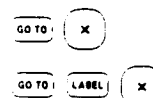
If you wish to run label 05 without label 06, replace the NOP instruction at step 0110 with a STOP instruction.

Branching

An absolute branch sends program control to the specified address or label. (9815S users, see Appendix 2.)



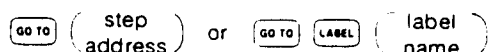
A computed branch sends program control to the address or label indicated by the current number in X.



- Absolute branching instructions require two program steps.
- The entire step address or numeric label name need not be keyed in; just key in the significant digits, followed by \odot or a non-numeric key.
- If a step address that exceeds the available program memory is specified, the program is halted at the incorrect instruction and "ILLEGAL ADDRESS" is printed.
- If a branch is made to an unavailable step address the program will go to the previous step.
- If a non-numeric key is pressed directly after **GO TO**, a "GOTO 0000" instruction is automatically entered. To cancel an incompleted GOTO instruction, press **RESET** **ENTER**.

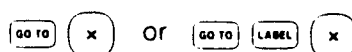
A branching instruction causes the program counter to go to a location other than the next sequential step in memory. Program execution then continues at the new step.

Two kinds of branching instructions are available: absolute and computed. An absolute branch causes the program to go to a fixed memory location (either a step number or a label name). The first sample program in this chapter contained an absolute branch. The general sequence for entering absolute branching instructions is:



As when keying in numeric register names and numeric label names, you need not key in the entire address; just key in the significant digits, followed by \odot or a non-numeric key.

A computed branch causes the program to go to a location indicated by the current number in X. Depending upon the branching instruction, the absolute integer portion of the number in X indicates either a step number or a numeric label name. The general sequence for entering computed branching instructions is:



For example, to branch from step 0031 to step 0100:

GO TO 100 **•** or **GO TO** 0100

```
0030 •
0031 GOTO 0100
0033 •
```

And to branch from step 0050 to label A:

GO TO **LABEL** **A** or **GO TO** **A**

```
0049 •
0050 GOTO A
0052 •
```

To branch from step 0150 to the step indicated by the current number in X:

GO TO **X**

```
0149 •
0150 GOTO X
0151 •
```

And to branch from step 0300 to the numeric label indicated by the number in register A:

RECALL **A** **GO TO** **LABEL** **X**

```
0298 •
0299 RCL A
0300 GOTO LX
0301 •
```

Notice that each absolute branching instruction uses two program steps, while each computed branch uses only one.

When you are entering a GOTO instruction, the display appears like this ♦

The branching address appears on the right as it's keyed in.

PC-0025 A-0000

But when you are entering a "GOTO LABEL" instruction, this display is shown ♦

A numeric label name appears on the right as it's keyed in.

PC-0025 LBL-00

Here is a part of the "Calendar" program available on the Utility and Test Cartridge. The beginning of the program requests a date (day, month, year) and calculates a number between 0 and 6 which indicates the day of the week. The number is placed in X after step 0148. This sequence shows how a computed branch at step 0149 is made to the label indicated by the number in X.

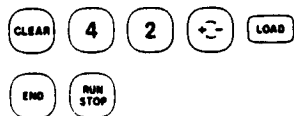
```

0148  •
0149  GOTO    LX
0150  LBL
----  00
0152  PRNTα
0154  S
0155  A
0156  T
0157  U
0158  R
0159  D
0160  A
0161  Y
0162  ENDα
0163  GOTO    0068
0165  ENDα
0166  LBL
----  01
0168  PRNTα
0170  S
0171  U
0172  N
0173  D
0174  A
0175  Y
0176  ENDα
0177  GOTO    0068
0179  LBL
----  02
0181  PRNTα
0183  M
0184  0
0185  N
0186  •

```

If you wish to run the program, first insert the Test Cartridge into the tape drive and press **REWIND**.

Then press



Now enter one or two-digit integers for the day, month and year:



The program is now ready to calculate another date.

CALENDAR
(1901+1999)

```

DAY=?
4
MONTH=?
7
YEAR=?
76

SUNDAY

```

DAY=?

IF Instructions

- IF instructions make decisions using current numbers in the stack or program flags. If the condition is "true", the next program instruction is executed. But if the condition is "false", the next instruction is skipped.
- These IF instructions are available.

To enter:

Press:

If $X < Y$ If $X = Y$ If $X \geq Y$ 

If X is +



If X is -

If $X = 0$ 

If flag n is set



If flag n is clear



The IF key need not be pressed.

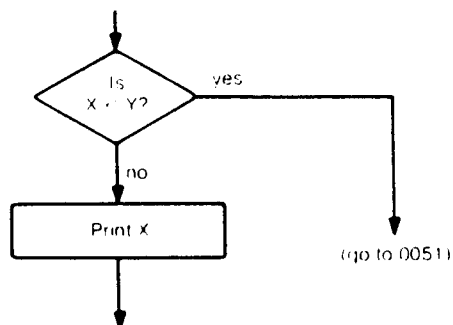
- The three "IF X & Y" instructions compare the entire number in each register, including the guard digits which are not displayed.

The IF instructions enable the calculator to make logical evaluations based on either the current numbers in X (and Y) or the current state of program flags (flags are described later). If the evaluation is "true", the next program instruction is executed. But if the evaluation is "false", the next instruction is skipped.

The step following an IF usually contains a branching instruction, which creates a most powerful means of branching. For example, step 0026 in the following sequence is an IF instruction which compares the numbers in X and Y. If the number in X is less than Y (result equals "true") step 0027 is executed. But if X is equal to or greater than Y, step 0027 is skipped.

```

0025  •
0026  IF X<Y
0027  COTO 0051
0029  PRINT
0030  •
  
```



Eight IF instructions are available. They ask the questions:

Evaluation:

Enter:

Is X less than Y?	<input type="button" value="F"/> <input type="button" value="G"/>	0026 IF X<Y
Is X equal to Y?	<input type="button" value="F"/> <input type="button" value="H"/>	0026 IF X=Y
Is X greater than or equal to Y?	<input type="button" value="F"/> <input type="button" value="I"/>	0026 IF X≥Y
Is X negative?	<input type="button" value="F"/> <input type="button" value="-"/>	0068 IF -
Is X positive?	<input type="button" value="F"/> <input type="button" value="+"/>	0068 IF +
Is X equal to 0?	<input type="button" value="F"/> <input type="button" value="0"/>	0068 IF 0
Is Flag n set?	<input type="button" value="F"/> <input type="button" value="SFG"/> <input type="button" value="CFG"/> (n)	0135 IF SFG 5
Is Flag n clear?	<input type="button" value="F"/> <input type="button" value="SFG"/> <input type="button" value="CFG"/> <input type="button" value="CFG"/> (n)	0135 IF CFG 1

The key does not have to be pressed to enter the first three instructions shown above.

You may recall that the first program in this section ("count by two") ran until you stopped it. Here is the same program, but now it will stop automatically at 100 because of the IF instruction. As the program runs, step 0013 compares the current result with 100; step 0014 is executed as long as the result is less than 100. But when $X = Y$, step 0014 is skipped, ending the program.

```

0000 FIX      0
0002 CLEAR
0003 2
0004 ENTER↑
0005 2
0006 +
0007 PAUSE
0008 PAUSE
0009 1
0010 0
0011 0
0012 X≠Y
0013 IF X<Y
0014 GOTO 0005
0016 END

```

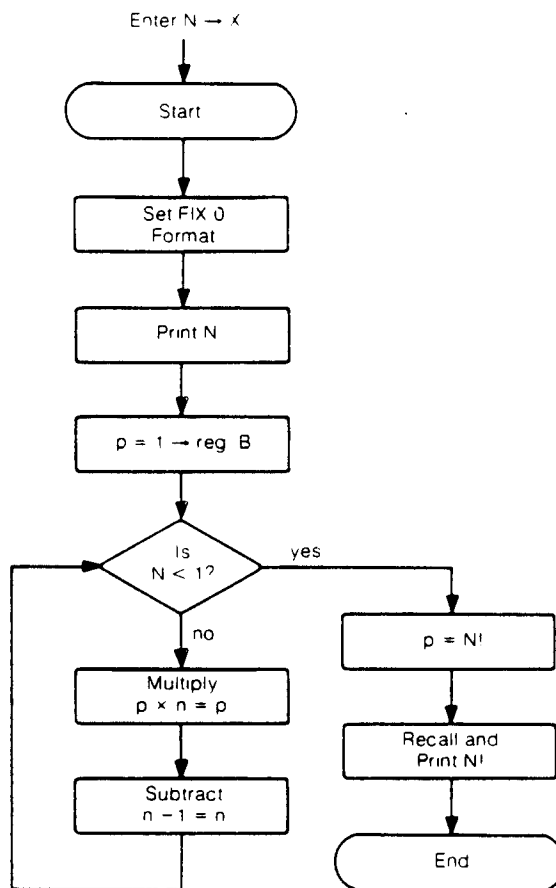
The next program calculates $N!$ (N factorial) for values of N from 0 through 69.

$$N! = N(N - 1)(N - 2) \dots (N - N + 2)(1)$$

$$0! = 1(\text{by definition})$$

For example: $6! = 6 \times 5 \times 4 \times 3 \times 2 \times 1 = 720$

Here is a flowchart and listing of the program:



```

0000 FIX      0
0002 PRINT
0003 1
0004 STO      B
0005 IF X≥Y
0006 GOTO     0015
0008 X⇐Y
0009 STO*     B
0010 X⇐Y
0011 -
0012 1
0013 GOTO     0005
0015 RCL      B
0016 PRINT
0017 SPACE
0018 END
  
```

n = current multiplier term
(N, N - 1, N - 2, etc.)
p = partial result of N!

After entering the program beginning at step 0000, set the Run mode, enter a value for N, say 13, and start the program:

13 END RUN STOP

13 N
6227020800 N!

After the results are printed, the END (step 0018) halts the program and resets the program counter to step 0000. Now you're ready to enter another value and run it again.

The step following an IF need not always contain a branching instruction. For instance, the sequence on the right checks the numbers in X and Y and prints the smaller number.

```

0200 •
0201 IF X≥Y
0202 ROLL↓
0203 PRINT
0204 •
  
```



Here's another example, which replaces X with its absolute value by changing its sign (step 0202) if it's negative.

```

0200 •
0201 IF -
0202 +⇐-
0203 •
  
```

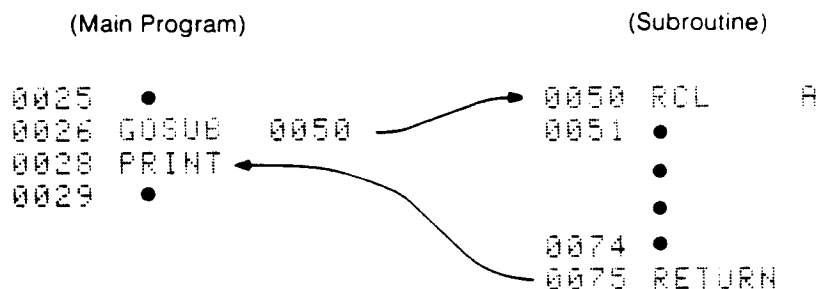
You'll find many examples using IF instructions in the rest of this manual.

Subroutines

- A subroutine is a program sequence used many times, but stored only one place in the memory.
- A GOSUB branching instruction "calls" and runs a specified subroutine. The last instruction in the subroutine is RETURN, which causes the program to return to the next instruction following the GOSUB.
- GOSUB instructions can specify absolute step addresses or label names, or they can be computed branches. The same rules for entering GOTO instructions apply here.
- Subroutines can be "nested" up to seven levels deep at any time. An attempt to nest to an eighth level causes the program to halt and print "GOSUB OVERFLOW".
- Pressing  or executing an END instruction clears all subroutine nesting levels.
- Executing a RETURN without a corresponding GOSUB (i.e., a return address is not available) causes the program to halt and print "MISSING GOSUB".
- If a non-numeric key is pressed directly after , a "GOSUB 0000" instruction is automatically entered.

A subroutine is a sequence of program instructions which is used many times, perhaps in several different programs, yet need be stored only once in the memory. A program can branch to (or call) a subroutine at any time by using a GOSUB instruction. Then, after the subroutine has been executed, a RETURN instruction located at the end of the subroutine causes operation to resume at the step ("return" address) following the GOSUB. Using subroutines saves both considerable space in the memory and program-writing time.

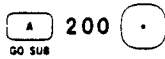
The following program segments show subroutine branching.



Program control branches to step 0050 and executes the subroutine. At step 0075 the RETURN instruction sends operation to the return address (the next step after the GOSUB instruction).

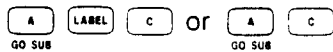
A GOSUB instruction can call a subroutine by specifying either a step address or a label name, or it can be a computed branch similar to the GOTO instruction.

For example, to call a subroutine beginning at step 0200 from step 0050:



```
0049  ●
0050  GOSUB  0200
0052  ●
```

Or to call a subroutine beginning at Label C:



```
0049  ●
0050  GOSUB  C
0052  ●
```

To call a subroutine beginning at the step address indicated by the current number in X:



```
0049  ●
0050  GOSUB  X
0051  ●
```

And to call a subroutine beginning at the label name indicated by the number in X:



```
0049  ●
0050  GOSUB  LX
0051  ●
```

Notice that each absolute GOSUB instruction requires two program steps.

Nesting Subroutines

Although any number of subroutines may be called individually during a program, it is also possible to use more than one subroutine at one time. The action of one subroutine calling a second subroutine which, in turn, can call a third subroutine, is known as "nesting". The calculator can remember up to seven return-addresses at a time, so that subroutines may be nested up to a depth of seven. Calling an eighth subroutine without first returning to at least the sixth level causes the program to halt and print "GOSUB OVERFLOW".

Returns are made on a "last-in, first-out" basis, the return always being made to the last return-address remembered. As soon as the return is made, that return-address is forgotten, so that the previous address now becomes the last one. Thus the returning order is always the opposite of the calling order.

NOTE

When preparing to run a program which nests subroutines, press **END** to insure that all seven levels of nesting are available. END automatically erases any return-addresses to which the return has not yet been made.

