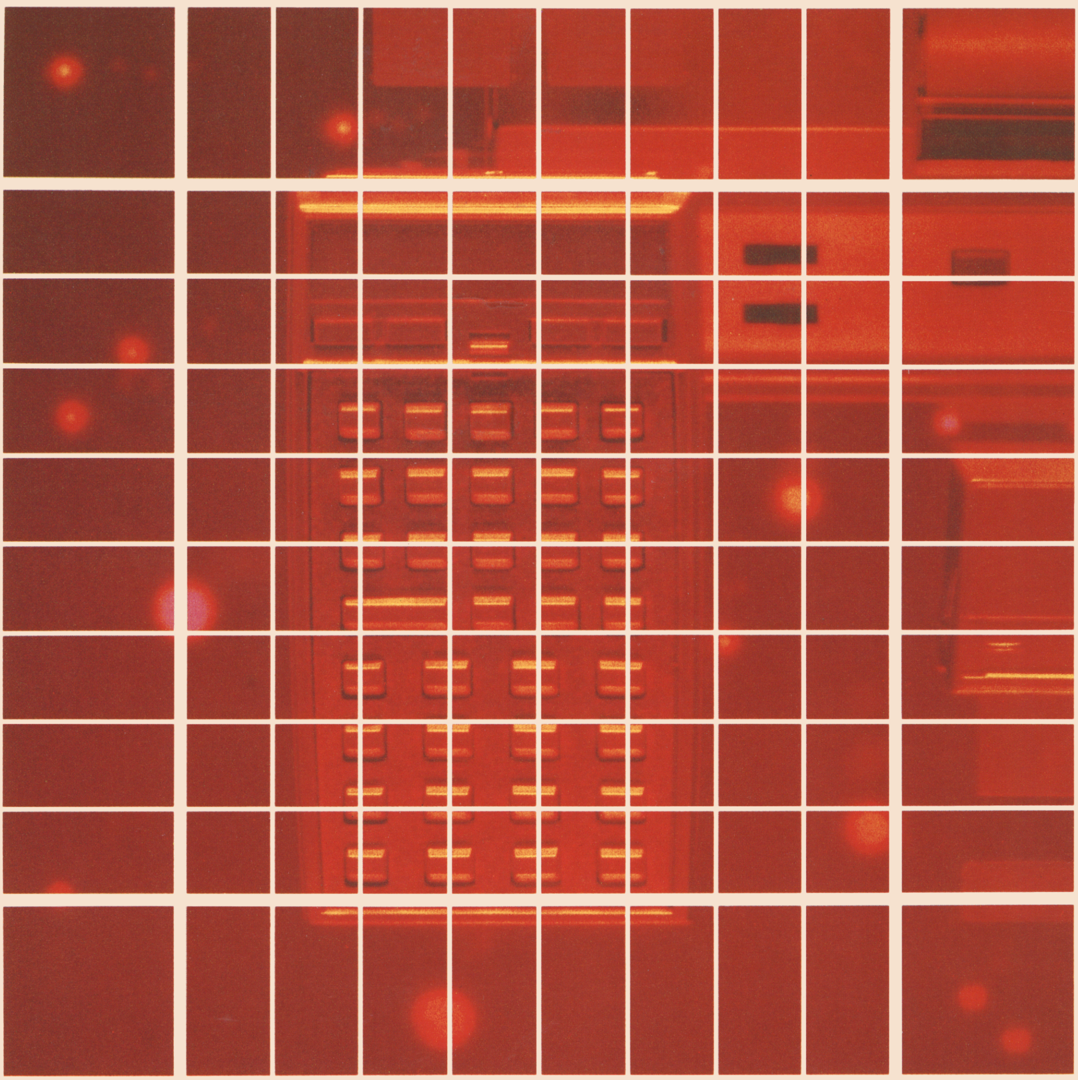
HEWLETT-PACKARD

HP-41CV

OWNER'S MANUAL

BASIC OPERATION



Notice

HEWLETT PACKARD

HP-41CV
Owner's Manual

Basic Operation

May 1984

00041-90526

Printed in Singapore

Introducing the HP-41CV

The HP-41CV belongs to the HP-41 family of compact, handheld computers. It combines a rich instruction set (over 100 built-in functions) with 319 registers of main memory for data storage, pro-program lines, and user-defined keys.

The HP-41CV features:

- A powerful set of numeric and mathematical functions.
- An alternate User keyboard: you can assign your most-used functions and programs to this keyboard for easy execution.
- Expandability: there are four input/output ports for connecting peripheral devices and applications software. Hewlett-Packard offers a range of peripherals—printers, card reader, cassette drive, video interface and monitor, bar code wand—and plug-in software modules to expand the capabilities and function set of the HP-41CV as the controller of a *system* of devices.
- Keystroke programming with easy program editing and automatic line numbering. Operations use RPN (Reverse Polish Notation) logic and a memory stack. Program features include labels, branching, subroutines, input prompting, loop control functions, indirect addressing, and flag operations.
- Continuous Memory, retaining data and program instructions indefinitely.
- Low power consumption and long battery life.

All programs written for the HP-41C (including those in plug-in modules) can be used with the HP-41CV without modification. Programs written for the HP-41CX, however, will not necessarily run on the HP-41CV, since the HP-41CX has more functions.

Basic Operation
Contents

List of Diagrams and Tables

Diagrams

Tables

How to Use This Manual

The HP-41 is a powerful handheld computer with a broad range of functions.* The *HP-41CV Owner's Manual* is an introduction to most aspects of operation, written for people who are new at using the HP-41. It is subtitled "Basic Operation" because it covers standard operation of the HP-41. It is organized for *tutorial use*, and will teach you how to use the HP-41 competently.

- Are you already familiar with RPN operating logic and HP calculators? If so, then you can skim most of sections 1, 3, and 5. However, don't skip the discussions of the Alpha register and Alpha characters. Read carefully sections 2 and 4.
- Are you interested in using the HP-41 *only* to run prewritten software? If so, read sections 1, 2, and 6 (noting especially the COPY command) so you will know how to handle the HP-41 and its software.

For more advanced users who would like to learn as much as possible about the HP-41—especially about programming—there is another manual, called *HP-41CV Operation in Detail.* You can order it from your HP dealer or from Hewlett-Packard. This book addresses different topics as well as a different level of experience and need. It comprehensively and extensively covers all topics, and is organized for *reference use.*

For First-Time Users

This manual is the place to start if you are not familiar with the HP-41 or other Hewlett-Packard calculators. It contains an introduction to HP-41 logic, keyboard and display lay-out, standard functions, and basic programming. This teaches you basic operation as quickly as possible, with a minimum of detailed or advanced information. You do not need any prior knowledge of handheld computers and programming. When you finish these chapters, you should feel well acquainted and comfortable with the HP-41. There is also an appendix with error messages, an appendix on warranty and service information, a Subject Index, and a Function Index.

As your understanding and use of the HP-41 increase, you might want to know more about its operation. Then it's time to order *HP-41 Operations in Detail.*

* For simplicity, this manual usually refers to the HP-41CV as the HP-41.

For More Experienced Users

If you are an experienced Hewlett-Packard calculator user, you might want to obtain *HP-41CV Operation in Detail.* This contains "Fundamentals in Detail" (including the memory stack) and "Programming in Detail", which comprehensively discuss the use of the HP-41. This manual is available from your HP dealer or from Hewlett-Packard.

Reference

For reference use, this manual includes:

- A comprehensive subject index.
- An alphabetical function index (on the inside of the back cover).
- Function Tables—a comprehensive summary of all functions (just before the subject index).
- Appendix information about error messages and warranty/service information.

Basic HP-41 Operation

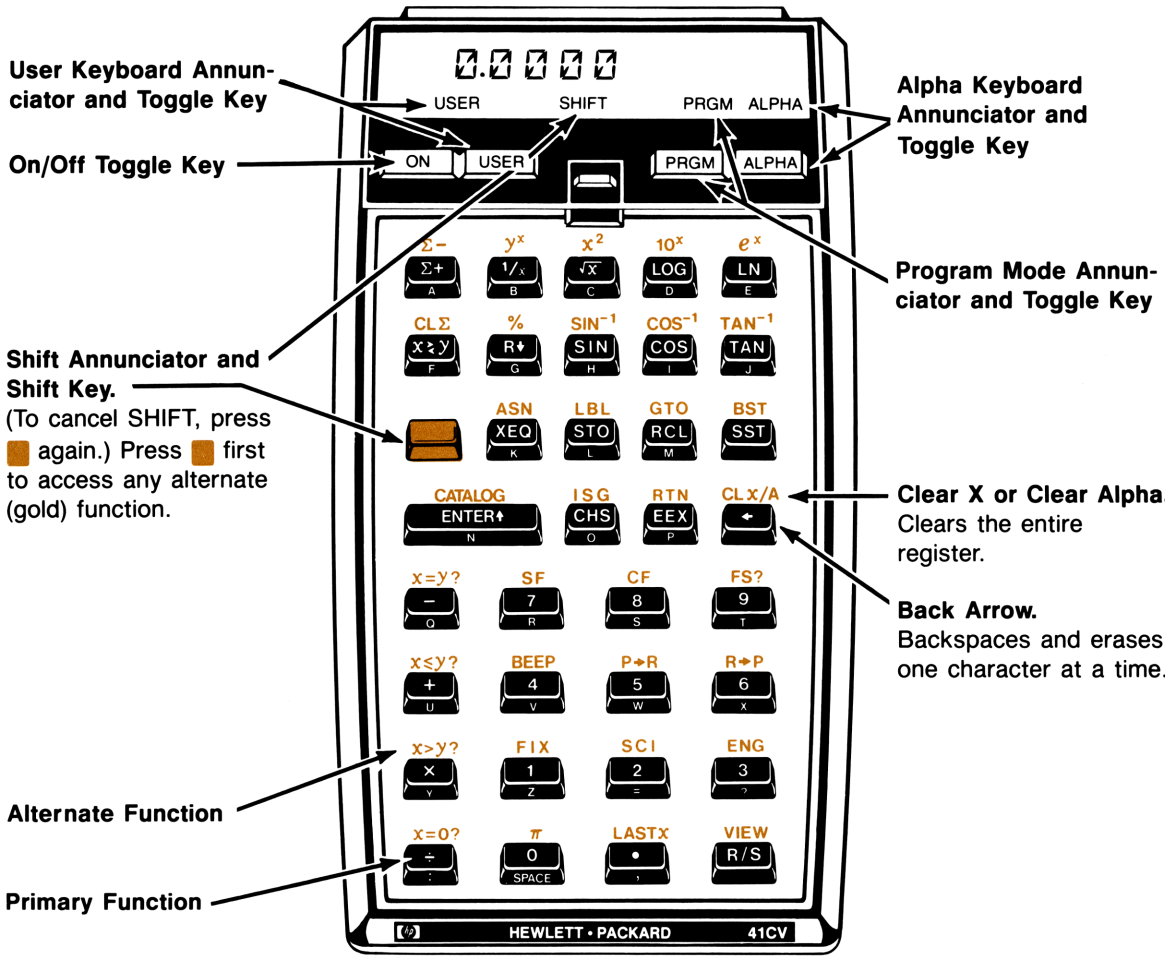
Section 1

Using the Keyboard

Contents

This section provides a detailed orientation to the HP-41 keyboard: what different parts of the keyboard mean, how to enter numbers and do simple calculations, how to clear the display, and what Continuous Memory does. These are the nuts and bolts you will need in order to utilize the sophistication and power of the HP-41. Some of these features are unique to the HP-41; some—like doing calculations—are common to other HP calculators.

Keyboard Overview



The Toggle Keys

The top four keys above the keyboard are very special: each one creates an operating condition that affects how the other 35 keys will be interpreted. They are called *toggle* keys since you press the key once to set a new condition, and press it again to restore the original condition.

The four conditions these toggle keys control are: turning the power on/off, activating/deactivating the User keyboard, going into/out of Program mode, and activating/deactivating the Alpha keyboard (explained below).

Power On and Off

The [ON] key turns the HP-41 on and off. After about 10 minutes of being on but inactive, the computer shuts itself off to conserve battery power. This is called "timing out." The Continuous Memory feature maintains programs, data, and most aspects of operating status while the computer is off.

Should you see **MEMORY LOST** in the display the first time you turn on the HP-41, this means the computer's memory has been cleared. Press the [←] key to remove this message.

The Normal Keyboard and the User Keyboard

The *Normal* keyboard of the HP-41 is comprised of the standard set of keys as you see them printed on and above the keys. The *primary* functions are printed in white on the keys, while the *alternate* functions are printed in gold above the keys.

The *User* keyboard is comprised of an alternative, "customized" set of functions. It is based on the Normal keyboard, but includes substitutions due to any new key definitions that you make. You might find it handy, as you learn more about the HP-41, to redefine some keys on the User keyboard to perform different operations.

Pressing [USER] will activate or deactivate the User keyboard. The **USER** annunciator appears in the bottom of the display when the User keyboard is active. (Assigning User functions is covered in section 3.)

The Alpha Keyboard

A major feature of the HP-41 is its alphabetic (or "Alpha") character set, which also includes digits and other special characters. Using the Alpha keyboard, you can store messages (especially useful in programs), and gain access to many more operations than those printed on the keyboard.

The character set of the Alpha keyboard consists of the blue primary characters printed on the slanted face of the keys *and* the shifted alternate characters shown (along with the primary characters) in the diagram on page 23, on the back plate of the HP-41CX, and in the Quick Reference Guide.

Pressing [ALPHA] activates or deactivates the Alpha keyboard. The **ALPHA** annunciator comes on when the Alpha keyboard is active. (Using the Alpha keyboard is discussed further in this section under "Entering Alphabetic Characters" on page 22.)

Execution Mode and Program Mode

The HP-41 has two basic operating modes: Execution mode and Program mode. It always "wakes up" in Execution mode, the standard mode for *executing* functions (performing calculations) from the keyboard and for *executing* programs. Program mode, on the other hand, is only for *storing* programs: pressing keys in Program mode does not execute operations, but instead records them as program instructions for later execution.

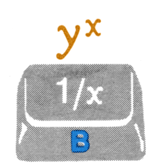
The [PRGM] toggle key switches the computer into or out of Program mode. The **PRGM** annunciator is on in Program mode, as well as during execution of a program (in Execution mode). (Programming is discussed in section 6.)

Keyboard Conventions

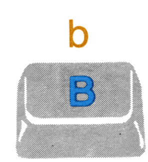
Primary and Alternate (Shifted) Keyboard Functions

The HP-41CV has three different sets of keyboard functions, referred to as *keyboards*. These keyboards—the Normal, User, and Alpha keyboards—use both primary and alternate functions. To execute alternate functions, press the gold shift key ( ) *before* pressing the function key. (This works differently than the shift key on a typewriter: do not hold the shift key *while* pressing the function key.)

- On the Normal (or User) keyboard, select the primary function by pressing only that key: [1/x].
- On the Normal (or User) keyboard, select the alternate function by first pressing the shift key, then the desired key: [y^x].
- On the Alpha keyboard (**ALPHA** annunciator on), select the primary character, which is printed in blue on the slanted face of the key, by pressing only that key: [B].
- On the Alpha keyboard, select the alternate character or function by first pressing the shift key, then the function key: [b].



Normal



Alpha

The alternate Alpha character or function is printed above the key in the Alpha keyboard diagrams on page 23 and in the Quick Reference Guide. Most of the Alpha keyboard functions are shown on the back plate of the HP-41. The face of the HP-41 itself does *not* indicate the alternate Alpha functions.

The SHIFT Annunciator. Whenever you press the shift key, the **SHIFT** annunciator appears in the display. The annunciator disappears when you finish executing the keystroke sequence.

Cancelling a Shift. To cancel a shift (before pressing another key), just press the shift key again. The **SHIFT** annunciator will turn off.

How This Manual Represents Keystrokes

This manual uses the following notation to represent keystrokes on the HP-41:

Key Boxes.

- Any HP-41 *function* is represented as it appears on the keyboard, with a box around it to simulate a key. For example: [STO], [ASTO].
- Digits and Alpha characters appear without boxes, even though they represent keystrokes. For example: A , a .
- Letters in black key boxes represent special functions, such as [X] (to indicate the X-register).

Key Colors.

- All primary Normal functions are printed in black—such as [RCL].
- Non-keyboard functions are printed in blue. For example: [MEAN].
- All primary Alpha characters are printed in blue—such as A.
- All shifted functions or characters are printed in gold—such as [GTO] and [ARCL]. *The shift key is not shown before shifted Normal and Alpha functions and characters*; it is implied by the gold color.
- Black letters in key boxes represent special functions, and *not Alpha characters*. For example: [X].

Typeface. Some functions must be followed by a parameter. A different typeface is used to indicate the required parameter, such as [GTO] *global label*.

Doing Simple Calculations

If you make a mistake while entering numbers, use [←] to correct it. (For more on this clearing function, see page 17.) Don't use digits from the Alpha keyboard for calculations. They are considered just characters, not numbers.

For doing calculations with more than one number (like simple arithmetic operations), the HP-41 uses RPN.* This is also called reverse entry: whenever you have two numbers for an operation, you enter both numbers *before* pressing the function key. The order of entering those two numbers is the same as it is when you write numbers in an equation from left to right. After pressing the function key, you will see the result. (There is no [=] key!) That is, pressing the function key *executes* that function.
The key to reverse entry is the [ENTER↑] key. Use [ENTER↑] between two sequentially keyed-in numbers to separate them. Then follow with the function key to execute the operation.

If you want to perform an operation that uses only one number, like [SIN], then you don't need to use [ENTER↑]. (This is true whether you key in that one number or whether you use a number that is already in the display—a number that was the result of a previous calculation.)

15 − 3 = *result*	sin 0 = *result*
becomes	becomes
15 [ENTER↑] 3 [−] *result*	0 [SIN] *result*

Example: The keystroke sequences below illustrate the execution of a one-number function and two two-number functions. Notice that [ENTER↑] is necessary only in the second calculation, and not in the first or third one. (If you make a typing error, use [←] to clear it.)

Find: 1. $\sqrt{45}$
2. 16.4/1.8
3. (the result of #2) + 67.1234

Keystrokes	Display	
45	45_	Digit entry not terminated.
[$\sqrt{x}$]	6.7082	[$\sqrt{x}$] function terminates digit entry and executes $\sqrt{45}$.
16.4	16.4_	New number.
[ENTER↑]	16.4000	Terminates digit entry. [ENTER↑] will separate two numbers keyed in sequentially.
1.8	1.8_	New number.
[÷]	9.1111	Result of 16.4/1.8.
67.1234	67.1234_	[ENTER↑] is not necessary between a result and a new number.
[+]	76.2345	Result of 9.1111 + 67.1234.

* HP operating logic is based on a mathematical logic known as "Polish Notation," developed by the noted Polish logician Jan Łukasiewicz (*Wookashye'veech*) (1878–1956). Conventional algebraic notation places operators *between* the relevant numbers or variables when evaluating algebraic expressions. Łukasiewicz's notation specifies the operators *before* the variables. A variation of this logic specifies the operators *after* the variables. This is termed "Reverse Polish Notation." For optimal efficiency in digital computation, HP adopted this convention of entering the operators after entering the variable(s).

