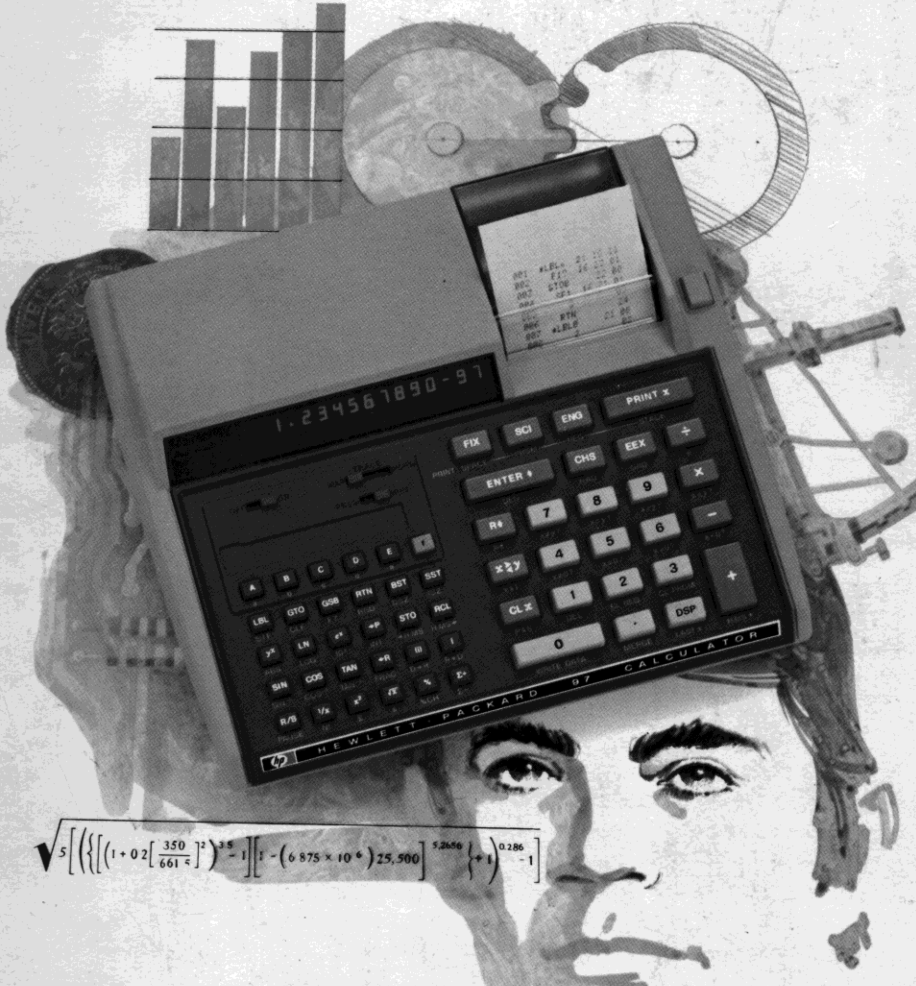
TOOL DESIGN GRP 26

HEWLETT-PACKARD

HP-97

Owner's Handbook and Programming Guide



$$\sqrt{5\left[\left(\left\{\left[\left(1+0.2\left[\frac{350}{661.5}\right]^{2}\right)^{3.5}-1\right]\left[1-\left(6.875\times10^{6}\right)25{,}500\right]^{5.2656}\right\}+1\right)^{0.286}-1\right]}$$

"The success and prosperity of our company will be assured only if we offer our customers superior products that fill real needs and provide lasting value, and that are supported by a wide variety of useful services, both before and after sale."

Statement of Corporate Objectives.
Hewlett-Packard

When Messrs. Hewlett and Packard founded our company in 1939, we offered one superior product, an audio oscillator. Today, we offer more than 3,000 quality products, designed and built for some of the world's most discerning customers.

Since we introduced our first scientific calculator in 1967, we've sold over a million world-wide, both pocket and desktop models. Their owners include Nobel laureates, astronauts, mountain climbers, businessmen, doctors, students, and housewives.

Each of our calculators is precision crafted and designed to solve the problems its owner can expect to encounter throughout a working lifetime.

HP calculators fill real needs. And they provide lasting value.

HEWLETT PACKARD

The HP-97
Programmable Printing Calculator
Owner's Handbook
and
Programming Guide

March 1977

00097-90001, Rev. D 3/77

Printed in U.S.A.

Contents

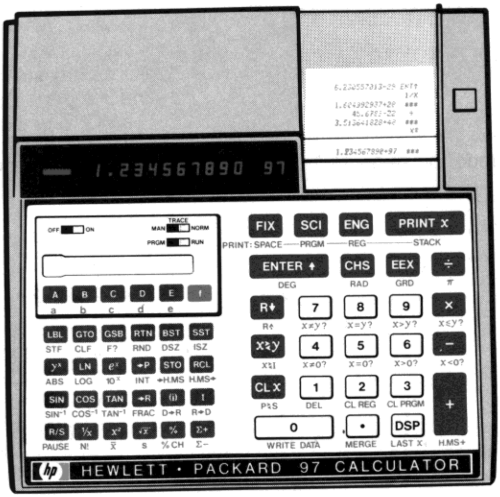
Lunar Module model on page 110 courtesy of NASA, AMES Research Center.

The HP-97 Programmable Printing Calculator

Automatic Memory Stack

Registers

T 0.00
Z 0.00
Y 0.00



Displayed X ......

LAST X

Addressable Storage Registers

Primary Registers

	(i) Address
I	25
R_E	24
R_D	23
R_C	22
R_B	21
R_A	20
R_9	9
R_8	8
R_7	7
R_6	6
R_5	5
R_4	4
R_3	3
R_2	2
R_1	1
R_0	0

Protected Secondary Registers

	(i) Address	
R_{S9}	19	n
R_{S8}	18	Σxy
R_{S7}	17	Σy^2
R_{S6}	16	Σy
R_{S5}	15	Σx^2
R_{S4}	14	Σx
R_{S3}	13	
R_{S2}	12	
R_{S1}	11	
R_{S0}	10	

Program Memory

000	
001	51
002	51
003	51
004	51
005	51
220	51
221	51
222	51
223	51
224	51

Function Key Index

Manual RUN Mode. PRGM-RUN switch PRGM ▮▯ RUN set to RUN.

Function keys pressed from the keyboard execute individual functions as they are pressed. Input numbers and answers are displayed. All function *keys* listed below operate either from the keyboard or as recorded instructions in a program.

☐ Paper advance pushbutton. Press to advance paper without printing **(page 25).**

OFF ▯▮ ON Power switch **(page 23).**

MAN TRACE NORM Print Mode switch. Selects printing option **(page 25).**

Printing Functions

PRINT: SPACE Advances paper one space without printing **(page 132).**

PRINT: REG Prints contents of all primary storage registers **(page 69).**

PRINT: STACK Prints contents of automatic memory stack **(page 47).**

PRINTx Prints contents of displayed X-register **(page 25).**

Digit Entry

ENTER↑ Enters a copy of number displayed in X-register into Y-register. Used to separate numbers **(page 51).**

CHS Changes sign of mantissa or exponent of 10 in displayed X-register **(page 24).**

EEX Enter exponent. After pressing, next numbers keyed in are exponents of 10 **(page 43).**

0 through 9 Digit keys **(page 24).**

• Decimal point **(page 24).**

Display Control

SCI Selects scientific notation display **(page 36).**

FIX Selects fixed point display **(page 37).**

ENG Selects engineering notation display **(page 38).**

DSP Followed by number key, selects number of displayed digits **(page 36).**

Number Manipulation

R↑ Rolls up contents of stack for viewing in displayed X-register **(page 49).**

R↓ Rolls down contents of stack for viewing in displayed X-register **(page 48).**

x⇄y Exchanges contents of X- and Y-registers of stack **(page 49).**

CLx Clears contents of displayed X-register to zero **(page 24).**

Number Alteration

ABS Gives absolute value of number in displayed X-register **(page 78).**

INT Leaves only integer portion of number in displayed X-register by truncating fractional portion **(page 78).**

FRAC Leaves only fractional portion of number in displayed X-register by truncating integer portion **(page 78).**

RND Rounds mantissa of 10-digit number in X-register to actual value seen in the display **(page 77).**

Mathematics

N! Computes factorial of number in displayed X-register **(page 79).**

1/x Computes reciprocal of number in displayed X-register **(page 79).**

x^2 Computes square of number in displayed X-register **(page 80).**

$\sqrt{x}$ Computes square root of number in displayed X-register **(page 80).**

π Places value of pi (3.141592654) into displayed X-register **(page 81).**

\+ − × ÷ Arithmetic operators **(page 27).**

Storage

STO Store. Followed by address key, stores displayed number in specified primary storage register (R_0 through R_9, R_A through R_E, I). Also used to perform storage register arithmetic **(page 64).**

RCL Recall. Followed by address key, recalls number from specified primary storage register (R_0 through R_9, R_A through R_E, I) into the displayed X-register **(page 64).**

CL REG Clears contents of all primary storage registers (R_0 through R_9, R_A through R_E, I) to zero **(page 70).**

LAST X Recalls number displayed before the previous operation back into the displayed X-register **(page 59).**

P⇄S Primary exchange secondary. Exchanges contents of primary storage registers R_0 through R_9 with contents of protected secondary storage registers R_{S0} through R_{S9} **(page 66).**

Trigonometry

→H.MS Converts decimal hours or degrees in displayed X-register to *hours, minutes, seconds* or *degrees, minutes, seconds* **(page 85).**

H.MS→ Converts *hours, minutes, seconds* or *degrees, minutes, seconds* in displayed X-register to decimal hours or degrees **(page 85).**

H.MS+ Adds *hours, minutes, seconds,* or *degrees, minutes, seconds* in Y-register to those in displayed X-register **(page 87).**

SIN^{-1} COS^{-1} TAN^{-1} Computes arc sine, arc cosine, or arc tangent of number in displayed X-register **(page 84).**

SIN COS TAN Computes sine, cosine, or tangent of value in displayed X-register **(page 84).**

D→R Converts degrees in displayed X-register to radians **(page 83).**

R→D Converts radians in displayed X-register to degrees **(page 83).**

DEG Sets decimal degrees mode for trigonometric functions **(page 84).**

RAD Sets radians mode for trigonometric functions **(page 84).**

GRD Sets grads mode for trigonometric functions **(page 84).**

Polar/Rectangular Conversion

→P Converts x, y rectangular coordinates placed in X- and Y-registers to polar magnitude *r* and angle θ **(page 89).**

→R Converts polar magnitude *r* and angle θ in X- and Y-registers to rectangular x and y coordinates **(page 90).**

Logarithmic and Exponential

y^x Raises number in Y-register to power of number in displayed X-register **(page 94).**

10^x Common antilogarithm. Raises 10 to power of number in displayed X-register **(page 93).**

e^x Natural antilogarithm. Raises *e* (2.718281828) to power of number in displayed X-register **(page 93).**

LOG Computes common logarithm (base 10) of number in displayed X-register **(page 93).**

LN Computes natural logarithm (base *e*, 2.718...) of number in displayed X-register **(page 93).**

Statistics

Σ+ Accumulates numbers from X- and Y-registers into secondary storage registers R_{S4} through R_{S9} **(page 97).**

Σ- Subtracts x and y values from storage registers R_{S4} through R_{S9} for correcting or subtracting Σ+ accumulation entries **(page 105).**

$\bar{x}$ Computes mean (average) of x and y values accumulated by Σ+ **(page 100).**

s Computes sample standard deviations of x and y values accumulated by Σ+ **(page 102).**

Percentage

% Computes x% of y **(page 82).**

%CH Computes percent of change from number in Y-register to number in displayed X-register **(page 83).**

Indirect Control

I Recalls number from I-register into displayed X-register. (To store number in I, use STO I.) **(page 65).**

(i) When preceded by DSP, GTO, GSB, STO, RCL, ISZ, or DSZ, the address or control value for that function is specified by the current number in I **(page 201).**

ISZ Increment and skip if zero. Followed by I, adds 1 to contents of I. Followed by (i), adds 1 to contents of storage register specified by value in I. Skips one step if contents are then zero **(page 192).**

DSZ Decrement and skip if zero. Followed by I, subtracts 1 from contents of I. Followed by (i), subtracts 1 from contents of storage register specified by value in I. Skips one step if contents are then zero **(page 192).**

x⇄I Exchanges contents of displayed X-register with those of I-register **(page 192).**

Flags

STF Set flag. Followed by flag designator (0, 1, 2, or 3), sets flag true **(page 231).**

CLF Clear flag. Followed by flag designator (0, 1, 2, or 3), clears flag **(page 231).**

Magnetic Card Control

W/DATA If a magnetic card is passed through the card reader immediately after this operation, the contents of the storage registers are recorded on the card **(page 251).**

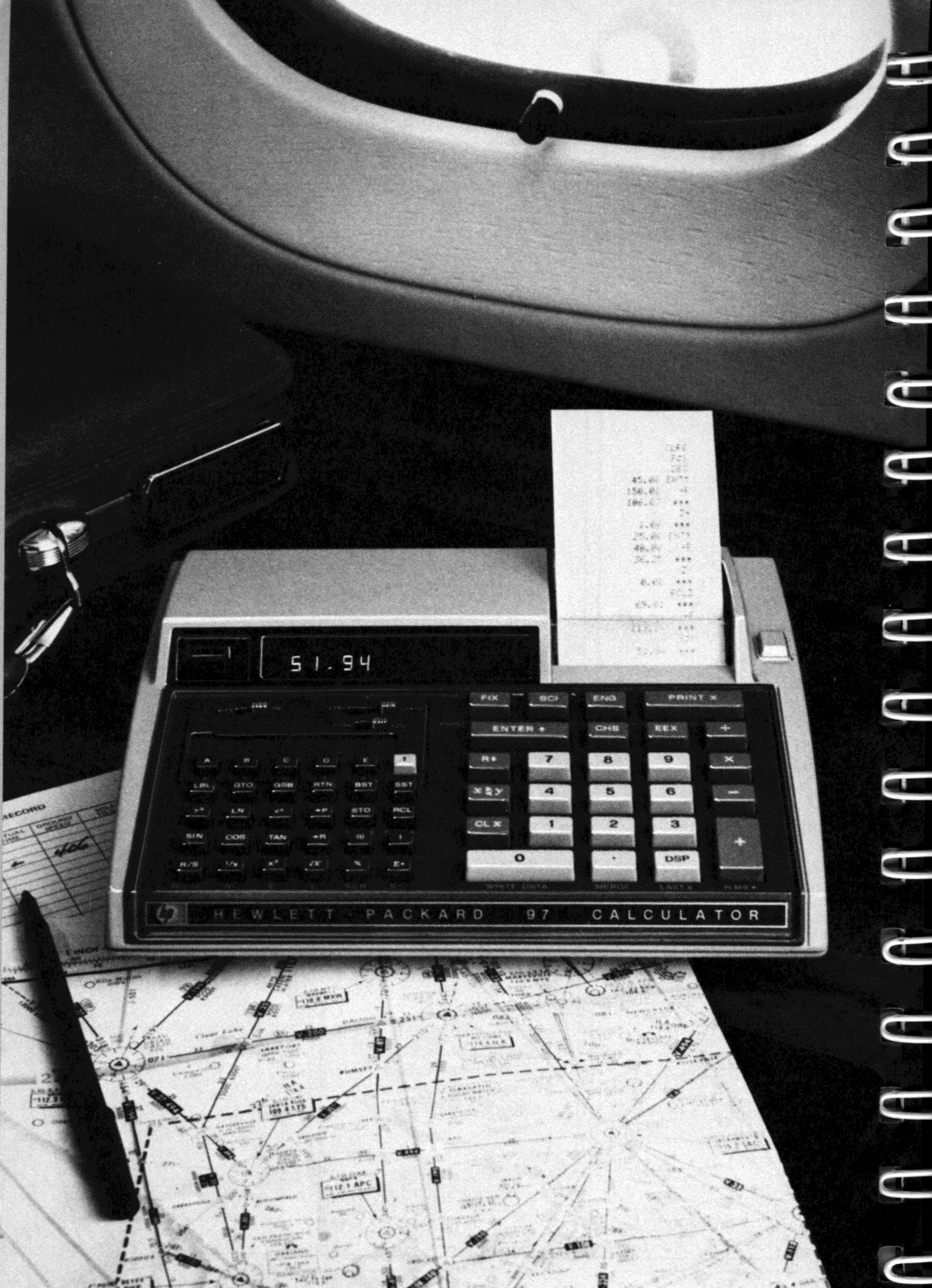
MERGE Merges, rather than overwrites, data or program from magnetic card with data or program in calculator **(page 247).**

Programming Key Index

PROGRAM Mode	Automatic RUN Mode	
PRGM-RUN switch set to PRGM PRGM RUN All function keys except the ones below are loaded into program memory when pressed. Program memory contents recorded upon magnetic card when card is passed through card reader.	PRGM-RUN switch set to RUN PRGM RUN Function keys may be executed as part of a recorded program or individually by pressing from the keyboard. Input numbers and answers are displayed by the calculator, except where indicated. Data or instructions loaded from magnetic card into calculator when card is passed through card reader.	
Active keys:	**Pressed from the keyboard:**	**Executed as a recorded program instruction:**
In PROGRAM mode only six operations are active. These operations are used to help record programs, and cannot themselves be recorded in program memory.	A B C D E a b c d e User-definable keys. Cause calculator to search downward through program memory to first designated label and begin execution there **(page 121).**	A B C D E a b c d e 0 1 2 3 4 5 6 7 8 9 Label designators. When preceded by LBL, define beginning of routine. When preceded by GTO or GSB, cause calculator to stop execution, search downward through program memory to first designated label, and resume execution there **(page 119).**
GTO Go to. Followed by • n n n, positions calculator to step n n n of program memory. No instructions are executed **(page 135).**	GTO Go to. Followed by • n n n, sets calculator to step n n n of program memory without executing instructions. Followed by label designator (A through E, f a through f e, 0 through 9) or (i), causes calculator to search downward through program memory to first designated label and stop there **(page 135).**	GTO Go to. Followed by label designator (A through E, f a through f e, 0 through 9) or (i), causes calculator to stop execution, search through program memory to first designated label, and resume execution there **(page 153).**
	GSB Go to subroutine. Followed by label designator,(A through E, f a through f e, 0 through 9) or (i), causes calculator to start executing instructions, beginning with designated label **(page 177).**	GSB Go to subroutine. Followed by label designator (A through E, f a through f e, 0 through 9) or (i), causes calculator to search through program memory to first designated label and execute that section of program memory as a subroutine **(page 177).**

PROGRAM Mode	**Automatic RUN Mode**	
Active keys:	**Pressed from the keyboard:**	**Executed as a recorded program instruction:**
	RTN Return. Sets calculator to step 000 of program memory **(page 138).**	RTN Return. If executed as a result of pressing a label designator or executing a GTO instruction, stops execution and returns control to keyboard. If executed as a result of a GSB instruction, returns control to next step after the GSB instruction **(page 119).**
PRINT: PRGM Print program. Prints contents of program memory, beginning with current step and continuing until two consecutive R/S instructions are encountered or step 224 is printed **(page 131).**	PRINT: PRGM Print program. Prints contents of program memory, beginning with current step and continuing until two consecutive R/S instructions are encountered or step 224 is printed **(page 131).**	
BST Back step. Moves calculator back one step in program memory **(page 135).**	BST Back step. Sets calculator to and displays step number and keycode of previous program memory step number when pressed; displays original contents of X-register when released. No instructions are executed **(page 135).**	
SST Single step. Moves calculator forward one step of program memory **(page 135).**	SST Single step. Displays step number and keycode of current program memory step when pressed; executes instruction, displays result, and moves calculator to next step when released **(page 135).**	
DEL Delete. Deletes current instruction from program memory. All subsequent instructions moved up one step **(page 136).**	DEL After f prefix key, cancels that key. After other keys, does nothing. Does not disturb program memory or calculator status **(page 136).**	

PROGRAM Mode	**Automatic RUN Mode**	
Active keys:	**Pressed from the keyboard:**	**Executed as a recorded program instruction:**
CLPRGM Clear program. Clears program memory to all R/S instructions, sets calculator to step 000, clears all flags, and specifies FIX 2 display and DEG modes **(page 135).**	CLPRGM After f prefix key, cancels that key. After other keys, does nothing. Does not disturb program memory or calculator status **(page 135).**	
		PAUSE Stops program execution and transfers control to keyboard for 1 second, then resumes program execution **(page 171).**
		x≠y? x=y? x>y? x≤y? x≠0? x=0? x>0? x<0? Conditionals. Each tests value in X-register against value in Y-register or 0 as indicated. If true, calculator executes instruction in next step of program memory. If false, calculator skips one step before resuming execution **(page 159).**
		F? Is flag true? Followed by flag designator (0, 1, 2, or 3), tests designated flag. If flag is set (true) calculator executes the instruction in the next step of program memory. If flag is cleared (false), calculator skips one step before resuming execution. F? clears flags F2 and F3 after tests **(page 231).**
	R/S Run/stop. Begins execution from current step of program memory. Stops execution if program is running **(page 169).**	R/S Run/stop. Stops program execution **(page 119).**
	Any key. Pressing any key on the keyboard stops execution of a running program.	


51.94
HEWLETT PACKARD 97 CALCULATOR

Meet the HP-97

Congratulations!

With your purchase of the HP-97 Programmable Printing Calculator, you have acquired a truly versatile and unique calculating instrument. Using the Hewlett-Packard RPN logic system that slices with ease through the most difficult equations, the HP-97 is without parallel:

As a scientific calculator. As a scientific calculator, the HP-97 features a familiar adding-machine style keyboard for rapid data entry wedded to a powerful calculator with dozens of mathematical, statistical, and scientific functions, and a three-way printer for archival permanence to your answers.

As a problem-solving machine. Anyone who can follow simple step-by-step instructions can use the prerecorded magnetic cards in the Standard Pac and the optional application pacs to solve common problems from the areas of engineering, mathematics, finance, statistics, medicine, and many other fields. *Immediately*.

As a personal programmable calculator. The HP-97 is so easy to program and use that it requires no prior programming experience or knowledge of arcane programming languages. Yet even the most sophisticated computer experts marvel at the programming features of the HP-97:

- Magnetic cards that record data or programs—permanently.
- 26 data storage registers.
- 224 steps of program memory.
- Fully merged prefix and function keys that mean more programming per step.
- Easy-to-use editing features for correcting and modifying programs.
- Powerful unconditional and conditional branching.
- Three levels of subroutines, four flags, 20 easily-accessed labels.
- Indirect addressing.
- Printer to record results, list programs, or trace executing programs.

And in addition, the HP-97 can be operated from its rechargeable battery pack for *complete portability,* anywhere.

Now let's take a closer look at the HP-97 to see how easy it is to use, whether we solve a problem manually, use one of the sophisticated prerecorded programs from the Standard Pac, or even write our own program.

Manual Problem Solving

To get the feel of your HP-97, try a few simple calculations. First, set the switches that are located in the upper left-hand corner of the keyboard as follows:

Set the OFF-ON switch OFF ON to ON.
Set the PRGM-RUN switch PRGM RUN to RUN.
Set the MAN-TRACE-NORM switch MAN TRACE NORM to MAN.

To solve:	**Press:**	**Display:**
5 + 6 = 11	5 ENTER↑ 6 +	11.00
8 ÷ 2 = 4	8 ENTER↑ 2 ÷	4.00
7 − 4 = 3	7 ENTER↑ 4 −	3.00
9 × 8 = 72	9 ENTER↑ 8 ×	72.00
$\frac{1}{5} = 0.20$	5 1/x	0.20
Sine of 30° = 0.50	3 0 SIN	0.50

Now let's try something a little more involved. To calculate the surface area of a sphere, the formula $A = \pi d^2$ can be used, where:

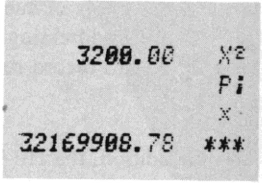
> A is the surface area of the sphere,
> d is the diameter of the sphere,
> π is the value of pi, 3.141592654.

Ganymede, one of Jupiter's 12 moons, has a diameter of 3200 miles. You can use the HP-97 to manually compute the area of Ganymede. Merely press the following keys in order.

First, ensure that a paper roll has been properly installed in the calculator, and slide the MAN-TRACE-NORM switch MAN TRACE NORM to NORM.

Then:

Press	**Display**	
3 2 0 0	3200.	Diameter of Ganymede.
x²	10240000.00	Square of the diameter.
f π	3.14	The quantity π.
×	32169908.78	Area of Ganymede in square miles.
PRINT x	32169908.78	The answer printed.



```
   3200.00   X²
             Pi
             x
32169908.78  ***
```

You can see that the paper tape has preserved a record of your calculation. Save this tape—you are going to use it to write a program for your HP-97. But first let's look at a prerecorded program, one of the 15 that are shipped with your calculator.

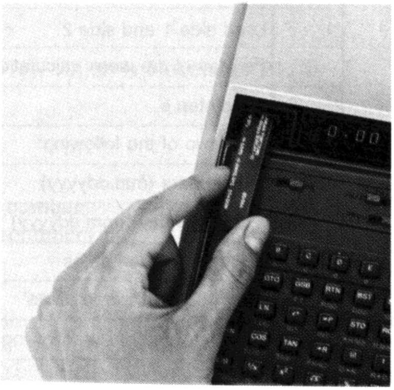
Running a Prerecorded Program

The Standard Pac shipped with your calculator contains 15 prerecorded magnetic cards, and each card contains a program. By using cards from the Standard Pac (or from any of the optional application pacs, available in areas like finance, statistics, mathematics, engineering, or medicine) you can use your HP-97 to perform extremely complex calculations just by following the cookbook-style directions in each pac. Let's try running one of these programs now.

1. Select the Calendar Functions program from the Standard Pac card case.

2. Ensure that the PRGM-RUN switch PRGM ■▥ RUN is set to RUN.

3. Set the Print Mode switch MAN ▥■ NORM (TRACE) to MAN. Insert side 1 of the Calendar Functions card, printed side up, into the card reader slot on the front of the calculator as shown. When the card is partially into the slot, a motor engages and passes the card through the calculator and out a similar slot at the rear of the calculator. Let the card move freely.



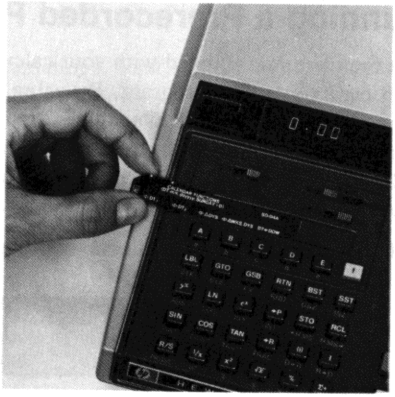
4. The calculator display should read `Crd` to prompt you that side 2 of the card must be read in.

5. Now insert side 2 of the Calendar Functions card, again face up, into the card reader slot on the front of the calculator and permit it to pass through the card reader to the rear of the calculator.

6. If after either pass of the card through the card reader, the display shows `Error`, that side of the card did not read properly. Press CLX, then insert that side of the card into the card reader slot and let it pass through again.

7. When both sides of the card have been read properly, the display will again show the previous answer.

8. Insert the card into the window slot, as shown. The markings on the card should be directly over the keys marked A B C D E. The markings, or mnemonics, on the card now identify the function of each of these five keys.



You are now ready to use the program.

Example: How many days are there between September 3, 1944 and November 21, 1975?

Solution: The figure below duplicates the user instructions for the Calendar Functions program. These instructions can also be found in the *HP-97 Standard Pac,* just as can the instructions for the other 14 programs in the Pac.

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYS	OUTPUT DATA/UNITS
1	Load side 1 and side 2.			
2	For day of the week calculations			
	go to step 6.			
3	Input two of the following:			
	First date (mm.ddyyyy)	DT_1	A	Day $\#_1$
	Second date (mm.ddyyyy)	DT_2	B	Day $\#_2$
	Days between dates	DAYS	C	Days
	or weeks between dates*	WKS. DYS	D	Days
4	Calculate one of the following:			
	First date		A	DT_1
	Second date		B	DT_2
	Days between dates		C	Days
	Weeks between dates		D	WKS. DYS
5	For a new case go to step 2.			
6	Input date and calculate day			
	of the week (0 = Sunday,			
	6 = Saturday).	DT	E	DOW
7	For a new case go to step 2.			
	*Either days between dates or			
	weeks between dates, but not			
	both, may be input in step 3.			

To solve the problem, just follow the User Instructions, beginning with step 1. Since you have already performed step 1, and you do not wish to perform step 2, you continue on to step 3. There you input the first date in the format *mm.ddyyyy*. (This means you key in the date as the month, from 00 to 12, then a decimal point, then the day as *dd,* and finally the year as *yyyy*.) Thus, to key in September 3, 1944:

Press	**Display**
09.031944	09.031944

Reading across the line, you can see that after you input the first date (DT_1), you are directed under the KEYS heading to press A .

Press	**Display**	
A	2431337.	Julian day number (number of days since the inception of the Julian calendar).

Now follow the instructions for the second date (DT_2), which is November 21, 1975.

Press	**Display**	
11.211975	11.211975	
B	2442738.	(Julian day number used by astronomers.)

Now you move to step 4, which gives the key you press for calculation. You can see that to calculate the days between dates, you press C .

Press	**Display**
C	11401.

The number of days between September 3, 1944 and November 21, 1975 is 11401.

You can run the program again as often as you like. With the calendar program, you can calculate the days between dates, the weeks between dates, or even the day of the week on which any date falls.

You have seen from this example how simple it is to use your HP-97. You can begin using your Standard Pac, or any of the optional applications Pacs, right *now*. All you have to do to begin taking advantage of the calculating power and programmability of the HP-97 is follow simple instructions like these.

Your Own Program

Earlier, you calculated the surface area of Ganymede, one of Jupiter's 12 moons, and you should have saved the paper tape with the keystroke list from that problem. Now, if you wanted the surface area of *each* moon, you could repeat that procedure 12 times, using a different value for the diameter *d* each time. An easier and faster method, however, is to create a *program* that will calculate the surface area of a sphere from its diameter, instead of pressing all the keys for each moon.

To calculate the area of a sphere using a program, you should first *create* the program, then you must *load* the program into the calculator, and finally you *run* the program to calculate each answer. If you want to save the program, you can *record* it permanently on a magnetic card.

Creating the Program. You have already created it! A program is nothing more than the series of keystrokes you would execute to solve the same problem manually. Two additional operations, a *label* and a *return* are used to define the beginning and end of the program.

Loading the Program. To load the keystrokes of the program into the calculator:

1. Slide the PRGM-RUN switch PRGM ▮ RUN to PRGM *(program)*.
2. Press f CL PRGM to clear the calculator.
3. Press the following keys in order. (When you are loading a program, the display gives you information that you will find useful later, but which you can ignore for now.)

LBL A — Defines the beginning of the program.

x²
f π
×
PRINT x — These are the same keys you pressed to solve the problem manually.

RTN — Defines the end of the program.

The calculator will now remember this keystroke sequence.

Running the Program. To run the program to find the area of any sphere from its diameter:

1. Slide the PRGM-RUN switch PRGM ▮ RUN back to RUN.
2. Key in the value of the diameter.
3. Press A to run the program.

When you press A, the sequence of keystrokes you loaded is automatically executed by the calculator, giving you the same answer you would have obtained manually.

For example, to calculate the area of Ganymede, with a diameter of 3200 miles:

Press	Display	
3200	3200.	
A	32169908.78	Square miles.

With the program you have loaded, you can now calculate the area of any of Jupiter's moons—in fact, of *any* sphere—using its diameter. You have only to leave the calculator in RUN mode and key in the diameter of each sphere for which you want the area, then press A . For example, to compute the surface area of Jupiter's moon Io, with a diameter of 2310 miles:

Press	**Display**	
2310 A	16763852.56	Square miles.

For the moons Europa, diameter 1950 miles, and Callisto, diameter 3220 miles:

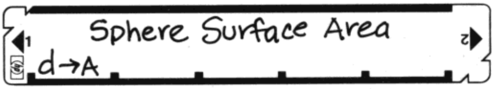
Press	**Display**	
1950 A	11945906.07	Area of Europa in square miles.
3220 A	32573289.27	Area of Callisto in square miles.

Programming the HP-97 is *that* easy! The calculator remembers a series of keystrokes and then executes them at the press of a single key. In fact the HP-97 can remember up to 224 separate operations (and many more keystrokes, since many operations require two or three keystrokes) and execute them at the press of one of the label keys. By using, say, label A for one program, label B for another, etc., your calculator can contain many different programs at one time.

Recording the Program. Just as the programs in the Standard Pac have been permanently recorded on magnetic cards, so you can record your program on a magnetic card. To record your program:

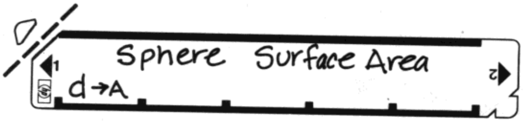
1. Select a blank, unprotected (unclipped) magnetic card.

2. Slide the PRGM-RUN switch PRGM ▯▮ RUN to PRGM.
3. Insert side 1 of the card into the front card reader slot on the calculator. Permit the card to pass through the card reader to the rear of the calculator. Since your program contains fewer than 113 instructions, you need to pass only one side of the card through the card reader. Your program is now recorded on the magnetic card.
4. Be sure to mark the card so you don't forget what program is on the card and what keys control the program. The marked card might look like this when you are through:



5. The program now on the card will remain there until you record another program over it. To save the program permanently, so that no other program can be recorded on the card, clip the corner of the card nearest side 1:

Side 1 protected.



Side 2 unclipped, unprotected.

That's all there is to it! You can reuse the program as often as you like—merely pass the card through the card reader with the PRGM-RUN switch set to RUN each time you want to load this program into the calculator.

Using this Handbook

New to Hewlett-Packard calculators? Part One of this handbook has been designed to teach you to use your HP-97 as a powerful scientific calculator. By working through these sections of the handbook, you'll learn every function that you can use to calculate answers manually, and you'll come to appreciate the calculating efficiency of the Hewlett-Packard logic system with RPN. And since the programmability of the HP-97 stems from its ability to remember a series of manual keystrokes, Part One, Using Your HP-97 Calculator, is invaluable in laying the groundwork for Part Two, Programming The HP-97.

Previous HP user? If you've already used Hewlett-Packard pocket or desktop calculators with RPN, you may want to familiarize yourself with the unique printer options of the HP-97 by reading page 25, and then turn directly to Part Two, Programming The HP-97. Later, though, you will undoubtedly wish to peruse Part One at your leisure in order to discover the many calculating advantages of the HP-97.

Whether an old hand or a novice, you'll find the Function and Programming Key Index on pages 8–11 invaluable as a quick reference guide, a programming guide, or even to illustrate the features of the HP-97 to your friends.

Part One
Using Your HP-97 Calculator

3.00
HEWLETT · PACKARD 97 CALCULATOR

Section 1

Getting Started

Your HP-97 is shipped fully assembled, including a battery. You can begin using your calculator immediately by connecting the cord from the ac adapter/recharger and plugging the charger into an ac outlet. If you want to use your HP-97 on battery power alone, you should charge the battery for 6 hours first. Whether you operate from battery power or from power supplied by the charger, *the battery pack must always be in the calculator*.

To begin:

Slide the PRGM-RUN switch PRGM ▮▯ RUN to RUN.

Slide the Print Mode switch MAN ▯ TRACE ▮ NORM to MAN.

Slide the OFF-ON switch OFF ▮▯ ON to ON.

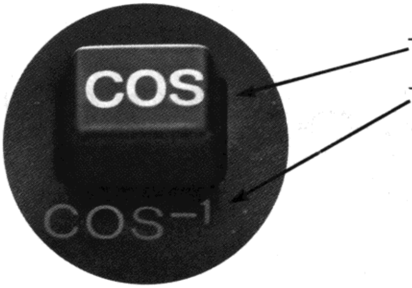
Display

Numbers that you key into the calculator and intermediate and final answers are always seen in the bright red display. When you first turn the calculator ON, the display is set to 0.00 to show you that all zeros are present there.

Keyboard

All keys on the keyboard perform two functions. One function is indicated by the symbol on the face of the key, while another function is indicated by the gold symbol written below the key.

- To select the function printed on the face of the key, press the key.
- To select the function printed in gold below the key, press the gold prefix key f, then press the function key.



To execute this function, press cos .

To execute this function, first press f, then press cos ($\cos^{-1}$).

In this handbook, the selected key function will appear in the appropriate color, outlined by a box, like this: cos , $\cos^{-1}$.

Keying In Numbers

Key in numbers by pressing the number keys in sequence, just as though you were writing on a piece of paper. The decimal point must be keyed in if it is part of the number (unless it is to be right of the last digit).

For example:

Key in 148.84 by pressing the keys	**Display**
1 4 8 . 8 4	148.84

The resultant number 148.84 is seen in the display.

Negative Numbers

To key in a negative number, press the keys for the number, then press CHS *(change sign)*. The number, preceded by a minus (–) sign, will appear in the display. For example, to change the sign of the number now in the display:

Press	**Display**
CHS	–148.84

You can change the sign of either a negative or a positive nonzero number in the display. For example, to change the sign of the –148.84 now in the display back to positive:

Press	**Display**
CHS	148.84

Notice that only negative numbers are given a sign in the display.

Clearing

You can clear any numbers that are in the display by pressing CL x *(clear X)*. This key erases the number in the display and replaces it with 0.00 .

Press	**Display**
CL x	0.00

If you make a mistake while keying in a number, clear the entire number string by pressing CL x. Then key in the correct number.

Printer

The printer has three modes of operation, which you control using the Print Mode switch MAN TRACE NORM :

- With the Print Mode switch MAN TRACE NORM set to MAN *(manual),* the printer is idle and does not print unless you press the **PRINT x** key or one of the other PRINT functions. This mode gives greatest economy of paper and battery power.
- With the Print Mode switch MAN TRACE NORM set to NORM *(normal),* the calculator records a history of the calculation sequence so that you can reconstruct your problem. In this mode you see digit entries and functions, but intermediate and final answers are not printed unless you press the **PRINT x** key.
- With the Print Mode switch MAN TRACE NORM set to TRACE, the calculator prints numbers, functions, and intermediate and final answers, just as they are seen in the display. The results of functions are printed with the symbol *** to the right of the number.

To advance the printer paper, press the paper advance pushbutton that is to the right of the paper output. Don't worry if the display blanks out while the paper advance is operating—this is normal. To advance the paper more than one space, simply hold the pushbutton down until the paper has advanced the desired amount. To replace the paper roll, refer to Your HP-97 Printer in appendix B of this handbook.

Note: If your calculator is out of paper, any operation which would normally cause the calculator to print onto the paper tape instead causes the word ***Error*** to appear in the display. This prompts you to install a fresh roll of paper.

No matter what print mode you choose, you seldom have to worry about "overrunning" the printer when you are calculating. Your HP-97 contains a key buffer that "remembers" up to seven keystrokes—no matter how fast you press the keys.

Functions

The best way to see how simple functions operate on your HP-97 is with the Print Mode switch set to TRACE to give you a complete record of inputs, functions, and answers.

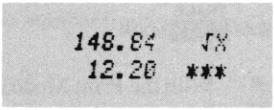
Slide the Print Mode switch MAN TRACE NORM to TRACE now.

In spite of the dozens of functions available on the HP-97 keyboard, you will find the calculator functions simple to operate by using a single, all-encompassing rule: *When you press a function key, the calculator immediately executes the function written on the key.*

Pressing a function key causes the calculator to immediately perform that function, and display the result.

For example, to calculate the square root of 148.84 merely:

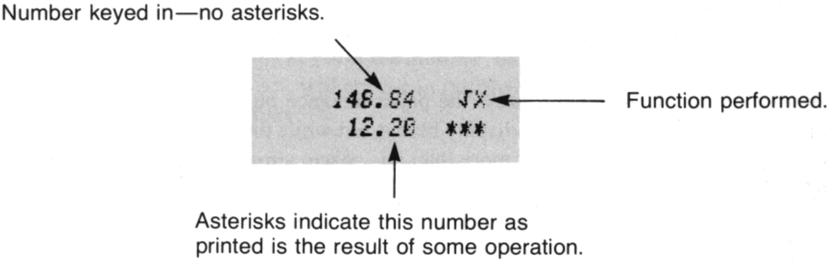
Press	**Display**
148.84	148.84
√x	12.20



```
148.84   √X
 12.20  ***
```

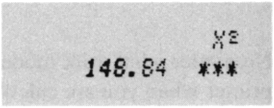
Let's look briefly at the printed copy of that problem to see the simple way that the HP-97 printer duplicates your calculations.

The paper tapes are printed just as you read, from left to right and top to bottom. The number, 148.84, is printed exactly as you keyed it in. A symbol for the function performed, √x, is printed next to it. The answer, 12.20, is printed with a three-asterisk label to its right, indicating that the HP-97 performed some operation in order to obtain the number as it is printed.



Now let's continue. To square the result of the previous calculation:

Press	**Display**
x^2	148.84



```
             X²
148.84  ***
```

√x and x^2 are examples of one-number function keys; that is, keys that execute upon a single number. All function keys in the HP-97 operate upon either one number or two numbers at a time (except for statistics keys like Σ+ and S—more about these later).

> Function keys operate upon either one number or two numbers.

One-Number Functions

To use any one-number function key:

1. Key in the number.
2. Press the function key (or press the prefix key, then the function key).

For example, to use the one-number function 1/x key, you first key in the number represented by *x*, then press the function key. To calculate ¼, key in 4 (the *x*-number) and press 1/x.

Press	Display
4	4.
1/x	0.25

```
4.00  1/X
0.25  ***
```

Now try these other one-number function problems. Remember, *first key in the number, then press the function:*

$\frac{1}{25}$	=	0.04	
$\sqrt{2500}$	=	50.00	
10^5	=	100000.00	(Use the 10ˣ key.)
$\sqrt{3204100}$	=	1790.00	
log 12.58925411	=	1.10	
71^2	=	5041.00	

Two-Number Functions

Two-number functions are functions that must have two numbers present in order for the operation to be performed. +, −, ×, and ÷ are examples of two-number function keys. You cannot add, subtract, multiply, or divide unless there are two numbers present in the calculator. Two-number functions work the same way as one-number functions—that is, the operation occurs when the function key is pressed. Therefore, *both numbers must be in the calculator before the function key is pressed.*

When more than one number must be keyed into the calculator before performing an operation, the ENTER↑ key is used to separate the two numbers.

> Use the ENTER↑ key whenever more than one number must be keyed into the calculator before pressing a function.

If you key in only one number, you never need to press ENTER↑. To place two numbers into the calculator and perform an operation:

1. Key in the first number.
2. Press ENTER↑ to separate the first number from the second.
3. Key in the second number.
4. Press the function key to perform the operation.

For example, to add 12 and 3:

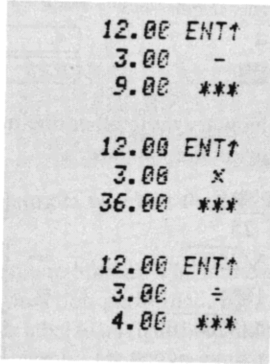
Press	
12	The first number.
ENTER↑	Separates the first number from the second.
3	The second number.
+	The function.
PRINT X	The answer.

```
12.00 ENT↑
 3.00    +
15.00  ***
```

The answer, 15.00, is displayed and printed.

Other arithmetic functions are performed the same way:

To perform	Press	Display
12 − 3	12 **ENTER↑** 3 **−**	9.00
12 × 3	12 **ENTER↑** 3 **×**	36.00
12 ÷ 3	12 **ENTER↑** 3 **÷**	4.00



```
12.00 ENT↑
 3.00    -
 9.00  ***

12.00 ENT↑
 3.00    ×
36.00  ***

12.00 ENT↑
 3.00    ÷
 4.00  ***
```

The **y^x** key is also a two-number operation. It is used to raise numbers to powers, and you can use it in the same simple way that you use every other two-number function key:

1. Key in the first number.
2. Press **ENTER↑** to separate the first number from the second.
3. Key in the second number (power).
4. Perform the operation (press **y^x**).

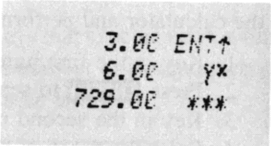
When working with any function key (including **y^x**), you should remember that the displayed number is always designated by x on the function key symbols.

The number displayed is always x.

So **$\sqrt{x}$** means square root of the displayed number, **$1/x$** means $\frac{1}{\text{displayed number}}$, etc.

Thus, to calculate 3^6:

Press	Display	
3	3.	
ENTER↑	3.00	
6	6.	x, the displayed number, is now six.
y^x	729.00	The answer.



```
 3.00 ENT↑
 6.00   y×
729.00 ***
```

Now try the following problems using the **y^x** key, keeping in mind the simple rules for two-number functions:

16^4 (16 to the 4th power) = 65536.00

81^2 (81 squared) = 6561.00 (You could also have done this as a one-number function using **x^2**.)

$225^{.5}$ (Square root of 225) = 15.00 (You could also have done this as a one-number function using **$\sqrt{x}$**.)

2^{16} (2 to the 16th power) = 65536.00

$16^{.25}$ (4th root of 16) = 2.00

Chain Calculations

The speed and simplicity of operation of the Hewlett-Packard logic system become most apparent during chain calculations. Even during the longest of calculations, you still perform only one operation at a time, and you see the results as you calculate—the Hewlett-Packard automatic memory stack stores up to four intermediate results inside the calculator until you need them, then inserts them into the calculation. This system makes the process of working through a problem as natural as it would be if you were working it out with pencil and paper, but the calculator takes care of the hard part.

For example, solve (12 + 3) × 7.

If you were working the problem with a pencil and paper, you would first calculate the intermediate result of (12 + 3).....

$$\cancel{(12+3)} \times 7 =$$
$$15$$

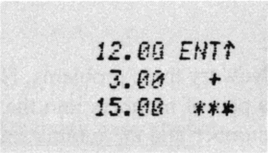
.....and then you would multiply the intermediate result by 7.

$$\cancel{(12+3)} \times 7 =$$
$$15 \times 7 = 105$$

You work through the problem exactly the same way with the HP-97, one operation at a time. You solve for the intermediate result first.....

(12 + 3)

Press	Display	
12	12.	
ENTER↑	12.00	
3	3.	
+	15.00	Intermediate result.

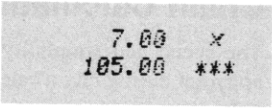


```
12.00 ENT↑
 3.00    +
15.00  ***
```

... and then solve for the final answer. You don't need to press ENTER↑ to store the intermediate result—the HP-97 automatically stores it inside the calculator when you key in the next number. To continue...

Press	Display	
7	7.	The intermediate result from the preceding operation is automatically stored inside the calculator when you key in this number.

×	105.00	Pressing the function key multiplies the new number and the intermediate result, giving you the final answer.



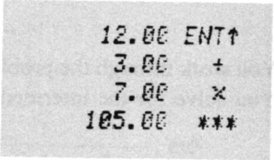
```
  7.00   ×
105.00 ***
```

Because the HP-97 stores intermediate results automatically, you don't need to print them. You can slide the Print Mode switch to NORM to preserve a record of your calculations, and then press PRINTx to print the final answer.

For example, when you solved the above problem in TRACE mode, you preserved *all* intermediate and final results. To solve the same problem and preserve only a history of the calculations:

Slide the Print Mode switch MAN TRACE NORM to NORM.

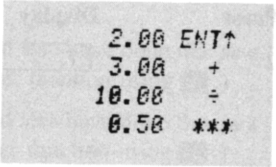
Press	**Display**	
12	12.	
ENTER↑	12.00	
3	3.	
+	15.00	
7	7.	
×	105.00	
PRINTx	105.00	Preserves the final answer in your printed record.



```
 12.00 ENT↑
  3.00    +
  7.00    ×
105.00  ***
```

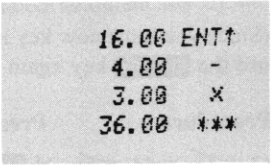
Now try these problems. Notice that for each problem you only have to press ENTER↑ to insert a pair of numbers into the calculator—each subsequent operation is performed using a new number and an automatically stored intermediate result.

To solve	**Press**	**Display**
$\frac{(2+3)}{10}$	2	
	ENTER↑	
	3	
	+	
	10	
	÷	0.50
	PRINTx	0.50



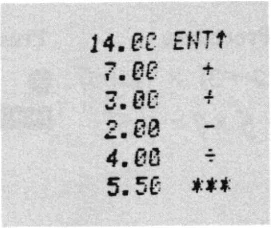
```
 2.00 ENT↑
 3.00    +
10.00    ÷
 0.50  ***
```

3 (16 − 4)	16	
	ENTER↑	
	4	
	−	
	3	
	×	36.00
	PRINTx	36.00



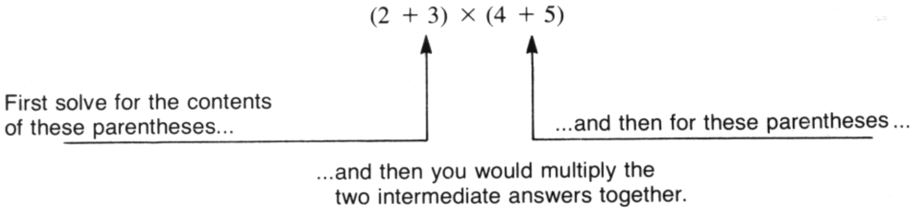
```
 16.00 ENT↑
  4.00   -
  3.00   ×
 36.00  ***
```

$\frac{14 + 7 + 3 - 2}{4}$	14	
	ENTER↑	
	7	
	+	
	3	
	+	
	2	
	−	
	4	
	÷	5.50
	PRINTx	5.50



```
 14.00 ENT↑
  7.00   +
  3.00   +
  2.00   -
  4.00   ÷
  5.50  ***
```

Problems that are even more complicated can be solved in the same simple manner, using the automatic storage of intermediate results. For example, to solve (2 + 3) × (4 + 5) with a pencil and paper, you would:



(2 + 3) × (4 + 5)

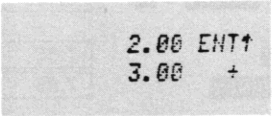
First solve for the contents of these parentheses...

...and then for these parentheses...

...and then you would multiply the two intermediate answers together.

You work through the problem the same way with the HP-97. First you solve for the intermediate result of (2 + 3)....

Press	**Display**	
2	2.	
ENTER↑	2.00	
3	3.	
+	5.00	Intermediate result.

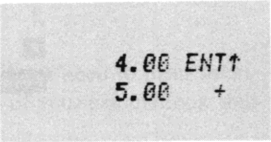


```
  2.00 ENT↑
  3.00   +
```

Then add 4 and 5:

(Since you must now key in another *pair* of numbers before you can perform a function, you use the **ENTER↑** key again to separate the first number of the pair from the second.)

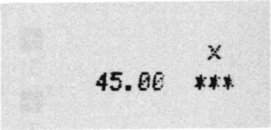
Procedure	Press	Display
(~~2 + 3~~) × (~~4 + 5~~) 5 9	4 **ENTER↑** 5 **+**	9.00



```
4.00 ENT↑
5.00    +
```

Then multiply the intermediate answers together for the final answer:

Procedure	Press	Display
(~~2 + 3~~) × (~~4 + 5~~)	**×**	45.00
5×9=45	**PRINTx**	45.00



```
           ×
45.00    ***
```

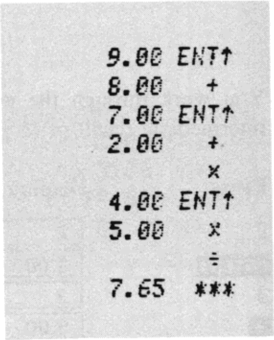
Notice that you didn't need to write down or key in the intermediate answers from inside the parentheses before you multiplied—the HP-97 automatically stacked up the intermediate results inside the calculator for you and brought them out on a last-in, first-out basis when it was time to multiply.

No matter how complicated a problem may look, it can always be reduced to a series of one- and two-number operations. Just work through the problem in the same logical order you would use if you were working it with a pencil and paper.

For example, to solve:

$$\frac{(9 + 8) \times (7 + 2)}{(4 \times 5)}$$

Press	Display	
9 **ENTER↑** 8 **+**	17.00	Intermediate result of (9 + 8).
7 **ENTER↑** 2 **+**	9.00	Intermediate result of (7 + 2).
×	153.00	(9 + 8) multiplied by (7 + 2).
4 **ENTER↑** 5 **×**	20.00	Intermediate result of (4 × 5).
÷	7.65	The final answer.
PRINTx	7.65	



```
9.00 ENT↑
8.00    +
7.00 ENT↑
2.00    +
        ×
4.00 ENT↑
5.00    ×
        ÷
7.65  ***
```

Now try these problems. Remember to work through them as you would with a pencil and paper, but don't worry about intermediate answers—they're handled automatically by the calculator.

$$(2 \times 3) + (4 \times 5) = \boxed{26.00}$$

$$\frac{(14 + 12) \times (18 - 12)}{(9 - 7)} = \boxed{78.00}$$

$$\frac{\sqrt{16.38 \times 5}}{.05} = \boxed{181.00}$$

$$4 \times (17 - 12) \div (10 - 5) = \boxed{4.00}$$

$$\sqrt{(2 + 3) \times (4 + 5)} + \sqrt{(6 + 7) \times (8 + 9)} = \boxed{21.57}$$

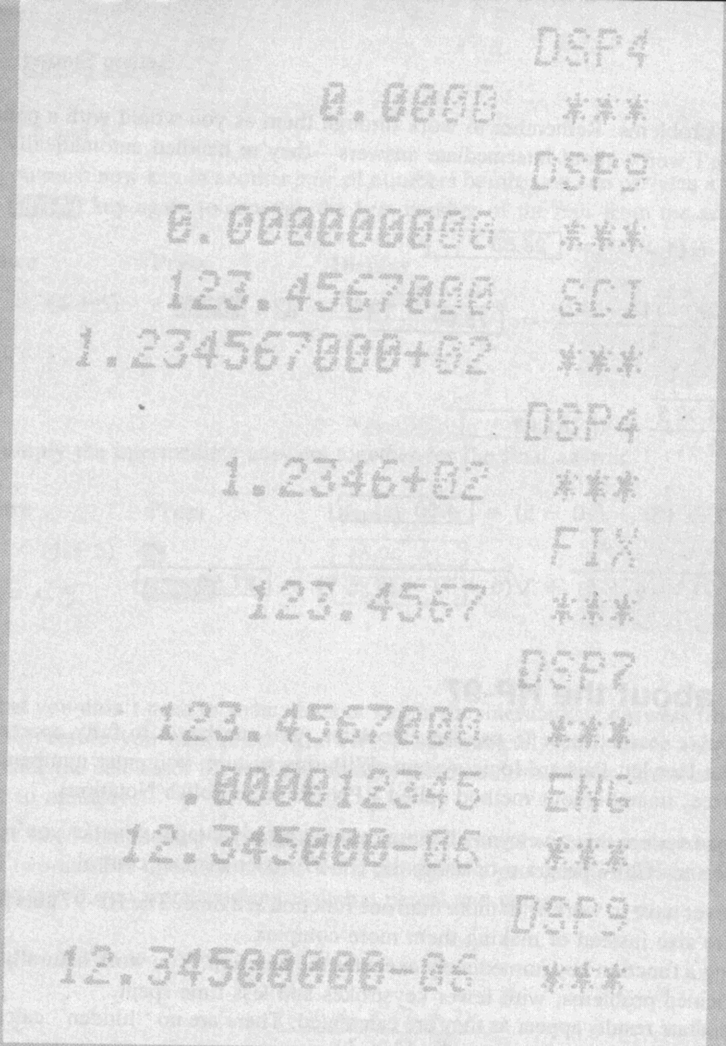
A Word about the HP-97

Now that you've learned how to use the calculator, you can begin to fully appreciate the benefits of the Hewlett-Packard logic system. With this system, you enter numbers using a parenthesis-free, unambiguous method called RPN (Reverse Polish Notation).

It is this unique system that gives you all these calculating advantages whether you're writing keystrokes for an HP-97 program or using the HP-97 under manual control:

- You never have to work with more than one function at a time. The HP-97 cuts problems down to size instead of making them more complex.
- Pressing a function key immediately executes the function. You work naturally through complicated problems, with fewer keystrokes and less time spent.
- Intermediate results appear as they are calculated. There are no "hidden" calculations, and you can check each step as you go.
- Intermediate results are automatically handled. You don't even have to print out long intermediate answers when you work a problem. (Of course, if you want intermediate answers, the HP-97 printer will record them in TRACE.)
- Intermediate answers are automatically inserted into the problem on a last-in, first-out basis. You don't have to remember where they are and then summon them.
- You can calculate in the same order that you do with pencil and paper. You don't have to think the problem through ahead of time.

The HP system takes a few minutes to learn. But you'll be amply rewarded by the ease with which the HP-97 solves the longest most complex equations. With HP, the investment of a few moments of learning yields a lifetime of mathematical dividends.



```
                 DSP4
      0.0000     ***
                 DSP9
  0.000000000    ***
  123.4567000    SCI
1.234567000+02   ***
                 DSP4
    1.2346+02    ***
                 FIX
     123.4567    ***
                 DSP7
  123.4567000    ***
   .000012345    ENG
 12.345000-06    ***
                 DSP9
12.34500000-06   ***
```

Section 2

Printer and Display Control

In the HP-97, you can select many different rounding options for display and printing of numbers. When you first turn on the HP-97, for example, the calculator "wakes up" with numbers appearing rounded to two decimal places. Thus, the fixed constant π, which is actually in the calculator as 3.141592654, will *appear in the display* as 3.14 (unless you tell the calculator to display the number rounded to a greater or lesser number of decimal places).

Although a number is normally shown to only two decimal places, the HP-97 always computes internally using each number as a 10-digit mantissa and a two-digit exponent of 10. For example, when you compute 2 × 3, you *see* the answer to only two decimal places:

Press	**Display**
2 ENTER↑ 3 ×	$\boxed{6.00}$

However, inside the calculator all numbers have 10-digit mantissas and two-digit exponents of 10. So the HP-97 *actually* calculates using full 10-digit numbers:

$$2.000000000 \times 10^{00} \text{ ENTER↑ } 3.000000000 \times 10^{00} \times$$

yields an answer that is actually carried to full 10 digits internally:

$$\boxed{\underbrace{6.00}_{\text{You see only these digits...}}\underbrace{0000000 \times 10^{00}}_{\text{...but these digits are also present.}}}$$

Display Control Keys

There are four keys, FIX, SCI, ENG, and DSP that allow you to control the manner in which numbers appear in the display and are printed in the HP-97. DSP followed by a number key changes the number of displayed digits without changing the format. FIX displays and prints numbers in fixed decimal point format, while SCI permits you to see numbers in scientific notation format. ENG displays and prints numbers in engineering notation, with exponents of 10 shown in multiples of three (e.g., 10^3, 10^{-6}, 10^{15}).

No matter which format or how many displayed digits you choose, these display control keys alter only the *manner* in which a number is displayed and printed in the HP-97. The actual number itself is not altered by any of the print options or the display control keys. No matter what type of display you select, the HP-97 always calculates internally with numbers consisting of full 10-digit mantissas multiplied by 10 raised to a two-digit exponent.

In NORM or TRACE, the printer immediately indicates when you change display format, and any new results will be shown in the new format.

Display Number Changes

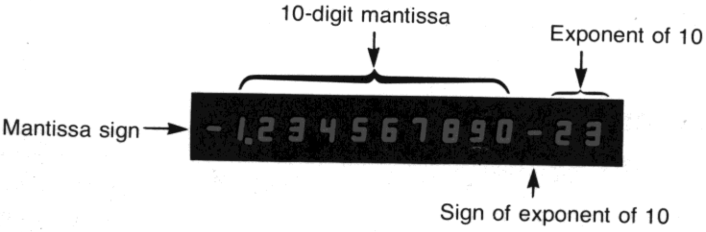
The DSP *(display)* key followed by a number key specifies the *number* of digits that your HP-97 will display and print. For example, when you turn the HP-97 ON, it "wakes up" with two digits displayed after the decimal point. Using the DSP key and the appropriate number key (0–9), you can display up to nine digits after the decimal point. For example:

Slide the Print Mode switch **MAN** **TRACE** **NORM** to MAN so that you can concentrate on the display changes.

Press	**Display**	
(Turn the calculator OFF, then ON.)	0.00	Calculator "wakes up" with two digits shown after the decimal point.
DSP 4	0.0000	Four digits shown after decimal point.
DSP 9	0.000000000	Nine digits shown after decimal point.
DSP 2	0.00	Two digits shown after decimal point.

In the next few pages, you will see how the DSP and number keys are used in conjunction with FIX, SCI, and ENG to display numbers in any of a wide variety of formats.

Scientific Notation Display



Scientific Notation Display

In scientific notation each number is displayed with a single digit to the left of the decimal point followed by a specified number of digits (up to nine) to the right of the decimal point and multiplied by a power of 10. Scientific notation is particularly useful when working with very large or small numbers.

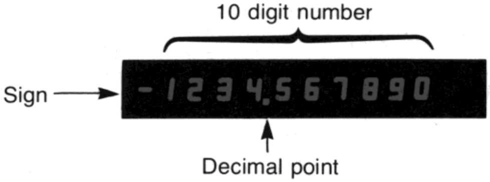
Scientific notation is selected by pressing SCI. The DSP key followed by a digit key is then used to specify the number of decimal places to which the number is rounded. The display is left-justified and includes trailing zeros within the setting selected by the DSP key. The printed copy is right-justified. To change the number of places displayed after the decimal point, use the DSP key followed by the appropriate number key.

For example:

Press	**Display**	
(Turn the calculator OFF, then ON.)	0.00	Calculator "wakes up" with two places displayed after the decimal point.
123.4567	123.4567	
SCI	1.23 02	Displays 1.23 × 10^2. Two decimal places shown after decimal point.
DSP 4	1.2346 02	Displays 1.2346 × 10^2. Notice that the display rounds if the first *hidden* mantissa digit is 5 or greater.
DSP 7	1.2345670 02	Displays 1.2345670 × 10^2.
DSP 9	1.234567000 02	Displays 1.234567000 × 10^2.
DSP 4	1.2346 02	Displays 1.2346 × 10^2.

Note: You can easily key in numbers in scientific notation format by using the EEX *(enter exponent)* key—more about this later.

Fixed Point Display



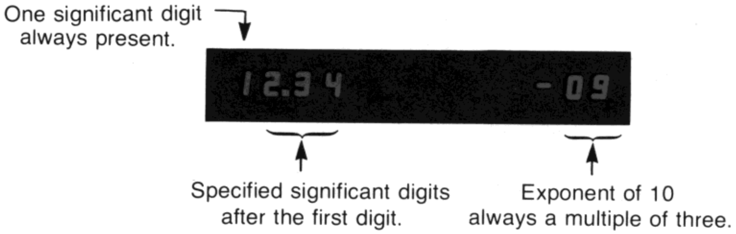
Fixed Point Display

When you first turn the HP-97 ON, the display you see is FIX 2—that is, fixed point display with two decimal places shown. In fixed point display, numbers are shown with a fixed number of displayed digits after the decimal point. The number begins at the left side of the display (or the right side of the printed tape) and includes trailing zeros within the setting selected. Fixed point format is selected from the keyboard with the FIX key. After you have specified fixed point format, you can use the DSP key followed by the appropriate number key (0–9) to select the number of places to which the display is rounded.

For example:

Press	**Display**	
123.4567	123.4567	
FIX	123.4567	Display is rounded to the four decimal places you specified earlier.
DSP 0	123.	
DSP 7	123.4567000	
DSP 1	123.5	Notice that the display rounds if the first *hidden* digit is 5 or greater.
DSP 2	123.46	Normal FIX 2 display.

Engineering Notation Display



Engineering Notation Display

Engineering notation allows all numbers to be shown with exponents of 10 that are multiples of three (e.g., 10^3, 10^{-6}, 10^{12}). This is particularly useful in scientific and engineering calculations, where units of measure are often specified in multiples of three. See the prefix chart below.

Multiplier	Prefix	Symbol
10^{12}	tera	T
10^9	giga	G
10^6	mega	M
10^3	kilo	k
10^{-3}	milli	m
10^{-6}	micro	μ
10^{-9}	nano	n
10^{-12}	pico	p
10^{-15}	femto	f
10^{-18}	atto	a

Engineering notation is selected by pressing ENG. The first significant digit is *always* present in the display. When you press DSP followed by a number key, you specify the number of *additional* displayed digits after the first one. The decimal point always appears in the display.

For example:

Press	**Display**	
.000012345	.000012345	
ENG	12.3 -06	Engineering notation display. Since you had specified DSP 2 in the previous example, the number appears here rounded off to two significant digits after the omnipresent first one. Power of 10 is proper multiple of three.
DSP 3	12.35 -06	Display is rounded off to third significant digit after the first one.
DSP 9	12.34500000-06	
DSP 0	10. -06	Display rounded off to first significant digit.

Notice that rounding can occur to the *left* of the decimal point, as in the case of ENG 0 specified above.

When engineering notation has been selected, the decimal point shifts to show the mantissa as units, tens, or hundreds in order to maintain the exponent of 10 as a multiple of three. For example, multiplying the number now in the calculator by 10 causes the decimal point to shift to the right without altering the exponent of 10:

Press	**Display**	
DSP 2	12.3 -06	ENG 2 display.
10 ×	123. -06	ENG 2 display.

However, multiplying again by 10 causes the exponent to shift to another multiple of three and the decimal point to move to the units position. Since you specified ENG 2 earlier, the HP-97 maintains two significant digits after the first one when you multiply by 10:

Press	**Display**	
10 ×	1.23 -03	Decimal point shifts. Power of 10 shifts to 10^{-3}. Display maintains two significant digits after the first one.

Format of Printed Numbers

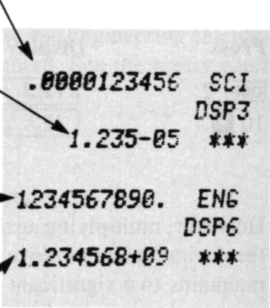
When using the printer, whether you are in MANUAL or NORMAL modes (where you must press **PRINT x** to see answers) or in TRACE (where the HP-97 automatically prints answers as they are calculated), printed numbers can be shown in any display format—fixed point, scientific notation, or engineering notation. By selecting the display format, you also select the print format.

Results from your HP-97 are always displayed and printed in the format that you have chosen. The three-asterisk label that you see printed next to a result is a guarantee that it is in the chosen display format. Although numbers in the display are left-justified, printed numbers are right-justified.

Numbers that you key in—that is, numbers that are *not* the results of operations—are also printed by the HP-97. When you key in a number with the Print Mode switch set to NORM or TRACE, the HP-97 does not print it until you change display format or press a function key. Then the number is printed exactly *as you keyed it in*. (One case is an exception to this rule—more about that later.) A number that you keyed in is not the result of an operation, and no asterisks are printed to its right. Subsequent *results,* of course, are printed in the selected format with a three-asterisk label. For example:

Slide the Print Mode switch MAN TRACE NORM to NORM.

Press	Display	
.0000123456	.0000123456	
SCI **DSP** 3	1.235 -05	When you press any function, the number is first printed just as you keyed it in.
PRINT x	1.235 -05	Results of functions, including display formatting, are printed in the selected format.
1234567890	1234567890.	
ENG **DSP** 6	1.234568 09	The number is printed as you keyed it in.
PRINT x	1.234568 09	The three-asterisk label guarantees that the number is now in the selected format.



```
.0000123456   SCI
             DSP3
   1.235-05   ***
1234567890.   ENG
             DSP6
1.234568+09   ***
```

(Notice that the HP-97 prints a + sign to show you positive exponents of 10.)

Thus, whenever you key in a number, the HP-97 prints it just as you keyed it in; *then* the format is changed. It is easy for you to reconstruct your calculation because your exact inputs are identifiable from your printed copy.

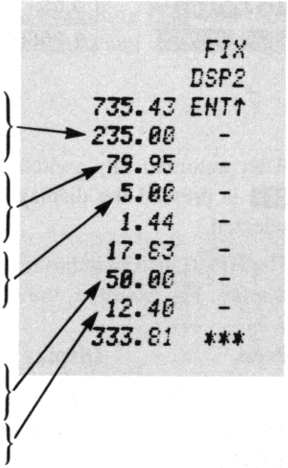
When you have keyed in a number, there is one time that the HP-97 will change its format *before* printing. If you have specified fixed point notation (by turning the calculator OFF, then ON, or by pressing FIX) and the number keyed in is also in fixed point format (i.e., you have not pressed EEX), the HP-97 will attempt to align the decimal points for easy readability on your printed copy. It will do this in fixed point notation by printing the number that you keyed in in the *specified* format (if the number can be printed without truncating), adding trailing zeros if necessary.

This feature permits you to key in numbers in fixed point notation and line up the decimal points in the printed record of your calculations.

Example: You begin the month with a balance of $735.43 in your checking account. During the month, you write checks for $235, $79.95, $5, $1.44, $17.83, $50, and $12.40. Calculate the closing balance for the account and preserve a printed record of your calculations.

First, ensure that the Print Mode switch MAN TRACE NORM is set to NORM.

Press	Display	
FIX DSP 2	0.00	Sets FIX 2 display mode. (Display shown assumes that no results remain from previous example.)
735.43 ENTER↑	735.43	
235 −	500.43	Two extra zeros printed so that decimal points will line up.
79.95 −	420.48	The number is printed exactly as you keyed it in.
5 −	415.48	Two extra zeros printed.
1.44 −	414.04	
17.83 −	396.21	
50 −	346.21	Two extra zeros printed.
12.4 −	333.81	One extra zero printed.
PRINTx	333.81	Closing balance.



```
           FIX
          DSP2
735.43    ENT↑
235.00       -
 79.95       -
  5.00       -
  1.44       -
 17.83       -
 50.00       -
 12.40       -
333.81     ***
```

You need not worry about "losing" digits on the printed copy. The HP-97 printer will never truncate digits (not even extra zeros) that you have keyed in. For example, if you wanted to set aside 5/10000 of the closing balance of your account for a present for your sister-in-law:

Press	Display	
.0005	.0005	
×	0.17	Entire number is printed—not rounded to FIX 2.
PRINTx	0.17	Amount set aside for sister-in-law's gift. Result of function is rounded to FIX 2.

```
   .0005   ×
    0.17  ***
```

Automatic Display Switching

The HP-97 switches the display from fixed point notation to full scientific notation (SCI 9) whenever the number is too large or too small to be seen with a fixed decimal point. This feature keeps you from missing unexpectedly large or small answers. For example, if you try to solve $(.05)^3$ in normal FIX 2 display, the answer is automatically shown in scientific notation:

Press	Display	
CLx	0.00	Normal FIX 2 display from previous example.
.05 ENTER↑	0.05	
3 yˣ PRINTx	1.250000000-04	Display automatically switched to SCI 9 to show answer.

```
                   CLX
              .05 ENT↑
             3.00   Yˣ
1.250000000-04     ***
```

After automatically switching from fixed to scientific, when a new number is keyed in or CLx is pressed the display automatically reverts back to the fixed point display originally selected.