Example Program

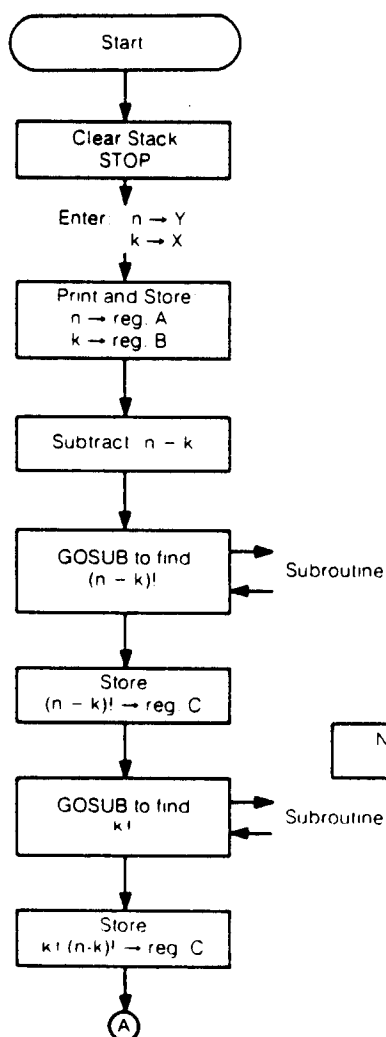
The next program uses a subroutine based on the N! program shown earlier. This program calculates the possible combinations of n objects taken k at a time. For example, if a box contains 15 differently colored balls (n), how many possible color combinations (c) are there if 5 balls (k) are selected (and then returned) at a time? The answer = 3003.

The formula used in this "possible combinations" program is:

$$C_k^n = \frac{n!}{k!(n-k)!}$$

The program calculates C for any values of n and k from 0 through 69.

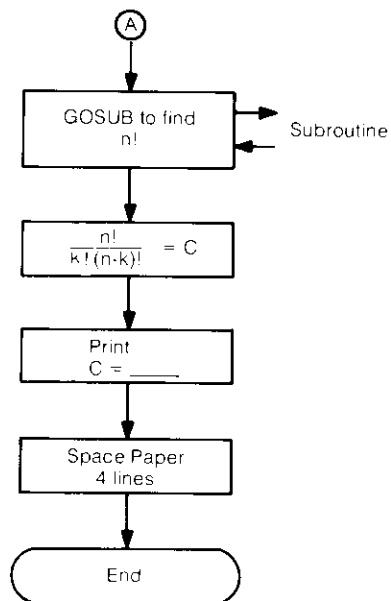
Here's a flowchart and listing of the program. A flowchart of the N! program is on page 52.



```

0000 CLEAR
0001 STOP
0002 X≠Y
0003 STO      A
0004 PRNTα
0006 N
0007 =
0008 PRINT
0009 ENDα
0010 X≠Y
0011 STO      B
0012 PRNTα
0014 K
0015 =
0016 PRINT
0017 ENDα
0018 -
0019 GOSUB    0043
0021 STO      C
0022 RCL      B
0023 GOSUB    0043
0025 STO*     C
0026 RCL      A
0027 GOSUB    0043
0029 RCL      C
0030 ÷
0031 PRNTα
0033 C
0034 =
0035 PRINT
0036 ENDα
0037 SPACE
0038 SPACE
0039 SPACE
0040 SPACE
0041 GOTO     0059
  
```

(con't)



```

0043 ST0      D
0044 1
0045 ST0      E
0046 IF X≥Y
0047 GOTO     0056
0049 X≠Y
0050 ST0*     E
0051 X≠Y
0052 -
0053 1
0054 GOTO     0046
0056 RCL      E
0057 PAUSE
0058 RETURN
0059 END
  
```

You need not key in this program, since it is on the Utility and Test Cartridge supplied with the calculator. To load and start the program, first insert the test cartridge into the tape drive and press **REWIND**.

Then press

CLEAR **43** **←** **LOAD** **END** **RUN/STOP**

0.00

Now enter a value for n in Y and a value for k in X:

Press

15 **ENTER** **5** **RUN/STOP**

N= 15.00
K= 5.00
C= 3003.00

Before the final answer appears, the display will flash three times, showing the results of the factorial subroutines for each value of N (n - k, k, and n respectively).

The N! subroutine could also be run "stand alone", by entering a value for n into X and pressing **END** **GO TO** **43** **RUN/STOP**.

Pressing **END** is needed to clear any return addresses in the calculator.

When running the subroutine "stand alone", it stops with N! in X. The error message "MISSING GOSUB" is printed since there is no return address for the RETURN instruction at step 0058 (ignore the error in this case).

FOR-NEXT Loops

- FOR-NEXT instructions allow controlled repetition of program sequences. Three sets of registers are used: A & F, B & G, and C & H. The first register, the loop counter, is incremented after each pass of the loop and compared with the final value in the second register. The loop is exited when the loop counter passes the final value.
- The loop counter is incremented by one when A & F and B & G are used. When C & H are used, however, C is incremented by the value in register D. If D = 0, the loop is repeated continuously.
- A FOR-NEXT loop is always executed once, regardless of the loop counter and final values.
- FOR-NEXT loops can be nested up to three deep.
- If a NEXT is executed without a corresponding FOR, the program is halted and "MISSING FOR STMT" is printed.

The FOR-NEXT instructions permit the repetition of any program sequence a given number of times. The FOR and NEXT instructions form a loop with the program segment to be repeated between them.





Each FOR-NEXT instruction is used with a set of data registers. Register sets A & F, B & G, and C & H are available for this use. The first register is specified in the FOR instruction and is called the "loop counter". The second register holds the final (or exit) value. When register sets A & F and B & G are used, the loop counter is incremented by one each time the loop is executed. When register set C & H is used, however, the loop counter is incremented by the value in register D.

In the example on the right, the loop counter (reg. A) and the final value (reg. F) are first initialized. The FOR instruction at step 0036 indicates the beginning of the loop. Each time NEXT is executed, the loop counter is incremented by one and compared with the final value; when the loop counter is greater than the final value the calculator exits the loop and continues with the step after NEXT. In this sequence, steps 0037 - 0059 are repeated five times.

```

0031  •
0032  1
0033  STO      A
0034  5
0035  STO      F
0036  FOR      A→F
0037  •
      •
      •
0059  •
0060  NEXT      A
0061  •

```

The FOR instruction shown above was entered by pressing   at step 0036 and the NEXT was entered by pressing   at step 0060. Keying in the first register name indicates which register set is to be used.

For many applications it's easier to use FOR-NEXT rather than IF instructions, as in this comparison of two simple programs which display the numbers 1 through 100 in succession:

Using IF	Using FOR-NEXT
0000 1	0000 1
0001 STO A	0001 STO A
0002 RCL A	0002 1
0003 PAUSE	0003 0
0004 1	0004 0
0005 0	0005 STO F
0006 0	0006 FOR A+F
0007 IF X<Y	0007 RCL A
0008 GOTO 0014	0008 PAUSE
0010 1	0009 NEXT A
0011 STO+ A	0010 END
0012 GOTO 0002	
0014 END	

The program using FOR-NEXT is easier to key in, takes up less memory, and executes faster than the one using IF.

The next example shows how a FOR-NEXT loop can be used to store values into a block of data registers. The program uses a second FOR-NEXT loop to print the stored values.

0000 1		
0001 STO B		
0002 9		
0003 STO G		
0004 FOR B+G		
0005 RCL B		
0006 2		
0007 Y↑X		
0008 STO I		
0010 NEXT B		
0011 FIX 0		
0013 1		
0014 STO B		
0015 FOR B+G		
0016 RCL I		
0018 PRINT		
0019 NEXT B		
0020 END		

Printout:

1
4
9
16
25
36
49
64
81

Store B^2
in registers
1 through 9.

Recall and
print registers
1 through 9.

It's not always necessary for the loop counter to be incremented by one after each pass in a FOR-NEXT loop. The next examples show how to increment by other values when using register set C & H and specifying an incremental value in register D. Printouts are shown below each program.

Increment by 5

```

0000 FIX      2
0002 0
0003 STO      C
0004 2
0005 5
0006 STO      H
0007 5
0008 STO      D
0009 FOR      C+HD
0010 RCL      C
0011 PRINT
0012 NEXT     C
0013 END

```

```

0.00
5.00
10.00
15.00
20.00
25.00

```

Increment by π

```

0000 FIX      4
0002 0
0003 STO      C
0004 2
0005 5
0006 STO      H
0007  $\pi$ 
0008 STO      D
0009 FOR      C+HD
0010 RCL      C
0011 PRINT
0012 NEXT     C
0013 END

```

```

0.0000
3.1416
6.2832
9.4248
12.5664
15.7080
18.8496
21.9911

```

Increment by $-.5$

```

0000 FIX      2
0002 1
0003 0
0004 STO      C
0005  $\div$ 
0006 STO      H
0007 .
0008 5
0009  $\div$ 
0010 STO      D
0011 FOR      C+HD
0012 RCL      C
0013 PRINT
0014 NEXT     C
0015 END

```

```

10.00
9.50
9.00
8.50
8.00
7.50
7.00
-8.50
-9.00
-9.50
-10.00

```

In each example the loop counter is incremented by the value in register D. Notice that the incremental value can be any positive or negative value. If the incremental value is 0, however, the loop is repeated continuously since C is not incremented.

The example on the right shows that a FOR-NEXT loop can be exited before the final value is reached. In this case, the values in A and E are printed when A is equal to or greater than E.

```

0145  •
0146  1
0147  STO      A
0148  5
0149  0
0150  STO      F
0151  FOR      A+F
0152  RCL      E
0153  RCL      A
0154  IF X≥Y
0155  GOTO     0169
0156  •
0157  •
0158  •
0159  •
0160  •
0161  •
0162  •
0163  •
0164  •
0165  •
0166  •
0167  •
0168  NEXT     A
0169  PRINT
0170  ROLL↓
0171  PRINT
0172  •

```

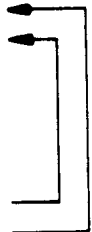
Nesting FOR-NEXT Loops

FOR-NEXT loops can be nested; that is, they can be located inside one another. Since three sets of registers are available, three FOR-NEXT loops can be nested at a time.




```

0025  FOR      A+F
0026  FOR      B+G
      •
      •
      •
      •
0060  NEXT     B
0061  NEXT     A

```



Flags

- Flags are used for indicating program conditions or making decisions. Flags 1 through 4 are for general use, while flags 5 through 8 have special meanings.
- Each flag can be set or cleared from either the keyboard or a program. END clears all flags.
- To set a flag or enter a "SET FLAG n" instruction, press  (n) . To clear a flag or enter a "CLEAR FLAG n" instruction, press   (n) .
- Use an IF instruction to check each flag's state. "IF FLAG" instructions do not change the state of a flag.

Flags are programmable indicators which allow the calculator to either make decisions or tell the user of program conditions. A flag has two states: "set" and "clear". Each flag can be set and cleared either from the keyboard or a program. In addition, all flags are cleared by pressing or executing END, or by switching the calculator on.

Eight flags are available. Flags 1 through 4 are for general program use, while flags 5 through 8 have dedicated meanings.








In general usage, a flag is set by some program sequence or event. Then, later in the program, the flag's state can be checked to determine a subsequent activity.

The example on the right shows a typical use for a flag. Steps 0025 - 0026 set flag 1 if the number in X is negative. Then steps 0035 - 0036 branch to step 0051 if flag 1 is clear; if flag 1 is set, (i.e., X is negative) the program continues at step 0038.

```

0024  •
0025  IF  -
0026  SFG      1
0027  •
      •
      •
0034  •
0035  IF  CFG  1
0036  GOTO    0051
0038  •

```

Flags can be set and cleared from the keyboard, too. As shown on the  key, to set, say, flag 1 from the keyboard press  (1) . And to clear flag 1, press   (1) . The same general sequences are used to enter FLAG instructions. For instance, the "IF CLEAR FLAG 1" instruction at step 0035 above was entered by pressing    (1) .

Flags 5 & 6 are used to intercept certain error messages. When flag 6 is set, the "suppressable" error messages such as "OVERFLOW" will not be printed. Instead, flag 5 is automatically set whenever a suppressable error occurs. A list of suppressable error messages is at the back of the manual.

Here is a sample use for flags 5 & 6 ♦

Setting flag 6 and monitoring flag 5 allows the program to continue running steps 0148 - 0194 until the calculating range is exceeded. When flag 6 is set (OVERFLOW) the program skips to step 0198.



```

0145  •
0146  SFG      6
0147  CFG      5
0148  •
      •
      •
0194  •
0195  IF CFG  5
0196  GOTO    0148
0198  PRINT
0199  CFG      6
0200  •

```

Flag 7 is automatically set whenever a STOP instruction is executed.¹ If a number is entered before the program is restarted, flag 7 is cleared. But if a number is not entered and the program is restarted, flag 7 remains set.



A typical use for flag 7 is shown in this program sequence ♦

which sums a series of numbers. The user keys in each data value at step 0035 and presses ; flag 7 will be cleared whenever a value is entered. When there are no more values to enter, the user presses  (but makes no entry), causing flag 7 to remain set. When flag 7 is set, steps 0036 - 0037 branch the program to step 0043.





```

0033  •
0034  CLRA+J
0035  STOP
0036  IF SFG  7
0037  GOTO    0043
0039  SUM+
0040  CLEAR
0041  GOTO    0035
0043  •

```

Flag 8 is "toggled" by pressing  while a program is running; that is, pressing  changes the flag's state from clear to set, or from set to clear.

As a simple exercise, enter this program ♦

To start the program, press  . Since flag 8 was cleared by pressing , "CLEAR" will be printed continuously. Now press  to toggle flag 8 back and forth.










```

0000  IF SFG  8
0001  GOTO    0013
0003  PRNTα
0005  C
0006  L
0007  E
0008  A
0009  R
0010  ENDα
0011  GOTO    0000
0013  PRNTα
0015  S
0016  E
0017  T
0018  ENDα
0019  GOTO    0000
0021  END

```

¹When single stepping a program with the  key, flag 7 will be set after each instruction.





Editing Programs

	Lists program steps, from the current step address to an END. This can be done in either Run or Program mode.
	Advances the program counter. Use in the Run mode to execute the program, one step at a time. Use in the Program mode to manually list the program.
Program mode only:	
	Use to decrement the program counter, and to clear an incorrect key sequence being entered.
	Deletes the instruction at the current step address and corrects any affected branching instructions.
 (new keys)	Inserts new instructions at the current step address. Terminate the insert operation by either pressing  again or setting the Run mode.
 	Erases the program memory, except assigned data storage, and sets the program counter to step 0000.
	Manually sets the Alpha mode to permit changing or inserting alpha instructions.

Program operation can be easily verified and changed by using the editing keys listed above. Please remember that editing operations are not programmable.

Program Listing

When a program does not run as expected, the first step in correcting it is to study a listing of the program.

To list an entire program, set the program counter to the first step address and press . To list only a portion of the program, set the program counter to the first step to be listed (use the  key) press , and then press  to halt the listing.

Now you can compare the listing with either your original list of program instructions or your flowchart; when the error is found, use the editing keys to correct the program.

A program listing is also a useful reference while single-stepping through the program, as described next.

Single Stepping

An easy method of checking a program is to run it one step at a time. This is done in the Run mode, by first setting the program counter to the beginning of the program, and then pressing **STEP** repeatedly. Each time you press **STEP** the current instruction is executed, the program counter is advanced to the next step, and the display returns so that you can view the results. If the current step contains a branching instruction, the program counter is advanced to the specified new address.