Entering Numbers

Terminating Digit Entry

When two numbers are keyed into the HP-41, they need to be separated. One of the purposes [ENTER↑] serves is to separate two numbers by terminating digit entry. For any number being *keyed in*, the computer needs a signal when digit entry is complete. [ENTER↑] does this for the first number you key in, and any other function key you press terminates the digit entry of the second number you key in.

If, on the other hand, one of the numbers you are using is already in the computer as the result of a previous operation, you do not need to use the [ENTER↑] key. This is because *all operations except the digit entry keys themselves have the effect of terminating digit entry.* The digit entry keys are: the digit keys, [·], [CHS], [EEX], and [←].

The HP-41 also provides you with its own visual indication of digit entry status: the input cue (_). While entering a number, you will see the _ after the rightmost digit, like a blank waiting to be filled in. (You can see this in the previous example.) The presence of the _ cue means that another digit can be accepted for the current number. When digit entry has been terminated, however, the _ is gone because that entry is complete.

Changing Signs

Pressing [CHS] (*change sign*) will change the sign (positive or negative) of a displayed number. To key in a negative number, press [CHS] after keying in its digits.

Keying in Exponents

Use [EEX] (*enter exponent*) to key in a number with an exponent. First key in the base part of the number, then press [EEX] and key in a one- or two-digit exponent. (A base part of one is assumed if you do not enter a number.) Notice [EEX] places the _ cue in the right side of the display to signal the numeric input it needs before completing its function.

For a negative exponent, press [CHS] after keying in the exponent. For a negative base, remember to press [CHS] *before* executing [EEX].

For example, key in Planck's constant (6.6262×10^{-34} Joule-seconds) and multiply it by 50:

Keystrokes	Display	
6.6262	6.6262_	
[EEX]	6.6262 _	The _ cue "asks" for the exponent.
3	6.6262 3_	Awaits a possible second digit.
4	6.6262 34	

Keystrokes	Display	
CHS	**6.6262 -34**	6.6262×10^{-34}.
ENTER↑	**6.6262 -34**	Enters number.
50 ×	**3.3131 -32**	Result in Joule-seconds.

Digits from the base part that spill into the exponent field will disappear from the display when EEX is pressed, but they will be retained internally.*

> **Note:** You should be aware that program instruction lines (Program mode) and printer output from the HP-41 use a different format for depicting numbers with exponents. In program lines and printer listings, the letter **E** appears before an exponent, such as **6.6262 E−34**. If you are trying to enter such a line, do it as shown above, using the EEX key. Do not press E (which is an Alpha character).

Pi

Pressing π places up to the first 10 digits of π into the display. (This also terminates digit entry, so π does not need to be separated from other numbers by ENTER↑.)

Note that the name of π used for all writing and printing purposes (such as in a program line or in printer output) is PI.

Clearing the Display

There are two types of display-clearing operations: CLx/A (*clear X or clear Alpha*) and ← (*back arrow*).

Clearing Digits and Characters

In Execution mode:

- CLx/A represents the functions CLx on the Normal keyboard and CLA on the Alpha keyboard. CLx clears the entire display (X-register) to zero, and CLA clears the Alpha display and Alpha register to blank.†
- ← deletes only the last digit or character in the display *if digit/character entry has not been terminated*. If digit/character entry *has* been terminated, then ← acts like CLx/A.

* EEX will operate with a base part having more than eight digits only if a decimal point is keyed in before the ninth digit.

† The X-register is part of the automatic memory stack, which is explained in *HP-41CV Operation in Detail*. The Alpha register is separate from the stack.

Example: To see how these two features operate, try the following keystroke sequence. What you see in the computer display should match the displays given below. (If your display does not show four decimal places, press [FIX] 4 to make it conform to the displays below.)

Keystrokes	Display	
12345	12,345_	Digit entry not terminated. The _ input cue indicates more digits can be added.
[CLx/A]	0.0000	[CLx/A] clears the entire display to zero.
12345	12,345_	Digit entry not terminated.
[←]	1,234_	Clears only the last digit.
9	12,349_	Digit entry can continue.
[√x]	111.1261	Square root operation terminates digit entry (no _ cue present).
[←]	0.0000	[←] now clears all digits to zero.

Using clearing functions in Program mode is covered in section 6.

Clearing Other Displays

Clearing Partial Key Sequences (Parameter Functions). There are several *parameter functions*, which require the entry of a key sequence: a *prefix* key (like [STO]) and then one or more digits (the parameter). You can recognize a parameter function because it appears in the display with one or more input cues (_), waiting for numeric or Alpha input. If you mistakenly press a prefix key and stop before completing the full key sequence, you can clear this function by pressing [←].

Clearing MEMORY LOST and Error Messages. If MEMORY LOST or any other error message appears and remains in the display, you can clear it by pressing [←] or any key (except ■ and [USER]). [←] will restore the previous display; any other key will execute its own function. (Error messages and their meanings are covered on page 32 in section 2, "The Display.")

Doing Longer Calculations

The use of RPN (page 15) and the "automatic memory stack" make it possible to do long calculations on the HP-41 without using parentheses or storing intermediate results. This is because the computer, using its own memory stack, stores and recalls previous (or *intermediate*) results, which you can then use in subsequent calculations.

Very briefly, the memory stack is four registers "high," labeled as shown below. It holds your current and most recent entries. Generally, it is the contents of the X-register that you see displayed—it holds your most recent numeric entry. When you press [ENTER↑] or key in another number following an operation, the number in the X-register gets "pushed up" into the Y-register. The contents of the Y-, Z-and T-registers also move up one level; the number that was in the T-register is then lost off the top of the stack. The stack is what determines how many intermediate results the computer can hold in the course of a calculation—since there are four registers, it can retain four numbers. As you execute operations, the stack generally drops as two numbers are replaced by one result. (In the diagram below, x, y, z, and t represent numbers.)

T	t	z	z	z	z
Z	z	y	y	z	z
Y	y	x	x	y	z
X	x	x	a	$x-a$	$y(x-a)$

Keys: [ENTER↑] a [−] [×]

Since knowledge of the HP-41 automatic memory stack is not essential to your being able to perform routine calculations on the HP-41, it is not explained in detail here. However, learning about the stack is very helpful in understanding the logic and operation of the computer, enabling you to do complicated calculations and to go beyond simple programming. Refer to *HP-41CV Operation in Detail* for a comprehensive discussion of stack operation.

In the following calculations, notice that:

- The [ENTER↑] key is used only for separating the sequential entry of two numbers.
- The operator is keyed in only after both operands (numbers) are present.
- *The result of any operation may itself become an operand.* Such intermediate results are stored and retrieved on a last-in, first-out basis. New digits keyed in following an operation are treated as a new number.

Example: Calculate (9 + 17 − 4) ÷ 4.

Keystrokes	Display	
9 [ENTER↑]	9.0000	Digit entry terminated.
17 [+]	26.0000	(9 + 17).
4 [−]	22.0000	(9 + 17 − 4).
4 [÷]	5.5000	(9 + 17 − 4) ÷ 4.

Even more complicated problems are solved in this same way—using automatic storage and retrieval of intermediate results. For a given equation, it is easiest to work from the inside of parentheses outwards, just as you would when calculating on paper.

Example: Calculate (6 + 7) × (9 − 3).

Keystrokes	Display	
6 [ENTER↑]	6.0000	First solve for the intermediate result of (6 + 7).
7 [+]	13.0000	The intermediate result is displayed, so you can keep track of the calculation.
9 [ENTER↑]	9.0000	Then solve for the intermediate result of (9 − 3).
3 [−]	6.0000	
[×]	78.0000	Then multiply the intermediate results together (13 and 6) for the final answer.

For nested calculations, begin calculating at the innermost number or pair of parentheses, just as you would for a calculation on paper.

Example: Calculate 3 [4 + 5 (6 + 7)].

Keystrokes	Display	
6 [ENTER↑] 7 [+]	13.0000	(6 + 7).
5 [×]	65.0000	5 (6 + 7).
4 [+]	69.0000	4 + 5 (6 + 7).
3 [×]	207.0000	3 [4 + 5 (6 + 7)].

If an operation is noncommutative (namely subtraction and division), you can still enter all the numbers in the same order as for addition and multiplication as follows:

- Change *subtraction* into the *addition of a negative number*. Use [CHS] and [+].
- Change *division* into the *multiplication of a reciprocal*. Use [1/x] and [×].

or

- Reverse the order of the numbers after they have been entered by pressing the [x⪤y] key (*X exchange Y*). This function exchanges the contents of the X- and Y-registers.

Both of these methods are illustrated on the next page.

Example: Calculate $3 \div [4 - 5\,(6 + 7)]$.

Keystrokes	Display	
6 [ENTER↑] 7 [+]	13.0000	
5 [×]	65.0000	
[CHS]	-65.0000	
4 [+]	-61.0000	$4 + [-5\,(6 + 7)]$. Since subtraction is not commutative, use [CHS] and [+] instead of [−].
3	3	
[x≷y]	-61.0000	Reverse the order of the operands to perform the proper division.
[÷]	-0.0492	Result of $3 \div (-61)$.

Alternatively, you *can* enter operands from left to right—but the computer can hold no more than four intermediate operands or results. By proceeding from left to right, you don't have to alter any noncommutative operations. The example below has just four operands, so the whole equation can be entered from left to right.

Example: Calculate $3 \div [4 - 5\,(13)]$.

Keystrokes	Display	
3 [ENTER↑] 4 [ENTER↑]	4.0000	Use [ENTER↑] to separate the successive operands.
5 [ENTER↑] 13	13_	There are four intermediate operands (3, 4, 5, 13).
[×]	65.0000	Calculates 5×13.
[−]	-61.0000	Calculates $4 - 65$.
[÷]	-0.0492	Calculates $3 \div (-61)$.

Using Constants in Arithmetic Calculations

The LAST X register is a register related to the memory stack. It is described in *HP-41CV Operation in Detail.* The LAST X register preserves the numeric value that was last in the display (the X-register) before the execution of most numeric functions. You can recover this value using the function [LASTx].

Recovering and reusing a previous number can be useful in short calculations that use the same number more than once. Or, if you need to repeat a calculation using a constant, you can do so easily (without using a data storage register) with [LASTx].

To use the [LASTx] function for calculating with a constant, remember to enter your constant second, just before executing the arithmetic operation, so that the constant is the number saved by the LAST X register.

Example: Calculate $\dfrac{96.704 + 52.394706}{52.394706}$.

Keystrokes	Display	
96.704 [ENTER↑]	96.7040	
52.394706 [+]	149.0987	Intermediate result.
[LASTx]	52.3947	Brings back display before [+].
[÷]	2.8457	Final result.

Example: Two close stellar neighbors of Earth are Rigel Centaurus (4.3 light-years away) and Sirius (8.7 light-years away). Use *c*, the speed of light (9.5×10^{15} meters per year) to figure the distances from the Earth to these stars in meters.

Keystrokes	Display		
4.3 [ENTER↑]	4.3000		Light-years to Rigel Centaurus.
9.5 [EEX] 15	9.5	15	Speed of light, *c*.
[×]	4.0850	16	Answer: distance to R. Centaurus.
8.7 [LASTx]	9.5000	15	Enter light-years to Sirius; then retrieve *c*.
[×]	8.2650	16	Answer: distance to Sirius (meters).

Out-of-Range Results

Overflow. The result of a calculation with a magnitude greater than $9.999999999 \times 10^{99}$ causes an out-of-range error.* The display shows **OUT OF RANGE** and the overflow-causing function is *not* executed. This condition also causes a running program to stop.

To clear the error condition, press [←]. (Error displays in general are covered in section 2, page 32.)

Underflow. If the result of a calculation is a number with a magnitude less than $1.000000000 \times 10^{-99}$, that number will be replaced by zero. Underflow does not cause an error.

Entering Alphabetic Characters

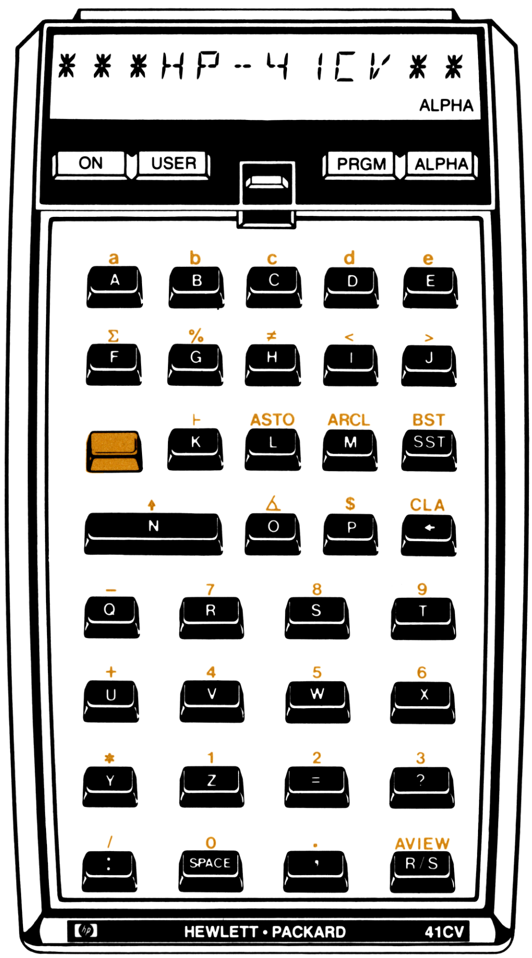
An alphabetic character set, which includes digits and special characters, is available on the HP-41 when you activate the Alpha keyboard. Press [ALPHA] to activate the Alpha keyboard. This simultaneously deactivates the Normal and User keyboards.

Refer to page 12 for a description of the Alpha keyboard character set (such as primary and shifted Alpha functions and how they are printed in this manual).

---

* Except when using the [Σ+] statistical function. Refer to "Statistical Functions" in section 5, page 54.

The Alpha Keyboard



The use of Alpha *strings*—sequences of Alpha characters—is very handy for program messages, and mandatory for executing HP-41 functions not printed on the keyboard.

Alpha Entry

Character Entry Termination. Once you activate the Alpha keyboard (**ALPHA** annunciator displayed) and enter one character, you will see the _ input cue in the display. This _ signals that the next character pressed will be added to the display—that is, character entry is not terminated.

When you are done with your entry, deactivate the Alpha keyboard. (Press [ALPHA]. Do not use [ENTER↑].) The Alpha display disappears and character entry is automatically terminated. This means that the next time you activate Alpha and enter characters, the old string will be erased and a new one will be entered.

Turning the computer off also terminates Alpha character entry. (Other functions that terminate Alpha character entry are listed in "Keying in Characters" in section 9.)

Alpha Clearing: [←] and [CLA]. The [←] operates during character entry just as it does during number entry. If character entry is not terminated, it clears—from the right—one character at a time. If character entry *is* terminated, then [←] acts like [CLA] and clears the entire Alpha display. [CLA] (like [CLx] in number entry) clears the entire display whether character entry is terminated or not.*

Alpha Digits. Remember, *digits from the Alpha keyboard* are simply Alpha characters and *cannot be used for numeric calculations.*

Example: The following keystrokes illustrate character entry with the Alpha keyboard.

Keystrokes	Display	
[ALPHA]		Display shows whatever message was last entered. **ALPHA** annunciator displayed.
HP-41 CX	HP-41CX_	The _ cue indicates that character entry is not yet terminated.
[←] V	HP-41CV_	Removes last character; adds V.
[ALPHA]		Terminates character entry and restores the prior numeric display.
[ALPHA]	HP-41CV	Displays Alpha register. No _ cue present. If you key in any characters, the previous characters will be erased and the new ones will start a new string.

*[CLx], the Normal keyboard function, and [CLA], the Alpha keyboard function, are printed on the keyboard in combined form as [CLx/A].

It is possible to append an Alpha string onto one whose character entry has already been terminated. This is done using the append key, [⊦] (*append*). (There is no analogous function for digit entry.)

Keystrokes	Display	
[⊦]	HP-41CV_	Notice the _ cue reappears. This signals that the next entry will be appended to the previous one.
/CX	HP-41CV/CX_	
[ALPHA]		Deactivates the Alpha keyboard.

The Alpha Display and the Alpha Register

Entering Alpha strings for any purpose requires the use of the Alpha keyboard. Where the Alpha strings get placed or stored, however, depends on other circumstances.

In the above examples, the Alpha characters being keyed in were entered into the *Alpha register*. Pressing [ALPHA] moves the display from the X-register to the Alpha register and back. (The usual display during numeric calculations is of the X-register.)

*Alpha entries are **not** put into the Alpha register in the following two important conditions:*

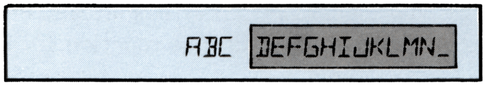
- When they are made in response to specific functions that require Alpha data input for parameter specification. The names of these functions (like [XEQ] *function name*) appear in the display followed by the input cue to indicate that input is needed.
- In Program mode. In this case, Alpha strings are stored in program lines rather than in the Alpha register. (An Alpha string stored in a program line is placed into the Alpha register when the program line is executed.)

Any Alpha entries made in these two conditions *do not affect the Alpha register contents.*

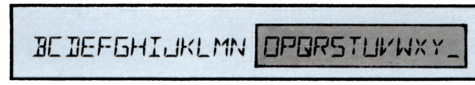
Capacity of the Alpha Register. The Alpha register can hold up to 24 characters, with each period, comma, and colon also counting as one character. However, as you can see, the HP-41 display can only show 12 characters at a time, *not* counting periods, commas, and colons, which fit between the other characters. If character entry is not terminated, the display includes the input cue and up to 11 characters.

Display Scrolling. If you enter more than 11 characters into the Alpha register, the characters already in the display will *scroll* (shift) to the left to display the rightmost characters. Although you can't see them all at once, the Alpha register will hold up to 24 characters. At the entry of the 24th character the computer sounds a tone to warn you that the entry of any more characters will cause a one-by-one loss of characters from the left end of the string.

Alpha Register



Display



Display

Whenever the Alpha register is initially displayed, any string longer than 12 characters is scrolled through the display from left to right. You can move the display to the far right by pressing any key during the scrolling.

Keystrokes	Display	
ALPHA		Activates Alpha keyboard.
SCROLLING E	SCROLLING E	
XAMPLE	ING EXAMPLE_	Characters scroll off the left of the display.
ALPHA ALPHA	SCROLLING EX ROLLING EXAM LING EXAMPLE	Terminates Alpha character entry and redisplays entire string by scrolling.
←		Clears Alpha register.
ALPHA		Deactivates Alpha keyboard.

Continuous Memory

Status

The Continuous Memory of the HP-41 retains the following operating information, even while the computer is turned off:

- All numeric data stored in the computer.
- All programs stored in the computer.
- The current program and program line.
- Display setting (FIX, SCI, ENG).
- Trigonometric mode (Degrees, Radians, or Grads).
- General purpose flag settings.
- Whether the User keyboard has priority; that is, whether it is active (**USER** is on).

When the HP-41 is turned on, it always "wakes up" in Execution mode with either the Normal or User keyboard active.