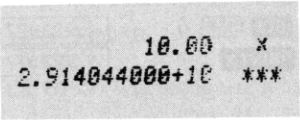
The HP-97 also switches to scientific notation if the answer is too large ($\geqslant 10^{10}$) for fixed point display. For example, the display will not switch from fixed if you solve 1582000 × 1842:

Press	Display	
1582000 ENTER↑	1582000.00	
1842 ×	2914044000.	Fixed point format.
PRINTx	2914044000.	

```
1582000.00 ENT↑
   1842.00    ×
2914044000.  ***
```

However, if you multiply the result by 10, the answer is too large for fixed point notation, and the calculator display switches automatically to scientific notation:

Press	Display	
10 **×** **PRINTx**	2.914044000 10	Scientific notation format.



```
          10.00    ×
2.914044000+10   ***
```

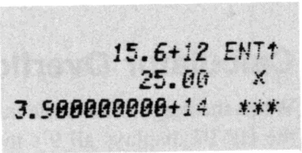
Notice that automatic switching is between fixed and scientific notation display modes only—engineering notation display must be selected from the keyboard.

Keying In Exponents of Ten

You can key in numbers multiplied by powers of 10 by pressing **EEX** *(enter exponent of 10)* followed by number keys to specify the exponent of 10. For example, to key in 15.6 trillion (15.6×10^{12}), and multiply it by 25:

Press	Display	
15.6	15.6	
EEX	15.6 00	
12	15.6 12	(This means 15.6×10^{12}.)

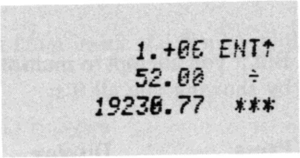
Now Press	Display
ENTER↑	1.560000000 13
25 **×** **PRINTx**	3.900000000 14



```
     15.6+12 ENT↑
       25.00    ×
3.900000000+14 ***
```

You can save time when keying in exact powers of 10 by merely pressing **EEX** and then pressing the desired power of 10. For example, key in 1 million (10^6) and divide by 52.

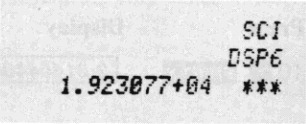
Press	Display	
EEX	1. 00	You do not have to key in the number 1 before pressing **EEX** when the number is an exact power of 10.
6	1. 06	
ENTER↑	1000000.00	Since you have not specified scientific notation, the number reverts to fixed point notation when you press **ENTER↑**.
52 **÷** **PRINTx**	19230.77	



```
  1.+06 ENT↑
  52.00    ÷
19230.77 ***
```

To see your answer in scientific notation with six decimal places:

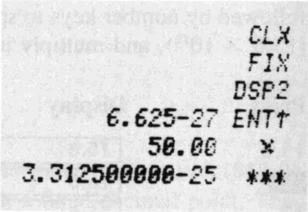
Press	Display
SCI DSP 6	1.923077 04
PRINTx	1.923077 04



```
                SCI
               DSP6
1.923077+04     ***
```

To key in negative exponents of 10, key in the number, press EEX, press CHS to make the exponent negative, then key in the power of 10. For example, key in Planck's constant (h)—roughly, 6.625 × 10^{-27} erg sec.—and multiply it by 50.

Press	Display	
CLx	0.000000 00	
FIX DSP 2	0.00	
6.625 EEX	6.625 00	
CHS	6.625 -00	
27	6.625 -27	
ENTER↑	6.625000000-27	
50 × PRINTx	3.312500000-25	Erg sec.

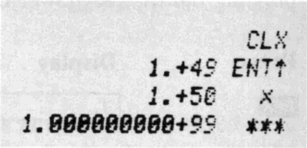


```
                CLX
                FIX
               DSP2
  6.625-27     ENT↑
     50.00        ×
3.312500000-25  ***
```

Calculator Overflow

When the magnitude of the number in the display would be greater than 9.999999999 × 10^{99}, the HP-97 displays all 9's to indicate that the problem has exceeded the calculator's range. For example, if you solve (1 × 10^{49}) × (1 × 10^{50}), the HP-97 will display the answer:

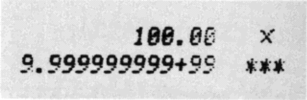
Press	Display
CLx	0.00
EEX 49 ENTER↑	1.000000000 49
EEX 50 ×	1.000000000 99
PRINTx	1.000000000 99



```
                CLX
     1.+49     ENT↑
     1.+50        ×
1.000000000+99  ***
```

But if you attempt to multiply the above result by 100, the HP-97 display indicates overflow by showing you all 9's:

Press	Display	
100 × PRINTx	9.999999999 99	Overflow indication.



```
    100.00        ×
9.999999999+99  ***
```

Error Display

If you happen to key in an improper operation, or if a magnetic card fails to read properly, the word `Error` will appear in the display. In addition, if the Print Mode switch MAN TRACE NORM is set to NORM or TRACE, the printer will print `ERROR`.

For example, if you attempt to calculate the square root of −4, the HP-97 will recognize it as an improper operation:

Ensure that the Print Mode switch MAN TRACE NORM is set to NORM.

Press	Display
4 **CHS**	-4.
√x	Error

```
-4.00    √X
       ERROR
```

Pressing *any key* clears the error and is *not* executed, while pressing the paper advance pushbutton clears the error and *is* executed. The number that was in the display before the error-causing function is returned to the display so that you can see it. Sliding the PRGM-RUN switch to PRGM also clears the error. When the PRGM-RUN switch is then returned to the RUN position, the number that was in the display before the error-causing function is again returned there. The rest of the calculator remains unchanged. To clear the error:

Press	Display
CLx	-4.00

As you know, when the calculator is out of paper, any operation that would normally cause the calculator to print instead causes an `Error` display. Notice that when the Print Mode switch is set to NORM or ALL, function keys like **ENTER↑** or **×** cause an `Error` display when the calculator is out of paper.

All those operations that cause an error condition are listed in appendix C.

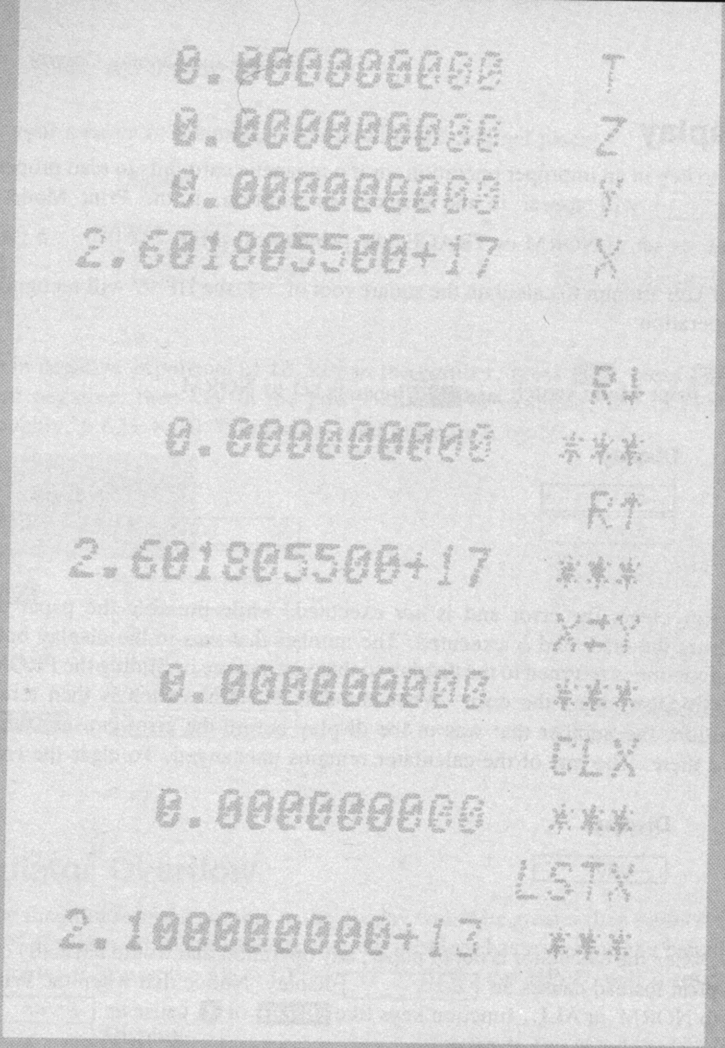
Low Power Display

When you are operating the HP-97 from battery power, a red lamp inside the display will glow to warn you that the battery is close to discharge.

● 6.02 23

Low Power Display

You must then connect the ac adapter/recharger to the calculator and operate from ac power, or you must substitute a fully charged battery pack for the one that is in the calculator. Refer to appendix B for descriptions of these operations.



```
    0.000000000   T
    0.000000000   Z
    0.000000000   Y
 2.601805500+17   X

                 R↓
    0.000000000  ***
                 R↑
 2.601805500+17  ***
                X⇄Y
    0.000000000  ***
                CLX
    0.000000000  ***
               LSTX
 2.100000000+13  ***
```

Section 3

The Automatic Memory Stack

The Stack

Automatic storage of intermediate results is the reason that the HP-97 slides so easily through the most complex equations. And automatic storage is made possible by the Hewlett-Packard automatic memory stack.

Initial Display

(You can work through this section with the Print Mode switch at any setting you desire. However, the printed tapes that illustrate the examples in this section were created with the Print Mode switch **MAN TRACE NORM** set to NORM.)

When you first switch the calculator ON, the display shows 0.00 . This represents the contents of the display, or "X-register."

Basically, numbers are stored and manipulated in the machine "registers." Each number, no matter how few digits (e.g., 0, 1, or 5) or how many (e.g., 3.141592654, −23.28362, or 2.87148907 × 10^{27}), occupies one entire register.

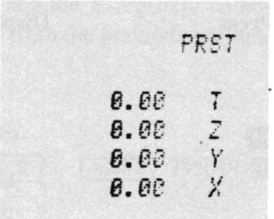
The displayed X-register, which is the only visible register, is one of four registers inside the calculator that are positioned to form the automatic memory stack. We label these registers X, Y, Z, and T. They are "stacked" one on top of the other with the displayed X-register on the bottom. When the calculator is switched ON, these four registers are cleared to 0.00.

Switch the HP-97 OFF, then ON.

Name	Register	
T	0.00	
Z	0.00	
Y	0.00	
X	0.00	Always displayed.

You can view the contents of the entire stack at any time by printing them using the PRINT: STACK *(print stack)* key.

Press	Display
f PRINT: STACK	0.00



```
                PRST

      0.00    T
      0.00    Z
      0.00    Y
      0.00    X
```

Notice that f PRINT: STACK, like PRINTx and the other print functions, operates regardless of the position of the Print Mode switch.

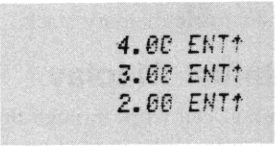
Manipulating Stack Contents

The R↓ *(roll down)*, R↑ *(roll up)*, and x≷y *(x exchange y)* keys allow you to review the stack contents or to shift data within the stack for computation at any time.

Reviewing the Stack

To see how the R↓ key works, first load the stack with numbers 1 through 4 by pressing:

4 ENTER↑ 3 ENTER↑ 2 ENTER↑ 1



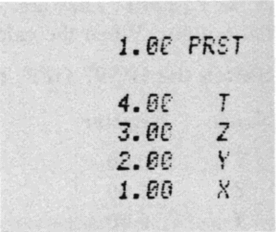
```
4.00 ENT↑
3.00 ENT↑
2.00 ENT↑
```

The numbers that you keyed in are now loaded into the stack, and its contents look like this:

T	*4.00*	
Z	*3.00*	
Y	*2.00*	
X	*1.*	Display

To see the contents of the stack now:

Press	**Display**
f PRINT: STACK	*1.00*

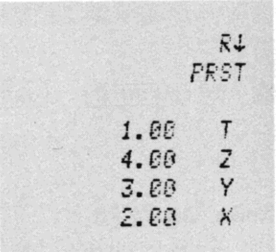


```
1.00 PRST

4.00    T
3.00    Z
2.00    Y
1.00    X
```

When you press the R↓ key, the stack contents shift downward one register. So the last number that you have keyed in will be rotated around to the T-register when you press R↓. When you press R↓ again, the stack contents again roll downward one register.

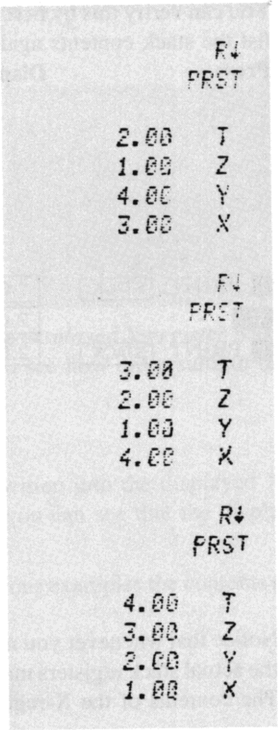
To see how the R↓ key operates, press f PRINT: STACK to list the stack contents after each press of the R↓ key:

Press	**Display**
R↓	
f PRINT: STACK	*2.00*



```
       R↓
     PRST

1.00    T
4.00    Z
3.00    Y
2.00    X
```

Press	Display
R↓ f PRINT: STACK	3.00
R↓ f PRINT: STACK	4.00
R↓ f PRINT: STACK	1.00



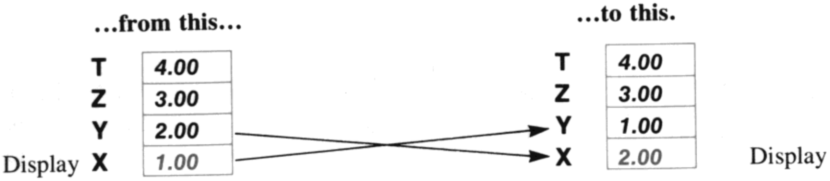
```
           R↓
         PRST

   2.00    T
   1.00    Z
   4.00    Y
   3.00    X

           R↓
         PRST

   3.00    T
   2.00    Z
   1.00    Y
   4.00    X

           R↓
         PRST

   4.00    T
   3.00    Z
   2.00    Y
   1.00    X
```

Once again the number 1.00 is in the displayed X-register. Four presses of the R↓ key roll the stack down four times, returning the contents of the stack to their original registers.

You can also manipulate the stack contents using the R↑ (*roll up*) key. This key rolls the stack contents *up* instead of down, but it otherwise operates in the same manner as the R↓ key.

Exchanging x and y

The x⇄y (*x exchange y*) key exchanges the contents of the X- and the Y-registers without affecting the Z- and T-registers. If you press x⇄y with data intact from the previous example, the numbers in the X- and Y-registers will be changed...



You can verify this by first listing the stack contents and then pressing x≷y. To see the results, list the stack contents again:

Press	Display
f PRINT: STACK	1.00
x≷y	2.00
f PRINT: STACK	2.00

```
         PRST

4.00     T
3.00     Z
2.00     Y
1.00     X

         X⇄Y
         PRST

4.00     T
3.00     Z
1.00     Y
2.00     X
```

Notice that whenever you move numbers in the stack using one of the data manipulation keys, the actual stack registers maintain their positions. Only the *contents* of the registers are shifted. The contents of the X-register are always displayed.

Clearing the Display

When you press CLx *(clear x),* the displayed X-register is cleared to zero. No other register is affected when you press CLx.

Press CLx now, and the stack contents are changed…

… from this …

T	4.00	
Z	3.00	
Y	1.00	
X	2.00	Display

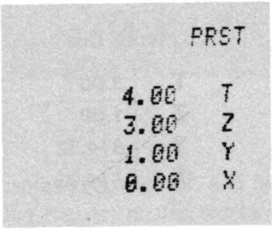
… to this.

T	4.00	
Z	3.00	
Y	1.00	
X	0.00	Display

```
          CLX
```

You can verify that only the X-register contents are affected by listing the stack contents after you have pressed **CLx**:

Press	**Display**
f PRINT: **STACK**	0.00



```
              PRST

         4.00  T
         3.00  Z
         1.00  Y
         0.00  X
```

Although it may be comforting, *it is never necessary to clear the displayed X-register when starting a new calculation*. This will become obvious when you see how old results in the stack are automatically lifted by new entries.

The ENTER↑ Key

When you key a number into the calculator, its contents are written into the displayed X-register. For example, if you key in the number 314.32 now, you can see that the display contents are altered.

When you key in 314.32 with the stack contents intact from previous examples the contents of the stack registers are changed...

... from this ...

T	4.00	
Z	3.00	
Y	1.00	
X	0.00	Display

... to this.

T	4.00	
Z	3.00	
Y	1.00	
X	314.32	Display

In order to key in another number at this point, you must first terminate digit entry—i.e., you must indicate to the calculator that you have completed keying in the first number and that any new digits you key in are part of a new number.

Use the **ENTER↑** key to separate the digits of the first number from the digits of the second.

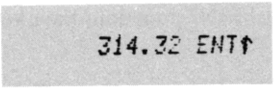
When you press the **ENTER↑** key, the contents of the stack registers are changed...

... from this ...

T	4.00	
Z	3.00	
Y	1.00	
X	314.32	Display

... to this.

T	3.00	
Z	1.00	
Y	314.32	
X	314.32	Display



```
314.32 ENT↑
```

As you can see, the number in the displayed X-register is copied into Y. The numbers in Y and Z have also been transferred to Z and T, respectively, and the number in T has been lost off the top of the stack.

Immediately after pressing ENTER↑, the X-register is prepared for a new number, and that new number writes over the number in X. For example, key in the number 543.28 and the contents of the stack registers change...

	... from this ...			**... to this.**	
T	*3.00*		**T**	*3.00*	
Z	*1.00*		**Z**	*1.00*	
Y	*314.32*		**Y**	*314.32*	
X	*314.32*	Display	**X**	*543.28*	Display

CLx replaces any number in the display with zero. Any new number then writes over the zero in X.

For example, if you had meant to key in 689.4 instead of 543.28, you would press CLx now to change the stack...

	... from this ...			**... to this.**	
T	*3.00*		**T**	*3.00*	
Z	*1.00*		**Z**	*1.00*	
Y	*314.32*		**Y**	*314.32*	
X	*543.28*	Display	**X**	*0.00*	Display

and then key in 689.4 to change the stack...

	... from this ...			**... to this.**	
T	*3.00*		**T**	*3.00*	
Z	*1.00*		**Z**	*1.00*	
Y	*314.32*		**Y**	*314.32*	
X	*0.00*	Display	**X**	*689.4*	Display

Notice that numbers in the stack do not move when a number is keyed in immediately after you press ENTER↑, CLx, or one of the PRINT functions. However, numbers in the stack *do* lift upward when a new number is keyed in immediately after you press most other functions, including R↑, R↓, and x≷y. See appendix D, Stack Lift and LAST X, for a complete list of the operations that cause the stack to lift. (If you follow a regular function like R↓ or x^2 with a PRINT function, then key in a number, the stack will lift.)

One-Number Functions and the Stack

One-number functions execute upon the number in the X-register only, and the contents of the Y-, Z-, and T-registers are unaffected when a one-number function key is pressed.

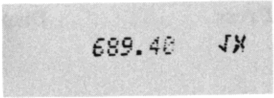
For example, with numbers positioned in the stack as in the earlier example, pressing the $\sqrt{x}$ key changes the stack contents...

... from this ...

T	3.00	
Z	1.00	
Y	314.32	
X	689.4	Display

... to this.

T	3.00	
Z	1.00	
Y	314.32	
X	26.26	Display



```
689.40      √X
```

The one-number function executes upon only the number in the displayed X-register, and the answer writes over the number that was in the X-register. No other register is affected by a one-number function.

Two-Number Functions and the Stack

Hewlett-Packard calculators do arithmetic by positioning the numbers in the stack the same way you would on paper. For instance, if you wanted to add 34 and 21 you would write 34 on a piece of paper and then write 21 underneath it, like this:

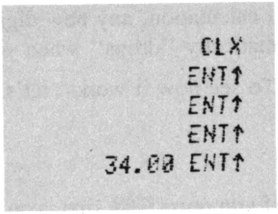
$$\begin{array}{r} 34 \\ \underline{21} \end{array}$$

and then you would add, like this:

$$\begin{array}{r} 34 \\ \underline{+21} \\ 55 \end{array}$$

Numbers are positioned the same way in the HP-97. Here's how it is done. (As you know, it is not necessary to remove earlier results from the stack before beginning a new calculation, but for clarity, the following example is shown with the stack cleared to all zeros initially. If you want the contents of your stack registers to match the ones here, first clear the stack by using the CLx and ENTER↑ keys to fill the stack with zeros.)

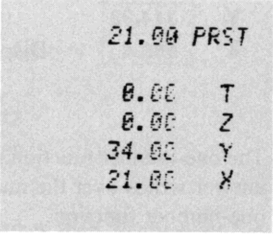
Press	Display	
CLx	0.00	Stack cleared to zeros initially.
ENTER↑	0.00	
ENTER↑	0.00	
ENTER↑	0.00	
34	34.	34 is keyed into X.
ENTER↑	34.00	34 is copied into Y.
21	21.	21 writes over the 34 in X.



```
           CLX
          ENT↑
          ENT↑
          ENT↑
    34.00 ENT↑
```

Use the f PRINT: STACK function to see how 34 and 21 are sitting vertically in the stack as shown below:

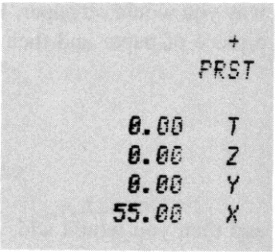
Press	Display
f PRINT: STACK	21.00



```
    21.00 PRST

     0.00    T
     0.00    Z
    34.00    Y
    21.00    X
```

Since the two numbers are now sitting vertically in the stack, we can add. Add the two numbers, then print the contents of the stack again to see how the two numbers combine and the answer is seen in the displayed X-register.

Press	Display	
+	55.00	The answer.
f PRINT: STACK	55.00	



```
             +
          PRST

     0.00    T
     0.00    Z
     0.00    Y
    55.00    X
```

The simple old-fashioned math notation helps explain how to use your calculator. Both numbers are always positioned in the stack in the natural order first, then the operation is executed when the function key is pressed. *There are no exceptions to this rule*. Subtraction, multiplication, and division work the same way. In each case, the data must be in the proper position before the operation can be performed.

Chain Arithmetic

You've already learned how to key numbers into the calculator and perform calculations with them. In each case you first needed to position the numbers in the stack manually using the ENTER↑ key. However, the stack also performs many movements automatically. These automatic movements add to its computing efficiency and ease of use, and it is these movements that automatically store intermediate results. The stack automatically ''lifts'' every calculated number in the stack when a new number is keyed in because it knows that after it completes a calculation, any new digits you key in are a part of a new number. Also, the stack automatically ''drops'' when you perform a two-number operation.

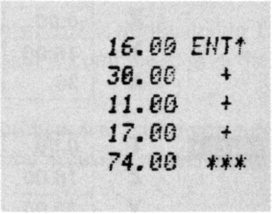
To see how it works, let's solve

$$16 + 30 + 11 + 17 = ?$$

If you press CLx first, you will begin with zeros in all the stack registers, as in the example below, but of course, you can also do the calculation without first clearing the stack.

Remember, too, that you can always monitor the contents of the stack at any time by using the f PRINT: STACK function.

Press	**Stack Contents**		
16	T	0.00	16 is keyed into the displayed X-register.
	Z	0.00	
	Y	0.00	
	X	16.	
ENTER↑	T	0.00	16 is copied into Y.
	Z	0.00	
	Y	16.00	
	X	16.00	
30	T	0.00	30 writes over the 16 in X.
	Z	0.00	
	Y	16.00	
	X	30.	
+	T	0.00	16 and 30 are added together. The answer, 46, is displayed.
	Z	0.00	
	Y	0.00	
	X	46.00	
11	T	0.00	11 is keyed into the displayed X-register. The 46 in the stack is automatically raised.
	Z	0.00	
	Y	46.00	
	X	11.	
+	T	0.00	46 and 11 are added together. The answer, 57, is displayed.
	Z	0.00	
	Y	0.00	
	X	57.00	
17	T	0.00	17 is keyed into the X-register. 57 is automatically entered into Y.
	Z	0.00	
	Y	57.00	
	X	17.	
+ PRINTx	T	0.00	57 and 17 are added together for the final answer.
	Z	0.00	
	Y	0.00	
	X	74.00	



```
16.00 ENT↑
30.00    +
11.00    +
17.00    +
74.00  ***
```

After any calculation or number manipulation, the stack automatically lifts when a new number is keyed in. Because operations are performed when the operations are pressed, the length of such chain problems is unlimited unless a number in one of the stack registers exceeds the range of the calculator (up to $9.999999999 \times 10^{99}$).

In addition to the automatic stack lift after a calculation, the stack automatically drops during calculations involving both the X- and Y-registers. It happens in the above example, but let's do the problem differently to see this feature more clearly. For clarity, first press **CLx** to clear the X-register. Now, again solve 16 + 30 + 11 + 17 = ?

Press	Stack Contents		
16	T	0.00	16 is keyed into the displayed X-register.
	Z	0.00	
	Y	0.00	
	X	16.	
ENTER↑	T	0.00	16 is copied into Y.
	Z	0.00	
	Y	16.00	
	X	16.00	
30	T	0.00	30 is written over the 16 in X.
	Z	0.00	
	Y	16.00	
	X	30.	
ENTER↑	T	0.00	30 is entered into Y. 16 is lifted up to Z.
	Z	16.00	
	Y	30.00	
	X	30.00	
11	T	0.00	11 is keyed into the displayed X-register.
	Z	16.00	
	Y	30.00	
	X	11.	
ENTER↑	T	16.00	11 is copied into Y. 16 and 30 are lifted up to T and Z respectively.
	Z	30.00	
	Y	11.00	
	X	11.00	
17	T	16.00	17 is written over the 11 in X.
	Z	30.00	
	Y	11.00	
	X	17.	

+	T	16.00	17 and 11 are added together and the rest of the stack drops. 16 drops to Z and is also duplicated in T. 30 and 28 are ready to be added.
	Z	16.00	
	Y	30.00	
	X	28.00	

```
16.00 ENT↑
30.00 ENT↑
11.00 ENT↑
17.00    +
         +
         +
74.00  ***
```

+	T	16.00	30 and 28 are added together and the stack drops again. Now 16 and 58 are ready to be added.
	Z	16.00	
	Y	16.00	
	X	58.00	

+ PRINTx	T	16.00	16 and 58 are added together for the final answer and the stack continues to drop.
	Z	16.00	
	Y	16.00	
	X	74.00	

The same dropping action also occurs with −, × and ÷. The number in T is duplicated in T and drops to Z, the number in Z drops to Y, and the numbers in Y and X combine to give the answer, which is visible in the X-register.

This automatic lift and drop of the stack give you tremedous computing power, since you can retain and position intermediate results in long calculations without the necessity of reentering the numbers.

Order of Execution

When you see a problem like this one:

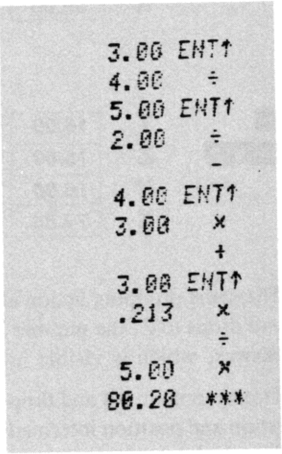
$$5 \times \left[(3 \div 4) - (5 \div 2) + (4 \times 3)\right] \div (3 \times .213)$$

you must decide where to begin before you ever press a key.

Experienced HP calculator users have determined that by starting every problem at its innermost number or parentheses and working outward, just as you would with paper and pencil, you maximize the efficiency and power of your HP calculator. Of course, with the HP-97 you have tremendous versatility in the order of execution.

For example, you could work the problem above by beginning at the left side of the equation and simply working through it in left-to-right order. All problems cannot be solved using left-to-right order, however, and the best order for solving any problem is to begin with the innermost parentheses and work outward. So, to solve the problem above:

Press	Display	
3	3.	
ENTER↑	3.00	
4	4.	
	0.75	Intermediate answer for (3 ÷ 4).
5	5.	
ENTER↑	5.00	
2	2.	
÷	2.50	Intermediate answer for (5 ÷ 2).
−	−1.75	Intermediate answer for (3 ÷ 4) − (5 ÷ 2).
4	4.	
ENTER↑	4.00	
3	3.	
×	12.00	Intermediate answer for (4 × 3).
+	10.25	Intermediate answer for (3 ÷ 4) − (5 ÷ 2) + (4 × 3).
3	3.	
ENTER↑	3.00	
.213	.213	
×	0.64	Intermediate answer for (3 × .213).
÷	16.04	
5	5.	The first number is keyed in.
×	80.20	The final answer.
PRINT X	80.20	



```
3.00 ENT↑
4.00   ÷
5.00 ENT↑
2.00   ÷
       -
4.00 ENT↑
3.00   ×
       +
3.00 ENT↑
.213   ×
       ÷
5.00   ×
80.20 ***
```

LAST X

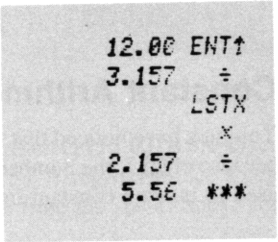
In addition to the four stack registers that automatically store intermediate results, the HP-97 also contains a separate automatic register, the LAST X register. This register preserves the value that was last displayed in the X-register before the performance of a function. To place the contents of the LAST X register into the display again, press f LAST X.

Recovering from Mistakes

LAST X makes it easy to recover from keystroke mistakes, such as pressing the wrong function key or keying in the wrong number.

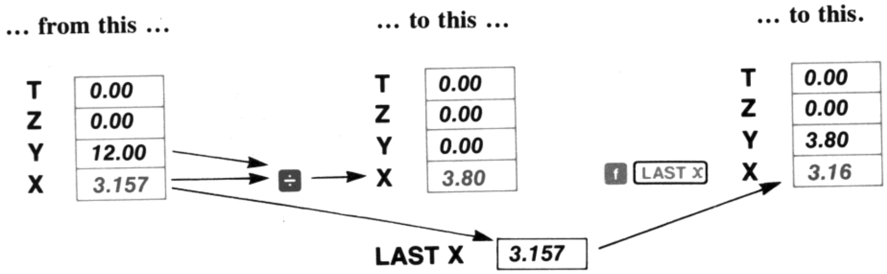
Example: Divide 12 by 2.157 after you have mistakenly divided by 3.157.

Press	**Display**	
12	12.	
ENTER↑	12.00	
3.157 ÷	3.80	Oops! You made a mistake.
f LAST X	3.16	Retrieves that last entry (3.157).
×	12.00	You're back at the beginning.
2.157 ÷	5.56	The correct answer.
PRINTx	5.56	



```
    12.00 ENT↑
    3.157    ÷
          LSTX
             ×
    2.157    ÷
     5.56  ***
```

In the above example, when the first ÷ is pressed, followed by f LAST X, the contents of the stack and LAST X registers are changed...



This makes possible the correction illustrated in the example above.

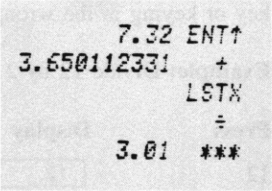
Recovering a Number for Calculation

The LAST X register is useful in calculations where a number occurs more than once. By recovering a number using LAST X, you do not have to key that number into the calculator again.

Example: Calculate

$$\frac{7.32 + 3.650112331}{3.650112331}$$

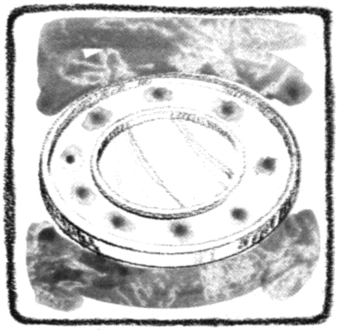
Press	Display	
7.32	7.32	
ENTER↑	7.32	
3.650112331	3.650112331	
+	10.97	Intermediate answer.
f LAST X	3.65	Recalls 3.650112331 to X-register.
÷	3.01	The answer.
PRINT x	3.01	



```
       7.32 ENT↑
3.650112331    +
            LSTX
               ÷
       3.01  ***
```

Constant Arithmetic

You may have noticed that whenever the stack drops because of a two-number operation (not because of R↓), the number in the T-register is reproduced there. This stack operation can be used to insert a constant into a problem.

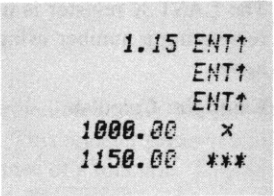


Example: A bacteriologist tests a certain strain whose population typically increases by 15% each day. If he starts a sample culture of 1000, what will be the bacteria population at the end of each day for six consecutive days?

Method: Put the growth factor (1.15) in the Y-, Z-, and T-registers and put the original population (1000) in the X-register. Thereafter, you get the new population whenever you press ×. Try working this problem with the Print Mode switch set to TRACE so that you'll have a record of all the answers without pressing PRINT x each time.

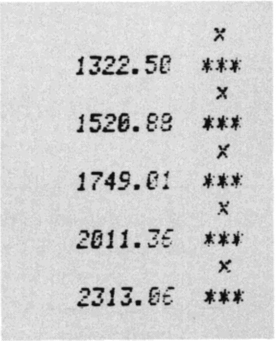
Slide the Print Mode switch MAN TRACE NORM to TRACE.

Press	Display	
1.15	1.15	Growth factor.
ENTER↑	1.15	
ENTER↑	1.15	
ENTER↑	1.15	Growth factor now in T.
1000	1000.	Starting population.
×	1150.00	Population after 1st day.



```
   1.15 ENT↑
        ENT↑
        ENT↑
1000.00    ×
1150.00  ***
```

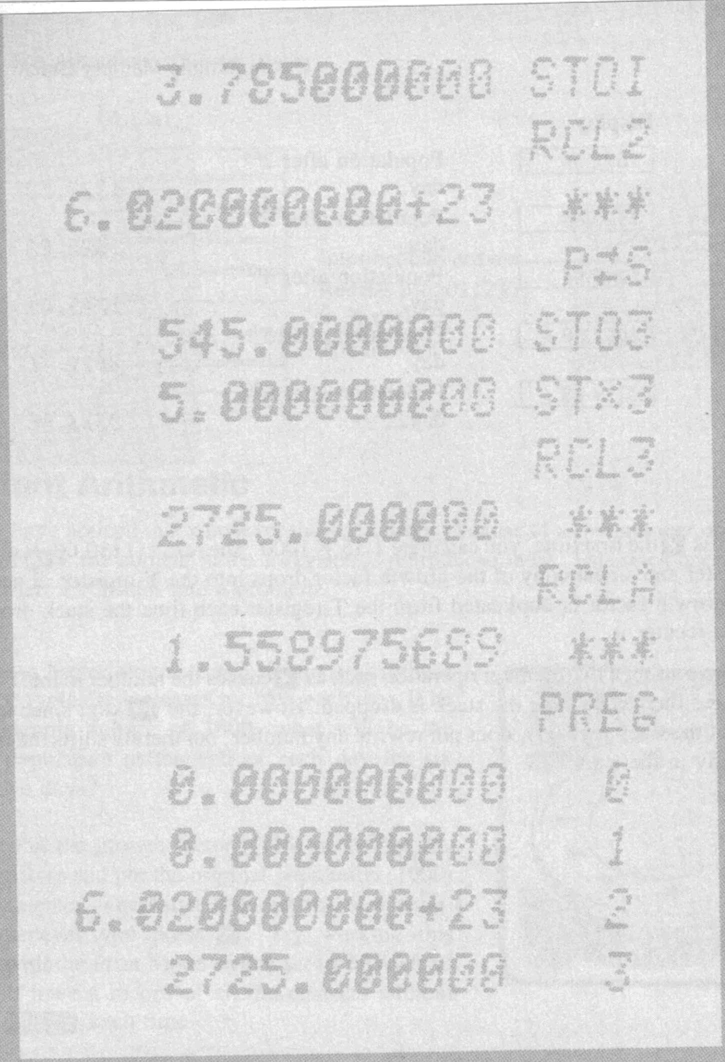
Press	Display	
×	1322.50	Population after 2nd day.
×	1520.88	Population after 3rd day.
×	1749.01	Population after 4th day.
×	2011.36	Population after 5th day.
×	2313.06	Population after 6th day.



```
          ×
1322.50 ***
          ×
1520.88 ***
          ×
1749.01 ***
          ×
2011.36 ***
          ×
2313.06 ***
```

When you press × the first time, you calculate 1.15 × 1000. The result (1150.00) is displayed in the X-register and a new copy of the growth factor drops into the Y-register. Since a new copy of the growth factor is duplicated from the T-register each time the stack drops, you never have to reenter it.

Notice that performing a two-number operation such as × causes the number in the T-register to be duplicated there each time the stack is dropped. However, the R↓ key, since it rotates the contents of the stack registers, does not rewrite any number, but merely shifts the numbers that are already in the stack.



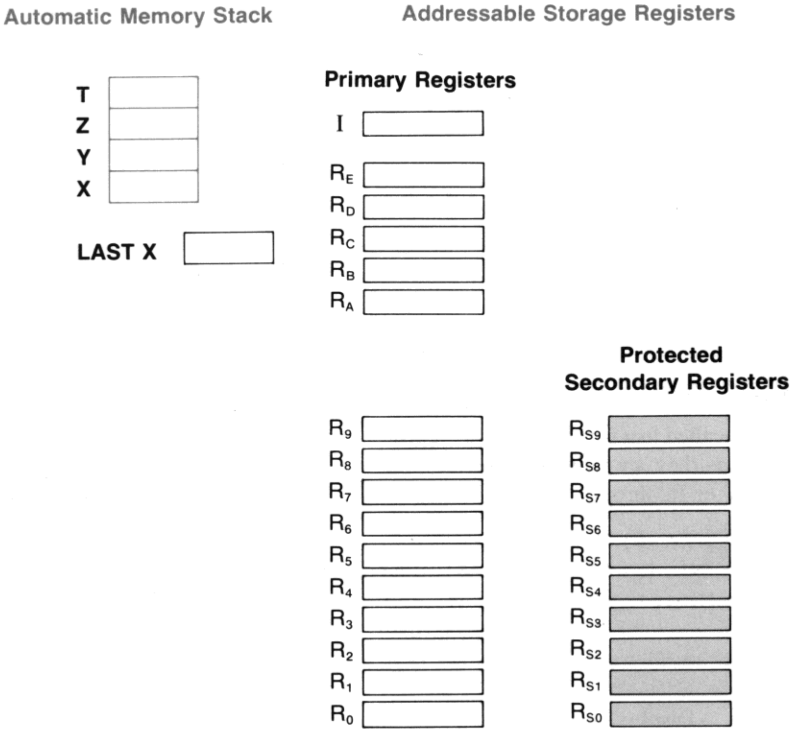
```
  3.785000000  STOI
               RCL2
6.020000000+23  ***
                P⇄S
  545.0000000  STO3
  5.000000000  ST×3
               RCL3
  2725.000000   ***
               RCLA
  1.558975689   ***
               PREG
  0.000000000    0
  0.000000000    1
6.020000000+23   2
  2725.000000    3
```

Section 4

Storing and Recalling Numbers

You have learned about the calculating power that exists in the four-register automatic memory stack and the LAST X register of your HP-97 calculator. In addition to the automatic storage of intermediate results that is provided by the stack, however, the HP-97 also contains 26 *addressable* data storage registers that are unaffected by operations within the stack. These registers allow you to manually store and recall constants or to set aside numbers for use in later calculations. Like all functions, you can use these storage registers either from the keyboard or as part of a program.

The diagram below shows the addressable storage registers. You can see that these registers consist of two banks, the *primary registers* and the *secondary registers*. The subscripts A through E and 0 through 9 refer to the register addresses.



Storing Numbers

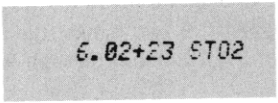
To store a displayed number in any of the primary storage registers:

1. Press **STO** *(store)*.
2. Press the letter key (**A** through **E**, **I**) or the number key (**0** through **9**) of the desired primary register address.

For example, to store Avogadro's number (approximately 6.02×10^{23}) in register R_2:

Slide the Print Mode switch to NORM MAN TRACE NORM if you want your printed tape to match the ones shown here.

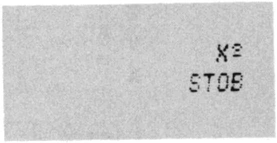
Press	Display
6.02 **EEX** 23	6.02 23
STO 2	6.020000000 23



```
6.02+23 ST02
```

Avogadro's number is now stored in register R_2. You can see that when a number is stored, it is merely copied into the storage register, so 6.02×10^{23} also remains in the displayed X-register. To store the square of Avogadro's number in register R_B:

Press	Display
x^2	3.624040000 47
STO **B**	3.624040000 47



```
 X²
STOB
```

The square of Avogadro's number has been copied into storage register R_B and also remains in the displayed X-register.

Recalling Numbers

Numbers are recalled from primary storage registers back into the displayed X-register in much the same way as they are stored. To recall a number from any of primary storage registers R_A through R_E or R_0 through R_9:

1. Press **RCL** *(recall)*.
2. Press the letter key (**A** through **E**) or the number key (**0** through **9**) of the desired primary storage register address.

For example, to recall Avogadro's number from register R_2:

Press	Display
RCL 2	6.020000000 23

```
RCL2
```

To recall the square of Avogadro's number from register R_B:

Press	Display
RCL B	3.624040000 47

```
RCLB
```

When you recall a number, it is copied from the storage register into the display, and it also remains in the storage register. You can recall a number from a storage register any number of times without altering it—the number will remain in the storage register as a 10-digit number with a two-digit exponent of 10 until you overwrite it by storing another number there, or until you clear the storage registers. For example, even though you earlier recalled Avogadro's number from storage register R_2, you can recall it again:

Press	Display
RCL 2	6.020000000 23

```
RCL2
```

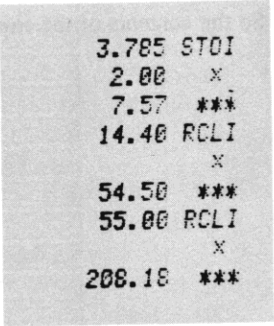
The I Register

The I register has a number of special properties that make it useful in programming, but these will be discussed later. When you are using the HP-97 manually, calculating from keyboard, the I-register is the most convenient storage register because you only need press I to recall its contents. You do not have to press RCL (although RCL I is a perfectly valid operation). To store a number in the I-register, you must press STO I.

Example: Three tanks have capacities in U.S. units of 2.0, 14.4, and 55.0 gallons, respectively. If 1 U.S. gallon is equivalent to 3.785 liters, what is the capacity in liters of each of the tanks?

Method: Place the conversion constant in one of the storage registers and bring it out as required.

Press	Display	
3.785 STO I	3.79	Constant placed in I-register.
2 ×	7.57	Capacity in liters of 1st tank.
PRINTx	7.57	
14.4 I ×	54.50	Capacity in liters of 2nd tank.
PRINTx	54.50	
55 I ×	208.18	Capacity in liters of 3rd tank.
PRINTx	208.18	



```
 3.785 STOI
  2.00    ×
  7.57  ***
 14.40 RCLI
          ×
 54.50  ***
 55.00 RCLI
          ×
208.18  ***
```

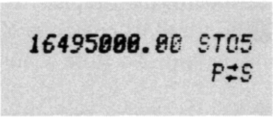
Protected Secondary Storage Registers

In addition to the primary storage registers, your HP-97 also provides you with 10 secondary storage registers that are protected; that is, you cannot access the secondary storage registers directly with **STO** and **RCL**. These registers are used most often by the statistical function **Σ+** (about which more later) and for programming purposes. However, they can be accessed manually from the keyboard by using the P⇄S key.

For example, in order to store a number from the displayed X-register into secondary storage register R_{S5}, you first store the number in primary register R_5 and then press **f** P⇄S *(primary exchange secondary)*. When you press P⇄S, the contents of the primary registers R_0 through R_9 are exchanged with the contents of secondary storage registers R_{S0} through R_{S9}. No other storage or stack registers are affected.

For example, to store 16,495,000 (the number of persons carried daily by the Japanese National Railway) in secondary storage register R_{S5}:

Press	Display	
16495000	16495000.	
STO 5	16495000.00	Number stored in register R_5.
f P⇄S	16495000.00	All secondary registers exchanged with numbered primary registers, so number is now stored in secondary storage register R_{S5}.



```
16495000.00 STO5
                P⇄S
```

With results from previous examples intact, when you pressed P⇄S in the above example, the contents of all *numbered* storage registers were exchanged.

So the contents of the storage registers changed…

... from this ...

Primary Registers

I	3.785
R_E	0.00
R_D	0.00
R_C	0.00
R_B	3.6240400000 47
R_A	0.00

Secondary Registers

R_9	0.00	⇄	R_{S9}	0.00
R_8	0.00	⇄	R_{S8}	0.00
R_7	0.00	⇄	R_{S7}	0.00
R_6	0.00	⇄	R_{S6}	0.00
R_5	16495000.00	⇄	R_{S5}	0.00
R_4	0.00	⇄	R_{S4}	0.00
R_3	0.00	⇄	R_{S3}	0.00
R_2	6.0200000000 23	⇄	R_{S2}	0.00
R_1	0.00	⇄	R_{S1}	0.00
R_0	0.00	⇄	R_{S0}	0.00

... to this.

Primary Registers

I	3.785
R_E	0.00
R_D	0.00
R_C	0.00
R_B	3.6240400000 47
R_A	0.00

Secondary Registers

R_9	0.00	R_{S9}	0.00
R_8	0.00	R_{S8}	0.00
R_7	0.00	R_{S7}	0.00
R_6	0.00	R_{S6}	0.00
R_5	0.00	R_{S5}	16495000.00
R_4	0.00	R_{S4}	0.00
R_3	0.00	R_{S3}	0.00
R_2	0.00	R_{S2}	6.0200000000 23
R_1	0.00	R_{S1}	0.00
R_0	0.00	R_{S0}	0.00

When you press P⇄S, the contents of *each* number-addressed primary storage register are exchanged with its opposite-numbered secondary storage register. Thus, in order to bring out the numbers that are now in the secondary storage registers, you must use the f P⇄S keys followed by the RCL key and the number key of the register address. For example, to recall the number of persons carried daily by the Japanese National Railway, you cannot merely press RCL 5 now, since the number in primary storage register R_5 is 0.00:

Press	**Display**
RCL 5	0.00

RCL5

However, you can press f P⇄S to bring the stored quantities back into the primary storage registers, then summon the desired quantities by pressing RCL followed by the number key of the desired address:

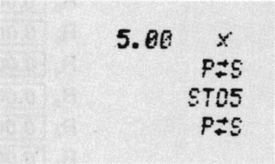
Press	**Display**	
f P⇄S	0.00	
RCL 5	16495000.00	Number of persons carried daily by the Japanese National Railway.

P⇄S
RCL5

When you press P⇄S, only the *contents* of the primary and secondary registers are exchanged. The actual registers remain intact and are not exchanged.

You can place numbers in corresponding primary and secondary registers and recall them at will. For example, to place the number of persons carried in *five* days by the Japanese National Railway into secondary register R_{S5} while leaving the number of persons carried *daily* intact in primary register R_5:

Press	**Display**
5 ×	82475000.00
f P⇄S	82475000.00
STO 5	82475000.00
f P⇄S	82475000.00



5.00 ×
P⇄S
STO5
P⇄S

You can now use RCL 5 to summon the number of persons carried daily, and f P⇄S followed by RCL 5 to summon the number of persons carried in five days:

Press	Display
RCL 5	16495000.00
f P⇄S	16495000.00
RCL 5	82475000.00

```
                RCL5
                P⇄S
                RCL5
```

Printing the Storage Registers

You can see the contents of all of the primary storage registers at any time with the PRINT: REG key. Simply press f PRINT: REG to print a listing of the contents of all the storage registers. For example, if you have worked through the examples as shown above, printing the contents of the storage registers should give you a listing like the one shown below.

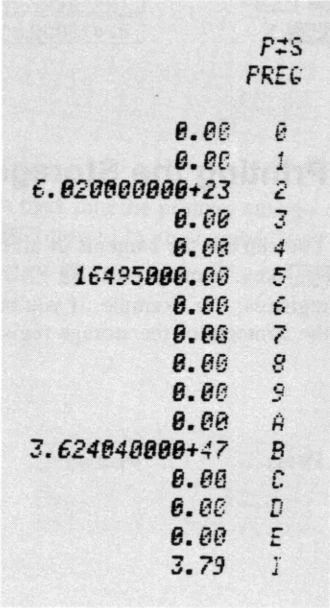
Press	Display
f PRINT: REG	82475000.00

```
                PREG

          0.00   0
          0.00   1
          0.00   2
          0.00   3
          0.00   4
   82475000.00   5
          0.00   6
          0.00   7
          0.00   8
          0.00   9
          0.00   A
3.624040000+47   B
          0.00   C
          0.00   D
          0.00   E
          3.79   I
```

If you want only a partial listing of the primary storage registers, you can stop the printing of them at any time by pressing R/S or any other key from the keyboard. The key function is *not* executed.

To see a listing of the contents of the secondary storage registers, simply press f P⇄S to bring those contents into the primary registers, then press f PRINT: REG to print all primary registers again. For example:

Press	Display
f P⇄S	82475000.00
f PRINT: REG	82475000.00



```
                P⇄S
               PREG

         0.00     0
         0.00     1
6.020000000+23    2
         0.00     3
         0.00     4
  16495000.00     5
         0.00     6
         0.00     7
         0.00     8
         0.00     9
         0.00     A
3.624040000+47    B
         0.00     C
         0.00     D
         0.00     E
         3.79     I
```

Naturally, if you want the present contents of primary registers R_0 through R_9 returned to the secondary storage registers, you must press f P⇄S again.

Clearing Storage Registers

Even though you have recalled the contents of a storage register into the displayed X-register, the number also remains in the storage register. You can clear primary storage registers in either of two ways:

- To replace a number in a single storage register, merely store another number there. To clear a storage register, replace the number in it with zero. For example, to clear storage register R_2, press 0 STO 2.
- To clear *all* primary storage registers back to zero at one time, press f CL REG. This clears all primary storage registers, while leaving the automatic memory stack and the secondary storage registers unchanged.

To clear the *secondary* storage registers, use the P⇄S key to bring their contents into the primary registers, then clear those registers in either of the methods described above.

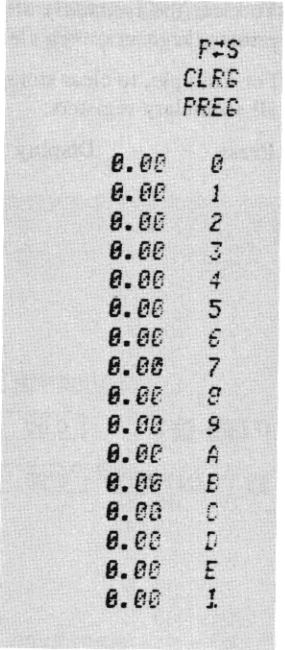
For example, to clear storage register R_B only, then to clear all primary registers, and finally all secondary registers:

Press	Display	
0 STO B	0.00	
f PRINT: REG	0.00	Storage register R_B is cleared to zero.
f CL REG	0.00	All primary storage registers cleared to zero. Secondary registers remain intact.
f PRINT: REG	0.00	

```
          0.00 STOB
               PREG

          0.00    0
          0.00    1
6.020000000+23    2
          0.00    3
          0.00    4
   16495000.00    5
          0.00    6
          0.00    7
          0.00    8
          0.00    9
          0.00    A
          0.00    B
          0.00    C
          0.00    D
          0.00    E
          3.79    I
```

```
               CLRG
               PREG

          0.00    0
          0.00    1
          0.00    2
          0.00    3
          0.00    4
          0.00    5
          0.00    6
          0.00    7
          0.00    8
          0.00    9
          0.00    A
          0.00    B
          0.00    C
          0.00    D
          0.00    E
          0.00    I
```

Press	Display	
f P⇄S	0.00	Contents of secondary registers exchanged with primary registers.
f CL REG	0.00	All storage registers have been cleared to zero.
f PRINT: REG	0.00	



```
         P⇄S
        CLRG
        PREG

   0.00    0
   0.00    1
   0.00    2
   0.00    3
   0.00    4
   0.00    5
   0.00    6
   0.00    7
   0.00    8
   0.00    9
   0.00    A
   0.00    B
   0.00    C
   0.00    D
   0.00    E
   0.00    I
```

Notice that the stack registers remain intact when you press f CL REG. To clear the displayed X-register, of course, you can press CL x. To clear the entire stack, press CL x ENTER↑ ENTER↑ ENTER↑. (Because of the automatic lift and drop of the stack, you should never have to clear it.) When the calculator is turned ON, it "wakes up" with the stack and *all* storage registers cleared to zero; so turning the calculator OFF, then ON clears the stack, the storage registers, and all program information. (This also should never be necessary.)

Storage Register Arithmetic

You can, of course, perform arithmetic (or any other function) in the normal manner by recalling and *using* the contents of any storage register just as if it were a number you keyed in. The HP-97 also permits you to perform storage register arithmetic in storage registers; that is, arithmetic *upon* the contents of the selected register.

Storage register arithmetic can be performed directly upon the contents of primary registers R_0 through R_9 only; it cannot be performed directly upon any other storage register. (Although storage register arithmetic *can* be performed indirectly upon the contents of *any* storage register, as you will see in section 12, Using the I-Register for Indirect Control.)

To perform storage register arithmetic directly, press STO followed by the arithmetic function key followed in turn by the number key (0 through 9) of the primary register address. For example:

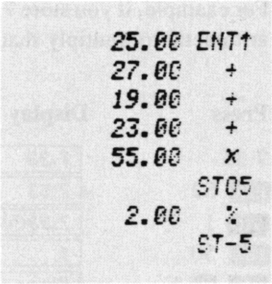
Press	Result
STO + 1	Number in displayed X-register added to contents of primary storage register R_1, and sum placed into R_1; $(r_1 + x \rightarrow R_1)$.
STO − 2	Number in displayed X-register subtracted from contents of primary storage register R_2, and difference placed into R_2; $(r_2 - x \rightarrow R_2)$.
STO × 3	Number in displayed X-register multiplied by contents of primary storage register R_3, and the product placed into R_3; $\left[(r_3)\ x \rightarrow R_3\right]$.
STO ÷ 4	Contents of storage register R_4 divided by number in displayed X-register, and quotient placed into register R_4; $(r_4 \div x \rightarrow R_4)$.

When storage register arithmetic operations are performed, the answer is written into the selected storage register, while the contents of the other storage registers and the displayed X-register and the rest of the stack remain unchanged.

Example: During harvest, farmer Flem Snopes trucks tomatoes to the cannery for three days. On Monday and Tuesday he hauls loads of 25 tons, 27 tons, 19 tons, and 23 tons, for which the cannery pays him $55 per ton. On Wednesday the price rises to $57.50 per ton, and Snopes ships loads of 26 tons and 28 tons. If the cannery deducts 2% of the price on Monday and Tuesday because of blight on the tomatoes, and 3% of the price on Wednesday, what is the Snopes' total net income?

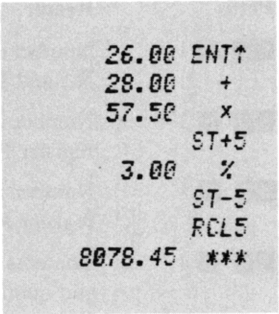
Method: Keep total amount in a storage register while using the stack to add tonnages and calculate amounts of loss.

Press	Display	
25 ENTER↑ 27 +		
19 + 23 +	94.00	Total of Monday's and Tuesday's tonnage.
55 ×	5170.00	Gross amount for Monday and Tuesday.
STO 5	5170.00	Gross placed in storage register R_5.
2 %	103.40	Deductions for Monday and Tuesday.
STO − 5	103.40	Deductions subtracted from total in storage register R_5.



```
25.00 ENT↑
27.00    +
19.00    +
23.00    +
55.00    x
      STO5
 2.00    %
      ST-5
```

26 ENTER↑ 28 +	54.00	Wednesday's tonnage.
57.50 ×	3105.00	Gross amount for Wednesday.
STO + 5	3105.00	Wednesday's gross amount added to total in storage register R_5.
3 %	93.15	Deduction for Wednesday.
STO − 5	93.15	Wednesday deduction subtracted from total in storage register R_5.
RCL 5	8078.45	Snopes' total net income from his tomatoes.
PRINT x	8078.45	



```
  26.00 ENT↑
  28.00    +
  57.50    ×
        ST+5
   3.00    %
        ST-5
        RCL5
8078.45  ***
```

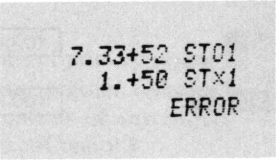
(You could also work this problem using the stack alone, but doing it as shown here illustrates how storage register arithmetic can be used to maintain and update different running totals.)

Storage Register Overflow

If you attempt a storage register arithmetic operation that would cause the magnitude of a number in any of the storage registers to exceed 9.999999999 × 10^{99}, the operation is not performed and the HP-97 display immediately indicates Error . In addition, if the Print Mode switch **MAN** TRACE **NORM** is set to NORM or TRACE, the printer also registers the error. When you then press any key, the error condition is cleared and the last value in the X-register before the error is again displayed. The storage registers all contain the values they held before the error-causing operation was attempted.

For example, if you store 7.33 × 10^{52} in primary register R_1 and attempt to use storage register arithmetic to multiply that value by 10^{50}, the HP-97 display will show Error :

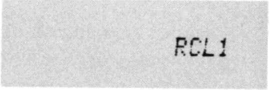
Press	Display
7.33	7.33
EEX 52	7.33 52
STO 1	7.330000000 52
EEX 50	1. 50
STO × 1	Error



```
7.33+52 STO1
  1.+50 ST×1
       ERROR
```

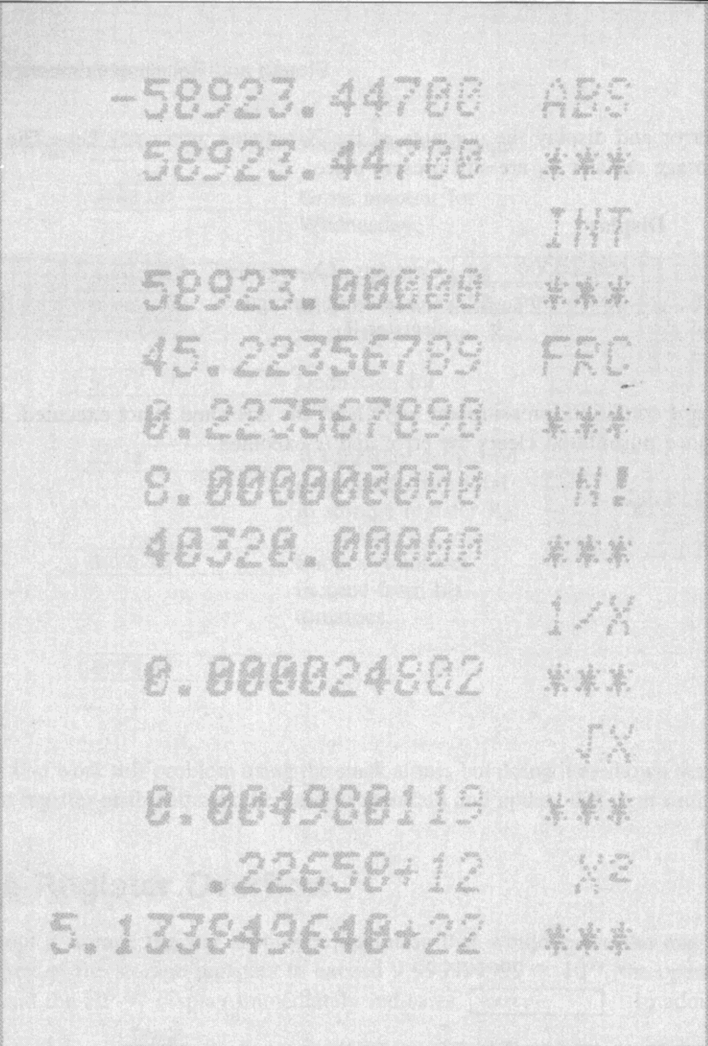
To clear the error and display the contents of the X-register, press any key. The original contents of storage register R_1 are still present there.

Press	Display	
CL x	1.000000000 50	Contents of X-register.
RCL 1	7.330000000 52	Contents of storage register R_1.



```
RCL1
```

As with any error condition, pressing any key clears the error and is not executed. Pressing the paper advance pushbutton clears the error and *is* executed.



```
-58923.44700  ABS
 58923.44700  ***
              INT
 58923.00000  ***
 45.22356789  FRC
 0.223567890  ***
 8.000000000   N!
 40320.00000  ***
              1/X
 0.000024802  ***
               √X
 0.004980119  ***
   .22650+12   X²
5.133849640+22 ***
```

Section 5

Function Keys

The HP-97 has dozens of internal functions that allow you to compute answers to problems quickly and accurately. Each function operates the same way, regardless of whether you press the function key manually or the function is executed as part of a program.

In this section, each function key is explained as it is used manually, with the Program Mode switch set to RUN. To save printing time and paper, you might wish to learn how to use the functions with the Print Mode switch set to MAN. Or you might wish to see every intermediate and final answer by setting the switch to TRACE. Except where indicated, however, all examples in this section are illustrated with the Print Mode switch set to NORM.

If you want your displays and printed copy to match the ones shown here, then:

Set the Print Mode switch MAN TRACE NORM to NORM.

Set the PRGM-RUN switch PRGM RUN to RUN.

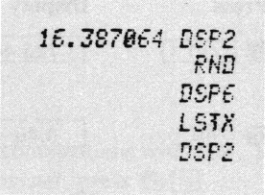
Number Alteration Keys

Besides CHS, there are four keys provided for altering numbers in the HP-97. These keys are RND, ABS, INT, and FRAC, and you will find them most useful when performing operations as part of a program.

Rounding a Number

As you know, when you change display formats with one of the display control keys (FIX, SCI, ENG, or DSP), the number maintains its full value to 10 digits multiplied by a two-digit exponent of ten no matter how many digits you see. When you press the f prefix key followed by the RND *(round)* key, however, the number that is in the display becomes the *actual* number in the calculator. For example, key in the number of cubic centimeters in one cubic inch, 16.387064, and round it to two decimal places:

Press	**Display**	
16.387064	16.387064	
DSP 2	16.39	Number rounded to two decimal places in display. Maintains entire value internally.
f RND	16.39	Number rounded to two decimal places internally.
DSP 6	16.390000	FIX 6 display shows that the number has been rounded.
f LAST X	16.387064	The original number.
DSP 2	16.39	Display mode reset.



```
16.387064 DSP2
           RND
          DSP6
          LSTX
          DSP2
```

RND rounds to 0.00 a number that has underflowed to scientific notation.

Absolute Value

Some calculations require the absolute value, or magnitude, of a number. To obtain the absolute value of the number in the displayed X-register, press the f shift key followed by the ABS (*absolute value*) key. For example, to calculate the absolute value of −3:

Press	Display	
3 CHS	-3.	
f ABS	3.00	$\lvert -3 \rvert$

```
-3.00    ABS
```

To see the absolute value of +3:

Press	Display	
f ABS	3.00	$\lvert +3 \rvert$

```
         ABS
```

Integer Portion of a Number

To extract and display the integer portion of a number, press the f prefix key followed by the INT (*integer*) key. For example, to display only the integers of the number 123.456:

Press	Display	
123.456	123.456	
f INT	123.00	Only the integer portion of the number remains.

```
123.456    INT
```

When f INT is pressed, the fractional portion of the number is lost. The entire number, of course, is preserved in the LAST X register.

Fractional Portion of a Number

To extract and display only the fractional portion of a number, press the f prefix key followed by the FRAC *(fraction)* key. For example, to see the fractional portion of the 123.456 used above:

Press	Display	
f LAST X	123.46	Summons the original number back to the X-register.
f FRAC	0.46	Only the fractional portion of the number is displayed, rounded here to FIX 2 display.

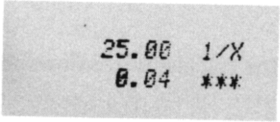
```
          LSTX
          FRC
```

When f FRAC is pressed, the integer portion of the number is lost. The entire number, of course, is preserved in the LAST X register.

Reciprocals

To calculate the reciprocal of a number in the displayed X-register, key in the number, then press ⅟x. For example, to calculate the reciprocal of 25:

Press	Display
25 ⅟x	0.04
PRINTx	0.04



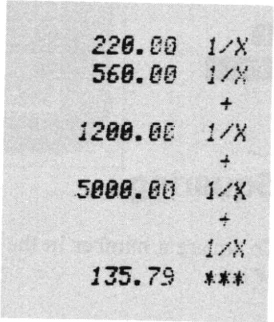
```
  25.00  1/X
   0.04  ***
```

You can also calculate the reciprocal of a value in a previous calculation without reentering the number.

Example: In an electrical circuit, four resistors are connected in parallel. Their values are 220 ohms, 560 ohms, 1.2 kilohms, and 5 kilohms. What is the total resistance of the circuit?

$$R_T = \frac{1}{\frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} + \frac{1}{R_4}} = \frac{1}{\frac{1}{220} + \frac{1}{560} + \frac{1}{1200} + \frac{1}{5000}}$$

Press	Display	
220 ⅟x	4.545454545-03	
560 ⅟x	1.785714286-03	
+	0.01	
1200 ⅟x	8.333333333-04	
+	0.01	
5000 ⅟x	2.000000000-04	
+	0.01	Sum of reciprocals.
⅟x	135.79	The reciprocal of the sum of the reciprocals yields the answer in ohms.
PRINTx	135.79	



```
  220.00  1/X
  560.00  1/X
            +
 1200.00  1/X
            +
 5000.00  1/X
            +
          1/X
  135.79  ***
```

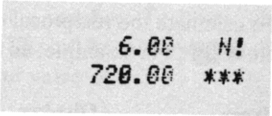
Factorials

The N! *(factorial)* key permits you to handle permutations and combinations with ease. To calculate the factorial of a positive integer in the displayed X-register, press f N!.

Example: Calculate the number of ways that six people can line up for a photograph.

Method: $P_6^6 = 6! = 6 \times 5 \times 4 \times 3 \times 2 \times 1$.

Press	Display	
6	6.	
f N!	720.00	The answer.
PRINT x	720.00	



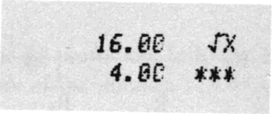
```
    6.00   N!
  720.00  ***
```

The calculator overflows for factorials of numbers greater than 69.

Square Roots

To calculate the square root of a number in the displayed X-register, press √x. For example, to find the square root of 16:

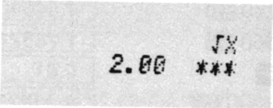
Press	Display
16 √x	4.00
PRINT x	4.00



```
   16.00   √X
    4.00  ***
```

To find the square root of the result:

Press	Display
√x	2.00
PRINT x	2.00

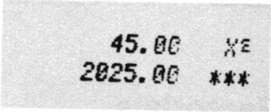


```
           √X
    2.00  ***
```

Squaring

To square a number in the displayed X-register, press x^2. For example, to find the square of 45:

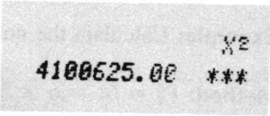
Press	Display
45 x^2	2025.00
PRINT x	2025.00



```
   45.00   X²
 2025.00  ***
```

To find the square of the result:

Press	Display
x^2	4100625.00
PRINT x	4100625.00

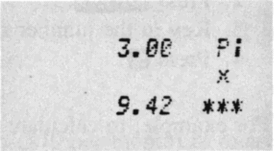


```
           X²
4100625.00 ***
```

Using Pi

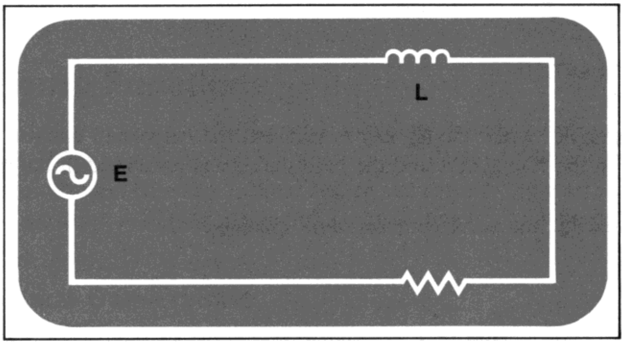
The value π accurate to 10 places (3.141592654) is provided as a fixed constant in the HP-97. Merely press **f** **π** whenever you need it in a calculation. For example, to calculate 3π:

Press	Display
3 **f** **π** **×**	9.42
PRINT x	9.42



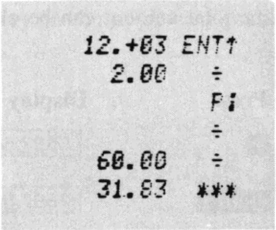
```
3.00   Pi
        ×
9.42  ***
```

Example: In the schematic diagram below, X_L is 12 kilohms, E is 120 volts, and f is 60 Hz. Find the inductance of the coil L in henries according to the formula: $L = \frac{X_L}{2\pi f}$.



$$L = \frac{X_L}{2\pi f} = \frac{12{,}000}{2 \times \pi \times 60}$$

Press	Display	
12 **EEX** 3	12. 03	
ENTER↑ 2 **÷**	6000.00	
f **π** **÷**	1909.86	
60 **÷**	31.83	Henries.
PRINT x	31.83	



```
12.+03 ENT↑
  2.00    ÷
         Pi
          ÷
 60.00    ÷
 31.83  ***
```

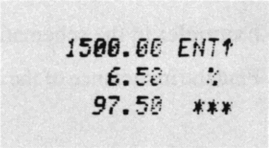
Percentages

The **%** key is a two-number function which allows you to compute percentages. To find the percentage of a number:

1. Key in the base number.
2. Press **ENTER↑**.
3. Key in the number representing percent rate.
4. Press **%**.

For example, to calculate a sales tax of 6.5% on a purchase of $1500:

Press	Display	
1500 **ENTER↑**	1500.00	Base number.
6.5	6.5	Percent rate.
%	97.50	The answer.
PRINTx	97.50	



1500.00 ENT↑
6.50 %
97.50 ***

6.5% of $1500 is $97.50.