When a program is single-stepped to a PRINT ALPHA instruction, the entire alpha message is printed; then the program counter is set to the step following END ALPHA.

Pressing **RESET** has no effect on the program when the Run mode is set.

Changing Program Instructions


To change a program step, first set the program counter to the step address (use **GO TO**), set the Program mode, key-in the correct instruction, and then set the Run mode.

If the new instruction requires two program steps, but the old instruction uses only one step, the calculator will automatically shift the rest of the program down one step and, in addition, renumber any affected branching instructions. Similarly, if the new instruction requires only one step, while the old step uses two, the calculator will shift the program up and correct any affected branching instructions.

For example, suppose that this "count by two" program introduced at the start of this section is still in your calculator, and now you wish to change steps 0003 and 0005 so that it will count by one:

```
0000 FIX      0
0002 CLEAR
0003 2
0004 ENTER↑
0005 2
0006 +
0007 PAUSE
0008 PAUSE
0009 GOTO     0005
0011 END
```





Press **GO TO** 3 and set  and 
 1 **STEP** 1

Now set 

```
0003 1
0004 ENTER↑
0005 1
```

Changing the operating mode terminates the GOTO instruction, just like pressing .

To see how the calculator automatically corrects branching instructions when you change the number of steps in use, change the "FIX 0" instruction at step 0000 to a NOP:



Press  and set 
 



0000 NOP

Now set the Run mode and list the program. Notice that the instructions were automatically moved up by one step and that the GOTO was corrected.

```
0000 NOP
0001 CLEAR
0002 1
0003 ENTER↑
0004 1
0005 +
0006 PAUSE
0007 PAUSE
0008 GOTO 0004
0010 END
```

The calculator also automatically changes branching instructions as you enter them, when they conflict with instructions already in the memory. For example, if you tried to load a "GOTO 0009" at step 0000:


Press  and set 


 9 and set 

0000 GOTO 0010




Notice that the calculator corrected the GOTO for you. GOTO sequences from the keyboard will be automatically changed too! See page 70 for an example.

Deleting Instructions

To delete a programmed instruction, first set the program counter to the unwanted instruction. Then, set the Program mode and press . The calculator automatically moves the rest of the program up to fill the vacant step and corrects any affected branching instructions.

To delete an entire sequence of instructions, set the program counter to the first unwanted step, set the Program mode, and then press  once for each instruction in the sequence.

For example, to delete the NOP in the last example:





Press  and set 


Now set the Run mode and list the program.
 Note that the GOTO address was automatically corrected.

```
0000 CLEAR
```

```
0000 CLEAR
0001 1
0002 ENTER↑
0003 1
0004 +
0005 PAUSE
0006 PAUSE
0007 GOTO 0003
0009 END
```

Now delete steps 0001 and 0002, since the program will run without them:

Press  1 and set 
 




```
0001 ENTER↑
0001 1
```

Each time an instruction is deleted, the step's new instruction is printed.



Now set the Run mode and list the program again.

```
0000 CLEAR
0001 1
0002 +
0003 PAUSE
0004 PAUSE
0005 GOTO 0001
0007 END
```

Inserting Instructions

To insert one or more instructions into a program, first set the program counter to the step address at which the first new instruction is needed. Then set the Program mode, press , and enter the new instruction(s); then terminate the operation either by pressing an editing key (except  or ) or by setting the Run mode. The calculator automatically corrects any affected branching instructions when the insertion is terminated.

For example, let's insert two more PAUSE's into the program shown above to slow it down. We'll insert the new instructions before the GOTO:

Press  5 and set 



INSERT

  
PAUSE PAUSE INSERT

PC-0005



PC-0005 I

0005 PAUSE
0006 PAUSE

Now set the Run mode and list the new program. Notice that the I symbol appears in the display while steps are being inserted. Then pressing  again terminates the insert operation and removes the I.



```
0000 CLEAR
0001 1
0002 +
0003 PAUSE
0004 PAUSE
0005 PAUSE
0006 PAUSE
0007 GOTO 0001
0009 END
```

Clearing the Memory

To erase the program memory, set the Program mode and press  . The program memory is filled with NOP's and the program counter is reset to step 0000.

This operation does not affect any assigned data storage, the stack, or any special HP programs in the memory. (See page 18 for instructions on clearing data storage.)

Editing Alpha



Instructions in an alpha message can be edited using the keys just described, however the Alpha mode must be set **before** the instructions can be successfully changed or inserted. Otherwise, pressing the keys will enter their non-alpha meanings. To manually set the Alpha mode ("ALPHA" appears in the display), press  .

For example, the program sequence on the right prints this line:

SALES = \$ 0.00

```
0125 PRNTα
0127 S
0128 A
0129 L
0130 E
0131 S
0132
0133 =
0134
0135 $
0136 PRINT
0137 ENDα
```

To change the "\$" character to "L":

Press  135 and set 



PC-0135 ALPHA

0135 L

Now set the Run mode and run the sequence:

SALES = L 0.00

If we had attempted to key in "L" like this (by not setting the Alpha mode):

Press  135 set 






0135 LD&GO

Running the sequence would result in:

SALES = & 0.00

Entering other instructions within an alpha message when the Alpha mode is **not** set will cause other unexpected characters to be printed. See "Additional Alpha Characters" on page 71.

Now let's replace the "L" in the last example with the "£" (Pound) symbol. To do this we have to first change step 0135 to SHIFT and then insert :

Press  135 and set 



PC-0135 ALPHA

0135 SHIFT

Set  (to terminate the Alpha mode)

Set 

PC-0136

Press  

PC 0136 £ ALPHA








0136 F

Now set the Run mode and run the sequence:



SALES = £ 0.00

Please remember these things when you are changing or inserting instructions into an alpha sequence:

- Before **changing** alpha instructions, be sure the Alpha mode is set. Press  to manually set the mode.
- To **insert** instructions first press . Then set the Alpha mode by pressing . Then load the new alpha instructions and terminate the insert operation. To terminate both insertion and the Alpha mode, switch to the Run mode.
- When **deleting** Alpha instructions be sure the Alpha mode is not set. Otherwise, pressing  will enter the alpha character.
- If the Alpha mode is not set, instructions changed or inserted will cause different characters to be printed.
- To **list** an Alpha sequence, set the Run mode, set the program counter to the PRINT ALPHA instruction, and press . If a listing is started from within an Alpha sequence, unexpected instructions will be listed (see page 71).

Programming Hints

An Editing Tip

When many steps are being changed in a program, it's easy to lose track of the exact address of each instruction. So before editing individual steps or sequences, be sure of the instruction at the current step address by pressing   (in the Program mode). This takes only a second, but it will assure you that the steps being changed (or deleted!) are the intended ones.

Branching Speed

Using step addresses in branching instructions generally enables the program to run much faster than when label names are used. This is because a "step address" branch goes **directly** to the specified address, while a "label" branch first goes to the top of memory, and then **searches** for the specified label.

To see the difference in branching speeds, first erase the program memory; then enter and run each of these programs:

(Using a step address)	(Using a label)
0000 CLEAR	0000 CLEAR
0001 FIX 9	0001 FIX 9
0003 GOTO 0300	0003 GOTO L00
0005 NOP	0005 NOP
0006 NOP	0006 NOP
•	•
•	•
•	•
0299 NOP	0299 NOP
0300 PAUSE	0300 LBL
0301 GOTO 0300	---- 00
0303 END	0302 PAUSE
	0303 GOTO L00
	0305 END

The program on the left produces a relatively constant display of ♦

0.00000000

The program on the right produces the same display, but now it is blinking. The blinking is caused by the extra time needed to search for label 00.

Branching Errors

Some unexpected results may occur if an incorrect branching address is used. For example, in each of the following sequences the GOTO indicates step 0200 but the branch is made to step 0199, since the specified step address is unavailable.

0050 •	0102 •
0051 GOTO 0200	0103 GOSUB 0200
0053 •	0105 •
•	•
•	•
•	•
0198 •	0198 •
0199 STO R005	0199 FIX 5
0201 •	0201 •

This is OK for program operation, but if you wish to enter a program sequence beginning at step 0200 (over one of the previous sequences) and press **GO TO** 200 from the keyboard

before setting  , the program counter would be automatically set to step 0199. In

this case, entering a NOP will make step 0200 available.

A more serious error occurs when a program branches into an alpha message, as in this sequence ♦

Executing alpha instructions without first setting the Alpha mode can cause unwanted results. In this case the calculator executes the instructions listed second. This second listing was made by setting the program counter to step 0300 (from the keyboard) and pressing

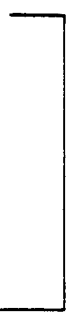
LIST

```

0248      •
0249 GOTO      0300
0251      •
      •
      •
0294      •
0295 PRNTα
0297 T
0298 0
0299 T
0300 A ←
0301 L
0302
0303 =
0304
0305 PRINT
0306 ENDα

0300 SFG      2
0301 IF SFG 5
0302 LOG
0303 SPACE
0304 LOG
0305 PRINT
0306 ENDα

```



Additional Alpha Characters

As described under "Editing Alpha", it's possible to print some characters not available via the standard Alpha mode. This is done by cancelling the Alpha mode and entering a regular instruction into an Alpha sequence. Then, running the entire alpha sequence prints the non-standard character within the alpha message.

Here is a list of the extra characters and the instruction to enter for each:

Character:	Instruction:
$\frac{1}{x}$	$\frac{1}{x}$
α	STO E
Σ	RCL ACC +
\div	IF CFG 4
\rightarrow	IF CFG 5
π	IF CFG 6
\downarrow	IF CFG 8

By using two of these additional characters, you can print this line ♦

by entering the program sequence on the right. Note that before entering each non-standard character, the Alpha mode must be cancelled by setting RUN and then PRGM. After entering each character, the Alpha mode

is reset by pressing  .









$X=Y \div 2\pi$

```
0000 PRNTα
0002 X
0003 =
0004 Y
0005 IF CFG 4
0006 2
0007 IF CFG 6
0008 ENDα
0009 END
```



After the entire sequence has been entered, listing it again will show the new characters ♦

```
0000 PRNTα
0002 X
0003 =
0004 Y
0005 ÷
0006 2
0007 π
0008 ENDα
0009 END
```

Peripheral CALL Instructions

The general key sequences  (n)  through  (n)  are reserved for use in controlling peripherals. Each interface currently in use defines the sequences as appropriate peripheral call instructions. For example, the sequence  1  executes (or programs) a PLOT instruction when an HP 98132A Plotter Interface is used. With an HP 98134A General I/O Interface, however, the sequence  2  executes (or programs) a WRITE instruction.

If you either execute or program any of those key sequences when the appropriate interface is **not** plugged in, the calculator will list the key sequence. The error message NO I/O DEVICE will appear, however, if the sequence is executed. For example, if the plotter interface is not plugged in:

Set  and .

Press  1 .

CALL 1A
NO I/O DEVICE

At times, it may be necessary to enter a program having peripheral call instructions when the appropriate interface is not plugged in. After the program has been entered, edited and listed, record it on tape. Then, when you wish to run the program, plug in the required interface(s), switch the calculator on, and load the program back into the calculator. Now the call sequences are defined with their appropriate instructions. Here are two program sequences showing call sequences listed with and without the plotter interface in use:

Without Interface	With Interface
0125 •	0125 •
0126 CALL 1A	0126 PLOT
0128 IF X<Y	0128 IF X<Y
0129 GOTO 0125	0129 GOTO 0125
0131 CALL 1I	0131 DRAW+
0133 CALL 1B	0133 PENT↑
0135 1	0135 1
0136 0	0136 0
0137 ENTER↑	0137 ENTER↑
0138 CALL 1D	0138 MOVE
0140 •	0140 •

4

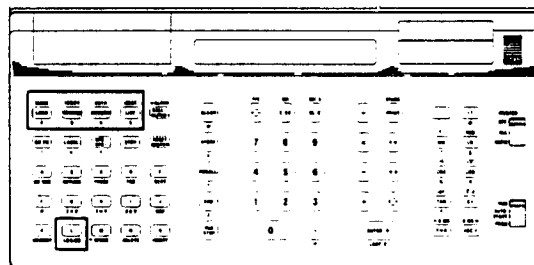
Tape Cartridge

Introduction

The built-in tape drive allows you to make permanent recordings of your programs and data, and then load each program or block of data back into the calculator as often as you wish.

The tape drive is also used to load pre-recorded programs into the calculator. If you do not wish to store your own programs and data on tape, you need only read the following introduction to the tape cartridge and cleaning the tape head. Then read "Loading Programs", beginning on page 84.

The tape drive keys are located with the programming keys. All tape-drive operations are programmable. A summary of the operations is on the next page.



Inserting a Cartridge

Tape Instructions


- MARK TAPE instructions are used to initialize a tape with blocks of empty files.
Enter: _____ and press

Size of each file (in steps) → Z

Number of files in block (≤ 254) → Y

first file number → X

To determine the file size needed for data, multiply the number registers by 8. To use the primary track use positive file numbers; to use the secondary track use negative file numbers. To erase the rest of the track, place a negative number in Y.

- To IDENTIFY a file, enter the file number in X and press . This information is placed in the stack:

File type → T

Steps in use → Z

File size → Y

File number → X

File Types:

0 = program file

1 = secured program

2 = data file

3 = special HP program¹4 = secured special HP program¹

5 = empty file

6 = extra file

- To RECORD PROGRAMS, enter: _____ and press

Beginning step address → Y

File number → X



The program memory is recorded into the file, until an END is recorded.

- To LOAD PROGRAMS, enter: _____ and press _____ or program

First step address → Y

File number → X



When the LOAD & GO instruction is used, the calculator runs the new program automatically.

- To RECORD DATA, enter: _____ and press

Number of registers → Z

First register number → Y

File number → X



- To LOAD DATA, enter: _____ and press

Starting register number → Y

File number → X



The data is loaded, register by register, beginning at the register specified.

¹See footnote on page 82.

- To VERIFY that the contents of a file is identical to that in the calculator memory,
enter: and press

Starting step address }
or } → Y
Starting register number }
File number → X



If the information in the file is not identical to that contained in the memory, "VERIFY FAILED" or "CHECKSUM ERROR" appears.

- To RECORD SECURE programs,

enter: and press

Starting step address → Y
File number → X



Once a secured program is loaded back in the calculator, any attempt to list, record, or edit programs will cause "SECURED MEMORY" to be printed. To clear the secured condition erase the memory or switch the calculator off.

The Tape Cartridge

The tape drive uses a specially-designed, miniature tape cartridge having about 140 feet (about 43 meters) of high-quality metal-oxide tape (see photo below). Typically, each cartridge can hold 48, 2008-step programs or the contents of 12,000 data registers. Space is provided on the top and front of each cartridge for you to label its contents.