If the computer is turned off, Continuous Memory will be preserved for a short period while the batteries are removed. *Do not remove the batteries while the HP-41 is turned on, nor turn on the HP-41 while the batteries are out.* Refer to appendix B for instructions on changing batteries.

Clearing Continuous Memory

To entirely clear the HP-41 Continuous Memory and reset the initial conditions:

1. Turn the computer off.
2. While holding the [←] key down, turn the computer back on.

When Continuous Memory has been cleared, the display shows **MEMORY LOST**. Press any key to clear the display.

If the computer is dropped or otherwise experiences a power interruption while it is on, Continuous Memory might be cleared.

Section 2

The Display

Contents

You have already seen in section 1 that the display is a window for different kinds of information—such as numbers for calculations, Alpha characters, and annunciators. The display window can show you information contained in various *registers* (storage locations) in the computer. Sometimes it shows you a message. (A *message* can be anything from an error message to a display of the time.)

The display most often shows the *X-register*, the memory compartment holding the number available for your next calculation (which might be the result of your last calculation). Some other common displays show the Alpha register, the time and date, program lines, catalog listings of things stored in memory, and the input-requesting *parameter functions.*

(Means of clearing the display are covered in section 1, pages 17-18.)

Parameter Functions and the Input Cue: Asking for Input

Parameter functions don't act on *operands* you have already entered; instead, they are followed by *parameters* to complete their function *specification.* Storage register addresses, for example, can be parameters. The display format functions discussed in this section are also parameter functions that use numeric parameters.

When you execute a parameter function, the name of the function appears in the display followed by one or more input cues (_). For example: **FIX _**. You are familiar with the input cue from digit and character entry, where it represents the fact that you can enter another digit or character. With parameter functions, the input cue indicates that you *must* specify a parameter to complete the sequence.

The input cue (_) following a function name in the display means either:

- Numeric parameter needed: the number of input cues matches the number of digits to "fill in."
- Name or label needed: one input cue in the display.

When you complete this key sequence, the function is executed.

You can clear a parameter-function display (also known as a *partial key sequence*) by pressing [←].

Display Control for Numbers

The HP-41 has three options for controlling the display format of numbers.* These options affect the number of decimal places shown and how exponents are shown. *Internally*, however, the computer always represents a number with 10 significant digits of accuracy. The display format is just a convenience for the user: it rounds the *display* of a number to the number of places you specify.†

The three display formats—[FIX] *n*, [SCI] *n*, and [ENG] *n*—would affect the display of the number 123,456 like this (four decimal places specified):

[FIX] 4 :	123,456.0000	
[SCI] 4 :	1.2346	05
[ENG] 4 :	123.46	03

During digit entry, all digits you key in (up to 10) are displayed. The display format takes effect when digit entry is terminated.

The display format setting is maintained by Continuous Memory; at initial turn-on or clearing of Continuous Memory, the display format "defaults" to [FIX] 4, which shows four decimal places and no exponent. (A default condition is one that is established automatically by the computer, and exists unless and until you specify a change.)

* "Numbers" means true numeric values only, and *not* digits used as Alpha characters.

† There is a rounding function ([RND]) that will round the actual number (in accordance with the display format), and not just its display. This function is most useful when a result needs to be rounded for a future calculation, since calculations normally use the full 10-digit number.

Fixed Decimal Place Display (FIX *n*)

FIX (*fixed decimal place*) format is the standard number display format. It shows no exponents, unless the number is too large or too small for the display. The number is shown rounded to the number of decimal places (0 through 9) that you specify—unless the integer portion of the number is too large to accommodate that number of decimal places.

The FIX function (like SCI and ENG) cues you for the one-digit parameter input (the number of decimal places) it needs.

Keystrokes	Display	
123.4567895	123.4567895	Since all 10 places are filled, there is no input cue even though digit entry is not yet terminated.
FIX	FIX _	Asks: How many decimal places?
4	123.4568	Display rounded to four decimal places; digit entry terminated.
FIX 6	123.456790	Display rounded to six decimal places. (Ten digits total are still stored internally.)
FIX 4	123.4568	Usual FIX 4 display.

Scientific Notation Display (SCI *n*)

In SCI (*scientific*) format, a number is displayed with one digit to the left of the decimal point, up to seven digits (which you specify) to the right of the decimal point, and a two-digit exponent. The display is rounded to the specified number of decimal places; however, if you specify more decimal places than the display can hold (SCI 8, or 9), rounding will occur in the undisplayed eighth or ninth decimal place.

Press SCI followed by the number (one digit) of decimal places to display. The display will cue you for the one-digit input.

With the previous number still in the display:

Keystrokes	Display	
SCI	SCI _	Asks: How many places?
6	1.234568 02	Rounds to and shows six decimal places.
SCI 8	1.2345679 02	Rounds to eight decimal places (1.23456790) but displays only seven (1.2345679).

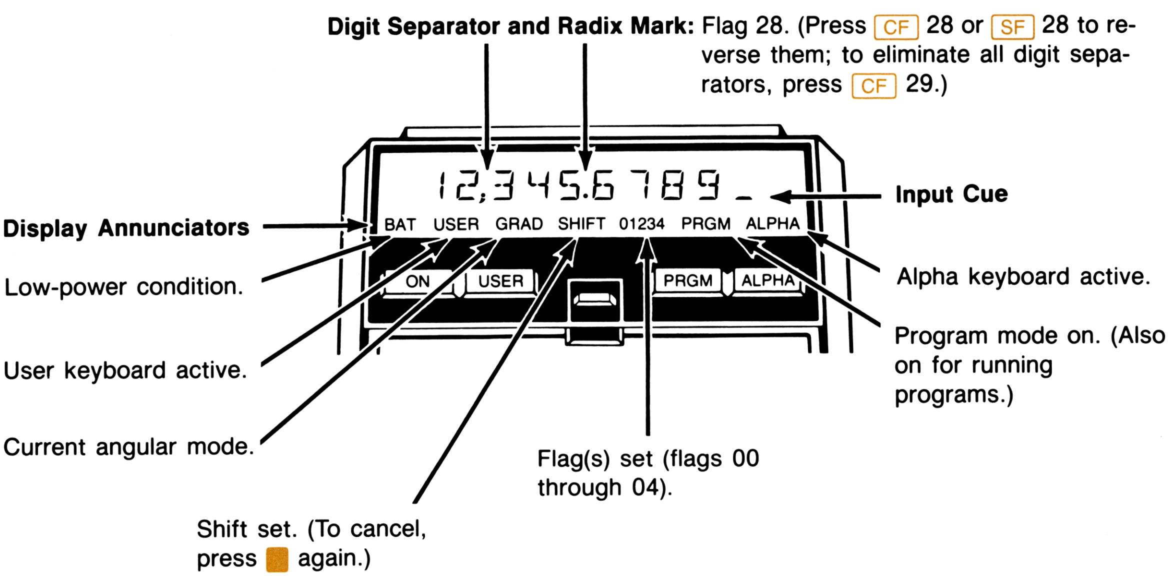
Note: In program instruction lines and in printer listings, numbers with exponents are printed with an **E** before the exponent. For example, **1.234568 02** would look like **1.234568 E02**. If you are trying to copy or enter such a line yourself, just use the [EEX] key, as explained on page 0000 Do not try to put an E into the display.

Engineering Notation Display ([ENG] *n*)

[ENG] (*engineering*) format displays numbers in a manner similar to [SCI], except:

- All exponents are in multiples of three. This is particularly useful for scientific and engineering calculations that use units that are specified in multiples of 10^3 (such as micro-, milli-, and kilo-units).
- The parameter you specify ([ENG] *n*) determines the number of digits besides the first significant digit to which the display will be rounded.

Display Features



Other Display Features

Annunciators

The HP-41 display contains eight annunciators (signals) that indicate the status of the computer for various operations. The meaning and use of these annunciators is discussed on the following pages:

BAT	Low power indication, page 34.
USER	User keyboard, pages 12 and 44.
GRAD and **RAD**	Angular modes, page 51.
SHIFT	Shift for alternate functions, page 13.
0 1 2 3 4	Flags 00, 01, 02, 03, 04. Refer to *HP-41CV Operation in Detail.*
PRGM	Program mode or running program, pages 13 and 62.
ALPHA	Alpha keyboard, pages 13 and 22.

Low Battery Power Indication (BAT)

When the **BAT** annunciator appears in your display, the battery power in the computer is low; you have several days of operating time left. Refer to appendix B for information on replacing the batteries.

Message and Error Displays

Generally, the HP-41 display shows you the numeric or Alpha contents of registers—your data, parameters, results, or Alpha register strings. Sometimes, the display shows you something special—a *message*. Commonly, a message display is an error message (**DATA ERROR**), a status message (**YES**), or a special display like the time. Since such displays are only messages, you cannot operate with or upon them, even if the messages have numbers. Messages do not affect the Alpha register or your calculations.

If you attempt an improper operation—either in a running program or directly from the keyboard—an error message will appear in the display. There is a list of error messages in appendix A.

To remove an error display, press [←]. Pressing any other key will execute that operation (or enter that digit) using the operands already present. (An error-causing operation is simply not executed, and does not affect the operands.)

An error in a program will cause an interruption when you try to execute the program. You can clear the error display by pressing [←] or [PRGM]. The latter places the computer in Program mode, and allows you to see which program instruction caused the error (because this is where the program stopped).

Using Commas and Periods to Separate Digits (Flags 28 and 29)

The HP-41 is set initially and when you reset Continuous Memory (page 27) to separate integral and fractional portions of a number with a period (a decimal point), and to separate groups of three digits in the integral portion with a comma. You can alter these settings to conform to other conventions by altering the status of two *flags*, flag 28—the radix mark flag, and flag 29—the digit grouping flag.

Flags. A flag is a status indicator that is either *set* (=true) or *clear* (=false).

When flag 28 is set (the default condition), the decimal point is the radix mark and the comma is the separator. Numbers appear like this: 1,234,567.01.

When flag 28 is clear, the comma is the radix and the point is the separator mark. Numbers appear like this: 1.234.567,01.

When flag 29 is set (the default condition), a digit separator is used between groups of three digits in the integral portion of a number. Which digit separator mark is used depends on whether flag 28 is set (comma) or clear (period).

When flag 29 is clear, *no* digit separator is used, only a radix mark.

Setting and Clearing Flags. To set or clear a given flag, just press [SF] *nn* (*set flag #nn*) or [CF] *nn* (*clear flag #nn*). [SF] and [CF] are parameter functions, as explained at the beginning of this section, so the display will cue you for the requisite two-digit input (parameter specification).

These flag settings (like most flag settings) are maintained by Continuous Memory.

Keystrokes	**Display**	
1234567.01 [ENTER↑]	1,234,567.010	(This assumes the default setting is in effect: flags 28 and 29 set.)
[CF]	CF _ _	Asks: *Clear which flag?*
28	1.234.567,010	Reverses radix and separator marks.
[CF] 29	1234567,010	Suppresses separator marks between integer digit groups.
[SF] 28 [SF] 29	1,234,567.010	Returns to original setting: flags 28 and 29 set.

The HP-41 has many other flags that control other aspects of computer operation, as well as some flags that you can define yourself. Furthermore, the status of a flag can be *tested*, with the result of that test affecting the conditions of program execution. Flag types and flag use are discussed in *HP-41CV Operation in Detail.*

Section 3

Storing and Recalling Numbers

Contents

The HP-41 stores information—data, Alpha strings, User keyboard function assignments, and programs—in its main memory. This memory is composed of discrete units called *registers*, which can be allocated for these different storage purposes. This section discusses how to store and retrieve numerical data in storage registers.

Storage Registers and Computer Memory

Main Memory Distribution

When you first turn the computer on or when you reset Continuous Memory, there are:

- 273 *data storage* registers (numbered R_{00} to $R_{(272)}$).*
- 46 *uncommitted* registers available in a common pool for programs and User-function assignments.

* Those above R_{99} are only *indirectly accessible*. Indirect access is explained in *HP-41CV Operation in Detail.*

To change the allocation of memory, use the [SIZE] function. When you execute [SIZE], the display will prompt you for *nnn*, a three-digit number (000 to 319) representing the number of registers to allocate to *data storage*. After you enter *nnn*, the allocation will change to *nnn* data registers and (319 − *nnn*) uncommitted registers.

Memory Limitations

If you attempt to store or recall data using a data storage register that *does not exist* (that is, is not part of the currently allocated data storage registers), the **NONEXISTENT** message will appear. You can then either: choose a smaller-numbered register; or reallocate memory to include more data storage registers.

Pressing [←] will remove the **NONEXISTENT** message.

Storage and Recall Operations

When numbers are stored or recalled, they are *copied* (not transferred) between two data storage registers.

To store a copy of a number from the display (X-register) into a directly accessible register:

1. Press [STO] (*store*). The HP-41 will respond with **STO _ _**. The two input cues tell you that the machine requires a two-digit input to execute the function.
2. Enter a two-digit register address (00 through 99). Execution immediately follows.

The newly stored contents will *replace* any prior contents of the addressed register.

To recall a copy of a number from a directly accessible register into the display (X-register):

1. Press [RCL] (*recall*). The **RCL _ _** display cues you to respond with a two-digit register number.
2. Enter a two-digit register address (00 through 99).

The newly recalled contents will replace the prior contents of the display (X-register).

Example: Store Avogadro's number (approximately 6.02×10^{23}) in register 00 (represented in this manual as R_{00}).

Keystrokes	Display		
6.02 [EEX] 23	6.02	23	Avogadro's number.
[STO]	STO _ _		Cues: *In which register?*
00	6.0200	23	Stores a copy in R_{00} and leaves the original in the X-register.
2 [×]	1.2040	24	You can continue calculating with Avogadro's number.
[RCL]	RCL _ _		Recall *from which register?*
00	6.0200	23	Returns a copy of Avogadro's number from R_{00} to the X-register (display).

[STO] and [RCL] are called *parameter functions* because they must be followed by a parameter—in this case, an address. After you press [STO] or [RCL], these operations cue you for the information they need by displaying the function name (**STO** or **RCL**) followed by the number of input cues (_) corresponding to the number of digits required. (This is shown in the example above.)

Clearing Data Storage Registers

The contents of a data storage register remain there—retained by Continuous Memory—until either different contents are stored there or the contents are *cleared*. *Clearing* numeric data means to replace it with zero(s).

- To clear a single storage register, simply store zero in it.
- You can clear all (currently allocated) data storage registers at once by executing [CLRG] (*clear registers*). This does not affect any other registers or parts of memory. ([CLRG] is not on the keyboard. To execute it, see the example below.)

Keystrokes	Display	Description
[XEQ]	XEQ _ _	Execute what?
[ALPHA] CLRG [ALPHA]		Clears all data registers (this is R_{00} to R_{272} if the initial allocation hasn't been changed). Display will show previous result (previous contents of X-register).
[RCL] 00	0.0000	Check contents of R_{00}.

Viewing Register Contents (VIEW)

The VIEW *nn* function shows you a temporary display of any storage register you specify. It acts like a moveable window by giving you a look at the contents of any register; it does not move or recall those contents, so you cannot use the number you are shown for any calculation. Likewise, the number you view does not affect any of the numbers you may have already keyed in or recalled.

- To view a directly accessible register, press VIEW and specify the two-digit register address (from 00 to 99)—as you would with RCL.
- Pressing ← terminates the viewing and returns a display of the current number (the X-register contents).

VIEW is a parameter function, like STO and RCL. Pressing VIEW will display **VIEW _ _**, the two input cues indicating the two-digit input needed.

Example: The following keystrokes illustrate VIEW operation:

Keystrokes	Display	
10 STO 00	10.0000	
√x	3.1623	The square root changes the contents of the X-register.
VIEW	VIEW _ _	Cue: *View which register?*
00	10.0000	Viewing R_{00}.
2 ×	6.3246	Further operations act on 3.1623, not the viewed 10.0000.

Exchanging Register Contents

Exchanging X and Y Contents: x≷y

Pressing x≷y will exchange the locations of the numbers in the X- and Y-registers. The prior Y contents move into X and are displayed; the previous X contents move into Y and are no longer displayed.

Exchanging the contents of the X- and Y-registers is very useful for:

- Obtaining the second result of an operation that calculates two separate values.
- Reversing the order of two operands that are intermediate results from a longer calculation. This is handy for carrying out noncommutative operations when the current operands are in the wrong order, and is a common manipulation in programs.

Exchanging X and Other Register Contents: [X<>]

[X<>] is a nonkeyboard function (see the example below) that exchanges the contents of the X-register and whatever other register you specify. It is a parameter function (like [RCL] and [VIEW]) requiring a two-digit register address for completion.

Keystrokes	Display	
4 [ENTER↑] 5	5_	Enters two numbers sequentially. 5 is in the X-register (displayed).
[x≷y]	4.0000	Exchanges the contents of X and Y. 4 *was* in Y; it is now in X.
[X<>]	X<> __	Exchanges the contents of X and the specified register.
00	10.0000	The number that *was* in R_{00} (from the last example).
[VIEW] 00	4.0000	You can see that R_{00} now holds what used to be in X.
[←]	10.0000	Clears the viewed number and restores the normal X-register display.

Register Arithmetic

Storage Arithmetic

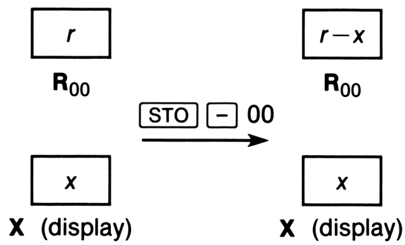
Suppose you not only wanted to store a number, but perform an arithmetic operation with it and store the result in the same register. You can do this directly—without using [RCL]—by doing storage arithmetic, as follows:

1. Have your second operand (besides the one in storage) in the display (X-register). (It can be keyed in, recalled, or the result of a calculation.)
2. Press [STO] {[+], [−], [×], [÷]}.*
3. Key in *nn*, the two-digit register address (00 to 99).

* The braces, { }, indicate that you select one of the enclosed functions.

The new number in the register is determined as follows:

$$\begin{pmatrix}\text{new contents}\\ \text{of register}\end{pmatrix} = \begin{pmatrix}\text{old contents}\\ \text{of register}\end{pmatrix} \left\{\begin{matrix} + \\ - \\ \times \\ \div \end{matrix}\right\} \begin{pmatrix}\text{number in}\\ \text{display (X)}\end{pmatrix}$$



Example: During harvest, Silas Farmer trucks tomatoes to the cannery for three days. On Monday and Tuesday he hauls loads of 25 metric tons, 27 tons, 19 tons, and 23 tons, for which the cannery pays him \$55 per metric ton. On Wednesday the price rises to \$57.50 per ton, and Farmer 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 deducts 3% of the price on Wednesday, what is Farmer's total net income?

A procedure for this problem is:

$$\begin{array}{rcl} & 55\,(25+27+19+23) - 2\%\,[55\,(25+27+19+23)] & \text{Monday and Tuesday} \\ + & 57.5\,(26+28) - 3\%\,[57.5\,(26+28)] & \text{Wednesday} \\ \hline & \text{Total Net Income} & \end{array}$$

Use storage register arithmetic. The keystrokes are shown on the next page.