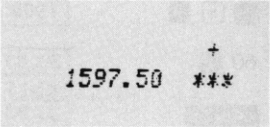
In the above example, when the **%** key is pressed, the calculated answer writes over the percentage rate in the X-register, and the base number is preserved in the Y-register.

When you pressed **%**, the stack contents were changed...

	... from this ...	**... to this.**
T	0.00	0.00
Z	0.00	0.00
Y	1500.00	1500.00
X	6.5	97.50

Since the purchase price is now in the Y-register and the amount of tax is in the X-register, the total amount can be obtained by simply adding:

Press	Display	
+	1597.50	Total of price and sales tax combined.
PRINTx	1597.50	



+
1597.50 ***

Percent of Change

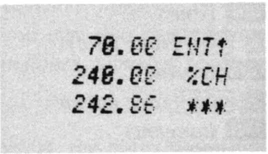
The %CH (*percent of change*) key is a two-number function that gives the percent increase or decrease from y to x. To find the percent of change:

1. Key in the base number (usually, the number that happens first in time).
2. Press **ENTER↑**.
3. Key in the second number.
4. Press **f** %CH.

The formula used is: $\frac{(x - y)\ 100}{y} = \%CH.$

Example: Find the percent of increase of your rent 10 years ago ($70 per month) to today ($240 per month).

Press	Display	
70 **ENTER↑**	70.00	
240 **f** %CH	242.86	Percent increase.
PRINTx	242.86	



```
 70.00 ENT↑
240.00 %CH
242.86 ***
```

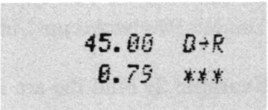
Trigonometric Functions

Your HP-97 provides you with six trigonometric functions, which operate in decimal degrees, radians, or grads. You can easily convert angles from decimal degrees to radians or vice versa, and you can convert between decimal degree and *degrees, minutes, seconds*. You can also add angles specified in *degrees, minutes, seconds* directly, without converting them to decimal.

Degrees/Radians Conversions

The D→R and R→D keys are used to convert angles between degrees and radians. To convert an angle specified in degrees to radians, key in the angle and press **f** D→R. For example, to change 45° to radians:

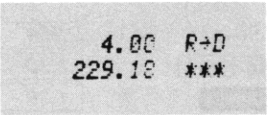
Press	Display	
45	45.	
f D→R	0.79	Radians.
PRINTx	0.79	



```
45.00 D→R
 0.79 ***
```

To convert an angle specified in radians to decimal degrees, key in the angle and press **f** R→D. For example, to convert 4 radians to decimal degrees:

Press	Display	
4	4.	
f R→D	229.18	Decimal degrees.
PRINTx	229.18	



```
  4.00 R→D
229.18 ***
```

Trigonometric Modes

For trigonometric functions, angles can be assumed by the calculator to be in decimal degrees, radians, or grads. When the HP-97 is first turned ON, it "wakes up" with angles assumed to be in decimal degrees. To select radians mode, press **f** RAD *(radians)* before using a trigonometric function. To select grads mode, press **f** GRD *(grads)*. To select decimal degrees again, press **f** DEG *(degrees)*.

Note: 360 degrees = 400 grads = 2π radians.

Functions

The six trigonometric functions provided by the calculator are:

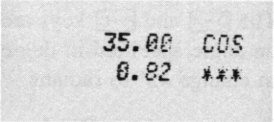
SIN *(sine)*
f SIN^{-1} *(arc sine)*
COS *(cosine)*
f COS^{-1} *(arc cosine)*
TAN *(tangent)*
f TAN^{-1} *(arc tangent)*

Each trigonometric function assumes that angles are in decimal degrees, radians, or grads, depending upon the trigonometric mode selected.

All trigonometric functions are one-number functions, so to use them, you key in the number, then press the function key(s).

Example 1: Find the cosine of 35°.

Press	Display
35	35.
COS	0.82
PRINTx	0.82

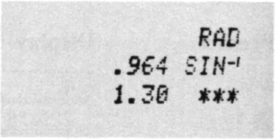


```
35.00  COS
 0.82  ***
```

The HP-97 "woke up" in degrees mode when you first turned it ON.

Example 2: Find the arc sine in radians of .964.

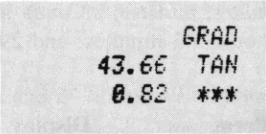
Press	Display	
f RAD	0.82	Selects radians mode. (Results remain from previous example.)
.964	.964	
f SIN^{-1}	1.30	Radians.
PRINTx	1.30	



```
           RAD
 .964  SIN-1
 1.30  ***
```

Example 3: Find the tangent of 43.66 grads.

Press	Display	
f GRD	1.30	Selects grads mode. (Results remain from previous example.)
43.66	43.66	
TAN	0.82	Grads.
PRINT x	0.82	

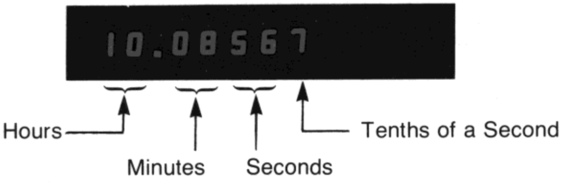


```
       GRAD
43.66  TAN
 0.82  ***
```

Hours, Minutes, Seconds/Decimal Hours Conversions

Using the HP-97, you can change time specified in decimal hours to *hours, minutes, seconds* format by using the →H.MS (*to hours, minutes, seconds*) key; you can also change from *hours, minutes, seconds* to decimal hours by using the H.MS→ *(from hours, minutes, seconds)* key.

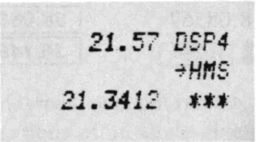
When a time is displayed or printed in hours, minutes, seconds format, the digits specifying *hours* occur to the left of the decimal point, while the digits specifying *minutes, seconds,* and *fractions of seconds* occur to the right of the decimal point.



Hours, Minutes, Seconds Display

To convert from decimal hours to *hours, minutes, seconds,* simply key in the value for decimal hours and press f →H.MS. For example, to change 21.57 hours to *hours, minutes, seconds:*

Press	Display	
21.57	21.57	Key in the decimal time.
DSP 4	21.5700	Reset display format to FIX 4.
f →H.MS	21.3412	This is 21 hours, 34 minutes, 12 seconds.
PRINT x	21.3412	

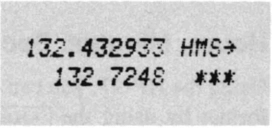


```
21.57    DSP4
         →HMS
21.3412   ***
```

Notice that the display is not automatically switched to show you more than the normal two digits after the decimal point (FIX 2), so to see the digits for *seconds,* you had to reset the display format to FIX 4.

To convert from *hours, minutes, seconds* to decimal hours, simply key in the value for *hours, minutes, seconds* in that format and press f H.MS→. For example, to convert 132 hours, 43 minutes, and 29.33 seconds to its decimal degree equivalent:

Press	**Display**	
132.432933	132.432933	This is 132 hours, 43 minutes, 29.33 seconds.
f H.MS→	132.7248	This is 132.7248 hours. (FIX 4 display remains specified from previous example.)
PRINT x	132.7248	

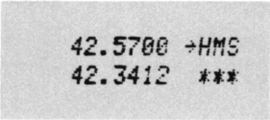


```
132.432933 HMS→
  132.7248  ***
```

Using the →H.MS and H.MS→ operations, you can also convert angles specified in decimal degrees to *degrees, minutes, seconds,* and vice versa. The format for *degrees, minutes, seconds* is the same as for *hours, minutes, seconds*.

Example: Convert 42.57 decimal degrees to *degrees, minutes, seconds*.

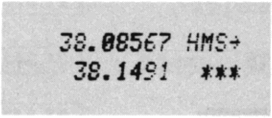
Press	**Display**	
42.57	42.57	Key in the angle.
f →H.MS	42.3412	This means 42°34′12″. (Display assumes FIX 4 notation remains specified from previous example.)
PRINT x	42.3412	



```
42.5700 →HMS
42.3412  ***
```

Example: Convert 38°8′56.7″ to its decimal equivalent.

Press	**Display**	
38.08567	38.08567	Key in the angle.
f H.MS→	38.1491	Answer in decimal degrees. (FIX 4 display specified from previous examples.)
PRINT x	38.1491	



```
38.08567 HMS→
 38.1491  ***
```

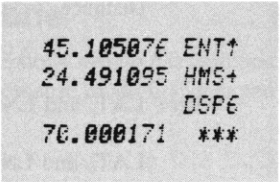
Adding and Subtracting Time and Angles

To add or subtract decimal hours, merely key in the numbers for the decimal hours and press + or −. To add *hours, minutes, seconds,* use the H.MS+ *(add hours, minutes, seconds)* key.

Likewise, angles specified in *degrees, minutes, seconds* are added by pressing f H.MS+.

Example: Find the sum of 45 hours, 10 minutes, 50.76 seconds and 24 hours, 49 minutes, 10.95 seconds.

Press	Display	
45.105076	45.105076	
ENTER↑	45.1051	FIX 4 notation from previous example.
24.491095	24.491095	
f H.MS+	70.0002	
DSP 6	70.000171	
PRINT x	70.000171	

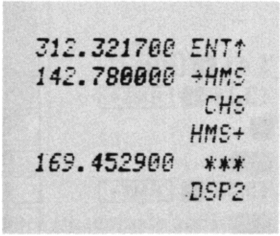


```
45.105076 ENT↑
24.491095 HMS+
               DSP6
70.000171  ***
```

To subtract a time specified in *hours, minutes, seconds* from another (or to subtract an angle specified in *degrees, minutes, seconds*), simply use the CHS key to make the second time (or angle) negative, then add with the H.MS+ key.

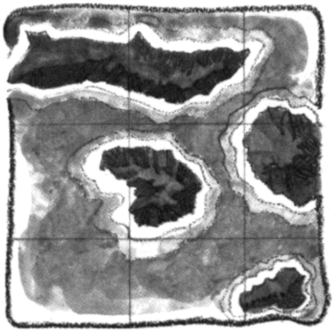
Example: Subtract 142.78° from 312°32′17″, with the answer in *degrees, minutes, seconds* format.

Press	Display	
312.3217	312.3217	
ENTER↑	312.321700	FIX 6 from previous example.
142.78	142.78	Decimal degrees.
f →H.MS	142.464800	To degrees, minutes, seconds.
CHS	−142.464800	Angle made negative.
f H.MS+	169.452900	This is 169°45′29″.
PRINT x	169.452900	
DSP 2	169.45	Display mode reset to FIX 2.



```
312.321700 ENT↑
142.780000 →HMS
                CHS
               HMS+
169.452900  ***
               DSP2
```

In the HP-97, trigonometric functions assume angles in decimal degrees, decimal radians, or decimal grads, so if you want to compute any trigonometric functions of an angle given in *degrees, minutes, and seconds,* you must first convert the angle to decimal degrees.



Example: Lovesick sailor Oscar Odysseus dwells on the island of Tristan da Cunha (37°03′S, 12°18′W), and his sweetheart, Penelope, lives on the nearest island. Unfortunately for the course of true love, however, Tristan da Cunha is the most isolated inhabited spot in the world. If Penelope lives on the island of St. Helena (15°55′S, 5°43′W), use the following formula to calculate the great circle distance that Odysseus must sail in order to court her.

$$\text{Distance} = \cos^{-1}\left[\sin(\text{LAT}_s)\sin(\text{LAT}_d) + \cos(\text{LAT}_s)\cos(\text{LAT}_d)\cos(\text{LNG}_d - \text{LNG}_s)\right] \times 60.$$

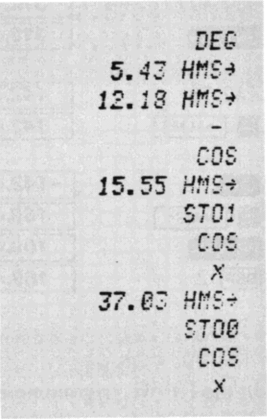
Where LAT_s and LNG_s = latitude and longitude of the source (Tristan da Cunha).

LAT_d and LNG_d = latitude and longitude of the destination.

Solution: Convert all *degrees, minutes, seconds* entries into decimal degrees as you key them in. The equation for the great circle distance from Tristan da Cunha to the nearest inhabited land is:

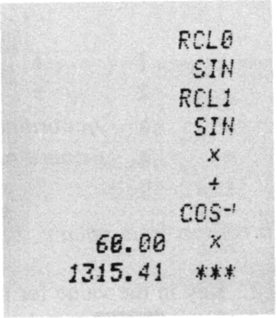
$$\text{Distance} = \cos^{-1}\left[\sin(37°03')\sin(15°55') + \cos(37°03')\cos(15°55')\cos(5°43'\text{ W} - 12°18'\text{ W})\right] \times 60$$

Press	Display	
f DEG	0.00	Selects degrees mode. (Display assumes no results remain from previous examples.)
5.43 f H.MS→	5.72	
12.18 f H.MS→	12.30	
−	-6.58	
COS	0.99	
15.55 f H.MS→	15.92	
STO 1	15.92	
COS	0.96	
×	0.96	
37.03 f H.MS→	37.05	
STO 0	37.05	
COS	0.80	
×	0.76	



```
              DEG
      5.43 HMS→
     12.18 HMS→
                -
              COS
     15.55 HMS→
             STO1
              COS
                ×
     37.03 HMS→
             STO0
              COS
                ×
```

Press	Display	
RCL 0 SIN	0.60	
RCL 1 SIN	0.27	
×	0.17	
+	0.93	
f COS^{-1}	21.92	
60 × PRINT x	1315.41	Distance in nautical miles that Odysseus must sail to visit Penelope.

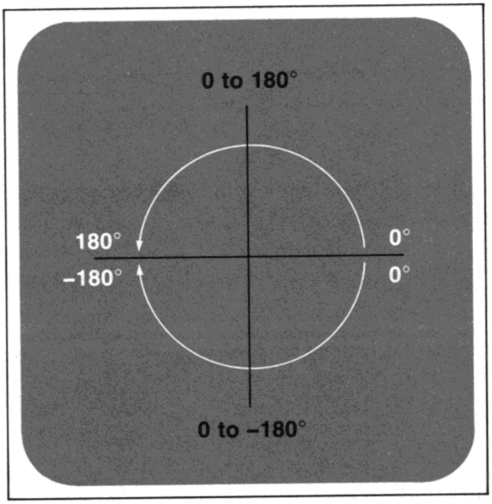


```
           RCL0
            SIN
           RCL1
            SIN
              x
              +
          COS-1
  60.00       x
1315.41     ***
```

Polar/Rectangular Coordinate Conversions

Two functions are provided for polar/rectangular coordinate conversions. Angle θ is assumed in decimal degrees, radians, or grads, depending upon the trigonometric mode first selected by DEG, RAD, or GRD.

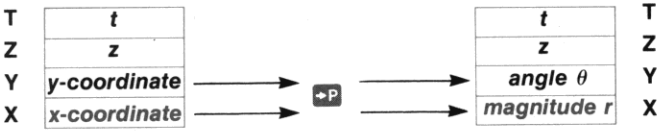
In the HP-97, angle θ is represented in the following manner:



To convert from rectangular x, y coordinates to polar r, θ coordinates (magnitude and angle, respectively):

1. Key in the y-coordinate.
2. Press ENTER↑ to raise the y-coordinate value to the Y-register of the stack.
3. Key in the x-coordinate.
4. Press the →P (*to polar*) key. Magnitude r then appears in the X-register and angle θ is placed in the Y-register. (To display the value for θ, you can press x⇄y.)

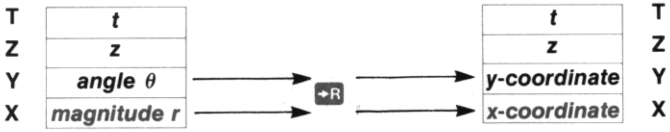
The following diagram shows how the stack contents change when you press →P.



To convert from polar *r*, *θ*, coordinates to rectangular x, y, coordinates:

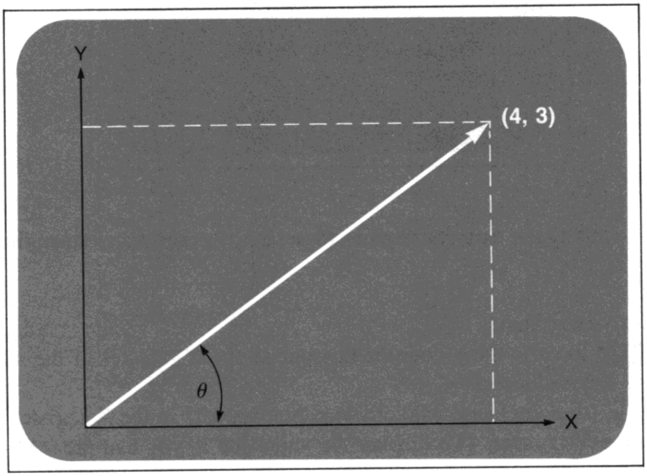
1. Key in the value for the angle *θ*.
2. Press ENTER↑ to raise the value for *θ* to the Y-register of the stack.
3. Key in the value for magnitude *r*.
4. Press the →R (*to rectangular*) key. The x-coordinate then appears in the displayed X-register and the y-coordinate is placed in the Y-register. (To display the value for the y-coordinate, you can press x⇄y.)

The following diagram shows how the stack contents change when you press →R.



After you have pressed →P or →R, you can use the x⇄y key to bring the calculated angle *θ* or the calculated y-coordinate into the X-register for viewing or further calculation.

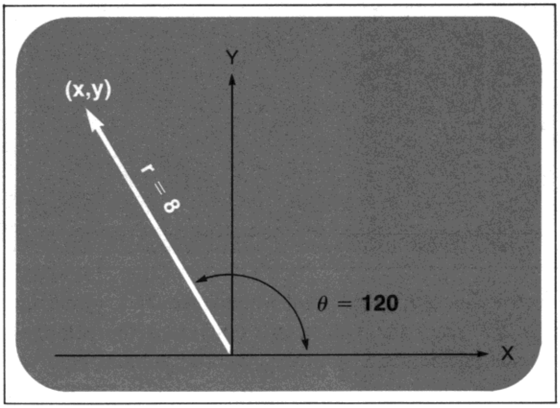
Example 1: Convert rectangular coordinates (4, 3) to polar form with the angle expressed in radians.



Press	Display	
f RAD	0.00	Radians mode selected. (Display assumes no results remain from previous examples.)
3 ENTER↑	3.00	y-coordinate entered into the Y-register.
4	4.	x-coordinate keyed into the X-register.
→P	5.00	Magnitude *r*.
PRINTx	5.00	
x⇄y	0.64	Angle *θ* in radians.
PRINTx	0.64	

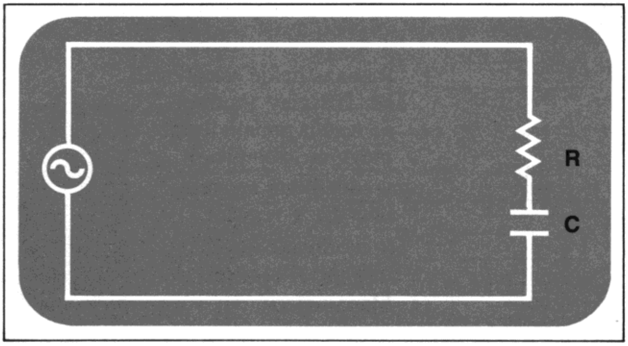
```
          RAD
3.00  ENT↑
4.00    →P
5.00   ***
        X⇄Y
0.64   ***
```

Example 2: Convert polar coordinates (8, 120 grads) to rectangular coordinates.

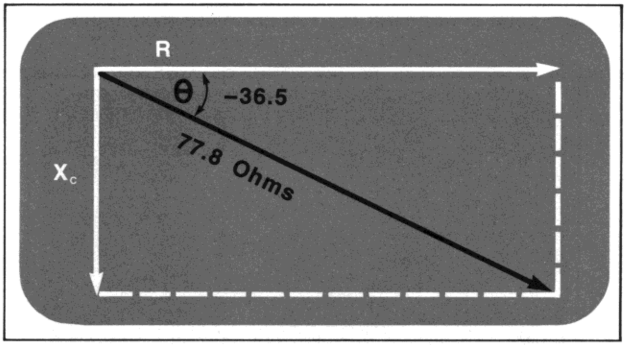


Press	Display	
f GRD	0.64	Grads mode selected. (Note that results can remain from previous examples.)
120 ENTER↑	120.00	Angle *θ* entered into the Y-register.
8	8.	Magnitude *r* placed in displayed X-register.
→R	-2.47	x-coordinate.
x⇄y	7.61	y-coordinate brought into displayed X-register.

```
            GRAD
120.00  ENT↑
  8.00    →R
          X⇄Y
```



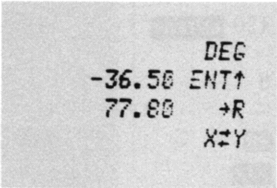
Example 3: Engineer Tobias Slothrop has determined that in the RC circuit shown above, the total impedance is 77.8 ohms and voltage lags current by 36.5°. What are the values of resistance R and capacitive reactance X_c in the circuit?



Method: Draw a vector diagram using total impedance 77.8 ohms for polar magnitude *r* and −36.5° for angle *θ*. When the values are converted to rectangular coordinates, the x-coordinate value yields resistance R in ohms, and the y-coordinate value yields reactance X_c in ohms.

Solution:

Press	Display	
f DEG	7.61	Degrees mode selected. (Note that results can remain from previous examples.)
36.5 CHS	-36.5	
ENTER↑	-36.50	
77.8	77.8	
→R	62.54	Resistance R in ohms.
x⇄y	-46.28	Reactance X_c, 46.28 ohms, available in displayed X-register.



```
          DEG
-36.50 ENT↑
 77.80   →R
         X⇄Y
```

Logarithmic and Exponential Functions

Logarithms

The HP-97 computes both natural and common logarithms as well as their inverse functions (antilogarithms):

LN is $\log_e$ (natural log). It takes the log of the value in the X-register to base *e* (2.718...).

e^x is antilog_e (natural antilog). It raises *e* (2.718...) to the power of the value in the X-register. (To display the value of *e*, press 1 e^x.)

LOG is $\log_{10}$ (common log). It computes the log of the value in the X-register to base 10.

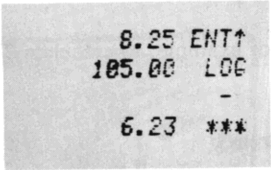
10^x is antilog_{10} (common antilog). It raises 10 to the power of the value in the X-register.

Example 1: The 1906 San Francisco earthquake, with a magnitude of 8.25 on the Richter Scale is estimated to be 105 times greater than the Nicaragua quake of 1972. What would be the magnitude of the latter on the Richter Scale? The equation is:

$$R_1 = R_2 - \log\frac{M_2}{M_1} = 8.25 - \left(\log\frac{105}{1}\right)$$

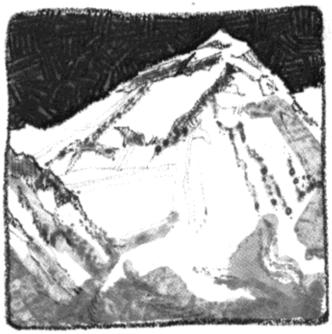
Solution:

Press	Display	
8.25 ENTER↑	8.25	
105 f LOG	2.02	
−	6.23	Rating on Richter scale.
PRINT x	6.23	



```
    8.25 ENT↑
  105.00  LOG
            -

    6.23  ***
```

Example 2: Having lost most of his equipment in a blinding snowstorm, ace explorer Jason Quarmorte is using an ordinary barometer as an altimeter. After measuring the sea level pressure (30 inches of mercury) he climbs until the barometer indicates 9.4 inches of mercury. Although the exact relationship of pressure and altitude is a function of many factors, Quarmorte knows that an *approximation* is given by the formula:

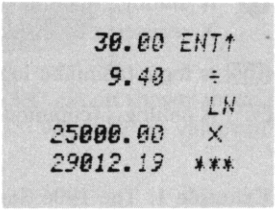


$$\text{Altitude (feet)} = 25{,}000\, ln \frac{30}{\text{Pressure}} = 25{,}000\, ln\, \frac{30}{9.4}$$

Where is Jason Quarmorte?

Solution:

Press	Display	
30 ENTER↑	30.00	
9.4 ÷	3.19	
LN	1.16	
25000	25000.	
×	29012.19	Altitude in feet.
PRINT x	29012.19	



```
   30.00 ENT↑
    9.40    ÷
           LN
25000.00    ×
29012.19  ***
```

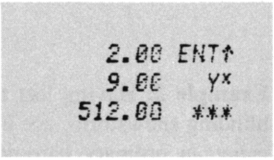
Quarmorte is probably near the summit of Mount Everest (29,028 feet).

Raising Numbers to Powers

The y^x key is used to raise numbers to powers. Using y^x permits you to raise a positive real number to any real power—that is, the power may be positive or negative, and it may be an integer, a fraction, or a mixed number. y^x also permits you to raise any negative real number to the power of any integer (within the calculating range of the HP-97, of course).

For example, to calculate 2^9 (that is, 2 × 2 × 2 × 2 × 2 × 2 × 2 × 2 × 2):

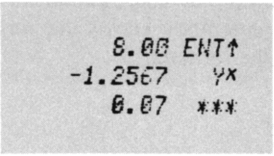
Press	Display
2 ENTER↑ 9	9.
y^x	512.00
PRINT x	512.00



```
  2.00 ENT↑
  9.00   Yx
512.00  ***
```

To calculate $8^{-1.2567}$:

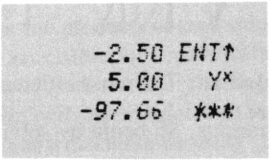
Press	Display
8 ENTER↑	8.00
1.2567 CHS y^x	0.07
PRINT x	0.07



```
   8.00 ENT↑
-1.2567   Yx
   0.07  ***
```

To calculate $(-2.5)^5$:

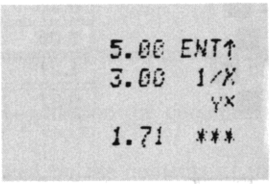
Press	Display
2.5 CHS ENTER↑	-2.50
5 y^x	-97.66
PRINT x	-97.66



```
   -2.50 ENT↑
    5.00   Yx
  -97.66  ***
```

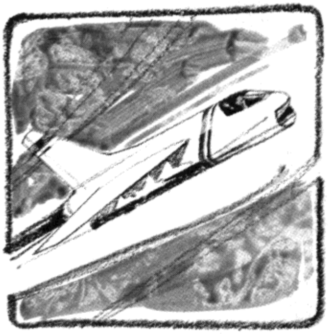
In conjunction with $1/x$ y^x provides a simple way to extract roots. For example, find the cube root of 5. (This is equivalent to $5^{1/3}$.)

Press	Display	
5 ENTER↑	5.00	
3 $1/x$	0.33	Reciprocal of 3.
y^x	1.71	Cube root of 5.
PRINT x	1.71	



```
    5.00 ENT↑
    3.00  1/X
           Yx
    1.71  ***
```

Example: In a rather overoptimistic effort to break the speed of sound, high-flying pilot Ike Daedalus cranks open the throttle on his surplus Hawker Siddeley Harrier aircraft. From his instruments he reads a pressure altitude (PALT) of 25,500 feet with a calibrated airspeed (CAS) of 350 knots. What is the flight mach number



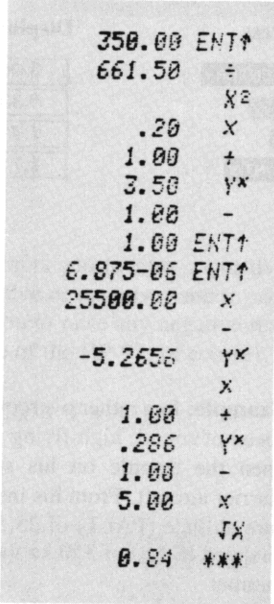
$$M = \frac{\text{speed of aircraft}}{\text{speed of sound}}$$

if the following formula is applicable?

$$M = \sqrt{5\left[\left(\left\{\left[\left(1+0.2\left[\frac{350}{661.5}\right]^{2}\right)^{3.5}-1\right]\left[1-(6.875\times10^{-6})\,25{,}500\right]^{-5.2656}\right\}+1\right)^{0.286}-1\right]}$$

Method: The most efficient place to begin work on this problem is at the innermost set of brackets. So begin by solving for the quantity $\left[\frac{350}{661.5}\right]^2$ and proceed outward from there.

Press	**Display**	
350 ENTER↑	350.00	
661.5 ÷	0.53	
x^2	0.28	Square of bracketed quantity.
.2 ×	0.06	
1 +	1.06	
3.5 y^x	1.21	
1 −	0.21	Contents of left-hand set of brackets are in the stack.
1 ENTER↑	1.00	
6.875 EEX	6.875 00	
CHS 6	6.875 -06	
ENTER↑	6.875000000 -06	
25500 ×	0.18	
−	0.82	
5.2656 CHS y^x	2.76	Contents of right-hand set of brackets are in the stack.
×	0.58	
1 +	1.58	
.286 y^x	1.14	
1 −	0.14	
5 ×	0.70	
$\sqrt{x}$	0.84	Mach number of Daedalus' Harrier.
PRINT x	0.84	



```
  350.00 ENT↑
  661.50   ÷
           X²
     .20   X
    1.00   +
    3.50   Yˣ
    1.00   -
    1.00 ENT↑
6.875-06 ENT↑
25500.00   X
           -
 -5.2656   Yˣ
           X
    1.00   +
    .286   Yˣ
    1.00   -
    5.00   X
           √X
    0.84  ***
```

In working through complex equations like the one containing six levels of parentheses above, you really appreciate the value of the Hewlett-Packard logic system. Because you calculate one step at a time, you don't get "lost" within the problem. You see every intermediate result, and you emerge from the calculation confident of your final answer.

Statistical Functions

Accumulations

Pressing the Σ+ key automatically gives you several different sums and products of the values in the X- and Y-registers at once. In order to make these values accessible for sophisticated statistics problems, they are automatically placed by the calculator into secondary storage registers R_{S4} through R_{S9}. *The only time that information is automatically accumulated in the storage registers is when* Σ+ *(or* Σ- *) is used.* Before you begin any calculations using the Σ+ key, you should first clear the protected secondary storage registers by pressing f CL REG followed by f P⇄S.

When you key a number into the display and press the Σ+ key, each of the following operations is performed:

1. The number that you keyed into the X-register is added to the contents of secondary storage register R_{S4}; ($\Sigma x \rightarrow R_{S4}$).
2. The square of the number that you keyed into the X-register is added to the contents of secondary storage register R_{S5}; ($\Sigma x^2 \rightarrow R_{S5}$).
3. The number in the Y-register of the stack is added to the contents of secondary storage register R_{S6}; ($\Sigma y \rightarrow R_{S6}$).
4. The square of the number in the Y-register of the stack is added to the contents of secondary storage register R_{S7}; ($\Sigma y^2 \rightarrow R_{S7}$).
5. The number that you keyed into the X-register is multiplied by the contents of the Y-register, and the product added to storage register R_{S8}; ($\Sigma xy \rightarrow R_{S8}$).
6. The number 1 is added to storage register R_{S9}, and the total number in R_{S9} then writes over the number in the displayed X-register of the stack. The stack does not lift;

$$\begin{tikzcd}
 & R_{S9} \\
n \arrow[ur] \arrow[dr] & \\
 & X
\end{tikzcd}$$

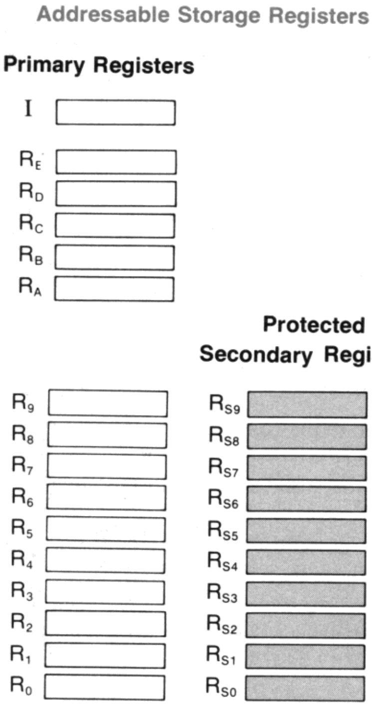
The number that you keyed into the X-register is preserved in the LAST X register, while the number in the stack Y-register remains in the Y-register.

Thus, when you press Σ+, the stack register contents are changed...

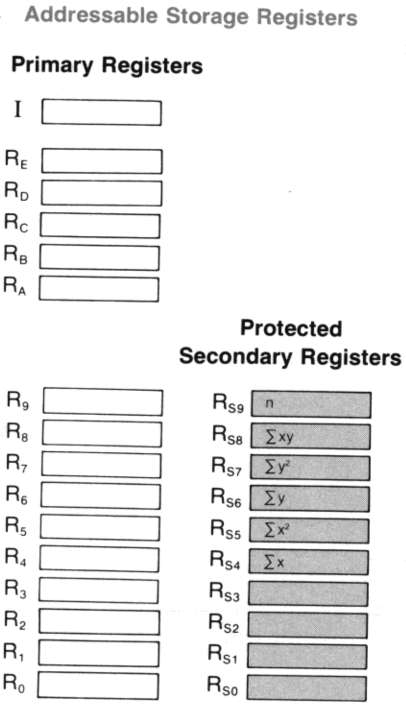
... from this ...		**... to this .**	
T	*t*	**T**	*t*
Z	*z*	**Z**	*z*
Y	*y*	**Y**	*y*
X	x	**X**	*n*
	LAST X	*x*	**LAST X**

... and the storage register contents are changed...

... from this ...



...to this.



Before you begin accumulating results in secondary storage registers R_{S4} through R_{S9} using the Σ+ key, you should first ensure that the contents of these registers have been cleared to zero by pressing f CL REG followed by f P⇄S.

> **Note:** Unlike storage register arithmetic, the Σ+ function allows overflows (i.e., numbers whose magnitudes are greater than 9.999999999 × 10^{99}) in storage register R_{S4} through R_{S9} without registering *Error* in the display or on the printed copy.

After you have accumulated these products and sums using the Σ+ key, they remain in the secondary storage registers, where they are used to compute mean and standard deviation using the x̄ and s functions. To see the contents of these registers, you can list the contents of all the secondary storage registers by pressing f P⇄S followed by f PRINT: REG. Don't forget to press P⇄S again when the listing is completed.

To use *only* the Σx and Σy that you have accumulated in the secondary storage registers, you can press RCL followed by Σ+. This brings Σx into the displayed X-register and Σy into the Y-register, overwriting the contents of those two stack registers. The stack does not lift. (This feature is particularly useful when performing vector arithmetic, like that illustrated on pages 106-108.)

To use *any* of the summations individually, simply exchange the contents of the secondary storage registers with the primary registers by pressing P⇄S; then recall the desired summation by pressing RCL followed by the number key of the register address.

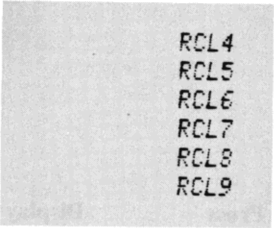
Example: Find Σx, Σx^2, Σy, Σy^2, and Σxy for the paired values of x and y listed below.

y	7	5	9
x	5	3	8

Press	Display	
f CL REG		
f P⇄S	0.00	Ensures that storage registers R_{S4} through R_{S9} contain all zeros initially. (Display assumes no results remain from previous example.)
7 ENTER↑	7.00	
5 Σ+	1.00	First pair is accumulated; n = 1.
5 ENTER↑	5.00	
3 Σ+	2.00	Second pair is accumulated; n = 2.
9 ENTER↑	9.00	
8 Σ+	3.00	Third pair is accumulated; n = 3.
f P⇄S	3.00	Brings contents of secondary registers into primary registers for viewing or individual use.
f PRINT: REG	3.00	You can see all the accumulations by listing the storage register contents.

```
            CLRG
             P⇄S
     7.00   ENT↑
     5.00     Σ+
     5.00   ENT↑
     3.00     Σ+
     9.00   ENT↑
     8.00     Σ+
             P⇄S
            PREG

     0.00      0
     0.00      1
     0.00      2
     0.00      3
    16.00      4
    98.00      5
    21.00      6
   155.00      7
   122.00      8
     3.00      9
     0.00      A
     0.00      B
     0.00      C
     0.00      D
     0.00      E
     0.00      I
```

Press	Display	
RCL 4	16.00	Sum of x values from register R_4.
RCL 5	98.00	Sum of squares of x values from register R_5.
RCL 6	21.00	Sum of y values from register R_6.
RCL 7	155.00	Sum of squares of y values from register R_7.
RCL 8	122.00	Sum of products of x and y values from register R_8.
RCL 9	3.00	Number of entries (n = 3) from register R_9.



```
RCL4
RCL5
RCL6
RCL7
RCL8
RCL9
```

By using the P⇄S function in conjunction with the Σ+ key, you can actually maintain *two* complete sets of products and sums in your HP-97.

Mean

The $\bar{x}$ *(mean)* key is the key you use to calculate the mean (arithmetic average) of data accumulated in secondary registers R_{S4}, R_{S6}, and R_{S9}.

When you press f $\bar{x}$:

1. The mean ($\bar{x}$) of x is calculated using the data accumulated in register R_{S4} (Σx) and R_{S9} (n) according to the formula:

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \quad \left(\text{That is, } \frac{R_{S4}}{R_{S9}} = \bar{x}\right)$$

   The resultant value for $\bar{x}$ is seen in the displayed X-register.

2. The mean ($\bar{y}$) of y is calculated using the data accumulated in register R_{S6} (Σy) and register R_{S9} (n) according to the formula:

$$\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i \quad \left(\text{That is, } \frac{R_{S6}}{R_{S9}} = \bar{y}\right)$$

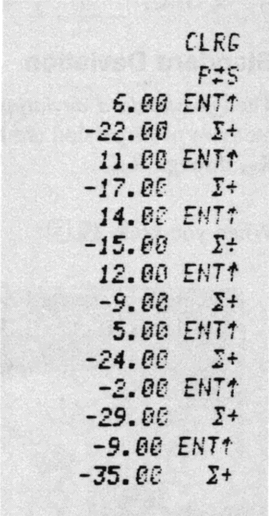
   The resultant value for $\bar{y}$ is available in the Y-register of the stack.

(The easiest way to accumulate the required data in the applicable secondary storage registers is through the use of the Σ+ key as described above. However, you can also place data in the accumulation registers manually, using the P⇄S and STO keys.)

Example: Below is a chart of a daily high and low temperatures for a winter week in Fairbanks, Alaska. What are the *average* high and low temperatures for the week selected?

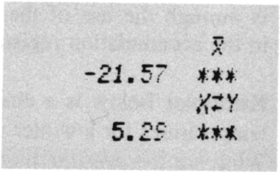
	Sun.	Mon.	Tues.	Wed.	Thurs.	Fri.	Sat.
High	6	11	14	12	5	−2	−9
Low	−22	−17	−15	−9	−24	−29	−35

Press	**Display**	
f CL REG f P⇄S	0.00	Ensures that secondary registers contain all zeros initially. (Display assumes no results remain from previous calculations.)
6 ENTER↑ 22 CHS Σ+	1.00	Number of data pairs (n) is now 1.
11 ENTER↑ 17 CHS Σ+	2.00	Number of data pairs (n) is now 2.
14 ENTER↑ 15 CHS Σ+	3.00	
12 ENTER↑ 9 CHS Σ+	4.00	
5 ENTER↑ 24 CHS Σ+	5.00	
2 CHS ENTER↑ 29 CHS Σ+	6.00	
9 CHS ENTER↑ 35 CHS Σ+	7.00	Number of data pairs (n) is now 7.



```
            CLRG
             P⇄S
     6.00 ENT↑
   -22.00   Σ+
    11.00 ENT↑
   -17.00   Σ+
    14.00 ENT↑
   -15.00   Σ+
    12.00 ENT↑
    -9.00   Σ+
     5.00 ENT↑
   -24.00   Σ+
    -2.00 ENT↑
   -29.00   Σ+
    -9.00 ENT↑
   -35.00   Σ+
```

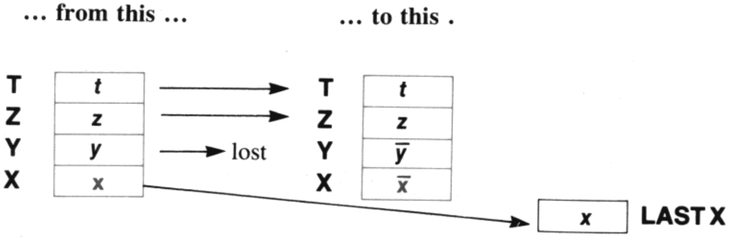
Press	**Display**	
f x̄	-21.57	Average low temperature.
PRINT x	-21.57	
x⇄y	5.29	Average high temperature.
PRINT x	5.29	



```
                x̄
      -21.57  ***
              X⇄Y
        5.29  ***
```

As shown, you can use the PRINT x and x⇄y keys to print the values for x and y.

The illustrations below represent what happens in the stack when you press f x̄. Press f x̄ and the contents of the stack registers are changed...



Standard Deviation

The s (*standard deviation*) key is the key you use to calculate the standard deviation (a measure of dispersion around the mean) of data accumulated in secondary storage registers R_{S4} through R_{S9}.

When you press f s:

1. Sample x standard deviation (s_x) is calculated using the data accumulated in storage registers R_{S5} (Σx^2), R_{S4} (Σx), and R_{S9} (n) according to the formula:

$$s_x = \sqrt{\frac{\Sigma x^2 - \frac{(\Sigma x)^2}{n}}{n - 1}}$$

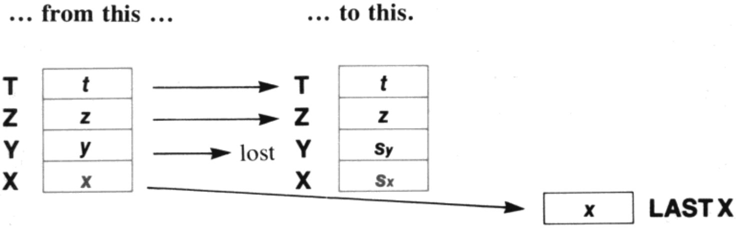
The resultant value for standard deviation of x (s_x) is seen in the displayed X-register.

2. Sample y standard deviation (s_y) is calculated using the data accumulated in storage registers R_{S7} (Σy^2), R_{S6} (Σy), and R_{S9} (n) according to the formula:

$$s_y = \sqrt{\frac{\Sigma y^2 - \frac{(\Sigma y)^2}{n}}{n - 1}}$$

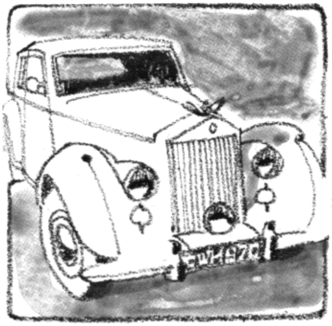
The resultant value for standard deviation of y (s_y) is available in the Y-register of the stack.

Thus, with data first accumulated in secondary storage registers R_{S4} through R_{S9}, when you press f s, the contents of the stack registers are changed...



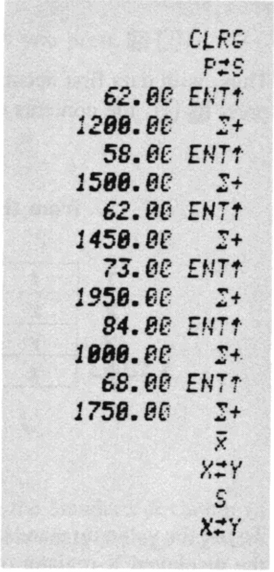
To use the value for standard deviation of y (s_y) simply use the x⇄y key to bring that value into the displayed X-register of the stack.

Example: In a recent survey to determine the age and net worth (in millions of dollars) of six of the 50 wealthiest persons in the United States, the following data were obtained (sampled). Calculate the average age and net worth of the sample, and calculate the standard deviations for these two sets of data.



Age	62	58	62	73	84	68
Net Worth	1200	1500	1450	1950	1000	1750

Press	**Display**	
f CL REG		
f P⇄S	0.00	Ensures that secondary storage registers used for accumulations are cleared to zero initially. (Display assumes no results remain from previous examples.)
62 ENTER↑		
1200 Σ+	1.00	Number of data pairs (n) is 1.
58 ENTER↑		
1500 Σ+	2.00	
62 ENTER↑		
1450 Σ+	3.00	
73 ENTER↑		
1950 Σ+	4.00	
84 ENTER↑		
1000 Σ+	5.00	
68 ENTER↑		
1750 Σ+	6.00	Number of data pairs (n) is 6.
f x̄	1475.00	Average value of net worth.
x⇄y	67.83	Average age of the sample.
f s	347.49	Standard deviation (s_x) of net worth of sample.
x⇄y	9.52	Standard deviation (s_y) of age of sample.



```
             CLRG
              P⇄S
   62.00     ENT↑
 1200.00       Σ+
   58.00     ENT↑
 1500.00       Σ+
   62.00     ENT↑
 1450.00       Σ+
   73.00     ENT↑
 1950.00       Σ+
   84.00     ENT↑
 1000.00       Σ+
   68.00     ENT↑
 1750.00       Σ+
                x̄
              X⇄Y
                S
              X⇄Y
```

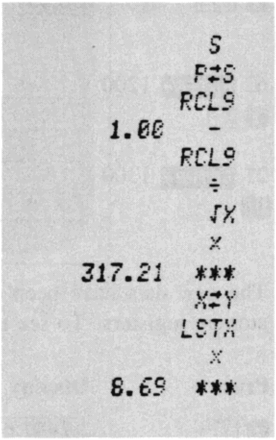
If the six persons used in the sample were actually the *six wealthiest persons,* the data would have to be considered as a population rather than as a sample. The relationship between sample standard deviation (s) and the population standard deviation (σ) is illustrated by the following equation.

$$\sigma = s\sqrt{\frac{n-1}{n}}$$

Since *n* is automatically accumulated in secondary register R_{S9} when data is accumulated, it is a simple matter to convert the sample standard deviations which have already been calculated to population standard deviations.

If the accumulations are still intact from the previous example in secondary registers R_{S4} through R_{S9}, you can calculate the population standard deviations this way:

Press	**Display**	
f s	347.49	Calculate s_x and x_y.
f P⇄S RCL 9	6.00	Recall n.
1 −	5.00	Calculate n − 1.
RCL 9 ÷	0.83	Divide n − 1 by n.
√x ×	317.21	Population standard deviation σ_x.
PRINTx	317.21	
x⇄y	9.52	Brings s_y to the X-register.
f LAST x	0.91	Recall conversion factor.
×	8.69	Population standard deviation σ_y.
PRINTx	8.69	



```
            S
          P⇄S
         RCL9
1.00        -
         RCL9
            ÷
           √X
            ×
317.21    ***
          X⇄Y
         LSTX
            ×
8.69      ***
```

Remember that the accumulations must always be stored in the *secondary* bank of storage registers. Thus, if you have accumulated data using Σ+ and then brought the summations out to the primary registers for viewing using P⇄S, you will have to replace them in the secondary registers by pressing P⇄S again before pressing x̄ or s.

Deleting and Correcting Data

If you key in an incorrect value and have not pressed Σ+, press CLx and key in the correct value.

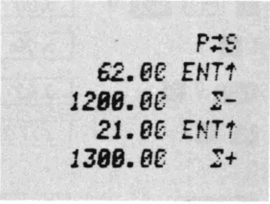
If one of the values is changed, or if you discover after you have pressed the Σ+ key that one of the values is in error, you can correct the summations by using the Σ− (*summation minus*) key as follows:

1. Key the *incorrect* data pair into the X- and Y-registers. (You can use LAST x to return a single incorrect data value to the displayed X-register.)
2. Press f Σ− to delete the incorrect data.
3. Key in the correct values for x and y. (If one value of an x, y data pair is incorrect, both values must be deleted and reentered.)
4. Press Σ+.

The correct values for mean and standard deviation are now obtainable by pressing f x̄ and f s.

For example, suppose the 62-year old member of the *sample* as given above were to lose his position as one of the wealthiest persons because of a series of ill-advised investments in cocoa futures. To account for the change in data if he were replaced in the sample by a 21-year old rock musician who is worth 1300 million dollars:

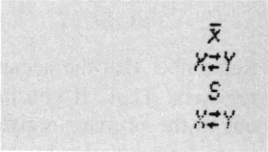
Press	Display	
f P⇄S	8.69	Accumulations replaced in secondary storage registers.
62 ENTER↑ 1200	1200.	Data to be replaced.
f Σ-	5.00	Number of entries (n) is now five.
21 ENTER↑ 1300	1300.	The new data.
Σ+	6.00	Number of entries (n) is six again.



```
         P⇄S
  62.00  ENT↑
1200.00    Σ-
  21.00  ENT↑
1300.00    Σ+
```

The new data have been calculated into each of the summations present in the secondary storage registers. To see the new mean and standard deviation:

Press	Display	
f x̄	1491.67	The new average (mean) worth.
x⇄y	61.00	The new average (mean) age available in X-register for use.
f s	333.79	The new standard deviation for worth.
x⇄y	21.60	The new standard deviation for age is now available in X-register for use.



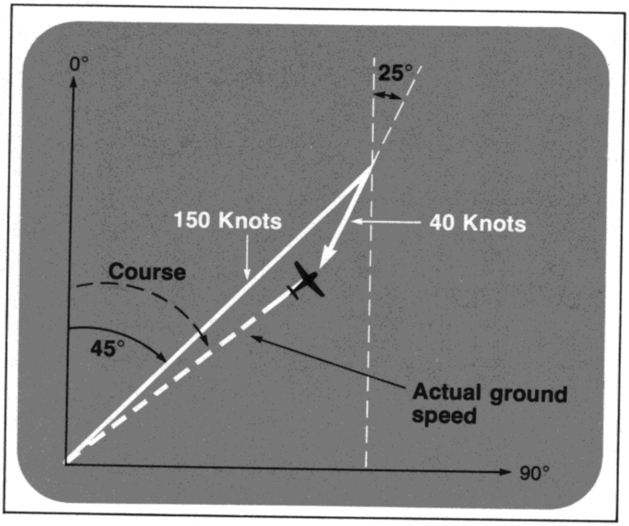
```
  x̄
X⇄Y
  S
X⇄Y
```

Vector Arithmetic

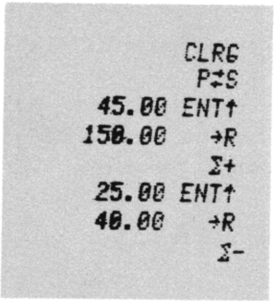
You can use your HP-97 to add or subtract vectors by combining the polar/rectangular conversion functions (the →P and →R keys) with the summation functions (the Σ+ and Σ- keys).

Example: Grizzled bush pilot Apeneck Sweeney's converted Swordfish aircraft has a true air speed of 150 knots and an estimated heading of 45°. The Swordfish is also being buffeted by a headwind of 40 knots from a bearing of 25°. What is the actual ground speed and course of the Swordfish?

Method: The course and ground speed are equal to the difference of the vectors. (Notice that North becomes the x-axis so that the problem corresponds to navigational convention.)

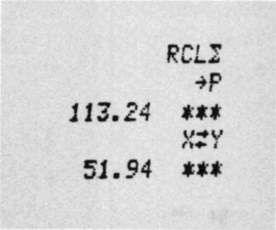


Press	Display	
f CL REG		
f P⇄S	0.00	Ensures that secondary registers used for accumulations are cleared to zero. (Display assumes no results remain from previous examples.)
45 ENTER↑	45.00	*θ* for 1st vector is entered to Y-register.
150	150.	*r* for 1st vector is keyed in.
→R	106.07	Converted to rectangular coordinates.
Σ+	1.00	1st vector coordinates accumulated in storage registers R_{S4} and R_{S6}.
25 ENTER↑	25.00	*θ* for 2nd vector is entered to Y-register.
40	40.	*r* for 2nd vector is keyed in.
→R	36.25	2nd vector is converted to rectangular coordinates.
f Σ-	0.00	2nd vector rectangular coordinates subtracted from those of 1st vector.



```
           CLRG
            P⇄S
     45.00 ENT↑
    150.00   →R
             Σ+
     25.00 ENT↑
     40.00   →R
             Σ-
```

Press	Display	
RCL Σ+	69.81	Recalls both R_{S6} and R_{S4}.
→P	113.24	Actual ground speed in knots of the Swordfish.
PRINT x	113.24	
x⇄y	51.94	Course in degrees of the Swordfish.
PRINT x	51.94	



```
           RCLΣ
             →P
113.24      ***
            X⇄Y
 51.94      ***
```

Part Two
Programming Your HP-97

```
001  *LBLA    21 11
002      5       05
003      0       00
004      0       00
005   STO6    35 06
006      5       05
007      0       00
008    CHS      -22
009   STO7    35 07
010      6       06
011      0       00
012   STO8    35 08
```

Section 6

Simple Programming

If you read the introduction to this handbook, you have already seen that by using the programming capability of your HP-97, you can increase the flexibility of the calculator a hundredfold or more, and you save hours of time in long computations.

With your HP-97 Programmable Printing Calculator, Hewlett-Packard has provided you with a Standard Pac, containing 15 programs already recorded on magnetic cards. You can begin using the programming power of the HP-97 by simply using any of the cards from the Standard Pac, or from one of the other Hewlett-Packard pacs in areas like finance, statistics, mathematics, engineering, or medicine. The growing list of applications pacs is continually being updated and expanded by Hewlett-Packard, to provide you with a wide variety of software support.

However, we at Hewlett-Packard cannot possibly anticipate every problem for which you may want to use your HP-97. In order to get the *most* from your calculator, you'll want to learn how to *program* the HP-97 to solve your every problem. This part of the *HP-97 Owner's Handbook* teaches you step-by-step to create simple programs that will solve complex problems, then introduces you to the many editing features of the HP-97, and finally gives you a glimpse of just how sophisticated your programming can become with the HP-97 Programmable Printing Calculator.

Programming your calculator is an extension of its use as a *manual* problem-solving machine, so if you haven't read Part One, Using Your HP-97 Calculator, you should go back and do so before you begin programming.

After most of the explanations and examples in this part, you will find problems to work using your HP-97. These problems are not essential to your basic understanding of the calculator, and they can be skipped if you like. But we urge that you work them. They are rarely difficult, and they have been designed to increase your proficiency, both in the actual use of the features of your calculator and in creating programs to solve your *own* problems. If you have trouble with one of the problems, go back and review the explanations in the text, then tackle it again.

So that you can apply your own creative flair to the problems, no solutions are given for them. In programming, any solution that gives the correct outputs is the right one—there is no *one* correct program for any problem. In fact, when you have finished working through this part, and learned all the capabilities of the HP-97, you may be able to create programs that will solve many of the problems faster, or in fewer steps, than we have shown in our illustrations.

Now let's start programming!

What Is a Program?

A *program* is nothing more than a series of calculator keystrokes that you would press to solve a problem manually. The calculator remembers these keystrokes when you key them in, then executes them in order at the press of a single key. If you want to execute the program again and again, you have only to press the single key each time.

If you worked through Meet the HP-97 (pages 13-20), you learned how to create, load, run, and record a simple program to solve for the area of a sphere. Now look at a more complex program.

Loading a Prerecorded Program

First, set the calculator controls as follows:

ON-OFF switch **OFF ▮▯ ON** to ON.

Print Mode switch **MAN ▯▮ NORM** (**TRACE**) to MAN.

PRGM-RUN switch **PRGM ▮▯ RUN** to RUN.

Now select the Moon Rocket Lander card from the Standard Pac shipped with your HP-97. Insert side 1 of the card, face up, into the front slot provided on the left side of the calculator, and press it into the slot until the reading mechanism picks it up and propels it out the rear slot. (Let go of the card as soon as you feel it begin to be propelled by the reading mechanism—don't try to restrain its progress). Then insert the card in the window provided above the keys marked **A** **B** **C** **D** **E**.

If the calculator displays `Error`, first clear the error by pressing any key. Then pass side 1 of the card through again.

1. Select the card.

2. Pass the card through the card reader slot.

3. Insert the card in the card window slot.

Some programs are recorded on *both* sides of a magnetic card, so the card must be run through the card reading mechanism twice—once on each side. If a second side of a magnetic card must be read, the calculator prompts you by displaying `Crd` after you have read the first side. However, the Moon Rocket Lander program is fairly short, and so the complete program has been recorded on each side of this factory prerecorded card. You can easily see when a card has been read completely because the calculator will then display the original contents of the X-register. The Moon Rocket Lander program has now been loaded into the calculator, and you can try to ''land'' the calculator on the moon without ''crashing''.

The Game. The game simulates a rocket attempting to land on the moon, with you as the pilot. As the game begins, you are descending at a velocity of 50 ft/sec from a height of 500 feet. Velocity and altitude are shown in a combined display as –50.500, the altitude appearing to the right of the decimal point and the velocity to the left. The negative sign on the velocity indicates downward motion. As the game begins, you have 60 units of rocket fuel.

The object of the game is to control your descent by keying in fuel ''burns'' so that when you reach the surface of the moon (altitude 0), your velocity is also zero and you settle down gently into the powdery moon dust.

When you press **A**, the game begins. The velocity and altitude are shown in the calculator display. Then the number of remaining fuel units are shown, and the display begins a countdown to burn time. The display counts ''3'', ''2'', ''1'', ''0''. When the countdown reaches zero, you have one second to key in a fuel burn. The best choices for fuel burns are digits of 1 through 9. A zero burn, which is very common, is accomplished by doing nothing.

After each burn, the calculator display will show first the new velocity and altitude, then the remaining fuel units, then will count down to zero for you to key in another burn. After each burn this sequence is repeated until you successfully land, (when the display will show you flashing zeros) or you smash into the lunar surface (when the display shows you the flashing crash velocity).

If you attempt to key in a fuel burn during any time other than the one-second ''fire window'', the rocket engine will shut off and you will have to restart it by pressing B. Restarting automatically uses up five units of fuel and gives no thrust.

So press A now and try to land on the moon with your HP-97.

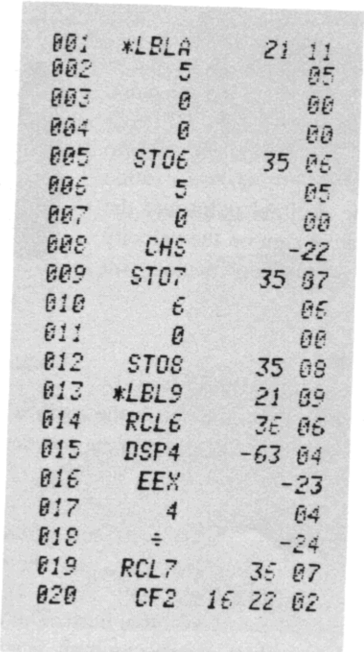
Printing a Program

The printer on your HP-97 will give you a listing of any programs contained in the calculator at any time. To see a listing of the Moon Rocket Lander program that is now loaded in the calculator, first press R/S to stop the running program, then the RTN *(return)* key to ensure that the listing will start from the beginning of the program.

Then press f PRINT: PRGM.

You can see that the calculator printed a long list of numbers and keys. This list is actually a complete list of the *program* that is now in the calculator. Each line on the print-out represents a single step of program memory. Stop the printing of the program after 20 lines or so by pressing R/S again.

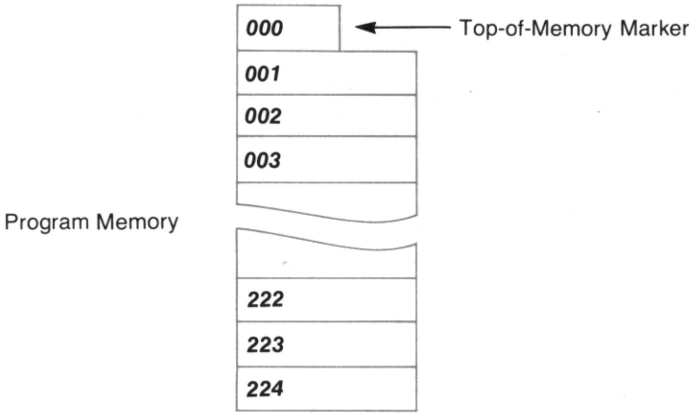
Your print-out should look like the one shown here:



```
001   *LBLA      21 11
002       5         05
003       0         00
004       0         00
005    STO6      35 06
006       5         05
007       0         00
008     CHS        -22
009    STO7      35 07
010       6         06
011       0         00
012    STO8      35 08
013   *LBL9      21 09
014    RCL6      36 06
015    DSP4     -63 04
016     EEX        -23
017       4         04
018       ÷        -24
019    RCL7      36 07
020     CF2   16 22 02
```

Program Memory

In your HP-97, keystrokes that make up a program are stored in a portion of the calculator called *program memory*. Program memory consists of 224 steps, and is separate from the stack and the storage registers.



Each step of program memory is identified by a *step number,* from step 001 to 224. These step numbers can be seen on the left side of the print-out.

In addition to the 224 steps of program memory in which you can load keystrokes for programs, program memory also contains step 000. No functions can be loaded into step 000, and in fact, step 000 serves only as a kind of marker within memory, a convenient ''starting point'' when you begin loading a program.

As you may remember from the program you created, loaded, executed, and recorded in Meet the HP-97, a program is nothing more than a series of keystrokes you would press to solve a problem manually. The center column of the print-out shows you a symbol for the instruction that is contained in that program memory step. Each instruction (or function) is contained in a single step of program memory, no matter how many keystrokes it requires.

For example, if you examine the print-out of the Moon Rocket Lander program, you will see that step 001 contains the instruction LBLA. These are the LBL A keys. Step 002 contains the digit 5, step 003 contains the digit 0, step 004 contains the digit 0, and step 005 contains the instruction STO 6. Thus, in order to load this portion of the program into the calculator from the keyboard, you would have pressed the following keys:

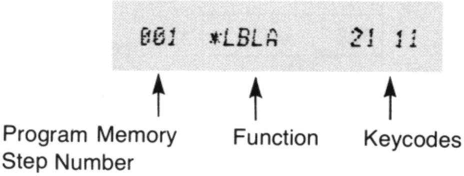
LBL A
500
STO 6

The column of numbers on the right of the print-out shows *keycodes* for the instructions.

Keycodes

When you press f PRINT: PRGM, you can easily identify what instruction is contained in each step of program memory by the key symbols. When you are actually keying in a program, however, you will find instructions identified *in the display* by their *keycodes*.

Refer to step 001 of the print-out for the Moon Rocket Lander. The step number is on the left, the symbol for the instruction is in the center, and on the right are the *keycodes* for the instruction. Each key on the HP-97 keyboard has a two-digit keycode.



```
001  *LBLA     21 11
```

For each keycode not preceded by a minus sign, the first digit denotes the *row* of the key on the left half of the keyboard and the second digit denotes the *number* of the key in that row. So keycode 21 identifies the 2nd row of the left keyboard, 1st key in that row. Always count from the top down and from left to right. Each key, no matter how large, counts as one.

The key in the 2nd row, 1st key of the left side of the keyboard is the LBL key.

The second keycode in the first line of the program is 11. This identifies the 1st row, 1st key; the A key. So the complete instruction identified by the keycodes 21 11 is LBL A.

Using this handy matrix system, you can easily identify any key by its keycode. For convenience digit keys are identified by keycodes 00 through 09. A keycode with a minus sign (–) before it identifies a key from the *right* side of the keyboard.

Thus you can see that the keycode 05 at step 002 of the Moon Rocket Lander program identifies the digit 5, and the 00 keycode in step 003 and step 004 each identifies the digit 0.

Now look at program step 015 on the print-out of the Moon Rocket Lander. The minus sign on the keycode –63 identifies the key as being on the right side of the keyboard, while the address 63 identifies the 6th row, 3rd key, the DSP key. Keycode 04 identifies the digit 4 key. So you can see that the complete instruction is DSP 4.

Notice that each step of program memory can hold a complete operation, no matter whether the operation consists of one (e.g., e^x), two (e.g., f π), or three (e.g., STO + 5) keystrokes.

Problems

1. What would be the keycodes for the following operations: 1/x, f GRD, f H.MS+, STO + 1?

2. What operations are identified by the following keycodes: –41, 16 54, 16 01, 16–63?

3. How many steps of program memory would be required to load the following sections of programs?

   a. 2 ENTER↑ 3 +

   b. 10 STO 6 RCL 6 ×

   c. 100 STO 2 50 STO × 2 RCL 2 f π ×

Clearing a Program

When you ran the magnetic card containing the Moon Rocket Lander program through the card reader with the PRGM-RUN switch set to RUN, the program was copied from the card into program memory in the calculator. Before you can key in a program, you will first want to clear, or erase, the Moon Rocket Lander program from the calculator's program memory. You can clear a program in any of three ways.

To clear a program from the calculator, you can *either:*

1. Press f CLPRGM with the PRGM-RUN switch PRGM ▯■ RUN set to PRGM. This replaces whatever instructions are in program memory with R/S instructions. In PRGM mode, f CLPRGM also specifies FIX 2 display mode, DEGREES trigonometric mode, and clears all flags. (A R/S instruction encountered in a program stops the execution of that program. A flag is a status indicator within a program. More about these later.) The stack and storage register contents remain intact when f CLPRGM is pressed.

2. Pass another magnetic card containing a program through the card reader with the PRGM-RUN switch PRGM ■▯ RUN set to RUN. This replaces whatever instructions are contained in the calculator's program memory with the instructions for the new program. (Reading a blank card does not alter the contents of program memory, and the calculator displays Error to indicate that the card has not been read.)

3. Turn the HP-97 OFF, then ON. This also replaces whatever instructions are in program memory with R/S instructions.

Now you are going to write your own program into the calculator from the keyboard, so to first clear the HP-97 of the previous program:

Slide the PRGM-RUN switch PRGM ▯■ RUN to PRGM.

Press f CLPRGM to clear program memory.

Creating Your Own Program

In Meet the HP-97, you created, loaded, ran, and recorded a program that solved for the surface area of a sphere, given the diameter of that sphere. Now let's create, load, and run another program to show you how to use some of the other features of the HP-97.

If you wanted to use the HP-97 to manually calculate the area of a circle using the formula $A = \pi r^2$ you could first key in the radius r, then square it by pressing x^2. Next you would summon the quantity pi into the display by pressing f π. Finally you would multiply the squared radius and the quantity pi together by pressing ×.

Remember that a *program* to solve a problem is nothing more than the keystrokes you would press to solve the problem manually. Thus, in order to create a program for the HP-97 that will solve for the area of *any* circle, you use the same keys you pressed to solve the problem manually.

The keys that you used to solve for area of a circle according to the formula $A = \pi r^2$ are:

x^2
f π
×

You will load these keystrokes into program memory. In addition, your program will contain two other operations, LBL A and RTN.

The Beginning of a Program

To define the beginning of a program, you should use a LBL *(label)* instruction followed by one of the letter keys (A, B, C, D or E, or f a through f e). The use of labels permits you to have several different programs or parts of programs loaded into the calculator at any time, and to run them in the order you choose.

The digit keys (0 through 9), when prefaced by LBL, can also be used to define the beginning of a program. However, since you must then press GSB n from the keyboard if you want to select and execute that program, LBL 0 through LBL 9 are usually reserved for defining *routines*–that is, parts of larger programs.

Ending a Program

To define the end of a program, you should use a RTN *(return)* instruction. When the calculator is executing a program and encounters a RTN instruction in program memory, it stops. For example, if the calculator were executing a program that had begun with LBL C, when it encountered a RTN, it would stop. Another instruction that will cause a running program to stop is R/S. When a running program executes a R/S instruction in program memory, it stops just as it does when it executes RTN. Good programming practice, however, dictates that you normally use a RTN instruction rather than R/S to define the end of your program.

The Complete Program

The complete program to solve for the area of any circle given its radius is now:

LBL A	Assigns name to and defines beginning of program.
x^2	Squares the radius.
f π	Summons pi into the display.
×	Multiplies r^2 by π and displays the answer.
RTN	Defines the end of and stops the program.

Loading a Program

You can load a program into the calculator in either of two ways:

1. By passing a magnetic card containing program instructions through the card reader with the PRGM-RUN switch PRGM ■▥ RUN set to RUN.
2. By setting the PRGM-RUN switch PRGM ▥■ RUN to PRGM *(program)* and pressing the keys from the keyboard in the natural order you would press them to solve a problem manually.

Since we do not have a magnetic card that contains the program we have written to solve for the area of a circle, we will use this second method to load our program.

To load a program from the keyboard, simply slide the PRGM-RUN switch PRGM ▯■ RUN to PRGM *(program)*. When the PRGM-RUN switch is in the PRGM position, the functions and operations that are normally executed when you press the keys are not executed. Instead, they are *stored* in program memory for later execution. *All operations on the keyboard except six can be loaded into program memory for later execution.* The six operations that cannot be loaded in as part of a program are:

f CLPRGM, BST, SST, f DEL, GTO . n n n, f PRINT: PRGM

These six operations are used to help you load, edit, and modify your programs in the calculator.

All other functions when pressed with the PRGM-RUN switch PRGM ▯■ RUN in PRGM mode are loaded into the calculator as program instructions to be executed later.

So if you have not already done so:

1. Slide the PRGM-RUN switch PRGM ▯■ RUN to PRGM.
2. Press f CLPRGM to clear program memory of any previous programs and to reset the calculator to the top of program memory.

You can tell that the calculator is at the top of program memory because the digits *000* will appear at the left of the display. The digits appearing at the left of the display with the PRGM-RUN switch PRGM ▯■ RUN set to PRGM indicate the *program memory step number* being shown at any time.

The keys that you must press to key in the program for the area of a circle are:

LBL A
x^2
f π
×
RTN

Press the first key LBL, of the program.

Press	**Display**
LBL	*000*

Now press the second key of the program, the A key.

Press	**Display**
A	001 21 11

When the step number (001) of program memory appears on the left of the display, it indicates that a complete operation has been loaded into that step. As you can see from the keycodes present on the right side of the display, the operation is LBL (keycode 21) A (keycode 11). Nothing is loaded into program memory until a complete operation (whether 1, 2, or 3 keystrokes) has been specified.