The HP Tape Cartridge

The photo above shows two built-in protection features. As the cartridge is removed from the tape drive, a small plastic cover snaps closed to protect the tape from dust and dirt. Positioning the "RECORD" slide to the **left** protects the tape from accidental erasure. This slide should be positioned to the **right only** when initializing the tape or recording information.

Inserting the Cartridge

To insert a cartridge into the tape drive, first open the tape drive door. Then slide the cartridge into the drive, as shown on page 75, and close the door. It's recommended that you first press **REWIND** before removing a cartridge. This fully rewinds the tape to shield the recorded information against damage. Then press the eject bar.

CAUTION

Always remove the cartridge when it's not in use to protect the tape from dust.

Storing Tapes

As with most magnetic tape products, the information recorded in the tape cartridge can be altered or destroyed if it is exposed to a strong magnetic field, such as one produced by a bulk tape eraser, a toy magnet, or a metal detection device (like the ones used in many airports). If you keep your tapes in a metal container, such as a card index box, they will be protected from most magnetic fields.

Cleaning the Tape Head

To ensure the reliability of tape operations, it is recommended that the tape head be cleaned after every eight hours of tape operations. Furthermore, it's always a good idea to clean the tape head before making important recordings.



Cleaning the Tape Head

The tape head is easily cleaned as follows:

1. Press the eject bar and remove the tape cartridge.
2. Clean the tape head with a cotton swab that has been dampened with head-cleaning solution. Just wipe the top of the tape head a few times with the swab. Also, remove any dirt on the rubber drive-wheel, using the same cotton swab. Then remove dust that has accumulated in the vicinity of the tape head.
3. Re-insert the tape cartridge. Make sure that the door closes completely to prevent excess dust from accumulating.

In conclusion, special care must be taken to ensure reliable tape cassette operations:

- Keep the tape-drive door closed to prevent dust from accumulating. Also, remove the cartridge whenever it's not being used to protect the tape from dust.
- Clean the tape head after every eight hours of tape operations and before making important recordings.
- Protect the tape from scratches, dust and magnetic fields, like those associated with high-voltage electrical equipment. This includes the rear panels of calculator peripheral devices.

Also read "Tape Conditioning" on page 95.

Error Detection

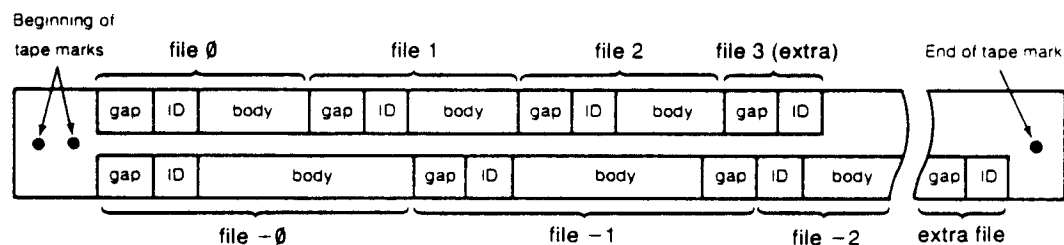
The tape drive routinely checks to ensure that all the information being loaded into the calculator corresponds exactly to that originally recorded. This is done by using a "checksum", which was stored on the tape when the information was first recorded. If an error is found during a LOAD DATA or LOAD PROGRAM operation, the file is automatically reloaded into the calculator. If the file cannot be loaded correctly after three tries, the operation is cancelled and "CHECKSUM ERROR" is printed. Typical causes for checksum errors are dirty, badly-worn, or partially-erased tapes or a dirty tape head.

A checksum error does not mean that the program loaded in the calculator is completely wrong – usually only a few steps are missing or changed. It may be easier to first list the program and then edit in the incorrect steps, rather than keying in the entire program.

Initializing the Tape

Tape File Structure

Before a new (blank) tape can hold programs or data, it must be initialized with one or more MARK TAPE operations. Each MARK TAPE instruction records a block of empty files onto one track of the tape. The user specifies the size of each file and the number of files in each block. Here is a typical tape file structure:



Tape File Structure

Tracks	Two tracks are available on each tape; each track can be initialized and used independently. To use the primary track, specify a positive file number in each tape-drive instruction; to use the secondary track, specify a negative file number in each instruction.
BOF Gap	The beginning of each file has a blank gap used to separate consecutive files.
File ID	The file identifier contains information such as: the file number, the file type (program,data,etc.), the absolute file size (in steps), the current file size (number of steps currently in use), etc.
File Body	This is where your program or data is stored. The absolute file size specified in the MARK TAPE instruction determines the size of this area.

Mark Tape




Each instruction initializes one track on the tape with a block of empty files. These values must be specified in the stack (the integer portion of each value is used):


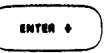


Size of each file	→ Z	
Number of files in block	→ Y	(max = 254)
Number of first file	→ X	

The file size is expressed in program steps. To determine the file size needed to hold a block of data (registers), multiply the number of registers by 8.

After the specified number of files have been marked, an extra file is marked and the tape is positioned in front of the extra file. Since the extra file is included to facilitate marking additional files at a later time, it has no body. Now you can store either data or a program in each file, or you can mark more files beginning with that extra file.

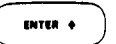



For example, to initialize a new tape with five 500-step files and two 300-step files on the primary track:

1. Set  and insert a new, unprotected tape cartridge.
2. Mark the first five files, beginning with file 0:

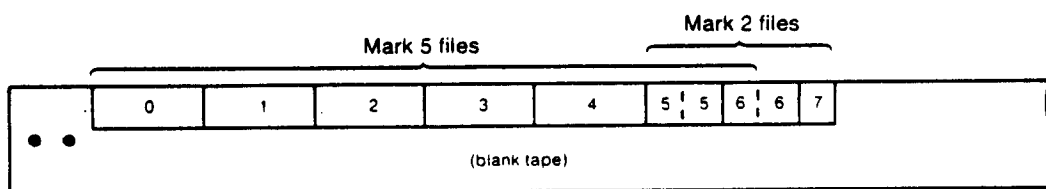
500  5 
 0  

After about 10 seconds the tape will be positioned in front of the extra file (file 5).

3. Now mark the two smaller files, beginning with file 5:





300  2 
 5  

After the tape drive stops, the tape is positioned in front of the new extra file (7). Now the tape is marked like this:

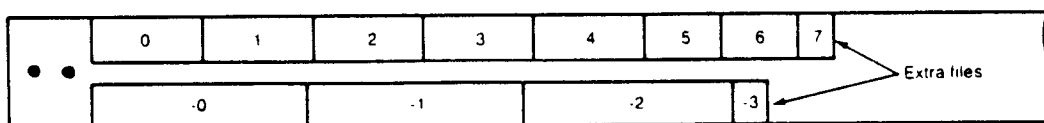


As shown above, the files appear in numerical order, from 0 through 7. The first file on each track must be either file 0 (primary track; or file -0 (secondary track).

Now let's mark three 1000-step files on the secondary track, numbering them files -0 through -2:

1000  3 
 0  

After the tape drive stops, the tape is marked like this:



Notice that each 1000-step file uses slightly less space than two 500-step files. This is due to the extra space needed by the beginning of file gap and file ID areas.

Marking a Used Tape

The MARK TAPE instruction is the same for both new and used tapes. But when marking a used tape, it is important to mark over, or erase, all old files which are no longer needed. This will prevent unexpected results, as described later. For example, to mark over files 3 through 7 in the last example with ten new 100-step files, press:



In this case all of the unwanted old files are erased by just marking the new ones.

But when a MARK TAPE operation will not erase all old files, the calculator can be instructed to quickly erase the rest of the track by specifying a **negative** number of files (in Y) for the MARK TAPE operation.¹ So, if you wished to replace all files on the primary track with one 500-step file, press:



After the new file 0 is marked, the calculator will automatically erase the rest of the track and position the tape at the new extra file (file 1).

Identify File



The IDENTIFY instruction transfers this ID information from the file indicated in X into the stack:

File Type	→	T
Steps in Use	→	Z
File Size	→	Y
File Number	→	X

File Types:
0 = program file
1 = secured program
2 = data file
3 = special HP program²
4 = secured special HP program²
5 = empty file
6 = extra file



For your convenience, the file ID information is printed automatically like this when the instruction is executed from the keyboard and the printer is switched on ♦

FILE	-3
TYPE	0
USED	47
MAX	200

¹The automatic erase operation may take up to one minute.

²These are programs which are written in a special language, loaded into data registers, and cannot be listed or modified by the user. To erase a special program, switch the calculator off.

Here's a short tape list program ♦

To run the program, enter the first file number
to be listed and press  .

```
0000 FIX      0
0002 STO      A
0003 IF -
0004 SFG      1
0005 IDENT
0006 ROLL↓
0007 PRTSTK
0008 SPACE
0009 1
0010 IF CFG 1
0011 STO+      A
0012 IF SFG 1
0013 STO-      A
0014 RCL      A
0015 GOTO      0005
0017 END
```

The program runs until the end of tape is
reached. Here's a sample printout ♦

A more-complete tape list program is available
on the Utility and Test Cartridge supplied with
your calculator. See the next page for user in-
structions.

```
file number      5
file type        0
steps in use     47
file size        500

              6
              0
             172
             500

              7
              5
              0
             500
```

Record & Load Programs

Recording Programs






This instruction records the contents of the program memory from the current step address until an END is recorded. Before executing the instruction, these values must be specified in the stack:

Beginning step address	→ Y
File number	→ X

If an END is not found, the rest of the program memory is recorded. If the file is too small or the tape is protected, the operation is cancelled (before the program is recorded) and an error message is printed.

For example, if the tape is still marked as shown on page 81 and you wish to record that 18-step tape list program just mentioned into file -2:

0  2  

To see if the program was recorded,

```
FILE    -2
TYPE    0
USED    18
MAX     100
```


Loading Programs






This key is used for loading both programs and data from tape into the calculator; the file type determines which operation is done. To load a program from a specified file into the calculator memory, these values must be in the stack:

First step address → Y
File number → X

The file is loaded, beginning at the specified step address. If the file is of the wrong type or there is not enough program memory available, the operation is cancelled and an error message is printed. See "Error Detection" on page 79.

For example, to load the tape list program from the Utility and Test Cartridge beginning at step 0000, first insert the test cartridge into the tape drive and press . Then press:


 40  

To verify that the program is loaded, run it.

```
**TAPE LIST**
```

```
FIRST FILE NO ?
```


Now enter the first file number to be listed and press  again. Here's a sample printout ▶

```

- 0
FILE NO.      -0
TYPE          0
IN USE        244
SIZE          500

FILE NO.      -1
TYPE          0
IN USE        95
SIZE          500

FILE NO.      -2
TYPE          3

```

When the LOAD instruction is executed in a program, it loads the new program starting at the specified address and then continues program execution at the next step.

For example, this program sequence loads the program in file 7 at step 0035. After the new program is loaded, the program continues from step 0020.

```

0014  ●
0015  3
0016  5
0017  ENTER↑
0018  7
0019  LOAD
0020  ●

```

Segmented Programs

The technique of recording program segments or subroutines in separate files and using a mainline program to call and run each segment, as needed, offers tremendous flexibility. This method enables one program to call hundreds of segments, each of which is loaded and run in the same area of memory.



The LOAD & GO instruction is a programmable means of automatically loading and running a specified program. The same values used for the LOAD PROGRAM instruction are used:

```

Starting step address  → Y
File number           → X

```

Here is a simple supervisory program that calls and runs a series of subroutines. The user entry at step 0017 determines which file is loaded first. Then each successive file is loaded in the same area of memory, beginning at step 0028.

The three subroutines called are shown below. The first two return control to the mainline program, but the last one ends the program.

File 1

```

0000 PRNTα
0002 F
0003 I
0004 L
0005 E
0006
0007 1
0008
0009 L
0010 O
0011 A
0012 D
0013 E
0014 D
0015 ENDα
0016 GOTO 0024
0018 END

```

File 2

```

0000 PRNTα
0002 F
0003 I
0004 L
0005 E
0006
0007 2
0008
0009 L
0010 O
0011 A
0012 D
0013 E
0014 D
0015 ENDα
0016 GOTO 0024
0018 END

```

File 3

```

0000 PRNTα
0002 F
0003 I
0004 L
0005 E
0006
0007 3
0008
0009 L
0010 O
0011 A
0012 D
0013 E
0014 D
0015 ENDα
0016 GOTO 0024
0018 END

```

```

0000 CLEAR
0001 FIX 0
0003 PRNTα
0005 L
0006 O
0007 A
0008 D
0009
0010 F
0011 I
0012 L
0013 E
0014
0015 ?
0016 ENDα
0017 STOP
0018 PRINT
0019 STO A
0020 2
0021 8
0022 RCL A
0023 LD&GO
0024 1
0025 STO+ A
0026 GOTO 0020
0028 END

```

Immediately after file 1 is loaded, the memory is arranged as shown below. Although this example shows loading program segments after the supervisory program, they could be loaded anywhere in the memory, even directly over the LOAD & GO step.

Supervisor			Subroutine		
0000	0		0022	PRNTα	
0000	CLEAR		0030	F	
0001	FIX	0	0031	I	
0003	PRNTα		0032	L	
0005	L		0033	E	
0006	0		0034		
0007	A		0035	1	
0008	D		0036		
0009			0037	L	
0010	F		0038	0	
0011	I		0039	A	
0012	L		0040	D	
0013	E		0041	E	
0014			0042	D	
0015	?		0043	ENDα	
0016	ENDα		0044	GOTO	0024
0017	STOP		0046	GOTO	0024
0018	PRINT		0048	END	
0019	STO	A			
0020	2				
0021	8				
0022	RCL	A			
0023	LD&GO				
0024	1				
0025	STO+	A			
0026	GOTO	0020			

As another example, this program sequence loads and runs either file 5 or 6, depending on whether flag 1 is set or clear. The file is loaded beginning at step 0000. Even if the new program is loaded over the LOAD & GO step, program control will still resume at the first step of the new program.

```

0199      •
0200 CLEAR
0201 IF SFG 1
0202 5
0203 IF CFG 1
0204 6
0205 LD&GO
0206 END

```

Linking Programs

An extremely long mainline program can be separated into parts, with each part being recorded into its own file. The LOAD & GO instruction can be added to the end of each part to call and run the parts in succession. An example is shown below.

Part A (file 0)

```
0466  •
0467 CLEAR
0468 1
0469 LD&GO
0470 END
```

Part B (file 1)

```
0460  •
0461 CLEAR
0462 2
0463 LD&GO
0464 END
```

Part C (file 2)

```
0470  •
0471 END
```

Each LOAD & GO instruction loads the next part directly over the old one and then transfers control to the new part. Each part has an END step to permit recording and loading, even though only the END in part C is executed.