Keystrokes	Display	
25 [ENTER↑] 27 [+] 19 [+] 23 [+]	94.0000	Total of Monday's and Tuesday's deliveries.
55 [×]	5,170.0000	Earnings from these deliveries.
[STO] 01	5,170.0000	Places earnings in R_{01}.
2 [%]	103.4000	Calculates deductions for Monday's and Tuesday's bad tomatoes.
[STO] [−] 01	103.4000	Deducts this from earnings in R_{01}.
26 [ENTER↑] 28 [+]	54.0000	Wednesday's deliveries.
57.5 [×]	3,105.0000	Earnings on Wednesday.
[STO] [+] 01	3,105.0000	Adds these earnings to R_{01}.
3 [%]	93.1500	Deductions for Wednesday's bad tomatoes.
[STO] [−] 01	93.1500	Subtracts this from earnings stored in R_{01}.
[RCL] 01	8078.4500	Total net income on the tomato deliveries.

Register Overflow and Underflow

If you attempt a register arithmetic operation that would cause the magnitude of the number in any of the storage registers to exceed $9.999999999 \times 10^{99}$ or be less than 1×10^{-99}, the results are out of range for the calculator. The consequences are explained in section 1, "Out-of-Range Results", page 22.

Section 4

How to Execute HP-41 Functions

Contents

You might not realize it, but the HP-41CV has over 100 built-in functions. (With application or extension modules added, there are several hundred possible functions.) Since there are only 35 keys on the keyboard, it's not possible to give each HP-41 function its own key. So, as on larger computers, you can perform HP-41 operations by writing their names in the display. This is called *Alpha execution.*

Furthermore, the HP-41 keyboard is *redefinable.* That is, you can change the definition of almost any given key so that it will execute another function (or program) of your choice.

Alpha Execution

The HP-41 has both "keyboard functions" and "nonkeyboard functions". The keyboard functions, like [−] and [+], can be executed by pressing the given function key. Most of them can also be executed using their Alpha names. Nonkeyboard functions (those you don't see printed on the keyboard) must be executed using their Alpha names, as explained under the next two topics. (You can also use a User-function key, as explained on pages 44-46.)

Alpha Names

Almost all functions in the HP-41—including those that appear on keys— have Alpha names. These are the function names used and recognized by the computer. The display and the printer (if you have one) use these names, just as you must use these names for Alpha execution. When you start programming, you will notice that the recorded program instructions use these names also.

Note: There is a complete list of HP-41 functions and their Alpha names in the Function Index at the back of this manual. For functions that are also printed on the keyboard, note that the Alpha name does not always match what is printed on the key. (Example: [$\sqrt{x}$] versus [SQRT].)

Other conventions for representing functions are printed on the inside of the front cover of this manual.

The Execution ([XEQ]) Procedure

Any function (or program) in the HP-41 can be executed by using the [XEQ] (*execute*) function and spelling out the function's Alpha name.

1. Press [XEQ]. The display shows **XEQ _ _**, telling you the function needs further input.

   Remember to use [XEQ] even in program lines. Otherwise, your input will be treated like an Alpha string, not an Alpha execution.

2. Press [ALPHA] to activate the Alpha keyboard. (The display changes to **XEQ _** as it awaits an Alpha character.)
3. Spell out the Alpha name of the desired function. (This does not use the Alpha register, just the Alpha keyboard and the display.*)
4. Press [ALPHA] to deactivate the Alpha keyboard and complete the sequence. This triggers the execution of the function. (If that function requires a parameter—like numeric input—its function name and input cue(s) appear in the display.)

   Note: If you neglect to press [ALPHA] before spelling out the name of the function or program to execute, the error **NONEXISTENT** can result (because what you are inadvertently telling the computer to execute does not exist). Clear the display and start over!

Example: Find 6! (factorial) using the [FACT] function. Since this function is not on the Normal keyboard, perform an Alpha execution.

Keystrokes	Display	
6	**6_**	
[XEQ]	**XEQ _ _**	
[ALPHA]	**XEQ _**	Activates Alpha keyboard.
FACT	**XEQ FACT_**	Type in function name (Alpha execution form).
[ALPHA]	**720.0000**	Result of 6!

* As explained on page 25, using the Alpha keyboard for parameter specification (that is, in response to an input cue, like **XEQ _**) does not affect the contents of the Alpha register.

Redefining the Keyboard (The User Keyboard)

The HP-41 allows the user to assign function keys on the keyboard to different functions, such as to nonkeyboard functions, to programs, and to functions from attached peripheral devices. This allows you to customize your HP-41 by defining which keys will execute which functions on the User keyboard (**USER** on).

These redefined keys represent your *User functions*, and are recorded for use on the User keyboard. At the same time, the Normal keyboard retains all of the original key definitions. So, even though you can redefine the User keyboard, you still have access to the Normal keyboard at any time! Use the extra keyboard overlays provided in the HP-41 package to label and identify your reassigned functions.

> **Note:** If you are going to run HP-41 application pacs or software solutions, any key redefinitions you make may interfere with the pacs' use of the User keyboard. Therefore, it is a good idea to clear your function reassignments before running HP-41 applications and software material. ("Cancelling User-Function Assignments" is covered after the next topic.)

Making User-Function Assignments ([ASN])

To assign a function (or a program) to a specific key on the User keyboard:

1. Press [ASN] (on the Normal or User keyboard). The display will cue you for input with **ASN _**.
2. Press [ALPHA] to activate the Alpha keyboard.
3. Enter the function name (Alpha name or program name) to be assigned.
4. Press [ALPHA] to deactivate the Alpha keyboard. (The input cue will move over one space to indicate it expects another input, but not another character.)
5. Press the key (or ■ and the key) to which you want the function assigned. The display will momentarily show the new function name, with the keycode of its new key location.*†

---

* A keycode is a two-digit number representing the row-column location of a key. The rows are numbered 1 to 8 from top to bottom; the columns are 1 to 5 from left to right. A minus sign indicates a shifted key.

† If you hold the key down for more than about one-half second in step 5, **NULL** will appear in the display and the key assignment will not be made.

Note: If you neglect to press [ALPHA] before spelling out the name of the function you want to assign, you will find that the display does not respond to your keystrokes (except to [ON] and [←]). Pressing [ASN] deactivates most keys on the keyboard.

User-function (but not program-label) key assignments require extra memory space, as explained in section 3 under "Main Memory Distribution," page 34.

Example: Assign the factorial function, [FACT], to the [Σ+] key (in the upper lefthand corner).

Keystrokes	Display	
[ASN]	**ASN _**	
[ALPHA] FACT [ALPHA]	**ASN FACT _**	Type in function name.
[Σ+]	**ASN FACT 11**	Completes the assignment of [FACT]. Display momentarily shows function name and keycode, then returns to whatever was previously in the display.

Cancelling User-Function Assignments

User-function assignments are maintained by Continuous Memory. To cancel a key redefinition, just follow the assignment procedure above, but omit entering the function name in step 3 (that is, press [ASN] [ALPHA] [ALPHA] and that key).

Executing User Functions

Pressing [USER], which displays the **USER** annunciator, activates the User keyboard*. Initially, the User keyboard consists of the same functions as the Normal keyboard. As you redefine keys, the original, Normal functions are replaced by the new, User ones. Anytime you want the Normal functions again, just press the [USER] toggle switch to restore the Normal keyboard. You always have access to *both* the original function keys *and* a customized set.

* Notice that while the Alpha keyboard is active, the User keyboard is not, even though its annunciator remains on.

Example: The following sequence illustrates User-function execution with the FACT function, which was assigned to the Σ+ key in the example above. Afterwards, the key reassignment for FACT is cleared.

Keystrokes	Display	
USER		Display does not change. **USER** annunciator lit.
23	**23_**	
Σ+	**2.5852 22**	User execution of FACT finds 23!
6 Σ+	**720.0000**	6!
ASN ALPHA ALPHA	**ASN _**	
Σ+	**ASN 11**	Returns Σ+ key assignment to Σ+ function.
USER		Returns to Normal keyboard and previous display.

Viewing Function Assignments

You can view the function name (Alpha name) of a given key by holding it down. If you hold the key down more than about ½ second, the function won't be executed and **NULL** appears in the display. If you're not sure if a particular key has been redefined, or to what, this is a quick way to find out.

Parameter functions, like STO, do not have function previews; they display input cues instead, such as **STO** _ _. To nullify an incomplete parameter function (like **STO** _ _), press ←.

> **Note:** From now on, the execution of nonkeyboard functions will be represented simply as, for example, FACT. How to execute such a function—whether by Alpha execution or as a User function—will be left up to you. You can refer to this section to review the steps for executing functions.

Section 5

The Standard HP-41 Functions

Contents

This section provides brief explanations of most of the HP-41 standard numeric functions. (Several specialized functions are left for section 11, "Numeric Functions," in part II.) Time functions are covered in section 6, functions specific to programming are in section 7, and files are in section 8.

Some functions are given on the Normal keyboard of the HP-41, but many more are not. To execute a nonkeyboard function, use Alpha execution or a User-defined key (as explained in section 4, "How to Execute HP-41 Functions"). The Alpha names for all the functions are given in the Function Index (at the back of this volume).

General Mathematical Functions

One-Number Functions

The following one-number math operations are all performed in the same way: put your number in the display (the X-register) and then execute the function. You will see the result in the display.

One-Number Functions

To Calculate	Press	Or Execute
Reciprocal	1/x	1/X
Square root	√x	SQRT
Square	x^2	X↑2
Common logarithm	LOG	LOG
Common exponential	10^x	10↑X
Natural logarithm	LN	LN
Natural exponential	e^x	E↑X
Factorial		FACT

For Example:

To Calculate	Keystrokes	Display
1/6	6 1/x	0.1667
$\sqrt{196}$	196 √x	14.0000
12.3^2	12.3 x^2	151.2900
log 0.1417	.1417 LOG	–0.8486
$10^{0.45}$	.45 10^x	2.8184
ln 63	63 LN	4.1431
$e^{1.5}$	1.5 e^x	4.4817
6!	6 FACT	720.0000

Two-Number Functions

For functions that use two operands, remember to use reverse entry: enter both operands first (in the same order you use for calculations on paper), then execute the function. (See "Doing Simple Calculations," page 14.)

Two-Number Functions

To Calculate	Press	Or Execute
Subtraction	–	–
Addition	+	+
Multiplication	×	*
Division	÷	/
Percentage	%	%
Percent Change		%CH
Powers	y^x	Y↑X

Arithmetic (−, +, ×, ÷). Arithmetic calculations are discussed in detail in section 1, under the topics of "Doing Simple Calculations" and "Doing Longer Calculations," pages 16 and 20.

Note also that the keystrokes CHS + find the same result as −, and the keystrokes 1/x × the same as ÷. This is very handy for longer calculations when an intermediate result must then be subtracted from or divided into the next number (refer to page 22).

Percentage (%). Key in the base number before the specified percentage.

Using % differs from using 0.01 × in that the % function preserves the value of the base number, so you can carry out subsequent calculations using the base number and the result without re-entering the base number.

Percent Change (%CH). Key in the base number, y (usually the number that occurs first in time), then the second number, x.

%CH is calculated as $\frac{(x - y)}{y}(100)$. **DATA ERROR** results if $y = 0$.

The Power Function (y^x). Key in the base number, y, before the exponent, x, to calculate y raised to the x power.

For $y > 0$, x can be any rational number. For $y < 0$, x must be an integer.

For Example:

To Calculate	Keystrokes	Result
13% of $8.30	8.3 ENTER↑ 13 %	1.0790
Add 13% of $8.30 to $8.30	8.3 ENTER↑ 13 % +	9.3790
% Change from 156 to 167(positive result)	156 ENTER↑ 167 %CH	7.0513 %
$2^{-1.4}$	2 ENTER↑ 1.4 CHS y^x	0.3789
$(-1.4)^3$	1.4 CHS ENTER↑ 3 y^x	-2.7440
$\sqrt[3]{2}$ or $2^{1/3}$	2 ENTER↑ 3 1/x y^x	1.2599

Trigonometric Operations

Angular Modes

The trigonometric functions operate in the current angular mode: decimal Degrees (*not* degrees-minutes-seconds), Radians, or Grads.

Unless you specify otherwise, the HP-41 assumes that any angles (input and output) are in decimal degrees. The angular mode is maintained by Continuous Memory.

- [DEG] sets Degrees mode. The format for degrees uses decimal fractions of degrees, not minutes and seconds. When Degrees mode is set, there is no accompanying annunciator (neither **RAD** nor **GRAD** is on).
- [RAD] sets Radians mode. The **RAD** annunciator turns on.
- [GRAD] sets Grads mode. The **GRAD** annunciator turns on.

Specifying an angular mode will *not convert* any number already in the computer; it merely tells the computer what unit of measure to assume for numbers that are used for trigonometric functions. 360 degrees = 2π radians = 400 grads.

Using Degrees in Degrees-Minutes-Seconds Format

Converting Between Forms for Hours and Degrees. Values for time (in hours) or angles (in degrees) can be converted between a decimal fraction form and a minutes-seconds form using the one-number functions [HR] (*to decimal hours*) and [HMS] (*to hours-minutes-seconds*).

Converting Time Values

Hours.Decimal hours (*H.h*)	← [HR] — [HMS] →	Hours.Minutes Seconds Decimal seconds (*H.MMSSs*)
Degrees.Decimal degrees (*D.d*)	← [HR] — [HMS] →	Degrees.Minutes Seconds Decimal seconds (*D.MMSSs*)

For example, with the operand in the display, execute [HMS] to convert

$$\underbrace{1.2345}_{\text{hours or degrees}} \quad \text{to} \quad \underbrace{1}_{\text{hours or degrees}}.\underbrace{14}_{\text{minutes}}\underbrace{04}_{\text{seconds}}$$

Executing HR would change 1.1404 (that is, 1:14:04 or 1°14′04″) back to 1.2345.

Adding and Subtracting Time Values. To add or subtract a time (or angle) in hours-minutes-seconds form, use the HMS+ (*hours-minutes-seconds, add*) function or the HMS− (*hours-minutes-seconds, subtract*) function.

Enter both times or angles, then execute HMS+ or HMS−.

Example: Find the difference between 20 hours, 16 minutes, 56.55 seconds and 11 hours, 23 minutes, 07.12 seconds. (Press FIX 6 to display all 6 decimal places.)

Keystrokes	Display	
FIX 6		Display shows previous result.
20.165655	**20.165655**	
11.230712 HMS−	**8.534943**	Result.

Converting Between Degrees and Radians (D−R, R−D)

The D−R (*degrees to radians*) function converts the number in the display from decimal degrees into radians. R−D (*radians to degrees*) does the reverse: it converts the number in the display from radians into decimal degrees.

Trigonometric Functions

The trigonometric functions are all one-number functions, so they calculate results using the number in the display (X-register). Before executing a trigonometric function, make sure that the desired angular mode is set: Degrees, Radians, or Grads.

Trigonometric Functions

To Calculate	Press	Or Execute
Sine of x	SIN	SIN
Cosine of x	COS	COS
Tangent of x	TAN	TAN
Arc sine of x	SIN^{-1}	ASIN
Arc cosine of x	COS^{-1}	ACOS
Arc tangent of x	TAN^{-1}	ATAN

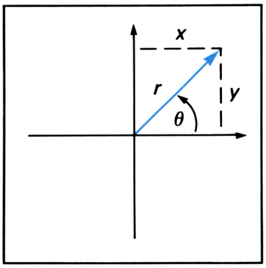
Example: Show that the cosine of $(5/7)\pi$ radians and cos 128.57° are the same.

Keystrokes	Display	
RAD		Sets Radians mode; **RAD** annunciator lit.
5 ENTER↑ 7 ÷	0.7143	5/7.
π ×	2.2440	$(5/7)\pi$.
COS	-0.6235	Cos $(5/7)\pi$.
DEG	-0.6235	Sets Degrees mode; no annunciator.
128.57 COS	-0.6235	Cos 128.57° = cos $(5/7)\pi$.

Converting Between Rectangular and Polar Coordinates (R→P, P→R)

The functions converting between rectangular and polar coordinates are the only two-number trigonometric functions. They are R→P (*rectangular to polar*) and P→R (*polar to rectangular*).

The rectangular coordinates (x, y) and the polar coordinates (r, θ) are measured as shown in the illustration at right. The angle θ is in the units set by the current angular mode.



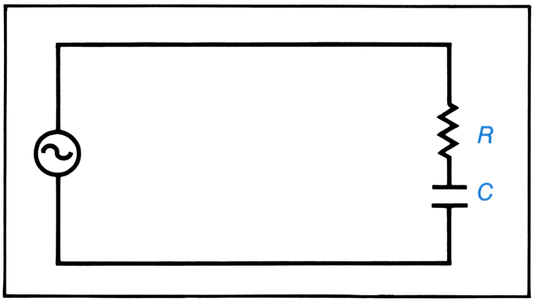
To Convert	Key In	Press	Result
(x,y) to (r,θ)	y-value	ENTER↑	
	x-value	R→P	r-value
		x≷y	θ-value
(r,θ) to (x,y)	θ-value	ENTER↑	
	r-value	P→R	x-value
		x≷y	y-value

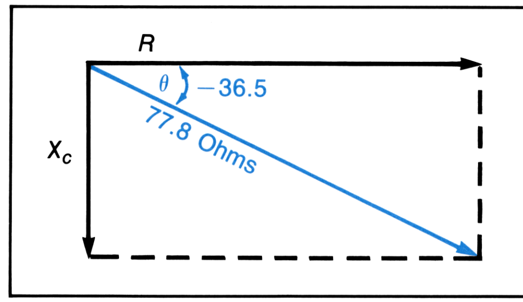
Both R→P and P→R require two numeric inputs and return you two outputs. For R→P, be sure to enter y before x. The resulting display will show r; press x≷y to see θ.

For P→R, enter θ before r. The resulting display will show x; press x≷y to see y.

Example: Engineer P.C. Bord has determined that in the RC circuit shown below left, the total impedance is 77.8 ohms and voltage lags current by 36.5°. What are the value of resistance, R, and capacitive reactance, X_c, in the circuit?

Use a vector diagram as shown below right, with impedance equal to the polar magnitude, r, and voltage lag equal to the angle, θ (in decimal degrees). When the values are converted to rectangular coordinates, the x-value yields R (in ohms), while the y-value yields X_c (in ohms).





Keystrokes	Display	
DEG		Sets Degrees angular mode. Display remains from previous results.
36.5 CHS ENTER↑	-36.5000	θ, degrees voltage lag.
77.8	77.8_	r, ohms total impedance.
P→R	62.5401	Result: x, ohms resistance, R.
x≷y	-46.2772	Result: y, ohms reactance, X_c.

Statistical Functions

Accumulating Data Points (Σ+)

The HP-41 can perform one- or two-variable statistical functions on one- or two-variable data points. The data points are automatically accumulated for the computer by the Σ+ (*summation plus*) function. With a data pair entered into the X- and Y-registers, Σ+ will automatically calculate and store the various sums and products necessary for statistical calculations. Specific storage registers, called *statistics registers*, are used for these accumulations. The statistics registers are R_{11} through R_{16} (unless you change their location, as discussed below).