Now load the remainder of the program by pressing the keys. Observe the program memory step numbers and keycodes.

Press	**Display**
x^2	002 53
f π	003 16-24
×	004 -35
RTN	005 24

The program for solving the area of a circle given its radius is now loaded into program memory of the HP-97. Notice that nothing could be loaded into the top of memory marker, step 000.

Running a Program

To run a program, you have only to slide the PRGM-RUN switch to RUN, key in any "unknown" data that is required, and press the letter key (A through E, f a through f e) that labels your program.

For example, to use the program now in the calculator to solve for circles with radii of 3 inches, 6 meters, and 9 miles:

First, slide the PRGM-RUN switch PRGM ▮▯ RUN to RUN.

Press	**Display**	
3 A	28.27	Square inches.
6 A	113.10	Square meters.
9 A	254.47	Square miles.

Now let's see how the HP-97 executed this program.

Searching for a Label

When you switched the PRGM-RUN switch PRGM ■▥ RUN to RUN, the calculator was set at step 005 of program memory, the last step you had filled with an instruction when you were loading the program. When you pressed the A key, the calculator began *searching* sequentially downward through program memory, beginning with that step 005, for a LBL A instruction. When the calculator searches, it does not execute instructions.

The calculator reached the last step of program memory, step 224, without encountering a LBL A instruction. It then passed step 000 again and continued searching sequentially through program memory for a LBL A instruction. Only when the calculator found a LBL A instruction in step 001 did it begin executing instructions.

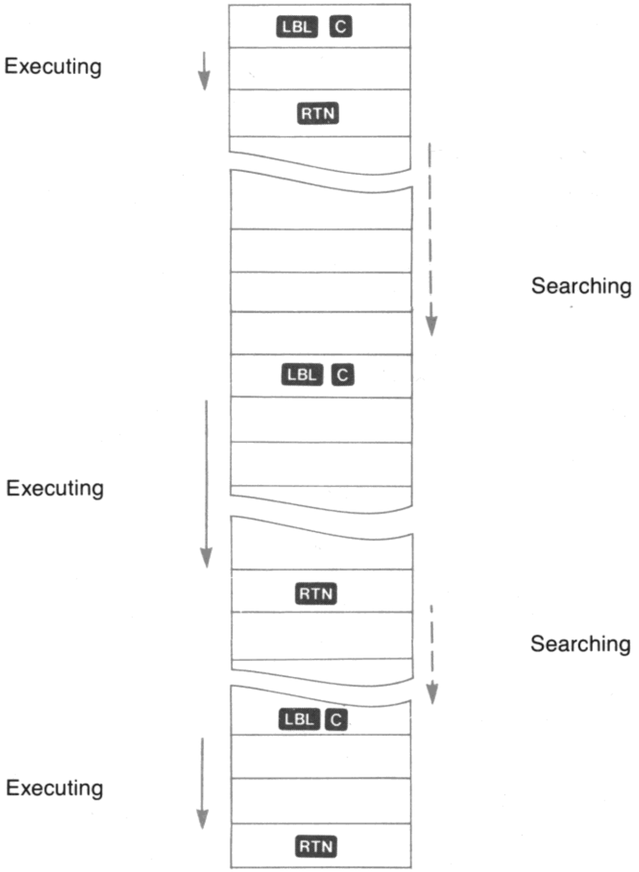
Executing Instructions

When the calculator found the LBL A instruction in step 001, it ceased *searching* and began *executing* instructions. The calculator executes instructions in exactly the order you keyed them in, performing the x^2 operation in step 002 first, then f π as in step 003, etc., until it executes a RTN instruction or a R/S *(run/stop)* instruction. Since a RTN instruction is executed in step 005, the calculator stops there and displays the contents of the X-register. (To see the next step number of program memory after the one at which the calculator has stopped, you can briefly switch the PRGM-RUN switch PRGM ▥■ RUN to PRGM.)

If you key in a new value for the radius of a circle in RUN mode and press A, the HP-97 repeats this procedure. It searches sequentially downward through program memory until it encounters a LBL A instruction, then sequentially executes the instructions contained in the next steps of program memory until it executes a RTN or a R/S instruction.

You can see that it is possible to have many different programs or parts of a program loaded in the HP-97 at any time. You can run any one of these programs by pressing the letter key (A through E, f a through f e) that corresponds with its label.

It is also possible to have several different programs or routines defined by the same label. For example, suppose you had three programs in your HP-97 that were defined by LBL C. When you pressed C, the calculator would search sequentially through program memory from wherever it was located until it encountered the first LBL C instruction. The HP-97 would then execute instructions until it executed a RTN or a R/S instruction and stopped. When you pressed C again, the calculator would resume seaching sequentially from the RTN or R/S through program memory until it encountered the second LBL C instruction, whereupon it would execute that LBL C and all subsequent instructions until it executed a RTN or a R/S instruction and stopped. When you pressed C a third time, the HP-97 would search downward to the third LBL C instruction and execute that program.



If you try to press a letter key (A through E , f a through f e) that is not contained as a label instruction in program memory, the HP-97 will execute no instruction and will display Error . For example, if your HP-97 contains only the program for area of a circle that you keyed in earlier, you can see this by simply pressing another letter key.

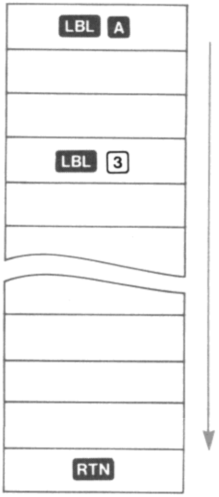
First, ensure that the PRGM-RUN switch PRGM ■▯ RUN is set to RUN.

Press	**Display**
D	Error

To clear the error from the display, you can press CLx, or any key on the keyboard, or you can slide the PRGM-RUN switch PRGM ▯■ RUN to PRGM. The calculator remains set at the current step of program memory.

Labels and Step 000

The labels (A through E, f a through f e, 0 through 9) in your programs act as addresses—they tell the calculator where to begin or resume execution. When a label is encountered as part of a program, execution merely ''falls through'' the label and continues onward. For example, in the program segment shown below, when you pressed A, execution would begin at LBL A and continue downward through program memory, on through the LBL 3 instruction, continuing until the RTN was encountered and execution was stopped.



When you press A

... execution begins here.

No RTN here...

... so execution falls through the LBL 3 instruction...

...and continues to the RTN.

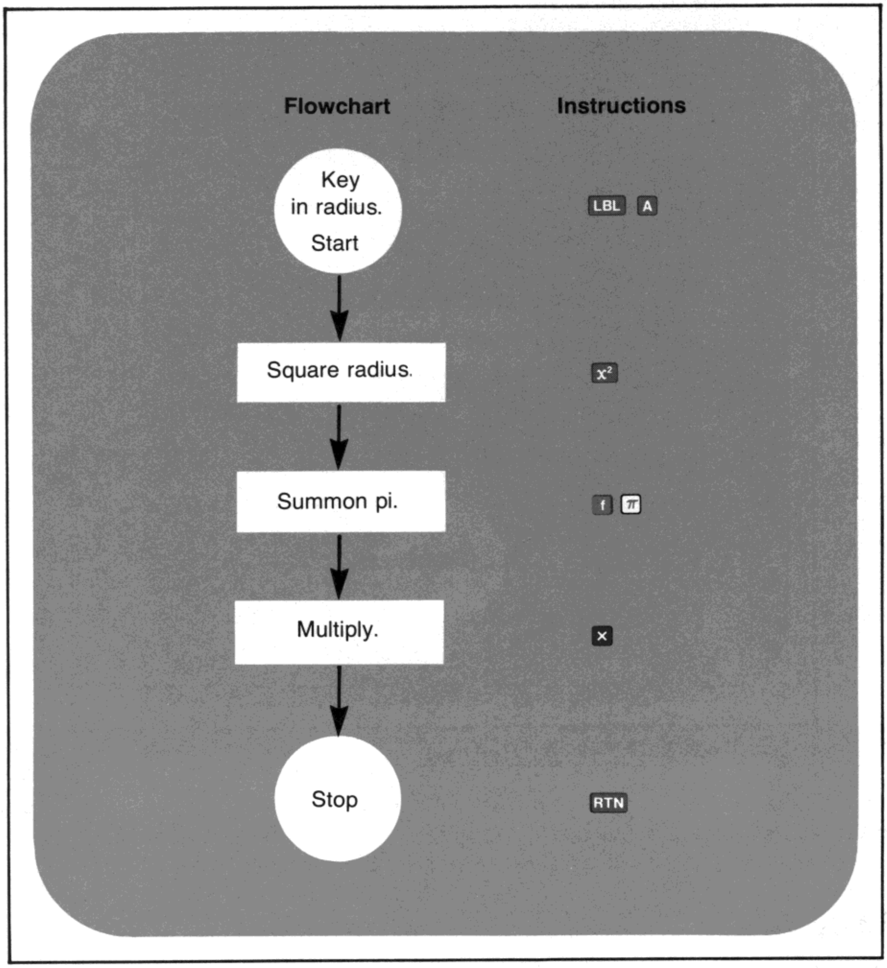
Execution falls through step 000, too. You can load instructions into steps 001 through 224 of program memory, but you cannot load an instruction into step 000. In fact, step 000 merely acts as a kind of label in program memory, a beginning point for the loading of a program. When step 000 is encountered by a running program, execution continues without a halt from step 224 to step 001, just as if step 000 were not there.

Flowcharts

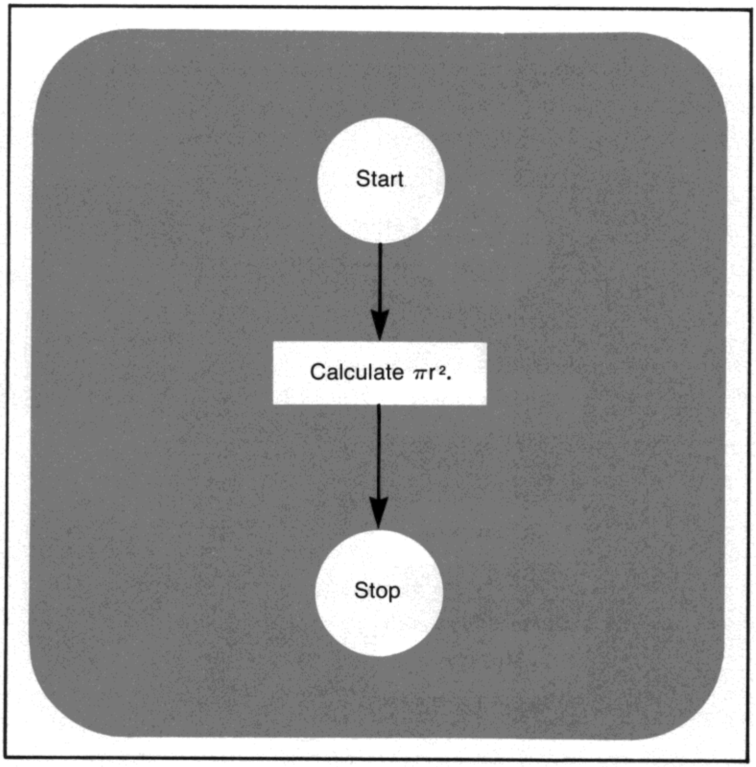
At this point, we digress for a moment from our discussion of the calculator itself to familiarize ourselves with a fundamental and extremely useful tool in programming—the flowchart.

A flowchart is an *outline* of the way a program solves a problem. With 224 possible instructions, it is quite easy to get ''lost'' while creating a long program, especially if you try to simply load the complete program from beginning to end with no breaks. A flowchart is a shorthand that can help you design your program by breaking it down into smaller groups of instructions. It is also very useful as documentation—a road map that summarizes the operation of a program.

A flowchart can be as simple or as detailed as you like. Here is a flowchart that shows the operations you executed to calculate the area of a circle according to the formula $A = \pi r^2$. Compare the flowchart to the actual instructions for the program:



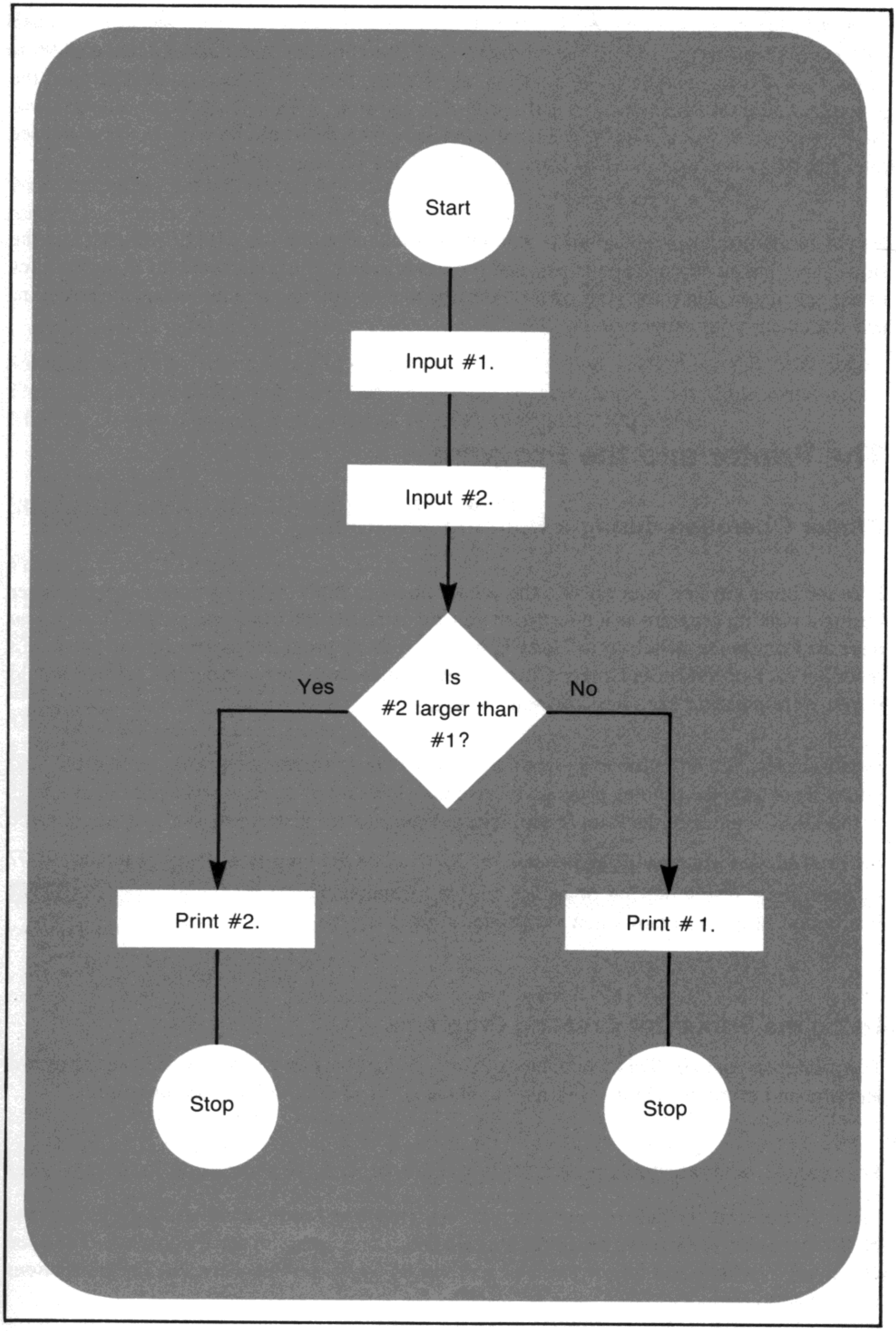
You can see the similarities. At times, a flowchart may duplicate the set of instructions exactly, as shown above. At other times, it may be more useful to have an entire group of instructions represented by a single block in the flowchart for the program to calculate the area of a circle:



Here an entire group of instructions was replaced by one block in the flowchart. This is a common practice, and one which makes a flowchart extremely useful in visualizing a complete program.

You can see how a flowchart is drawn linearly, from the top of the page to the bottom. This represents the general flow of the program, from beginning to end. Although flowcharting symbols sometimes vary, throughout this handbook and in the Standard Pac, we have held to the convention of circles for the beginning and end of a program or routine, and rectangles to represent groups of functions that take an input, process it, and yield a single output. We have used a diamond to represent a *decision,* where a single input can yield either of two outputs.

For example, if you had two numbers and wished to write a program that would print only the larger, you might design your program by first drawing a flowchart that looks like the one shown here on the following page:


Start
Input #1.
Input #2.
Is
#2 larger than
#1?
Yes
No
Print #2.
Print # 1.
Stop
Stop

After drawing the flowchart, you would go back and substitute groups of instructions for each element of the flowchart. When the program was loaded into the calculator and run, if #2 was larger than #1, the answer to the question "Is #2 larger than #1?" would be YES, and the program would take the left-hand path, print #2, and stop. If the answer to the question was NO, the program would execute the right-hand path, and #1 would be printed. (You will see later the many decision-making instructions available on your HP-97.)

As you work through this handbook, you will become more familiar with flowcharts. Use the flowcharts that illustrate the examples and problems to help you understand the many features of the calculator, and draw your own flowcharts to help you create, edit, eliminate errors in, and document your programs.

The Printer and the Program

Printer Operation during a Running Program

Like the other keys on your HP-97, the printer operates in the same natural, normal manner during a running program as it does when you are using the calculator manually. Thus, if you have the Print Mode switch set to TRACE during a running program, the printer will preserve a record of each operation and intermediate result calculated during the program, just as it would if you were pressing each key yourself.

In most cases, you will probably want the HP-97 to run a program as quickly as possible, and you will not want the running program to have to "slow down" to engage the printer, as it does in the TRACE mode of the Print Mode switch. To ensure the fastest program execution, place the Print Mode switch MAN TRACE NORM in the MAN or NORM position. If you want the HP-97 to print some results during or at the end of a running program, simply place a PRINTx instruction in the program wherever printed results are required.

Using the Printer for Creating Programs

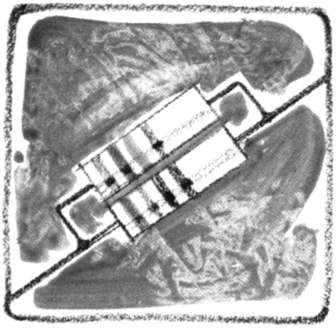
The printer on your HP-97 is a valuable and time-saving tool that you can use not only to record answers and program listings, but also to aid you in creating and editing programs.

For example, when creating a program you can use the printer to generate the list of keystrokes that will later form the bulk of your program. With the Print Mode switch MAN TRACE NORM set to NORM, the printer records a history of the calculations necessary to solve a problem. Then you can simply go back and follow the listing produced by the printer when loading instructions into program memory.

Example: The formula to calculate the total resistance of two parallel resistances in an electrical circuit is:

$$R_T = \frac{R1R2}{R1 + R2}$$

Write a program that will permit you to key in any two parallel resistances and calculate the total resistance.

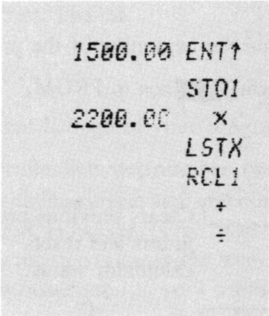


Solution: Begin by selecting a pair of sample resistances; say, 1500 ohms and 2200 ohms. Then solve for the total resistance using the formula above with the Print Mode switch set to NORM. To solve for parallel resistances of 1500 ohms and 2200 ohms:

Slide the Print Mode switch MAN ▮ TRACE ▯ NORM to NORM.

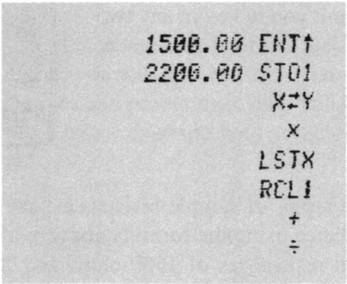
Press	**Display**
1500 ENTER↑	1500.00
STO 1	1500.00
2200	2200.
×	3300000.00
f LAST X	2200.00
RCL 1	1500.00
+	3700.00
÷	891.89

As you can see you have generated a list of keystrokes that solves the problem. Your list should look like this:



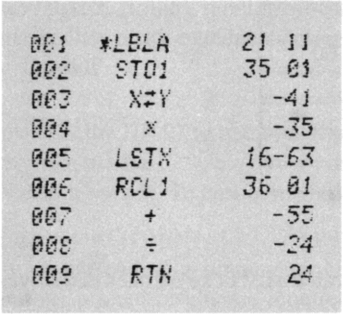
```
1500.00 ENT↑
        STO1
2200.00    ×
        LSTX
        RCL1
           +
           ÷
```

Now all you have to do is to add to this list slightly so that it will work for *all* values of R1 and R2. With the Print Mode switch still set to NORM, solve the problem again so that the value for R2 will be stored in register R_1 and the value for R1 will be placed in the LAST X register when the multiplication operation is performed:



```
1500.00 ENT↑
2200.00 STO1
           X⇄Y
             ×
          LSTX
          RCL1
             +
             ÷
```

Now assume that the values for R1 and R2 have been input to the Y- and X-registers, respectively. Define the program with LBL A and RTN, and it looks like this:



```
001  *LBLA      21 11
002   STO1      35 01
003    X⇄Y        -41
004      ×        -35
005   LSTX      16-63
006   RCL1      36 01
007      +        -55
008      ÷        -24
009    RTN         24
```

Switch to PRGM mode and define the beginning of the program:

Slide the PRGM-RUN switch PRGM ▮ RUN to PRGM.

Press	**Display**	
f CLPRGM	000	Clears previous programs and resets calculator status.
LBL A	001 21 11	

Now follow the list of keystrokes to key in the rest of the program. After the last keystroke, define the end of the program with RTN.

Press	**Display**
STO 1	002 35 01
x≷y	003 -41
×	004 -35
f LAST x	005 16-63
RCL 1	006 36 01
+	007 -55
÷	008 -24
RTN	009 24

Now switch back to RUN mode and run the program to see that it yields the same answer for your test case:

Slide the PRGM-RUN switch PRGM ▮▯ RUN to RUN.

Press	**Display**
1500 ENTER↑	1500.00
2200	2200.
A	891.89

As you can see, the printer is a great aid in the creation of programs. To view the complete program listing, press RTN f Print: PRGM.

Program Listing

The printer gives you two different listings of your programs. With the Print Mode switch MAN TRACE NORM set to MAN, the printer lists step number, operation, and keycode for each step of program memory, beginning with the current step and continuing until step 224 or two R/S instructions in a row are encountered. However, in the NORM or TRACE positions of the Print Mode switch, the printer lists only the step number and instruction, omitting the keycode. This type of listing is generated much faster, and is very useful for preserving a record of your program without the keycodes.

Here are examples of these listings:

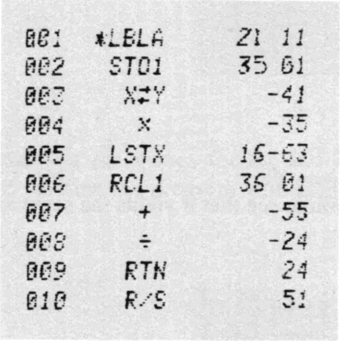
Print Mode switch

set to MAN.

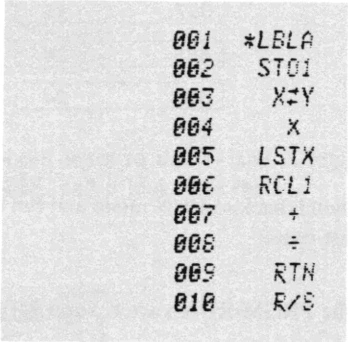
Print Mode switch

set to TRACE or

set to NORM.



```
001  *LBLA      21 11
002   STO1      35 01
003    X⇄Y        -41
004     ×         -35
005   LSTX      16-63
006   RCL1      36 01
007     +         -55
008     ÷         -24
009    RTN         24
010    R/S         51
```



```
001  *LBLA
002   STO1
003    X⇄Y
004     ×
005   LSTX
006   RCL1
007     +
008     ÷
009    RTN
010    R/S
```

Printing a Space

If you wish to insert a space between portions of your print-out or between answers, you can use the f PRINT: SPACE *(print space)* function. Like the paper advance pushbutton, this function advances the paper one space without printing. Digit entry is not terminated by the paper advance pushbutton or by the f PRINT: SPACE function.

For example:

Press	**Display**	
123.	123.	
f PRINT: SPACE	123.	Paper advances one space.
456.	123456.	Digit entry was not terminated by printing a space.

From the keyboard, of course, the paper advance pushbutton is the easiest way to advance the paper without printing. From a program, however, the use of f PRINT: SPACE instructions allows you to place as many spaces as you desire in your printed results.

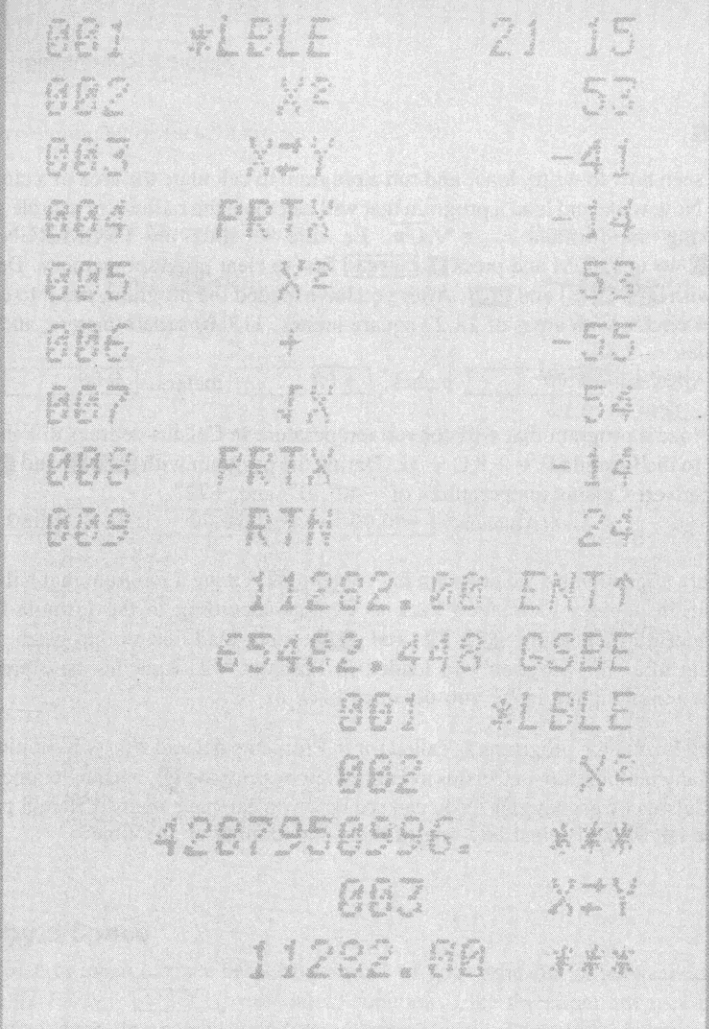
Problems

1. You have seen how to write, load, and run a program to calculate the area of a circle from its radius. Now write and load a program that will calculate the radius r of a circle given its area A using the formula $r = \sqrt{A/\pi}$. Be sure to slide the PRGM-RUN switch PRGM ■ RUN to PRGM and press f CL PRGM first to clear program memory. Define the program with LBL f b and RTN. After you have loaded the program, run it to calculate the radii of circles with areas of 28.27 square inches, 113.10 square meters, and 254.47 square miles.
   (Answers: 3.00 inches, 6.00 meters, 9.00 miles.)

2. Write and load a program that will convert temperature in Celsius degrees to Fahrenheit, according to the formula F = 1.8 C + 32. Define the program with LBL C and RTN, and run it to convert Celsius temperatures of −40°, 0°, and +72°.
   (Answers: -40.00 ° F, 32.00 ° F, 161.60 °F.)

3. Immediately after running the program in Problem #2, create a program that will convert temperature in degrees Fahrenheit back to Celsius according to the formula C = 5/9 (F–32), defining it using LBL D and RTN, and load it into program memory immediately after the program you loaded in Problem #2. Run this new program to convert the temperatures in °F you obtained back to °C.

If you wrote and loaded the programs as called for in Problems #2 and #3, you should now be able to convert any temperature in Celsius to Fahrenheit by pressing C, and any temperature in Fahrenheit to Celsius by pressing D. You can see how you can have many different programs loaded into the HP-97 and select any one of them for running at any time.



```
001  *LBLE      21 15
002     X²         53
003    X⇄Y        -41
004   PRTX        -14
005     X²         53
006      +        -55
007     √X         54
008   PRTX        -14
009    RTN         24
         11282.00 ENT↑
        65482.448 GSBE
              001  *LBLE
              002     X²
      4287950996.   ***
              003    X⇄Y
         11282.00   ***
```

Section 7

Program Editing

Often you may want to alter or add to a program that is loaded in the calculator. On your HP-97 keyboard, you will find several editing functions that permit you to easily change any steps of a loaded program *without* reloading the entire program.

As you may recall all functions and operations on the HP-97 keyboard can be recorded as instructions in program memory except six. These six functions are *program editing and manipulation functions,* and they can aid you in altering and correcting your programs.

Nonrecordable Operations

f **CLPRGM** is one keyboard operation that cannot be recorded in program memory. When you press **f** **CLPRGM** with the PRGM-RUN switch **PRGM** ▯■ **RUN** set to PRGM, program memory is cleared to **R/S** instructions and the calculator is reset to top of memory (step 000) so that the first instruction will be stored in step 001 of program memory. **f** **CLPRGM** also sets the trigonometric mode to DEGREES, the display mode to FIX 2, and clears flags F0, F1, F2, and F3 (more about flags in section 13). With the PRGM-RUN switch set to RUN, **CLPRGM** only cancels the **f** prefix key that you have pressed before it.

SST *(single step)* is another nonrecordable operation. When you press **SST** with the PRGM-RUN switch **PRGM** ▯■ **RUN** set to PRGM, the calculator moves to and displays the next step of program memory. When you press **SST** down with the Program Mode switch **PRGM** ■▯ **RUN** set to RUN, the calculator displays the next step of program memory—when you release the **SST** key, the calculator executes the instruction loaded in that step. **SST** permits you to single step through a program, executing the program one step at a time or merely viewing each step without execution, as you choose.

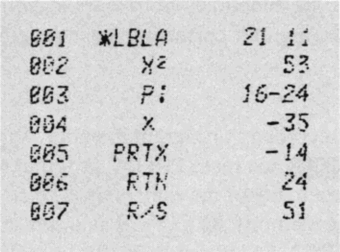
BST *(back step)* is a nonrecordable keystroke that displays the *previous* step of program memory. When you press **BST** with the Program Mode switch **PRGM** ▯■ **RUN** set to PRGM, the calculator moves to and displays the previous step of program memory. When you press down **BST** with the Program Mode switch **PRGM** ■▯ **RUN** set to RUN, the calculator moves to and displays the contents of the previous step of program memory. When you then release **BST**, the original contents of the X-register are displayed. No instructions are executed.

GTO **•** **n** **n** **n** is another keyboard operation that cannot be loaded as an instruction. (**GTO** **A** or **GTO** followed by any other label, however, *can* be loaded as a program instruction. More about the use of this instruction later.) Whether the Program Mode switch is set to RUN or PRGM, when you press **GTO** **•** followed by a three-digit step number, the calculator transfers execution so that the next operation or instruction will begin at that step number. No instructions are executed. If the calculator is in RUN mode, you can verify that the calculator is set to the specified step by briefly sliding the Program Mode switch **PRGM** ▯■ **RUN** to PRGM. The **GTO** **•** **n** **n** **n** operation is especially useful in PRGM mode because it permits you to jump to any location in program memory for editing of or additions or corrections to your programs.

The DEL *(delete)* key is a nonrecordable operation that you can use to delete instructions from program memory. When the Program Mode switch PRGM ■ RUN is set to PRGM and you press f DEL, the instruction at the current step of program memory is erased, and all subsequent instructions in program memory move upward one step. The section of program memory shown below illustrates what would happen when you press f DEL with the calculator set to step 005.

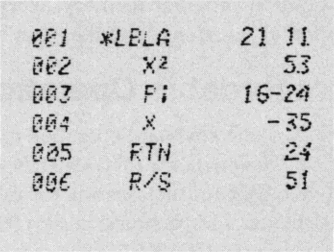
With the calculator set to step 005, when you press f DEL, program memory is changed…

… from this …



```
001  *LBLA     21 11
002     X²        53
003     Pi     16-24
004     X        -35
005   PRTX       -14
006    RTN        24
007    R/S        51
```

… to this.



```
001  *LBLA     21 11
002     X²        53
003     Pi     16-24
004     X        -35
005    RTN        24
006    R/S        51
```

In RUN mode, pressing DEL merely clears (cancels) an f prefix key that you have pressed.

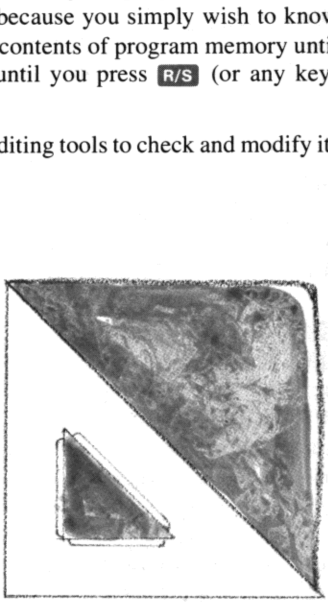
Another key, f PRINT: PRGM is not recordable in program memory but otherwise operates the same way whether the Print Mode switch is set to PRGM or RUN.

When you press f PRINT: PRGM, regardless of the setting of the Print Mode switch, the HP-97 immediately begins printing the contents of program memory beginning with the step to which the calculator is set. This feature allows you to list a complete program at any time, whether you are loading it, single-stepping through it, or because you simply wish to know the contents of program memory. The calculator prints the contents of program memory until it encounters two R/S instructions in succeeding steps, until you press R/S (or any key) from the keyboard, or until step 224 is printed.

Now let's load a program from the keyboard and use these editing tools to check and modify it.

Pythagorean Theorem Program

The following program computes the hypotenuse of any right triangle, given the other two sides. The formula used is $c = \sqrt{a^2 + b^2}$.



On the following page are instructions for the program (basically, the same keys you would press to solve for c manually), assuming that values for sides a and b have been input to the X- and Y-registers of the stack.

To load the program:

First slide the Program Mode switch PRGM ▯▮ RUN to PRGM. Then press **f** **CLPRGM** to clear program memory of any previous programs and reset the calculator to step 000 of program memory.

Finally, load the program by pressing the keys shown below.

Press	Display
LBL **E**	001 21 15
x^2	002 53
x⇄y	003 -41
x^2	004 53
+	005 -55
$\sqrt{x}$	006 54
RTN	007 24

With the program loaded into the HP-97, you can run the program. For example, calculate the hypotenuse of a right triangle with side *a* of 22 meters and side *b* of 9 meters.

Before you can run the program, you must *initialize* it.

Initializing a Program

Initialization of a program means nothing more than setting up the program (providing inputs, setting display mode, etc.) prior to the actual running of it. Some programs contain initialization routines that set up the data to run the program. In other programs, you may have to initialize manually from the keyboard before running. In the case of the program for calculating the hypotenuse of a triangle, to initialize the program you must place the values for sides *a* and *b* in stack registers X and Y. (Notice that the *order* does not matter in this case.) Thus, to initialize this program:

First, slide the Program Mode switch PRGM ▮▯ RUN to RUN.

Press	Display
22 **ENTER↑**	22.00
9	9.

The program for hypotenuse of a right triangle using the Pythagorean Theorem is now initialized for sides of 22 and 9 meters.

Running the Program

To run the program you have only to press the user-definable key that selects this program.

Press	Display	
E	23.77	Length of side *c* in meters.

To compute the hypotenuse of a right triangle with a side *a* of 73 miles and a side *b* of 99 miles:

Press	Display	
73 ENTER↑	73.00	
99	99.	Program initialized for new set of data before running.
E	123.00	Length of side *c* in miles.

Now let's see how we can use the nonrecordable editing features of the HP-97 to examine and alter this program.

Resetting to Step 000

As you know, when you press f CLPRGM with the Program Mode switch set to PRGM, the calculator is reset to step 000 and all instructions in program memory of the HP-97 are erased and replaced with R/S instructions. However, you can reset the HP-97 to step 000 of program memory while *preserving* existing programs in program memory by pressing GTO ⊡ 000 in PRGM or RUN mode, or RTN in RUN mode.

To set the calculator to step 000 with the Pythagorean Theorem program loaded into program memory:

Press	Display	
GTO ⊡ 000	123.00	Length of side *c* remains in display from previous running of program.

You could also have pressed RTN in RUN mode to set the calculator to step 000. Slide the Program Mode switch PRGM ▮ RUN to PRGM to verify that the calculator is now set at step 000 of program memory.

Display
000

Single-Step Execution of a Program

With the Program Mode switch set to RUN, you can execute a recorded program one step at a time by pressing the SST *(single-step)* key.

To single-step through the Pythagorean Theorem program using a triangle with side *a* of 73 miles and side *b* of 99 miles:

First slide the Program Mode switch PRGM ▮▯ RUN to RUN.

Press	Display	
73 ENTER↑	73.00	
99	99.	Program initialized for this set of data before running.

Now, press SST and hold it down to see the keycode for the next instruction. When you release the SST key, the next instruction is executed.

Press	Display	
SST	001 21 15	Keycode for LBL E seen when you hold SST down.
	99.00	LBL E executed when you release SST.

The first instruction of the program has executed when you pressed and release SST. (Notice that you didn't have to press E—when you are executing a program one step at a time, pressing the SST key begins the program without the need to press the user-definable E key.)

Continue executing the program by pressing SST again. When you hold SST down, you see the keycode for the next instruction. When you release SST, that instruction is executed.

Press	Display	
SST	002 53	Keycode for x^2.
	9801.00	Executed.

When you press SST a third time in RUN mode, step 003 of program memory is displayed. When you release the SST key, the instruction in that step, x≷y, is executed, and the calculator halts.

Press	Display	
SST	003 -14	Keycode for x≷y.
	73.00	Executed.

Continue executing the program by means of the SST key. When you have executed the RTN instruction in step 007, you have completed executing the program and the answer is displayed, just as if the calculator had executed the program automatically, instead of via the SST key.

Press	**Display**
SST	004 53 5329.00
SST	005 -55 15130.00
SST	006 54 123.00
SST	007 24 123.00

You have seen how the SST key can be used in RUN mode to single-step through a program. Using the SST key in this manner can help you create and correct programs. Now let's see how you can use SST, BST, and GTO . n n n in PRGM mode to help you modify a program.

Modifying a Program

Since you have completed execution of the above program, the HP-97 is set at step 008. You can verify that the calculator is set at this step by sliding the Program Mode switch PRGM RUN to PRGM and observing the step number and keycode in the display.

Now let's modify this Pythagorean Theorem program so that it will *print* the values of sides *a* and *b* and the calculated value of the hypotenuse, side *c*. First, print the program by going to step 000 and pressing f PRINT: PRGM so that you can see where instructions will have to be inserted to modify the program as required.

Press	**Display**	
RTN	123.00	In RUN mode, pressing RTN from the keyboard moves the calculator to step 000 of program memory.
f PRINT: PRGM	123.00	You want to insert PRINTx instructions after each of these instructions.
		Printer lists program instructions until it encounters second R/S in a row.

```
001  *LBLE  21 15
002     X²     53
003    X⇄Y    -41
004     X²     53
005      +    -55
006     √X     54
007    RTN     24
008    R/S     51
```

To begin modification of the loaded program, again reset the calculator to step 000 of program memory without erasing the program:

Press	Display	
GTO ⊡ 000	123.00	Calculator reset to step 000 of program memory.

Single-Step Viewing without Execution

You can use the SST key in PRGM mode to single-step to the desired step of program memory *without* executing the program. When you slide the Program Mode switch to PRGM, you should see that the calculator is reset to step 000 of program memory. When you press SST once, the calculator moves to step 001 and displays the contents of that step of program memory. No instructions are executed.

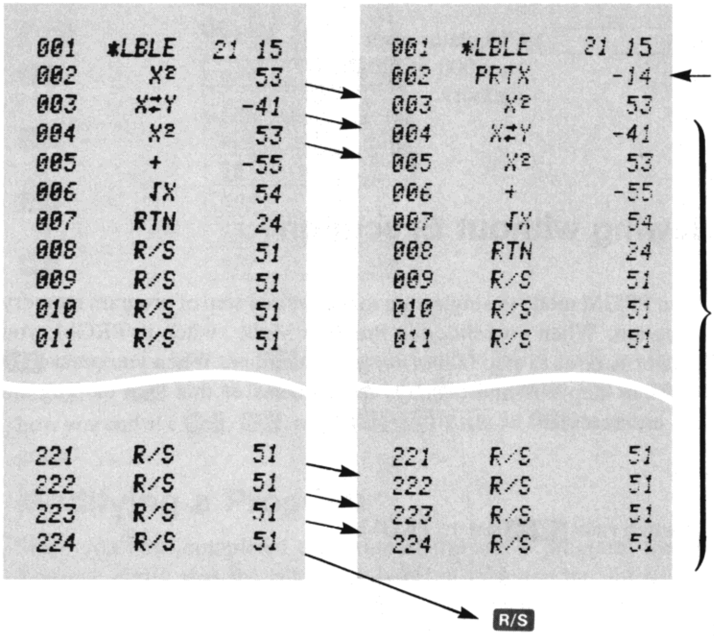
Slide the PRGM-RUN switch PRGM ▮ RUN to PRGM.

Press	Display	
	000	Step 000 of program memory displayed.
SST	001 21 15	Calculator moves to step 001 without executing instructions.

You can see that the calculator is now set at step 001 of program memory. If you press a recordable operation now, it will be loaded in the *next* step, step 002, of program memory, and all subsequent instructions will be "bumped" down one step in program memory. Thus, to load the PRINT x instruction so that the calculator will print the value of side *b* during execution:

Press	Display
PRINT x	002 -14

Now let's see what happened in program memory when you loaded that PRINTx instruction. With the calculator set at step 001, when you pressed PRINTx, program memory was altered…



… from this …

```
001  *LBLE  21 15
002     X²     53
003    X⇄Y    -41
004     X²     53
005      +    -55
006     √X     54
007    RTN     24
008    R/S     51
009    R/S     51
010    R/S     51
011    R/S     51

221    R/S     51
222    R/S     51
223    R/S     51
224    R/S     51
```

… to this.

```
001  *LBLE  21 15
002   PRTX    -14
003     X²     53
004    X⇄Y    -41
005     X²     53
006      +    -55
007     √X     54
008    RTN     24
009    R/S     51
010    R/S     51
011    R/S     51

221    R/S     51
222    R/S     51
223    R/S     51
224    R/S     51
```

PRINTx instruction inserted here.

All subsequent instructions are "bumped" down one step of program memory.

One instruction lost here.

You can see that when you insert an instruction in a program, all instructions after the inserted one are moved down one step of program memory, and the instruction formerly loaded in step 224 is lost and cannot be recovered. In this case, the last instruction was a R/S instruction and was not used in the program. Note, however, that if you inserted an instruction into program memory when step 224 contained an instruction used in a program, that instruction, too, would be lost. You should always view the contents of the last few steps of program memory before adding instructions to a program to ensure that no vital instructions will be lost from there.

Going to a Step Number

It is easy to see that if you wanted to single-step from step 000 to some remote step number in program memory, it would take a great deal of time and a number of presses of the SST key. So the HP-97 gives you another nonrecordable operation, GTO • n n n, that permits you to go to *any* step number of program memory.

Whether the Program Mode switch is set to PRGM or to RUN, when you press GTO • n n n, the calculator immediately jumps to the program memory step number specified by the three-digit number n n n. No instructions are executed. In RUN mode, you can momentarily slide the Program Mode switch to PRGM to view this program information, while if the calculator is already in PRGM mode, the step number and keycode for the instruction contained in that step are displayed. Program printing, searching, or execution then will begin with that step of program memory. Loading will begin with the next step of program memory.

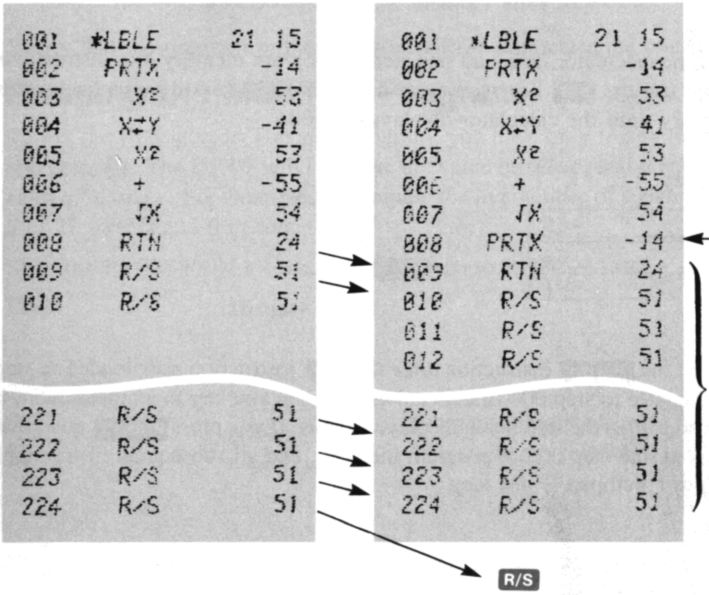
For example, to add a PRINTx instruction to print the value for the hypotenuse that is now calculated by the instruction in step 007, you can first press GTO *(go to)* followed by a decimal point and the appropriate three-digit step number of program memory. Then press PRINTx to place that instruction in the *following* step of program memory. Remember that when you add an instruction in this manner, each subsequent instruction is moved down one step in program memory, and the last instruction is lost. To add a PRINTx instruction after the √x instruction that is now loaded into step 007:

Press	**Display**
GTO · 007	007 54
PRINTx	008 -14

As you load the PRINTx instruction into step 008, the instruction that was *formerly* in step 008 is moved to step 009, and the instructions in subsequent steps are similarly moved down one step. The R/S instruction in step 224 is lost from program memory.

When you added the PRINTx instruction after step 007, program memory was altered...

... from this ... **... to this.**



```
001  *LBLE    21 15
002   PRTX      -14
003     X²       53
004    X⇄Y      -41
005     X²       53
006      +      -55
007     √X       54
008    RTN       24
009    R/S       51
010    R/S       51

221    R/S       51
222    R/S       51
223    R/S       51
224    R/S       51
```

```
001  *LBLE    21 15
002   PRTX      -14
003     X²       53
004    X⇄Y      -41
005     X²       53
006      +      -55
007     √X       54
008   PRTX      -14
009    RTN       24
010    R/S       51
011    R/S       51
012    R/S       51

221    R/S       51
222    R/S       51
223    R/S       51
224    R/S       51
```

PRINTx instruction inserted here.

All subsequent instructions are moved down one step of program memory.

One instruction lost here.

Stepping Backward through a Program

The BST (*back step*) key allows you to back step through a loaded program for editing whether the calculator is in RUN or PRGM mode. When you press BST, the calculator backs up one step in program memory—if the calculator is in RUN mode, the calculator moves to and displays the contents of the previous step of program memory when you hold the BST key depressed, then displays the original contents of the X-register when released. If the calculator is in PRGM mode, you see only the step number and keycode of the previous step.

You now have one more PRINT x instruction to add to the Pythagorean Theorem program. The PRINT x instruction should be added after the x⇄y instruction, keycode –41, that is now loaded in step 004 of program memory. If you have just completed loading a PRINT x instruction in step 008 as described above, the calculator is set at step 008 of program memory. You can use the BST key to back the calculator up to step 004, then insert the PRINT x instruction in step 005. To begin:

Ensure that the Program Mode switch PRGM ▯■ RUN is set to PRGM.

Press	**Display**	
	008 –14	Calculator initially set to step 008.
BST	007 54	Pressing BST once moves the calculator back one step in program memory.

When you press BST, the calculator backs up one step in program memory. No instructions are executed when you use the BST key. Continue using the BST key to move backwards through program memory until the calculator displays step 004.

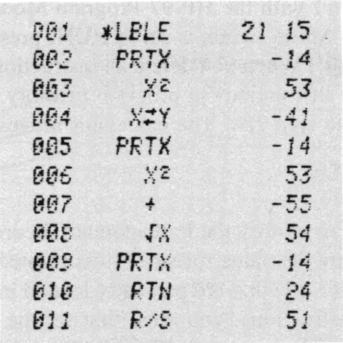
Press	**Display**
BST	006 –55
BST	005 53
BST	004 –41

Since you wish to insert the PRINT x instruction *after* the x⇄y instruction now loaded in step 004, you move the calculator to step 004 first. As always, when you key in an instruction, it is loaded into the next step *after* the step being displayed. Thus, if you press PRINT x now, that instruction will be loaded into step 005 of program memory, and all subsequent instructions will be moved down, or "bumped", one step.

Press	**Display**
PRINT x	005 –14

Verifying Program Changes

You have now finished modifying the Pythagorean Theorem program to print the values of all three sides when it is run. The altered program is shown below:



```
001  *LBLE      21 15
002   PRTX        -14
003     X²         53
004    X⇄Y        -41
005   PRTX        -14
006     X²         53
007      +        -55
008     √X         54
009   PRTX        -14
010    RTN         24
011    R/S         51
```

You can verify that the program in your HP-97 is the same as the one shown by setting the calculator to step 000 and printing the program. To print the modified Pythagorean Theorem program now loaded in your calculator:

Press	Display	
GTO · 000	000	Returns calculator to step 000.
f PRINT: PRGM		Prints contents of program memory. Halts printing of program memory after step 011.

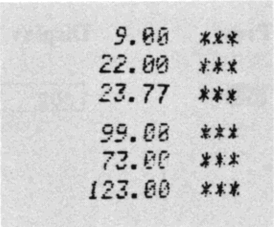
Verify that the program printed by your HP-97 duplicates the one shown above.

Running the Modified Program

To run the Pythagorean Theorem program, you have only to key in values for sides *a* and *b* and press E. The HP-97 will calculate the value of side *c*, and it should now print values for sides *a, b,* and *c*. For example, to compute the hypotenuse of a right triangle with sides *a* and *b* of 22 meters and 9 meters:

Slide the PRGM-RUN switch PRGM ■▯ RUN to RUN.

Press	Display	
22 ENTER↑	22.00	
9	9.	Program initialized.
E	23.77	The answer in meters.



```
  9.00  ***
 22.00  ***
 23.77  ***

 99.00  ***
 73.00  ***
123.00  ***
```

Now run the program for a right triangle with sides *a* and *b* of 73 miles and 99 miles.

(Answer: 123 miles.)

Deleting an Instruction

Often in the modification of a program you may wish to delete an instruction from program memory. To delete the instruction to which the calculator is set, merely press the nonrecordable operation [f] [DEL] *(delete)* with the HP-97 Program Mode switch PRGM ▮ RUN set to PRGM. (When the Program Mode switch is set to RUN, pressing [DEL] does nothing except cancel a pressed prefix key [f].) When you delete an instruction from program memory using the [DEL] key, all subsequent instructions in program memory are moved *up* one step, and a [R/S] instruction is loaded into step 224. The calculator moves to the step *before* the deleted step and displays it.

For example, if you wanted to modify the Pythagorean Theorem program that is now loaded into the calculator so that *only* the value for side *c* was printed, you would have to delete the [PRINTx] instructions, keycode –14, that are presently loaded in steps 002 and 005 of program memory. To delete these instructions, you must first set the calculator at these steps using [SST], [BST] or [GTO] [•] [n] [n] [n], then press [f] [DEL]. To delete the [PRINTx] instruction now loaded in step 002:

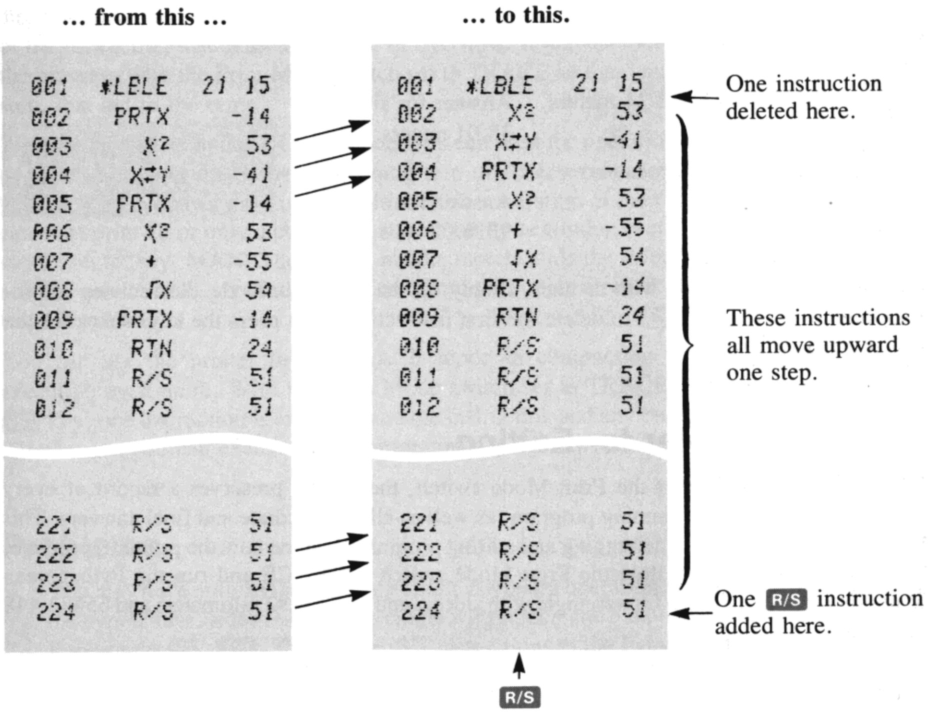
First, slide the Program Mode switch PRGM ▮ RUN to PRGM.

Press	Display	
[GTO] .002	002 -14	Step 002 is displayed.
[f] [DEL]	001 21 15	The instruction in step 002 is deleted and the calculator moves to step 001.

You can use the [SST] key to verify that the [PRINTx] instruction, keycode –14, has been deleted, and subsequent instructions have been moved up one step.

Press	Display	
[SST]	002 53	The instruction formerly in step 003 was moved up to step 002, and all subsequent instructions were moved up one step, when you pressed [f] [DEL].

When you set the calculator to step 002 of program memory and pressed f DEL, program memory was altered…



… from this …

```
001  *LBLE  21 15
002   PRTX    -14
003     X²     53
004    X⇄Y    -41
005   PRTX    -14
006     X²     53
007      +    -55
008     √X     54
009   PRTX    -14
010    RTN     24
011    R/S     51
012    R/S     51

221    R/S     51
222    R/S     51
223    R/S     51
224    R/S     51
```

… to this.

```
001  *LBLE  21 15
002     X²     53
003    X⇄Y    -41
004   PRTX    -14
005     X²     53
006      +    -55
007     √X     54
008   PRTX    -14
009    RTN     24
010    R/S     51
011    R/S     51
012    R/S     51

221    R/S     51
222    R/S     51
223    R/S     51
224    R/S     51
```

One instruction deleted here.

These instructions all move upward one step.

One R/S instruction added here.

R/S

To delete the PRINTx instruction now loaded in step 004 you can use the SST key to single-step down to that step number and then delete the instruction with the f DEL operation.

Press	Display	
SST	003 -41	
SST	004 -14	
f DEL	003 -41	The PRINTx instruction, keycode –14, is deleted from step 004 and the calculator displays step 003. Subsequent instructions move up one step of program memory.

If you have modified the program as described above, the HP-97 should now print only the computed value for the hypotenuse, side *c*, of a right triangle when sides *a* and *b* are input to the X- and Y-registers of the stack and the Pythagorean Theorem program is run.

Slide the Program Mode switch **PRGM** **RUN** to RUN, and run the program for right triangles with:

Sides *a* and *b* of 17 and 34 meters.	Answer for side *c* is 38.01 meters.
Sides *a* and *b* of 5500 rods and 7395 rods.	Answer for side *c* is 9216.07 rods.

```
   38.01  ***
 9216.07  ***
```

To replace any instruction with another, simply set the calculator to the desired step of program memory, press **f** **DEL** to delete the first instruction, then press the keystrokes for the new instruction.

Using the Printer for Editing

In the TRACE position of the Print Mode switch, the printer preserves a record of every instruction executed in a running program, as well as all intermediate and final answers. This feature is a valuable aid in debugging and editing programs. To see how the printer reproduces the action of a program, slide the Print Mode switch to TRACE and run the Pythagorean Theorem program to solve for a triangle with sides *a* and *b* of 11282 kilometers and 65482.448 kilometers:

Slide the Print Mode switch **MAN** **TRACE** **NORM** to TRACE.

Press	Display	
11282 **ENTER↑**	11282.00	
65482.448	65482.448	
E	66447.23	Kilometers.

```
   11282.00 ENT↑
  65482.448 GSBE
        001  *LBLE
        002     X²
 4287950996.   ***
        003    X⇄Y
   11282.00    ***
        004     X²
 127283524.0   ***
        005      +
 4415234520.   ***
        006     √X
   66447.23    ***
        007   PRTX
   66447.23    ***
        008    RTN
```

The printer shows every keystroke, every instruction executed, and, where calculated, every intermediate and final result.

When a program halts in the middle of execution because of an error or because of an overflow, you can slide the PRGM-RUN switch to PRGM to see the step number and keycode of the instruction that caused the error or overflow. It may be more helpful, however, to run the program with the Print Mode switch set to TRACE so that you chart the events, step-by-step, that led to the error.

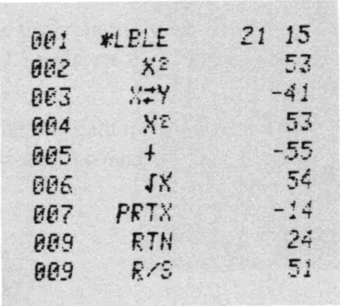
With the printer set to the TRACE mode, you can print the operation of the entire program, or by first addressing the desired beginning step of program memory with GTO • n n n, SST, or BST, you can print only a portion of the operation of the program if you desire. You can even print all or only a portion of a program by beginning execution with the Print Mode switch set to, say, MAN. Then in the middle merely slide the Print Mode switch to TRACE, and the printer will record all operations from that point onward. There is no need even to stop execution.

You can use the printer in the TRACE mode in conjunction with SST to slow down execution even more. With the Print Mode switch set to TRACE, each time you press the SST key, one instruction is executed and the instruction and any results are also printed. With this feature, you can examine your programs step-by-step with a fine-toothed comb!

Problems

1. You may have noticed that there is a single keyboard operation, the →P key, that calculates the hypotenuse, side *c*, of a right triangle with sides *a* and *b* input to the X- and Y-registers. Replace the x², x⇄y, x², +, and √x instructions in the Pythagorean Theorem program with the single →P instruction, as follows:

   a. Use the GTO • n n n and f PRINT: PRGM keys to verify that the Pythagorean Theorem program in your HP-97 contains the instructions shown below.



```
001  *LBLE      21 15
002     X²         53
003    X⇄Y        -41
004     X²         53
005      +        -55
006     √X         54
007   PRTX        -14
009    RTN         24
009    R/S         51
```

Replace all of these instructions with a →P instruction.

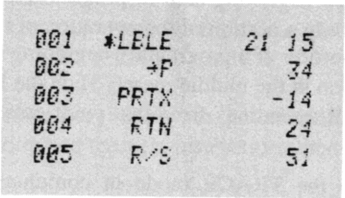
   b. Use the GTO • n n n keyboard operation to go to step 006, the last instruction to be deleted in the program.

c. Use the DEL keyboard operation in PRGM mode to delete the instructions in steps 006, 005, 004, 003, and 002.

> **Note:** When modifying a program, you should always *delete* instructions before you *add* others, to ensure that no vital instructions are "bumped" from the bottom of program memory and lost.

d. Load the →P instruction into step 002.

e. Verify that the modified program looks like the one below.

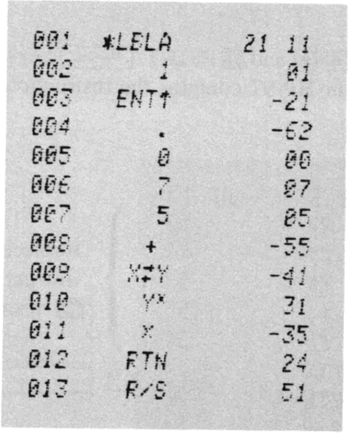


```
001  *LBLE   21 15
002     →P      34
003   PRTX     -14
004    RTN      24
005    R/S      51
```

f. Switch to RUN mode and run the program for a right triangle with sides *a* and *b* of 73 feet and 112 feet.

(Answer: 133.69 feet)

2. The following program is used by the manager of a savings and loan company to compute the future amounts of savings accounts according to the formula FV = PV $(1 + i)^n$, where FV is future value or amount, PV is present value, *i* is the periodic interest rate expressed as a decimal, and *n* is the number of periods. With PV entered into the Y-register, *n* keyed into the X-register, and an annual standard interest rate of 7.5%, the program is:



```
001  *LBLA   21 11
002      1      01
003   ENT↑     -21
004      .     -62
005      0      00
006      7      07
007      5      05
008      +     -55
009    X⇄Y     -41
010     Yˣ      31
011      ×     -35
012    RTN      24
013    R/S      51
```

a. Load the program into the calculator.

b. Run the program to find the future amount of $1,000 invested for 5 years.

(Answer: $1,435.63)

Of $2,300 invested for 4 years. (Answer: $3,071.58)

c. Alter the program to account for a change of the annual interest rate from 7.5% to 8%, and add a **PRINTx** instruction so that the answer (future value) is printed.

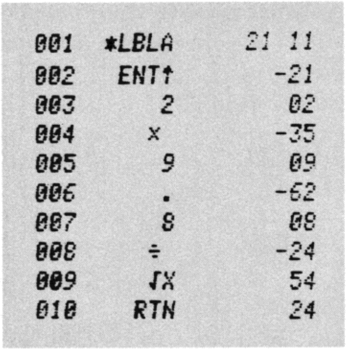
d. Run the program for the new interest rate to find the future value of $500 invested for 4 years; of $2,000 invested for 10 years.

(Answers: $680.24; $4,317.85)

3. The following program calculates the time it takes for an object to fall to the earth when dropped from a given height. (Friction from the air is not taken into account.) When the program is initialized by keying the height *h* in meters into the displayed X-register and **A** is pressed, the time *t* in seconds the object takes to fall to earth is computed according to the formula

$$t = \sqrt{\frac{2d}{9.8 \text{ meters/sec}^2}}$$

a. Clear all previously recorded programs from the calculator and load the program below.



```
001  *LBLA     21 11
002   ENT↑       -21
003      2        02
004      x       -35
005      9        09
006      .       -62
007      8        08
008      ÷       -24
009     √X        54
010    RTN        24
```

b. Run the program to compute the time taken by a stone to fall from the top of the Eiffel Tower, 300.51 meters high. From a blimp stationed 1000 meters in the air.

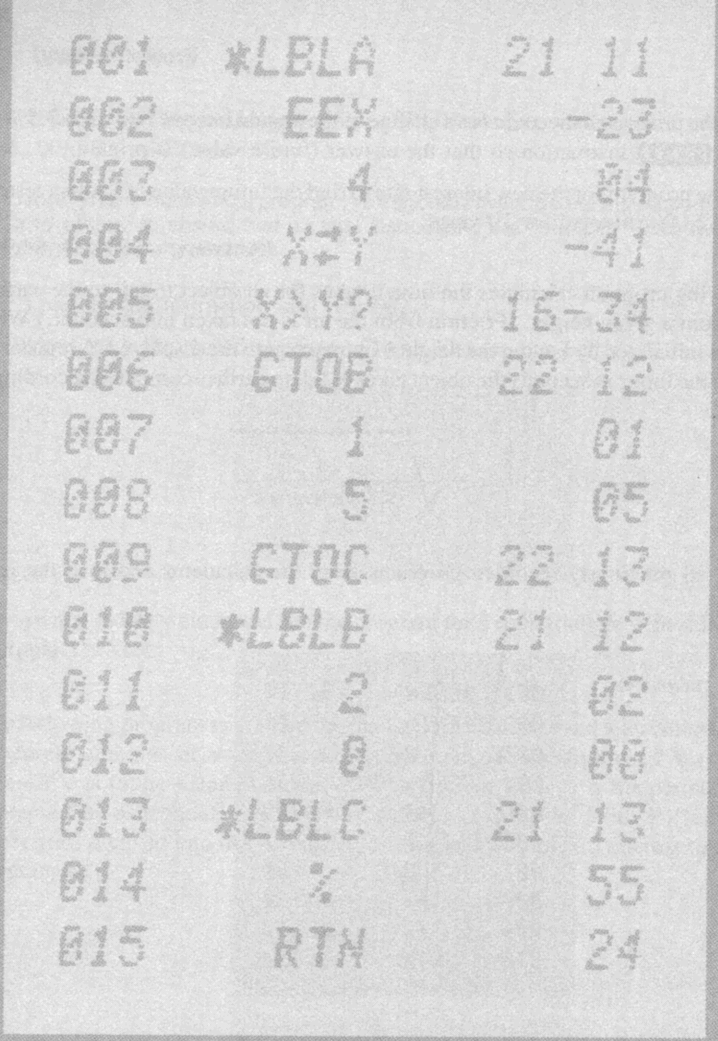
(Answers: 7.83 seconds, 14.29 seconds)

c. Alter the program to compute the time of descent when the height in *feet* is known, according to the formula

$$t = \sqrt{\frac{2d}{32.1740 \text{ feet/sec}^2}}$$

d. Run the altered program to compute the time taken by a stone to fall from the top of the Grand Coulee Dam, 550 feet high. From the 1350-foot height of the World Trade Center buildings in New York City.

(Answers: 5.85 seconds, 9.16 seconds)



```
001  *LBLA   21 11
002    EEX     -23
003      4      04
004    X⇄Y     -41
005    X>Y?  16-34
006   GTOB   22 12
007      1      01
008      5      05
009   GTOC   22 13
010  *LBLB   21 12
011      2      02
012      0      00
013  *LBLC   21 13
014      %      55
015    RTN      24
```

Section 8

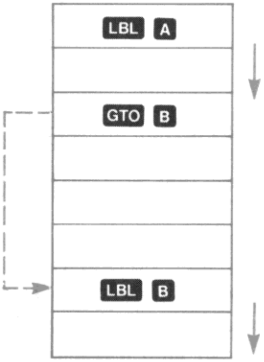
Branching

Unconditional Branching and Looping

You have seen how the nonloadable operation GTO · n n n can be used from the keyboard to transfer execution to any step number of program memory. You can also use the GTO *(go to)* instruction as part of a program, but in order for GTO to be *recorded* as an instruction, it must be followed by a label designator (A through E, f a through f e, or 0 through 9). It can also be followed by (i)—more about using (i) later.

When the calculator is executing a program and encounters a GTO B instruction, for example, it immediately halts execution and begins searching sequentially downward through program memory for that label. When the first LBL B instruction is then encountered, execution begins again.

By using a GTO instruction followed by a label designator in a program, you can transfer execution to any part of the program that you choose.



Execution branches to next LBL B.

A GTO instruction used this way is known as an *unconditional branch*. It always *branches* execution from the GTO instruction to the specified label. (Later, you will see how a conditional instruction can be used in conjunction with a GTO instruction to create a *conditional* branch—a branch that depends on the outcome of a test.)

A common use of a branch is to create a "loop" in a program. For example, the following program calculates and prints the square roots of consecutive whole numbers beginning with the number 1. The HP-97 continues to compute the square root of the next consecutive whole number until you press R/S to stop program execution (or until the calculator overflows).

To key in the program:

First, slide the Program Mode switch PRGM ▮▯ RUN to PRGM.

Press f CLPRGM to clear program memory and reset the calculator to step 000.

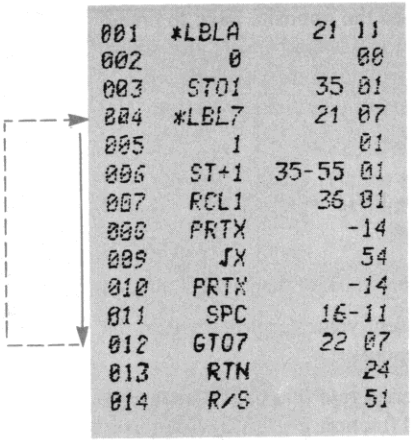
Press	Display
LBL A	001 21 11
0	002 00
STO 1	003 35 01
LBL 7	004 21 07
1	005 -01
STO + 1	006 35-55 01
RCL 1	007 36 01
PRINTx	008 -14
√x	009 54
PRINTx	010 -14
f PRINT: SPACE	011 16-11
GTO 7	012 22 07
RTN	013 24

To run the program, slide the Program Mode switch PRGM ▯▮ RUN to RUN and press A. The program will begin printing a table of integers and their square roots and will continue until you press R/S from the keyboard or until the calculator overflows.

How it works: When you press A, the calculator searches through program memory until it encounters the LBL A instruction that begins the program. It executes that instruction and each subsequent instruction in order until it reaches step 012, the GTO 7 instruction.

The GTO 7 instruction causes the calculator to *search* once again, this time for a LBL 7 instruction in the program. When it encounters the LBL 7 instruction loaded in step 004, execution begins again from that LBL 7. (Notice that the address after a GTO instruction in a program is a *label,* not a step number.)

Since execution is transferred to the LBL 7 instruction in step 004 each time the calculator executes the GTO 7 instruction in step 012, the calculator will remain in this "loop," continually adding one to the number in storage register R_1 and printing the new number and its square root.



```
001  *LBLA     21 11
002      0        00
003   STO1     35 01
004  *LBL7     21 07
005      1        01
006   ST+1  35-55 01
007   RCL1     36 01
008   PRTX       -14
009     √X        54
010   PRTX       -14
011    SPC     16-11
012   GTO7     22 07
013    RTN        24
014    R/S        51
```

Looping techniques like the one illustrated here are common and extraordinarily useful in programming. By using loops, you take advantage of one of the most powerful features of the HP-97—the ability to update data and perform calculations automatically, quickly, and, if you so desire, endlessly.

You can use unconditional branches to create a loop, as shown above, or in any part of a program where you wish to transfer execution to another label. When the calculator executes a **GTO** instruction, it searches sequentially downward through program memory and begins execution again at the first specified label it encounters.