Record & Load Data

Record Data



This instruction records the contents of a block of numbered data registers into a specified file. These values must be specified:



Number of registers	→ Z
First register number	→ Y
File number	→ X

The record operation begins with the first register number. If the specified registers are not assigned, or if the file is too small or of the wrong type, or if the tape is protected, the record operation is cancelled and an error message is printed. Also, if the number of registers (in Z) is less than 1, an error message appears.

Here are two example uses for the RECORD DATA instruction:

<p>Example #1</p> <pre> 0099 • 0100 1 0101 0 0102 ENTER↑ 0103 0 0104 ENTER↑ 0105 PRNTα 0107 F 0108 I 0109 L 0110 E 0111 # 0112 0113 ? 0114 ENDα 0115 STOP 0116 PRINT 0117 RCDATA 0118 END </pre>	<p>Example #2</p> <pre> 0000 STO A 0001 ROLL↓ 0002 STO B 0003 CLX 0004 STO C 0005 RCL C 0006 IDENT 0007 ROLL↓ 0008 STO D 0009 ROLL↓ 0010 ROLL↓ 0011 5 0012 IF X=Y 0013 GOTO 0019 0015 1 0016 STO+ C 0017 GOTO 0005 0019 RCL B 0020 8 0021 * 0022 RCL D 0023 IF X<Y 0024 GOTO 0015 0026 RCL B 0027 RCL A 0028 RCL C 0029 RCDATA 0030 PRNTα 0032 D 0033 A 0034 T 0035 A 0036 0037 I 0038 N 0039 0040 F 0041 I 0042 L 0043 E 0044 PRINT 0045 ENDα 0046 END </pre>
--	--

Example #1 records the contents of registers 0 through 9 into the file specified by the user entry at step 0115.

Example #2 records the contents of the registers specified in X and Y into the first available file on the primary track. To run the program, first insert a tape with at least one empty file of an adequate size to hold the data. Then enter the number of registers to be recorded in Y and the starting (lowest) register number in X and press  . the program searches the tape, beginning at file 0, until an empty file of adequate size is found. After the data is recorded, the file number is printed.

Load Data



This key is used for both LOAD DATA and LOAD PROGRAM instructions: the file type determines which operation is done. To reproduce a block of data into a specified area of the calculator memory, these values must be in the stack:

Starting register number	→ Y
File number	→ X

The data is loaded, register by register, beginning at the specified register. If the file is not type 2, or if there are not enough data registers available to hold the data, the operation is cancelled and an error message is printed. See "Error Detection" on page 79.

A programmed use of the Load Data instruction is shown here ♦

The user enters the number of registers to be established (Y) and the file number (X) at step 0099. The program then establishes the needed registers and loads the data, beginning at register 000.

```

0094  ●
0095  FIX      0
0097  CLX
0098  #REGS
0099  STOP
0100  X=Y
0101  #REGS
0102  CLX
0103  X=Y
0104  LOAD
0105  ●
  
```

Additional Operations

Verify



This instruction compares the information recorded on a file with the existing program or data in the calculator memory.

To verify a program file, these values must be in the stack:

Starting step address	→ Y
File number	→ X

To verify a data file, these values must be in the stack:

Starting register number	→ Y
File number	→ X

The VERIFY instruction is most easily done directly after a LOAD or RECORD operation, since the required values are already in the stack.

If the information in the file is not identical to that contained in the memory, the error message "VERIFY FAILED" or "CHECKSUM ERROR" is printed. Neither of these two errors will halt program operation when program flag 6 is set. In this case, program flag 5 is automatically set by either error.

An example use of the VERIFY instruction is this program segment ♦ which compares the data just recorded with the original data.

If the recorded data is identical to the original data, the file number is printed. But if a VERIFY FAILED or CHECKSUM error occurs, then flag 5 is set, the program loads the data into the next consecutive file, and the new file is checked.

0099	•		
0100	2		
0101	5		
0102	ENTER†		
0103	0		
0104	RCL	A	} record data in file "A".
0105	RCDATA		
0106	SFG	6	
0107	CFG	5	
0108	VERIFY		
0109	IF CFG	5	} If verify OK go to step 0116.
0110	GOTO	0116	
0112	1		
0113	STO+	A	} If verify failed, increment A and go to step 0105.
0114	GOTO	0105	
0116	RCL	A	} Print: DATA IN FILE X.
0117	FIX	0	
0119	PRNTα		
0121	D		
0122	A		
0123	T		
0124	A		
0125			
0126	I		
0127	N		
0128			
0129	F		
0130	I		
0131	L		
0132	E		
0133	PRINT		
0134	ENDα		
0135	•		

Secure Programs



The RECORD SECURE instruction is a means of recording private programming on tape. The instruction puts a program into a specified file, like the RECORD PROGRAM instruction, except that the program is identified as "file type 1" – secured. These values must be in the stack:

Starting step address	→ Y
File number	→ X

This operation does not affect the current programming in the memory; only the recorded program is secured. A secured program can be loaded back into the calculator by using a normal load instruction and then run in the normal way.

Once a secured program is in the calculator, any attempt to list, record, or edit the program will cause "SECURED MEMORY" to be printed. The user can run, but not see, a secured program.

NOTE

When a secured program is in the calculator, all other programming with it is also secured. Data registers, however, are not affected.

To clear this secured condition, either erase the memory (set the Program mode and press




) or switch the calculator off.

The Autostart Feature

When the mode switch is set to AUTOSTART, switching the calculator on automatically loads and runs the program in file 0. The Utility and Test Cartridge uses this feature to automatically load and run a directory of test programs. This allows you to select and run any test by pressing just one key.

Operating Hints

Halting the Tape Drive

In general, any tape-drive instruction can be halted by pressing  once. An alternate way to halt the drive immediately is to press the eject bar.

Tape Storage Capacity

The following table lists typical storage capacities for each track of an HP tape cartridge. Remember that each data register is equal to 8 program steps.

Typical Storage Capacities

Size of File		Typical Number of Files per Track	Tolerance
Program Steps	Data Registers		
8	1	254	+0 -5%
40	5	254	
80	10	254	
120	15	229	±10%
200	25	174	
320	40	135	
472	59	100	
800	100	64	
1200	150	44	
1600	200	34	
2008	251	27	
2600	325	21	
3000	375	18	
3800	—	14	

A program is available on the Utility and Test Cartridge to help you to determine your tape storage needs. To load the program, first insert the test cartridge and press **REWIND**. Then press:

CLEAR 4 1 **←** **LOAD**

To start the program, press **END** **RUN STOP**.

Now enter the number of files and file size for each block of files needed, pressing **RUN STOP** after each entry. The program computes and prints the typical percentage of tape (one track) needed for each block of files. A sample print-out is shown here ♦

TAPE CAPACITY

NO. OF FILES?

10

FILE SIZE?

500

FILES NEED

10.3% OF TRACK

NO. OF FILES?

255

(MAX= 254 FILES
PER TRACK)

NO. OF FILES?

28

FILE SIZE?

2000

FILES NEED

99.1% OF TRACK

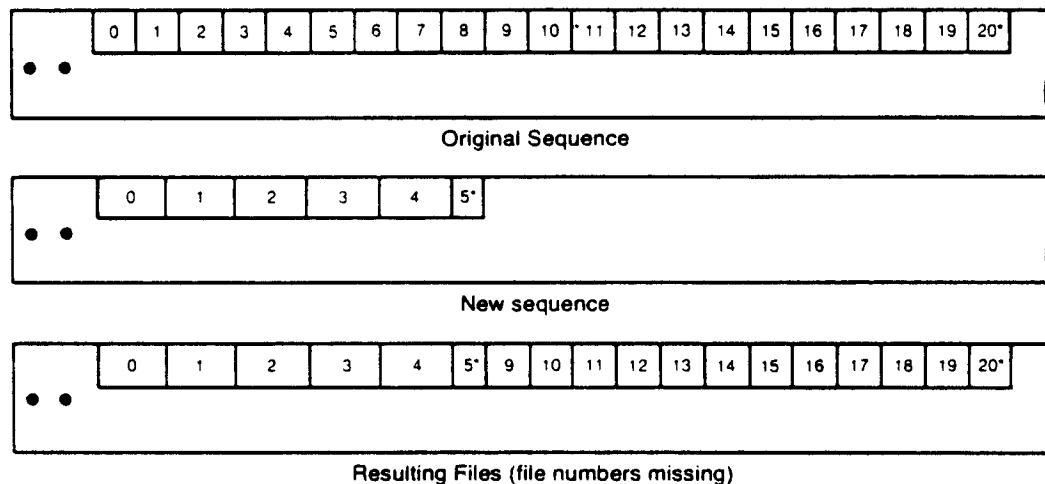
Master Recordings

Since a vast amount of information can be stored on a single tape cartridge, loss of the tape's contents (e.g., normal tape wear, physical damage, exposure to a strong magnetic field, or instrument failure) could be extremely expensive to the user ... both in time and resources lost. One method of preventing such a loss is to make a duplicate recording of each often-used or valuable tape, and then store the "master tapes" in a safe place. The practice of making master tapes should be considered mandatory when tapes are used on a daily basis!

Re-Marking Used Tapes

A tape which is not protected (the RECORD slide is positioned to the right) may be re-marked by performing the correct MARK TAPE operation (s). If possible, the track should be completely re-marked by using the auto-erase option (i.e., mark over **all** existing files), since, if only a portion of the track is re-marked, the old files which remain may cause problems.

Here are two situations that may occur when old files remaining on a track are not erased. The first diagram shows the file sequence resulting after a track, which originally contained 20 five-register files, is re-marked with 5 ten-register files. Notice that files 6 through 8 are now absent¹ and some of the old files (and their contents) remain.

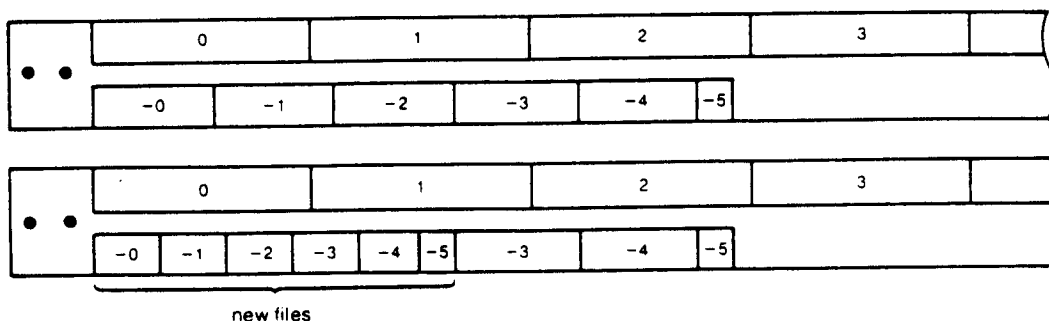


An attempt to use one of the missing files would cause the tape drive to search back and forth between files 5 and 9. When the missing file is not found after five passes, the operation is cancelled and "FILE NOT FOUND" is printed.

The next diagram shows the file sequence resulting after a track which originally contained 5 eight-register files is re-marked with 5 four-register files. Some file numbers on the track are now duplicated.

¹The exact number of missing files may vary.

In some cases this arrangement may cause the tape to go to an old file, rather than a new file of the same number. For example, if the tape is positioned at file 2, an instruction to use file -4 would use the old file -4!



Recording Special Functions

A complete set of special functions can be recorded into one file by placing an END after the last RETURN instruction in the last function and specifying the beginning step address of the first function in the RECORD PROGRAM instruction. Be sure that an END is used after the last function to be recorded, otherwise, the calculator will attempt to record the rest of the program memory – until an END is found.

Conditioning the Tape

Repeated operations over a short length of tape (usually less than 5 feet or 1.5m) can cause slack. (Extreme changes in temperature can also cause slack.) The outer layer of tape may slip and rub on the cartridge, causing damage to the tape. If this 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 the tape.



96 Cartridge

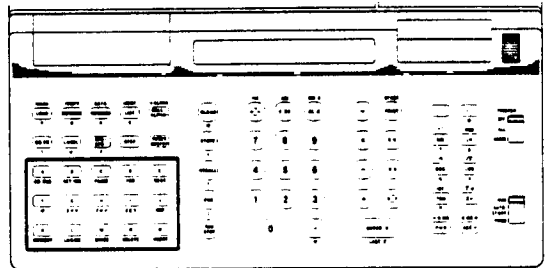
If a particular application requires such operation, this slack can be prevented by periodically moving the tape 15 feet or more toward mid-tape. For example, when using a tape with 80 files where only files 0 and 1 are used, the tape may be conditioned by executing the following program segment after every 200 operations between files 0 and 1:

```
0089      ●  
0090 8  
0091 0  
0092 IDENT  
0093 REWIND  
0094      ●
```

5

Special Functions

The lower-left block of keys,  through , can be defined to call and run your own special functions. A summary of special functions is listed below.



- Each special function is actually a subroutine beginning with a label name (A through O) and ending with a RETURN. Blank key-overlays are provided for you to identify each defined function.
- To run each defined function from the keyboard, press the corresponding key.
- To call the function as a subroutine from a program, use a GOSUB.
- Special functions can be nested seven deep. "GOSUB OVERFLOW" appears when an eighth nesting level is attempted.

Defining Functions

Each special function key is defined by entering the function steps in the program memory. Each function must begin with the key's corresponding label name and end with a RETURN instruction. Any programmable instruction can be used in special functions.

For example, the three hyperbolic functions shown below can easily be defined as special functions.

$$\text{Function A} = \sinh(X) = \frac{e^x - e^{-x}}{2}$$

$$\text{Function B} = \cosh(X) = \frac{e^x + e^{-x}}{2}$$

$$\text{Function C} = \tanh(X) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Each function can be stored anywhere in the program memory. Here are the functions, together, beginning at step 0300:

Function A				Function B			
0300	LBL			0312	LBL		
----	A			----	B		
0302	STO	A		0314	STO	A	
0303	e↑X			0315	e↑X		
0304	RCL	A		0316	RCL	A	
0305	+÷-			0317	+÷-		
0306	e↑X			0318	e↑X		
0307	-			0319	+		
0308	2			0320	2		
0309	÷			0321	÷		
0310	PRINT			0322	PRINT		
0311	RETURN			0323	RETURN		

Function C			
0324	LBL		
----	C		
0326	ENTER↑		
0327	e↑X		
0328	STO	A	
0329	X÷Y		
0330	+÷-		
0331	e↑X		
0332	STO	B	
0333	-		
0334	RCL	A	
0335	RCL	B	
0336	+		
0337	÷		
0338	PRINT		
0339	RETURN		

Running Functions

To execute each special function from the keyboard, enter the appropriate values in the stack and press the assigned special function key. The calculator will automatically go to the label and run the function. When the RETURN is executed, the calculator will reset the program counter to its original address and halt.