Before beginning to accumulate a new set of data, execute CLΣ (*clear statistics registers*) to clear the statistics registers.

For two-variable statistical calculations, enter a data pair (x- and y-values), *y-value first*. For one-variable statistics calculations, for your first entry use zero for y, then enter x. You only need to do this once; then just enter x-values.

1. Key in y; press [ENTER↑].
2. Key in x.
3. Press [Σ+].

   The display will show the current number of accumulated data points, n. The x-value is saved in the LAST X register and y remains in the Y-register.

The accumulated data points are compiled as follows:

The Statistics Registers

Register	Contents	
R_{11}	Σx	Summation of x-values.
R_{12}	Σx^2	Summation of squares of x-values.
R_{13}	Σy	Summation of y-values.
R_{14}	Σy^2	Summation of squares of y-values.
R_{15}	Σxy	Summation of products of x- and y-values.
R_{16}	n	Number of data points accumulated. (Displayed.)

You can recall any of these summations with [RCL] *nn*, where *nn* is the register address. Or, you can simply view any of these contents with [VIEW] *nn*, which does not change any register's contents.

Changing the Statistics Registers. You can specify your own block of six statistics registers with the [ΣREG] function. Execute [ΣREG] *nn*. The display **ΣREG _ _** cues you for the two-digit address of the first of six consecutive registers. This register assignment is maintained by Continuous Memory.

Locating the Statistics Registers. If you have not changed the location of the statistics registers, they start at R_{11}. If you have changed the location (or don't remember), you can recall their location by executing [ΣREG?]. The display will return the address of the first register in the six-register block.

Register Overflow. Unlike the effect of other functions, the execution of [Σ+] will not ever cause an overflow error. If the execution of [Σ+] causes the contents of any of the statistics registers to exceed $\pm 9.999999999 \times 10^{99}$, the calculation is completed and $\pm 9.999999999 \times 10^{99}$ is placed in the register(s) that overflowed.

However, the execution of other statistical calculations, like [SDEV], *can* cause an overflow error (**OUT OF RANGE**). If you are doing two-variable calculations, very large or very small values for x *or* y can make it impossible to find the mean or standard deviation for *both* x and y.

Correcting Data Entry (Σ−)

Digit entry can be corrected as usual with ◆ and CLx *before* Σ+ has been pressed. If you discover that you have entered and accumulated incorrect data points, you can delete the data point(s) and correct the summations by using Σ− (*summation minus*). Even if only one value of an (x, y) data pair is incorrect, you must delete and re-enter *both* values.

If the incorrect data point or pair is the most recent one entered and Σ+ has been pressed, you can execute LASTx Σ− to remove the incorrect data. Otherwise:

1. Enter the *incorrect* data pair into the X- and Y-registers. (Remember to use zero for y if you are using only one variable, x.)
2. Press Σ−.
3. Enter the correct values for x and y.
4. Press Σ+.

Example: Below is a chart of maximum and minimum monthly winter (October–March) rainfall values from a 79-year period in Corvallis, Oregon. Enter and accumulate the data values. Remember to use Σ− to correct any incorrect data entries you make.

	October	November	December	January	February	March
X **MINIMUM, *x*** (inches rain)	0.10	0.22	2.33	1.99	0.12	0.43
Y **MAXIMUM, *y*** (inches rain)	9.70	18.28	14.47	15.51	15.23	11.70

Keystrokes	Display	
CLΣ		Clears the statistics registers. (Display shows previous result.)
9.7 ENTER↑	9.7000	Enters y first (max. rainfall).
.10 Σ+	1.0000	Number of data pairs is now one.
18.28 ENTER↑	18.2800	
.22 Σ+	2.0000	Number of data pairs is two.
14.47 ENTER↑	14.4700	
2.33 Σ+	3.0000	
15.51 ENTER↑	15.5100	
1.99 Σ+	4.0000	
15.33 ENTER↑	15.3300	(Incorrect data entry.)
.12 Σ+	5.0000	

Keystrokes	Display	
11.70 ENTER↑	11.7000	
.43 Σ+	6.0000	$n = 6$.
15.33 ENTER↑ .12 Σ−	5.0000	Deletes incorrect data pair; n decremented.
15.23 ENTER↑ .12 Σ+	6.0000	Adds correct data pair; n incremented.
RCL 12	9.6467	Returns the value for Σx^2.

Mean (MEAN)

The MEAN function computes the arithmetic mean (average) of the x- and y-values that have been stored and accumulated using Σ+. The mean of x ($\bar{x}$) is placed in the display (X-register), and the mean of y ($\bar{y}$) is simulataneously placed in the Y-register. Press x≷y to bring $\bar{y}$ into the display.

Example: From the corrected statistics data entered and accumulated above, calculate the average monthly rainfall minimum, $\bar{x}$, and average monthly rainfall maximum, $\bar{y}$.

Keystrokes	Display	
MEAN	0.8650	Average minimum inches of rain per month, $\bar{x}$.
x≷y	14.1483	Average maximum inches of rain per month, $\bar{y}$.

Standard Deviation (SDEV)

The SDEV (*standard deviation*) computes the sample standard deviations, s_x and s_y, of the data accumulated using Σ+. The sample standard deviation gives an estimate of the population standard deviation from the sample data.*

Executing SDEV places the value for s_x in the display (X-register), and simultaneously places the value for s_y in the Y-register. Press x≷y to bring s_y into the display.

Example: Calculate the standard deviations about the means calculated above.

Keystrokes	Display	
SDEV	1.0156	Standard deviation about mean of minimum rainfall per month, s_x.
x≷y	3.0325	Standard deviation about mean of maximum rainfall per month, s_y.

* If your data does not form just a sample of a population but *all* of it, you can easily find the true population standard deviation by adding the mean to the accumulated data before using SDEV.

Vector Arithmetic

The statistics accumulation functions can be used to perform vector addition and subtraction. The input needs to be in rectangular coordinates, so convert polar vector coordinates to rectangular vector coordinates first. Use the following sequence for *each* vector.

For Rectangular Coordinates:

1. Enter y, press [ENTER↑].
2. Enter x.
3. [Σ+] **or** [Σ−] (for addition **or** subtraction).
4. [RCL] 11 for the resulting x-coordinate.*
5. [RCL] 13 for the resulting y-coordinate.*

For Polar Coordinates:

1. Enter θ, press [ENTER↑].
2. Enter r.
3. [P→R].
4. [Σ+] **or** [Σ−].
5. [RCL] 11 for x-coordinate.*
6. [RCL] 13 for y-coordinate.*
7. [R→P] if polar coordinates are desired. This displays the polar x-coordinate; press [x≷y] to view the polar y-coordinate.

A programmed example of vector addition is given on page 87 in section 6.

Defining Your Own Functions

In section 7, "Elementary Programming," you'll see how to write and customize your own functions—using vector addition as an example—by storing them as routines in program memory. Programs can be assigned to User-defined keys, and then executed in one keystroke just like any other function.

* This assumes that the statistics registers are still assigned to R_{11} through R_{16}. If so, R_{11} holds Σx and R_{13} holds Σy.

Section 6

Elementary Programming

Contents

The HP-41 operations covered in sections 1 through 5 are very useful for doing many kinds of calculations. Using these operations *manually* is only one side of your HP-41. The other side is programming. Just as the ability to redefine keys lets you design your own keyboard, the ability to write programs lets you design your own operations.

Note: If you are interested mainly in running application pacs and prewritten programs, and not in writing your own programs, consider especially the following topics in this section:

1. "Entering a Program into Main Memory," page 67, if you have only a listing of a program and need to enter it yourself. (If you have an application module, bar code, or magnetic cards, you do not need to key in the program yourself. Refer to the owner's manuals for those products.)
2. "Executing a Program," page 68.
3. "Copying a Program from an Application Module," page 85, if you want to alter an application program, or if you don't want to keep the module plugged in, but still want to use a program from it.

While you write a program, you are actually storing program instructions into program memory. This is done in *Program mode*. The computer does not react to or act upon operations as you store them during Program mode. Running—or *executing*—a program is done in *Execution mode*. This is the only time the computer performs the operations that were stored, and it carries out the operations just as if you were keying them in manually.

As mentioned before, an understanding of the automatic memory stack helps greatly in being able to write efficient and powerful programs. However, the discussion in this section does not presume knowledge of the stack. There is a more in-depth treatment of HP-41 programming in *HP-41CV Operation in Detail.*

What Programs Can Do

The very simplest program is just a recording of the keystrokes necessary to perform a series of operations, as, for instance, a series of calculations.* It is very similar to what you would do manually (that is, from the keyboard) to solve an equation, with one great difference: it can repeat the calculations over and over, and all you do is enter any new, variable numbers. What you can do with the computer *manually*, a program can do *automatically*.

* To be more exact, these keystrokes are recorded within the uncommitted registers of main memory. The HP-41 memory is summarized in section 3 and covered in detail in section 12.

For instance, even if you program something as straightforward as the quadratic formula, you will notice two immediate benefits:

- You save the time of repeating all the keystrokes every time you want to use the formula.
- You do not have to look up the formula each time you need to use it!

This means, in effect, that you can define your own functions (like a cubing function, for example) for the HP-41, just by recording the steps for it in program memory.

Furthermore, there are two other convenient capabilities of a program:

- It can make a decision based on a certain condition. For instance, the program could proceed differently depending upon whether the quadratic equation found real or complex roots.
- It can repeat an operation by "looping" through it more than once.

There are special considerations you have to make while composing a program, however. Since a program runs automatically, without your controlling it, it has to be designed to acquire and use data values (input) at the right time, store and recall intermediate calculations as necessary, and give you the results (output) in a manner that you will understand later (when you might not remember the program as well).

After covering the mechanics of program structure, we will return to programming the quadratic formula as an example for exploring elementary aspects of programming techniques.

Program Lines and Program Memory

When the HP-41 is in Program mode (**PRGM** annunciator lit, with a line number on the left of the display), the keystroke sequences that you enter are *stored* as operations in program memory. (This is called "keystroke programming.") Each instruction, such as [+] or [STO] 01, occupies a *program line*, which is automatically numbered.

These program lines automatically are allocated space in program memory, which draws from the uncommitted registers in main memory. Since program memory allocation is automatic, you do not need to be concerned with it unless the memory space available among the uncommitted registers is insufficient.

Memory Limitations

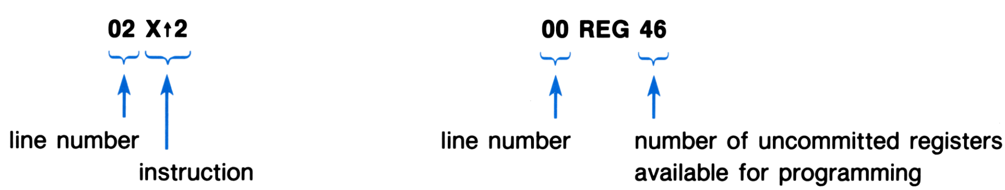
One *register* of memory can provide up to seven *lines* of program instruction. If there is not enough uncommitted register space available to store an instruction you are adding in Program mode, the HP-41 will "pack" its memory and then display **TRY AGAIN**. (*Packing* memory means to arrange the program instructions so as to close any unused gaps in program memory.) You should try keying in the program instruction again; if the message **TRY AGAIN** appears again, this means that you have run up against a limitation in available memory. Before you can enter any more program instructions, you need to make more registers available for program storage. You do this by executing the [SIZE] function as described on page 35. (You can also make registers available by clearing other programs, or clearing User-function assignments.) For more information, refer to section 4 in *HP-41CV Operation in Detail.*

Memory Structure

You do not need to be familiar with the HP-41 memory structure to start programming, so it is not discussed here. In *HP-41CV Operation in Detail*, there is a complete description of the structure of the HP-41 memory, how program memory fits into the larger picture of memory, and how to allocate more (or less) space to program memory. This includes a diagram of the computer memory. In this section, it is assumed that the allocation of memory has not been changed from the original 273 registers for data storage, 46 registers for all else.

The Pointer in Program Memory

When you press [PRGM] and switch to Program mode, the display will show you a two- (or three-) digit line number (on the left) and the *current program line*. If your program memory is empty, you will see **00 REG 46**. The displayed line—the current line in the current program—is determined by the *program pointer*.



The current program will generally be the one last worked on or run. (There are some functions, such as the catalog listing, that change the location of the program pointer and, hence, change which program is current.)

The Basic Parts of an HP-41 Program

The bare skeleton of the simplest HP-41 program looks something like this:

Display		Printer Listing
01 LBL^{T}AREA	← Global Label →	01 LBL^{T}AREA
02 X↑2 03 PI 04 *	Body (calculates πx^2).	02 X↑2 03 PI 04 *
05 END	← Program END →	05 END

The body of this program will square whatever is in the X-register and multiply it by π. If the value given in X is a radius, then this routine will calculate the area of a circle, πr^2.

> **Note:** The display (and printer listing) of program lines makes a distinction between Alpha character strings you create (like **AREA**) and Alpha characters that spell out the name of an HP-41-defined function (like **PI** or **END**). Notice that a character string is preceded by a "raised T" (^{T}) in the display: ^{T}**AREA**. The printer encloses strings in quotation marks, and precedes labels with a diamond. *If you forget to press* [XEQ] *before spelling out the name of a function you want executed, the function name will appear as a character string, not as a function.* (Refer to the explanation of Alpha execution in section 4, page 43.)

The program name—that is, its *global label*—and the program **END** are extremely important. They supply the identity and the boundary for a program. *An HP-41 program generally runs from a global label to an* **END** *statement.* You should make sure to put a global label at the beginning of a program, so it has a name by which you can call it. There is always an automatic, permanent **END** statement (displayed as **.END.**) at the very end of program memory.

To enter the above program into program memory, you can use the following keystrokes. The operations are explained in the following text.

Keystrokes	Display	
[PRGM]	**00 REG 46**	Program mode; **PRGM** lit. (Display assumes no other programs present.)
[GTO] [.] [.]	**00 REG 46**	This positions the computer to the end of program memory, packs program memory, *and* inserts an **END** statement (if needed) between the last program and the one to come. (This step is not necessary here.)

Keystrokes	Display	
[LBL]	01 LBL _ _	
[ALPHA] AREA [ALPHA]	01 LBL^{T}AREA	The global label heads the program. Remember to use the Alpha keyboard to enter Alpha characters.
[x^2]	02 X↑2	All functions appear as their Alpha names.
[π]	03 PI	
[×]	04 *	
[GTO] [·] [·]	00 REG 44	This line is optional. It automatically adds an **END** statement, and then displays the number of registers still available for programming.
[PRGM]		Returns to Execution mode.

Program Mode

To enter, modify, delete, or view program lines, the computer must be set to Program mode. Pressing [PRGM] once will activate Program mode, turn the **PRGM** annunciator on, and display the current line in program memory.

Pressing [PRGM] again will deactivate Program mode and return to Execution mode. You will see that the **PRGM** annunciator lights up during one other special circumstance: when a program is being executed (run). This is as a reminder to you that a program is running, and does not signify Program mode.

Labels

A label is an identifier placed at the head of a series of program steps. The entire program generally starts with a *global label* (like **AREA** in the above example). Within a larger program, there may be smaller "routines" that are identified by *local labels*. There are important differences between global and local labels, but the general purposes of a label are to:

- Mark the beginning of a program (global label) or program segment (local label).
- Provide access to a program (global label) or program segment (local label).

Be sure to include a global label in a program. This allows you to access the program easily. Without a global label, it is tricky to run, modify, or delete a given program, since you cannot refer to the specific program you want. (To gain access to a program without a global label, use catalog 1, as explained under "Modifying Programs in Main Memory.")

Global Labels. A global label is defined as a label consisting of up to seven Alpha characters (including Alpha digits). The keystroke sequence is LBL followed by up to any seven characters *except* { . , : ⊢ }. Also, you cannot use the single letters A through J or a through e *alone*. A global label is special in the following ways:

- You can access a global label (and, subsequently, all its program lines—that is, all lines up to the **END** statement) *from anywhere in program memory.*
- A list of global labels (along with their **END** statements) is kept in catalog 1, which provides you with a program directory. (Catalog 1 is explained on page 76.) If a program has no global label, there is no name for that program in the directory.
- Global labels (and not local labels) can be assigned to keys on the User keyboard. This lets you execute a program with one keystroke, instead of having to type out the program's global label.

Obviously, for a global label to be effective it must be unique: you want only one program with any given global label.

Local Labels. Local labels are of two types: *numeric* and *Alpha.* (Global labels are always Alpha.) The keystroke sequence is LBL followed by {00 . . . 99} or {A . . . J} or {a . . . e}.

- (Local) numeric labels have two digits only, 00 through 99. (00 through 14 are called "short form" because they use less memory.)
- Local Alpha labels use the single Alpha character A through J or a through e. (These are *not* considered global labels.)

Local labels are used *within a program* to mark and provide access to various segments or *routines* within the same program. Labels within programs mainly are useful for program *branching*, which serves to modify how a program is executed.

- Local labels can only be accessed if the one you want is within the current program (the one currently pointed at). References to local labels cannot cross program boundaries (**END** statements).
- Local labels are not listed in catalog 1 and cannot be assigned to the User keyboard. This is because local labels are for strictly "local" use; they can only be used within the context of a single program.

Local labels do not need to be unique within program memory, but they should be unique within a program. Since a local label can be accessed only from within a given program, it will never be confused with a like-named local label in another program.

Program ENDs and GTO · ·

As mentioned before, the END instruction separates one program from another. There is always at least one **END** in program memory: the "permanent **END**," which appears in the display as **.END.** So the last program in memory always has an automatic **END**, even if you neglect to include an END instruction in it.

After the first program in memory, you should insert an **END** between subsequent programs so they will be considered as separate programs, and not just labeled routines within the same program. You can accomplish this by entering an END instruction at the end of your programs. There is another way to accomplish the same thing:

After entering a program, or before entering a new program, press GTO · ·. This does several things:

- It packs program memory, shifting the contents of memory to use up any intervening memory space left among the program instructions. (The display flashes the message **PACKING**.)
- It automatically inserts an END instruction at the end of the previous program (if an **END** does not already exist).
- It sets the current position of the program pointer to line 00 in a new program and shows you the number of registers remaining available for programming.

Entering a Program into Main Memory

If you followed the keystroke sequence listed on page 86-87 for entering the program called AREA, then you have entered and stored a program for calculating the area of a circle. When a program is entered, it is stored in main memory and will be saved until you either delete it line by line, clear it entirely, or clear Continuous Memory. Program memory is preserved even when the computer is off.

The general steps for keying in and storing an HP-41 program are:

1. Activate Program mode. (Press PRGM; **PRGM** displayed.)
2. Press GTO · ·. (Explained above.)
3. Enter a global label of up to seven Alpha characters.
4. Enter each subsequent instruction.

   If the instruction uses a nonkeyboard function, remember to precede the function name with XEQ—otherwise your input will appear as an Alpha string only, and not be executed. Or, you can use User-defined keys (with the User keyboard active).
5. Press GTO · ·. This step is optional; it adds an END statement to the program.
6. Return to Execution mode. (Press PRGM.)

If you make any mistakes, use the ← key to delete individual characters and entire lines.