Problems

1. The following program calculates and prints the square of the number 1 each time it is run. Key the program in with the PRGM-RUN switch **PRGM RUN** set to PRGM, then switch to RUN and run the program a few times to see how it works. Finally, modify the program by inserting a **LBL** **D** instruction after the **STO** 1 instruction in step 003, and a **GTO** **D** instruction after the **f** PRINT: **SPACE** instruction. This should create a loop that will continually print a new number and its square, then increment the number by 1, print it, and compute and print *its* square, etc. To load the original program, before modification, slide the PRGM-RUN switch **PRGM RUN** to PRGM. Then:

Press	Display
LBL **B**	001 21 12
0	002 00
STO 1	003 35 01
1	004 01
STO **+** 1	005 35–55 01
RCL 1	006 36 01
PRINT x	007 –14
x^2	008 53
PRINT x	009 –14
f PRINT: **SPACE**	010 16–11
RTN	011 24

2. Use the flowchart on the opposite page to create a program that computes and prints the future value *(FV)* of a compound interest savings account in increments of one year, according to the formula:

$$FV = PV(1 + i)^n$$

where *FV* = future value of the savings account.

PV = present value (or principal) of the account.

i = interest rate (expressed as a decimal fraction; e.g., 6% is expressed as 0.06).

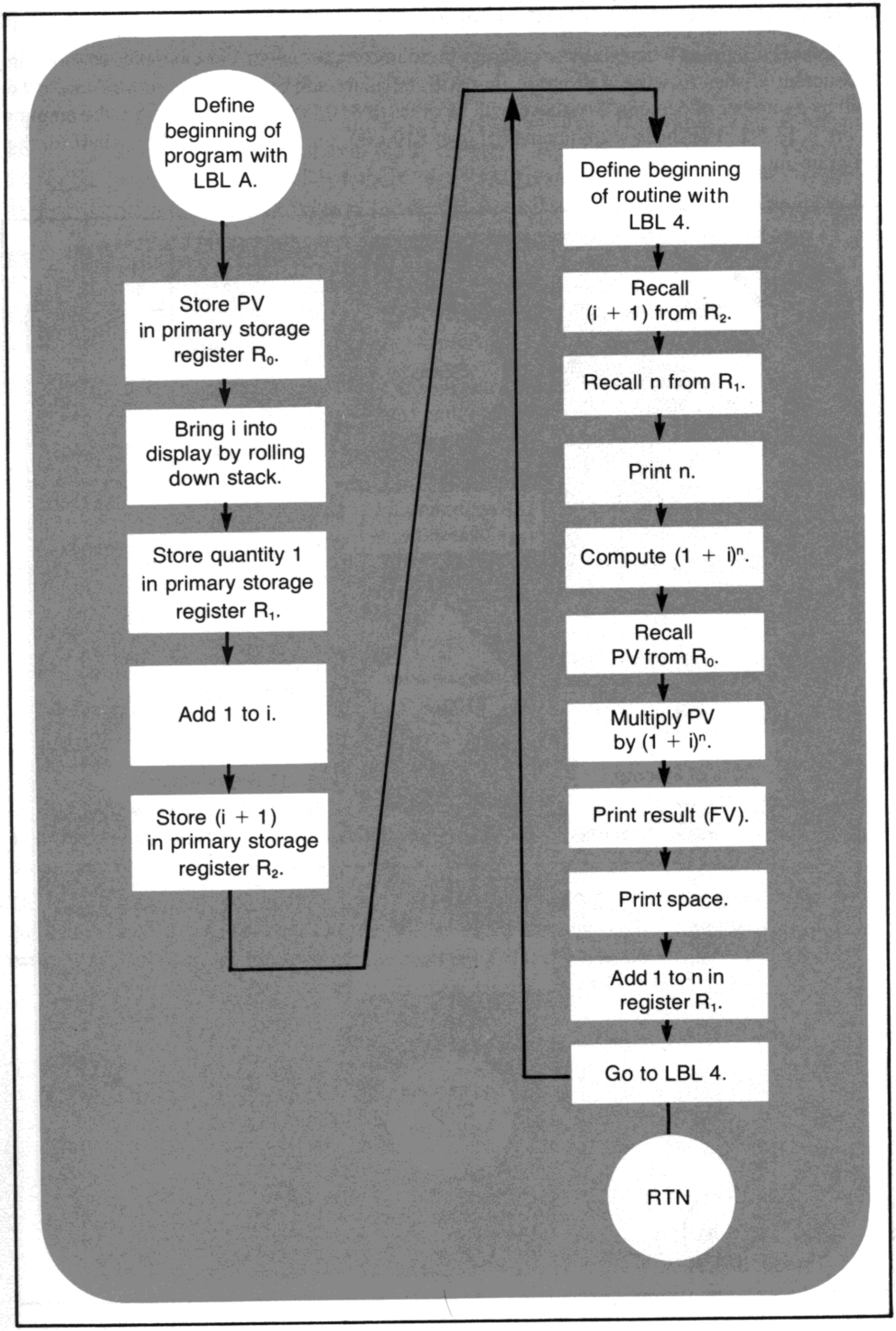
n = number of compounding periods (usually, years).

Assume that program execution will begin with *i* entered into the Y-register of the stack and with *PV* keyed into the displayed X-register.

After you have written and loaded the program, run it for an initial interest rate *i* of 6% (keyed in as .06) and an initial deposit (or present value, *PV)* of $1000.
(Answer: 1st year, $1060; 2nd year, $1123.60; 3rd year, $1191.02, etc.)

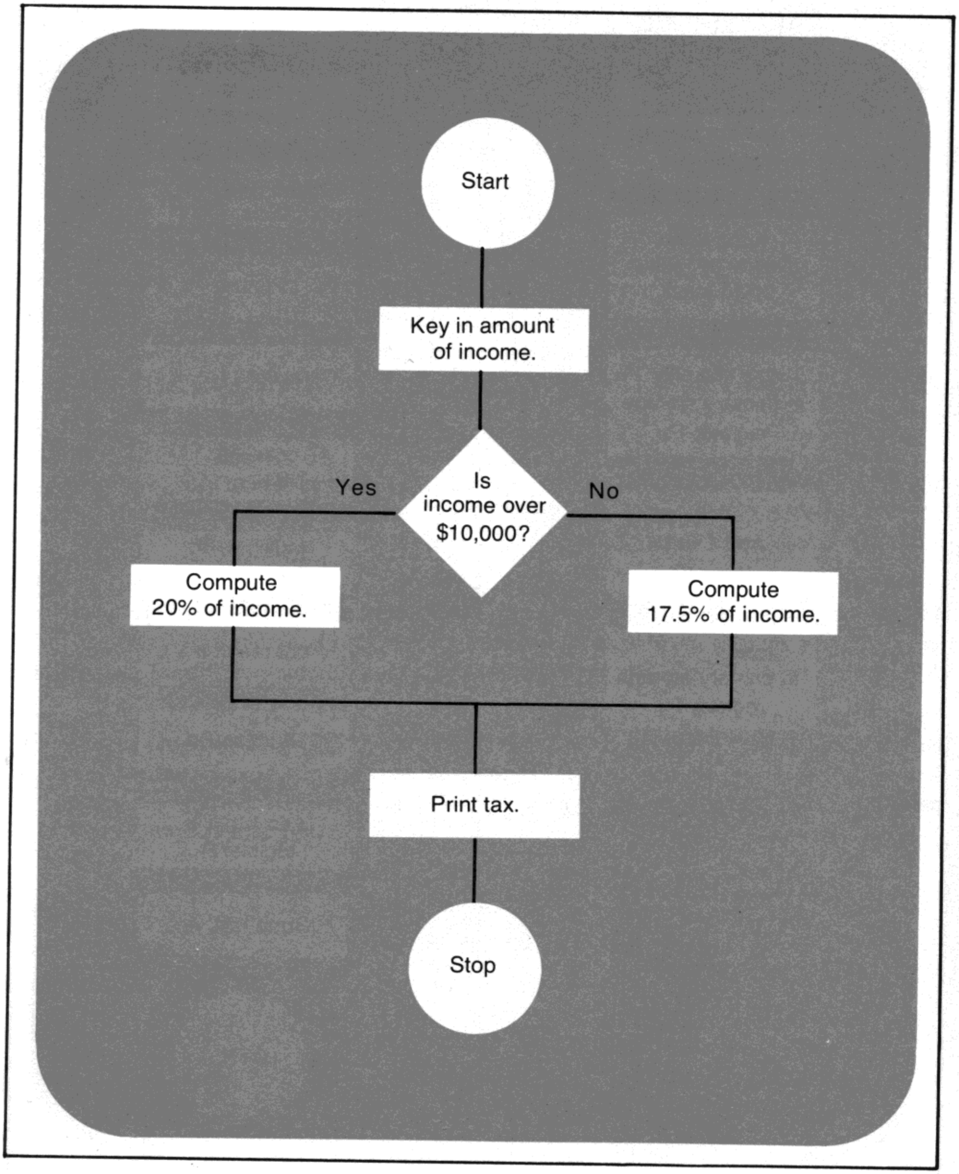
The program will continue running until you press **R/S** (or any key) or the HP-97 overflows. You can see how your savings would grow from year to year. Try the program for different interest rates *i* and values of *PV*.

3. Write a program using **GTO** that will use the factorial function (N!) to calculate and print the factorials of successive integers beginning with the number 1. (Hint: Place 1 in a storage register, recall it, then use storage register arithmetic to increment the number in the storage register, etc.)


Define beginning of program with LBL A.
Store PV in primary storage register R_0.
Bring i into display by rolling down stack.
Store quantity 1 in primary storage register R_1.
Add 1 to i.
Store (i + 1) in primary storage register R_2.
Define beginning of routine with LBL 4.
Recall (i + 1) from R_2.
Recall n from R_1.
Print n.
Compute $(1 + i)^n$.
Recall PV from R_0.
Multiply PV by $(1 + i)^n$.
Print result (FV).
Print space.
Add 1 to n in register R_1.
Go to LBL 4.
RTN

Conditionals and Conditional Branches

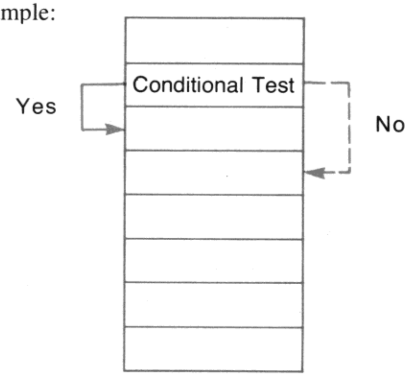
Often there are times when you want a program to make a decision. For example, suppose an accountant wishes to write a program that will calculate and print the amount of tax to be paid by a number of persons. For those with incomes of $10,000 per year or under, the amount of tax is 17.5%. For those with incomes of over $10,000, the tax is 20%. A flow chart for the program might look like this:



The *conditional* operations on your HP-97 keyboard are useful as program instructions to allow your calculator to make decisions like the one shown above. The eight conditionals that are available on your HP-97 are:

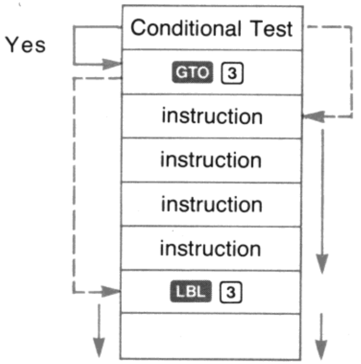
f x≠y?	tests to see if the value in the X-register is unequal to the value in the Y-register.
f x=y?	tests to see if the value in the X-register is equal to the value in the Y-register.
f x>y?	tests to see if the value in the X-register is greater than the value in the Y-register.
f x≤y?	tests to see if the value in the X-register is less than or equal to the value in the Y-register.
f x≠0?	tests to see if the value in the X-register is unequal to zero.
f x=0?	tests to see if the value in the X-register is equal to zero.
f x>0?	tests to see if the value in the X-register is greater than zero.
f x<0?	tests to see if the value in the X-register is less than zero.

Each conditional essentially asks a question when it is encountered as an instruction in a program. If the answer is YES, program execution continues sequentially downward with the next instruction in program memory. If the answer is NO, the calculator branches *around* the next instruction. For example:



You can see that after it has made the conditional test, the calculator will *do* the next instruction if the test is *true*. This is the "DO if TRUE" rule.

The step immediately following the conditional test can contain any instruction. The most commonly used instruction, of course, will be a GTO instruction. This will branch program execution to another section of program memory if the conditional test is true.



Now let's look at that accountant's problem again. For persons with incomes of more than $10,000 he wants to compute a tax of 20%. For persons with incomes of $10,000 or less, the tax is 17.5%. The following program will test the amount in the X-register and compute and print the correct percentage of tax.

To key in the program:

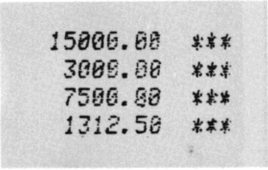
Slide the PRGM-RUN switch PRGM ▯■ RUN to PRGM.

Press	**Display**	
f CLPRGM	000	
LBL A	001 21 11	
PRINTx	002 -14	Prints amount of income.
EEX	003 -23	Amount of $10,000 placed in Y-register.
4	004 04	
x⇄y	005 -41	
f x>y?	006 16-34	If amount of income is greater than $10,000, go to portion of program defined by label B.
GTO B	007 22 12	
1	008 01	Tax percentage for this portion of program is 17.5.
7	009 07	
·	010 -62	
5	011 05	
GTO C	012 22 13	
LBL B	013 21 12	Tax percentage for this portion of program is 20.
2	014 02	
0	015 00	
LBL C	016 21 13	
%	017 55	
PRINTx	018 -14	
RTN	019 24	

To run the program to compute taxes on incomes of $15,000 and $7,500:

Slide the PRGM-RUN switch PRGM ■▥ RUN to RUN.

Press	Display
15000 A	3000.00
7500 A	1312.50

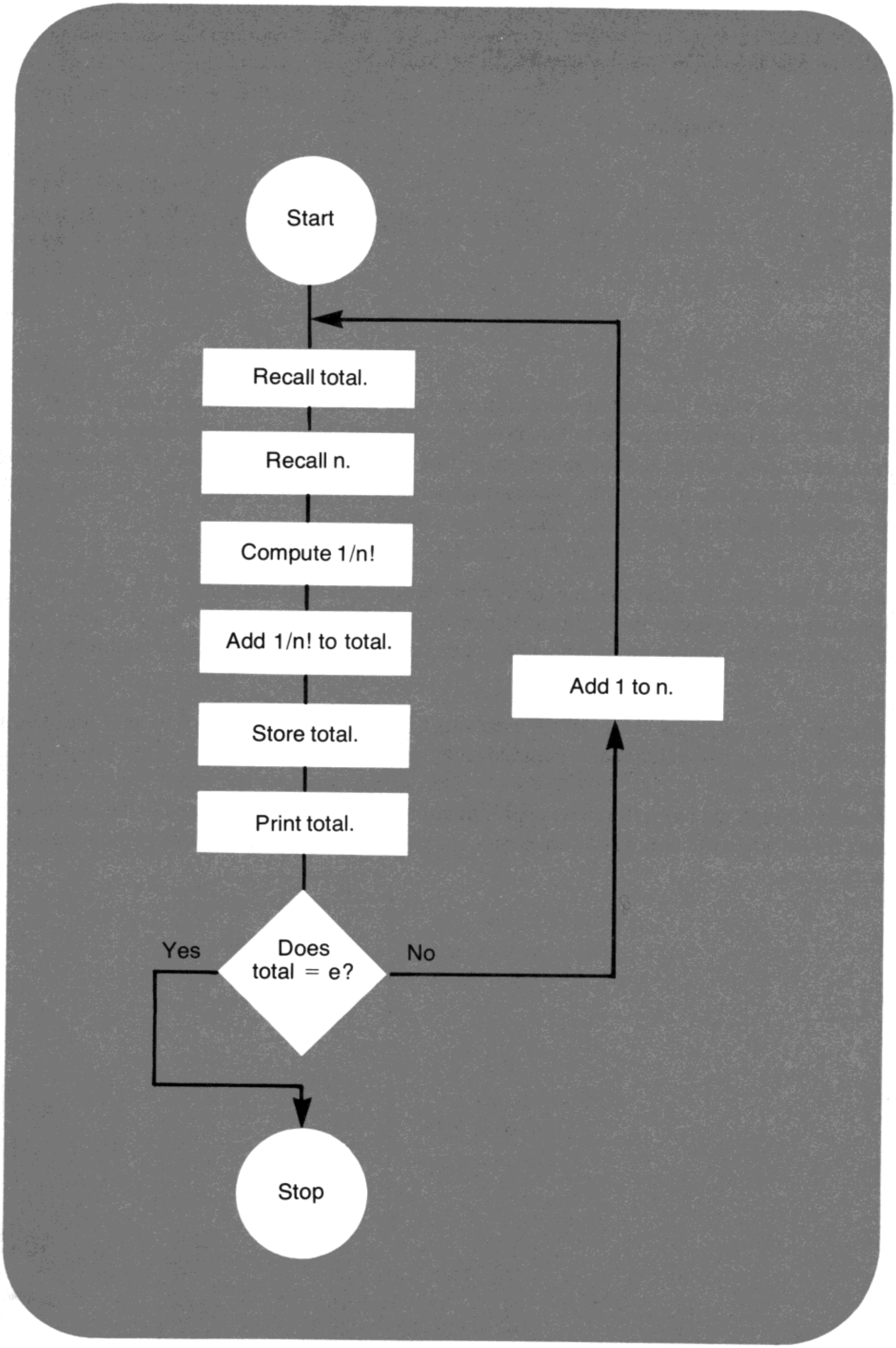


```
15000.00  ***
 3000.00  ***
 7500.00  ***
 1312.50  ***
```

Another place where you often want a program to make a decision is within a loop. The loops that you have seen have to this point been *infinite* loops—that is, once the calculator begins executing a loop, it remains locked in that loop, executing the same set of instructions over and over again, forever (or, more practically, until the calculator overflows or you halt the running program by pressing R/S or any other key).

You can use the decision-making power of the conditional instructions to shift program execution out of a loop. A conditional instruction can shift execution out of a loop after a specified number of iterations or when a certain value has been reached within the loop.

Example: As you know, your HP-97 contains a value for e, the base of natural logarithms. (You can display the calculator's value for e by pressing 1 e^x.) The following program uses the series $e = 1/0! + 1/1! + 1/2! + \ldots + 1/n!$ to *approximate* the value for e. After each iteration through the loop, the latest approximation is printed and compared to the *calculator's* value for e. When the two values are equal, the execution is transferred out of the loop to stop the program.


Start
Recall total.
Recall n.
Compute 1/n!
Add 1/n! to total.
Store total.
Print total.
Does total = e?
Yes
No
Add 1 to n.
Stop

To load the program into the calculator:

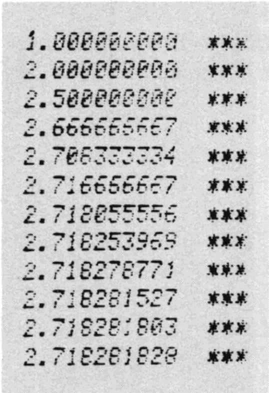
Slide the PRGM-RUN switch PRGM RUN to PRGM.

Press	**Display**
f CLPRGM	000
LBL A	001 21 11
RCL 1	002 36 01
RCL 0	003 36 00
f N!	004 16 52
1/x	005 52
+	006 -55
DSP 9	007 -63 09
STO 1	008 35 01
PRINT x	009 -14
1	010 01
e^x	011 33
f x=y?	012 16-33
GTO 7	013 22 07
1	014 01
STO + 0	015 35-55 00
GTO A	016 22 11
LBL 7	017 21 07
RTN	018 24

To initialize the program ensure that the primary storage registers are cleared to zero. Then press A to run the program.

First, slide the PRGM-RUN switch PRGM RUN to RUN.

Press	**Display**	
f CL REG	0.00	Ensures that storage registers are cleared to zero initially.
A	2.718281828	

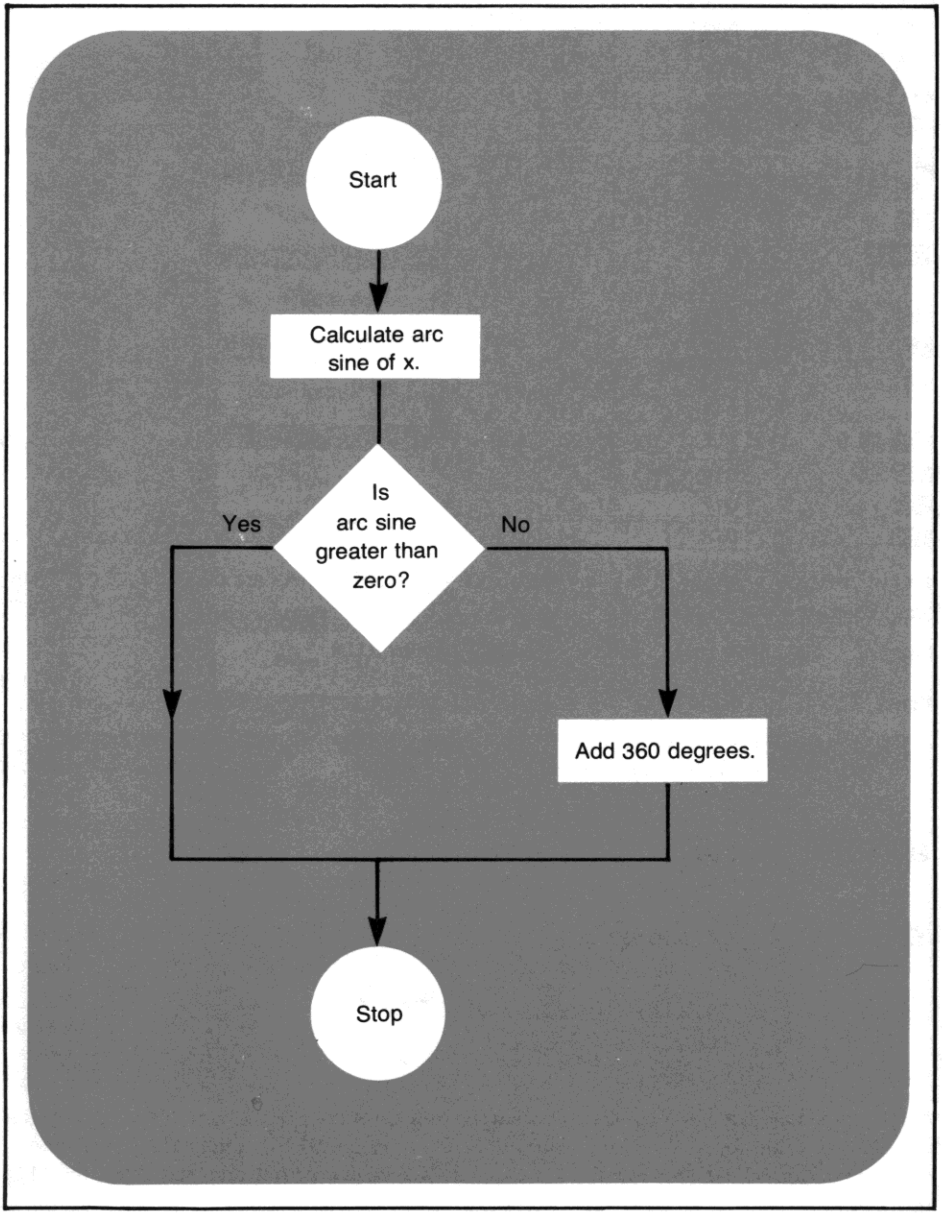


```
1.000000000  ***
2.000000000  ***
2.500000000  ***
2.666666667  ***
2.708333334  ***
2.716666667  ***
2.718055556  ***
2.718253969  ***
2.718278771  ***
2.718281527  ***
2.718281803  ***
2.718281828  ***
```

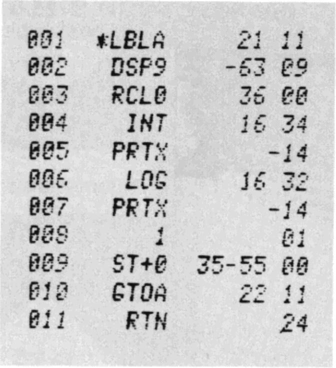
You can see that execution continues within the loop until the approximation for *e* equals the calculator's value for *e*. When the instruction x=y? in step 012 is finally true, execution is transferred out of the loop by the subsequent GTO 7 instruction and halted by the RTN instruction.

Problems

1. Write a program that will calculate the arc sine (that is, $\sin^{-1}$) of a value that has been keyed into the displayed X-register. Test the resulting angle with a conditional, and if it is negative or zero, add 360 degrees to it to make the angle positive. Use the flowchart below to help you write the program:



2. The program below contains a loop that prints a table of consecutive integers and their common logarithms. You can specify the *lowest* integer by storing a number in primary storage register R_0, but the program will continue printing until you press R/S or any other key from the keyboard, or until the calculator's capacity for display is exceeded.

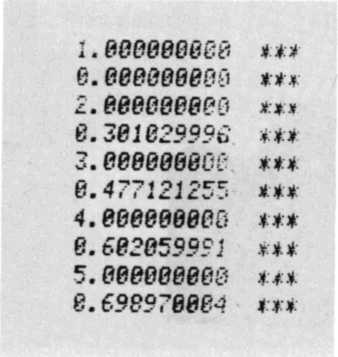


```
001  *LBLA      21 11
002   DSP9     -63 09
003   RCL0      36 00
004    INT      16 34
005   PRTX        -14
006    LOG      16 32
007   PRTX        -14
008      1         01
009   ST+0  35-55 00
010   GTOA      22 11
011    RTN         24
```

Using the additional instructions RCL 8, x>y?, GTO B and LBL B, you should be able to modify this program to halt execution when a certain number is reached. As you add these instructions, assume that the value for the upper limit has been stored in primary storage register R_8.

When the program is running and the value in register R_0 becomes greater than the limit you store in register R_8, program execution should be transferred out of the loop to the RTN instruction to halt the running program.

Modify the program, key it into the calculator, and initialize the calculator by storing a lower limit of 1 in register R_0 and an upper limit of 5 in register R_8. Then run the program. Your printed copy should look like the one below. Try other upper and lower limits. (The lower limit must always be greater than zero, and the upper limit should be greater than the lower limit.)

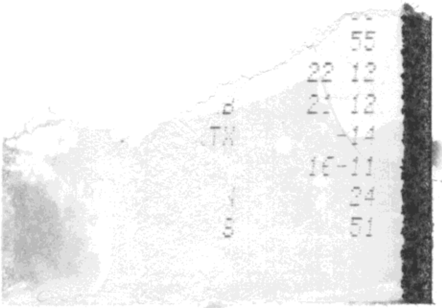


```
1.000000000  ***
0.000000000  ***
2.000000000  ***
0.301029996  ***
3.000000000  ***
0.477121255  ***
4.000000000  ***
0.602059991  ***
5.000000000  ***
0.698970004  ***
```

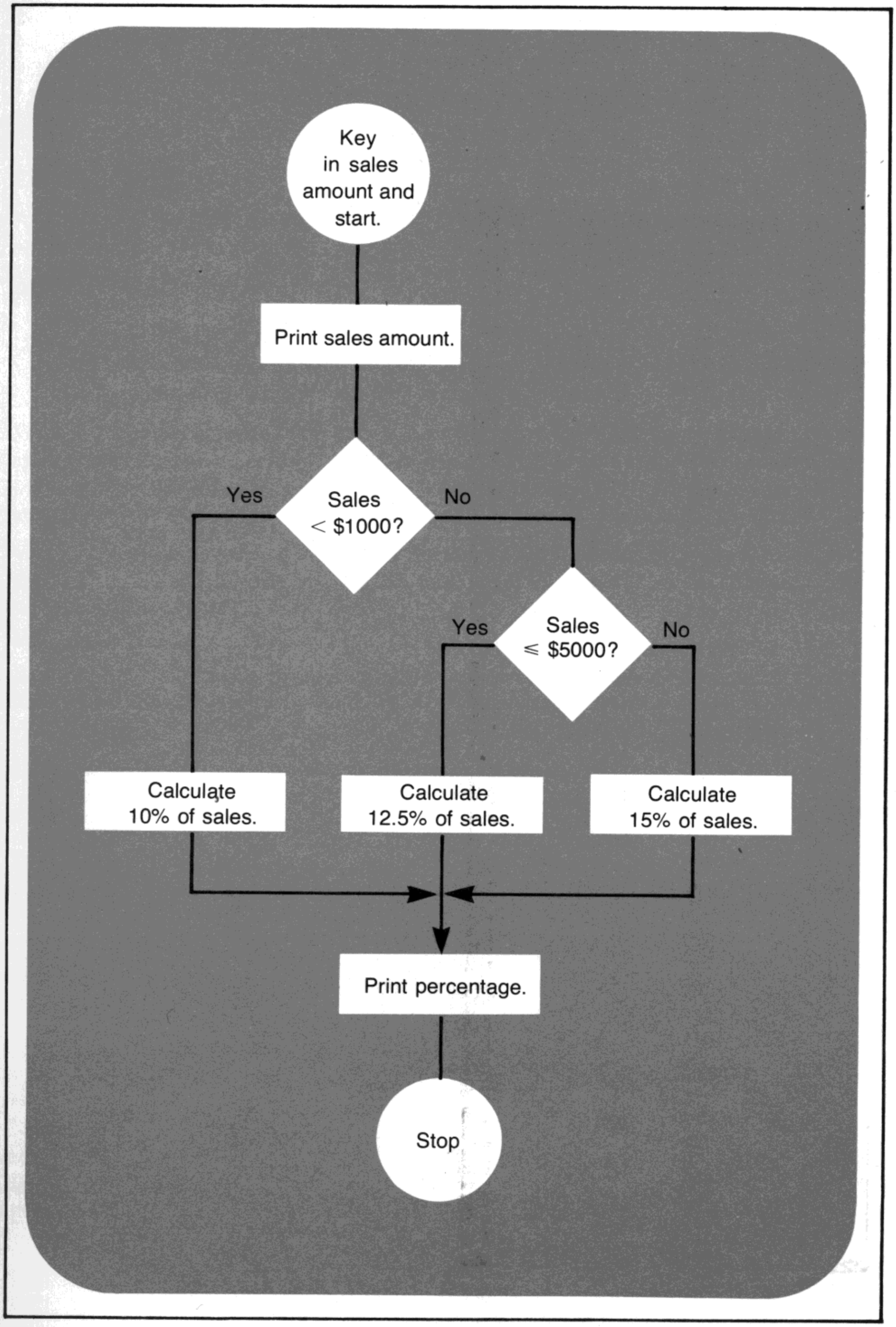
3. Use the flowchart on the opposite page to help you write a program that will allow a salesman to compute his commissions at the rates of 10% of sales of up to $1000, 12.5% for sales of $1000 to 5000, and 15% for sales of over $5000. The program should print the amount of sales and the amount of commission.

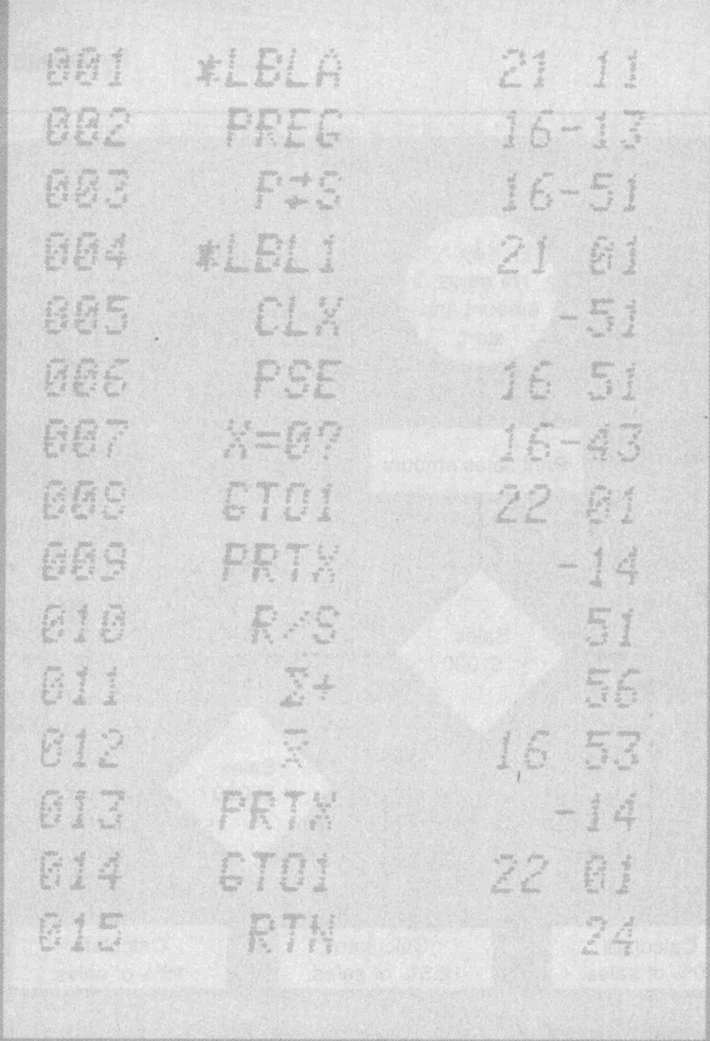
   Load the program and run it for sales amounts of $500, $1000, $1500, $5000, and $6000.
(Answers: $50.00, $125.00, $187.50, $625.00, $900.00)

```
21 11
35 01
  -41
16-75
```



```
     55
  22 13
B 21 12
TX   -14
  16-11
     34
5    51
```


Key in sales amount and start.
Print sales amount.
Yes
Sales < $1000?
No
Yes
Sales ≤ $5000?
No
Calculate 10% of sales.
Calculate 12.5% of sales.
Calculate 15% of sales.
Print percentage.
Stop



```
001  *LBLA     21 11
002   PREG     16-13
003    F⇄S     16-51
004  *LBL1     21 01
005    CLX       -51
006    PSE     16 51
007   X=0?     16-43
008   GT01     22 01
009   PRTX       -14
010    R/S        51
011     Σ+        56
012     x̄      16 53
013   PRTX       -14
014   GT01     22 01
015    RTN        24
```

Section 9

Interrupting Your Program

Using R/S

As you know, the R/S *(run/stop)* function can be used either as an instruction in a program or pressed from the keyboard.

When pressed from the keyboard:

1. If a program is running, R/S stops the program.
2. If a program is stopped or not running, and the calculator is in RUN mode, R/S starts the program running beginning with the current location in program memory.

When executed as an instruction during a running program, R/S stops program execution after its step of program memory. If R/S is then pressed from the keyboard, execution begins with the current step of program memory. (When R/S is pressed, it displays the step number and keycode of that current step—when released, execution begins with that step.)

You can use these features of the R/S instruction to stop a running program at points where you want to key in data. After the data has been keyed in, restart the program using the R/S key from the keyboard.

Example: The following program lets you key in a percentage discount and calculates the cumulative cost of various quantities of differently priced items from which the discount has been subtracted. R/S instructions are inserted in the program to allow you to key in data at various points.

Slide the PRGM-RUN switch PRGM ▮ RUN to PRGM.

Press	**Display**	
f CLPRGM	000	
LBL f a	001 21 16 11	
f CL REG	002 16-53	
STO 0	003 35 00	Store discount percentage in R_0.
LBL 9	004 21 09	
R/S	005 51	Stop to key in quantity.
ENTER↑	006 -21	
R/S	007 51	Stop to key in price.
×	008 -35	
RCL 0	009 36 00	
%	010 55	
−	011 -45	
STO + 1	012 35-55 01	Add to running total in R_1.
RCL 1	013 36 01	Recall running total for display.
GTO 9	014 22 09	
RTN	015 24	

Now run the program to calculate the cumulative total of the following purchases at a discount of 15%:

Quantity	Price of Each
5	$ 7.35
7	$12.99
14	$14.95

Then run the program to calculate the cumulative total of the following purchases at a discount of 25%:

Quantity	Price of Each
7	$ 4.99
12	$ 1.88
37	$ 8.50

In order to calculate the cumulative total for each percentage of discount, merely key in the percentage value and press **f** **a**. When the calculator stops executing the first time, key in the quantity of an item and press **R/S** from the keyboard to resume execution.

When the program stops a second time, key in the price of that item and again press **R/S** to resume program execution from that point.

To run the program:

Slide the PRGM-RUN switch PRGM ■▯ RUN to RUN.

Press	Display	
15	15.	Key in percentage of discount.
f **a**	15.00	
5 **R/S**	5.00	The first quantity.
7.35 **R/S**	31.24	Running total.
7 **R/S**	7.00	
12.99 **R/S**	108.53	Running total.
14 **R/S**	14.00	
14.95 **R/S**	286.43	Cumulative cost for items at 15% discount.
25	25.	Percentage of discount.
f **a**	25.00	
7 **R/S**	7.00	The first quantity.

Press	**Display**	
4.99 R/S	26.20	Running total.
12 R/S	12.00	
1.88 R/S	43.12	Running total.
37 R/S	37.00	
8.50 R/S	278.99	Cumulative cost for items at 25% discount.

If you have a number of halts for data entries like the ones shown here, it may be helpful to "identify" each step by recording a familiar number into the program immediately before each R/S instruction. When the calculator then stops execution because of the R/S instruction, you can look at the displayed X-register to see the "identification number" for the required data input at that point. For example, if your program contained eight stops for data inputs, it might be helpful to have the numbers 1 through 8 appear so that you would know which input was required each time. (Don't forget that the "identification number" will be pushed up into the Y-register of the stack when you key in a new number.)

Using Pause

Pausing to View Output

You now know two instructions that will slow or halt a running program for data output—PRINTx and R/S. PRINTx can print the value contained in the X-register at any point in the program, while R/S stops a running program and allows you to view results in the display.

Another instruction that can be used to slow a running program for data input or to view output is the PAUSE instruction. An f PAUSE instruction executed in a program momentarily interrupts program execution. The length of the pause is about one second, although more PAUSE instructions in subsequent steps of program memory can be used to lengthen viewing time, if desired.

You can use f PAUSE in a program to monitor the operation of the program without printing every result.

Example: The following program calculates an approximate value for pi according to the equation $\pi = \sqrt{\frac{6}{1^2} + \frac{6}{2^2} + \frac{6}{3^2} + \cdots \frac{6}{n^2}}$.

After each iteration of $\frac{6}{n^2}$, the calculator pauses to show the latest approximation and then compares it to the calculator's internal value for pi. When the approximation and the calculator's value for pi are equal, execution is transferred out of the loop and the approximated value for pi is printed, along with the number of iterations it took for the value to approximate π.

Note: In this program, the DSP 1 and RND instructions are used to round both pi and its approximation to one decimal place so that the approximation converges quickly.

To key in the program:

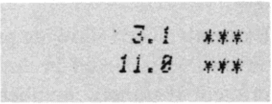
Slide the PRGM-RUN switch PRGM RUN to PRGM.

Press	Display		
f CL PRGM	000		
LBL A	001 21 11		
f CL REG	002 16-53		
FIX	003 -11		
DSP 1	004 -63 01		
LBL 1	005 21 01		Iterative loop
1	006 01		
STO + 2	007 35-55 02	Iteration number maintained in R_2.	
RCL 1	008 36 01		
RCL 2	009 36 02		
f PAUSE	010 16 51	Pause to display iteration number.	
x^2	011 53	Approximation for pi calculated.	
1/x	012 52		
6	013 06		
×	014 -35		
+	015 -55		
STO 1	016 35 01		
√x	017 54		
f RND	018 16 24	Approximation rounded to one decimal place.	
f PAUSE	019 16 51	Pause to display approximation.	
f π	020 16-24	Calculator's value for pi summoned, rounded, and displayed.	
f RND	021 16 24		
f PAUSE	022 16 51		
f x≠y?	023 16-32	If values are not equal, go to LBL 1.	
GTO 1	024 22 01		
x⇄y	025 -41		
PRINT x	026 -14	Print approximation for pi.	
RCL 2	027 36 02		
PRINT x	028 -14	Print number of loop iterations.	
RTN	029 24		

To run the program:

Slide the PRGM-RUN switch PRGM RUN to RUN.

Press	Display
A	11.0



```
 3.1  ***
11.0  ***
```

You can see that during each iteration through the loop, the calculator pauses to display the iteration number, the latest approximation of pi, and the calculator's value for pi. You can actually watch the approximation converge to the actual value. When the approximation and the calculator's value for pi (to one decimal place) are no longer unequal, execution is transferred out of the loop and the approximation and the number of iterations are printed.

If you wish, you can change the DSP 1 instruction in step 004 to a DSP 2 instruction and run the program again to see how many iterations it takes for the series to converge to two decimal places. (This program will take several minutes to run.)

Pausing for Input

When the calculator executes a PAUSE instruction, program execution actually halts for the period of time (about one second) of the pause. You can use a pause like this to key data into or perform functions from the keyboard in a program, instead of using a R/S instruction to stop the running program completely.

When you press any key during the one-second "window" while the calculator is executing a PAUSE instruction, that key actually operates, and you have an additional one second of time to view the result or to press another key. If you press yet another key during the subsequent one second, the calculator will perform that operation and pause for another second.

If you press a function key during a pause, the function key operates upon the number contained in the X-register at the time. The result of the function is then seen in the display for about one second. Any function key that is programmable can also be operated from the keyboard during a PAUSE.

If you press a digit key, or a series of digit keys, during a pause, the number appears in the display for the length of a pause (about one second) after you key in the number. (If a number has been input from the program immediately before the pause, that number is first terminated by the PAUSE instruction.) The number that you key in is terminated at the end of the pause. Any subsequent digits in a program will then be part of a new number.

When a PAUSE instruction has completed execution, the program continues to be executed sequentially. If you have performed a function, or keyed in a number, program execution begins with the next instruction using the number that is in the displayed X-register at the end of the pause. (You can also read a magnetic card during a PAUSE. More about this in section 14, Card Reader Operations.)

Number termination occurs at the end of each PAUSE, so you should not attempt to key in a number during more than one subsequent pause. Since you have about one second after your last keystroke to continue keying in digits or functions, you don't need more than one PAUSE instruction to key in even a very long number.

Example: The following program calculates the average value of any number of values. Each new value is keyed in during the pause, and the value of the average is then updated and printed. If no new number is keyed in, no new value is printed.

To key in the program:

Slide the PRGM-RUN switch PRGM ▮▯ RUN to PRGM.

Press	**Display**	
f CLPRGM	000	
LBL A	001 21 11	
f CL REG	002 16-53	Clears secondary storage registers.
f P⇄S	003 16-51	
LBL 1	004 21 01	
CL x	005 -51	
f PAUSE	006 16 51	Pause for input.
f x=0?	007 16-43	If no input, go to LBL 1.
GTO 1	008 22 01	
f PRINT: SPACE	009 16-11	Otherwise, print input, and calculate and print new average.
PRINT x	010 -14	
Σ+	011 56	
f x̄	012 16 53	
PRINT x	013 -14	
GTO 1	014 22 01	Then go to LBL 1 again.
RTN	015 24	

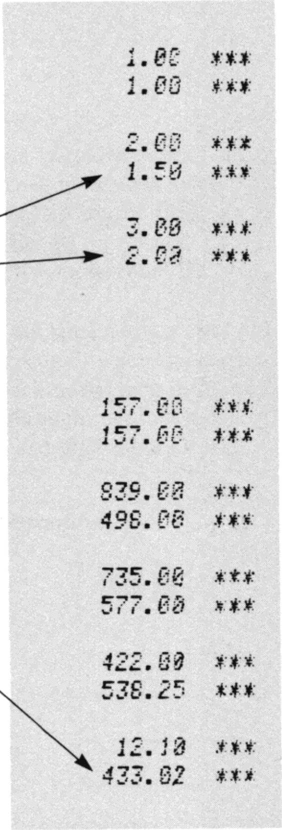
Now run the program to find the average of 1, 2, and 3; of 157, 839, 735, 422, and 12.1.

Slide the PRGM-RUN switch PRGM ▯▮ RUN to RUN.

Press	**Display**
A	0.00

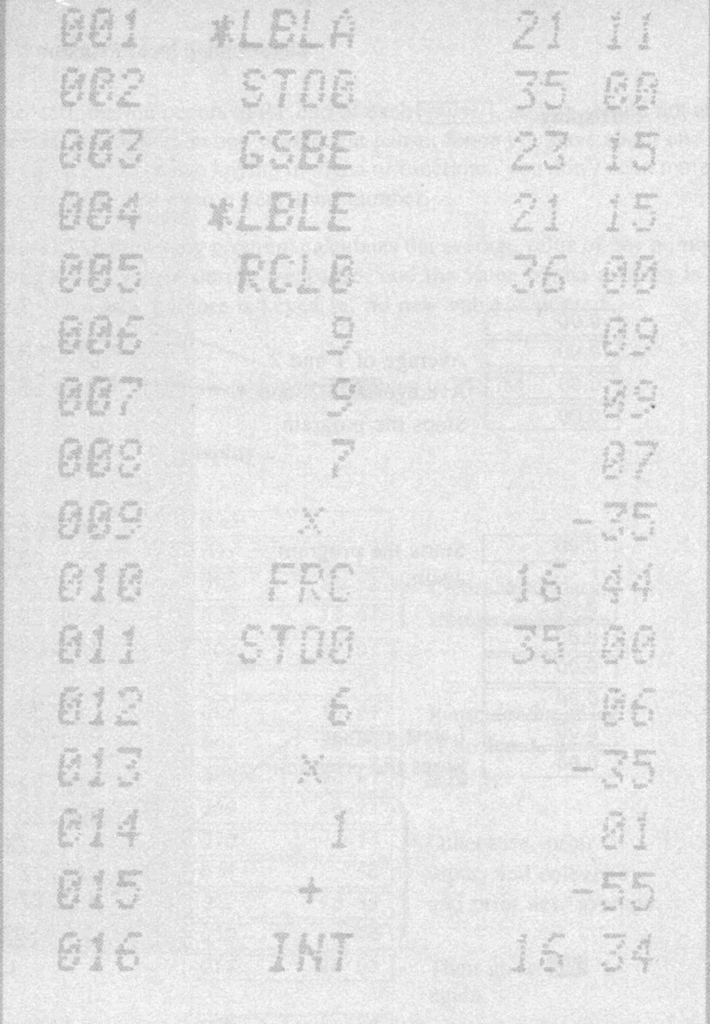
The program is now running. Each time the display pauses and shows 0.00, you can key in a value and it will be averaged with previous values in the calculator. If you key in no value at the pause, no new average will be computed and printed. To stop the program and start it for a new set of data, press R/S (or any key).

Press	Display	
1	0.00	
2	0.00	Average of 1 and 2.
3	0.00	Average of 1, 2, and 3.
R/S	0.00	Stops the program.
A	0.00	Starts the program again.
157	0.00	
839	0.00	
735	0.00	
422	0.00	
12.1	0.00	Latest average.
R/S	0.00	Stops the program.



```
    1.00  ***
    1.00  ***

    2.00  ***
    1.50  ***

    3.00  ***
    2.00  ***

  157.00  ***
  157.00  ***

  839.00  ***
  498.00  ***

  735.00  ***
  577.00  ***

  422.00  ***
  538.25  ***

   12.10  ***
  433.82  ***
```

You can see that it is easy to key in a number of any length during the execution of a PAUSE instruction.



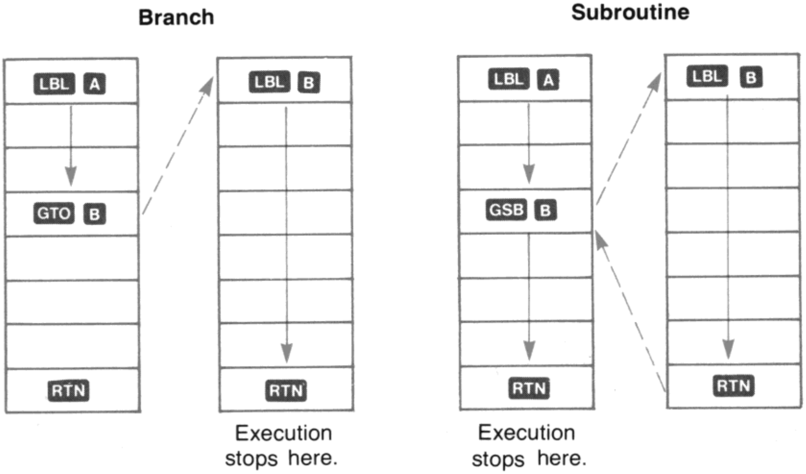
```
001  *LBLA      21 11
002   STO0      35 00
003   GSBE      23 15
004  *LBLE      21 15
005   RCL0      36 00
006      9         09
007      9         09
008      7         07
009      x        -35
010    FRC      16 44
011   STO0      35 00
012      6         06
013      x        -35
014      1         01
015      +        -55
016    INT      16 34
```

Section 10

Subroutines

Often, a program contains a certain series of instructions that are executed several times throughout the program. When the same set of instructions occurs more than once in a program, it can be executed as a subroutine. A subroutine is selected by the GSB (*go to subroutine*) operation, followed by a label address (A through E, f a through f e, and 0 through 9). You can also select a subroutine with GSB (i)—more about (i) later.

A GSB instruction transfers execution to the routine specified by the label address, just like a GTO instruction. However, after a GSB instruction has been executed, when the running program then executes a RTN (*return*), execution is transferred back to the next instruction after the GSB. Execution then continues sequentially downward through program memory. The illustration below should make the distinction between GTO and GSB more clear.



In the illustration on the left if you pressed A from the keyboard, the program would execute instructions sequentially downward through program memory. If it encountered a GTO B instruction, it would then search for the next LBL B and continue execution from there, until it encountered a RTN. When it executed the RTN instruction, execution would stop.

However, if the running program encounters a GSB B (*go to subroutine B*) instruction, as shown on the right, it searches downward for the next LBL B and resumes execution. When it encounters a RTN (*return*), program execution is once again transferred, this time back to the point of origin of the subroutine, and execution *resumes* with the next instruction after the GSB B.

As you can see, the only difference between a subroutine and a normal branch is the transfer of execution *after* the RTN. After a GTO, the next RTN halts a running program; after a GSB, the next RTN returns execution back to the main program, where it continues until another RTN (or a R/S) is encountered. The same routine may be executed by GTO and GSB any number of times in a program.

Example: A quadratic equation is of the form $ax^2 + bx + c$. Its two roots may be found by the formulas $r_1 = \dfrac{-b + \sqrt{b^2 - 4ac}}{2a}$ and $r_2 = \dfrac{-b - \sqrt{b^2 - 4ac}}{2a}$. Notice the similarity between the solutions for r_1 and r_2. The program below permits you to key the values for *a, b,* and *c* beneath user-definable keys A, B, and C; the resultant roots r_1 and r_2 are available by pressing D and E. Were you to record this program on a magnetic card, the card might look like this:

Here is a complete program for calculating the two roots of a quadratic equation:

Input a

```
001  *LBLA    21 11
002   STO1    35 01
003    RTN       24
```

Input b

```
004  *LBLB    21 12
005   STO2    35 02
006    RTN       24
```

Input c

```
007  *LBLC    21 13
008   STO3    35 03
009    RTN       24
```

Calculate r₁

```
010  *LBLD      21 14
011   RCL2      36 02
012    CHS        -22
013   RCL2      36 02
014     X²         53
015   RCL1      36 01
016   RCL3      36 03
017      ×        -35
018      4         04
019      ×        -35
020      -        -45
021     √X         54
022      +        -55
023   RCL1      36 01
024      2         02
025      ×        -35
026      ÷        -24
027    RTN         24
```

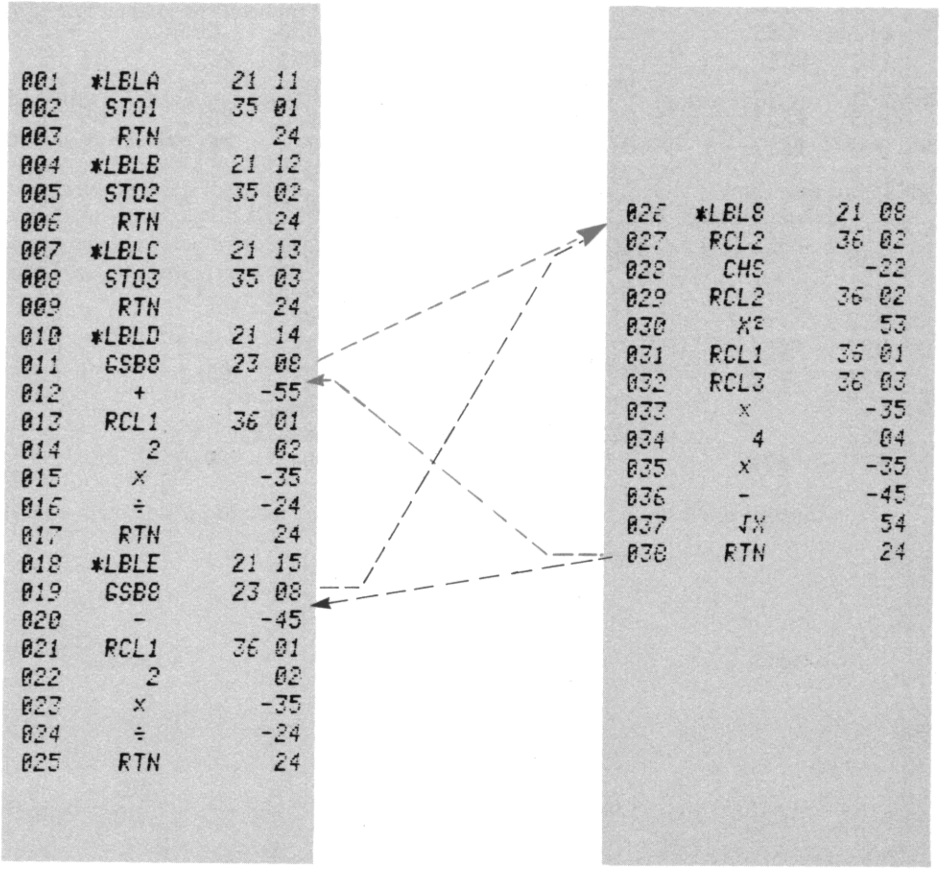
These sections of program memory are identical.

Calculate r₂

```
028  *LBLE      21 15
029   RCL2      36 02
030    CHS        -22
031   RCL2      36 02
032     X²         53
033   RCL1      36 01
034   RCL3      36 03
035      ×        -35
036      4         04
037      ×        -35
038      -        -45
039     √X         54
040      -        -45
041      2         02
042   RCL1      36 01
043      ×        -35
044      ÷        -24
045    RTN         24
```

Since the routine for calculating r_1 contains a large section of program memory that is identical to a large section in the routine for calculating r_2, you can simply create a *subroutine* that will execute this section of instructions. The subroutine is then called up and executed in both the solution for r_1 and the solution for r_2.

This illustrates how the subroutine is used in the program.



```
001  *LBLA    21 11
002   STO1    35 01
003    RTN       24
004  *LBLB    21 12
005   STO2    35 02
006    RTN       24
007  *LBLC    21 13
008   STO3    35 03
009    RTN       24
010  *LBLD    21 14
011   GSB8    23 08
012      +      -55
013   RCL1    36 01
014      2       02
015      ×      -35
016      ÷      -24
017    RTN       24
018  *LBLE    21 15
019   GSB8    23 08
020      -      -45
021   RCL1    36 01
022      2       02
023      ×      -35
024      ÷      -24
025    RTN       24
```

```
026  *LBL8    21 08
027   RCL2    36 02
028    CHS      -22
029   RCL2    36 02
030     X²       53
031   RCL1    36 01
032   RCL3    36 03
033      ×      -35
034      4       04
035      ×      -35
036      -      -45
037     √X       54
038    RTN       24
```

With the modified program, when you press **D** execution begins with the **LBL** **D** instruction in step 010. When the **GSB** 8 instruction in step 011 is encountered, execution transfers to **LBL** 8 in step 026 and computes the quantities $-b$ and $\sqrt{b^2 - 4ac}$, placing them in the X- and Y-registers of the stack, ready for addition or subtraction. When the **RTN** instruction in step 038 is encountered, execution transfers back to the main routine and continues with the **+** instruction in step 012. Thus the root r_1 is computed and displayed, and the routine stops with the **RTN** in step 017.

When you press **E**, execution begins with **LBL** **E**, transfers out to execute the **LBL** 8 subroutine, and returns. This time $\sqrt{b^2 - 4ac}$ is *subtracted* from $-b$, and root r_2 is computed. By using a subroutine, seven steps of program memory are saved!

To key in the program and the subroutine:

Slide the PRGM-RUN switch PRGM RUN to PRGM.

Press	Display	
f CL PRGM	000	
LBL A	001 21 11	Stores a in R_1.
STO 1	002 35 01	
RTN	003 24	
LBL B	004 21 12	Stores b in R_2.
STO 2	005 35 02	
RTN	006 24	
LBL C	007 21 13	Stores c in R_3.
STO 3	008 35 03	
RTN	009 24	
LBL D	010 21 14	Calculates $r_1 = \dfrac{-b + \sqrt{b^2 - 4ac}}{2a}$.
GSB 8	011 23 08	
+	012 -55	
RCL 1	013 36 01	
2	014 02	
×	015 -35	
÷	016 -24	
RTN	017 24	
LBL E	018 21 15	Calculates $r_2 = \dfrac{-b - \sqrt{b^2 - 4ac}}{2a}$.
GSB 8	019 23 08	
−	020 -45	
RCL 1	021 36 01	
2	022 02	
×	023 -35	
÷	024 -24	
RTN	025 24	
LBL 8	026 21 08	Subroutine places $-b$ in Y-register and $\sqrt{b^2-4ac}$ in X-register, ready for addition or subtraction.
RCL 2	027 36 02	
CHS	028 -22	
RCL 2	029 36 02	
x^2	030 53	
RCL 1	031 36 01	
RCL 3	032 36 03	
×	033 -35	
4	034 04	
×	035 -35	
−	036 -45	
$\sqrt{x}$	037 54	
RTN	038 24	

To initialize the program, you key in *a* and press A, key in *b* and press B, and key in *c* and press C. Then, to find root r_1, press D. To find root r_2, press E.

Run the program now to find the roots of the equation $x^2 + x - 6 = 0$; of $3x^2 + 2x - 1 = 0$.

To run the program:

Slide the PRGM-RUN switch PRGM ▮▯ RUN to RUN.

Press	**Display**	
1 A	1.00	
1 B	1.00	
6 CHS C	-6.00	
D	2.00	Calculates the first root, r_1.
E	-3.00	Calculates the second root, r_2.
3 A	3.00	
2 B	2.00	
1 CHS C	-1.00	
D	0.33	Calculates r_1.
E	-1.00	Calculates r_2.

If the quantity $b^2 - 4ac$ is a negative number, the calculator will display Error and the running program will stop. For a more efficient and accurate method of finding the roots of a quadratic equation, see the Polynomial Evaluation program in your *HP-97 Standard Pac*.

Note: When loading instructions into the calculator in PRGM mode, you can load a GSB A through E or GSB f a through f e by simply pressing the appropriate user-definable key(s). For example, to load the instruction GSB A you can simply press A; the keycode for GSB A, 23 11, will appear in the display. For clarity and ease of reference, however, in this handbook the complete keystroke sequence is always shown.

Routine-Subroutine Usage

Subroutines give you extreme versatility in programming. A subroutine can contain a loop, or it can be executed as part of a loop. Another common and space-saving trick is to use the same routine both as a subroutine and as part of the main program.

Example: The program below simulates the throwing of a pair of dice, printing first the value of one die (an integer from one to six) and then that of the second die (another integer from one to six). The "heart" of the program is a random number generator (actually a pseudo random number generator) that is executed first as a subroutine and then as part of the main program. When you key in a first number (called a "seed"), and press A, the digit for the first die is generated and printed using the E routine as a subroutine. Then the digit for the second die is generated using the same routine as part of the main program.

To key in the program:

Slide the PRGM-RUN switch PRGM RUN to PRGM.

Press	Display	
f CLPRGM	000	
LBL A	001 21 11	
STO 0	002 35 00	
f PRINT: SPACE	003 16-11	
GSB E	004 23 15	E executed first as subroutine.
LBL E	005 21 15	E then executed as a routine.
RCL 0	006 36 00	
9	007 09	
9	008 09	
7	009 07	
×	010 -35	
f FRAC	011 16 44	
STO 0	012 35 00	
6	013 06	
×	014 -35	
1	015 01	
+	016 -55	
f INT	017 16 34	
DSP 0	018 -63 00	
PRINT x	019 -14	
RTN	020 24	

Now slide the PRGM-RUN switch to RUN and "roll" the dice with your HP-97. To roll the dice, key in a decimal "seed" (that is, $0 < n < 1$). Then press A. The calculator will print the number rolled by the first die, then the number rolled by the second. To make another roll, key in a new seed and press A again.

You can play a game with your friends using the "dice." If your first "roll" is 7 or 11, you win. If it is another number, that number becomes your "point." You then keep "rolling" (keying in seeds and pressing A) until the dice again total your point (you win) or you roll a 7 or an 11 (you lose). To run the program:

Slide the PRGM-RUN switch PRGM RUN to RUN.

Press

.2315478 A

Your point is 10. →

```
6.    ***
4.    ***
```

.3335897 A

You missed your point. →

```
4.    ***
1.    ***
```

.9987562 A

Missed it again. →

```
5.    ***
4.    ***
```

.9987563 A

Congratulations! You win. →

```
5.    ***
5.    ***
```

Now try it again.

Press

.21387963 A

Your point is 4. →

```
2.    ***
2.    ***
```

.6658975 A

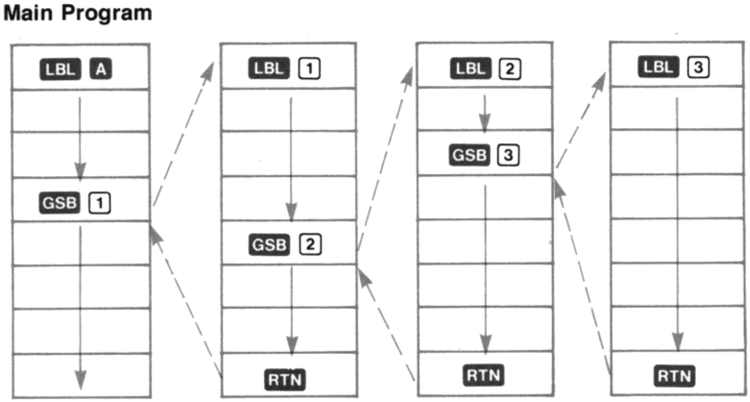
Whoops! You lose. →

```
6.    ***
1.    ***
```

Subroutine Limits

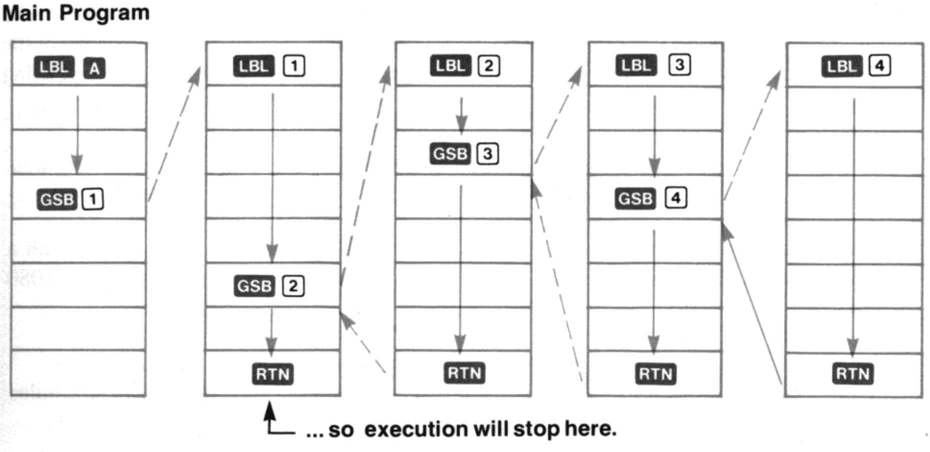
A subroutine can call up another subroutine, and that subroutine can call up yet another. Subroutine branching is limited only by the number of *returns* that can be held pending by the HP-97. Three subroutine returns can be held pending at any one time in the HP-97. The diagram below should make this more clear.

Three returns can be pending.



The calculator can return back to the main program from subroutines that are three deep, as shown. However, if you attempt to call up subroutines that are four deep, the calculator will execute only three returns:

Only three returns can be pending...



... so execution will stop here.

Naturally, the calculator can execute the RTN instruction as a *stop* any number of times. Also, if you press A through E, f a through f e, or GSB A through E, GSB f a through f e, or GSB 0 through 9 from the *keyboard,* all *pending* RTN instructions are forgotten by the calculator.

If you are executing a program one step at a time with the SST key and encounter a GSB instruction, the calculator will execute the entire subroutine before continuing to the next step. However, only one RTN instruction may be executed as the result of a GSB instruction during single-step execution; so if a program contains a subroutine within a subroutine, execution will not return to the main program during SST execution.

Problems

1. Look closely at the program for finding roots r_1 and r_2 of a quadratic equation (page 181). Can you see other instructions that could be replaced by a subroutine? (Hint: Look at steps 013 through 016 and steps 021 through 024.) Modify the program by using another subroutine and run it to find the roots of $x^2 + x - 6 = 0$; of $3x^2 + 2x - 1 = 0$.

   Answers: 2, −3; 0.33, −1.

   How many more steps of program memory did you save?

2. The area of a sphere can be calculated according to the equation $A = 4\pi r^2$, where r is the radius. The formula for finding the volume of a sphere is $V = \frac{4\pi r^3}{3}$. This may also be expressed as $V = \frac{r \times A}{3}$.

   Create and load a program to calculate the area A of a sphere given its radius r. Define the program with LBL A and RTN and include an initialization routine to store the value of the radius. Then create and load a second program to calculate the volume V of a sphere, using the equation $V = \frac{r \times A}{3}$. Define this second program with LBL B and RTN, and include the instruction GSB 1 to use a portion of program A as a subroutine calculating area.

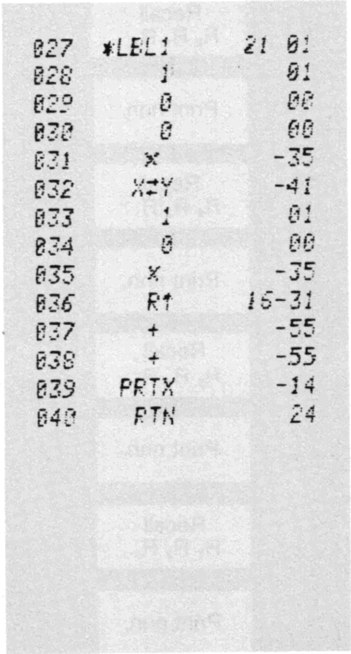
   Run the two programs to find the area and volume of the planet earth, a sphere with a radius of about 3963 miles. Of the earth's moon, a sphere with a radius of about 1080 miles.

   Answers: Earth area = 197359487.5 square miles
   Earth volume = 2.6071188×10^{11} cubic miles
   Moon area = 14657414.69 square miles
   Moon volume = 5276669290 cubic miles.