For example, to execute each of the hyperbolic functions from the keyboard, enter a value in X and press the assigned key.

First calculate $\sinh \pi$ and display the result to four places:



Now calculate $\tanh 1.07$:



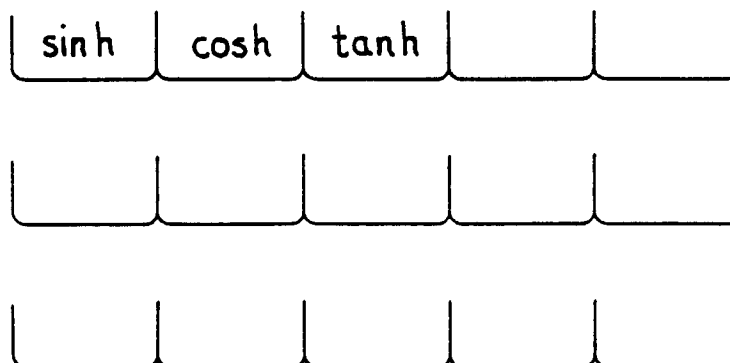
Defined functions can also be called from a program as a subroutine. For example, to call the $\sinh(X)$ function from step 0201 of a program, use one of these sequences:

0200	•	0200	•
0201	GOSUB A	0201	GOSUB 0300
0203	•	0203	•

In each sequence, the program will return to the next step after the function is executed. If a GOTO is used to call a function, however, the function will be executed, but the RETURN at the end of the function will cause "MISSING GOSUB" to appear and halt the program.

Key Overlays


The key overlays supplied with the calculator can be used to identify defined special-function keys. You can write on each overlay using either a pencil or a ball-point pen. Then place the overlay over the function keys when the functions are in the memory. A key overlay identifying the three hyperbolic functions defined earlier is shown below.



To order additional blank key overlays, specify HP part number 7120-4691.

Nesting Special Functions

Special functions called from the keyboard can be nested seven deep; that is, one function can call another function, which calls a third function, etc. If an eighth function is called, however, without first returning to at least the sixth function, the error message "GOSUB OVERFLOW" appears and the program halts.



Before calling functions which use nesting, press  to ensure that the nesting level is set to zero. This is because the same device used to remember subroutine return-addresses is used when special functions are nested.

Special functions called from a program are treated exactly like subroutines and can be nested, together with subroutines, to a depth of seven. Again, pressing or executing END resets the nesting level to zero.

A Program Directory

The following pages show how the special function keys can be used to create a convenient "program directory" – a means of indexing or calling a program on tape by pressing a single key.

A part of a directory program which shows the basic sequence for calling programs is shown here ♦

After this program is loaded,  is defined to load and run the program in file 1,  is defined to load and run file 2, etc. Each function loads the new program over the directory, beginning at step 0000. A key overlay for these functions could be used to identify the program available via each key.

```
0000 PRNTα
0002 D
0003 I
0004 R
0005 E
0006 C
0007 T
0008 O
0009 R
0010 Y
0011
0012 L
0013 O
0014 A
0015 D
0016 E
0017 D
0018 ENDα
```

(con't)

The alpha message "DIRECTORY LOADED" and the STOP instruction are included because the directory could be recorded on file 0 and automatically loaded and run using the AUTO START mode.

```

0019 STOP
0020 LBL
---- A
0022 CLEAR
0023 1
0024 LD&GO
0025 RETURN
0026 LBL
---- B
0028 CLEAR
0029 2
0030 LD&GO
0031 RETURN
0032 •

```

Load and Run file 1 {

Load and Run file 2 {

This method of calling programs has one disadvantage: the subroutine nesting level is set one deep as the function is started; and since the RETURN is not executed, the level remains one deep when the new program is loaded. This limits the new program to only six levels of nesting.

An alternate method of writing functions permits calling a function **without** changing the nesting level. To do this, begin each function with two consecutive LABEL instructions and omitting the RETURN. Here's an example ♦

Now, when A is pressed, the calculator initially remembers the current step address but erases it when the consecutive labels are executed; thus, the nesting level remains unchanged. This latter method is used in many of the prerecorded programs available from HP.

```

0150 •
0151 STOP
0152 LBL
---- A
0154 LBL
---- A
0156 CLEAR
0157 1
0158 LD&GO
0159 LBL
---- B
0161 LBL
---- B
0163 CLEAR
0164 2
0165 LD&GO
0166 LBL
---- C
0168 LBL
---- C
0170 CLEAR
0171 3
0172 LD&GO
0173 •

```


Appendix 1

Installation Procedure

This appendix contains inspection and installation procedures for your calculator. Procedures on installing other parts of your calculator system are described in the manuals furnished with each peripheral or interface.

Inspection Procedure

Each part of your calculator system was carefully inspected before it was shipped to you. All equipment should, therefore, be free of scratches and should operate properly. Carefully inspect the calculator, peripheral equipment, cables, etc., for 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. Decals located under the calculator's top-cover show the number of any internal option installed. Also check to ensure that all accessories are present (refer to "Equipment Supplied").

If you wish to check the operation of all or part of your system, refer to Appendix 3. Before running any test, however, be certain that your calculator is properly installed.

If you have any difficulties with your system, if it is not operating properly, or if any items are missing, contact your nearest HP Sales and Service Office; addresses are supplied in Appendix 3.

Equipment Supplied

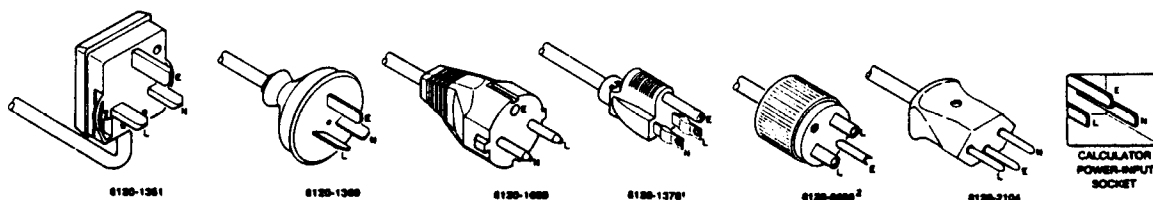
The following items are packaged with each HP 9815A Calculator. Peripheral devices and interface cards are packaged separately; each of these has its own manual or operating note and may also have extra items packaged with it.

Standard Calculator Accessories

DESCRIPTION	QUANTITY	HP PART NUMBER
Operating Manual	1	09815-90000
Beginner's Guide	1	09815-90001
Quick Reference	1	09815-90010
Utility and Test Cartridge	1	09815-10004
Data Cartridge (blank)	1	9162-0061
Printer Paper	3 rolls	(see below)
Program Pad	1	09815-90011
Key Overlays (blank)	6	7120-4691
Utility Programs	1	09815-10000
Power Cord	1	(see next figure)
Spare Fuses:		
1.5 A normal blo	1	2110-0043
.75 A normal blo	1	2110-0033
Dust Cover	1	9222-0490
Tape Head Cleaner	1 can	8500-1251

When ordering paper specify six-roll packs, HP part number 9270-0479.

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.



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

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

Power Requirements

The 9815A Calculator has the following power requirements:

- **Line Voltage:** The calculator operates from nominal powerline voltages of 100, 120, 200, and 240 Vac. The range of operation is from –10% to +5% of each nominal voltage. Two switches on the rear panel enable any one of the four voltages to be selected.
- **Line Frequency:** The calculator can be operated with any line frequency from 48 Hz to 66 Hz (nominally 50 Hz and 60 Hz).
- **Power Consumption:** The calculator requires a maximum of 75 voltamps.

Grounding Requirements

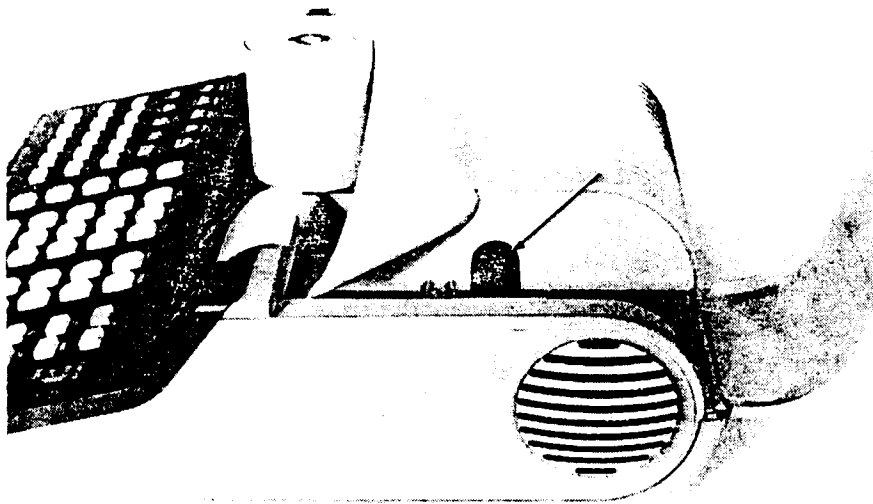
To protect operating personnel, the National Electrical Manufacturers' Association (NEMA) recommends that the calculator chassis be grounded. The calculator is equipped with a three-conductor power cable which, when connected to an appropriate power receptacle, grounds the chassis. To preserve this protection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

Fuses

The calculator fuse is located under the top cover, as shown in the next figure. The fuse is either a 1.5A fuse for 100 or 120 Vac operation, or a .75A fuse for 220 or 240 Vac operation.

WARNING
TO AVOID THE POSSIBILITY OF SERIOUS INJURY, ALWAYS DISCONNECT THE CALCULATOR FROM ITS POWER SOURCE BEFORE CHANGING A FUSE.

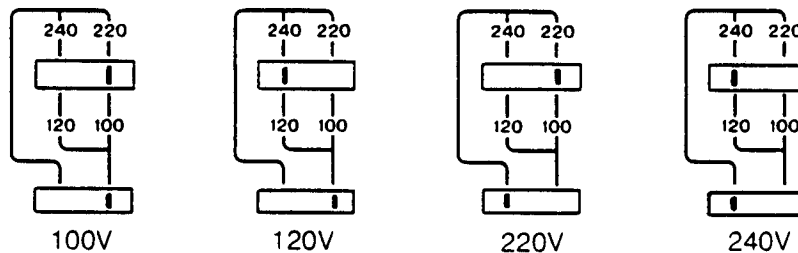
To change a fuse, first disconnect the power cable from the calculator. Then remove the roll of printer paper and press inward on the fuse-holder cap while twisting it counterclockwise. Now remove the cap and fuse, remove the fuse from the cap, and insert the replacement fuse (either end) into the cap. Finally, put the fuse and cap back into the fuse holder, press on the cap, and twist it clockwise until it locks in place.




The Calculator Fuse

Initial Turn-On


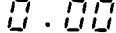
1. With the calculator disconnected from its ac power source, verify that the correct calculator fuse has been installed for the powerline voltage in your area (refer to the previous section).
2. Next ensure that the two switches on the rear panel are set for the correct powerline voltage. The figure below shows the correct settings for each nominal line voltage. If it is necessary to alter the setting of either switch, use a screwdriver to slide the switch so that the positions of the switches correspond to the desired voltage, as shown in the second figure.



Switch Settings for the Nominal Powerline Voltages

3. Set the power switch, located on the front of the calculator, to the  (off) position.

Connect the power cord to the power input connector on the rear panel; then plug the other end of the power cord into a suitable ac power outlet.

Switch the calculator on ; the display  indicates that the calculator is ready to operate. If any other display appears, or if there is no display, switch the calculator off and turn to Appendix 3, "Service".

Appendix 2

9815S Supplemental Information

Introduction





The Hewlett-Packard 9815S is an HP 9815 calculator with 4K bytes of memory space and two I/O ports. This appendix explains the operating differences between the 9815S and the 9815A. It is assumed that you have already read the main text of this manual and are familiar with the statements used by the 9815 calculator family.

Programs

Program Storage

The HP 9815S provides a maximum of 3800 program steps. The memory capacity can be verified by performing the following operations:

1. Disconnect all peripheral interfaces and turn on the calculator.

2. Press:    

3. Select:

RUN
AUTO START
PRGM

4. The display should indicate that 3800 steps of program memory are available.

Branching and Subroutines

Branching instructions can be used to reach any of the 3800 program steps. However, the instruction GOTO $\left(\begin{smallmatrix} \text{step} \\ \text{address} \end{smallmatrix} \right)$ can only branch to addresses **below** 2048. To branch to address 2048 and above, use any of the following three instructions. These instructions have no address limitations.

GOTO LABEL $\left(\begin{smallmatrix} \text{label} \\ \text{name} \end{smallmatrix} \right)$
 GOTO X
 GOTO LABEL X

The GOSUB instructions have the same addressing characteristics as the GOTO instructions just described.

Editing

Setting the Program Counter

The program counter can be set to any of the 3800 program steps for editing purposes. Pressing $\text{GO TO} \left(\begin{smallmatrix} \text{step} \\ \text{address} \end{smallmatrix} \right)$ works only for addresses 0000 through 2047. The program counter can be set to addresses above 2047 by using any of the following techniques.

1. Key in the desired step address and press $\text{GO TO} \text{ (X)}$.
2. Pressing $\text{GO TO} \text{ (LABEL)} \left(\begin{smallmatrix} \text{label} \\ \text{name} \end{smallmatrix} \right)$ sets the program counter to the address of the step immediately following the specified label.
3. Key in the desired label number and press $\text{GO TO} \text{ (LABEL)} \text{ (X)}$. The program counter is set to the address of the step immediately following the specified label.

Automatic Adjustment of Addresses

It is recommended that you use labels for all GOTO and GOSUB instructions except when speed is critical. First of all, this is good programming technique. When you have only a short program with a few branches, it is not difficult to keep track of line numbers. But the extended program memory of the HP 9815S makes it possible to create very large, complex programs. Experience has shown that GOTO <line number> type instructions are a frequent source of error in large programs, especially during program revision or modification. Problems of this type are reduced substantially by the use of labels.

Additionally, the GOTO $\left(\begin{smallmatrix} \text{step} \\ \text{address} \end{smallmatrix} \right)$ and GOSUB $\left(\begin{smallmatrix} \text{step} \\ \text{address} \end{smallmatrix} \right)$ instructions can cause a problem for the editor in the HP 9815S. This editor recomputes the destination address of these GOTO and GOSUB instructions whenever you do an "insert line" or "delete line" operation. However, the upper address limit of the editor is step 2047. For example, if you have a program with a GOTO 2045 and you insert five lines after this step, the editor will produce only a GOTO 2047. It will **not** produce a GOTO 2050, and it will **not** generate an error message to inform you of the problem. GOTO X instructions are not the answer either, because the internal editor never modifies or adjusts the contents of the X-register. The simplest and most general solution to potential editing problems is the use of labels.