When entering instructions in Program mode, you can use the Normal, Alpha, and User keyboards, just as you can in Execution mode. (User-defined function keys are very convenient in programming.) However, certain specific functions cannot be included in a program. These *nonprogrammable functions* are listed in *HP-41CV Operation in Detail* under "Nonprogrammable Operations."

Executing a Program

Regular Execution

To run or *execute* a program, the computer must be in Execution mode (no **PRGM** annunciator on). Once the program has started running, however, the **PRGM** annunciator goes on automatically, and the program-execution indicator (⊱) appears.

The general steps for executing an HP-41 program are:

1. Make sure Execution mode is set (not Program mode).
2. Key in or store any data that the program needs before it starts. If there is only one number, you don't need to press [ENTER↑], because starting the program accomplishes the same thing. (There are ways to put in data at different times in the course of a program, discussed under "How to Enter Data," page 70.)
3. Execute the program by name (global label) just as you execute nonkeyboard functions: either (a) using the Alpha name, or (b) using a redefined key on the User keyboard. (Refer to section 4.)
   a. Press [XEQ], then spell out the global label of the program you want to execute ([ALPHA] *label* [ALPHA]).

   or

   b. Press a User-defined key to which you have assigned the program's global label.

To run the same program over again, press [RTN] [R/S] (*return, run/stop*). This returns the program pointer to the beginning of the program, and then starts execution from there.

While a program is running (⊱ displayed), no keys on the keyboard are active (that is, pressing them has no effect) except [R/S] and [ON].

> **Note:** Do not stop a running program, do a calculation, then restart the program. Operations that you perform might interfere with the calculations being carried out by the interrupted program.

> **Note:** The same convention that is used to represent nonkeyboard functions ([DATE]) is used to represent program execution ([AREA]), since they are the same to the computer and are executed in the same ways.

Example: To execute program AREA (assuming you have stored it, as shown on page 65), use the following key sequence. Find the areas of circles of radius 1.6, $\pi\sqrt{2}$, and 32×10^{-6}. This program requires only one piece of data input, which is supplied before the program starts.

Keystrokes	Display	
		Make sure Program mode is *not* set.
1.6	1.6_	Key in the input value.
[AREA]	8.0425	Execute the program like you would a nonkeyboard function (see above and section 4).
2 [√x] [π] [×]	4.4429	Radius.
[AREA] (or [RTN] [R/S])	62.0126	Area. (To execute the same program again, you can simply press [RTN] [R/S].)
32 [EEX] 6 [CHS]	32 −6_	Radius.
[AREA] (or [RTN] [R/S])	3.2170 −09	Area.

Stepwise Program Execution—Debugging a Program

If you know there is an error in a program you have stored, but are not sure where the error (or errors) is, then a good way to "debug" the program (find and correct the "bugs") is by stepwise execution. To follow the execution of a program *line by line*, use [SST] (*single step*) in Execution mode. This shows you the result after each program line is executed, letting you see exactly where something specific (perhaps unexpected) happens. (For editing programs, refer also to "Modifying Programs in Main Memory," page 76.)

To use [SST] for stepwise execution:

1. Set the computer to Execution mode (**PRGM** *not* on). If any data is needed at the start of the program, enter it.
2. (Optional.) Press [GTO] *label* to set the program pointer to the label at which you want to start execution. (Otherwise, execution will start at the current program line.)
3. Press [SST]. As you hold [SST] down, you will see the current program instruction displayed. (If you hold the key too long, **NULL** appears and the [SST] function is not executed.)

   When you release [SST], the current program line is executed. The pointer then moves to the next line.

When the program pointer reaches the end of the current program, it "wraps around" to the first line of the program.

If the computer is in *Program* mode, [SST] does not effect line-by-line *execution*, but instead only line-by-line *viewing* of the program. [BST] (*back step*) moves the program pointer backwards one line; no execution ever takes place, regardless of mode.

Program Data Input and Output

How to Enter Data

A program needs to provide for data entry. Data values will vary each time a program is run, so such *variables* do not get written into a program; they must be supplied each time the program is executed. Data input can be given just before running a program, or else during an interruption in the program. These two methods are *prior entry* and *entry during program interruption.*

Prior Entry. If a variable will be used in the first calculation the program makes, you can enter it (into the X-register) before executing the program.

Entry During Program Interruption. You can include *within* the program an instruction to make it stop at a certain point where you know that data input is needed. A programmed stop instruction ([R/S] or [STOP]) will do this, as will the [PROMPT] function (covered in the next topic). In other words, the program includes instruction(s) to interrupt itself; it resumes execution when you press [R/S] from the keyboard. The user needs to know what kind of data value is needed when the program halts, which is easy to do with an Alpha message ("Using Messages in Programs," page 72).

If more than one data value is needed, they can be entered either as they become needed, or all at once at the beginning of the program, from whence they are stored into storage registers until they are needed. There is an example of this in program QUAD on page 83.

While a program is stopped, all keys on the keyboard are again active, so you can put in new data that will then be used by the program. It is possible, however, that doing calculations will interfere with numbers the program will use later for calculations.

Note that when a program is interrupted, the **PRGM** annunciator is not on. An interrupted program is the same as no program running.

Viewing and Recording Data Output ([VIEW], [PSE], [R/S])

Assuming you do not have a printer for your HP-41, the way to view an intermediate calculation or result in your program is to have the program stop, pause, or display a specific register. (While a program is running, only the ⊁ is displayed.)

If a program returns only one result, and it is the last quantity calculated (as in the program example AREA), you do not need to make the program interrupt itself or display the X-register because when it finishes, it stops, and the display shows the final result of the program.

If, on the other hand, a program calculates more than one result, you need to have the program display the result by interrupting the program so that the X-register (or another specified register) will display its current contents.

View (VIEW *nn*). If you want a display of an intermediate result *while the program is running*, use VIEW *nn*. When a program executes VIEW *nn*, it displays what is in the specified register (*nn*) at that time. To display the current result in the X-register, press VIEW · X. (This appears in the display as **VIEW ST _** and then **VIEW X**.)

The display called by VIEW *nn* will remain until another instruction (like VIEW) specifically changes the display, the display is cleared, or the program is interrupted. CLD (*clear display*) is a programmable function to clear the display. An interrupted program will display the ⊱ again when restarted.

> **Note:** If, during program execution, you have a turned-off printer attached to the HP-41CV, a VIEW or AVIEW instruction will stop the program. This is to give you time to write down the results. Press R/S to restart the program.*

Pause (PSE). If you include a PSE instruction in a program, it will temporarily suspend execution for about 1 second. (You can insert more than one PSE to create a longer pause.) During the pause, the display shows the current X-register or Alpha register contents, so you can view and record that value. Each time a PSE is executed, the **PRGM** annunciator blinks, letting you know that the program is still running.

Run/Stop (R/S). If you include a R/S (STOP) instruction in a program, the program halts indefinitely until you restart it by pressing R/S again.† This gives you plenty of time to record a result, or, as mentioned above, to enter a new number for data input.

If, for example, in the program **AREA** you wanted to know the result of r^2 as well as πr^2, you could insert a VIEW *nn*, PSE , or R/S in the programmed sequence:

01 LBL^{T}AREA	In the display, the ^{T} ("raised T") always precedes an Alpha string.‡
02 X↑2	
03 PSE	Display shows result of r^2.
04 PI	
05 *	
06 END	Display shows result of πr^2.

* If you don't want the VIEW and AVIEW instructions to interrupt the program, you should turn the printer on or disconnect the printer or clear flag 21.

† Pressing R/S in Program mode stores **STOP** (the same as STOP). The *run* portion of the R/S function is *not* programmable.

‡ The printer encloses Alpha strings with quotation marks and precedes a LBL with a diamond, as in 01♦LBL "VECTOR".

Providing an Auditory Signal (BEEP)

The BEEP function produces a series of tones. The BEEP instruction in a program can be used to provide a signal that a program is finished, that it has stopped and is waiting for input, or that some particular stage or condition in the program has been reached. For example:

01 LBL⊤AREA	
02 X↑2	
03 PSE	
04 PI	
05 *	
06 BEEP	Sounds tones when program is done.
07 END	

Using Messages in Programs

To have a message displayed during program execution, you need two instructions: one instruction consisting of an Alpha string (the message), and one instruction to display the Alpha register. When the message-containing program line is executed, that message is placed into the Alpha register. An AVIEW instruction will then display it.

Displaying the Contents of the Alpha Register (AVIEW)

The AVIEW (*Alpha view*) function in a program will display (as a message) whatever is currently in the Alpha register when AVIEW is executed. Like VIEW , AVIEW maintains its display until that display is replaced by another display instruction, or until the display is cleared. (For the use of this function with a printer, see the note on the previous page.)

CLD (*clear display*) clears message displays.

Normally, the program line before the AVIEW instruction contains the Alpha message. Just press ALPHA , enter the message, and press ALPHA again (activating and deactivating the Alpha keyboard). This stores the message in the program, but does *not* affect the Alpha register *until* that instruction is executed. *The maximum number of Alpha characters in a program line is 15.*

AVIEW can be used to provide a commentary on the progress of program execution, or it can be used in conjunction with ARCL to label (that is, provide a message with) data output. (The latter is discussed under "Labeling Data Output," after the next topic.)

01 LBL^{T}AREA	
02^{T}AREA OF CIRCLE 03 AVIEW 04 PSE	The ^{T} in the display signifies an Alpha string. Displays message that this program is for the area of a circle. The **PSE** (pause) prolongs the display.
05 X↑2	
06 PI	
07 *	
08 CLD	Clears the message display so that the result in the X-register will show.
09 END	

Prompting for Data Input (PROMPT)

The easiest programs to use are those that are self-explanatory. You can use the Alpha capability of the HP-41 to include messages (prompts) in a program when input is needed (or output is given). A PROMPT instruction in a program is the easiest way to ask for data input. (It can also be used to label data output, if you want the program to stop when it does so.)

The PROMPT operation combines two distinct operations in one, which are ideally suited for prompting for data input:

1. It stops program execution.
2. It displays the Alpha register (which must already contain the message you want displayed).

After reading the message, and supplying input if needed, you restart the interrupted program with R/S. (The data you enter can be Alpha data if you activate the Alpha keyboard.)

For instance, in the AREA program as written (page 64), it is assumed that the value for the radius will be entered before executing the program. However, it would make the program easier to use if the program were rewritten to ask for the data it needs:

01 LBL^{T}AREA	
02^{T}AREA OF CIRCLE	
03 AVIEW	
04 PSE	
05^{T}RADIUS?	New Alpha register contents.
06 PROMPT	Stops program and displays **RADIUS?**. After putting in r, restart program with [R/S].
07 X↑2	Finds r^2, etc.
08 PI	
09 *	
10 END	Displays final result.

Labeling Data Output ([ARCL])

The most readable method of displaying data results is with an [ARCL] . . . [AVIEW] sequence to display the contents of the X-register appended to a message in the Alpha register. (Alpha recall can be executed as a shifted function on the Alpha keyboard—[ARCL]—or by its Alpha name—[ARCL].)

[ARCL] *nn* (*Alpha recall*) operates by recalling the contents of the specified register (*nn*) into the Alpha register. To use the contents of the X-register, *nn* is [·] [X]. The recalled contents are then appended to whatever the Alpha register already holds. (This is different from the [RCL] function, which does not append what it recalls to pre-existing contents.) This sequence is used to append a result, say 36, to a message previously placed in the Alpha register by the program, such as **AREA=**. When the function [ARCL] [·] [X] is executed, the Alpha register will contain **AREA=36**.

The programmed sequence is:

1. Put message in Alpha register ([ALPHA] *message* [ALPHA]).
2. [ARCL] [·] [X] to append current contents of X-register to what's in Alpha register.
3. [AVIEW] to display the combined data output and its label.

This sequence does not make the program stop (which [PROMPT] does). However, the display of the appended Alpha register persists until it is replaced (see the explanation of [AVIEW], above). If you want to make sure the display will remain long enough for you to copy it down, you can add a [PSE] or a [R/S] after step 3.

Example: Take the original AREA routine (page 64) and revise it to display messages for a heading description (**AREA OF CIRCLE**), input (**RADIUS?**) and output (**AREA=**). Use the [AVIEW], [PROMPT], and [ARCL] functions. Enter this new program, labeled CIRCLE, into program memory.

Keystrokes	Display	
PRGM		Program mode. (Display depends on contents and position of program memory.)
GTO · ·	00 REG 44	Packs memory and inserts END after previous routine, if needed.
LBL	01 LBL _ _	Cues for label.
ALPHA CIRCLE ALPHA	1 LBL^TCIRCLE	New global label for new program.
ALPHA AREA OF CIRCLE	A OF CIRCLE _	Scrolls through display.
AVIEW ALPHA	03 AVIEW	
PSE	04 PSE	Use XEQ or a User key.
ALPHA RADIUS? ALPHA	05^TRADIUS?	Alpha message.
PROMPT	06 PROMPT	Use XEQ or a User key.
x^2	07 X↑2	
π	08 PI	
×	09 *	
ALPHA AREA=	10^TAREA=	Alpha message.
ARCL ·	11 ARCL ST _	The · (and the ST) indicates a stack register.
X	11 ARCL X	Instruction to recall contents *from* X-register *into* Alpha register.
AVIEW ALPHA	12 AVIEW	
GTO · ·	00 REG 37	(Optional. Adds END.)
PRGM		Returns to Execution mode.

Now try running program CIRCLE for circles of radius 1.6, $\pi\sqrt{2}$, and 32×10^{-6} (as done before in the example on page 69).

Keystrokes	Display	
CIRCLE	AREA OF CIRC EA OF CIRCLE	Scrolls through display.
	RADIUS?	
1.6 R/S	AREA=8.0425	
CIRCLE (or RTN R/S)	RADIUS?	(After AREA OF CIRCLE display.)
2 $\sqrt{x}$ π × R/S	AREA=62.0126	
CIRCLE (or RTN R/S)	RADIUS?	You can use RTN R/S to run the same program again.
32 EEX 6 CHS R/S	AREA=3.2170E-9	Result is 3.2170×10^{-9}.

Error Stops

If an error occurs in the course of a running program, program execution halts and an error message appears in the display. (There is a list of all error messages and conditions in appendix A.)

To see the line in the program containing the error-causing instruction, set Program mode. The program will have stopped at the illegal instruction. You can then correct the instruction, as will be explained under "Inserting, Deleting, and Altering Program Lines," page 78.

It is a good idea always to go into Program mode to check which program instruction caused a given error. Note especially that the error **NONEXISTENT**, which can occur when you execute a program, does not necessarily mean that the program you called is nonexistent. It often means that a register (such as with a [STO] instruction) or label called from *within* the program does not exist. Pressing any key (including [PRGM]) serves to clear the error display and carry out its own operation. Pressing [←] clears the error display without doing anything else.

Modifying Programs in Main Memory

To modify a program you have stored, you need to first position the program pointer to the right program, then position it to the right line, then add or delete the necessary instructions. If a program has an error that you need to locate and then correct, use single-step execution (page 69) to locate the error.

Catalog 1: The Catalog of Programs

Catalog 1 contains a list of all of the global labels and **END** instructions recorded in program memory. Along with the permanent **END** (**.END.**) is the number of remaining registers available for programming.*

- Pressing [CATALOG] 1 starts a fast display of all global labels and **END**s. When the catalog finishes, the display returns to the X-register.
- [R/S] will stop and restart this listing.
- [SST] (*single step*) and [BST] (*back step*) will step through the halted listing one line at a time.
- While the catalog is stopped, pressing any key besides [R/S], [SST], [BST], or [USER] will change the display and break off the catalog operation.
- While the catalog is running, pressing any key besides [R/S] or [ON] speeds up the listing.

(With a printer, the catalogs only print in Trace mode.)

* A *program* literally represents the instruction series between two **END** statements (or the top of memory and one **END**). Most program instructions take one byte; some take two. Each register equals seven bytes of memory.

If you keyed in the AREA program (from page 64) and the CIRCLE program (from page 75), then program memory looks like this:

```
01 LBL^T AREA
02 X↑2
03 PI
04 *
05 END
01 LBL^T CIRCLE
02^T AREA OF CIRCLE
03 AVIEW
04 PSE
05^T RADIUS?
06 PROMPT
07 X↑2
08 PI
09 *
10^T AREA=
11 ARCL X
12 AVIEW
13 END
.END.
```

If you execute CATALOG 1, you will see the following:

Keystrokes	Display	
CATALOG	CAT _	CATALOG is a parameter function, so it cues you: *Which catalog?*
1 R/S	LBL^{T}AREA	After specifying 1, immediately press R/S so you can see and read each entry.
SST	END	
SST	LBL^{T}CIRCLE	
SST	END	
SST	.END. REG 37	37 uncommitted registers still available for programming.*
SST		Going past the permanent .END. cancels the catalog operation and returns the X-register display.

Using CATALOG 1 to Position the Program Pointer. As the listing of catalog 1 proceeds, the program pointer moves to the global label or **END** currently displayed. One way to gain access to a particular program (with a global label) is to stop the catalog listing at its global label, then go into Program mode.

Every time catalog 1 runs to completion, the last program listed becomes the current program, and the permanent **.END.** becomes the current line.

Program Routines Without Global Labels. If you have neglected to include any global label for a sequence of program lines between two **END**s, then only the **END** for that routine will appear in the catalog 1 listing. The only way to gain access to a program that has no global label is by using catalog 1 to position the program pointer to such a solo **END**. This makes that program the current program. Then switch to Program mode to delete or alter those lines.

Moving the Program Pointer (GTO)

There is another way to locate a particular program routine besides using catalog 1. The GTO (*go to*) function is a parameter function that will move the program pointer to any specified global label. In Execution mode, it will also move to a local label if that local label is within the current program.

In Execution Mode: The key sequence is GTO *label*. (You cannot use a User-defined key to supply that parameter: it must be spelled out.) To go to a particular line number within the current program, press GTO ⊡ *nnn*, where *nnn* is a three-digit line number.

Pressing RTN (*return*) returns the pointer to line 00 of the current program.

In Program Mode: The sequence is GTO ⊡ *global label* or GTO ⊡ *nnn* (for a line number). *You must include the* ⊡ *to prevent recording the* GTO *instruction.* Notice that from within Program mode, you cannot go to a *local* label. (For a definition of global and local labels, refer to pages 65-66.)

Viewing a Program Line by Line (SST, BST)

In Program mode, pressing SST (*single step*) or BST (*back step*) will move the program pointer forward or backward one line within the current program. This displays the line only; no execution takes place. When the line position reaches the end of the program, it "wraps around" to the first line of the same program. (SST in Execution mode effects stepwise execution; page 69.)

SST and BST are nonprogrammable, that is, they are not recorded as program instructions.

Inserting, Deleting, and Altering Program Lines

To alter an instruction, first delete it, then add the new version.

Deleting a Line (←). Use ← (*back arrow*) *in Program mode* to delete a program line: just move to the line you want to delete (use GTO, SST, or BST) and press ←.

As the effect of [←] varies in Execution mode, so it does in Program mode. [←] has a different effect depending on whether a program instruction is in the process of being entered or has already been recorded.