3. Create, load, and run a program that will print all permutations of three integers that you have stored in registers R_1, R_2, and R_3. For example, all permutations of the integers 1, 2, and 3 might be printed as:

```
123
132
213
231
312
321
```

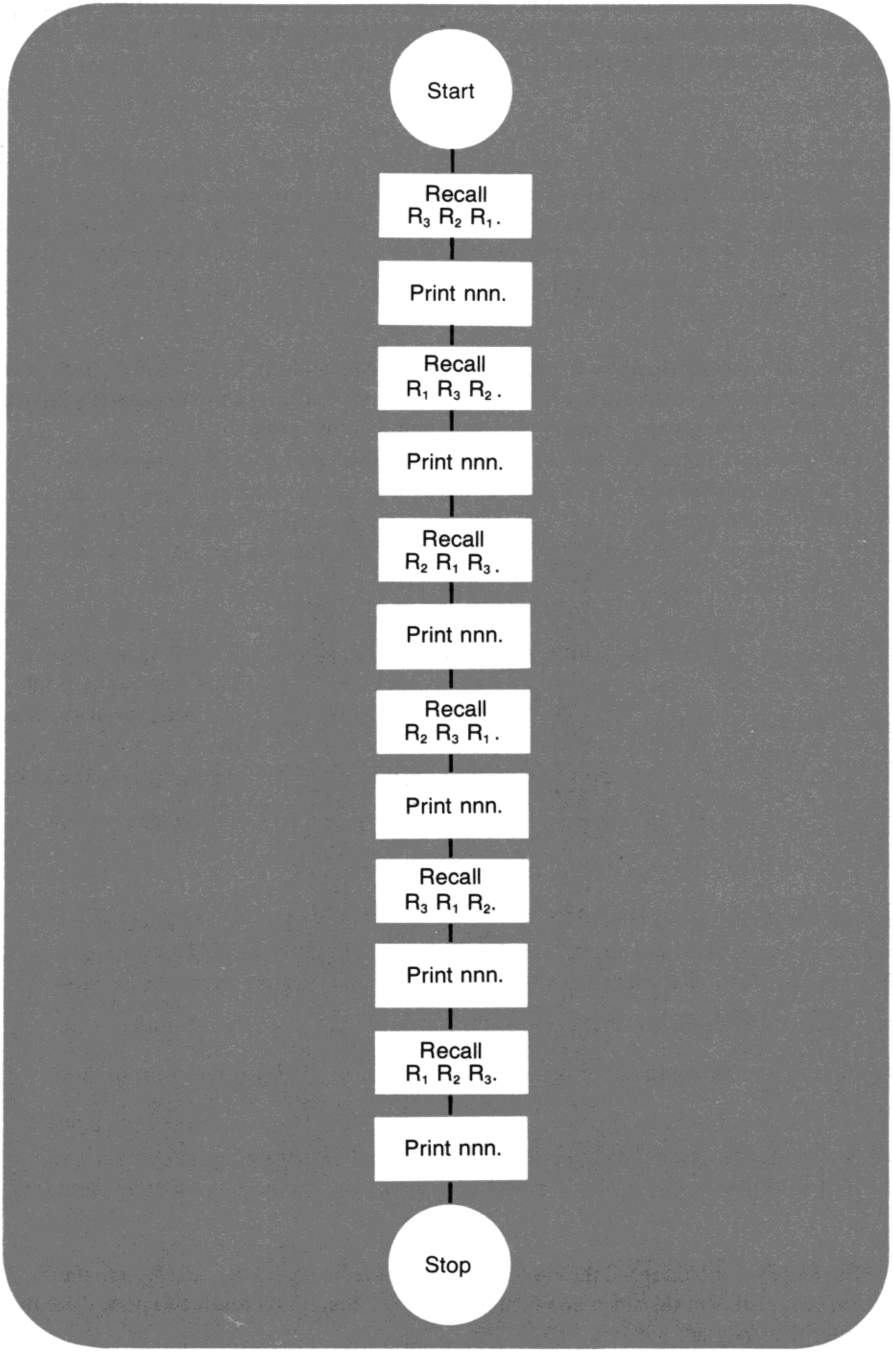
The following subroutine will cause the digits you recall from R_1, R_2, and R_3 to be printed as a permutation in the order you have recalled them. Use this subroutine and the flowchart on the following page to help you create and load the program.



```
027  *LBL1     21 01
028      1         01
029      0         00
030      0         00
031      ×        -35
032    X⇄Y        -41
033      1         01
034      0         00
035      ×        -35
036     R↑      16-31
037      +        -55
038      +        -55
039   PRTX        -14
040    RTN         24
```

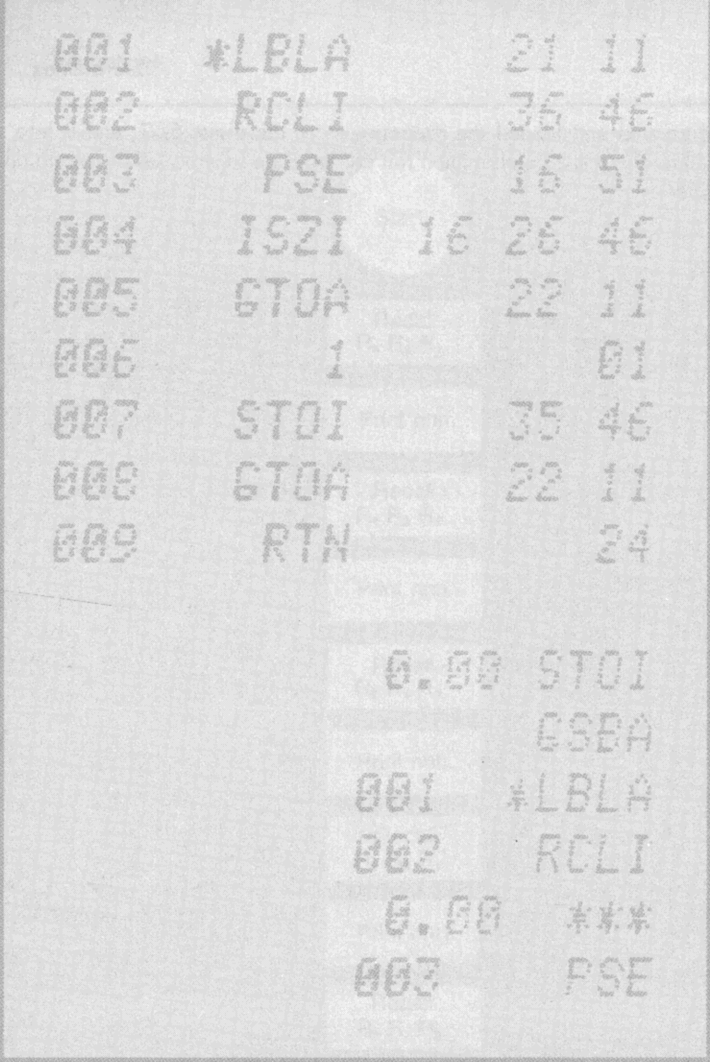
This subroutine prints numbers recalled into the Z-, Y-, and X-registers of the stack as nnn.

The program should recall the contents of storage registers R_1, R_2, and R_3 into the Z-, Y-, and X-registers of the stack and then use the "print nnn" subroutine to print them in the order that they are recalled.


Start
Recall R_3 R_2 R_1.
Print nnn.
Recall R_1 R_3 R_2.
Print nnn.
Recall R_2 R_1 R_3.
Print nnn.
Recall R_2 R_3 R_1.
Print nnn.
Recall R_3 R_1 R_2.
Print nnn.
Recall R_1 R_2 R_3.
Print nnn.
Stop

When you have created and loaded the program, store the digits 5, 7, and 9 into storage registers R_1, R_2, and R_3, respectively. Then run the program to show all the permutations of these three numbers.

Answer:
579
795
957
597
759
975



```
001  *LBLA       21 11
002   RCLI       36 46
003    PSE       16 51
004   ISZI    16 26 46
005   GTOA       22 11
006      1          01
007   STOI       35 46
008   GTOA       22 11
009    RTN          24


             6.00 STOI
                  GSBA
            001  *LBLA
            002   RCLI
             6.00  ***
            003    PSE
```

Section 11

Controlling the I-Register

The I-register is one of the most powerful programming tools available to you on your HP-97. In a preceding section, Storing and Recalling Numbers, you learned about the use of the I-register as a simple storage register, similar to registers R_0 through R_9, R_A through R_E, and R_{S0} through R_{S9}. And of course, you can always use the I-register this way, as another storage register, whether you are using it as an instruction in a program or operating manually from the keyboard.

But the I-register is much more powerful than a mere storage register. Using the instructions I, (i), and x⇄I in conjunction with other instructions, you can specify the storage register addresses of STO and RCL, the label addresses of GTO and GSB, or the number of digits displayed by a DSP instruction. By storing a negative number in the I-register, you can even transfer execution to any step number of program memory. The ISZ and DSZ instructions allow you to increment (add one to) or decrement (subtract one from) the current value in I (or, using (i), to increment or decrement *any* storage register). This is a feature that you will find extremely useful in controlling loops.

Storing a Number in I

To store a number in the I-register, you can use the STO I operation. For example, to store the number 7 in the I-register:

Ensure that the PRGM-RUN switch PRGM ▮▯ RUN is set to RUN.

Press	**Display**
7 STO I	7.00

To recall a number from the I-register into the displayed X-register, you do not have to use the RCL operation—you merely press I.

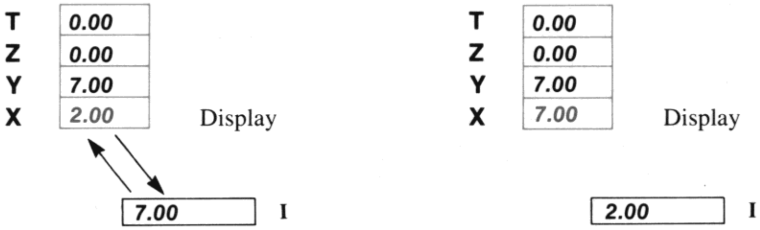
Press	**Display**	
CL X	0.00	
I	7.00	The number stored in I is recalled.

Exchanging x and I

In a manner similar to the x⇄y and P⇄S operations, the f x⇄I operation exchanges the contents of the displayed X-register with those of the I-register. For example, key the number 2 into the displayed X-register and exchange the contents of the X-register with those of the I-register now:

Press	Display	
2	2.	
f x⇄I	7.00	Contents of X-register and I-register exchanged.

When you pressed x⇄I, the contents of the stack and the I-register were changed...



... from this ... ... to this.

To restore the X-register and I-register contents to their original positions:

Press	Display
f x⇄I	2.00

Incrementing and Decrementing the I-Register

You have seen how a number can be stored in the I-register and then changed, either by storing another number there, or by using the f x⇄I operation. You will find both of these methods useful, whether you are utilizing them as instructions in a program or using them manually from the keyboard.

Another way of altering the contents of the I-register, and one that is most useful during a program, is by means of the f ISZ I (*increment, skip if zero*) and f DSZ I (*decrement, skip if zero)* instructions. These instructions either add the number 1 to (increment) or subtract the number 1 from (decrement) the I-register each time they are executed. In a running program, if the number in the I-register has become zero, program execution *skips* the next step after the ISZ I or DSZ I instruction and continues execution (just like a false conditional instruction).

The f ISZ and f DSZ instructions always increment or decrement first; *then* the test for zero is made. For test purposes, numbers between but not including −1 and +1 are the same as zero.

Example: Here is a program that illustrates how f ISZ I works. It contains a loop that pauses to display the current value in the I-register, then uses the f ISZ I instruction to increment that value. The program will continue to run, continually adding one to and displaying the contents of the I-register, until you press R/S (or any key) from the keyboard.

To key in the program:

Slide the PRGM-RUN switch PRGM ▮ RUN to PRGM.

Press	**Display**	
f CLPRGM	000	
LBL A	001 21 11	
I	002 36 46	Recalls I-register contents.
f PAUSE	003 16 51	Pauses to display contents.
f ISZ I	004 16 26 46	Adds 1 to I-register.
GTO A	005 22 11	If contents of I-register are not zero, execution transfers back to LBL A.
1	006 01	If contents of I-register are zero, 1 is placed in I-register.
STO I	007 35 46	
GTO A	008 22 11	
RTN	009 24	

Now run the program beginning with a value of 0 in the I-register. Stop the program after five iterations or so by pressing R/S.

Slide the PRGM-RUN switch PRGM RUN to RUN.

Press	**Display**	
0 STO I	0.00	Zero stored in I-register.
A	0.00	
	1.00	
	2.00	
	3.00	
	4.00	
R/S	5.00	

Although the ISZ I and DSZ I instructions increment and decrement the I-register by 1, the value of the I-register need not be a whole number. For example:

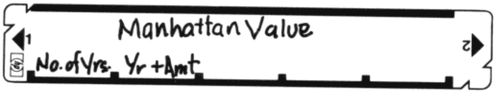
Press	**Display**
5.28 CHS	-5.28
STO I	-5.28
A	-5.28
	-4.28
	-3.28
	-2.28
	-1.28
R/S	1.00

In practice, you will find that you will usually use ISZ I and DSZ I with numbers that are integers, since these instructions are most useful as counters—that is, to control the number of iterations of a loop—and to select storage registers, subroutines, or display settings. (More about using the I-register as a selection register later.)

The DSZ (*decrement, skip if zero*) instruction operates in the same manner as the increment instruction, except that it subtracts, rather than adds, one each time it is used. When a running program executes an f DSZ I instruction, for example, it subtracts 1 from the contents of the I-register, then tests to see if the I-register is 0. (A number between +1 and -1 tests as zero.) If the number in the I-register is greater than zero, execution continues with the next step of program memory. If the number in the I-register *is* zero, the calculator skips one step of program memory before resuming execution.

Example: The island of Manhattan was sold in the year 1624 for $24.00. The program below shows how the amount would have grown each year if the original amount had been placed in a bank account drawing 5% interest compounded annually. The number of years for which you want to see the amount is stored in the I-register, then the DSZ I instruction is used to keep track of the number of iterations through the loop.

Were you to prepare a magnetic card to store this program, it might look like this:



To key in the program:

Slide the PRGM-RUN switch PRGM ▯▮ RUN to PRGM.

Press	Display	
f CLPRGM	000	
LBL A	001 21 11	Initialization routine.
STO I	002 35 46	
1	003 01	
6	004 06	
2	005 02	
4	006 04	
STO 1	007 35 01	
2	008 02	
4	009 04	
STO 2	010 35 02	
RTN	011 24	

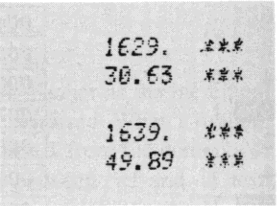
Press	Display	
LBL B	012 21 12	Counting loop, controlled by I-register and DSZ I.
RCL 2	013 36 02	
5	014 05	
%	015 55	
STO + 2	016 35-55 02	
1	017 01	
STO + 1	018 35-55 01	
f DSZ I	019 16 25 46	
GTO B	020 22 12	
f PRINT: SPACE	021 16-11	← When value in I becomes zero, execution skips to here, and year and amount are printed.
RCL 1	022 36 01	
DSP 0	023 -63 00	
PRINT x	024 -14	
RCL 2	025 36 02	
DSP 2	026 -63 02	
PRINT x	027 -14	
RTN	028 24	

To run the program, key in the number of years for which you want to see the amount. Press A to store the number of years in the I-register and otherwise initialize the program. Then press B to run the program.

For example, to run the program to find the amount of the account after 5 years; after 15 years:

Slide the PRGM-RUN switch PRGM RUN to RUN.

Press	Display	
5 A	24.00	Program initialized.
B	30.63	After five years, in 1629, the account would have been worth $30.63.
15 A	24.00	Program initialized.
B	49.89	After 15 years, in 1639, the account would have been worth $49.89.



```
1629.     ***
30.63     ***

1639.     ***
49.89     ***
```

How it works: When you key in the number of years and initialize the program by pressing A, the number of years is stored in the I-register by the STO I instruction. The year (1624) is stored in primary storage register R_1, and the amount ($24.00) is stored in primary storage register R_2.

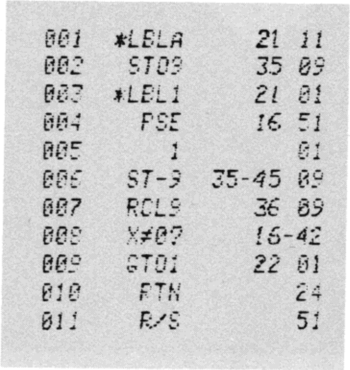
When you then press B, calculation begins. Each time through the loop, 5% of the amount is computed and added to the amount in R_2, and one (1) year is added to the year in R_1. The DSZ I instruction subtracts one from the I-register; if the value in I is not then zero, execution is transferred back to LBL B and the loop is executed again.

The loop continues to be executed until the value in the I-register becomes zero. Then execution skips to the f PRINT: SPACE instruction in program memory step 021. Execution continues sequentially downward from step 021, recalling the current year from R_1 and formatting and printing it, then recalling the current amount from R_2 and formatting and printing that underneath the year.

To see what the amount in the account would be in 1976, you can key in the number of years from 1624 to 1976 (the number is 352) and initialize and run the program. (This will take 4–5 minutes to run, plenty of time to go get a cup of coffee.)

Problems

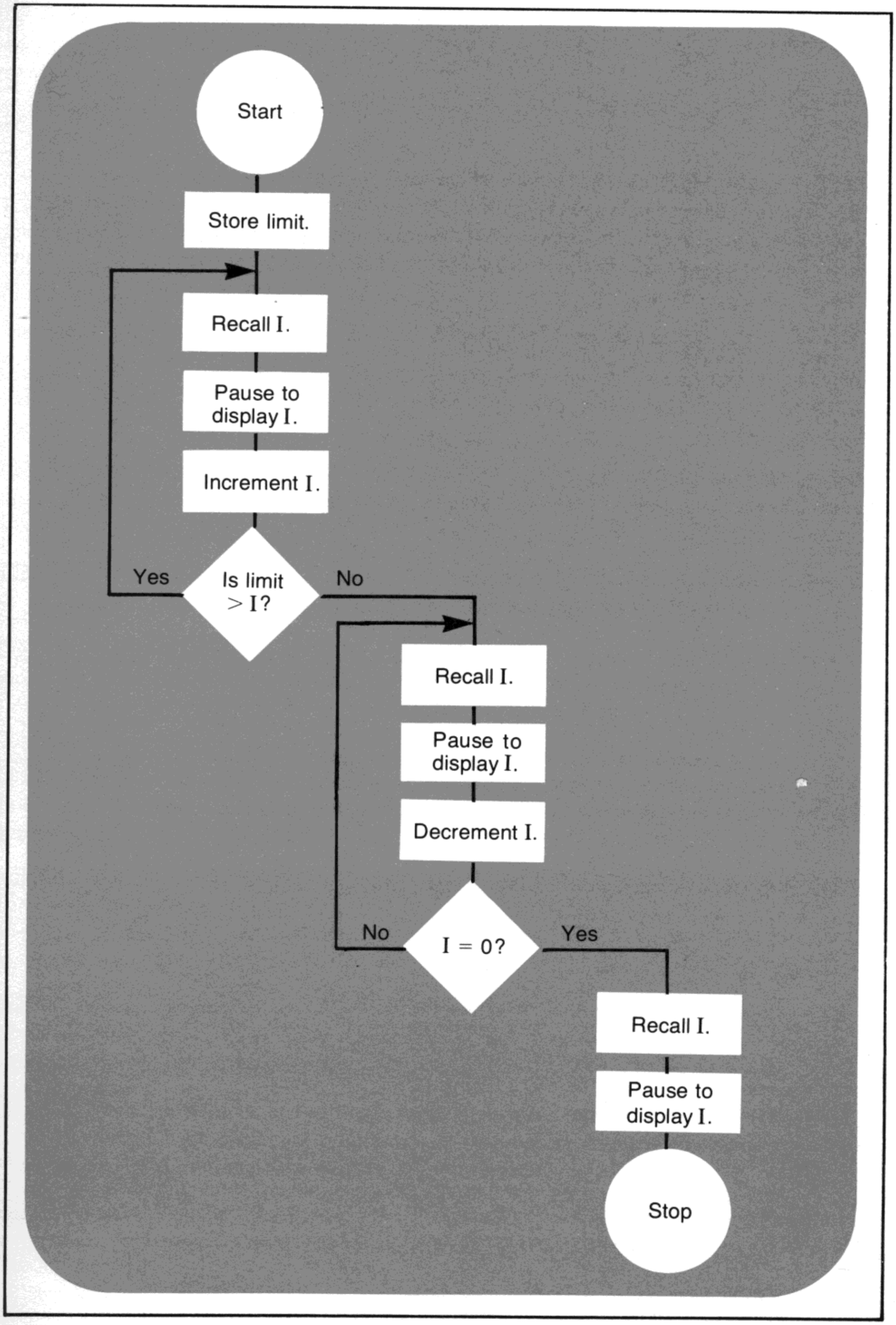
1. When you press A, the program below stores in primary storage register R_9 a number that you have keyed in, then decrements the value in R_9 using storage register arithmetic. Each time through the loop, the program pauses to show the current value in R_9. When the value in R_9 reaches zero, the program stops. Write, load, and run a program that uses the I-register and f DSZ I instead of R_9 and f x≠0? to give the same results.

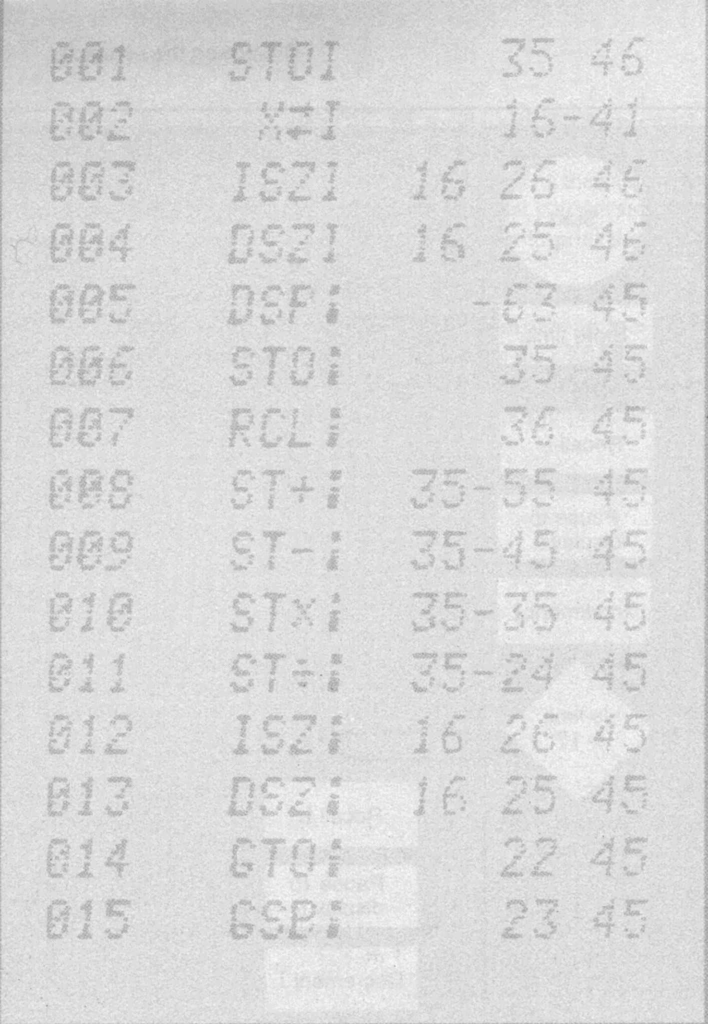


```
001  *LBLA      21 11
002   STO9      35 09
003  *LBL1      21 01
004    PSE      16 51
005      1         01
006   ST-9   35-45 09
007   RCL9      36 09
008   X≠0?      16-42
009   GTO1      22 01
010    RTN         24
011    R/S         51
```

2. Write and load a program using ISZ I to illustrate how an initial deposit of $1000 would grow year-by-year at a yearly compound interest rate of 5.5%. The program should print the current year and subsequent years, together with the value of the account for each year. The program should contain an infinite loop that you can stop by pressing R/S from the keyboard whenever you wish. Run the program to print the year and amount for at least 5 years.

3. Write, load, and run a program that will count from zero *up* to a limit using the ISZ I instruction, and then count back down to zero using the DSZ I instruction. The program can contain two loops, and it can contain a conditional instruction besides the ISZ I and DSZ I instructions. Use the flowchart on the opposite page to help you.


Start
Store limit.
Recall I.
Pause to display I.
Increment I.
Is limit > I?
Yes
No
Recall I.
Pause to display I.
Decrement I.
I = 0?
No
Yes
Recall I.
Pause to display I.
Stop



```
001     STOI         35 46
002      X⇄I         16-41
003     ISZI      16 26 46
004     DSZI      16 25 46
005     DSPi        -63 45
006     STOi         35 45
007     RCLi         36 45
008     ST+i      35-55 45
009     ST-i      35-45 45
010     STxi      35-35 45
011     ST÷i      35-24 45
012     ISZi      16 26 45
013     DSZi      16 25 45
014     GTOi         22 45
015     GSBi         23 45
```

Section 12

Using the I-Register for Indirect Control

You have seen how the value in the I-register can be altered using the STO, x⇄I, ISZ I and DSZ I operations. But the value contained in the I-register can also be used to *control* other operations. The (i) *(indirect)* function combined with certain other functions allows you to control those functions using the current number in the I-register. (i) uses the number stored in the I-register as an *address*.

The indirect operations that can be controlled by the I-register are:

DSP (i), when the number in the I-register is 0 through 9, changes display formatting so that the number in the display contains the number of decimal places specified by the current number in the I-register.

STO (i), when the number in the I-register is 0 through 25, stores the value that is in the display in the primary or secondary storage register addressed by the current number in the I-register.

RCL (i), when the number in the I-register is 0 through 25, recalls the contents of the primary or secondary storage register addressed by the current number in the I-register.

STO + (i), STO − (i), STO × (i), and STO ÷ (i), when the number in the I-register is 0 through 25, perform storage register arithmetic upon the contents of the primary or secondary storage register addressed by the current number in the I-register.

ISZ (i), when the number in the I-register is 0 through 25, increments (adds 1 to) the contents of the primary or secondary storage register addressed by the current number in the I-register. In a running program, one step is skipped if the contents of the addressed register are then zero.

DSZ (i), when the number in the I-register is 0 through 25, decrements (subtracts 1 from) the contents of the primary or secondary storage register addressed by the current number in the I-register. In a running program, one step is skipped if the contents of the addressed register are then zero.

GTO (i), when the number in the I-register is 0 or a positive 1 through 19, transfers execution of a running program sequentially downward through program memory to the next label specified by the current number in the I-register.

GTO (i), when the number in the I-register is a negative number between −1 and −999, transfers execution of a running program *back* in program memory the number of steps specified by the current negative number in the I-register.

GSB (i), when the number in the I-register is 0 through 19, transfers execution of a running program to the subroutine specified by the current number in the I-register. Like a normal subroutine, when a RTN is then encountered, execution transfers forward and continues with the step following the GSB (i) instruction.

GSB (i), when the number in the I-register is a negative number between −1 and −999, transfers execution of a running program *back* in program memory the number of steps specified by the current negative number in the I-register. Operation is then like a normal subroutine.

If the number in the I-register is outside the specified limits when the calculator attempts to execute one of these operations, the display will show Error. When using the (i) function, the calculator uses for an address only the integer portion of the number currently stored in the I-register. Thus, 25.99998785 stored in the I-register retains its full value there, but when used as address (i), it is read as 25 by the calculator.

> In all cases using the (i) *(indirect)* function, the HP-97 looks at only the integer portion of the current number stored in the **I**-register.

You can already see that using the I-register and the (i) function in conjunction with these other functions gives you a tremendous amount of computing power and exceptional programming control. Now let's have a closer look at these operations.

Indirect Display Control

You can use the current number in the I-register in conjunction with the DSP key to control the number of decimal places to which a number is displayed and printed. When DSP (i) is performed, the display is seen rounded to the number of decimal places specified by the current value contained in the I-register. (The display is *seen* rounded, but of course, the calculator maintains its full accuracy, 10 digits multiplied by 10 raised to a two-digit exponent, internally.) The number in the I-register can be any value, positive or negative, from 0 through 9. The DSP (i) operation is most useful as part of a program, but it can also be executed manually from the keyboard. For example:

Slide the PRGM-RUN switch PRGM RUN to RUN.

Press	**Display**	
5 STO I	5.00	
CL X	0.00	Normal FIX 2 display.
DSP (i)	0.00000	FIX 5 display specified because of the number 5 that is stored in the I-register.
9 STO I	9.00	
DSP (i)	9.000000000	FIX 9 display selected by the number in the I-register.

Thus, by controlling the number in the I-register, you can control many different display options with very few instructions in a program.

Example: The following program prints an example of each display format that is available on your HP-97. It utilizes a subroutine loop containing the DSZ I and DSP (i) instructions to automatically change the number of decimal places printed.

To key in the program:

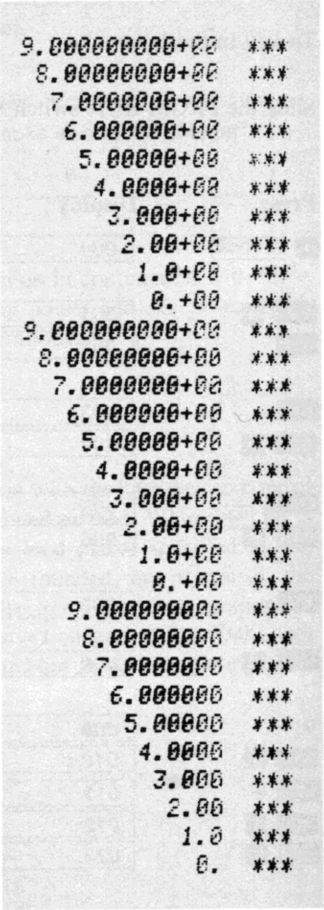
Slide the PRGM-RUN switch PRGM RUN to PRGM.

Press	**Display**			
f CLPRGM	000			
LBL A	001 21 11	Initializes program.		
CL x	002 -51			
SCI	003 -12	Illustrates scientific notation.		
GSB B	004 23 12			
ENG	005 -13	Illustrates engineering notation.		
GSB B	006 23 12			
FIX	007 -11	Specifies fixed point notation.		
LBL B	008 21 12			Subroutine B
9	009 09	Initializes I-register to 9.		
STO I	010 35 46			
LBL 0	011 21 00		Decrementing loop.	
RCL I	012 36 46			
DSP (i)	013 -63 45	Sets displayed decimal places to current value in I-register.		
PRINT x	014 -14			
f DSZ I	015 16 25 46			
GTO 0	016 22 00			
RCL I	017 36 46			
DSP (i)	018 -63 45			
PRINT x	019 -14			
RTN	020 24			

To run the program and see the types of display formatting available on your HP-97:

Slide the PRGM-RUN switch PRGM RUN to RUN.

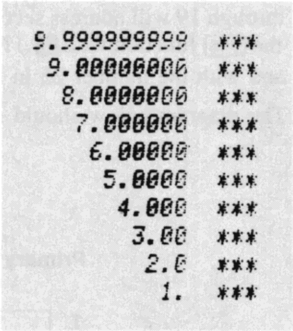
Press	**Display**
A	0.



```
9.000000000+00  ***
 8.00000000+00  ***
  7.0000000+00  ***
   6.000000+00  ***
    5.00000+00  ***
     4.0000+00  ***
      3.000+00  ***
       2.00+00  ***
        1.0+00  ***
         0.+00  ***
9.000000000+00  ***
 8.00000000+00  ***
  7.0000000+00  ***
   6.000000+00  ***
    5.00000+00  ***
     4.0000+00  ***
      3.000+00  ***
       2.00+00  ***
        1.0+00  ***
         0.+00  ***
   9.000000000  ***
    8.00000000  ***
     7.0000000  ***
      6.000000  ***
       5.00000  ***
        4.0000  ***
         3.000  ***
          2.00  ***
           1.0  ***
            0.  ***
```

If a number containing a fraction is stored in the I-register, DSP (i) reads only the integer portion of the number. Thus, the I-register can contain a number as large as 9.999999999, and the DSP (i) operation will still execute. For example:

Press	Display	
9.999999999	9.999999999	
STO I	10.	Display is rounded, but number maintains its original value inside the calculator.
GSB 0	1.	Since the HP-97 is in FIX mode, executing the loop yields the illustration of fixed point notation. (Notice that rounding occurs. Thus, 8.999999999 is rounded to 9.00000000 when printed in FIX 8, etc.)



```
 9.999999999  ***
  9.00000000  ***
   8.0000000  ***
    7.000000  ***
     6.00000  ***
      5.0000  ***
       4.000  ***
        3.00  ***
         2.0  ***
          1.  ***
```

The HP-97 displays Error if the number in the I-register is greater than 9.999999999 when a DSP (i) instruction is executed. For example:

Press	Display
10 STO I	10.
GSB 0	Error

As with all error conditions, pressing any key clears the error and returns to the display the last value present there before the error.

Press	Display
R/S	10.

By using DSP (i), you have tremendous versatility in the types of output formats your HP-97 produces. With DSP (i) instructions, for example, the width of a printed number (that is, the number of characters printed) can be made dependent on data. This means that simple bar graphs can be created with the printer.

Indirect Store and Recall

You can use the number in the I-register to address the 26 storage registers that are in your HP-97. When you press STO (i), the value that is in the display is stored in the storage register addressed by the number in the I-register. RCL (i) addresses the storage registers in a like manner, as do the storage register arithmetic operations STO + (i), STO − (i), STO × (i), and STO ÷ (i). (If you have forgotten the normal operation of the storage registers, or of storage register arithmetic, go back and review section 4, Storing and Recalling Numbers, in this handbook.)

When using STO (i), RCL (i), or any of the storage register arithmetic operations utilizing the (i) function, the I-register can contain positive or negative numbers from 0 through 25. The numbers 0 through 9 address primary storage registers R_0 through R_9, while numbers from 10 through 19 will address secondary storage registers R_{S0} through R_{S9}. (You do not have to use the P⇄S function with (i).) Numbers 20 through 24 address storage registers R_A through R_E, and with the number 25 in the I-register, (i) addresses the I-register itself!

The diagram below should illustrate these addresses more clearly.

Primary Registers

	(i) Address
I	25
R_E	24
R_D	23
R_C	22
R_B	21
R_A	20

Secondary Registers

	(i) Address
R_{S9}	19
R_{S8}	18
R_{S7}	17
R_{S6}	16
R_{S5}	15
R_{S4}	14
R_{S3}	13
R_{S2}	12
R_{S1}	11
R_{S0}	10

	(i) Address
R_9	9
R_8	8
R_7	7
R_6	6
R_5	5
R_4	4
R_3	3
R_2	2
R_1	1
R_0	0

By using the calculator manually, you can easily see how STO (i) and RCL (i) are used in conjunction with the I-register to address the different storage registers:

Ensure that the PRGM-RUN switch PRGM ■□ RUN is set to RUN.

Press	**Display**	
CL x DSP 2	0.00	
f CL REG	0.00	Clears all storage registers, including the I-register, to zero.
f P⇄S	0.00	
f CL REG	0.00	
5 STO I	5.00	Stores the number 5 in the I-register.
1.23 STO (i)	1.23	Stores the number 1.23 in the storage register addressed by the number in I—that is, storage register R_5.
24 STO I	24.00	The number 24 is stored in the I-register.
85083 STO (i)	85083.00	This number stored in the storage register (R_E) addressed by the current number (24) in I.
12 STO I	12.00	Stores the number 12 in the I-register.
77 EEX 43	77. 43	
STO (i)	7.700000000 44	Stores the number 7.7 $\times 10^{44}$ in the storage register addressed by the number in I—that is, in secondary storage register R_{S2}.

Notice that the number was stored *directly* in secondary storage register R_{S2}. You do not have to use the P⇄S function to access the secondary storage registers when using the (i) function.

To recall numbers that are stored in any register, you can use the RCL *(recall)* key followed by the number or letter key of the register address. (For secondary storage registers, use the P⇄S function to exchange contents of the primary and secondary registers before using the RCL function.) However, when the address currently stored in the I-register is correct, you can recall the contents of a storage register by simply pressing (i) (or RCL (i)). For example:

Press	Display	
RCL 5	1.23	Contents of storage register R_5 recalled to displayed X-register.
(i)	7.700000000 44	Since the I-register still contains the number 12, this operation recalls the contents of the storage register (secondary register R_{S2}) addressed by the number 12.

By changing the number in the I-register, you change the address specified by STO (i) or RCL (i). For example:

Press	Display	
24 STO I	24.00	
RCL (i)	85083.00	Contents of storage register R_E recalled to displayed X-register.
5 STO I	5.00	
RCL (i)	1.23	Contents of storage register R_5 recalled to displayed X-register.

Storage register arithmetic is performed upon the contents of the register addressed by I by using STO + (i), STO − (i), STO × (i), and STO ÷ (i). Again, you can access any storage register, primary or secondary—you never have to use the P⇄S function when using the I-register for addressing. For example:

Press	Display	
1 STO + (i)	1.00	1 added to number in storage register (R_5) currently addressed by the I-register.
RCL (i)	2.23	
2 STO × (i)	2.00	
RCL (i)	4.46	
CL X	0.00	
RCL 5	4.46	

Naturally, the most effective use of the I-register as an address for STO and RCL is in a program.

Example: The following program uses a loop to place the number representing its address in storage registers R_0 through R_9, R_{S0} through R_{S9}, and R_A through R_E. During each iteration through the loop, program execution pauses to show the current value of I. When I reaches zero, execution finally is transferred out of the loop by the f DSZ I instruction and the program stops.

To key in the program:

Slide the PRGM-RUN switch PRGM ▮ RUN to PRGM.

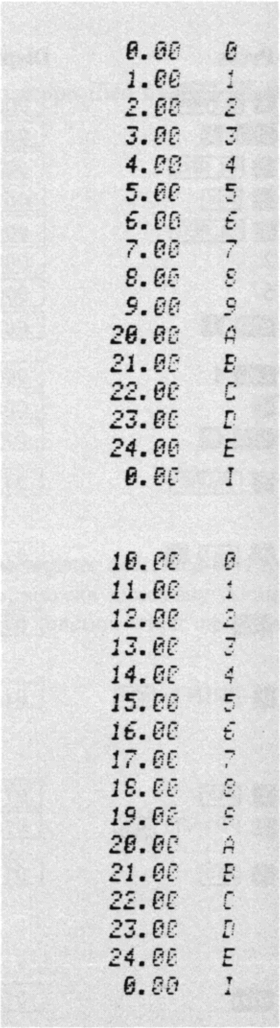
Press	Display	
f CL PRGM	000	Program initialized.
LBL A	001 21 11	
f CL REG	002 16-53	
f P⇄S	003 16-51	
f CL REG	004 16-53	
2	005 02	
5	006 05	
STO I	007 35 46	
LBL 1	008 21 01	Current value in I stored in storage register addressed by (i).
I	009 36 46	
STO (i)	010 35 45	
f PAUSE	011 16 51	Pause to display current value of I.
f DSZ I	012 16 25 46	Subtract one from value in I-register.
GTO 1	013 22 01	If I ≠ 0, execute loop again.
f PRINT: REG	014 16-13	Otherwise, print the contents of all the storage registers.
f P⇄S	015 16-51	
f PRINT: REG	016 16-13	
f P⇄S	017 16-51	Restores contents of secondary storage registers for possible later calculations.
RTN	018 24	

When the program is run, it begins by clearing the storage registers and placing 25 in the I-register. Then execution begins, recalling the current value in the I-register and storing that number in the corresponding address—for example, when the I-register contains the number 17, that number is recalled and stored in the storage register (R_{S7}) that is addressed by the number 17. Each time through the loop, the number in the I-register is decremented, and the result is used both as data and as an address by the STO (i) instruction. When the number in the I-register reaches zero, execution transfers out of the loop and the contents of all storage registers are printed.

To run the program:

Slide the PRGM-RUN switch PRGM ▮ RUN to RUN.

Press	Display
A	0.00



```
 0.00  0
 1.00  1
 2.00  2
 3.00  3
 4.00  4
 5.00  5
 6.00  6
 7.00  7
 8.00  8
 9.00  9
20.00  A
21.00  B
22.00  C
23.00  D
24.00  E
 0.00  I


10.00  0
11.00  1
12.00  2
13.00  3
14.00  4
15.00  5
16.00  6
17.00  7
18.00  8
19.00  9
20.00  A
21.00  B
22.00  C
23.00  D
24.00  E
 0.00  I
```

Notice that the contents of the I-register have been decremented to zero.

You do not have to address secondary storage registers R_{S0} through R_{S9} indirectly by using STO (i) and RCL (i). In some cases, in fact, using the P⇄S function in conjunction with STO (i) and RCL (i) can be a powerful programming tool, since you can use the same instructions to process two sets of data.

For example, suppose you had quantities A_1, A_2, A_3, A_4, A_5 stored in primary storage registers R_1 through R_5, and quantities B_1, B_2, B_3, B_4, and B_5 stored in secondary storage registers R_{S1} through R_{S5}. If you wanted to find the average value of $\frac{A_1}{B_1} + \frac{A_2}{B_2} + \ldots \frac{A_n}{B_n}$ (where $n = 5$, in this case) you could use RCL (i) and DSZ I in conjunction with the P⇄S function as shown in the program below.

To key in the program:

Slide the PRGM-RUN switch PRGM ▥ RUN to PRGM.

Press	**Display**	
f CLPRGM	000	
LBL C	001 21 13	
5	002 05	Sets number of iterations through loop.
STO I	003 35 46	
0	004 00	
STO 0	005 35 00	
LBL 8	006 21 08	
RCL (i)	007 36 45	A_n and B_n brought into Y- and X-registers and printed.
f PRINT: SPACE	008 16–11	
PRINT x	009 –14	
f P⇄S	010 16–51	
RCL (i)	011 36 45	
PRINT x	012 –14	
÷	013 –24	
f P⇄S	014 16–51	Original contents of secondary storage registers restored to those registers.
STO + 0	015 35–55 00	Total stored and updated in register R_0.

Press	Display	
f DSZ I	016 16 25 46	
GTO 8	017 22 08	If, after decrementing, I has not reached zero, execute the loop again.
RCL 0	018 36 00	Otherwise, compute, format, and print the average and stop.
5	019 05	
÷	020 -24	
DSP 9	021 -63 09	
f PRINT: SPACE	022 16-11	
PRINT x	023 -14	
DSP 2	024 -63 02	
RTN	025 24	

Now run the program for the following values of *A* and *B*.

A	73	81	97.6	115.9	244.8
B	21	47	68	102.88	179

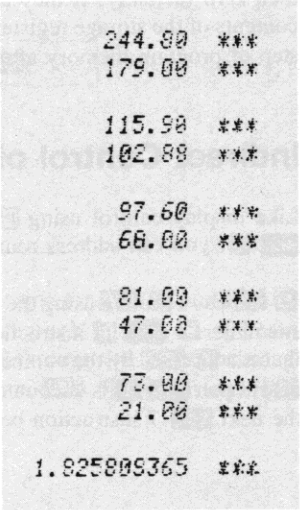
First initialize the program by placing the values for *B* in secondary storage registers R_{S1} through R_{S5} and the values for *A* in corresponding primary registers R_1 through R_5. To initialize and run the routine:

Slide the PRGM-RUN switch PRGM RUN to RUN.

Press	Display
21 STO 1	21.00
47 STO 2	47.00
68 STO 3	68.00
102.88 STO 4	102.88
179 STO 5	179.00
f P⇄S	179.00
73 STO 1	73.00
81 STO 2	81.00
97.6 STO 3	97.60
115.9 STO 4	115.90
244.8 STO 5	244.80

Now press C to run the program and print the data and the average.

Press	Display
C	1.83



```
    244.00  ***
    179.00  ***

    115.90  ***
    102.80  ***

     97.60  ***
     66.00  ***

     81.00  ***
     47.00  ***

     73.00  ***
     21.00  ***

1.825809365  ***
```

Although for this illustration we stored the data manually before beginning, it would be a simple matter to create an initialization routine that, when loaded into the calculator, would permit you to key in data during a PAUSE instruction. The routine could use the STO (i) function to store the original data in the proper registers as you keyed it in.

Indirect Incrementing and Decrementing of Storage Registers

In section 11, you learned how to increment or decrement the I-register by using the instructions ISZ I and DSZ I. By using the number in the I-register as an *address,* the instructions f ISZ (i) and f DSZ (i) increment or decrement the contents of the *storage register* addressed by the number in I.

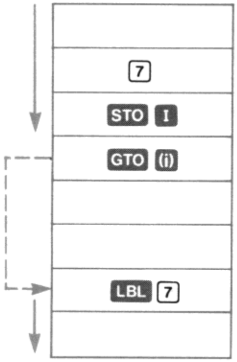
The indirect addressing of the storage registers for ISZ (i) and DSZ (i) is the same as that for STO (i), RCL (i), and storage register arithmetic using (i). When using ISZ (i) and DSZ (i), the calculator looks at only the integer portion of the absolute value of the number stored in the I-register. An attempted ISZ (i) or DSZ (i) operation when the number in I is 26 or greater results in an error condition.

ISZ (i) and DSZ (i) function very similarly to ISZ I and DSZ I. When an ISZ (i) or DSZ (i) instruction is performed in a running program, the calculator first increments (adds 1 to) or decrements (subtracts 1 from) the contents of the storage register addressed by the number in the I-register. If the contents of the storage register addressed by the number in I are then zero (actually, if they are between −1 and +1), the calculator skips one step. If the contents of the storage register addressed are *not* then zero, execution continues with the next step of program memory after the ISZ (i) or DSZ (i) instruction.

Indirect Control of Branches and Subroutines

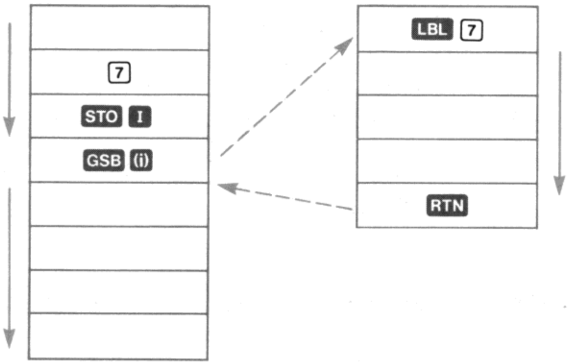
Like display control using DSP (i) and addressing of storage registers using STO (i) and RCL (i), you can address routines, subroutines, even entire programs with the I-register.

To address a routine using the I-register, use the instruction GTO (i). When a running program encounters a GTO (i) instruction, execution is transferred sequentially downward to the LBL that is addressed by the number in the I-register. Thus, with the number 7 stored in I, when the instruction GTO (i) is encountered, execution is transferred downward in program memory to the next LBL 7 instruction before resuming.



Naturally, you can also press GTO (i) from the keyboard to begin execution from the specified LBL.

Subroutines can also be addressed and utilized with the I-register. When GSB (i) is executed in a running program (or pressed from the keyboard), execution transfers to the specified LBL and executes the subroutine. When a RTN is encountered, execution transfers back to the next instruction after the GSB (i) and resumes. For example, with the number 7 stored in the I-register, GSB (i) causes execution of the subroutine defined by LBL 7 and RTN.



The simple-to-remember addressing using the I-register is the same for GTO (i) and GSB (i). If the I-register contains zero or a positive number from 1 through 9, GTO (i) addresses LBL 0 through LBL 9. When the number in I is a positive 10 through 14, LBL A through LBL E are addressed, while positive 15 through 19 address LBL f a through LBL f e. Label addressing is illustrated below.

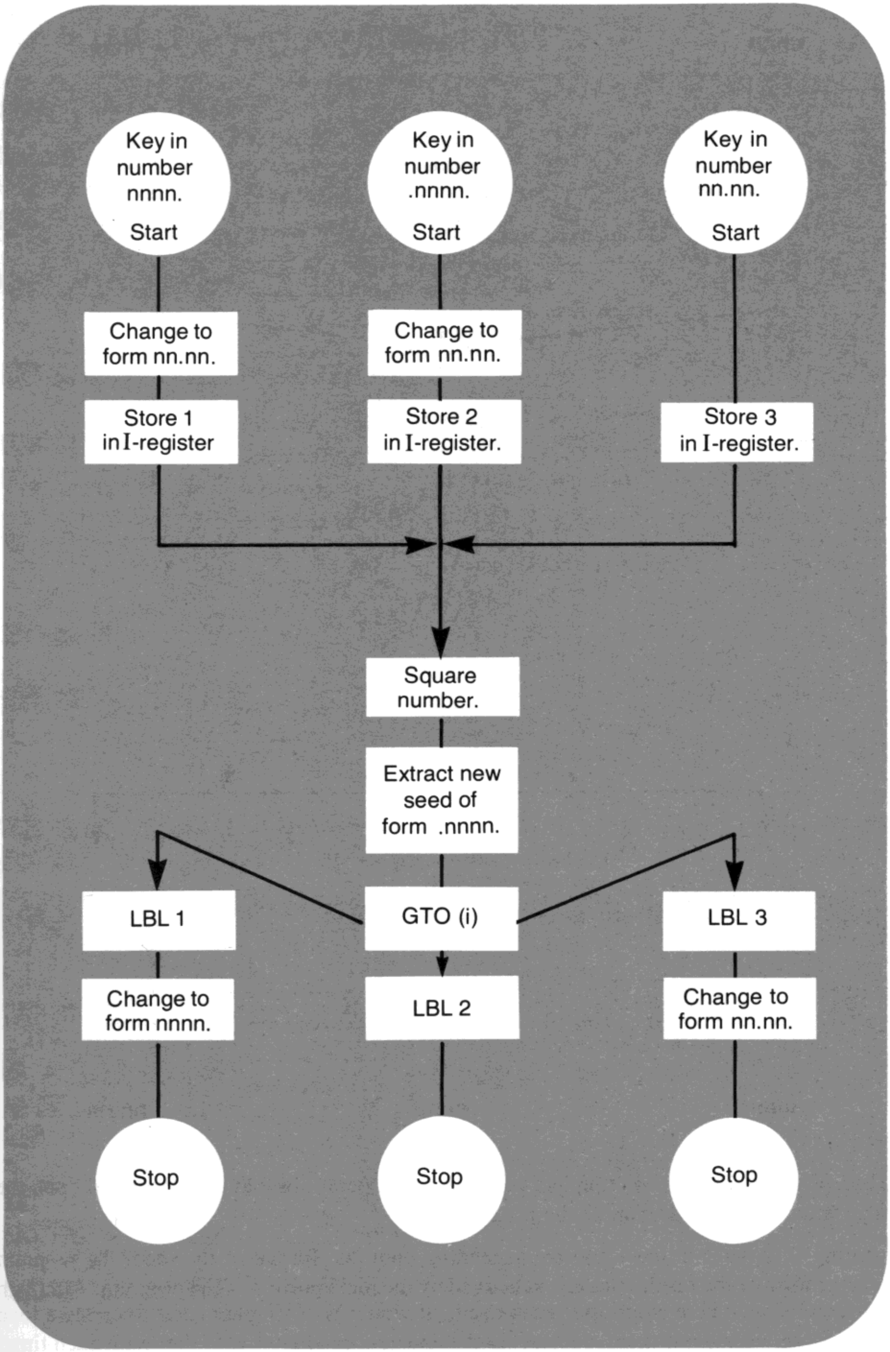
If the number in I is:	GTO (i) or GSB (i) transfers execution to:
0	LBL 0
1	LBL 1
2	LBL 2
3	LBL 3
4	LBL 4
5	LBL 5
6	LBL 6
7	LBL 7
8	LBL 8
9	LBL 9
10	LBL A
11	LBL B
12	LBL C
13	LBL D
14	LBL E
15	LBL f a
16	LBL f b
17	LBL f c
18	LBL f d
19	LBL f e

Remember that the numbers in the I-register must be positive or zero (negative numbers cause rapid reverse branching, which we will discuss later), and that the calculator looks at only the integer portion of the number in I when using it for an address.

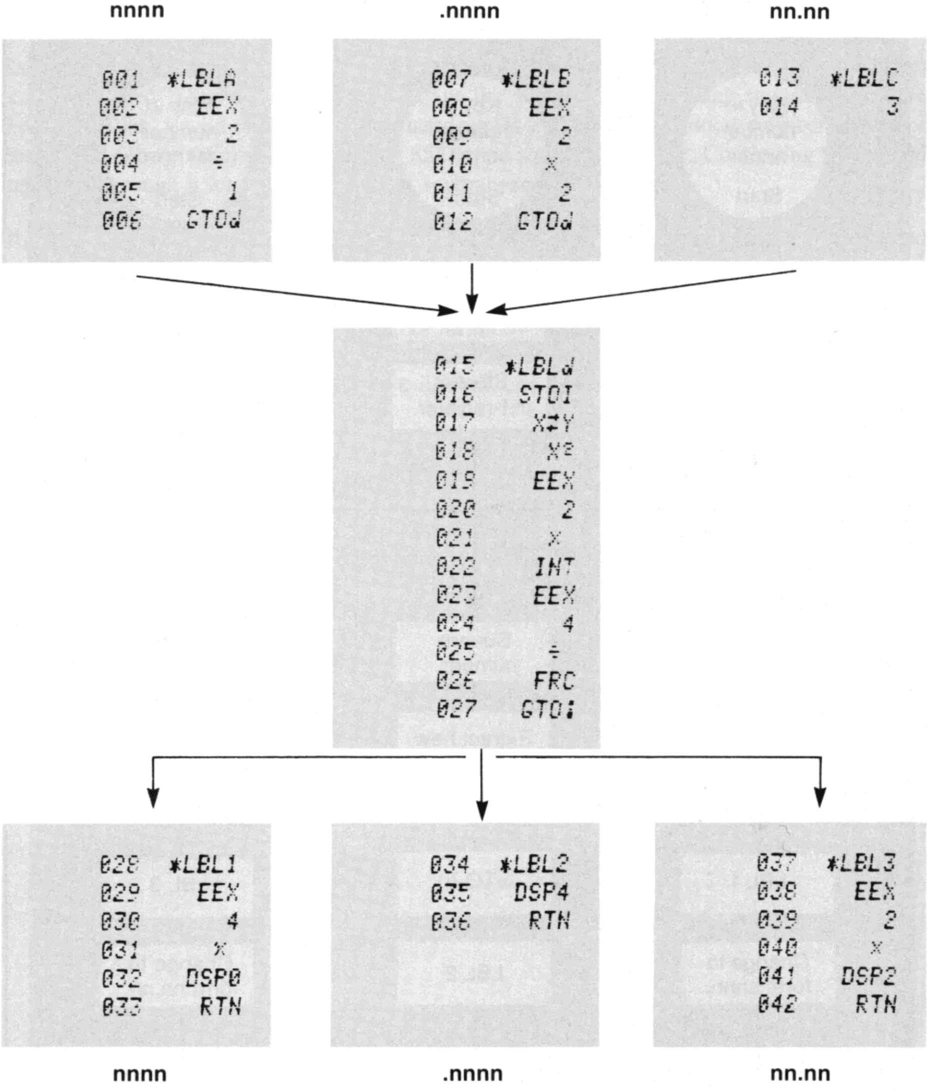
Example: One method of generating pseudo random numbers in a program is to take a number (called a "seed") square it, and then remove the center of the resulting square and square *that*, etc. Thus, a seed of 5182 when squared yields 26853124. A random number generator could then extract the four center digits, 8531, and square that value. Continuing for several iterations through a loop would generate several random numbers.

The following program uses the GTO (i) instruction to permit you to key in a four-digit seed in any of three forms: *nnnn, .nnnn,* or *nn.nn*. The seed is squared and the square truncated by the main part of the program, and the resulting four-digit random number is displayed in the form of the original seed: *nnnn, .nnnn,* or *nn.nn*.

A flowchart for the program might look like the one on page 217.


Key in number nnnn. Start
Key in number .nnnn. Start
Key in number nn.nn. Start
Change to form nn.nn.
Change to form nn.nn.
Store 1 in I-register
Store 2 in I-register.
Store 3 in I-register.
Square number.
Extract new seed of form .nnnn.
LBL 1
GTO (i)
LBL 3
Change to form nnnn.
LBL 2
Change to form nn.nn.
Stop
Stop
Stop

You can see how actual implementation of the program matches the flowchart.



```
001  *LBLA
002    EEX
003      2
004      ÷
005      1
006   GTOd
```

```
007  *LBLB
008    EEX
009      2
010      ×
011      2
012   GTOd
```

```
013  *LBLC
014      3
```

```
015  *LBLd
016   STOI
017    X⇄Y
018     X²
019    EEX
020      2
021      ×
022    INT
023    EEX
024      4
025      ÷
026    FRC
027   GTOi
```

```
028  *LBL1
029    EEX
030      4
031      ×
032   DSP0
033    RTN
```

```
034  *LBL2
035   DSP4
036    RTN
```

```
037  *LBL3
038    EEX
039      2
040      ×
041   DSP2
042    RTN
```

The use of the GTO (i) instruction lets you select the operations that are performed upon the number after the main portion of the program.

By storing 1, 2, or 3 in the I-register depending upon the format of the seed, the program selects the form of the result after it is generated by the main portion of the program. Although the program shown here stops after each result, it would be a simple matter to create a loop that would iterate several times, increasing the apparent randomness of the result each time.

To key in the complete program:

Slide the PRGM-RUN switch PRGM ▯■ RUN to PRGM.

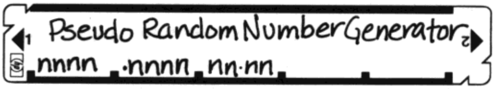
Press	Display	
f CLPRGM	000	
LBL A	001 21 11	
EEX	002 -23	Changes *nnnn* to *nn.nn*.
2	003 02	
÷	004 -24	
1	005 01	Places 1 in X-register for storage in I.
GTO f d	006 22 16 14	
LBL B	007 21 12	Changes *.nnnn* to *nn.nn*.
EEX	008 -23	
2	009 02	
×	010 -35	
2	011 02	Places 2 in X-register for storage in I.
GTO f d	012 22 16 14	
LBL C	013 21 13	
3	014 03	Places 3 in X-register for storage in I.
LBL f d	015 21 16 14	
STO I	016 35 46	Stores address of later operation in I.
x⇄y	017 -41	Brings *nn.nn* to X-register.
x^2	018 53	Squares *nn.nn*.
EEX	019 -23	Truncates two final digits of square.
2	020 02	
×	021 -35	
f INT	022 16 34	
EEX	023 -23	Truncates two leading digits of square.
4	024 04	
÷	025 -24	
f FRAC	026 16 44	
GTO (i)	027 22 45	Transfers execution to appropriate operational routine.

Press	Display	
LBL 1	028 21 01	Result appears as *nnnn.*
EEX	029 -23	
4	030 04	
×	031 -35	
DSP 0	032 -63 00	
RTN	033 24	
LBL 2	034 21 02	Result appears as *.nnnn.*
DSP 4	035 -63 04	
RTN	036 24	
LBL 3	037 21 03	Result appears as *nn.nn.*
EEX	038 -23	
2	039 02	
×	040 -35	
DSP 2	041 -63 02	
RTN	042 24	

We could also have used a subroutine for the digits for 100 (that is, EEX 2) in steps 002-003, 008-009, 019-020, and 038-039, but we have used this more straightforward program to illustrate the use of the GTO (i) instruction.

When you key in a four-digit seed number in one of the three formats shown, an address (1, 2, or 3) is placed in the I-register. This address is used by the GTO (i) instruction in step 027 to transfer program execution to the proper routine so that the new random number is seen in the same form as the original seed.

Were you to record this program on a magnetic card, you might wish to mark your card so that it looked like this:



Now run the program for seeds of 5182, .5182 and 51.82. To run the program:

Slide the PRGM-RUN switch PRGM ▮▯ RUN to RUN.

Press	Display	
5182 A	8531.	Random number generated in the proper form.
.5182 B	0.8531	
51.82 C	85.31	

The program generates a random number of the same form as the seed you keyed in. To use the random number as a new seed (simulating the operation of an actual random number generator, in which a loop would be used to decrease the apparent predictability of each succeeding number), continue pressing the appropriate user-definable key:

Press	Display	
C	77.79	Each succeeding number appears to be more random.
C	51.28	
C	29.63	

With a few slight modifications of the program, you could have used a GSB (i) instruction instead of a GTO (i) instruction.

Rapid Reverse Branching

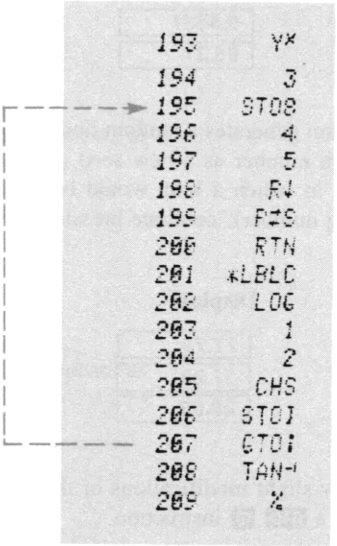
Using GTO (i) and GSB (i), with a negative number stored in I, you can actually branch to any *step number* of program memory.

As you know, when a GTO or GSB instruction is executed, the calculator does not execute further instructions until it has searched downward through program memory and located the next *label* addressed by GTO or GSB. When GTO (i) or GSB (i) is executed in a running program, with 0 or a positive 1 through 19 stored in the I-register, the running program searches downward through program memory until it locates the next LBL addressed by the number in I. Then execution resumes.

With a *negative* number stored in the I-register, however, execution is actually transferred *backward* in program memory when GTO (i) or GSB (i) is executed. The calculator does not search for a label, but instead transfers execution *backward* the number of steps specified by the negative number in the I-register. (This is advantageous because the search is often much faster than searching for a label, and because you can thus transfer execution even though all labels in the calculator have been used for other purposes.)

For example, in the section of program memory shown below, −12 is stored in the I-register. Then, when step 207, GTO (i), is executed, the running program jumps backward 12 steps through program memory to step 195 (that is, step 207 − 12 = 195), and execution resumes again with step 195 of program memory.

With −12 stored in I, execution transferred backward 12 steps by GTO (i).

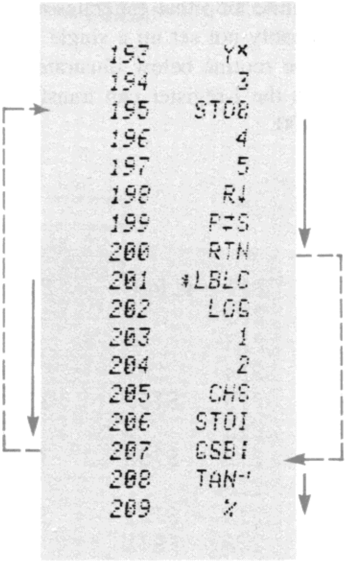


```
193      Y^X
194        3
195     STO8
196        4
197        5
198       R↓
199      P⇄S
200      RTN
201    *LBLC
202      LOG
203        1
204        2
205      CHS
206     STOI
207     GTOi
208    TAN-1
209        %
```

When GTO (i) has been performed in a running program, execution then continues until the next RTN or R/S instruction is encountered, whereupon the running program stops. Thus, if you pressed C with the instructions shown above loaded into the calculator, the instructions in steps 201 through 207 would be executed in order. Then the program would jump backward and execute step 195 next, continuing with 196, 197, etc., until the RTN instruction was encountered in step 200. The running program would then stop.

With a negative number stored in the I-register, GSB (i) also transfers execution backward the number of steps specified by I. However, subsequent instructions are then executed as a *subroutine,* so when the next RTN instruction is encountered, execution transfers back to the instruction following the GSB (i) instruction (just like a normal subroutine would be executed).

The section of program memory on the next page shows how GSB (i) operates. If you press C, −12 will be stored in the I-register. When GSB (i) is then executed a running program jumps back 12 steps from step 207 and resumes execution with step 195. When the RTN *(return)* instruction in step 200 is encountered, execution returns and continues with step 208.



```
193     Yˣ
194      3
195   STO8
196      4
197      5
198     R↓
199    P⇄S
200    RTN
201  *LBLC
202    LOG
203      1
204      2
205    CHS
206   STOI
207   GSBi
208   TAN⁻¹
209      %
```

With –12 stored in I, execution transferred backward 12 steps by GSB (i).

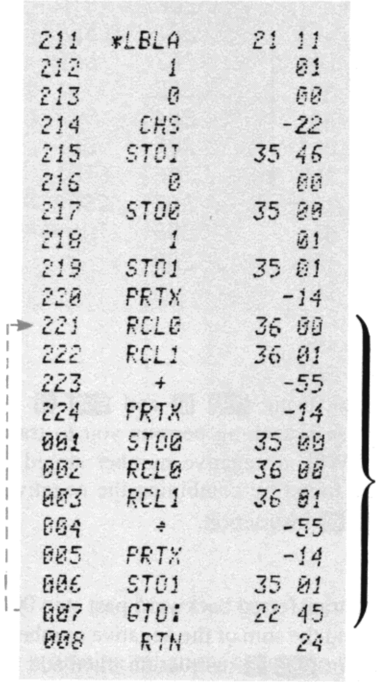
Then the RTN instruction causes a return, and execution resumes with step 208.

Rapid reverse branching using GTO (i) and GSB (i) is extremely useful as part of your programs. Rapid reverse branching permits you to transfer execution to *any* step number of program memory. With a negative number stored in the I-register, the resulting step number can always be found by combining the negative number in I with the step number of the GTO (i) or GSB (i) instruction.

Execution can even be transferred backward past step 000. To find the resulting step number of program memory, find the sum of the negative number in the I-register and the step number containing the GTO (i) or GSB (i) instruction, then add 224. Thus, if the I-register contained –12 and a GTO (i) instruction were encountered in step 007, execution would be transferred to step 219 of program memory (7–12 + 224 = 219).

Example: Named after a 13th-century mathematician, the Fibonacci series is a series of numbers that expresses many relationships found in mathematics, architecture, and nature. (For example, in many plants, the proliferation of branches follows a series of Fibonacci numbers.) The series is of the form 0, 1, 1, 2, 3, 5, 8, 13..., where each element is the sum of the two preceding elements.

The program below contains an infinite loop that generates and prints the Fibonacci series. Although you normally would probably not set up a single routine that began in step 211 and continued through step 008, the routine below illustrates how the GTO (i) instruction coupled with a negative number in the I-register can transfer program execution back in program memory, even past step 000.



```
211  *LBLA      21 11
212      1         01
213      0         00
214    CHS        -22
215   STOI      35 46
216      0         00
217   STO0      35 00
218      1         01
219   STO1      35 01
220   PRTX        -14
221   RCL0      36 00
222   RCL1      36 01
223      +        -55
224   PRTX        -14
001   STO0      35 00
002   RCL0      36 00
003   RCL1      36 01
004      ÷        -55
005   PRTX        -14
006   STO1      35 01
007   GTOi      22 45
008    RTN         24
```

Execution transferred −10 steps.

Infinite loop.

When the program is run, steps 212 through 215 store −10 in the I-register. Thereafter, execution of the GTO (i) instruction in step 007 causes the running program to jump back 10 steps and resume execution with step 221 (that is, 007 − 10 + 224 = 221). Thus, an infinite loop is set up that generates and prints the Fibonacci series until you stop the program by pressing R/S (or any key) from the keyboard.

To load the complete program, you must first load the instructions in steps 001 through 008, then go to step 210 and load the instructions into steps 211 through 224. To load the program into the calculator:

Slide the PRGM-RUN switch **PRGM** ▮ **RUN** to PRGM.

Press	**Display**
f CLPRGM	000
STO 0	001 35 00
RCL 0	002 36 00
RCL 1	003 36 01
+	004 -55
PRINT x	005 -14
STO 1	006 35 01
GTO (i)	007 22 45
RTN	008 24

Now go to step 210 and continue loading instructions, beginning with the LBL A contained in step 211:

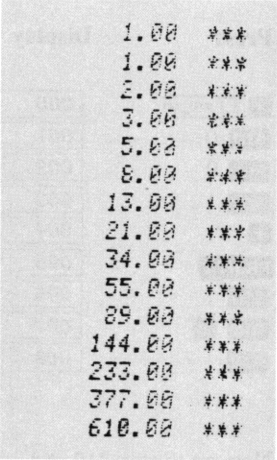
Press	**Display**
GTO .210	210 51
LBL A	211 21 11
1	212 01
0	213 00
CHS	214 -22
STO I	215 35 46
0	216 00
STO 0	217 35 00
1	218 01
STO 1	219 35 01
PRINT x	220 -14
RCL 0	221 36 00
RCL 1	222 36 01
+	223 -55
PRINT x	224 -14

Now switch to RUN mode and run the program. Press R/S (or any key) to stop the program after you have seen how quickly the Fibonacci series increases.

To run the program:

Slide the PRGM-RUN switch PRGM RUN to RUN.

Press	Display
A	
R/S	610.00



```
   1.00  ***
   1.00  ***
   2.00  ***
   3.00  ***
   5.00  ***
   8.00  ***
  13.00  ***
  21.00  ***
  34.00  ***
  55.00  ***
  89.00  ***
 144.00  ***
 233.00  ***
 377.00  ***
 610.00  ***
```

Each element in the Fibonacci series is the sum of the previous two elements in the series.

Rapid reverse branching can be specified with numbers from –1 through –999 in the I-register. If the number in I is greater than 224, the search continues backward through program memory the number of steps specified. If you attempt to execute GTO (i) or GSB (i) when magnitude of the integer portion of the negative number in I is greater than 999, the calculator displays Error .

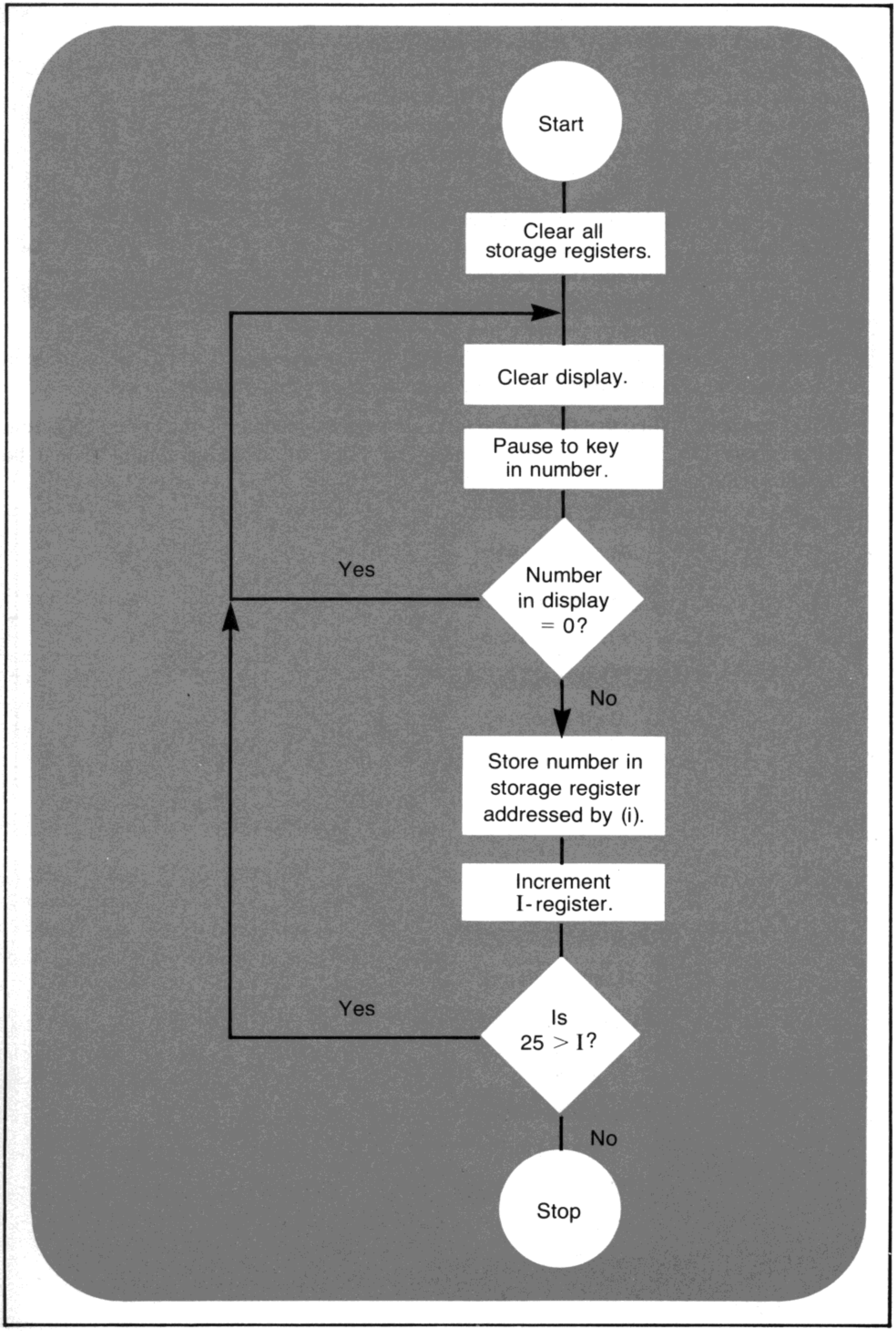
Problems

1. a. Create and load a program using ISZ I and STO (i) that permits you to key in a series of values during successive pauses. The values should be stored in storage registers R_0 through R_9, R_{S0} through R_{S9}, and R_A through R_E in the order you key them in. Use the flowchart on the opposite page to help you.

   b. Now create and load a program immediately after the first one that will recall and print the contents of each storage register in reverse order (that is, print R_E first, then R_D, etc.). The program should stop running after it has printed the contents of R_0.

   Run the program you loaded for problem 1a, keying in a series of 25 different values. Then run the program you loaded for 1b. All 25 values should be printed, but the last one you keyed in should be the first printed, etc.