Data

Standard Data Storage

The HP 9815S provides a full set of 256 numbered data registers (000 through 255), along with the 10 registers A through J, and still has up to 1752 steps of program storage. The registers 000 through 255 and A through J are assigned and accessed using the standard procedures described in the "Number Storage" section of this manual. Any time a **specific register number** is used in an instruction, that register number **must** be within the range of 000 through 255.

Extended Data Storage

The purpose of the register allocation in the HP 9815S is not to provide a significantly greater amount of data storage than the HP 9815A Option 001, but to provide enough program space to make meaningful use of the available registers. However, an added feature of the HP 9815S is the capability to assign more numbered data registers than the 256 which can be directly accessed. A maximum of 474 numbered data registers can be assigned without generating an error. Access to the upper 218 data registers is limited to certain instructions available with the 98135A interface (HP-IB). These instructions specify data locations with two parameters: "register number" and "byte number". Although the "register number" parameter has a maximum limit of 255, there is no upper limitation on the "byte number" parameter. Therefore, HP-IB instructions of this type can reach anywhere in data memory.

Split Register Storage

The 9815 Utility Routines cartridge contains a special program which allows the storage of two data items in one data register. The 9815S uses a different special program file for this function than the 9815A. If you are using the split register demonstration program on file 45, tell the calculator that you have "4K", and it will load the proper file for you. If you are using split register storage as a subroutine to your own programs, load the special program on file 48, rather than file 46. The null program used to clear the special program registers is on file 49, rather than file 47. Except for the different file numbers, all other operating details of the split register storage program are the same as described in the Utility Routines manual. You can use 256 numbered data registers for split precision storage on the 9815S.

Appendix 3

Service

This appendix describes using the Utility and Test Cartridge (HP Part No. 09815-10004) for checking operation of your calculator. Also included here are maintenance procedures which should be done periodically, and lists of peripheral tests and example programs available on the test cartridge.

Service Contracts



Service contracts are available for all HP calculators and calculator-related equipment. For further information contact any HP sales and service office.



Calculator Tests

The Utility and Test Cartridge supplied with your calculator contains a complete set of calculator test programs. If any of these tests fails after repeated tries, your calculator requires service – contact an HP sales and service office for assistance. Office locations are listed on pages 119-120.

To load and run the test program directory:

1. Open the tape drive door;
2. Slide in the cartridge, so that the label is right-side-up and facing you;

3. Set  and 

Switch the calculator off  and on .

AUTO START
09815-10004
REV 0

UTILITY & TEST
CARTRIDGE

FOR CALC. TESTS
PRESS (1) (RUN)

FOR UTILITY
ROUTINES
PRESS (RUN)

4. To run the test programs, press:



9815A CALCULATOR
TEST EXERCISER
PROGRAMS

5. If the calculator has 3800 program steps (9815S), press:

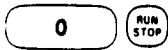


MEMORY SIZE?
0=472
1=2008
2=3800

If the calculator has 2008 program steps (option 001), press:



If the calculator has 472 program steps (basic memory), press:



PRESS ONE OF
THE FOLLOWING
KEYS.

6. After this directory is printed ♦ select the test by pressing the indicated key. Then compare your results with the following descriptions. Typical running times are listed with each.

KEY	TEST
A	DISPLAY
B	PRINTER
C	KEYBOARD
D	ROM
E	MEMORY
F	COMPLETE

Each test, except the Memory test, runs once and halts. The Memory test runs continuously until the calculator is switched off.

If you wish to run any test (except "Keyboard") continuously, press



1


before pressing the indicated key.

NOTE

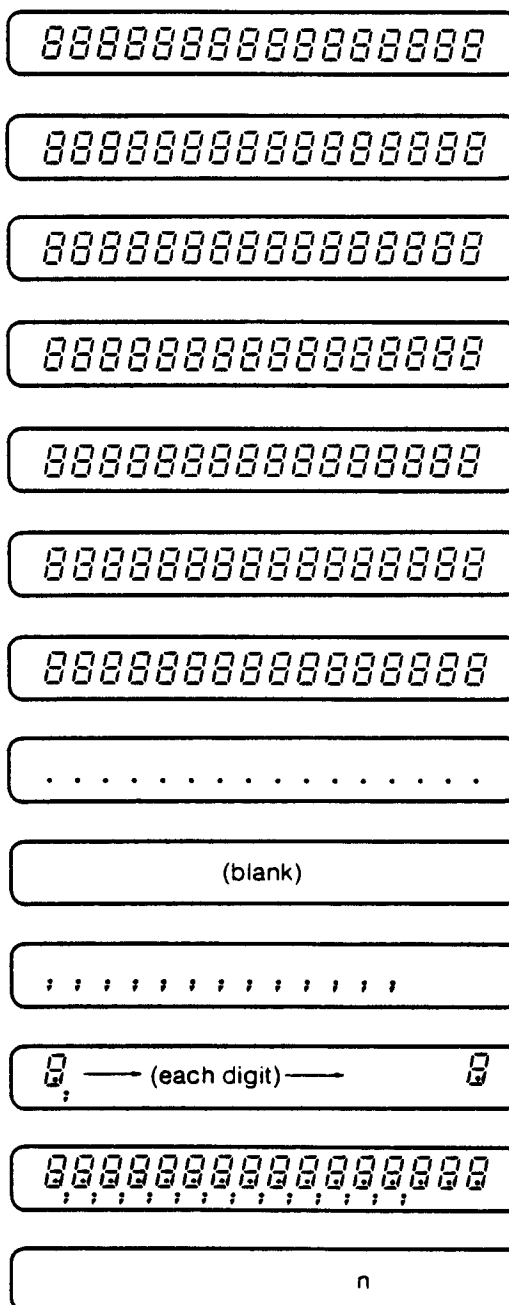
If MEMORY OVERFLOW is printed after the directory, the wrong memory size was indicated in step 5. Return to step 3 and reload the directory program.

Display Test (30 seconds)

Run this test to check each display digit and segment. The display sequence is shown below. The test is complete when 0 (n) is displayed.


When the test is run continuously, n indicates the number of completed tests. Press  once to halt the test.

(start)




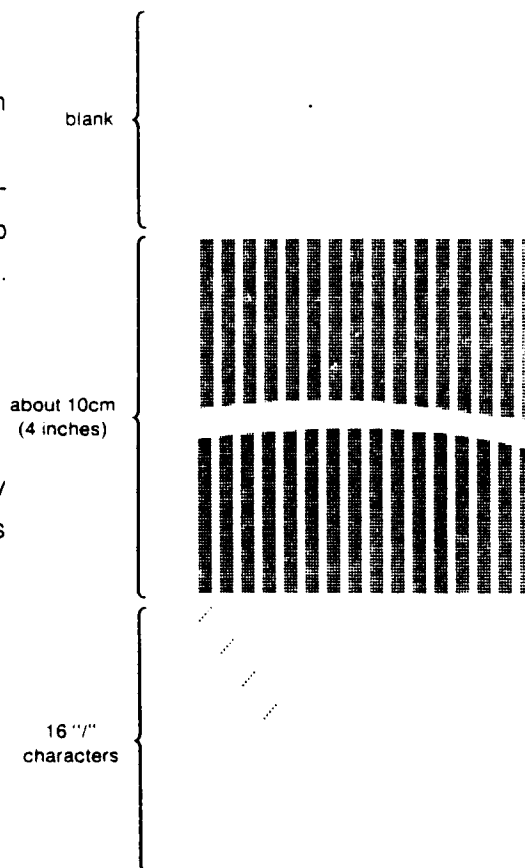
(end)

Printer Test (25 seconds)


Compare your printout with the sample shown here 

Check for missing or extra dots (the print density will not be completely uniform). Also check the length of the solid block of dots. When 0 is displayed, the test is complete.

When the test is run continuously, the display indicates the number of completed tests. Press  once to halt the test.

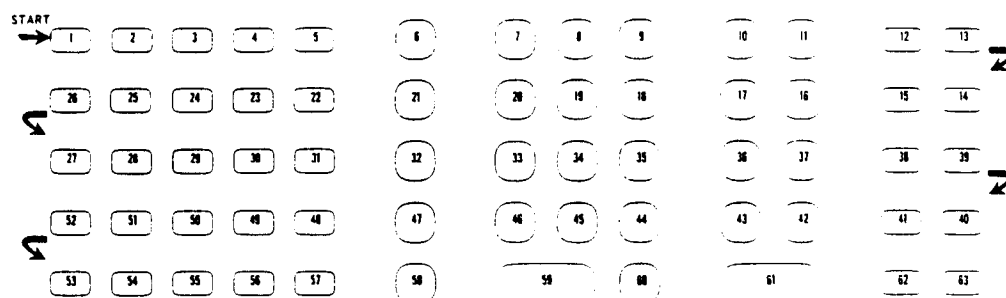


Keyboard Test (1 minute)

This test checks each key and switch on the calculator keyboard. After pressing , wait for the tape drive to halt. Then set:

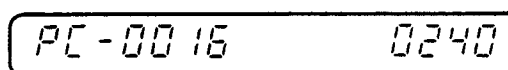



After 1 is displayed, press each key once in the order shown by the next figure. The display increments as each key is pressed, indicating the next key to be pressed. If a key is pressed out of sequence, or if a key is defective, FAIL n is printed. n indicates either the correct key to press or the defective key. And, if FAIL n is **not** printed when a key is pressed continuously, that key is defective.



Keyboard Test

The final display is ♦



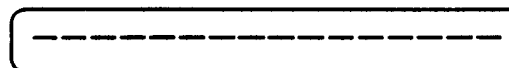
Now set  and the test is complete.


If FAIL n is printed repeatedly, restart the test by switching the calculator off and return to step 3 on page 111.

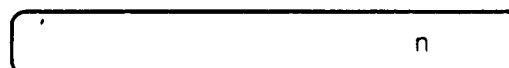
ROM Test (3 seconds)

This test checks all read-only-memories in the calculator and in any interfaces plugged in.

The correct displays are ♦



When the test is run continuously, n indicates the number of completed tests. Press  once to halt the test.




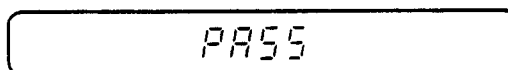
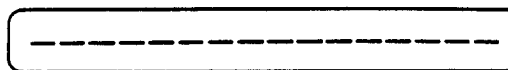
If FAIL n is displayed, switch the calculator off, disconnect all interfaces and run the test again (return to step 3 on page 111). If the test still fails, the calculator requires service. However, if a failure occurs only when an interface is plugged in, that interface is probably defective.

Memory Test (5 seconds)

This test checks every step of the program memory. The correct displays are ♦

When PASS is displayed, the test is complete.

This test runs continuously until either  is pressed or the calculator is switched off. To run another test, reload the directory (see step 3 on page 111).



If FAIL n is displayed, the calculator requires service.

Complete Exerciser (1 minute)

This key automatically runs the Display, Printer, ROM and Memory tests, one at a time (see the foregoing descriptions). Then the Memory test is automatically repeated until the calculator is switched off.

Cleaning the Calculator

The calculator can be cleaned with a soft cloth dampened either in clean water or in water containing a mild detergent. Do not use an excessively wet cloth or allow water to get inside the calculator. Also, do not use any abrasive cleaners, especially on the display windows or the tape-drive door!

The air filter should normally be cleaned about every three months. To clean the filter, first turn the calculator off. Then remove the filter and either hold it under running water, or wash it in warm, soapy water, followed by a rinse in clean water. Dry the filter thoroughly before replacing it.

Cleaning the Tape Head

The tape head in the tape drive should be cleaned after every eight hours of tape operation. A procedure is described on page 79.

Loading Paper

The printer uses special heat-sensitive paper. When ordering paper specify HP Part No. 9270-0479.

To load a roll of paper:

1. Lift the calculator top cover and remove the metal spindle. When replacing an old roll, discard the old paper core and remove any paper left in the printer.
2. Install the new roll as shown below.
3. Insert the free end of the paper as shown and advance it slowly through the printer using the thumb wheel.

If the paper catches and folds up under the printer window, remove the window (see below) and straighten the paper into its intended position. Then replace the window.



Loading Paper



Removing the Printer Window

Peripheral Tests

The Utility and Test Cartridge has programs for checking operation of each calculator peripheral and its interface card. If any procedure cannot be completed successfully after repeated tries:

1. Disconnect the interface(s) from the calculator and verify calculator operation by running the ROM and Memory Tests (see page 111).
2. Connect one peripheral interface at a time and run the associated test.
3. When you find the test that fails repeatedly, contact the nearest HP sales and service office for assistance (see the following pages).

Here is a list of peripheral tests, showing the manual in which user instructions are described.

Peripheral Test:	File:	For Instructions See:
HP 9862A Plotter:		98132A Plotter Interface
Accuracy Test	-12	Operating Manual
Performance Test	-13	
HP 9863A Tape Reader	-17	Operating Note, P/N 98134-90011
HP 9864A Digitizer	-18	Operating Note, P/N 98134-90012
HP 9866A Thermal Printer	-19	Operating Note, P/N 98134-90013
HP 9871A Printer	-21	98131A Printer Interface Operating Manual
HP 9883A Tape Reader	-23	Operating Note, P/N 98134-90014
HP 9884A Tape Punch	-24	Operating Note, P/N 98134-90015

Example Programs

Some of the example programs described in this manual, and in the interface operating manuals, are recorded for your convenience on the Utility and Test Cartridge. Here is a list of the example programs and the page (or other manual) where user instructions are given.

Program:	File:	For Instructions See:
Tape List	-40	page 84
Tape Capacity	-41	page 93
Calendar	-42	page 49
Combinations (N!)	-43	page 56
HP 9862A Plotter:		98132A Plotter Interface
Y = X ² Plot	-45	Operating Manual
Lettering Program	-46	
Sin X/X Plot	-47	
Sample Data for Digitizing	-48	
Complete Digitizing	-49	
HP 9864A Digitizer:		Operating Note, P/N 98134-90012
Document Alignment	-50	
Basic Digitizing	-51	
Line Length and Area	-52	
HP 9866A Thermal Printer:		Operating Note, P/N 98134-90013
Math Table	-55	
HP 9871A Printer:		98131A Printer Interface
Math Table	-56	Operating Manual
Y = X ² Plot	-57	
Sin X/X Plot	-58	

Appendix 4

Operating Limits

Calculating Range

The dynamic range is from $-9.999999999 \times 10^{99}$ through $9.999999999 \times 10^{99}$. Whenever a result exceeds that range the error message "OVERFLOW" is printed. (Calculations which normally result in zero, such as subtracting a number from a number equal to itself, do not exceed the range of the calculator.)

Error Messages

Improper key or program sequences and incorrect calculations are indicated by printed error messages. A complete list of error messages is on the inside-back cover.