- If you are in the process of entering a program instruction that is not yet complete (a string of digits or Alpha characters, or a parameter function), [←] deletes the last digit or character or function keyed in.
- If an instruction has already been completed, pressing [←] deletes the entire line.

Don't use [CLx] *for clearing program lines*, because it will be recorded as an instruction. [←] is nonprogrammable.

Example: Delete the message AREA OF CIRCLE from line 02 of program CIRCLE.

Keystrokes	Display	
[GTO] [ALPHA] CIRCLE [ALPHA]		Move program pointer to CIRCLE.
[PRGM]	1 LBL^{T}CIRCLE	Program mode: displays first line of CIRCLE.
[SST]	EA OF CIRCLE	Moves to second line (display scrolls).
[←]	1 LBL^{T}CIRCLE	Delete that line; display moves back one line.
[SST]	02 AVIEW	See that second line is now different.

Deleting Several Lines ([DEL]). The parameter function [DEL] (*delete*) is used *in Program mode* to delete more than one line. It is nonprogrammable (that is, it cannot be recorded as a program line). The function [DEL] requires a three-digit parameter specification to indicate the number of lines—from the current line on—to delete.

Example: Delete the last three lines before the END of CIRCLE—the output display lines. (Refer to the listing of the program on page 77.) These are now lines 09 to 11 owing to the one-line deletion above. The HP-41 should be in Program mode, and CIRCLE should be the current program.

Keystrokes	Display	
[GTO]	03 GTO _ _	
[.]	GTO _ _ _	The [.] key means not to record the function. Asks: *Which three-digit line number?*
009	09^{T}AREA=	Goes to line 09.
[DEL]	DEL _ _ _	
003	08 *	Display moves back one line.
[SST]	09 END	The lines previously numbered 09 to 11 are gone.

Inserting Lines. To make additions to a program, move the program pointer (using GTO, SST, or BST) to the line *preceding* the desired point of insertion. The instruction you then key in (in Program mode) will be added *following* the currently displayed line. Any subsequent lines are "bumped" down a line number.

This procedure is the same as when you are entering a new program: whenever Program mode is active, any programmable operation you enter is stored as a program line *after* the line previously displayed.

Example: Find the program AREA (as entered on pages 64-65) and add the **RADIUS?** input prompt to it, as well as the **AREA=** output labeling (that were in CIRCLE).

Keystrokes	Display	
GTO · ALPHA AREA ALPHA	01 LBL⊤AREA	In Program mode, use · with GTO. Position is at line 01.
ALPHA RADIUS? ALPHA	02⊤RADIUS?	Adds this line as line 02.
PROMPT	03 PROMPT	Adds as line 03.
SST SST SST	06 *	Last line currently in AREA (before the **END**).
ALPHA AREA=	07⊤AREA=	You don't need to turn off the Alpha keyboard yet, since the next two functions also use the Alpha keyboard.
ARCL · X	08 ARCL X	
AVIEW ALPHA	09 AVIEW	Three new lines added.
SST	10 END	End of AREA.

Clearing a Program (CLP)

CLP (*clear program*) is a nonprogrammable parameter function that can be executed in Program or Execution mode. It requires (and cues for, using an input cue) a global label to complete operation. (You cannot use a User-defined key to supply that parameter: it must be spelled out.)

When CLP *global label* is executed, that global label and all preceding lines (up to the preceding **END**) and all following lines (up to and including the next **END** instruction) are deleted. Any User key assignment for that global label is also cancelled, and program memory is packed.

Executing CLP ALPHA ALPHA clears the current program.

Example: Clear program CIRCLE from memory.

Keystrokes	Display	
CLP	CLP _	
ALPHA CIRCLE ALPHA	00 REG 215	
PRGM		Returns to Execution mode.

Program memory (if you have followed all the examples in this section) now looks like this:

```
01 LBLᵀAREA
02ᵀRADIUS?
03 PROMPT
04 X↑2
05 PI
06 *
07ᵀAREA=
08 ARCL X
09 AVIEW
10 END
.END. REG 215
```

Structuring a Program

Many problems you want to program will probably be more complicated than AREA. It takes time to structure a program to incorporate the right input at the right time, perform several different calculations and manipulations, and return to you the output (results) you want in an understandable form. In fact, defining your input and output goes a long way toward helping you decide how to set up a program. The key in the HP-41 is to use the alphanumeric capacity to store and display messages, and to stop a program to wait for your response (input).

Stating the Problem

As a somewhat more complicated example, consider a program to find the roots of the equation $ax^2 + bx + c = 0$, where a, b, and c are constants. The solution can be found using the quadratic formula, namely:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

If the radicand (the expression under the root sign) is negative, indicate that the roots are complex.

To solve this problem, you can break it down into the following steps.

1. Find $b^2 - 4ac$.
2. If the difference is positive, find $\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$. These are two real roots.
3. If the difference is negative, display a message that the roots are complex.

If you were working on this problem from the keyboard (that is, not in a program), it would be straightforward: you would enter the variables as necessary and, after finding the radicand, determine by inspection whether the roots were real or complex and how to finish the calculation. In a program, however, all these steps should be taken care of automatically, which makes for a more efficient program.

Refining the Problem for the HP-41

Now try to express the above steps as the HP-41 must take them, for instance:

1. Find $2a$; store in storage register R_{00} for later use.
2. Find $-b$; store in R_{01}.
3. Find $4ac$ and b^2, then find $b^2 - 4ac$.
4. If this result is negative, display **ROOTS COMPLEX** and stop.
5. If this result is positive, take the square root of this value and store in R_{02}.
6. Add this value to $-b$ (R_{01}) and divide by $2a$ (R_{00}) for the first root. Display the result.
7. To find the second root, find $R_{01} - R_{02}$, then divide by R_{00}. Display the result.

These steps illustrate the importance of using data storage registers to store intermediate results of calculations. Note also that since the original a- and b-values are used more than once, they also should be stored in registers for later recall. In fact, it is often most convenient to prompt for and enter all data input at the beginning of a running program, and store them in storage registers until they're needed.

Step 4 involves the programming of a "conditional test," which checks whether a certain condition is true or false, with the outcome affecting how program execution continues. The HP-41 provides many conditional tests, four of which are printed on the keyboard ([x = y?], [x ≤ y?], [x > y?], and [x = 0?]). Most of the conditional tests check the value in the X-register (x) versus the value in the Y-register (y) or versus zero. This example program uses the conditional test [X < 0?] to illustrate the versatility of an HP-41 program. There is more discussion of conditional tests in *HP-41CV Operation in Detail.*

So here's an HP-41 program to find the real roots of the quadratic equation, given a, b, and c, the coefficients in an equation of the form $ax^2 + bx + c = 0$.

Step 1.

Keystrokes	Display	
PRGM		Display depends on the current program line.
GTO · ·	00 REG 42	
LBL ALPHA QUAD ALPHA	01 LBL^{T}QUAD	Global label for program.
ALPHA a=? ALPHA	02^{T}a=?	Asks for *a*-value.
PROMPT	03 PROMPT	First prompt for data.
2	04 2_	
×	05 *	
STO 00	06 STO 00	Stores $2a$ in R_{00}.

Step 2.

Keystrokes	Display	
ALPHA b=? ALPHA	07^{T}b=?	Prompt for *b*-value.
PROMPT	08 PROMPT	
CHS	09 CHS	
STO 01	10 STO 01	Stores $-b$ in R_{01}.

Step 3.

Keystrokes	Display	
ALPHA c=? ALPHA	11^{T}c=?	
PROMPT	12 PROMPT	Prompt for *c*-value.
RCL 00	13 RCL 00	Recalls $2a$.
×	14 *	Calculates $2ac$.
2	15 2_	
×	16 *	Calculates $4ac$. ($2ac$ was in the Y-register, 2 was in X.)
RCL 01	17 RCL 01	Recalls $-b$.
x^2	18 X↑2	Calculates $(-b)^2$.
x≷y	19 X<>Y	Switches the positions of $4ac$ and b^2.
−	20 −	Finds $b^2 - 4ac$.

Step 4.

Keystrokes	Display	
X<0?	21 X<0?	Checks whether value in X-register is negative. If yes, the next instruction gets executed. If not, one instruction is skipped.
GTO 01	22 GTO 01	If condition true, skip the rest of the program and "go to" label 01 (at line 42). *Notice that the "go to" instruction is just* GTO 01 *and not* GTO LBL 01.

Step 5.

Keystrokes	Display	
$\sqrt{x}$	23 SQRT	$\sqrt{b^2 - 4ac}$.
STO 02	24 STO 02	Stores into R_{02}.

Step 6.

Keystrokes	Display	
RCL 01	25 RCL 01	Recalls $-b$.
+	26 +	$-b + \sqrt{b^2 - 4ac}$.
RCL 00	27 RCL 00	Recalls $2a$.
÷	28 /	The first root.
ALPHA ROOTS=	29ᵀROOTS=	(Include a space after the equals sign.) This sequence displays the message **ROOTS=** and the first root.
ARCL · X	30 ARCL X	
AVIEW ALPHA	31 AVIEW	
PSE	32 PSE	

Step 7.

Keystrokes	Display	
RCL 01	33 RCL 01	Recalls $-b$.
RCL 02	34 RCL 02	Recalls $\sqrt{b^2 - 4ac}$.
−	35 −	$-b - \sqrt{b^2 - 4ac}$.
RCL 00	36 RCL 00	Recalls $2a$.
÷	37 /	Second root.
ALPHA AND	38ᵀAND	(Include a space after **AND**.) This sequence displays the message **AND** and the second root.
ARCL · X	39 ARCL X	
AVIEW ALPHA	40 AVIEW	
RTN	41 RTN	Stops the main part of the program.

Step 4, continued.

Keystrokes	Display	
LBL 01	42 LBL 01	This routine, labeled 01, is only executed when the radicand < 0 (see lines 21 and 22).
ALPHA ROOTS COMPLEX	ROOTS COMPL OTS COMPLEX_	Display for complex-roots condition.
AVIEW ALPHA	44 AVIEW	
GTO · ·	00 REG 29	(Optional. Adds **END**.)
PRGM		Returns to Execution mode.

To test this program, try executing it to solve for the roots of the equation $y = x^2 + 7x + 12$. (The roots should be -3, -4.)

Then try $1.5x^2 - x + 13$.

Keystrokes	Display	
QUAD	a=?	Start program. Put in *a* and restart program.
1 R/S	b=?	Put in *b* and restart program.
7 R/S	c=?	Put in *c* and restart program.
12 R/S	ROOTS= −3.000 OOTS= −3.0000 AND −4.0000	
QUAD (or RTN R/S)	a=?	
1.5 R/S	b=?	
1 CHS R/S	c=?	
13 R/S	ROOTS COMPLE OOTS COMPLEX	Scrolls through display.

Copying a Program from an Application Module (COPY)

Programs provided to the HP-41 by a plug-in application module or by a peripheral device can be executed just like programs that you have entered and stored in main program memory. They can also be accessed and copied by the HP-41.*†

* Application modules and peripheral devices contain both programs and functions. You can only copy programs, and not functions, into the HP-41 program memory. Catalog 2 is a list of all functions and programs currently plugged in from external sources. All these user-accessible programs appear in catalog 2 preceded by a ᵀ. These are the only ones the user can view or copy.

† Programs on magnetic cards that have been made private cannot be copied, viewed, or altered—only executed. For more information, refer to the *HP 82104A Card Reader Owner's Handbook*.

If you want to keep an application program in main memory (so it will still be available if you remove the module), then copy that program into main memory.

If you want to alter an applicaton program, you need to first copy it into main memory, then edit the version in main memory and keep it there. Your version of the program (the one in main memory) will run preferentially, whether the application module is plugged in or not, so you don't need to rename your version of the of the program.

- To access a program in a peripheral device, use [GTO] [·] *global label*. (In Execution mode, [GTO] *global label* suffices.) [SST] and [BST] will step through the program lines for review or stepwise execution. (Just as in "Moving the Program Pointer" and "Viewing a Program Line by Line," page 78.)

  However, *you cannot modify these programs* before they have been copied into the program memory of the HP-41. Attempting to do so will result in the error message **ROM** (*read-only memory*).

- To copy a program from a peripheral device and into HP-41 program memory, execute [COPY] *global label*. (The display **COPY _** will cue for the global label).

  If the program pointer is already positioned to the program you want to copy (it is the current program), just execute [COPY] [ALPHA] [ALPHA]. (With no parameter given, [COPY] defaults to the current program.)

If the computer cannot find the global label you specify, **NONEXISTENT** results. If the program you seek to copy already exists in program memory, the message **RAM** results (meaning the program is already in RAM, *random-access memory*).

If there is not enough room in program memory to copy the program, the display will show **PACKING** and **TRY AGAIN**. Refer to "Memory Limitations," page 85.

Initializing Computer Status

When you store a program that someone else has written, or when you write your own program, it is a good idea to be sure that any necessary status conditions are set. For example, if the program will do any calculations with angles, the program should include a setting for the angular mode that these calculations assume. Solutions books from the HP Users' Library include a table of "Registers, Status, Flags, Assignments" that tells you the status assumed for the display format, User keyboard, and angular mode, as well as the number of registers of memory needed for the program.

Flag settings (explained in detail in *HP-41CV Operation in Detail*) can also affect program operation. Unintentional and incompatible flag settings can occur with peripheral devices (like the printer) when certain procedures are not carried out as intended. For instance, running a program with the printer attached *but off* will alter normal program execution.

Programs as Customized Functions

The first paragraph of this section stated that writing programs is a way to design your own operations. For example, the **QUAD** program, which solves the quadratic equation for a set of coefficients, could be assigned to a key on the User keyboard, making it executable in one keystroke, like a function. The programmed quadratic solution would then be a specialized function key.

Below is another example of a programmed, customized function. This example is for vector addition, a technique that uses the [Σ+] function. The method of calculation was shown under "Vector Arithmetic," page 59. The input data values are assumed to be θ and r (any angular mode), and the statistics registers are assumed to still be R_{11} to R_{16}. (To make the program more foolproof, you can include a [ΣREG] 11 instruction right after the label to make sure that the statistics registers are located at R_{11} through R_{16}.)

Keystrokes	Display	
[PRGM]	00 REG 29	
[GTO] [.] [.]		
[LBL] [ALPHA] VECTOR [ALPHA]	1 LBL^{T}VECTOR	
[CLΣ]	02 CLΣ	
[ALPHA] THETA1=? [ALPHA]	03^{T}THETA1=?	As always, prompts are optional—but they make a program much easier to use. If you skip the prompts, just remember to enter θ_1 first, then r_1, separated by [ENTER↑]. Do this *before* executing the program.
[PROMPT]	04 PROMPT	
[ALPHA] R1=? [ALPHA]	05^{T}R1=?	
[PROMPT]	06 PROMPT	
[P→R]	07 P−R	
[Σ+]	08 Σ+	
[ALPHA] THETA2=? [ALPHA]	09^{T}THETA2=?	If you don't use prompts, use a [R/S] ([STOP]) instruction. Then put θ_2 in Y (enter first) and r_2 in X (enter second).
[PROMPT]	10 PROMPT	
[ALPHA] R2=? [ALPHA]	11^{T}R2=?	
[PROMPT]	12 PROMPT	
[P→R]	13 P−R	
[Σ+]	14 Σ+	
[RCL] 13	15 RCL 13	
[RCL] 11	16 RCL 11	

Keystrokes	Display	
R→P	17 R−P	Results: $\Sigma\theta$ in Y-register, Σr in X-register.
ALPHA ΣTHTA=	18ᵀΣTHTA=	Again, the lines after 17 are strictly for ease in reading the results. In fact, this routine would be more like an HP-41 standard function if it didn't include messages. Just remember: the resulting $\Sigma\theta$ is in the Y-register; Σr is in the X-register.
ARCL · Y ALPHA	19 ARCL Y	
PROMPT	20 PROMPT	
ALPHA ΣR=	21ᵀΣR=	
ARCL · X	22 ARCL X	
AVIEW ALPHA	23 AVIEW	
		The PROMPT function is used to display the Alpha register (with $\Sigma\theta$) and interrupt the program.
GTO · ·	00 REG 19	(Optional. Packs memory and inserts **END**.)
PRGM		Returns to Execution mode.

Using this program to find the sum of the vectors (150, 45°) and (40, 205°):

Keystrokes	Display	
ASN ALPHA VECTOR ALPHA LN	ASN VECTOR_ SN VECTOR 15	This assigns VECTOR to the LN key on the User keyboard.
VECTOR	THETA1=?	Execute VECTOR by pressing LN on the User keyboard (**USER** on).
45 R/S	R1=?	
150 R/S	THETA2=?	
205 R/S	R2=?	
40 R/S	ΣTHTA=51.9389	Programs stops, displays $\Sigma\theta$.
R/S	ΣR=113.2417	Press R/S to continue program and see Σr. (By abbreviating **ΣTHETA** to **ΣTHTA**, the whole display fits on one line.)

Appendices

Appendix A

List of Errors

Following is a simplified description of each HP-41CV error message.

To clear an error message, press [←]. A function that causes an error does not get executed.

Error	Meaning
ALPHA DATA	Nonnumeric data used.
DATA ERROR	Illegal operand.
MEMORY LOST	Continuous Memory has been cleared and reset.
NONEXISTENT	The register, label, or function specified does not exist.
OUT OF RANGE	Number too large.
PRIVATE	Program on card or cassette is private.
RAM	The global label specified already exists in main memory.
ROM	You cannot access a program in ROM.

Appendix B

Battery, Warranty, and Service Information

Contents

The Input/Output Ports

Keep the caps on the input/output ports whenever nothing is plugged into them.

CAUTION

Do not insert your fingers or any object other than an HP module or plug-in accessory into an input/output port. Doing so could interrupt Continuous Memory and possibly damage the computer.

Batteries and Power Use

The HP-41 is powered by four batteries. Depending on how it is used, the HP-41 can operate up to six months or more on a set of alkaline batteries. The batteries supplied with the computer are alkaline N cells, but a rechargeable battery pack (nickel cadmium cells) can also be used.

The total number of operating hours supplied by the batteries depends greatly on what kinds of operations you do and how much you use peripheral devices. Without using peripheral devices, a set of four fresh alkaline batteries will provide about 45 to 85 hours of *continuous* program running (the most power-consuming kind of computer use—refer to "Power Consumption," below). When only the display is on and no operations are being performed, much less power is consumed.

The actual lifetime of the batteries depends on how often you use the HP-41 and its peripherals, whether you use the HP-41 more for running programs or more for manual calculations, and which functions you use. Next to using peripheral devices, the most power-consuming operations are running programs and using the catalogs.

If the computer remains turned off, a set of fresh batteries will preserve the contents of Continuous Memory for as long as the batteries would last outside of the computer—about 1½ years for alkaline batteries.

Power Consumption

The actual rate of power consumption depends upon how the HP-41 is being used at any given time. There are three basic power consumption modes:

- **Operating:** high current drain (5 to 20 mA). This corresponds to running a program or performing an operation (pressing a key).
- **Idle:** moderate current drain (0.5 to 2.0 mA). This mode corresponds to the display being on.
- **Off:** low current drain (0.01 to 0.05 mA). Exists when the computer is off.