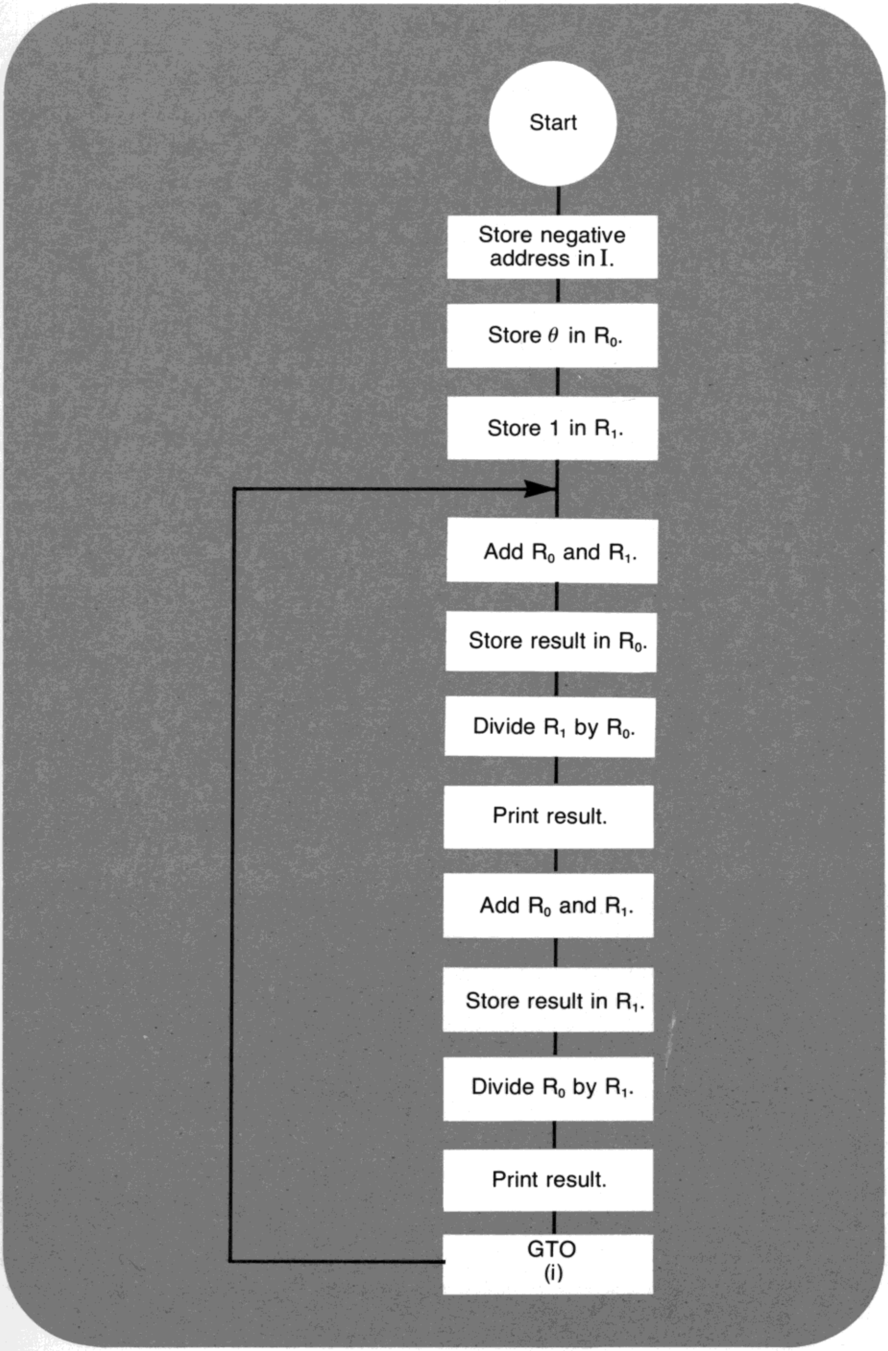
2. Modify the Random Number Generator program on pages 219-220 to use GSB (i) instead of GTO (i) for control. Run the program with the same seed numbers to ensure that it still runs correctly.

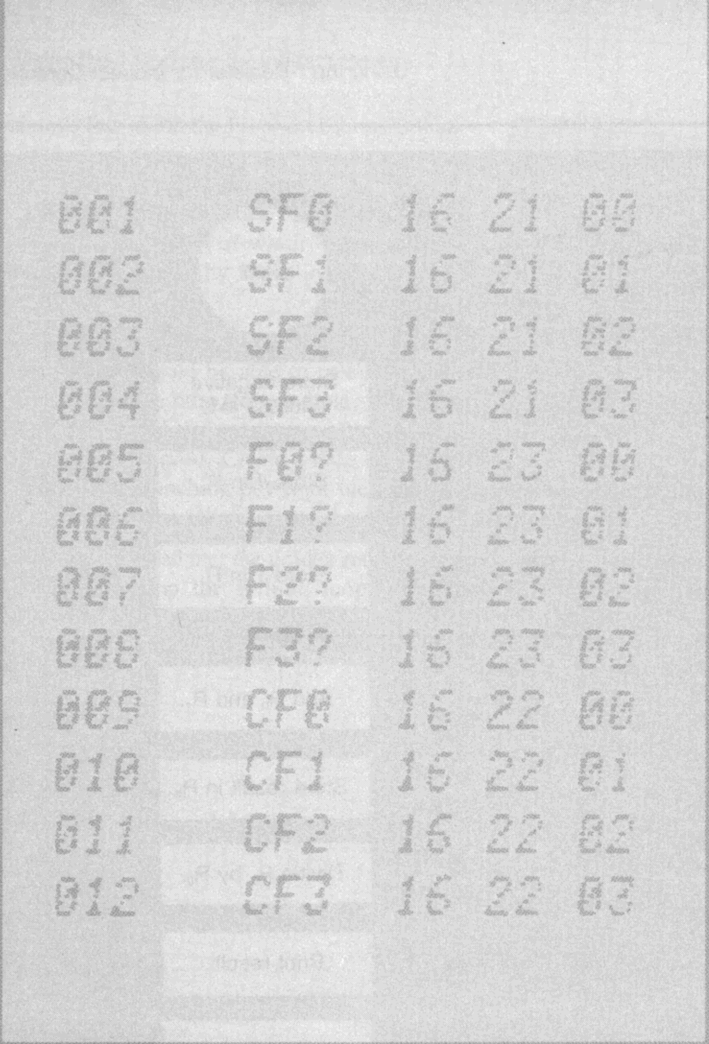

Start
Clear all storage registers.
Clear display.
Pause to key in number.
Yes
Number in display = 0?
No
Store number in storage register addressed by (i).
Increment I-register.
Yes
Is 25 > I?
No
Stop

3. One curious fact about the Fibonacci series is that the quotients of successive terms converge to a common value. This value was known to the ancient Greeks as the "golden ratio" because it expressed the ideal ratio of width to length that gave the most aesthetically appealing building or room.

   Create, load and run a program that will yield this ideal ratio. You should be able to calculate and print each successive ratio (for example, 2/3, 3/5, 5/8, 8/13, etc.) until the series converges to the value of the golden ratio. Create a loop by using the rapid reverse branching power of the **GTO** **(i)** instruction with a negative number in the I-register. Use the flowchart on the opposite page to help you.

   When you are satisfied that the golden ratio has been calculated, you can press **R/S** from the keyboard to stop the infinite loop. (The value of the golden ratio should be 0.618033989.)


Start
Store negative address in I.
Store θ in R₀.
Store 1 in R₁.
Add R₀ and R₁.
Store result in R₀.
Divide R₁ by R₀.
Print result.
Add R₀ and R₁.
Store result in R₁.
Divide R₀ by R₁.
Print result.
GTO (i)



```
001     SF0  16 21 00
002     SF1  16 21 01
003     SF2  16 21 02
004     SF3  16 21 03
005     F0?  16 23 00
006     F1?  16 23 01
007     F2?  16 23 02
008     F3?  16 23 03
009     CF0  16 22 00
010     CF1  16 22 01
011     CF2  16 22 02
012     CF3  16 22 03
```

Section 13

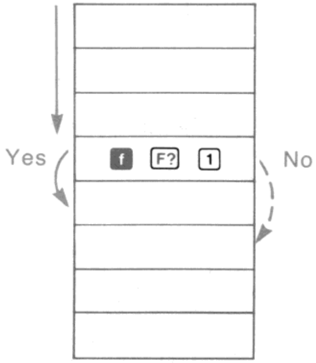
Flags

Besides the conditionals (x=y?, x>0?, etc.) and the tests for zero (ISZ (i), DSZ (i), ISZ I, DSZ I), you can also use *flags* for tests in your programs. A flag actually is a memory device that can be either SET (true) or CLEAR (false). A running program can then *test* the flag later in the program and make a decision, depending upon whether the flag was set or clear.

There are four flags, F0, F1, F2, and F3, available for use in your HP-97. To set a flag true, use the instruction STF (*set flag*) followed by the digit key (0, 1, 2, 3) of the desired flag. The instruction CLF (*clear flag*) is used to clear flags.

When using flags, decisions are made using the instruction F? (*is flag true*?) followed by the digit key (0, 1, 2, 3) specifying the flag to be tested. When a flag is tested by an f F? instruction, the calculator executes the next step if the flag is set (this is the ''DO if TRUE'' rule again). If the flag is clear, the next step of program memory is skipped before execution resumes.

Is flag F1 true?



If YES, continue execution with next step.

If NO, skip one step before resuming execution.

Command-Cleared Flags

There are two types of flags. Flags F0 and F1 are *command-cleared flags*—that is, once they have been set by an f STF 0 or f STF 1 operation, they remain set until they are commanded to change by the f CLF 0 or f CLF 1 operations. Command-cleared flags are generally used to remember program status (e.g., are printed outputs desired?).

Test-Cleared Flags

Flags F2 and F3 are *test-cleared flags*. They are cleared by a test operation. For example, if you had set flag F2 with an f STF 2 operation and then it was tested later in a program with an f F? 2 instruction, flag F2 would be cleared by the test—execution would continue with the next step of program memory (the ''DO if TRUE'' rule), but the flag would then be cleared and would remain cleared until it was set again. The test-cleared flags are used to save the f CLF operation after a test. (However, test-cleared flags can be cleared by the f CLF operation, if desired.)

Besides being a test-cleared flag, flag F3 alone is *set by digit entry*—that is, as soon as you key in a number from the keyboard, flag F3 is set. It is also set when the magnetic card reader is used to load data into the storage registers from a card.

Even though you do not test or use flag F3 in a program, it is nevertheless set by digit entry from the keyboard or data loading from the magnetic card reader.

All flags are cleared when the HP-97 is first turned ON or when f CLPRGM is pressed in PRGM mode.

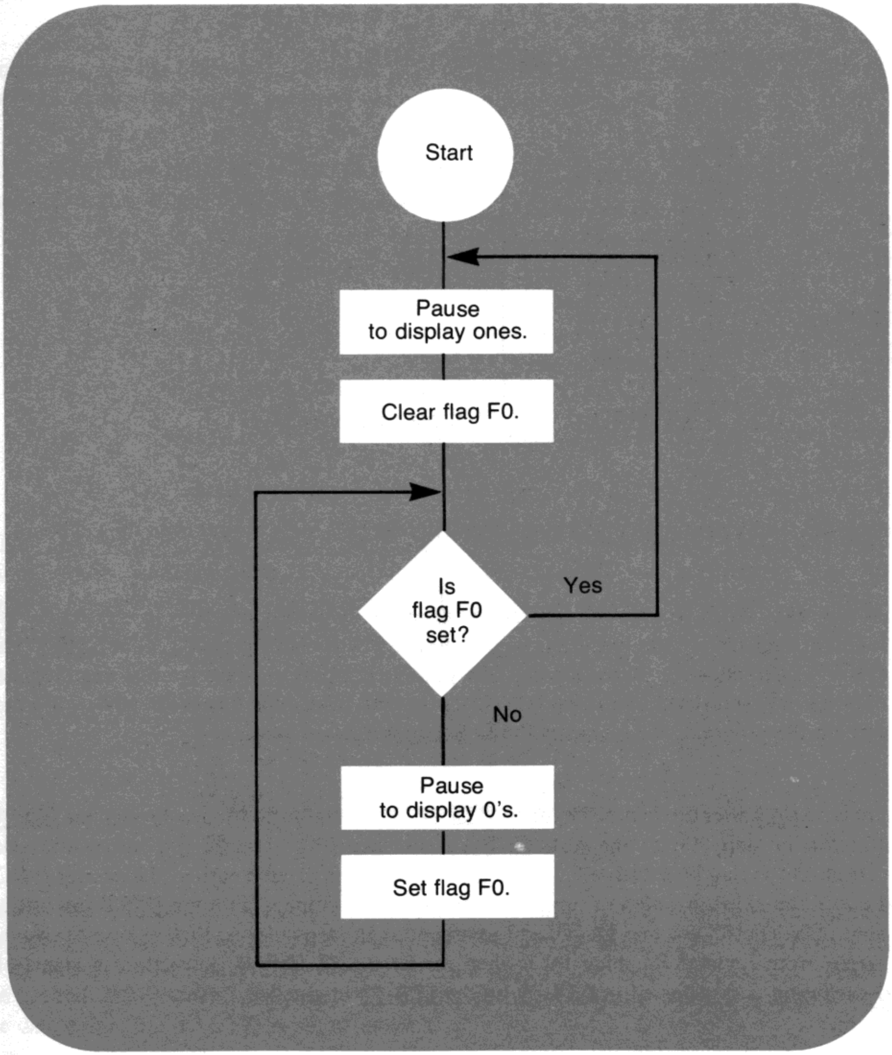
Now look at the way these flags can be used in programs.

Example: The following program contains an infinite loop that illustrates the operation of a flag. (In this case, the flag used is command-cleared flag F0.) The program alternately displays all 1's and all 0's by changing the status of the flag, and thus, the result of the test in step 007, each time through the loop. A flowchart for the simple program might look like the one on the opposite page.

The program assumes that you have the number 0 in storage register R_0 and the number 1.111111111 has been stored in storage register R_1.

Slide the PRGM-RUN switch **PRGM** ▮ **RUN** to PRGM.

Press	**Display**	
f CLPRGM	000	
LBL A	001 21 11	
DSP 9	002 -63 09	Recalls and displays ones from register R_1.
RCL 1	003 36 01	
f PAUSE	004 16 51	
f CLF 0	005 16 22 00	Clears flag F0.
LBL B	006 21 12	
f F? 0	007 16 23 00	Test flag F0.
GTO A	008 22 11	If set (true), go to LBL A.
RCL 0	009 36 00	Otherwise, recall and display zeros from register R_0, set flag F0, and go to LBL B.
f PAUSE	010 16 51	
f STF 0	011 16 21 00	
GTO B	012 22 12	
RTN	013 24	


Start
Pause
to display ones.
Clear flag F0.
Is
flag F0
set?
Yes
No
Pause
to display 0's.
Set flag F0.

Now switch to RUN mode and initialize and run the program. To run the program:

Slide the PRGM-RUN switch PRGM ▮▯ RUN to RUN.

Press	Display	
0	0.	Initializes the program.
FIX	0.00	
DSP 9	0.000000000	
STO 0	0.000000000	
1.111111111	1.111111111	
STO 1	1.111111111	
A	1.111111111	All ones and all zeros displayed alternately.
	0.000000000	

To stop the running program at any time, merely press R/S (or any key) from the keyboard.

How it works. After you have initialized the program by storing all zeros in register R_0 and all ones in register R_1, the program begins running when you press A. The RCL 1 and f PAUSE instructions in steps 003 and 004 pause to display all ones from storage register R_1. The f CLF 0 instruction in step 005 clears flag F0. (Since the flag is already clear when you begin the program, the status of the flag simply remains the same.)

There is no RTN after the routine begun by LBL A, so execution continues through the LBL B instruction in step 006 to the test, f F? 0, in step 007. The f F? instruction asks the question ''Is flag F0 set (true)?'' Since the flag has been cleared earlier, the answer is NO, and execution skips one step of program memory and continues with the RCL 0 instruction in step 009. The RCL 0 and f PAUSE instructions in steps 009 and 010 pause to display all zeros from register R_0. Flag F0 is then *set* by the f STF 0 instruction in step 011, and execution is transferred to LBL B by the GTO B instruction in step 012.

With flag F0 now set, the test f F? 0 (''Is flag F0 true?'') is now YES, so the calculator executes the GTO A instruction in step 008, the next step after the test. After again pausing to display all zeros, the flag is cleared, and the program continues in an endless cycle, alternately displaying ones and zeros, until you stop execution from the keyboard.

The above program utilized one of the two command-cleared flags, so an f CLF instruction was required to clear it each time. However, you should also be able to modify this program using one of the test-cleared flags, F2 or F3, and shorten the program, thereby saving one step of memory.

Data Entry Flag

The data entry flag, flag F3, is a flag that is set for data entry and cleared upon test. These features of this flag can be used for interchangeable solutions in a program.

Example: The program below calculates the distance *(d)*, speed *(s)*, or time *(t)* for a moving body according to the following formulas:

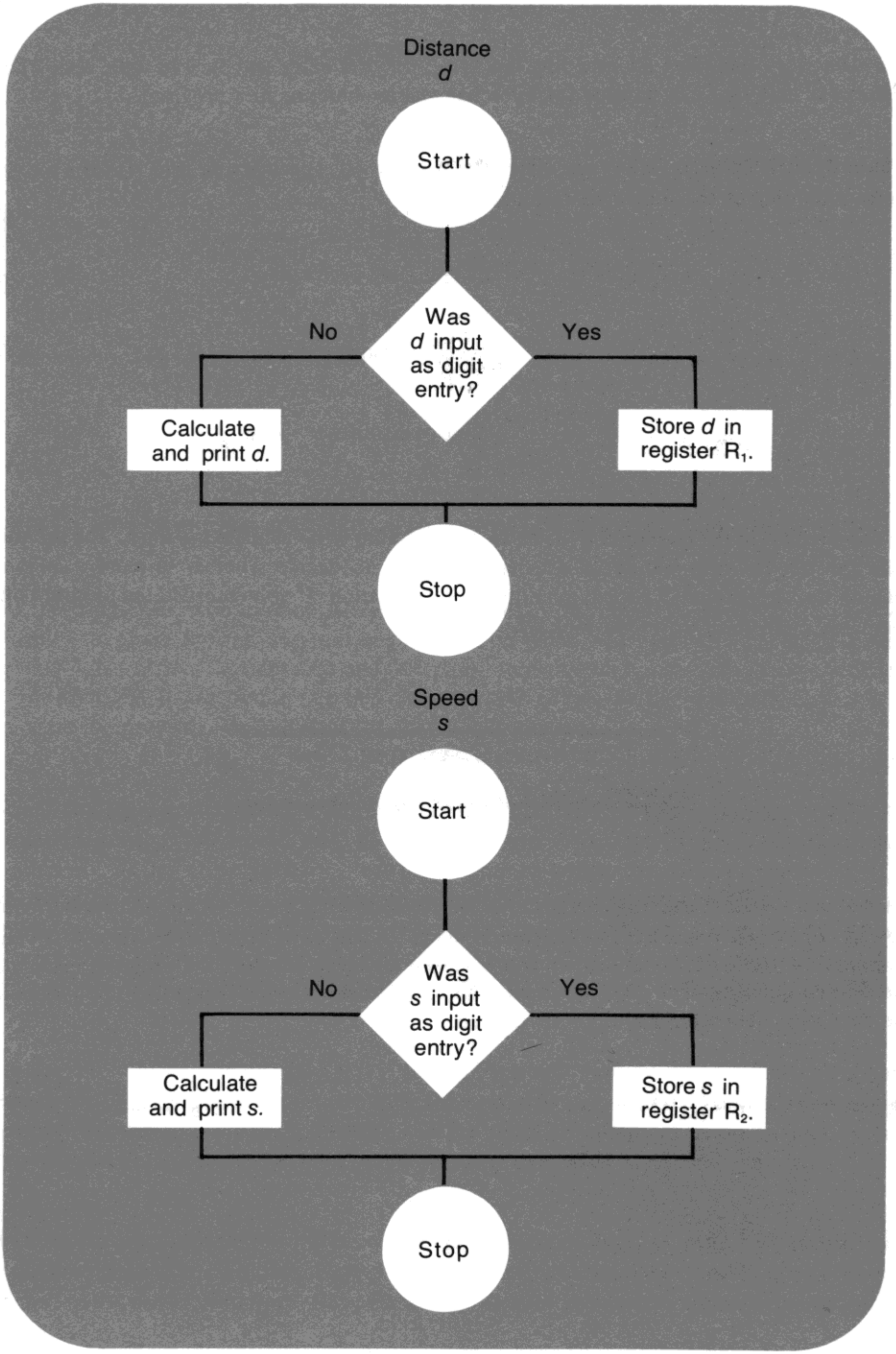
$$d = st \qquad \text{distance} = \text{speed} \times \text{time}$$

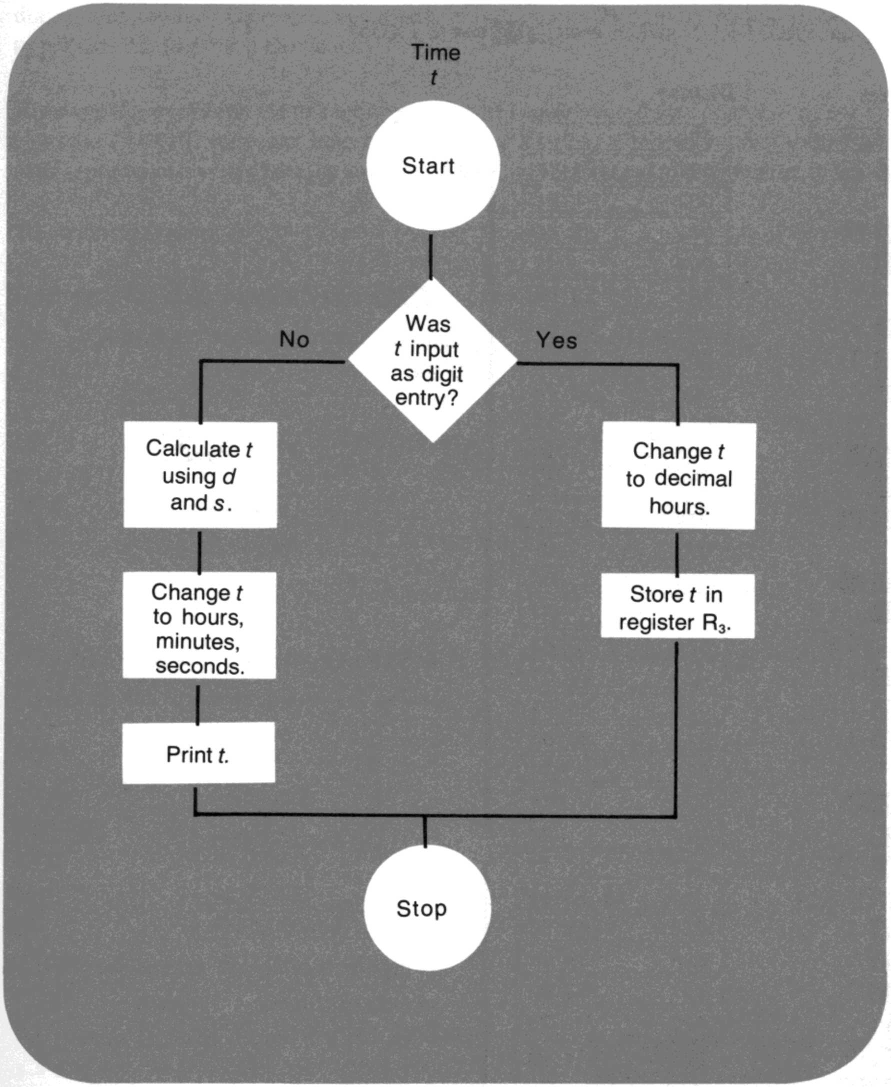
$$s = \frac{d}{t} \qquad \text{speed} = \text{distance} \div \text{time}$$

$$t = \frac{d}{s} \qquad \text{time} = \text{distance} \div \text{speed}$$

Given any two of the quantities *d*, *s*, and *t*, the program will calculate the third. The program uses the test-clearing feature of data entry flag F3 to decide whether to store a quantity away or to use previously stored quantities for calculation. If you recorded the program on a magnetic card, the card might look like this:

As you can see from the flowcharts shown on the following pages, when the user-definable key **A**, **B**, or **C** is pressed, a decision is made. If you have keyed in a value, that value is stored for further calculations. If you have not keyed in a value, the program calculates the desired quantity. The decision to store or to calculate is made depending upon whether the data entry flag, flag F3, is set or cleared.


Distance
d
Start
Was d input as digit entry?
No
Yes
Calculate and print d.
Store d in register R₁.
Stop
Speed
s
Start
Was s input as digit entry?
No
Yes
Calculate and print s.
Store s in register R₂.
Stop


Time
t
Start
Was t input as digit entry?
No
Yes
Calculate t using d and s.
Change t to hours, minutes, seconds.
Print t.
Change t to decimal hours.
Store t in register R_3.
Stop

To key in the program:

Slide the PRGM-RUN switch PRGM ▮ RUN to PRGM.

Press	Display	
f CLPRGM	000	
LBL A	001 21 11	If digit entry flag set, distance is stored. If flag is cleared, distance is calculated.
1	002 01	
STO I	003 35 46	
x⇄y	004 -41	
f F? 3	005 16 23 03	
GTO 1	006 22 01	
RCL 2	007 36 02	
RCL 3	008 36 03	
×	009 -35	
PRINT x	010 -14	
RTN	011 24	
LBL B	012 21 12	If digit entry flag set, speed is stored. If flag is cleared, speed is calculated.
2	013 02	
STO I	014 35 46	
x⇄y	015 -41	
f F? 3	016 16 23 03	
GTO 1	017 22 01	
RCL 1	018 36 01	
RCL 3	019 36 03	
÷	020 -24	
PRINT x	021 -14	
RTN	022 24	
LBL C	023 21 13	If digit entry flag set, time is stored. If flag is cleared, time is calculated.
f F? 3	024 16 23 03	
GTO 2	025 22 02	
RCL 1	026 36 01	
RCL 2	027 36 02	
÷	028 -24	
f →H.MS	029 16 35	
PRINT x	030 -14	
RTN	031 24	
LBL 1	032 21 01	Routine to store distance or speed in appropriate storage register.
STO (i)	033 35 45	
RTN	034 24	
LBL 2	035 21 02	Routine to convert time from *hours, minutes, seconds* format to decimal hours for calculation.
f H.MS→	036 16 36	
STO 3	037 35 03	
RTN	038 24	

Since the data entry flag F3 is also a test-clearing flag, it is cleared as soon as it is tested during each routine. Therefore, you do not have to use an f CLF instruction in each routine to prepare the flag for a new case.

Running the program. At this writing, the world speed record for an aircraft over a straight course is 2070.101 miles per hour by a Lockheed YF12A. Run the program to find the time at this speed that it would take the aircraft to travel the 3500 miles from New York to London.

To run the program:

Slide the PRGM-RUN switch PRGM ▮▯ RUN to RUN.

Press	Display	
DSP 6	0.000000	Initializes program.
3500 A	3500.000000	
2070.101 B	2070.101000	
C	1.412666	The time would be 1 hour, 41 minutes, 26.66 seconds.

```
1.412666  ***
```

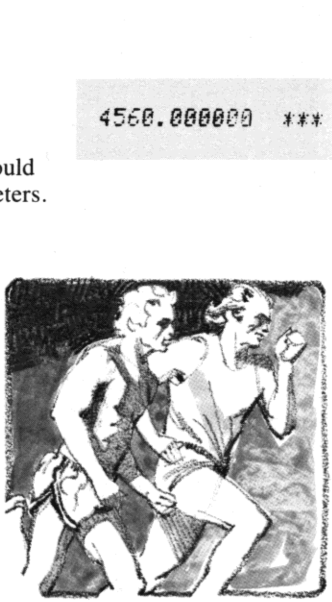
Now run the program to find out how far an automobile averaging 95 kilometers per hour could travel in 2 days.

Press	Display	
95 B	95.000000	
2 ENTER↑	2.000000	
24 ×	48.000000	
C	48.000000	
A	4560.000000	The automobile would travel 4560 kilometers.

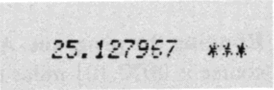
```
4560.000000  ***
```

The present Olympic record for the 1500-meter run is 3 minutes, 34.9 seconds, set at the 1968 Olympic Games by Kipchoge Keino of Kenya. What was Keino's speed in kilometers per hour?

(A kilometer is equal to 1000 meters, so key in the distance as 1.5 kilometers.)



Press	Display	
1.5 A	1.500000	Distance keyed in.
.03349 C	0.059694	Time converted to decimal hours.
B	25.127967	Keino's speed was about 25 kilometers per hour.


25.127967 ***

Notice in the above program how a flag can be used to make a decision and change the execution of a program based upon past events. Remember, too, that the status of any flag can be changed from the keyboard or from a running program.

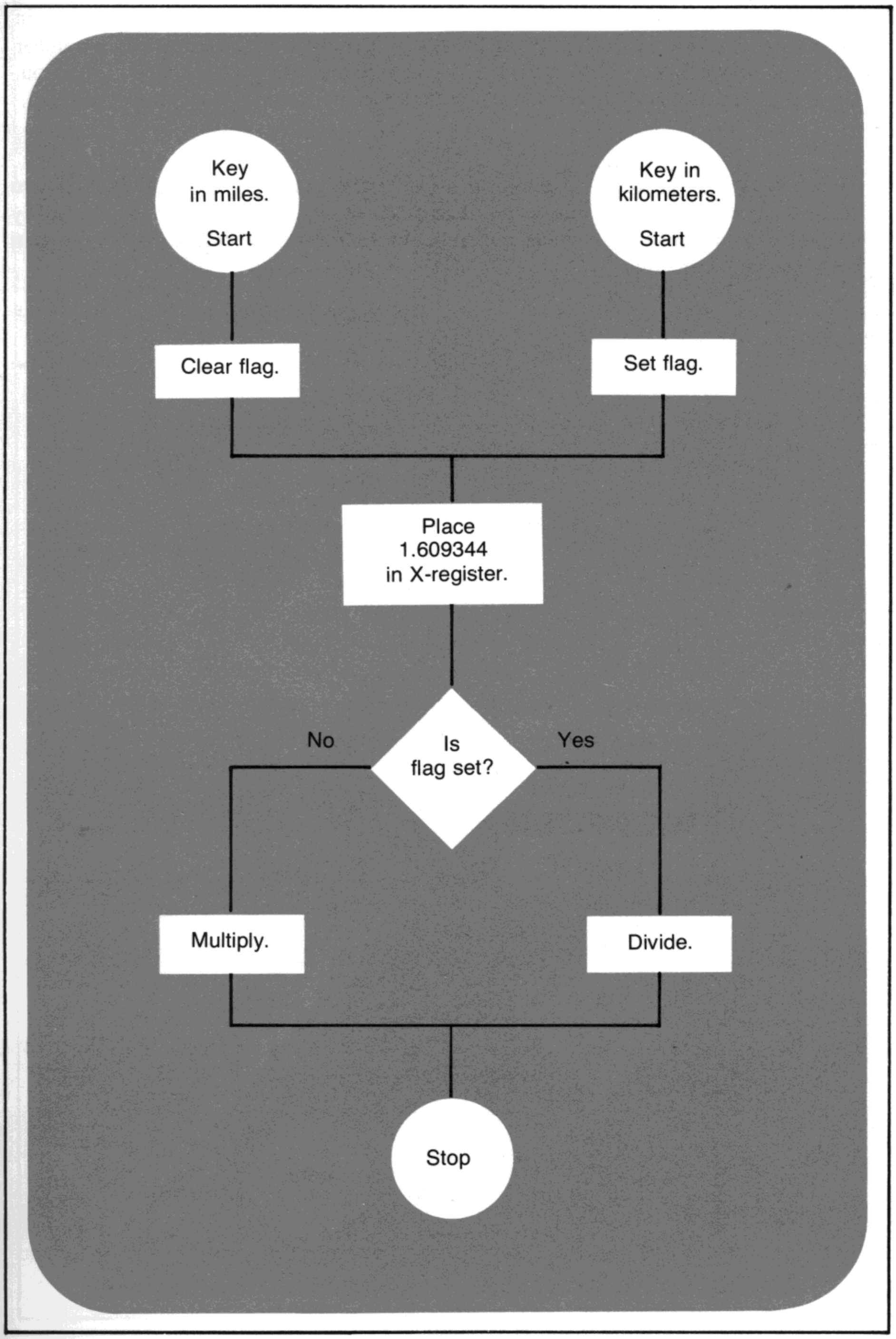
Problems

1. Modify the program on page 232 that alternately displays all zeros and all ones. Use test-clearing flag F2 or F3 instead of command-clearing flag F0. Your program should be one step shorter, since flags F2 and F3 clear when they are tested, and do not require an f CLF instruction.

2. One mile is equal to 1.609344 kilometers. Use the flowchart on the opposite page to create and load a program that will permit you to key in distance in either miles (define the routine with LBL B) or kilometers (define this routine with LBL f b) and, using a flag and a subroutine, either multiply or divide to convert from one unit of measure to the other. (Hint: × 1/x yields the same result as ÷.

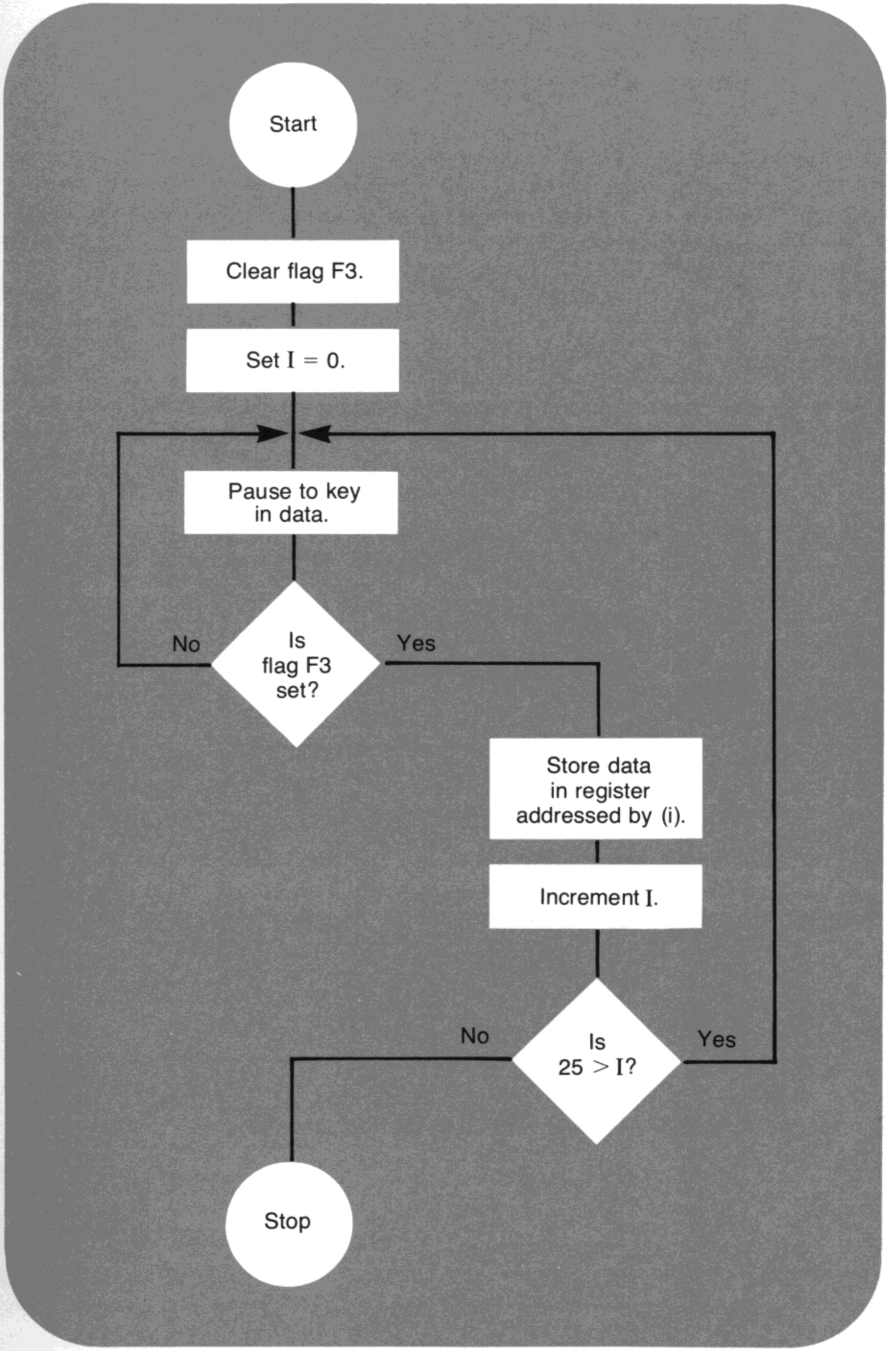
   Run the program to convert 26 miles into kilometers; to convert 1500 meters (1.5 kilometers) into miles.

   (Answers: 41.84 kilometers; 0.93 miles.)


Key
in miles.
Start
Key in
kilometers.
Start
Clear flag.
Set flag.
Place
1.609344
in X-register.
No
Is
flag set?
Yes
Multiply.
Divide.
Stop

3. Create and load a program that stores in successive storage registers values that you key in during a pause. Use the data entry flag F3 to make a decision whether to store the number or merely to wait for another input. Use the flowchart on the opposite page to help you. By using the data entry flag F3, you can key in values for zero and have them stored too.

   When you have loaded the program, run it to check its operation. You should be able to store up to 26 values (including values of zero) in succeeding storage registers. Manually recall a few random values from some of the storage registers to ensure that the program has operated correctly.


Start
Clear flag F3.
Set I = 0.
Pause to key in data.
Is flag F3 set?
No
Yes
Store data in register addressed by (i).
Increment I.
Is 25 > I?
No
Yes
Stop

```
           WDTA
           GSBA
   001    *LBLA
   002     STO9
   003        2
   004     STOI
   005    *LBL1
   006     DSZI
   007     GTO1
   005    *LBL1
   006     DSZI
   008      MRG
   009      PSE
    2.00    ***
   010    *LBLB
```

Section 14

Card Reader Operations

The programs that you have manually loaded into the HP-97 can be preserved permanently on magnetic cards. In addition, data from the storage registers can also be preserved on magnetic cards. By using magnetic cards and the card reader in your HP-97 Programmable Printing Calculator, you can increase the capability of your machine almost infinitely.

Magnetic Cards

The prerecorded magnetic cards and the blank cards that you received with your HP-97 and Standard Pac are all alike—the only difference is the information that is recorded upon them. Each card contains two sides, or tracks, where program information or data can be recorded.

Note: Whether passing side 1 or side 2 of the card through the card reader, always have the printed face of the card up.

Each side of the card is the same, and it does not matter which side is used first. In this handbook, we have adopted the convention of using side 1 first, then side 2; but as you will see, you can record onto or load from a magnetic card in any order you choose. Each side may contain either data *or* program information, but not both at once.

All magnetic cards are alike physically. Depending upon the type of information recorded upon it, however, a card may be considered a *program card*, a *data* card, or even a *mixed* card (where one side contains program information and the other side contains data).

Program Cards

Recording a Program onto a Card

A program that you have loaded into the HP-97 is not permanent—it will be lost when you turn off the calculator. You can, however, save any program permanently by recording it on a magnetic card.

To record a loaded program from program memory onto a magnetic card:

1. Set the PRGM-RUN switch PRGM ▥■ RUN to PRGM.
2. Select a blank, unprotected (unclipped) magnetic card from the packet of blank cards shipped with your HP-97.
3. Pass side 1 of the card through the card reader exactly as you did when loading a prerecorded program from the card to the calculator.

   a. If a program fills up only 112 steps or fewer of program memory, the contents of *all* of program memory (that is, the program instructions in steps 001 through 112 and the R/S instructions in steps 113 through 224) are recorded on side 1 of the card. The R/S instructions in steps 113 through 224 are in a "compressed" form. The calculator displays the current program memory step to show you that the entire program has been recorded.

   b. If the program fills up more than 112 steps of program memory (that is, if steps 113 through 224 contain instructions other than R/S) the calculator displays Crd to prompt you that another side of the card must be passed through the card reader to record the entire program. Pass the second side of the card through the card reader. The calculator then displays the current program memory step to show you that the entire program has been recorded.

4. The entire program is now recorded on the magnetic card, and also remains loaded in program memory of the calculator. The contents of the data storage registers and the stack of the calculator remain unchanged.

When you pass an unprotected card through the card reader with the PRGM-RUN switch set to PRGM, whatever program instructions or data previously recorded on the card are wiped out and replaced by the contents of the HP-97 calculator's program memory.

Besides the actual program memory step numbers and instructions, the HP-97 also records the following information on a program card on both the *first* pass and the *second* pass through the card reader:

1. The fact that a program (not data) is being recorded.
2. The fact that this is side 1 (or side 2).
3. Whether or not two passes are required.
4. Current status of flags F0, F1, F2, and F3 within the calculator.
5. Current status of trigonometric mode (i.e., DEG, RAD, or GRD) within the calculator.
6. Current display format of the calculator.
7. A checksum (a code to verify that the program is complete when it is reloaded).

All of this information is later read by the card reader when the program is reloaded back into the calculator.

If any of the required information or program memory steps are not recorded during a read, the HP-97 display will show Error to indicate that the recording of the card was not complete. Clear the error by pressing any key (the key function is not executed), then pass the same side of the card through the card reader again.

Reloading a Recorded Program from a Card

Once a program has been recorded on a magnetic card, you can reload it into the calculator any number of times. The procedure for reloading a program from a magnetic card is the same as that for loading a prerecorded program from a magnetic card into the calculator (see page 112). The position of the Print Mode switch MAN TRACE NORM does not matter when loading or recording any card.

The status information recorded on the magnetic card along with the program makes it unnecessary to load the card in any order—you can load either side 1 or side 2 first. The flag status, trigonometric mode, and display format information recorded on the program card save initialization time and program memory space because when the card is loaded, the calculator's flags, trigonometric mode, and display format are *immediately* specified according to the information on the program card.

If a program card does not read correctly, or if information on the card has been altered (perhaps by a strong magnetic field), the checksum will be wrong. When you attempt to load the program from the card into the calculator by passing it through the card reader, the calculator will display Error . You can clear the error by pressing any key. If a card read fails after a portion of the card has been loaded, that portion of the calculator's program memory which would have been altered by reading the card is cleared to R/S instructions, and the calculator display indicates Error . An error is also indicated if you attempt to load a blank magnetic card, but the contents of the calculator's program memory are preserved.

The contents of the stack and of the data storage registers in the calculator remain unchanged when a program is loaded, whether from a card or manually from the keyboard.

To clear a program that has been recorded on a magnetic card, simply load another program onto the card.

Merging Programs

Normally, whenever you load a program from a magnetic card into the calculator, that program replaces the entire contents of program memory, either with program instructions or R/S instructions. All 224 steps of program memory are replaced.

However, you can also *merge* programs in your HP-97; that is, you can add a program that is recorded on a magnetic card into the calculator, beginning with any step of program memory. When you merge a program in from a card, steps 000 through *nnn* of the original program are preserved. This feature permits you to add to or alter a program that is already loaded in the calculator.

To merge a program from a magnetic card into program memory:

1. Set the PRGM-RUN switch PRGM RUN to RUN.
2. Use the GTO · n n n operation from the keyboard to set the calculator to the last step of the loaded program that you want to save.
3. Press f MERGE (*merge*).
4. Pass one side of the magnetic card containing the new program through the card reader. If the second side of the card must also be loaded, the calculator display will prompt you with Crd .

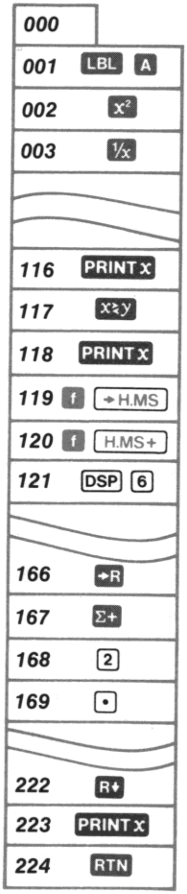
5. If the calculator display shows Crd, pass the second side of the card through the card reader. The calculator will again display the original contents of the X-register to indicate that the merged load has been completed.

When you merge a program from a card into program memory, the instructions from the card are loaded into the calculator beginning with the step of program memory following the step to which the calculator is set. Thus, if you first set the calculator to step 118 using the operation GTO . 1 1 8 from the keyboard, the first instruction from the magnetic card would be loaded into step 119, the second instruction into step 120, etc. All instructions in program memory after the merge step are replaced by instructions from the magnetic card.

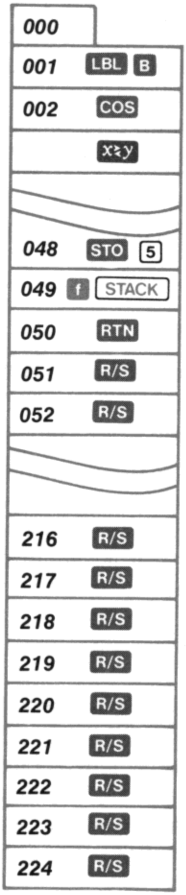
Remember, in some cases even one side of the program card may contain 224 steps (although the last 112 steps are compressed R/S instructions).

Thus, a 224-step program loaded in the calculator and a magnetic card containing a 50-step program and 174 R/S instructions might look like the illustration below:

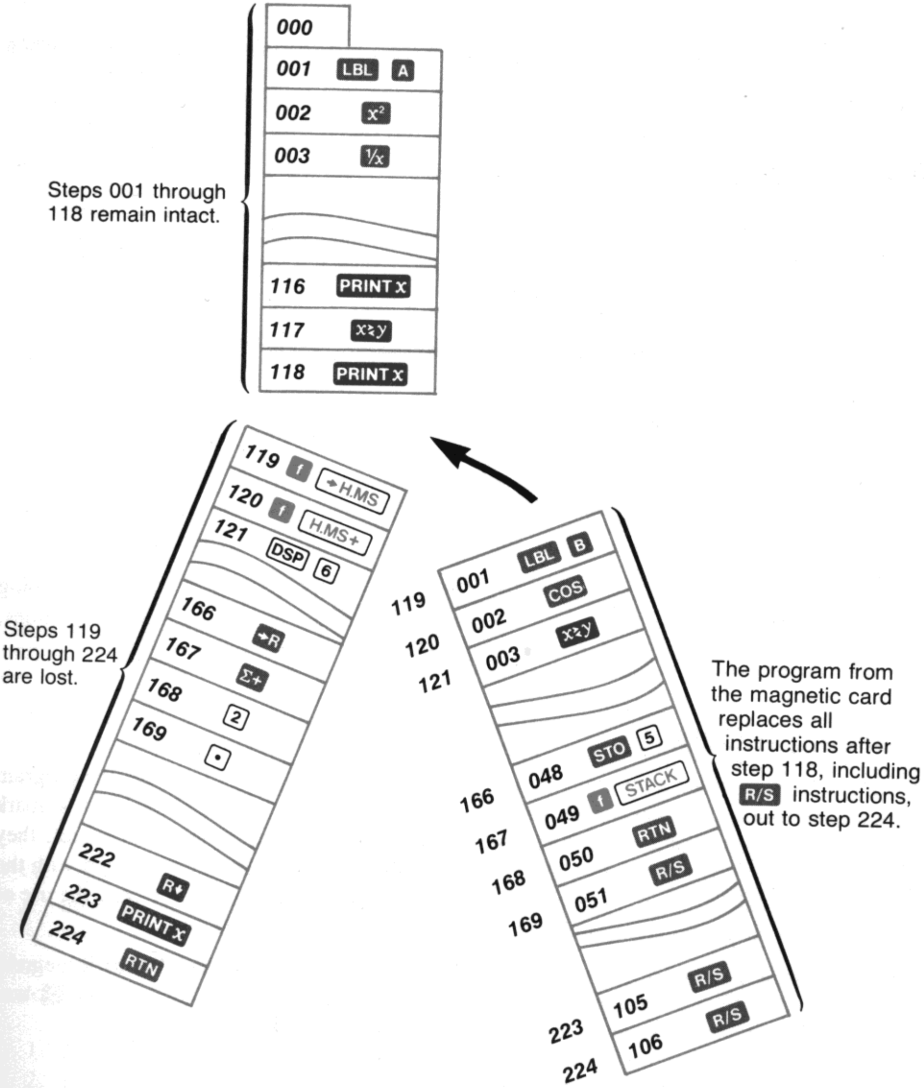
Program loaded in calculator.



Program recorded on magnetic card.



If you set the calculator to step 118, press f MERGE, and pass the card through the card reader, the instructions from the card will be merged in beginning with step 119 and continuing through step 168. (That is, 118 + 50 = 168.) All instructions after step 118 will be replaced in the original program, either by program instructions or R/S instructions from the magnetic card.


Steps 001 through 118 remain intact.

Steps 119 through 224 are lost.

The program from the magnetic card replaces all instructions after step 118, including R/S instructions, out to step 224.

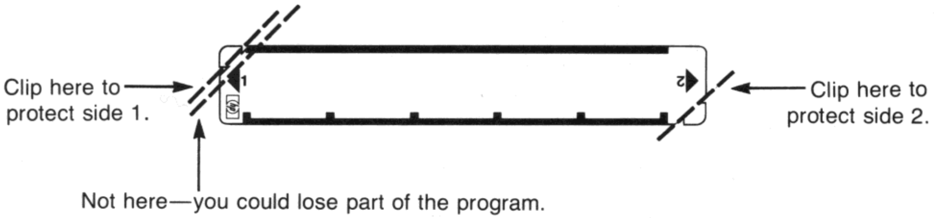
When merging a program from a magnetic card with a program already loaded into the calculator, only the instructions from the card for which there are enough steps of program memory will be loaded. Thus, in the example above, if you had merged the card from step 200 of the loaded program, only the first 24 instructions of the card would be loaded. (That is, 224 − 200 = 24.)

When merging, the instructions that are replaced in program memory by the instructions from the magnetic card are lost. The entire program on the card, of course, remains recorded there permanently, until another program (or data) is recorded upon the card.

Calculator status information (flags, display and trigonometric modes) is not changed when merging a new program with the one in program memory.

Protecting a Card

Information (whether program or data) that you have recorded upon a magnetic card can be cleared or replaced unless the card is protected. To protect a side of a recorded card, clip the notched corner of the card nearest the side you want to protect.



Clip here to protect side 1.

Clip here to protect side 2.

Not here—you could lose part of the program.

When you have protected a recorded program or recorded data on a side of a card by clipping the corner of the card, you can load that information into the HP-97 any number of times, but you will not be able to replace or add to the program or data on the card.

Marking a Card

After you have recorded a program on a card, you will probably want to assign the program a name and to mark the name onto the card. In addition, you will probably want to mark symbols onto the magnetic card so that when the card is inserted in the window slot, they will appear above the letter keys (A through E, f a through f e) associated with the labels in the program. These symbols, or mnemonics, should help you remember the use of the letter keys in the program, and they will aid in running the program.

For example, if you had written a program that would convert degrees Celsius to degrees Fahrenheit when C was pressed, and degrees Fahrenheit to degrees Celsius when D was pressed, you might wish to mark the card with the information shown here.

You can write on the non-magnetic face of a card as shown above using any writing implement that does not emboss the card. Annotating magnetic cards with a typewriter may impair the load/record properties of the cards. To permanently mark a card, you can use India ink or a permanent felt-tip pen.

Data Cards

As you know, you can record programs on magnetic cards for permanent storage, and then simply pass the card containing a program through the card reader whenever you want to run it again. You can also record *data* from the storage registers onto a magnetic card for permanent storage or for use at a later time. Then, a day, a week, a year later, simply pass the data card through the card reader to restore the original contents of the storage registers.

With this feature of the HP-97, you can store extremely large quantities of data for future use, or you can use each card to preserve a series of constants.

Recording Data onto a Card

The f W/DATA function and the card reader on your HP-97 allow you to record as much data as you wish on magnetic cards. To record data on a magnetic card:

1. Set the PRGM-RUN switch PRGM ■▯ RUN to RUN.
2. Store data in any storage register—R_0 through R_9, R_{S0} through R_{S9}, R_A through R_E, or I.
3. Press f W/DATA *(write data onto card)*. The calculator will display Crd to indicate that you are to pass a card through the card reader.
4. Select an unclipped magnetic card. Pass side 1 of the card through the card reader.
    a. The contents of the primary storage registers (R_0 through R_9, R_A through R_E, I) are recorded on side 1. If the calculator's protected secondary registers (R_{S0} through R_{S9}) all contain zero data, those contents are "compressed" and recorded on side 1 also. The calculator displays the original contents of the X-register to indicate that all data has been correctly recorded.
    b. If any of the secondary storage registers (R_{S0} through R_{S9}) contain *nonzero* data, the display shows Crd to indicate that a second side is necessary to record all the data.
    c. Pass side 2 of the card through the card reader. The actual contents of the secondary storage registers R_{S0} through R_{S9} are recorded on the second side of the card.
5. All data has now been recorded on the magnetic card. In addition, the data remains intact in the calculator.

Besides the data from the storage registers, the HP-97 also records the following information onto a data card on either or both passes through the card reader:

1. The fact that data (not a program) is being recorded.
2. The fact that this is side 1 (or side 2).
3. Whether or not two passes are required.
4. A checksum (a code to verify that the data is complete when it is reloaded).

No calculator status information is recorded when data is recorded onto a magnetic card.

When data is recorded on a side of a magnetic card, it wipes out whatever information was previously on that side. To record data permanently on a card, so that it can never be lost, you can clip a corner of the recorded magnetic card, just as you do to permanently save a recorded program.

Loading Data from a Card

To load data from a recorded card back into the storage registers, simply pass the card through the card reader with the PRGM-RUN switch set to RUN. The HP-97 identifies the type of information (whether data or a program) and automatically places it into the proper portion of the calculator. Thus, to load data back into the calculator from a magnetic card:

1. Ensure that the PRGM-RUN switch **PRGM** ■▯ **RUN** is set to RUN.
2. Select the magnetic card with data recorded upon it.
3. Pass side 1 of the magnetic card through the card reader.
   a. Data from the card has now replaced data in the 16 primary storage registers of the calculator. In addition, if the secondary registers contained all zeros when recorded, those zeros replace the contents of the secondary registers (R_{S0} through R_{S9}) of the calculator. The calculator displays the original contents of the X-register to indicate that by loading only one side, the card was loaded correctly.
   b. If a second side of the card is required, the calculator prompts you by displaying *Crd* .
   c. Pass side 2 of the card through the card reader to load nonzero data into the secondary storage registers. The calculator then displays the original contents of the X-register to indicate that the data card has been loaded completely.

It does not matter which side of the card you record or load first. The HP-97 records the contents of the primary registers on the first side recorded, and the contents of the secondary storage registers on the second side. (If all secondary registers contain only zero, the contents of all storage registers are recorded on the first side. When the card is then read later, the data is loaded into the proper registers, regardless of which side of the data card you first pass through the reader. However, for ease of later reference, it is generally best to record primary registers on side 1 and secondary registers on side 2.)

Whenever the calculator indicates *Crd* , you can clear the *Crd* display and return control to the keyboard by pressing **CL x** or any key from the keyboard. In this way, you can record only *part* of the storage registers onto a card or load only part from a card.

Neither the stack nor the contents of program memory are altered in the calculator when you record data onto or load data from a card.

Now let's store data in some of the storage registers and see how these features of your HP-97 work.

Example: Store 1.00 in primary register R_1, 2.00 in secondary register R_{S2}, and 3.00 in the I-register. Record the contents of these registers on a magnetic card, then turn the HP-97 OFF then ON. Finally, restore the data on the magnetic card to the proper registers and list the contents of the registers to verify that the data has been loaded correctly.

First, ensure that the PRGM-RUN switch PRGM ▮▯ RUN is set to RUN.

Press	Display	
f CL REG	0.00	
f P⇄S	0.00	
f CL REG	0.00	All storage registers cleared to zero initially. (Display assumes no results remain from previous examples.)
2 STO 2	2.00	
f P⇄S	2.00	2 stored in secondary storage register R_{S2}.
1 STO 1	1.00	
3 STO I	3.00	

To record the data contained in the storage registers, first select a blank and unclipped magnetic card. To then record the contents of the storage registers onto the card:

Press	Display	
f W/DATA	Crd	The calculator prompts you to insert the first side of the card.

Insert side 1 of the magnetic card into the front slot of the card reader and permit it to be passed through the reader.

Display	
Crd	After recording the first side of the card, the calculator prompts you that it has additional data to record on side 2.

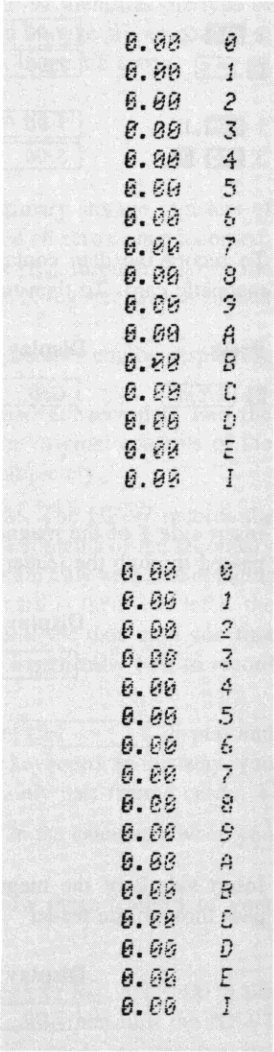
Insert side 2 of the magnetic card into the front slot of the card reader and allow it to pass through the reader.

Display	
3.00	Indicates that all data has been recorded on the magnetic card.

The data that is stored in the registers remains there now, and is also recorded on the magnetic card. Even though you turn the calculator OFF or otherwise clear the storage registers, the data is recorded upon the magnetic card for future use. For example:

Turn the HP-97 power switch OFF, then ON.

Press	**Display**	
f PRINT: REG	0.00	
f P⇄S	0.00	
f PRINT: REG	0.00	Verifies that none of the storage registers, primary or secondary, contain data.



```
0.00  0
0.00  1
0.00  2
0.00  3
0.00  4
0.00  5
0.00  6
0.00  7
0.00  8
0.00  9
0.00  A
0.00  B
0.00  C
0.00  D
0.00  E
0.00  I

0.00  0
0.00  1
0.00  2
0.00  3
0.00  4
0.00  5
0.00  6
0.00  7
0.00  8
0.00  9
0.00  A
0.00  B
0.00  C
0.00  D
0.00  E
0.00  I
```

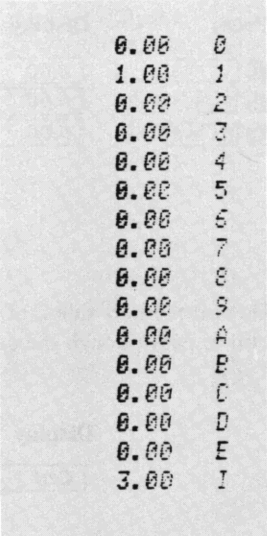
Now load the data back into the storage registers by passing the data card through the card reader. No special instructions are necessary to the HP-97—the calculator automatically recognizes that the card contains data, and reloads the data into the proper storage registers. To load the data from the card into the calculator:

Insert side 1 of the magnetic card into the front card reader slot. Allow the card to pass through the reader.

Display	
Crd	Data loaded into primary storage registers. The calculator prompts you that the card contains data for the secondary registers as well.

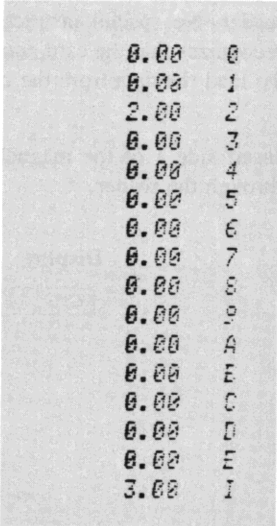
Insert side 2 of the magnetic card into the front card reader slot. Allow the card to pass through the reader.

Press	**Display**	
	0.00	Original contents of X-register again displayed to indicate that entire card has been loaded.
f PRINT: REG	0.00	



```
0.00    0
1.00    1
0.00    2
0.00    3
0.00    4
0.00    5
0.00    6
0.00    7
0.00    8
0.00    9
0.00    A
0.00    B
0.00    C
0.00    D
0.00    E
3.00    I
```

Press	Display	
f P⇄S	0.00	
f PRINT: REG	0.00	Verifies that contents of data card have been loaded into the proper storage registers.



```
0.00  0
0.00  1
2.00  2
0.00  3
0.00  4
0.00  5
0.00  6
0.00  7
0.00  8
0.00  9
0.00  A
0.00  B
0.00  C
0.00  D
0.00  E
3.00  I
```

You can see that all the data contained on the magnetic card has been loaded into the proper storage registers. The data also remains recorded on the magnetic card, and can be loaded into the calculator over and over, until you record other data or a program on that card. Data (like programs) may be loaded into the calculator from a card in any order—no matter which side you first pass through the card reader, the calculator identifies it and places the contents in the proper storage registers of the calculator.

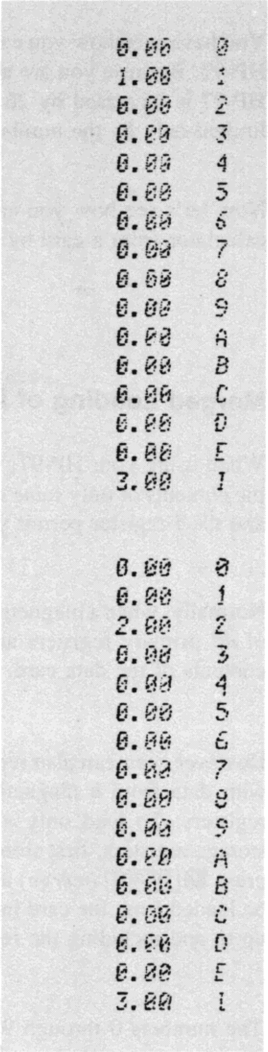
Press	Display	
f CL REG	0.00	
f P⇄S	0.00	
f CL REG	0.00	All primary and secondary storage registers again cleared to zero.

This time, insert side 2 of the magnetic card into the front card reader slot first. Allow the card to pass through the card reader.

Display	
Crd	Display indicates that the card contains more data to be loaded.

Now insert side 1 of the data card into the front card reader slot and allow it to pass through the card reader.

Press	**Display**	
	0.00	Original contents of X-register returned to indicate that contents of entire magnetic card have been loaded.
f PRINT: REG	0.00	
f P⇄S	0.00	
f PRINT: REG	0.00	Verifies that data from the magnetic card has again been loaded into the proper storage registers.



```
0.00  0
1.00  1
0.00  2
0.00  3
0.00  4
0.00  5
0.00  6
0.00  7
0.00  8
0.00  9
0.00  A
0.00  B
0.00  C
0.00  D
0.00  E
3.00  I

0.00  0
0.00  1
2.00  2
0.00  3
0.00  4
0.00  5
0.00  6
0.00  7
0.00  8
0.00  9
0.00  A
0.00  B
0.00  C
0.00  D
0.00  E
3.00  I
```

If only the primary registers contain nonzero data when you press f W/DATA, the calculator will display Crd only until you have passed *one* side of the magnetic card through the card reader. Then the calculator will display the original contents of the X-register to indicate that you again have control from the keyboard. Likewise, if data is contained on only one side (side 1 or side 2) of a magnetic card, you need load only that side.

You have seen how you can store data temporarily or permanently on magnetic cards with the HP-97. Because you are able to record data on magnetic cards, the storage capacity of your HP-97 is increased by 26 registers on each card. The amount of data you want to store is limited only by the number of cards you have!

Now let's see how you can load the contents of only *part* of the storage registers into the calculator from a card by using the I-register and the MERGE function.

Merged Loading of Data

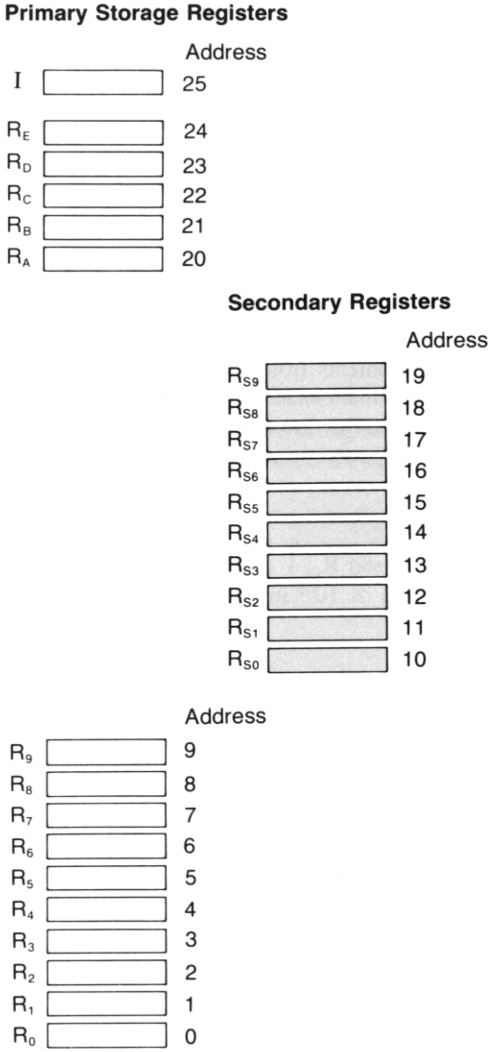
When using your HP-97, there may be occasions when you want to load into the calculator the contents of only *some* of the storage registers recorded on a card. The f MERGE function and the I-register permit you to select the number of registers that you want to load.

Normally, when a magnetic card containing data is passed through the card reader, the contents of *all* primary registers and *all* secondary registers in the calculator are replaced with the contents of the data card.

However, you can also replace the contents of *some* of the storage registers in the calculator with data from a magnetic card, while preserving the contents of the rest of the storage registers. To load only a portion of the data from a magnetic card into the calculator's storage registers, first store a number from 0 through 25 as an address in the I-register. Then press f MERGE (*merge*) and pass the card containing data through the card reader. Data will be loaded from the card into the storage registers, beginning with register R_0 and continuing up to and including the register addressed by the number in I.

The numbers 0 through 9 in I, when used as an address for merged data, refer to primary storage registers R_0 through R_9. The numbers 10 through 19 refer to secondary storage registers R_{S0} through R_{S9}, while the numbers 20 through 24 refer to registers R_A through R_E, and the number 25 addresses the I-register itself. As is normal for I-register operations, only the absolute value of the integer part of the number in I is significant in addressing. Also, if the absolute value of the number in I is 26 or greater, all registers will be read in just as in an unmerged data read.

The illustration below should refresh your memory of the addressing scheme for the storage registers:



To merge data from a magnetic card into selected storage registers:

1. Store in I the number address of the last storage register you want loaded from the card.
2. Press f MERGE (*merge*) to select the merge mode.
3. Pass either side of the magnetic card containing data through the card reader. If more data is to be loaded, the calculator will display Crd
4. If the HP-97 display shows Crd , pass the other side of the data card through the card reader.
5. Data will be loaded from the card beginning with register R_0 up to and including the storage register specified by the number in I.

Thus, if you had stored the number 7 in I, then pressed f MERGE and loaded a magnetic card containing data, the contents of the first 8 registers in the calculator (R_0 through R_7) would have been replaced by contents from the data card. The remainder of the storage registers in the calculator would remain intact. If you had stored the number 15 in I, calculator primary registers R_0 through R_9 and R_{S0} through R_{S5} (the register addressed by the number 15) would have had their contents replaced by data from the card. This includes registers containing only zero data.

Example: Store 1×10^{10} in register R_1, 1×10^{-20} in register R_9, 1×10^{30} in register R_{S5}, 1×10^{-40} in register R_{S6}, and 1×10^{50} in register R_B in the calculator. Record this data on a magnetic card.

Press	**Display**
f CL REG	0.00
EEX 30	1. 30
STO 5	1.000000000 30
EEX 40 CHS	1. -40
STO 6	1.000000000-40
f P⇄S	1.000000000-40
f CL REG	1.000000000-40
EEX 10	1. 10
STO 1	1.000000000 10
EEX 20 CHS	1. -20
STO 9	1.000000000-20
EEX 50	1. 50
STO B	1.000000000 50

Now record this data on a magnetic card. You can record it onto any unclipped magnetic card—it will replace whatever is on the card with the contents of the storage registers.

Press	Display	
f W/DATA	Crd	Calculator prompts you to insert a magnetic card.

Pass side 1 of the card through the card reader.

Display
Crd

Now pass side 2 of the magnetic card through the card reader.

Display
1.000000000 50

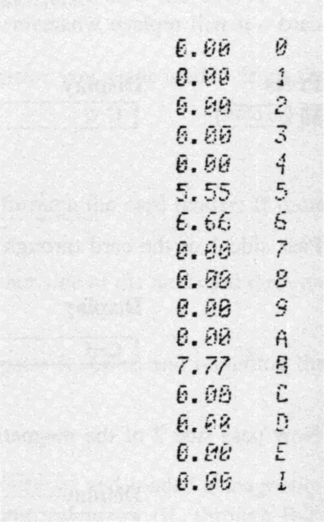
The calculator again displays the contents of the X-register to indicate that all data in the calculator's storage registers has been recorded onto the card as well.

Now change the data in the calculator. Store 1.11 in R_1, 2.22 in R_5, 5.55 in R_{S5}, 6.66 in R_{S6}, and 7.77 in R_B. Print the contents of all the registers when you are through.

Press	Display	
f CL REG	1.000000000 50	All storage registers cleared to zero.
f P⇄S	1.000000000 50	
f CL REG	1.000000000 50	
5.55 STO 5	5.55	Data stored in secondary registers.
6.66 STO 6	6.66	
f P⇄S	6.66	
1.11 STO 1	1.11	
2.22 STO 5	2.22	
7.77 STO B	7.77	
f PRINT: REG	7.77	

```
0.00    0
1.11    1
0.00    2
0.00    3
0.00    4
2.22    5
0.00    6
0.00    7
0.00    8
0.00    9
0.00    A
7.77    B
0.00    C
0.00    D
0.00    E
0.00    I
```

Press	Display
f P⇄S	7.77
f PRINT: REG	7.77



```
0.00    0
0.00    1
0.00    2
0.00    3
0.00    4
5.55    5
6.66    6
0.00    7
0.00    8
0.00    9
0.00    A
7.77    B
0.00    C
0.00    D
0.00    E
0.00    I
```

Now store the number 15 in I, press the f MERGE function, and load the contents of the first 16 storage registers from the magnetic card into the calculator, while preserving the contents of the last 10 storage registers in the calculator.