An error message causes the program to halt, unless it is indicated as a "suppressable" error and program flag 6 is set. For more information see "Flags" in Section 3.


Accuracy Limits

All calculations are computed to 12 significant digits, however, the accuracy depends upon the actual operation. Arithmetic operations (+, -, × and ÷) are accurate to within one count in the 12th (least significant) digit.

Guard Digits


All calculations use 12 significant digits and a two-digit exponent. Up to the 10 most-significant digits can be displayed and printed. The two least-significant digits are called "guard digits".

The purpose of the guard digits (which should not be confused with the exponent) is to maintain maximum accuracy during calculations and also to automatically round the tenth digit for displayed and printed numbers. The next example shows how to view the guard digits.

When  is pressed, the value for pi is entered with 12 digits, although only the first ten are displayed. The last displayed digit is rounded up, from 3 to 4, because the guard digits contain 60:

Press   

By subtracting 3.1 from pi, the guard digits can be seen:

3.1 

3.141592654 00
 digit rounded exponent

4.159265360 -02
 former guard digits

Environmental Limits

Temperature range: 5° C (41° F) to 40° C (104°) ambient.

Humidity: 80%, relative.

Tape Specifications

Tape Cartridge

Usuable tape length: 43m (140 feet) ¹

Tape capacity: 96,000 bytes (program steps) or 12,000 data registers. ¹

Dimensions: 63.5mm × 82.5mm × 12.7mm (2.5 inches × 3.25 inches × 0.5 inches).

Tape Drive

Search speed: 1,524mm/sec (60 inches/sec), bidirectional.

Read/Write speed: 254mm/sec (10 inches/sec).

¹Typical values are given.

Instruction-Key Index

Instruction	Keys	Listing	See Page
Clear Stack		CLEAR	3
Clear X		CLX	3
Clear Registers A through J		CLRA+J	12
Assign Numeric Registers		#REGS	12, 16
Enter		ENTER↑	6
Arithmetic		+ - * ÷	4
Exchange X & Y		X↔Y	} 6
Roll Stack Up		ROLL↑	
Roll Stack Down		ROLL↓	
Print X		PRINT	3
Print Stack		PRTSTK	6
Print Alpha	(alpha keys)	9999 PRINT* 9999 END*	28, 67, 71
Peripheral calls	(n) (A → O)	CALL 2A	73
Recall Last X		LSTX	6
Set FIX Format	(n)	FIX 0	} 10
Set SCI Format	(n)	SCI 0	
Set SCI 3 Format	(n)	SCI3 0	
Set Degrees	1	DEGS	} 18
Set Radians	2	RADS	
Set Grads	3	GRADS	
SIN X		SIN	} 19
ARC SIN X		ASIN	
COS X		COS	

Instruction	Keys	Listing	See Page
ARC COS X	π° COS	ACOS	} 19
TAN X	TAN	TAN	
ARC TAN X	π° TAN	ATAN	
Enter pi	π	π	21
Log e	LN	LN	} 20
e^x	π° LN	$e^{\uparrow}X$	
Log ₁₀	LOG	LOG	
10^x	π° LOG	$10^{\uparrow}X$	
Summation +	Σ^+	SUM+	} 25
Summation -	π° Σ^+	SUM-	
Basic Stat	π° Σ^+	MN&SD	
Accumulate +	ACC+	ACC+	} 24
Accumulate -	π° ACC+	ACC-	
Recall Total	RECALL ACC+	R ACC+	
Polar \rightarrow Rectangular	P \leftrightarrow R	P \rightarrow R	} 22
Rectangular \rightarrow Polar	π° P \leftrightarrow R	R \rightarrow P	
Decimal Degrees \rightarrow D.MS	\rightarrow D.MS P \leftrightarrow R	\rightarrow D.MS	
D.MS \rightarrow Decimal Degrees	D.MS \rightarrow ACC+	D.MS \rightarrow	} 21
Y^x	π° SIN	$Y^{\uparrow}X$	
$1/x$	π° COS	$1^{\uparrow}X$	
Integer X	INT TAN	INT	} 12
Round X	ROD LN	ROUND	
\sqrt{x}	\sqrt{x} LOG	SQRT	
Store in register n	STORE (n)	STO A	} 12
Recall from register n	RECALL (n)	RCL +005	
Indirect Store	STORE RECALL (n)	STO (R000	

Instruction	Keys	Listing	See Page
Indirect Recall	(n)	RCL I B	} 12
Register Arithmetic	$\left\{ \begin{array}{c} + \\ - \\ \times \\ \div \end{array} \right\} (n)$	STO+ D	
Indirect Register Arithmetic	$\left\{ \begin{array}{c} + \\ - \\ \times \\ \div \end{array} \right\} \text{ (n) }$	STO+ I E	

Program Instructions

Stop Program		0000 STOP	} 35
End Program		0001 END	
Pause for 1/8 sec.	PAUSE	0002 PAUSE	
Advance Paper	SPACE	0003 SPACE	
NO OPERATION	NOP	0004 NOP	
Absolute Branches	(n)	0005 GOTO 0125	} 47, 69, 70
	LABEL (n)	0007 GOTO L01	
		0009 GOTO A	
		0026 GOTO X	
Computed Branches		0026 GOTO X	} 45, 47, 53, 97
	LABEL	0015 GOTO L01	
Label	(n)	0016 LBL ---- 0	
IF X is +		0014 IF +	} 50
IF X is -		0015 IF -	
IF X is 0		0016 IF 0	
IF X < Y	$X < Y$	0017 IF X<Y	
IF X = Y	$X = Y$	0018 IF X=Y	
IF X ≥ Y	$X \geq Y$	0019 IF X≥Y	
IF Flag n is set	(n)	0020 IF SFG 1	
IF Flag n is clear	(n)	0021 IF OFG 5	

Instruction	Keys	Listing	See Page
Subroutine Branches	 	0022 GOSUB 0075 0024 GOSUB L05 0034 GOSUB X 0035 GOSUB LX	53, 70, 97
Subroutine Return		0028 RETURN	
Set Flag n		0029 SFG 2	61
Clear Flag n		0030 CFG 8	
FOR-NEXT Loops	 	0031 FOR B+G 0032 NEXT B	57

Tape Cartridge

Rewind Tape		0033 REWIND	76
Mark Tape		0034 MARK	80
Record Program		0035 ROPROG	83
Record Data		0036 RODATA	88
Load Program or Data		0037 LOAD	84, 90
Load & Run Program		0039 LD&GO	85
Verify File		0039 VERIFY	90
Identify File		0040 IDENT	82
Record Secured Program		0041 R0SLC	91

Editing Keys (not programmable)

List Program		_____	35, 63, 69
Increment Program Counter		_____	36, 63, 64, 69
Decrement Program Counter		_____	36, 63, 69
Insert Steps		_____	63, 66
Delete Steps		_____	63, 65
Erase Program Memory		_____	34, 67
Set Alpha Mode		_____	63, 67

Subject Index

a

Absolute branching	47
Accessories list	103
Accumulate + (ACC+) function	24
Accuracy (significant digits)	121
Alpha:	
additional characters	71
editing	67
mode	28
printing	28
Assigning data registers	16
Arccosine; Arcsine; Arctangent	19
Arithmetic:	
and the stack	6
chain	8
register	14
AUTOSTART mode	92

b

BACKSPACE/RESET key	4, 63
BCD Interface (HP 98133A)	xiii
Branching:	
absolute	47
computed	47
errors	70
speed	69
subroutine	53
symbolic (labels)	45
"Busy" display	2

c

Calculating range	121
Calculator:	
cleaning	112
fuses	105
grounding	105
initial turn-on	106
inspection	103
installation	103
keyboard	viii (foldout)
power requirements	111
overflow	121
CALL sequences (peripherals)	72
Cartridge, data	77
Change sign key	3
Changing program instructions	64

Clearing:

key sequences	4
program memory	67
stack	4
storage registers	13, 18
X-register	4
Common log (\log^{10})	20
Complex arithmetic	24
Computed branching	47
Conversions:	
D.MS/decimal degrees	23
rectangular/polar	22
Correcting (editing) programs	63

d

Data storage	12
Data (tape) Cartridge	77
Decimal degrees/D.MS conversion ...	23
Degrees (DEGS) units	19
Definable (special) functions	97
Deleting:	
program steps	65
summation data	25
Digitizer (HP 9864A)	xii
Display:	
"busy" display	2
FIX format	10
initial (turn-on)	106
program entry	37
overflow	117
rounding of	10
scientific notation	10

e

Editing:	
alpha	67
programs	63
END instruction	35
ENTER instruction	6
Equipment list	103
Enter exponent (E EX) key	3
Error messages	(inside-back cover)
Error detection (tape)	79
Example programs on tape	118
Exchange X and Y key	6
Expanded memory (option 001)	x
Exponential function	20

f

Factorial, program for calculating	52
File structure of tape	80
Fixed point display	10
FLAG instructions	61
Flowcharting	40
Formats (numbers)	10
FOR-NEXT instructions	57
Functions:	
arithmetic	4
editing	63
program control	35
scientific	18
special (definable)	97
trigonometric	19
Fuse, calculator	105

g

General I/O Interface (HP 98134A)	xiii
GOSUB (and RETURN) instructions ...	53
GOTO instructions	47
Grads units	19
Grounding requirements (calculator) ..	105
Guard digits	121

h

HP-IB Interface (HP 98135A)	xiv
-----------------------------------	-----

i

IDENTIFY instruction	82
IF instructions	50
Improper operations	2
Increasing calculator memory	x
Index, key	123
Indirect storage & recall	15
Initial display	1
INSERT key	63
Inspection, calculator	103
Installation, calculator	103
Instructions, program:	
changing	64
entering	36
recording	83
securing	91
Integer (INT) function	21
Interfaces	xiii
I/O channels (option 002)	x

k

Keyboard drawing	viii
Keyboard magazine	xiv
Key index	123
Keying in exponents of ten	3
Keying in numbers	3

l

LABEL instructions:	
with GOTO	47
with GOSUB	53
with special functions	97
LAST X function	6
Listing programs	63
LOAD instructions:	
data	90
programs	84
Logarithmic functions	20
Looping (in a program)	57

m

Manipulating stack contents	6
Manual summary	ii
MARK (tape) instruction	80
Mean	25
Memory:	
data	16
operational stack	6
program	34

n

Natural logarithms	20
Negative numbers	3
NEXT (and FOR) instructions	57
No operation (NOP) instruction	35
Normal (NORM) mode	27
Number formats	10

o

Overflow	ix
Overlays, key	99

p

Paper (printer):	
loading	117
ordering	117
PAUSE instruction	35
Peripheral control	x, 72
Peripheral tests	118
Pi (π)	21
Plotter (HP 9862A)	xi
Polar/rectangular conversion	23
Positioning numbers in the stack	6
Power requirements, calculator	105
Powers, raising numbers to	20
Print-all (ALL) mode	27
PRINT ALPHA (PRNT α) instruction	28
Printers (HP 9866A and 9871A)	xi
PRINT instruction	27
Program:	
correcting	64
counter (PC)	35
directory	100
erasing	34
recording	83
running	37
writing	39
Protected tape cartridges	77

r

Radians (RADS) units	19
Raising numbers to powers	20
Range, calculating	121
Recalling data	13
Reciprocal (1/x) function	21
RECORD instructions:	
data	88
program	83
secure program	91
Rectangular/polar conversion	23
Register(s):	
arithmetic	14
LAST X	6
storage	12
RETURN instructions:	
with GOSUB	53
with special functions	97
Reverse Polish Notation (RPN)	ix
REWIND instruction	78
ROLL DOWN & ROLL UP keys	6
Root (SQRT) function	21
ROUND function	21
Rounding of display	10
RUN mode	2
Running a program	37
RUN/STOP key	35

s

Scientific functions	18
Scientific notation	11
Service, calculator	111
Sine (SIN) function	19
Single-stepping a program	64
SPACE instruction	35
Square root (SQRT) function	21
Special (defined) functions	97
Special HP programs	82
Stack:	
arithmetic	6
clearing	4
manipulating contents	6
position of numbers	6
printing	6
Standard deviation	25
Statistical functions	25
STEP key	63
STOP instruction	35
Storage registers:	
arithmetic	14
clearing	13, 18
indirect operations	15
numeric-named	12
and stat	25
and summation ($\Sigma +$, $\Sigma -$)	24
Symbolic branching (labels)	45

t

Tangent (TAN) function	19
Tape cartridge	77
Tape-drive operations	76
Tape Punch (HP 9884A)	xii
Tape Readers (HP 9863A and 9883A)	xii
Temperature range, calculator	122
Thermal Printer (HP 9866A)	xi
Trigonometric functions	19
Turn-on procedure, calculator	106










u

Utility & Test Cartridge	111, 118
--------------------------	----------

v

Vector (ACC+, ACC-) functions	24
VERIFY (tape) instructions	90

ERROR MESSAGES

* OVERFLOW	Number or result exceeds calculating range.
* SQRT OF NEG #	
* DIVISION BY ZERO	
* LOG OF # ≤ 0	
* NO I/O DEVICE	Peripheral device or interface not connected.
ILLEGAL ADDRESS	Improper step address or storage register specified.
ILLEGAL ARGUMENT	Improper value for operation (e.g., ASIN, ACOS of no. >1 or <-1 ; over 7 digits converted to D.MS format).
MEMORY OVERFLOW	Program instruction, storage register assignment, or program loaded from tape exceeds available memory.
GOSUB OVERFLOW	More than seven subroutines (including special functions) nested at a time.
KEY NOT DEFINED	Special function just called is not defined.
IMPROPER SYNTAX	Incorrect use of        or 
* CHECKSUM ERROR	Program or data loaded into calculator not identical to that in file; this usually indicates a dirty tape head or a worn tape. See "Cleaning the Tape Head" in Section 4.
* VERIFY FAILED	Program or data in file not identical to that in calculator. See "Error Detection" in Section 4.
WRONG FILE TYPE	Attempting to load an empty, extra, or binary file; recording on an extra file.
END OF TAPE	End of tape reached during MARK operation. Also indicates a broken or defective tape; if the tape does not appear to be broken, (advance it using the drive wheel), replace the cartridge, press  , and continue.
PROTECTED TAPE	The cartridge RECORD slide is positioned to prevent MARK and RECORD operations.
SECURED MEMORY	Attempting to list, edit, or record a secured program.
MISSING FOR STMT	
LABEL NOT FOUND	
FILE NOT FOUND	
FILE TOO SMALL	
CARTRIDGE OUT	
MISSING GOSUB	

*These messages are suppressable; see "Flags" in Section 3.

Scan Copyright ©
The Museum of HP Calculators
www.hpmuseum.org

Original content used with permission.

Thank you for supporting the Museum of HP
Calculators by purchasing this Scan!

Please to not make copies of this scan or
make it available on file sharing services.