Press	Display	
15 STO I	15.00	Merge address stored in I.
f MERGE	15.00	

Now pass side 1 of the data card through the card reader.

Display	
Crd	Calculator prompts you that additional data must be loaded from the card.

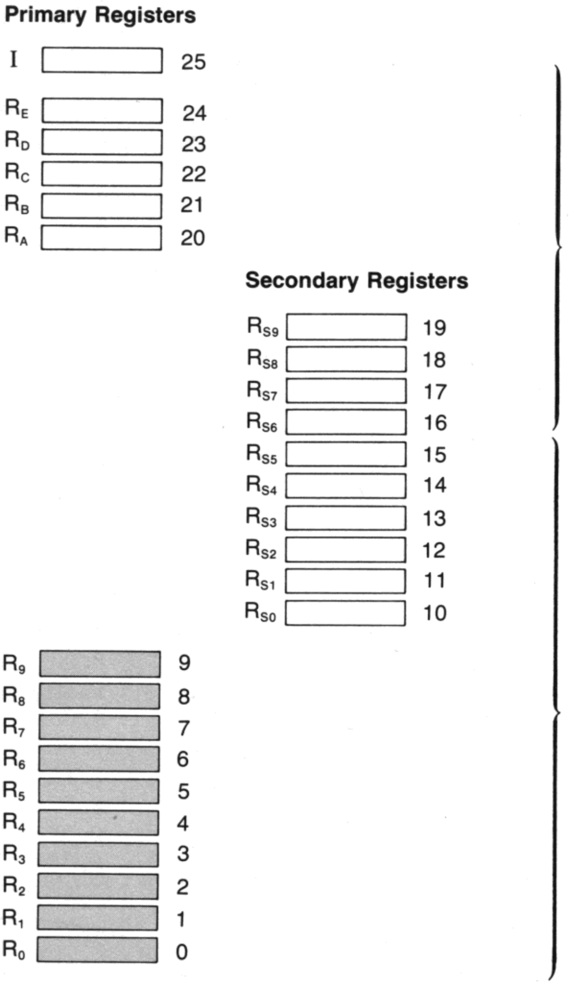
Pass side 2 of the data card through the card reader. Print the new contents of the storage registers and compare them with the old.

Press	Display	
	15.00	Contents of X-register returned to display to indicate that all necessary data has been loaded from the card into the calculator's storage registers.
f PRINT: REG	15.00	
f P⇄S	15.00	
f PRINT: REG	15.00	

```
          0.00  0
1.000000000+10  1
          0.00  2
          0.00  3
          0.00  4
          0.00  5
          0.00  6
          0.00  7
          0.00  8
1.000000000-20  9
          0.00  A
          7.77  B
          0.00  C
          0.00  D
          0.00  E
         15.00  I

          0.00  0
          0.00  1
          0.00  2
          0.00  3
          0.00  4
1.000000000+30  5
          0.00  6
          0.00  7
          0.00  8
          0.00  9
          0.00  A
          7.77  B
          0.00  C
          0.00  D
          0.00  E
         15.00  I
```

You can see that the contents of the storage registers addressed by numbers 0 through 15 (that is, storage registers R_0 through R_9 and secondary storage registers R_{S0} through R_{S5}) have had their contents replaced by data from the magnetic card, while the contents of the remaining storage registers are preserved intact.

When you stored an address of 15 in the I-register, pressed f MERGE, and passed a magnetic card containing data through the card reader:



The contents of these registers remained intact.

The contents of these registers were replaced by data from the magnetic card.

If you do not wish to load or record data when the calculator displays `Crd`, you can press any key to return the contents of the X-register to the display, then continue with your calculations. This allows you to load only the primary or only the secondary registers from a card without pressing f MERGE.

As soon as you have finished loading data or pressed any key from the keyboard, the f MERGE function is forgotten. You must press it each time just before you merge data.

For example, if you load data from the magnetic card now without pressing f MERGE first, the contents of *all* storage registers in the calculator will be replaced by data from the card.

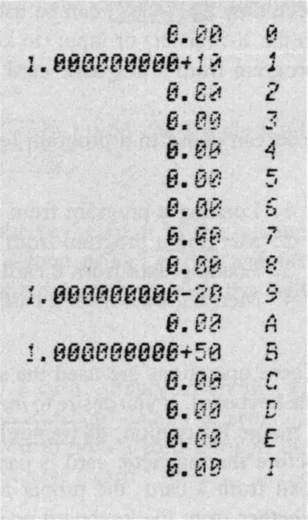
Pass side 1 of the data card through the card reader.

Display

`Crd`

Now pass side 2 of the data card through the card reader.

Press	**Display**
	15.00
f PRINT: REG	15.00



```
            0.00   0
1.000000000+10     1
            0.00   2
            0.00   3
            0.00   4
            0.00   5
            0.00   6
            0.00   7
            0.00   8
1.000000000-20     9
            0.00   A
1.000000000+50     B
            0.00   C
            0.00   D
            0.00   E
            0.00   I
```

Press	Display
f P⇄S	15.00
f PRINT: REG	15.00

```
          0.00    0
          0.00    1
          0.00    2
          0.00    3
          0.00    4
1.000000000+30    5
1.000000000-40    6
          0.00    7
          0.00    8
          0.00    9
          0.00    A
1.000000000+50    B
          0.00    C
          0.00    D
          0.00    E
          0.00    I
```

Note that when the card was recorded, many storage registers, including I, contained zero. When the card was then read and data loaded into the calculator, the zeros in these registers replaced the previous contents of their corresponding registers in the calculator.

Pausing to Read a Card

As you know, the f PAUSE instruction in a program returns control from the running program to the keyboard for the length of a pause (about one second). You have already seen how f PAUSE can be used in a program for output (to display information contained in the X-register) or input (to key in numbers). You can also use a PAUSE to load data or a program from a magnetic card into the calculator.

You can pause in a program for any or all of the following operations:

1. Loading a program from a card into program memory.
2. Merging a program from a card into a selected part of program memory.
3. Loading data from a card into the storage registers.
4. Merging data from a card into selected storage registers.

These operations are used the same way as part of a program as you would use them from the keyboard. If you desire to merge data or a program from a magnetic card into the calculator, a merge instruction, f MERGE, must be executed as an instruction or pressed immediately before the magnetic card is passed through the card reader. In addition, for a merged data load from a card, the proper address must be first placed in the I-register at some point, whether from the keyboard or as part of the program.

To use PAUSE to load a program or data from a magnetic card into the calculator while a program is running:

1. a. Place an f PAUSE instruction at the point in the program where you want to load the data or program.
   b. If the data or the program from the magnetic card is to be *merged* with that in the calculator, insert an f MERGE instruction immediately preceding the f PAUSE instruction.
   c. If *data* is to be merged into the registers, ensure that the proper address number has been placed in I, either from the keyboard or from the program.
2. Slide the PRGM-RUN switch **PRGM** ■▥ **RUN** to RUN.
3. Initialize and run the program.

   You may go ahead and insert the leading edge of the first side to be read into the card reader now, while the program is running. (Insert the tip of the card firmly into the front card reader slot, but do not attempt to force the card in. This may take practice.) You need not hold the card—merely allow it to rest in the card reader slot.
4. Using a PAUSE, when the calculator pauses, the side of the card you have previously inserted in the card reader slot will automatically be read during the PAUSE. If more data or program instructions are to be loaded from the card, the calculator will stop and prompt you with a display of *Crd*.
5. If the calculator display shows *Crd*, insert the second side of the magnetic card into the card reader.
6. Execution will resume automatically when the card is read.

The automatic card read during PAUSE even allows the programmer to be absent when the card is read! If no card is read, the program naturally continues execution after the one-second pause. If a complete card is not accurately read, the program stops and the calculator displays *Error* to indicate that the read was not successful.

If you wish, instead of utilizing an automatic read during a PAUSE, you can simply hold the magnetic card poised in your hand and insert it during a PAUSE.

The data entry flag, flag F3, is set either by digit entry from the keyboard or by the loading of data from a magnetic card. Thus you could also set up a loop using PAUSE and the data entry flag F3 that would halt the program until you pass a data card through the card reader, then resume execution automatically.

If you wish a program to stop execution to record data on a card, simply insert the f W/DATA instruction in the program at the point where you wish to record data. The program will stop at the point and display *Crd* to prompt you to pass a card through the card reader slot. After starting a program running, you can even have a card "waiting" in the card reader for an automatic recording of data onto the card.

Example: The example shown here illustrates how you can pause to read a program from a magnetic card and how a program from a card may be merged into a program already loaded into the calculator. The program that you record onto the magnetic card calculates the area of a circle from its radius. The program that you then load into the calculator performs 100 iterations, then merges the program from the card that you have waiting in the card reader into the calculator, and finally calculates the area of a circle.

To record the program for calculating the area of a circle onto a magnetic card:

Slide the PRGM-RUN switch PRGM ▮ RUN to PRGM.

Press	Display
f CLPRGM	000
LBL B	001 21 12
RCL 9	002 36 09
x^2	003 53
f π	004 16-24
×	005 -35
RTN	006 24

Now select an unprotected (unclipped) side of a magnetic card and pass side 1 of it through the card reader to record the above program onto the magnetic card.

When you have recorded the program for the area of a circle, clear the program from the calculator and record the program that will perform 100 iterations, then will merge the program from the card with the program in the calculator:

Press	Display	
f CLPRGM	000	
LBL A	001 21 11	
STO 9	002 35 09	
1	003 01	
0	004 00	
0	005 00	
STO I	006 35 46	
LBL 1	007 21 01	
f DSZ I	008 16 25 46	When I=0, skip to step 010.
GTO 1	009 22 01	Otherwise, go back to LBL 1.
f MERGE	010 16-62	Sets merge mode.
f PAUSE	011 16 51	Pause to merge program into calculator.
RTN	012 24	

To run the program to find the area of a circle with a radius of 15 centimeters:

Slide the PRGM-RUN switch PRGM ▮▯ RUN to RUN.

Press	Display	
15	15.	The radius keyed in.
A		The program running.

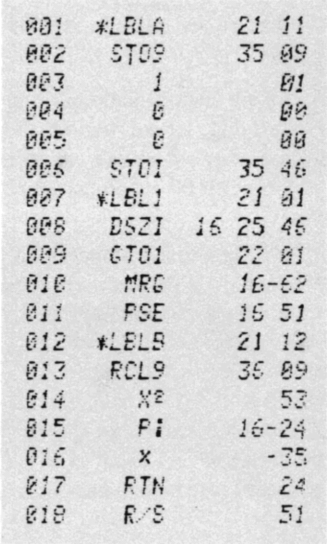
While the program is running, insert side 1 of the card containing the program for the area of a circle into the card reader slot. Insert the card until you just feel pressure against the end, then stop and let go of the card, permitting it to ''wait'' in the card reader slot.

When the program in the calculator has gone through 100 iterations and the value in I reaches zero, the card waiting in the card reader slot will be read, that program merged with the program already in the calculator, and the area of the circle calculated.

If you see Error after the card is read, or if the card does not pass through the card reader and you see Error , remove the card, clear the error by pressing any key, then key in the radius again and restart the program by pressing A. Then reinsert the card while the program is running. When the program has run correctly, you will see the area of the circle, 706.86 centimeters, displayed.

To see the present contents of program memory:

Press	Display
RTN	706.86
f PRINT: PRGM	706.86



```
001  *LBLA       21 11
002   STO9       35 09
003      1          01
004      0          00
005      0          00
006   STOI       35 46
007  *LBL1       21 01
008   DSZI    16 25 46
009   GTO1       22 01
010    MRG       16-62
011    PSE       16 51
012  *LBLB       21 12
013   RCL9       36 09
014     X²          53
015     Pi       16-24
016      x         -35
017    RTN          24
018    R/S          51
```

When merging programs, any instruction executed after the f MERGE instruction cancels the merge mode *except* a PAUSE. Notice, too, that while normally instructions after a MERGE are overwritten by a new program, a PAUSE after a MERGE in a program is not overwritten.

```
001  PRTX      -14
002  PRST    16-14
003  PREG    16-13
004   SPC    16-11
```

```
001  PRTX      -14
002  PRST    16-14
003  PREG    16-13
004   SPC    16-11
```

```
001  PRTX
002  PRST
003  PREG
004   SPC
```

Section 15

The HP-67 and the HP-97: Interchangeable Software

Programs that have been recorded onto cards by your calculator can be loaded and run in your calculator as often as you like. They can also be run on *any* HP-67 Programmable Pocket Calculator or on *any* HP-97 Programmable Printing Calculator. In fact, software (i.e., a group of programs) is completely interchangeable between these two Hewlett-Packard calculator models—any programs that have been recorded onto a card using the HP-97 can be loaded into and run on the HP-67, and vice versa. Data cards are also interchangeable between the two calculators. All functions and switches operate alike on the two calculators, with the exception of the printing functions on the HP-97 and the automatic list functions of the HP-67.

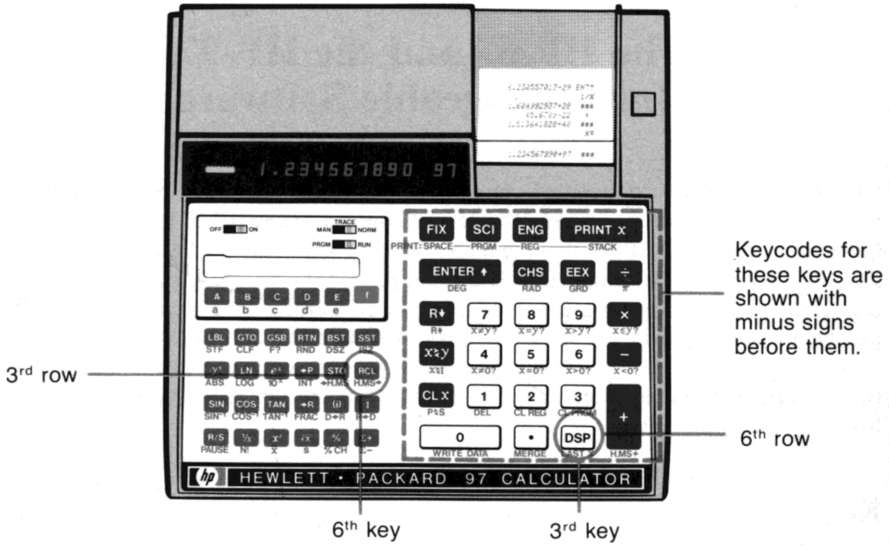
Keycodes

You can see the similiarities of the HP-67 and the HP-97 in the illustration shown on page 272. Although some of the nomenclature on the keys varies, the difference is so slight that you will find it easy to operate either model of calculator if you are familiar with the other. For example, PAUSE on the HP-97 operates exactly the same as PAUSE on the HP-67, and h STI on the HP-67 is the same as STO I on the HP-97.

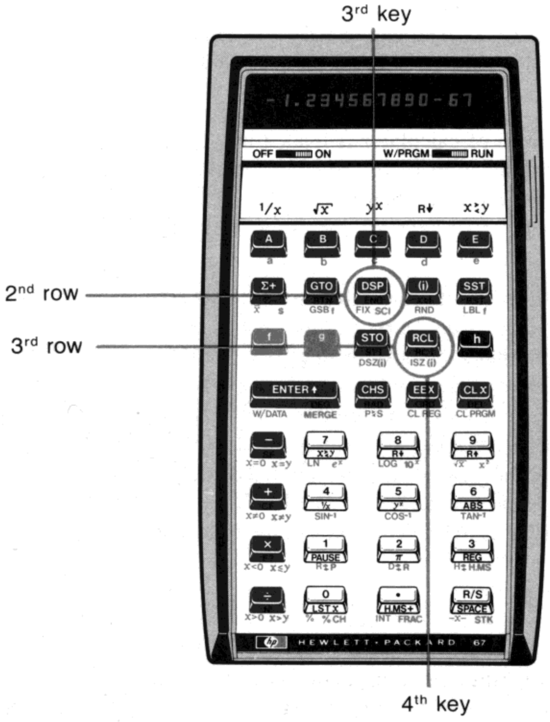
When a program from one model of calculator is loaded into the other model, the keycode is transmogrified to reflect the location of the function on the new model. Keycodes are read alike on both models; first the row, then the number of the key in the row, from the top down and from right to left. Digit keys are represented by keycodes 00 through 09.

On the HP-67, keycodes for functions that are selected by first pressing a prefix key and then a digit key are referenced by the keycode matrix address, not by 00 through 09. On the HP-97, however, the prefixed functions that are below the digit keys are referenced by keycodes of 00 through 09. On the HP-97, the keycodes for keys on the right-hand keyboard (except for the digit keys) are generated with minus signs before them. For example, if you had loaded the operation DSP 9 into the HP-67, the keycode would be 23 09 (that is, 2nd row, 3rd key, followed by the keycode 09 for the 9 key). If you recorded that operation on a magnetic card, then read the card to load the same operation into an HP-97, the keycode for the operation on the HP-97 would be –63 09 (that is, 6th row, 3rd key of the right-hand keyboard, followed by the 9 key).

HP-97 Programmable Printing Calculator



HP-67 Programmable Pocket Calculator



Keycodes representing keys on the left keyboard of the HP-97 have keycodes with no sign before them, so a RCL 3 instruction loaded into the HP-97 would have a keycode of 36 03. If that operation were recorded onto a magnetic card, and then loaded from the card into an HP-67, the keycode for the operation in the HP-67 would be 34 03.

Although the keycodes are altered, the operations are the same from calculator to calculator, from HP-67 to HP-97. And since each operation, no matter how many keystrokes it takes to perform, is loaded into a single step of program memory, the steps of program memory into which a program is loaded are the same when a program is changed from one calculator to another.

For example, if you loaded the program for the area of a sphere into an HP-67 from the keyboard, then recorded it onto a magnetic card, and finally passed the card through the card reader of an HP-97, the contents of the program memory of the two calculators might look like this:

HP-67 Program Memory

Step	Instruction	Keycode
001	f LBL A	31 25 11
002	g x^2	32 54
003	h π	35 73
004	×	71
005	h RTN	35 22

HP-97 Program Memory

Step	Instruction	Keycode
001	LBL A	21 11
002	x^2	53
003	f π	16-24
004	×	-35
005	RTN	24

You can see that the program is exactly the same in the two calculators, even though some operations are prefixed in one calculator and not in the other. Operations are always loaded correctly for the particular calculator, so you can examine and edit the program, no matter the model calculator into which it is loaded. For a complete list of keycodes and corresponding instructions on the two calculators, refer to appendix E of this handbook.

Print and Automatic Review Functions

The only functions that operate differently between the two calculators are the automatic review functions and -x- pause on the HP-67, and the print functions on the HP-97. For example, you can print a list of the stack contents with the printer on the HP-97 by using the PRINT: STACK function. Since the HP-67 Programmable Pocket Calculator does not have a printer, however, the stack contents are reviewed by passing them through the display, one register at a time, with the STK (*automatic stack review*) function.

Since the purpose of the two operations is essentially the same (review of the stack contents), these two operations are interchangeable between the two models of calculators. If you load a program containing a g STK instruction into an HP-67 Programmable Pocket Calculator, then record that program onto a magnetic card and use the card to load the same program into an HP-97 Programmable Printing Calculator, the g STK instruction from the HP-67 will automatically be loaded into the HP-97 as an f PRINT: STACK instruction.

Similiarly, the PRINT x function on the HP-97 and the 5-second -x- pause on the HP-67 have the same purpose—to allow you to record the current contents of the X-register while a program is running. Thus, a PRINT x instruction in a program recorded onto a magnetic card by an HP-97 printing calculator will be loaded into an HP-67 as an -x- pause when the card is passed through the card reader on the HP-67. In the HP-67, this 5-second -x- pause is designed to give you enough time to write down or view the contents of the X-register while a program is running.

The chart below illustrates the keys that are different yet interchangeable between the HP-67 and the HP-97.

Operation on:		**When encountered as an instruction by the HP-67:**	**When encountered as an instruction by the HP-97:**
HP-67	**HP-97**		
-x-	PRINT x	Causes program execution to pause for about 5 seconds, displaying current contents of X-register with decimal point blinking eight times.	Prints current contents of X-register.
STK	PRINT: STACK	Displays current contents of each stack register sequentially; T, Z, Y, and X, with decimal point blinking twice for each stack register.	Prints current contents of stack registers.
REG	PRINT: REG	Displays contents of each primary storage register, beginning with registers R_0 through R_9, then R_A through R_E, and finally I. Each display is preceded by a displayed number indicating the register's address.	Prints contents of all primary storage registers; R_0 through R_9, R_A through R_E, I.
SPACE	PRINT: SPACE	Performs no operation.	Prints a blank space on paper tape.

Naturally, all other keys perform exactly the same function on the HP-67 as on the HP-97.

> **Note:** On the HP-67 only, the five default functions (1/x $\sqrt{x}$ y^x R↓ x⇄y) above the top-row keys are present on the calculator to enhance its usability in RUN mode and cannot be recorded in program memory. However, these same functions are duplicated elsewhere on the calculator by prefixed functions, and these prefixed operations *can* be recorded.

Notice that the SPACE function on the HP-67 Programmable Pocket Calculator performs no operation on that calculator. It is used only in programs which may be recorded onto magnetic cards and later loaded and run on the HP-97 with its built-in printer. On the HP-97 SPACE prints a blank space on the paper tape, just as if you had pressed the paper advance pushbutton, and it is extremely useful in separating answers or portions of your HP-97 print-out.

Remember: Any program or data recorded on a magnetic card can be loaded from that card and used in *either* the Hewlett-Packard HP-67 Programmable Pocket Calculator or the HP-97 Programmable Printing Calculator with built-in printer. Because the design of these two calculators has been integrated, you can use your own program cards, or preprogrammed cards like those in your Standard Pac or any of the optional application pacs, in *any* HP-67 or HP-97.

A Word about Programming

You have now been exposed to the features of the HP-97 Programmable Printing Calculator. But exposure is not enough—to gain confidence in and fully appreciate the power of this small computer, you must *use* it to solve your problems, simple and complex. Although the best program for a given application is usually the most straightforward, don't be afraid to experiment, to forge into the unknown, to stamp upon your programs your own personal trademarks. And you'll probably want to learn some of the ''tricks'' of sophisticated programmers to apply in your own software.

A good place to begin is with the *HP-97 Standard Pac*. At the end of its text you will find detailed explanations for some of the techniques used in actual programs in the Pac. You can also learn a great deal about programming by ''reading'' the programs in the Pac—following them, step-by-step, to see what makes them tick, to attempt to find out why the programmer used the steps he did.

With the HP-97 the problems of the world can be solved!

Appendix A

Accessories

Standard Accessories

Your HP-97 comes equipped with one each of the following standard accessories:

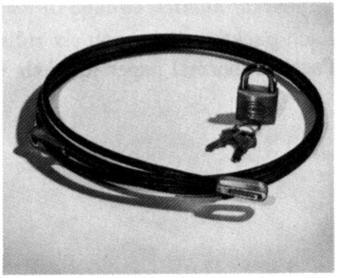
Accessory	**Part Number**
Battery Pack (installed in calculator before shipping)	82033A
AC Adapter/Recharger	82059B replaces → 82040A
HP-97 Owner's Handbook and Programming Guide	00097-90001
Two Rolls of Paper	
Carrying Case	82035A
Standard Pac, including:	00097-13101
HP-97 Standard Pac (instruction book)	
14 Prerecorded Program Cards	
1 Prerecorded Magnetic Card containing Diagnostic Program	
1 Head Cleaning Card	
24 Blank Magnetic Cards	
Card Holder for Magnetic Cards	
Programming Pad	00097–13154

Optional Accessories

In addition to the standard accessories shipped with your HP-97, Hewlett-Packard also makes available the optional accessories seen below. These accessories have been created to help you maximize the usability and convenience of your calculator.

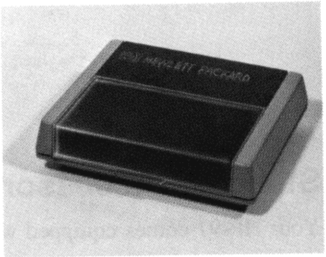
Security Cable 82044A

A tough, 6-foot long steel cable that prevents unauthorized borrowing or pilferage of your calculator by locking it to a desk or work surface. The cable is plastic-covered to eliminate scarring of furniture, and you have full access to all features of your HP-97 at all times. Comes complete with lock.



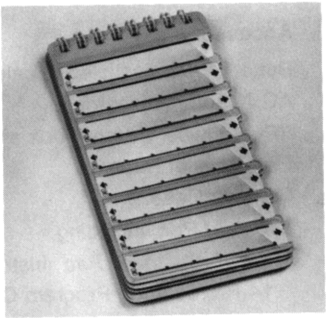
Reserve Power Pack 82037A

The reserve power pack attaches to the calculator's ac adapter/recharger to keep an extra battery pack freshly charged and ready for use. Comes complete with extra battery pack.



Blank Magnetic Cards 00097-13141

Each pack of blank magnetic cards contains 40 HP program cards for recording of your own programs. Any card can be labeled to indicate program title and functions of user-definable keys. Includes personal card holder.



Multiple Card Packs 00097-13143

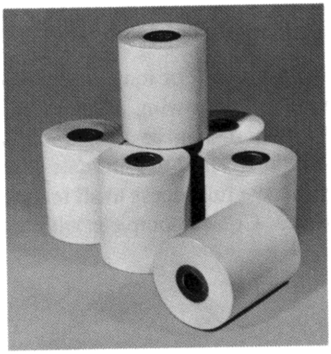
Consists of three packs of blank magnetic cards (120 cards total) with three personal card holders.

Program Card Holders 00097-13142

Each package contains three program holders like the one shipped with your calculator for carrying extra magnetic cards.

Paper Rolls 82045A

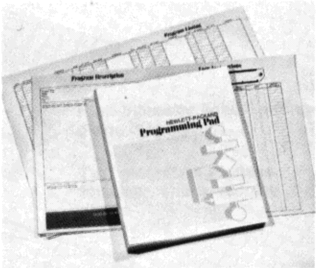
Each package gives you six rolls of special Hewlett-Packard thermal paper for your HP-97 printer.



HP-97 Application Pacs

Each pac provides approximately 20 prerecorded programs in a particular field or discipline. Each comes complete with a detailed instruction book and prerecorded magnetic cards.

Programming Pad 00097-13154



Each pad provides 40 worksheets to help you develop programs for your HP-97.

To order additonal standard or optional accessories for your HP-97 see your nearest dealer or fill out an Accessory Order Form and return it with check or money order to:

HEWLETT-PACKARD
Corvallis Division
1000 N.E. Circle Blvd.
Corvallis, OR 97330

If you are outside the U.S., please contact the Hewlett-Packard Sales Office nearest you.

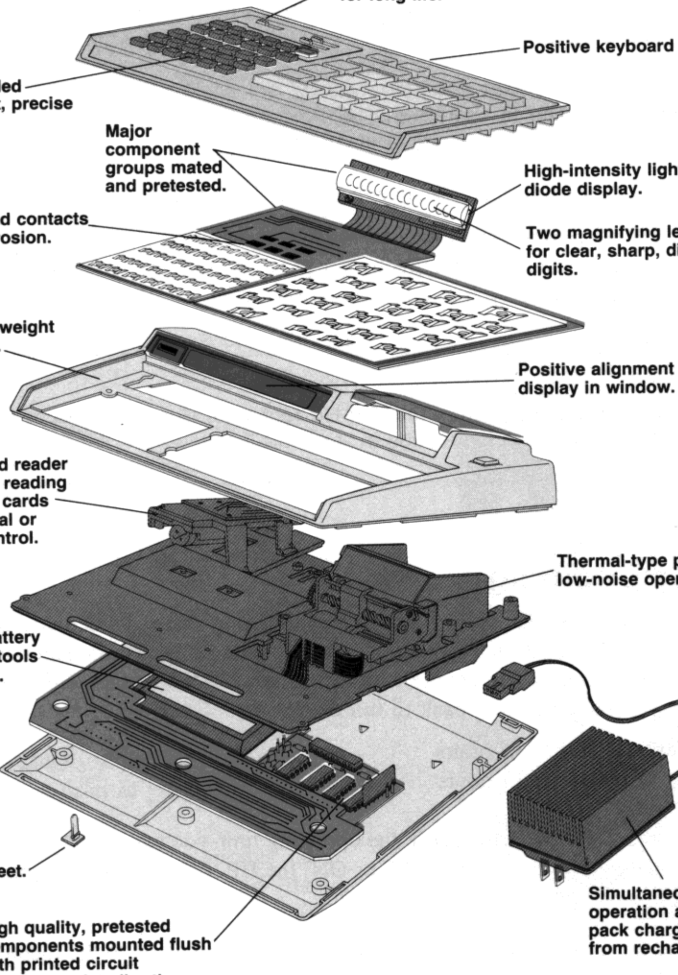
Availability of all accessories, standard or optional, is subject to change without notice.

Appendix B

Service and Maintenance

Your Hewlett-Packard Calculator

Your HP-97 is another example of the award-winning design, superior quality, and attention to detail in engineering and construction that have marked Hewlett-Packard electronic instruments for more than thirty years. Each Hewlett-Packard calculator is precision crafted by people who are dedicated to giving you the best possible product at any price.



Switches silicon greased for long life.

Positive keyboard feel.

Keys double injection-molded for permanent, precise symbols.

Major component groups mated and pretested.

High-intensity light-emitting-diode display.

Gold-plated contacts resist corrosion.

Two magnifying lenses for clear, sharp, display digits.

Sturdy, lightweight plastic case.

Positive alignment of display in window.

"Smart" card reader for accurate reading of magnetic cards under manual or program control.

Thermal-type printer for low-noise operation.

One-piece rechargeable battery pack needs no tools for replacement.

Non-skid PVC feet.

Simultaneous ac line operation and battery pack charging from recharger.

High quality, pretested components mounted flush with printed circuit board to resist vibration.

After construction, every calculator is thoroughly inspected for electrical or mechanical flaws, and each function is checked for proper operation.

When you purchase a Hewlett-Packard calculator, you deal with a company that stands behind its products. Besides an instrument of unmatched professional quality, you have at your disposal many extras— accessories to make your calculator more usable, service that is available worldwide, and support expertise in a host of applications areas.

AC Line Operation

Your calculator contains a rechargeable battery pack that consists of nickel-cadmium batteries. When you receive your calculator, the battery pack inside may be discharged, but you can operate the calculator immediately by using the ac adapter/recharger. Even though you are using the ac adapter/recharger, the batteries must remain in the calculator whenever the calculator is used.

Note: Attempting to operate the HP-97 from the ac line with the battery pack removed may result in wrong or improper displays.

The procedure for using the ac adapter/recharger is as follows:

1. The HP-97 power switch may be set to ON or OFF.
2. Insert the female ac adapter/recharger plug into the rear connector of the HP-97.
3. Insert the power plug into a live ac power outlet.

CAUTION

The use of a charger other than an HP recharger like the one supplied with the calculator may result in damage to your calculator.

Battery Charging

The rechargeable batteries in the battery pack are being charged when you are operating the calculator from the ac adapter/recharger. With the batteries in the calculator and the recharger connected, the batteries will charge with the calculator OFF or ON. Normal charging times from fully discharged battery pack to full charge are:

Calculator OFF:	6 hours
Calculator ON:	17 hours

Shorter charging periods will reduce the operating time you can expert from a single battery charge. Whether the calculator is OFF or ON, the HP-97 battery pack is never in danger of becoming overcharged.

Note: It is normal for the ac adapter/recharger to be warm to the touch when it is plugged into an ac outlet.

Battery Operation

To operate the HP-97 from battery power alone, simply disconnect the female recharger plug from the rear of the calculator. Even when not connected to the calculator, the ac adapter/recharger may be left plugged into the ac outlet.

Using the HP-97 on battery power gives the calculator full portability, allowing you to carry it nearly anywhere. A fully charged battery pack provides approximately 3 to 6 hours of continuous operation. By turning the power OFF when the calculator is not in use, the charge on the HP-97 battery pack should easily last throughout a normal working day.

The printer is the most power-consuming part of your HP-97, and you can maximize battery operating time by leaving the calculator in MANUAL MAN TRACE NORM printing mode when printing is not necessary.

Battery Pack Replacement

If it becomes necessary to replace the battery pack, use only another Hewlett-Packard battery pack like the one shipped with your calculator.

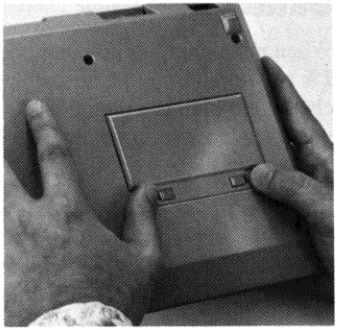
CAUTION

Use of any batteries other than the Hewlett-Packard battery pack may result in damage to your calculator.

To replace your battery pack, use the following procedure:

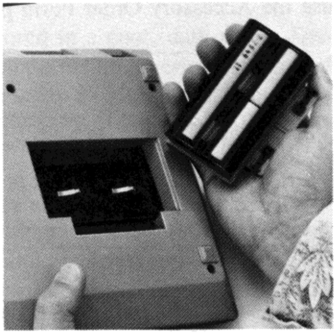
1. Turn the ON-OFF switch to "OFF" and disconnect the ac adapter/recharger from the calculator.

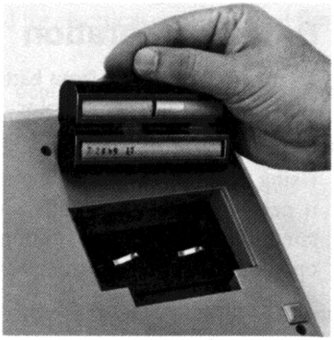
2. Slide the two battery door latches inward.



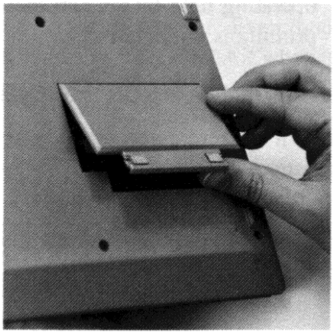
3. Let the battery door and battery pack fall into the palm of your hand.

4. If the battery connector springs have been flattened inward, bend them slightly outward again.





5. Insert the new battery pack so that its contacts face the calculator and line up with the connector springs.



6. Insert the end of the battery door that is opposite the latches behind the retaining groove in the calculator and close the door.

7. Secure the battery door by pressing it gently while sliding the two battery door latches outward.

Battery Care

When not being used, the batteries in your HP-97 have a self-discharge rate of approximately 1% of available charge per day. After 30 days, a battery pack could have only 50 to 75% of its charge remaining, and the calculator might not even turn on. If a calculator fails to turn on, you should substitute a charged battery pack, if available, for the one in the calculator. The discharged battery pack should be charged for at least 14 hours.

If a battery pack will not hold a charge and seems to discharge very quickly in use, it may be defective. The battery pack is warranted for one year, and if the warranty is in effect, return the defective pack to Hewlett-Packard according to the shipping instructions. (If you are in doubt about the cause of the problem, return the complete HP-97 along with its battery pack and ac adapter/recharger.) If the battery pack is out of warranty, see your nearest dealer or use the Accessory Order Form provided with your HP-97 to order a replacement.

WARNING

Do not attempt to incinerate or mutilate your HP-97 battery pack—the pack may burst or release toxic materials.

Do not connect together or otherwise short-circuit the battery pack terminals—the pack may melt or cause serious burns.

To maximize the life you get from your battery pack, keep printing to a minimum and display only the fewest number of digits necessary during portable operation.

Your HP-97 Printer

The printing device in your HP-97 is a thermal printer that uses a moving print head to print upon a special heat-sensitive paper. When the print head is energized, it heats the paper beneath it. The heat causes a chemical change in the paper, changing its color. The printer in your HP-97 prints answers quickly and quietly, and has been expressly designed to give you a permanent record of your computations in a portable scientific calculator.

Paper for Your HP-97

Because the printer in your HP-97 is a thermal printer, it requires special heat-sensitive paper. You should use only the Hewlett-Packard thermal paper available in 80-foot rolls from your nearest HP distributor or sales office, or by mail order from:

HEWLETT-PACKARD
Corvallis Division
1000 N.E. Circle Blvd.
Corvallis, OR 97330

Because of the special heat-sensitive requirements of the paper, standard adding machine paper will *not* work in the HP-97. Also, since different types of thermal paper vary in their sensitivities, the use of thermal paper other than that available from Hewlett-Packard may result in poor print quality or even in damage to your calculator.

CAUTION

To prevent damage to your calculator use only Hewlett-Packard paper in your HP-97.

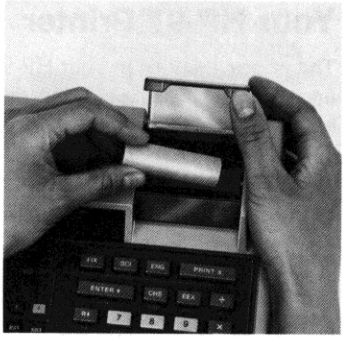
The heat-sensitive paper used in your HP-97 should be stored in a cool, dark place. Discoloration of paper may occur if it is exposed to direct sunlight for long periods of time, if storage temperatures rise above 50°C (122°F), or if the paper is exposed to acetone, ammonia, or other organic compounds. Exposure to gasoline or oil fumes will not harm your HP-97 paper supply.

Printed tapes from your HP-97 will last 30 days or more without fading under fluorescent light, but to ensure the permanence of your records, you should store printed tapes at room temperature in a dark place away from direct sunlight, heat, or fumes from organic compounds. For added permanence, you can copy tapes with a suitable office copier.

Replacing Paper

To replace the paper roll in your HP-97, proceed as follows:

1. Open the paper roll cover and remove the empty core from the paper well.

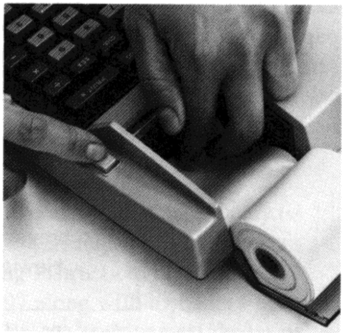


2. Before inserting the new roll of paper into the calculator, discard the first 2/3 turn to ensure that no glue, tape, or other foreign matter is on the paper.

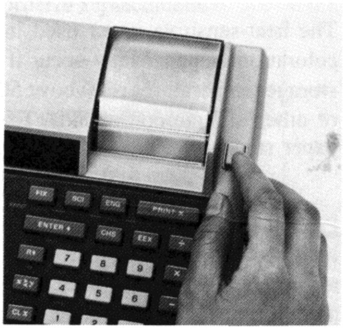
3. Fold the leading edge of the paper and crease the fold with your fingernail.

4. Temporarily place the paper roll into the paper roll cover and insert the leading edge of paper into the slot near the bottom of the paper well.

5. Turn the calculator ON-OFF switch to ON and press the paper advance pushbutton several times until the leading edge of paper becomes visible beneath the clear plastic tear bar.



6. Drop the roll of paper into the paper well and close the paper roll cover.



When there is no paper in the calculator, the paper advance pushbutton operates, but any operation which would normally cause printing to occur instead causes a display of `Error`.

Printer Maintenance

The printer in your HP-97, like the rest of the calculator, is crafted for excellence and is designed to give trouble-free operation with a minimum of maintenance. All moving parts in the printer mechanism contain self-lubricating compound, and no lubrication, cleaning, or servicing of the mechanism is ever required. You may want to occasionally remove the clear plastic tear bar and clean it with mild soap and water solution. (Do not use acetone or alcohol to clean the tear bar.)

You should *never* attempt to insert a tool, such as a screwdriver, pencil, or other hard object, into the printer or its mechanism. If the paper tape should become jammed and fail to feed properly, clear it by grasping the tape and pulling it forward or backward through the printer mechanism. You can remove the plastic tear bar for accessability.

If the paper is feeding properly through the printer mechanism, but no printing appears on the tape, the paper roll is probably inserted backwards. (The paper is chemically treated, and will print on only one side.) Tear off the leading edge of paper, open the paper roll cover and grasp the paper roll, and pull it backward to remove the paper tape that is in the print mechanism. Reverse the paper roll and feed it back into the printing mechanism as described earlier under Replacing Paper.

If, after reversing, there is still no printing on the tape when you press PRINTx or other print functions, remove the paper roll and insert a roll of Hewlett-Packard thermal paper.

Magnetic Card Maintenance

Try to keep your cards as clean and free of oil, grease, and dirt as possible. Dirty cards can only degrade the performance of your card reader. Cards may be cleaned with alcohol and a soft cloth.

Minimize the exposure of your calculator to dusty, dirty environments by storing it in the soft carrying case when not in use.

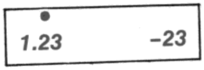
Each card pack contains one head cleaning card.

The magnetic recording head is similar to magnetic recording equipment. As such, any collection of dirt or other foreign matter on the head can prevent contact between the head and card, with consequent failure to read cards. The head cleaning card consists of an abrasive underlayer designed to remove such foreign matter. However, the use of the card without the presence of a foreign substance will remove a minute amount of the head itself. Thus, extensive use of the cleaning card can reduce the life of the card reader in your HP-97. If you suspect that the head is dirty, or if you have trouble reading or recording cards, by all means use the cleaning card; that's what it is for. If one to five passes of the cleaning card do not clear up the situation, refer to Improper Card Reader Operation.

Service

Low Power

When you are operating from battery power, a bright red lamp inside the display will glow to warn you that the battery is close to discharge.



Low Power Display

You must then either connect the ac adapter/recharger to the calculator as described under AC Line Operations, or you must substitute a fully charged battery pack for the one in the calculator. When connecting the recharger with the low power lamp on, allow a few minutes with out printing for the battery to be charged and the lamp to go out.

Blank Display

If the display blanks out, turn the HP-97 OFF, then ON. If 0.00 does not appear in the display, check the following:

1. If the ac adapter/recharger is attached to the HP-97, make sure it is plugged into an ac outlet.
2. Examine battery pack to see if the contacts are dirty.
3. Substitute a fully charged battery pack, if available, for the one that was in the calculator.
4. If display is still blank, try operating the HP-97 using the recharger (with the batteries in the calculator).
5. If, after step 4, display is still blank, service is required. (Refer to Warranty paragraphs.)

Blurring Display

During execution of a program, the display continuously changes and is purposely illegible to indicate that the program is running. When the program stops, the display is steady.

Improper Card Reader Operation

If your calculator appears to be operating properly except for the reading or loading of program cards, check the following:

1. Make sure that the PRGM-RUN switch is in the correct position for desired operation: RUN position for loading cards, PRGM for recording cards.
2. If the drive motor does not start when a card is inserted, make sure the battery pack is making proper contact and has ample charge. A recharger may be used in conjunction with a partially charged battery in order to drive the card reader motor. If the battery has been completely discharged, plug in the recharger and wait 5 minutes before attempting to operate the card reader.

3. If the card drive mechanism functions correctly, but your HP-97 will not read or record program cards, the trouble may be due to dirty record/playback heads. Use the head cleaning card as directed. Then test the calculator using the diagnostic program card furnished with it, following the instructions provided. If difficulty persists your HP-97 should be taken or sent to an authorized Hewlett-Packard Customer Service Facility.
4. Cards must move freely past the record/playback heads. Holding a card back or bumping a card after the card drive mechanism engages could cause a card to be misread.

CAUTION

Cards can be accidentally erased if subjected to strong magnetic fields. (Magnetometers at airports are in the safe range.)

5. Check the condition of your magnetic cards. Cards that are dirty or that have deep scratches will sometimes not read properly.
6. If you are trying to operate the calculator outside the recommended temperature range, you may experience problems with the card reader. Low temperatures may cause the calculator to register `Error` when a magnetic card is read.

Temperature Range

Temperature ranges for the calculator are:

Operating	+10° to 40°C	+50° to 104°F
Charging	+15° to 40°C	+59° to 104°F
Storage	−40° to +55°C	−40° to +131°F

Warranty

Full One-Year Warranty

All Hewlett-Packard calculators and their accessories are warranted against defects in materials and workmanship for one (1) year from the date of delivery. During the warranty period Hewlett-Packard will repair or, at its option, replace at no charge components that prove to be defective, provided the calculator or accessory is returned, shipping prepaid, to Hewlett-Packard's Customer Service Facility. (Refer to Shipping Instructions.)

This warranty does not apply if the calculator or accessory has been damaged by accident or misuse or as a result of service or modification by other than an authorized Hewlett-Packard Customer Service Facility. No other expressed warranty is given by Hewlett-Packard. HEWLETT-PACKARD SHALL NOT BE LIABLE FOR CONSEQUENTIAL DAMAGES.

Some states do not allow the exclusion or limitation of incidental or consequential damages, so the above limitation or exclusion may not apply to you.

This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

Warranty Information Toll-Free Number
800/648-4711 (In Nevada call collect 702/323-2704.)

Obligation to Make Changes

Products are sold on the basis of specifications applicable at the time of sale. Hewlett-Packard shall have no obligation to modify or update products once sold.

Repair Policy

Repair Time

Hewlett-Packard calculators are normally repaired and reshipped within five (5) working days of receipt at any Customer Service Facility. This is an average time and could possibly vary depending upon time of year and work load at the Customer Service Facility.

Shipping Instructions

The calculator should be returned, along with completed Service Card, in its shipping case (or other protective package) to avoid in-transit damage. Such damage is not covered by warranty, and Hewlett-Packard suggests that the customer insure shipments to the Customer Service Facility. A calculator returned for repair should include the ac adapter/recharger and the battery pack. Send these items to the address shown on the Service Card.

Shipping Charges

Whether the calculator is in-warranty or out-of-warranty, the customer should prepay shipment to the Hewlett-Packard Customer Service Facility. During warranty, Hewlett-Packard will prepay shipment back to the customer.

Further Information

Service contracts are not available. Calculator circuitry and design are proprietary to Hewlett-Packard, and Service Manuals are not available to customers.

Should other problems or questions arise regarding repairs, please call your nearest Hewlett-Packard Sales Office or Customer Service Facility.

Appendix C

Improper Operations

If you attempt a calculation containing an improper operation—say, division by zero—the calculator display will show Error. In addition, if the Print Mode switch MAN TRACE NORM is set to NORM or TRACE, the word ERROR will be printed (unless the calculator is out of paper).

The following are improper operations:

÷	where x = 0
y^x	where y = 0 and x ⩽ 0
y^x	where y < 0 and x is non-integer
$\sqrt{x}$	where x < 0
$1/x$	where x = 0
LOG	where x ⩽ 0
LN	where x ⩽ 0
SIN^{-1}	where $\lvert x \rvert$ is > 1
COS^{-1}	where $\lvert x \rvert$ is > 1
STO ÷	where x = 0
$\bar{x}$	where n = 0
s	where n ⩽ 1

STO + n, STO − n, STO × n, STO ÷ n, where magnitude of number in storage register n would then be larger than $9.999999999 \times 10^{99}$.

%CH	where y = 0
DSP (i)	where ABS (INT I) > 9
STO (i)	where ABS (INT I) > 25
RCL (i)	where ABS (INT I) > 25

STO + (i), STO − (i), STO × (i), STO ÷ (i), where ABS (INT I) > 25, or where magnitude of number in storage register addressed by I would be larger than $9.999999999 \times 10^{99}$.

ISZ (i), DSZ (i), where ABS (INT I) > 25

GTO (i), GSB (i), where −999 > INT I > 19

Card reader malfunction.

Attempting to print when there is no paper in the calculator.

Attempting to record on a protected side of a magnetic card.

Appendix D

Stack Lift and LAST X

Stack Lift

A number keyed in following one of these operations lifts the stack:

−
+
×
÷
e^x
LN
LOG
10^x
SIN
SIN^{-1}
RTN
RCL Σ+
N!
R/S
x≠y?
x=y?
x>y?
x≤y?
x≠0?
x=0?
x>0?
x<0?
COS
COS^{-1}
TAN
TAN^{-1}
INT
FRAC
√x
x^2
π
y^x
F? 0 - 3
CLF 0 - 3
STF 0 - 3
ISZ I, (i)
DSZ I, (i)

W/DATA
H.MS→
→H.MS
ABS
1/x
%
$\bar{x}$
s
x⇄y
R↓
GTO n, A - E, (i)
GSB n, A - E, (i)
LBL n, A - E
DEG
RAD
GRD
P⇄S
CL REG
R→D
D→R
H.MS+
% CH
CHS, if not preceded by digit, EEX or ·.
PAUSE
STO n, A - E, I (i)
RCL n, A - E, I (i)
STO − n, (i)
STO + n, (i)
STO × n, (i)
STO ÷ n, (i)
→R
→P
RND

A number keyed in following one of these keys does not affect the stack:

CHS, when preceded by •, EEX, or digit.
FIX
SCI
ENG
DSP n, (i)
EEX
CLPRGM } in RUN mode
DEL } in RUN mode
MERGE
PRINT x
PRINT: SPACE
PRINT: REG
•
PRINT: PRGM
PRINT: STACK
0 through 9

A number keyed in following one of these operations writes over the number in the X-register and the stack does not lift:

CL X
ENTER↑
Σ+
Σ−

LAST X

The following operations save x in LAST X:

−
+
×
÷
→H.MS
H.MS→
ABS
RND
D→R
R→D
Σ+
Σ−
% CH
RCL Σ+
$\bar{x}$
s
FRAC
INT
LN
LOG
SIN
SIN^{-1}
N!
COS
COS^{-1}
TAN
TAN^{-1}
$\sqrt{x}$
x^2
1/x
y^x
e^x
10^x
→R
→P
H.MS+

Appendix E

Calculator Functions and Keycodes

This table shows the corresponding functions that can be loaded into program memory on the HP-67 Programmable Pocket Calculator and the HP-97 Programmable Printing Calculator.

HP-97 Tape Symbol	HP-97 Keystrokes	HP-97 Keycode	HP-67 Keystrokes	HP-67 Keycode
0	0	00	0	00
1	1	01	1	01
2	2	02	2	02
3	3	03	3	03
4	4	04	4	04
5	5	05	5	05
6	6	06	6	06
7	7	07	7	07
8	8	08	8	08
9	9	09	9	09
.	•	-62	•	83
1/X	1/x	52	h 1/x	35 62
10^x	f 10^x	16 33	g 10^x	32 53
ABS	f ABS	16 31	h ABS	35 64
CF0	f CLF 0	16 22 00	h CF 0	35 61 00
CF1	f CLF 1	16 22 01	h CF 1	35 61 01
CF2	f CLF 2	16 22 02	h CF 2	35 61 02
CF3	f CLF 3	16 22 03	h CF 3	35 61 03
CHS	CHS	-22	CHS	42
CLRG	f CL REG	16-53	f CL REG	31 43
CLX	CL x	-51	CL x	44
COS	COS	42	f COS	31 63
COS^{-1}	f COS^{-1}	16 42	g COS^{-1}	32 63
DEG	f DEG	16-21	h DEG	35 41
÷	÷	-24	÷	81
D→R	f D→R	16 45	g →R	32 73
DSP0	DSP 0	-63 00	DSP 0	23 00
DSP1	DSP 1	-63 01	DSP 1	23 01
DSP2	DSP 2	-63 02	DSP 2	23 02
DSP3	DSP 3	-63 03	DSP 3	23 03
DSP4	DSP 4	-63 04	DSP 4	23 04
DSP5	DSP 5	-63 05	DSP 5	23 05
DSP6	DSP 6	-63 06	DSP 6	23 06
DSP7	DSP 7	-63 07	DSP 7	23 07
DSP8	DSP 8	-63 08	DSP 8	23 08
DSP9	DSP 9	-63 09	DSP 9	23 09

HP-97 Tape Symbol	HP-97 Keystrokes	HP-97 Keycode	HP-67 Keystrokes	HP-67 Keycode
DSPi	DSP (i)	-63 45	DSP (i)	23 24
DSZI	f DSZ I	16 25 46	f DSZ	31 33
DSZi	f DSZ (i)	16 25 45	g DSZ(i)	32 33
EEX	EEX	-23	EEX	43
ENG	ENG	-13	h ENG	35 23
ENT↑	ENTER↑	-21	ENTER↑	41
e^x	e^x	33	g e^x	32 52
F0?	f F? 0	16 23 00	h F? 0	35 71 00
F1?	f F? 1	16 23 01	h F? 1	35 71 01
F2?	f F? 2	16 23 02	h F? 2	35 71 02
F3?	f F? 3	16 23 03	h F? 3	35 71 03
FRC	f FRAC	16 44	g FRAC	32 83
FIX	FIX	-11	f FIX	31 23
GRAD	f GRD	16-23	h GRD	35 43
GSB0	GSB 0	23 00	f GSB 0	31 22 00
GSB1	GSB 1	23 01	f GSB 1	31 22 01
GSB2	GSB 2	23 02	f GSB 2	31 22 02
GSB3	GSB 3	23 03	f GSB 3	31 22 03
GSB4	GSB 4	23 04	f GSB 4	31 22 04
GSB5	GSB 5	23 05	f GSB 5	31 22 05
GSB6	GSB 6	23 06	f GSB 6	31 22 06
GSB7	GSB 7	23 07	f GSB 7	31 22 07
GSB8	GSB 8	23 08	f GSB 8	31 22 08
GSB9	GSB 9	23 09	f GSB 9	31 22 09
GSBA	GSB A	23 11	f GSB A	31 22 11
GSBB	GSB B	23 12	f GSB B	31 22 12
GSBC	GSB C	23 13	f GSB C	31 22 13
GSBD	GSB D	23 14	f GSB D	31 22 14
GSBE	GSB E	23 15	f GSB E	31 22 15
GSBa	GSB f a	23 16 11	g GSBf a	32 22 11
GSBb	GSB f b	23 16 12	g GSBf b	32 22 12
GSBc	GSB f c	23 16 13	g GSBf c	32 22 13
GSBd	GSB f d	23 16 14	g GSBf d	32 22 14
GSBe	GSB f e	23 16 15	g GSBf e	32 22 15
GSBi	GSB (i)	23 45	f GSB (i)	31 22 24
GTO0	GTO 0	22 00	GTO 0	22 00
GTO1	GTO 1	22 01	GTO 1	22 01
GTO2	GTO 2	22 02	GTO 2	22 02
GTO3	GTO 3	22 03	GTO 3	22 03
GTO4	GTO 4	22 04	GTO 4	22 04
GTO5	GTO 5	22 05	GTO 5	22 05
GTO6	GTO 6	22 06	GTO 6	22 06
GTO7	GTO 7	22 07	GTO 7	22 07
GTO8	GTO 8	22 08	GTO 8	22 08
GTO9	GTO 9	22 09	GTO 9	22 09

HP-97 Tape Symbol	HP-97 Keystrokes	HP-97 Keycode	HP-67 Keystrokes	HP-67 Keycode
GTOA	GTO A	22 11	GTO A	22 11
GTOB	GTO B	22 12	GTO B	22 12
GTOC	GTO C	22 13	GTO C	22 13
GTOD	GTO D	22 14	GTO D	22 14
GTOE	GTO E	22 15	GTO E	22 15
GTOa	GTO f a	22 16 11	GTO f a	22 31 11
GTOb	GTO f b	22 16 12	GTO f b	22 31 12
GTOc	GTO f c	22 16 13	GTO f c	22 31 13
GTOd	GTO f d	22 16 14	GTO f d	22 31 14
GTOe	GTO f e	22 16 15	GTO f e	22 31 15
GTOi	GTO (i)	22 45	GTO (i)	22 24
→HMS	f →H.MS	16 35	g →H.MS	32 74
HMS→	f H.MS→	16 36	f H←	31 74
HMS+	f H.MS+	16-55	h H.MS+	35 83
INT	f INT	16 34	f INT	31 83
ISZI	f ISZ I	16 26 46	f ISZ	31 34
ISZi	f ISZ (i)	16 26 45	g ISZ (i)	32 34
*LBL0	LBL 0	21 00	f LBL 0	31 25 00
*LBL1	LBL 1	21 01	f LBL 1	31 25 01
*LBL2	LBL 2	21 02	f LBL 2	31 25 02
*LBL3	LBL 3	21 03	f LBL 3	31 25 03
*LBL4	LBL 4	21 04	f LBL 4	31 25 04
*LBL5	LBL 5	21 05	f LBL 5	31 25 05
*LBL6	LBL 6	21 06	f LBL 6	31 25 06
*LBL7	LBL 7	21 07	f LBL 7	31 25 07
*LBL8	LBL 8	21 08	f LBL 8	31 25 08
*LBL9	LBL 9	21 09	f LBL 9	31 25 09
*LBLA	LBL A	21 11	f LBL A	31 25 11
*LBLB	LBL B	21 12	f LBL B	31 25 12
*LBLC	LBL C	21 13	f LBL C	31 25 13
*LBLD	LBL D	21 14	f LBL D	31 25 14
*LBLE	LBL E	21 15	f LBL E	31 25 15
*LBLa	LBL f a	21 16 11	g LBL f a	32 25 11
*LBLb	LBL f b	21 16 12	g LBL f b	32 25 12
*LBLc	LBL f c	21 16 13	g LBL f c	32 25 13
*LBLd	LBL f d	21 16 14	g LBL f d	32 25 14
*LBLe	LBL f e	21 16 15	g LBL f e	32 25 15
LN	LN	32	f LN	31 52
LOG	f LOG	16 32	f LOG	31 53
LSTX	f LAST x	16-63	h LST x	35 82
–	–	-45	–	51
MRG	f MERGE	16-62	g MERGE	32 41
N!	f N!	16 52	h N!	35 81
→P	→P	34	g →P	32 72

HP-97 Tape Symbol	HP-97 Keystrokes	HP-97 Keycode	HP-67 Keystrokes	HP-67 Keycode
%	%	55	f %	31 82
%CH	f %CH	16 55	g %CH	32 82
Pi	f π	16–24	h π	35 73
+	+	–55	+	61
PREG	f REG	16–13	h REG	35 74
PRST	f STACK	16–14	g STK	32 84
PRTX	PRINT x	–14	f –x–	31 84
P⇄S	f P≷S	16–51	f P≷S	31 42
PSE	f PAUSE	16 51	h PAUSE	35 72
→R	→R	44	f R←	31 72
R↓	R↓	–31	h R↓	35 53
R↑	f R↑	16–31	h R↑	35 54
RAD	f RAD	16–22	h RAD	35 42
R→D	f R→D	16 46	f D←	31 73
RCL0	RCL 0	36 00	RCL 0	34 00
RCL1	RCL 1	36 01	RCL 1	34 01
RCL2	RCL 2	36 02	RCL 2	34 02
RCL3	RCL 3	36 03	RCL 3	34 03
RCL4	RCL 4	36 04	RCL 4	34 04
RCL5	RCL 5	36 05	RCL 5	34 05
RCL6	RCL 6	36 06	RCL 6	34 06
RCL7	RCL 7	36 07	RCL 7	34 07
RCL8	RCL 8	36 08	RCL 8	34 08
RCL9	RCL 9	36 09	RCL 9	34 09
RCLA	RCL A	36 11	RCL A	34 11
RCLB	RCL B	36 12	RCL B	34 12
RCLC	RCL C	36 13	RCL C	34 13
RCLD	RCL D	36 14	RCL D	34 14
RCLE	RCL E	36 15	RCL E	34 15
RCLI	RCL I	36 46	h RC I	35 34
RCLi	RCL (i)	36 45	RCL (i)	34 24
RCLΣ	RCL Σ+	36 56	RCL Σ+	34 21
RND	f RND	16 24	f RND	31 24
R/S	R/S	51	R/S	84
RTN	RTN	24	h RTN	35 22
S	f s	16 54	g s	32 21
SCI	SCI	–12	g SCI	32 23
SF0	f STF 0	16 21 00	h SF 0	35 51 00
SF1	f STF 1	16 21 01	h SF 1	35 51 01
SF2	f STF 2	16 21 02	h SF 2	35 51 02
SF3	f STF 3	16 21 03	h SF 3	35 51 03
Σ+	Σ+	56	Σ+	21
Σ-	f Σ–	16 56	h Σ–	35 21
SIN	SIN	41	f SIN	31 62

HP-97 Tape Symbol	HP-97 Keystrokes	HP-97 Keycode	HP-67 Keystrokes	HP-67 Keycode
SIN^{-1}	f SIN^{-1}	16 41	g SIN^{-1}	32 62
SPC	f SPACE	16–11	h SPACE	35 84
√X	√x	54	f √x	31 54
ST÷0	STO ÷ 0	35–24 00	STO ÷ 0	33 81 00
ST÷1	STO ÷ 1	35–24 01	STO ÷ 1	33 81 01
ST÷2	STO ÷ 2	35–24 02	STO ÷ 2	33 81 02
ST÷3	STO ÷ 3	35–24 03	STO ÷ 3	33 81 03
ST÷4	STO ÷ 4	35–24 04	STO ÷ 4	33 81 04
ST÷5	STO ÷ 5	35–24 05	STO ÷ 5	33 81 05
ST÷6	STO ÷ 6	35–24 06	STO ÷ 6	33 81 06
ST÷7	STO ÷ 7	35–24 07	STO ÷ 7	33 81 07
ST÷8	STO ÷ 8	35–24 08	STO ÷ 8	33 81 08
ST÷9	STO ÷ 9	35–24 09	STO ÷ 9	33 81 09
ST-0	STO − 0	35–45 00	STO − 0	33 51 00
ST-1	STO − 1	35–45 01	STO − 1	33 51 01
ST-2	STO − 2	35–45 02	STO − 2	33 51 02
ST-3	STO − 3	35–45 03	STO − 3	33 51 03
ST-4	STO − 4	35–45 04	STO − 4	33 51 04
ST-5	STO − 5	35–45 05	STO − 5	33 51 05
ST-6	STO − 6	35–45 06	STO − 6	33 51 06
ST-7	STO − 7	35–45 07	STO − 7	33 51 07
ST-8	STO − 8	35–45 08	STO − 8	33 51 08
ST-9	STO − 9	35–45 09	STO − 9	33 51 09
ST+0	STO + 0	35–55 00	STO + 0	33 61 00
ST+1	STO + 1	35–55 01	STO + 1	33 61 01
ST+2	STO + 2	35–55 02	STO + 2	33 61 02
ST+3	STO + 3	35–55 03	STO + 3	33 61 03
ST+4	STO + 4	35–55 04	STO + 4	33 61 04
ST+5	STO + 5	35–55 05	STO + 5	33 61 05
ST+6	STO + 6	35–55 06	STO + 6	33 61 06
ST+7	STO + 7	35–55 07	STO + 7	33 61 07
ST+8	STO + 8	35–55 08	STO + 8	33 61 08
ST+9	STO + 9	35–55 09	STO + 9	33 61 09
ST×0	STO × 0	35–35 00	STO × 0	33 71 00
ST×1	STO × 1	35–35 01	STO × 1	33 71 01
ST×2	STO × 2	35–35 02	STO × 2	33 71 02
ST×3	STO × 3	35–35 03	STO × 3	33 71 03
ST×4	STO × 4	35–35 04	STO × 4	33 71 04
ST×5	STO × 5	35–35 05	STO × 5	33 71 05
ST×6	STO × 6	35–35 06	STO × 6	33 71 06
ST×7	STO × 7	35–35 07	STO × 7	33 71 07
ST×8	STO × 8	35–35 08	STO × 8	33 71 08
ST×9	STO × 9	35–35 09	STO × 9	33 71 09

HP-97 Tape Symbol	HP-97 Keystrokes	HP-97 Keycode	HP-67 Keystrokes	HP-67 Keycode
ST÷i	STO ÷ (i)	35-24 45	STO ÷ (i)	33 81 24
ST-i	STO − (i)	35-45 45	STO − (i)	33 51 24
ST+i	STO + (i)	35-55 45	STO + (i)	33 61 24
ST×i	STO × (i)	35-35 45	STO × (i)	33 71 24
ST00	STO 0	35 00	STO 0	33 00
ST01	STO 1	35 01	STO 1	33 01
ST02	STO 2	35 02	STO 2	33 02
ST03	STO 3	35 03	STO 3	33 03
ST04	STO 4	35 04	STO 4	33 04
ST05	STO 5	35 05	STO 5	33 05
ST06	STO 6	35 06	STO 6	33 06
ST07	STO 7	35 07	STO 7	33 07
ST08	STO 8	35 08	STO 8	33 08
ST09	STO 9	35 09	STO 9	33 09
STOA	STO A	35 11	STO A	33 11
STOB	STO B	35 12	STO B	33 12
STOC	STO C	35 13	STO C	33 13
STOD	STO D	35 14	STO D	33 14
STOE	STO E	35 15	STO E	33 15
STOI	STO I	35 46	h ST I	35 33
STOi	STO (i)	35 45	STO (i)	33 24
TAN^{-1}	f TAN^{-1}	16 43	g TAN^{-1}	32 64
TAN	TAN	43	f TAN	31 64
×	×	-35	×	71
WDTA	f W/DATA	16-61	f W/DATA	31 41
X≠0?	f x≠0?	16-42	f x≠0	31 61
X=0?	f x=0?	16-43	f x=0	31 51
X>0?	f x>0?	16-44	f x>0	31 81
X<0?	f x<0?	16-45	f x<0	31 71
X≠Y?	f x≠y?	16-32	g x≠y	32 61
X=Y?	f x=y?	16-33	g x=y	32 51
X>Y?	f x>y?	16-34	g x>y	32 81
X≤Y?	f x≤y?	16-35	g x≤y	32 71
$\bar{x}$	f $\bar{x}$	16 53	f $\bar{x}$	31 21
X^2	x^2	53	g x^2	32 54
X⇄I	f x⇄I	16-41	h x⇄I	35 24
X⇄Y	x⇄y	-41	h x⇄y	35 52
Y^X	y^x	31	h y^x	35 63

Index

Service Information

Must be **completed** and **returned** with your calculator, charger and batteries

Owner's Name Date Purchased

Home Phone Work Phone

Ship-to address for returning repaired calculator

Street Address City State Zip

Describe Problem:

Model No. Serial No.

Preferred method of payment for out-of-warranty repairs. **If not specified, unit will be returned C.O.D.** ☐ BankAmericard ☐ Master Charge

Card No. Expiration Date

Name appearing on credit card

☐ Purchase Order, companies with established Hewlett-Packard credit only. (Include copy of Purchase Order with shipment.)

Authorized Signature P.O. Number

HEWLETT PACKARD

Calculator Catalog and Buying Guide Request Card

A friend or associate might also want to know about Hewlett-Packard calculators. If you would like us to send the Hewlett-Packard Calculator Catalog and Buying Guide (with interesting articles, features and letters), please mail his/her name and address on this postage paid Request Card.

Primary Interest:

☐ Scientific Calculators

☐ Scientific Printing Calculators

☐ Fully Programmable Calculators

☐ Business Calculators

☐ All

Name

Title

Company

Street

City State Zip

Service Card

Refer to the appendix of your Owner's Handbook to diagnose a calculator malfunction. The warranty period for your calculator is one year from date of purchase. Unless **Proof of Purchase** is enclosed (sales slip or validation) Hewlett-Packard will assume any unit over 12 months old is out of warranty. (**Proof of Purchase** will be returned with your calculator.) Should service be required, please return your calculator, charger, batteries and this card protectively packaged to avoid in-transit damage. Such damage is not covered under warranty.

Inside the U.S.A.
Complete the reverse side of this card and return items safely packaged directly to:

Hewlett-Packard
APD Service Department
P.O. Box 999
Corvallis, Oregon 97330

We advise that you insure your calculator and use priority (AIR) mail for distances greater than 300 miles to minimize transit times. All units will be returned via priority mail.

Outside the U.S.A.
Where required please fill in the validation below and return your unit to the nearest designated Hewlett-Packard Sales and Service Office. Your warranty will be considered invalid if this completed card is not returned with the calculator.

Model No.	Serial No.	Date Received

Invoice No./Delivery Note No.

Sold By:

FIRST CLASS

Permit No.
33

Corvallis,
Oregon

BUSINESS REPLY MAIL

No postage stamp necessary if mailed in the United States

Postage will be paid by:

Hewlett-Packard
1000 N.E. Circle Blvd.
Corvallis, Oregon 97330

Useful Conversion Factors

The following factors are provided to 10 digits of accuracy where possible. Exact values are marked with an asterisk. For more complete information on conversion factors, refer to *Metric Practice Guide E380-74* by the American Society for Testing and Materials (ASTM).

Length

1 inch = 25.4 millimeters*
1 foot = 0.304 8 meter*
1 mile (statute)† = 1.609 344 kilometers*
1 mile (nautical)† = 1.852 kilometers*
1 mile (nautical)† = 1.150 779 448 miles (statute)†

Area

1 square inch = 6.451 6 square centimeters*
1 square foot = 0.092 903 04 square meter*
1 acre = 43 560 square feet
1 square mile† = 640 acres

Volume

1 cubic inch = 16.387 064 cubic centimeters*
1 cubic foot = 0.028 316 847 cubic meter
1 ounce (fluid)† = 29.573 529 56 cubic centimeters
1 ounce (fluid)† = 0.029 573 530 liter
1 gallon (fluid)† = 3.785 411 784 liters*

Mass

1 ounce (mass) = 28.349 523 12 grams
1 pound (mass) = 0.453 592 37 kilogram*
1 ton (short) = 0.907 184 74 metric ton*

Energy

1 British thermal unit = 1 055.055 853 joules
1 kilocalorie (mean) = 4 190.02 joules
1 watt-hour = 3 600 joules*

Force

1 ounce (force) = 0.278 013 85 newton
1 pound (force) = 4.448 221 615 newtons

Power

1 horsepower (electric) = 746 watts*

Pressure

1 atmosphere = 760 mm Hg at sea level
1 atmosphere = 14.7 pounds per square inch
1 atmosphere = 101 325 pascals

Temperature

Fahrenheit = 1.8 Celsius + 32
Celsius = 5/9 (Fahrenheit − 32)
kelvin = Celsius + 273.15
kelvin = 5/9 (Fahrenheit + 459.67)
kelvin = 5/9 Rankine

---

† U.S. values chosen. * Exact values.

HEWLETT hp PACKARD

1000 N.E. Circle Blvd., Corvallis, OR 97330

For additional Sales and Service Information contact your local Hewlett-Packard Sales Office or call 800-648-4711 toll-free. (In Nevada call collect, 702-323-2704.)

00097-90001, Rev. F 9/78

Printed in Singapore