While the computer is turned on, typical computer use is a mixture of idle time and operating time. Therefore, the actual lifetime of the batteries depends on how much time the computer spends in each of the three modes.

A freshly charged HP 82120A Rechargeable Battery Pack has a capacity of 65 mAH (milliampere-hours). A fresh set of alkaline batteries provides approximately 500 mAH.

Power Consumption by Peripheral Devices

When you use peripheral devices that draw power from the HP-41 batteries (such as the card reader or the optical wand), total battery life will be reduced considerably. If you use peripherals frequently, it is recommended that you power the HP-41 with an HP 82120A Rechargeable Battery Pack.

Effects of Clearing Memory, Power Interruptions, and Low Power

Clearing Memory. Resetting Continuous Memory ([←]/[ON]) does:

- Clear all of main memory.
- Clear all User function assignments.
- Reset all flags to their initial, power-on settings.
- Reset the allocation of registers in main memory to 273 registers for data storage.

Refer also to "Continuous Memory" in section 1 for information about clearing and resetting memory.

Low-Power Indication

The **BAT** (*battery*) annunciator appears in the display when the available battery power is running low. If a peripheral is in use, disconnecting it (after turning off both the HP-41 and the peripheral) will significantly extend battery life.

With alkaline batteries installed (and no peripheral attached):

- The computer can be used for about 2 to 7 hours of continuous program running after **BAT** first appears.*
- If the computer remains turned off, the contents of its Continous Memory will be preserved for about a month after **BAT** first appears.

---

* Note that this is the time available for *continuous operation*. If you are using the computer for manual calculations—a mixture of the idle and operating modes—the computer can be used for a much longer time after the **BAT** first appears.

Battery Replacement and Installation

The batteries supplied with the HP-41, as well as the alkaline batteries listed below for replacement, are *not* rechargeable.

WARNING

Do not attempt to recharge the batteries; do not store batteries near a source of high heat; do not dispose of batteries in fire. Doing so may cause the batteries to leak or explode.

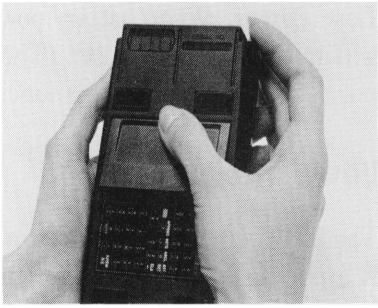
The following batteries are recommended for replacement in your HP-41:

Eveready E90*	Mallory MN9100	VARTA 7245
National AM5(s)	Panasonic AM5(s)	

The contents of the computer's Continuous Memory are preserved for a short time while the batteries are out of the computer (provided that you turn off the computer before removing the batteries). This allows you ample time to replace the batteries without losing data or programs. If the batteries are left out of the computer for an extended period, the contents of Continuous Memory may be lost.

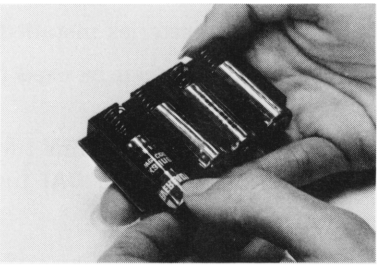
To install new batteries, use the following procedure:

1. Be sure the computer is off.
2. Holding the computer as shown, push up on the battery holder until it pops out.



* Not available in the United Kingdom or Republic of Ireland.

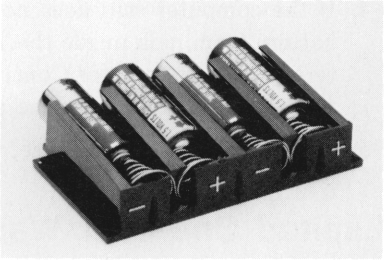
3. Remove the batteries from the battery holder.



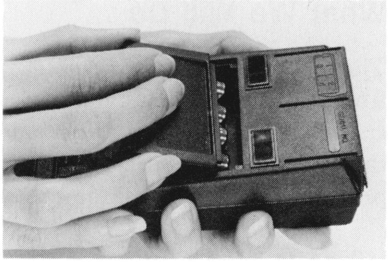
CAUTION

In the next step, replace all four batteries with fresh ones. If you leave an old battery inside, it may leak. Furthermore, be careful not to insert the batteries backwards. If you do so, the contents of Continuous Memory may be lost.

4. Insert the new batteries by matching the position of their polarity marks to those on the battery holder. If any of the batteries are inserted backwards, the computer will not turn on.



5. Insert the battery pack into the computer such that the exposed ends of the batteries are pointing toward the input/output ports.
6. Push the upper edge of the battery pack into the HP-41 until it goes no further. Then snap the lower edge of the holder into place.



7. Turn the computer on. If for any reason memory has been cleared (that is, its contents have been lost), the display will show **MEMORY LOST**. Pressing any key will clear this message from the display.

Verifying Proper Operation

If it appears that the computer will not turn on or otherwise is not operating properly, review the following steps.

1. Be sure that all the batteries are inserted with the correct polarity and that the battery contacts are not dirty.
2. If the computer does not respond to keystrokes, try to reset it as follows: press and hold the ON and ENTER↑ keys simultaneously, then release them. Turn the computer on, if necessary, and test for a response to keystrokes.
3. If there is no response, remove and reinsert the battery pack.

   If the computer still does not turn on, install fresh batteries.

   If this does not suffice, remove the battery pack and let the computer discharge overnight. When you reinstall the batteries and turn the computer on, if the display shows **MEMORY LOST**, then memory and the computer have been cleared and reset.
4. If the computer still does not respond to keystrokes, remove the battery pack and short the end battery terminals inside the HP-41 together. *Only momentary contact is required.* Replace the batteries. The contents of Continuous Memory will be lost, and you might need to press the ON key more than once to turn the computer back on.
5. If there is still no response, the computer requires service.

Limited One-Year Warranty

What We Will Do

The HP-41 (*except the batteries and damage caused by the batteries*) is warranted by Hewlett-Packard against defects in material and workmanship for one year from the date of original purchase. If you sell your unit or give it as a gift, the warranty is automatically transferred to the new owner and remains in effect for the original one-year period. During the warranty period, we will repair or, at our option, replace at no charge a product that proves to be defective, provided you return the product, shipping prepaid, to a Hewlett-Packard service center.

What Is Not Covered

The batteries or damage caused by the batteries are not covered by this warranty. However, certain battery manufacturers may arrange for the repair of the computer if it is damaged by the batteries. Contact the battery manufacturer first if you computer has been damaged by the batteries.

This warranty does not apply if the product has been damaged by accident or misuse or as the result of service or modification by other than an authorized Hewlett-Packard service center.

No other express warranty is given. The repair or replacement of a product is your exclusive remedy. ANY OTHER IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS IS LIMITED TO THE ONE-YEAR DURATION OF THIS WRITTEN WARRANTY. Some states, provinces, or countries do not allow limitations on how long an implied warranty lasts, so the above limitation may not apply to you. IN NO EVENT SHALL HEWLETT-PACKARD COMPANY BE LIABLE FOR CONSEQUENTIAL DAMAGES. Some states, provinces, or countries 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, province to province, or country to country.

Warranty for Consumer Transactions in the United Kingdom

This warranty shall not apply to consumer transactions and shall not affect the statutory rights of a consumer. In relation to such transactions, the rights and obligations of Seller and Buyer shall be determined by statute.

Obligation to Make Changes

Products are sold on the basis of specifications applicable at the time of manufacture. Hewlett-Packard shall have no obligation to modify or update products once sold.

Warranty Information

If you have any questions concerning this warranty, please contact:

- In the United States:

  Hewlett-Packard Company
  Personal Computer Group
  Customer Communications
  11000 Wolfe Road
  Cupertino, CA 95014
  Toll-Free Number: (800) FOR-HPPC (800/367-4772)

- In Europe:

  Hewlett-Packard S.A.
  150, route du Nant-d'Avril
  P.O. Box
  CH-1217 Meyrin 2
  Geneva
  Switzerland
  Telephone: (022) 83 81 11

Note: Do *not* send computers to this address for repair.

- In other countries:

Hewlett-Packard Intercontinental
3495 Deer Creek Rd.
Palo Alto, California 94304
U.S.A.
Telephone: (415) 857-1501

Note: Do *not* send computers to this address for repair.

Service

Hewlett-Packard maintains service centers in most major countries throughout the world. You may have your unit repaired at a Hewlett-Packard service center any time it needs service, whether the unit is under warranty or not. There is a charge for repairs after the one-year warranty period.

Hewlett-Packard handheld computer products normally are repaired and reshipped within five (5) working days of receipt at any service center. This is an average time and could vary depending upon the time of year and work load at the service center. The total time you are without your unit will depend largely on the shipping time.

Obtaining Repair Service in the United States

The Hewlett-Packard United States Service Center for handheld and portable computer products is located in Corvallis, Oregon:

Hewlett-Packard Company
Service Department
P.O. Box 999
Corvallis, Oregon 97339, U.S.A.

or

1030 N.E. Circle Blvd.
Corvallis, Oregon 97330, U.S.A.

Telephone: (503) 757-2000

Obtaining Repair Service in Europe

Service centers are maintained at the following locations. For countries not listed, contact the dealer where you purchased your computer.

AUSTRIA
HEWLETT-PACKARD Ges.m.b.H.
Kleinrechner-Service
Wagramerstrasse-Lieblgasse 1
A-1220 Wien (Vienna)
Telephone: (0222) 23 65 11

BELGIUM
HEWLETT-PACKARD BELGIUM SA/NV
Woluwedal 100
B-1200 Brussels
Telephone: (02) 762 32 00

DENMARK
HEWLETT-PACKARD A/S
Datavej 52
DK-3460 Birkerød (Copenhagen)
Telephone: (02) 81 66 40

EASTERN EUROPE
Refer to the address listed under Austria.

FINLAND
HEWLETT-PACKARD OY
Revontulentie 7
SF-02100 Espoo 10 (Helsinki)
Telephone: (90) 455 02 11

FRANCE
HEWLETT-PACKARD FRANCE
Division Informatique Personnelle
S.A.V. Calculateurs de Poche
F-91947 Les Ulis Cedex
Telephone: (6) 907 78 25

GERMANY
HEWLETT-PACKARD GmbH
Kleinrechner-Service
Vertriebszentrale
Berner Strasse 117
Postfach 560 140
D-6000 Frankfurt 56
Telephone: (611) 50041

ITALY
HEWLETT-PACKARD ITALIANA S.P.A.
Casella postale 3645 (Milano)
Via G. Di Vittorio, 9
I-20063 Cernusco Sul Naviglio (Milan)
Telephone: (2) 90 36 91

NETHERLANDS
HEWLETT-PACKARD NEDERLAND B.V.
Van Heuven Goedhartlaan 121
NL-1181 KK Amstelveen (Amsterdam)
P.O. Box 667
Telephone: (020) 472021

NORWAY
HEWLETT-PACKARD NORGE A/S
P.O. Box 34
Oesterndalen 18
N-1345 Oesteraas (Oslo)
Telephone: (2) 17 11 80

SPAIN
HEWLETT-PACKARD ESPANOLA S.A.
Calle Jerez 3
E-Madrid 16
Telephone: (1) 458 2600

SWEDEN
HEWLETT-PACKARD SVERIGE AB
Skalholtsgatan 9, Kista
Box 19
S-163 93 Spanga (Stockholm)
Telephone: (08) 750 2000

SWITZERLAND
HEWLETT-PACKARD (SCHWEIZ) AG
Kleinrechner-Service
Allmend 2
CH-8967 Widen
Telephone: (057) 31 21 11

UNITED KINGDOM
HEWLETT-PACKARD Ltd
King Street Lane
GB-Winnersh, Wokingham
Berkshire RG11 5AR
Telephone: (0734) 784 774

International Service Information

Not all Hewlett-Packard service centers offer service for all models of HP computer products. However, if you bought your product from an authorized Hewlett-Packard dealer, you can be sure that service is available in the country where you bought it.

If you happen to be outside of the country where you bought your unit, you can contact the local Hewlett-Packard service center to see if service is available for it. If service is unavailable, please ship the unit to the address listed above under "Obtaining Repair Service in the United States." A list of service centers for other countries can be obtained by writing to that address.

All shipping, reimportation arrangements, and customs costs are your responsibility.

Service Repair Charge

There is a standard repair charge for out-of-warranty repairs. The repair charges include all labor and materials. In the United States, the full charge is subject to the customer's local sales tax. In European countries, the full charge is subject to Value Added Tax (VAT) and similar taxes wherever applicable. All such taxes will appear as separate items on invoiced amounts.

Computer products damaged by accident or misuse are not covered by the fixed repair charges. In these situations, repair charges will be individually determined based on time and material.

Service Warranty

Any out-of-warranty repairs are warranted against defects in materials and workmanship for a period of 90 days from date of service.

Shipping Instructions

Should your unit require service, return it with the following items:

- A completed Service Card, including a description of the problem.
- A sales receipt or other proof of purchase date if the one-year warranty has not expired.

The product, the Service Card, a brief description of the problem, and (if required) the proof of purchase should be packaged in the original shipping case or other adequate protective packaging to prevent in-transit damage. Such damage is not covered by the one-year limited warranty; Hewlett-Packard suggests that you insure the shipment to the service center. The packaged unit should be shipped to the nearest Hewlett-Packard designated collection point or service center. Contact your dealer for assistance. (If you are not in the country where you originally purchased the unit, refer to International Service Information above.)

Whether the unit is under warranty or not, it is your responsibility to pay shipping charges for delivery to the Hewlett-Packard service center.

After warranty repairs are completed, the service center returns the unit with postage prepaid. On out-of-warranty repairs in the United States and some other countries, the unit is returned C.O.D. (covering shipping costs and the service charge).

Further Information

Circuitry and designs 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 service center.

When You Need Help

Hewlett-Packard is committed to providing after-sale support to its customers. To this end, our customer-support department has established phone numbers that you can call if you have questions about this product.

Product Information. For information about Hewlett-Packard dealers, products, and prices, call the toll-free number below:

(800) FOR-HPPC
(800 367-4772)

Technical Assistance. For technical assistance with your product, call the number below:

(503) 754-6666

For either product information or technical assistance, you can also write to:

Hewlett-Packard Company
Personal Computer Group
Customer Communications
11000 Wolfe Road
Cupertino, CA 95014

Temperature Specifications

- Operating: 0° to 45°C (32° to 113°F)
- Storage: 0° to 45°C (32° to 113°F).

If the batteries are removed then the storage temperature tolerances for the HP-41 are:

−20° to 60°C (−4° to 140°F)

Potential for Radio and Television Interference (For U.S.A. Only)

The HP-41 generates and uses radio frequency energy and, if not installed and used properly, that is, in strict accordance with the manufacturer's instructions, may cause interference to radio and television reception. It has been type tested and found to comply with the limits for a Class B computing device in accordance with the specifications in Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference in a residential installation. However, there is no guarantee that interference will not occur in a particular installation. If your HP-41 does cause interference to radio or television reception, you are encouraged to try to correct the interference by one or more of the following measures:

- Reorient the receiving antenna.
- Relocate the computer with respect to the receiver.
- Move the computer away from the receiver.

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

Function Tables

Function Tables

Contents

Introduction

These ten tables describe the functions in the computer. Each table describes functions with common characteristics, and some functions appear in more than one table. Most tables include the information found in "Explanation of Table Entries".

Locating a Function

- To find a function that performs a particular operation, look through the function table whose title describes the desired type of operation.
- To find out what a function does when you know only its name, refer to the Function Index inside the back cover. The last page reference listed will direct you to the proper function table.

Explanation of Table Entries

Alpha Name. This is how the the function is named in catalog 2 or 3, in a program listing, and when you hold down a key for function preview. This is how you must specify the function to assign it to the User keyboard; if the function has no entry in this column, you can't assign it to the User keyboard.

Keyboard Name. This is how the function is indicated on the Normal or Alpha keyboard. (If the entry is printed in gold, you must press ■ before the appropriate key.) If the function has no entry in this column, you must use [XEQ] and the Alpha name or else assign the function to the User keyboard.

IND. An "I" in this column indicates that you can indirectly specify the parameter for this function. To do so, enter the function and press ■; **IND** will then appear in the display following the function name. Then specify the register holding the *address* of the register to access.

Stack. This shows how the function affects the automatic memory stack.

L = LAST X. The previous contents of the X-register are copied into the LAST X register.

↓ = The stack drops. The contents of the Z-register are copied into the Y-register and the contents of the T-register are copied into the Z-register.

↑ = The stack lifts. The contents of the X-, Y- and Z-registers are copied into the Y-, Z-, and T-registers respectively; the previous contents of the T-register are lost. (This assumes that stack lift was previously enabled.)

E = Stack lift enabled. If the next function executed shows "↑" in the "Stack" column or if you key in a number, the stack will lift. (Almost all functions enable stack lift.)

D = Stack lift disabled. If the next function executed shows "↑" in the "Stack" column or if you key in a number, the new number in the X-register replaces the previous contents and the stack doesn't lift. (Only [CLx], [ENTER↑], [Σ+], and [Σ−] disable stack lift.)

N = Neutral. Stack lift is neither enabled nor disabled; the previous status is maintained.

Flags. These are the flags that affect or are affected by the function's operation.

Bytes. This is the number of bytes of program memory required when the function is used in a program. If the function has no entry in this column, it is not programmable.

System/Format Functions

Most of these functions involve options that remain in effect indefinitely: display formats, angular mode, main memory allocation, User-keyboard assignments, and so on. Included are certain system operations such as the toggle keys and the catalogs.

Alpha Name	Keyboard Name	Description	IND	Stack	Flags	Bytes
	ALPHA	Activates/deactivates Alpha keyboard.			48	
AOFF		Deactivates Alpha keyboard.		E	48	1
AON		Activates Alpha keyboard.		E	48	1
ASN	ASN	Assigns specified function or global label to specified key on User keyboard.		E		
CAT *n*	CATALOG *n*	Executes catalog *n*, $1 \leqslant n \leqslant 3$.				
		Catalogs 1, 2, 3.		N		
CF *nn*	CF *nn*	Clears flag *nn*, $00 \leqslant nn \leqslant 29$.	I	E	*nn*	2
DEG		Selects decimal Degrees angular mode.		E	42-43	1
ENG *n*	ENG *n*	Selects engineering display format with $n + 1$ digits.	I	E	36-41	2
FIX *n*	FIX *n*	Selects fixed-point display format with *n* decimal places.	I	E	36-41	2
GRAD		Selects Grads angular mode.		E	42-43	1
	ON	Turns computer on/off.			11-26, 45-55	
ON		Selects continuous on (disables time-out).			44	
	PRGM	Enters, exits Program mode.		N		
RAD		Selects Radians angular mode.		E	42-43	1
SCI *n*	SCI *n*	Selects scientific display format with *n* decimal places.	I	E	36-41	2
SF *nn*	SF *nn*	Sets flag *nn*, $00 \leqslant nn \leqslant 29$.	I	E	*nn*	2
ΣREG *nn*		Assigns statistics registers to R_{nn} through R_{nn+5}.	I	E		2
SIZE *nnn*		Allocates *nnn* main memory registers for data storage.		E		
	USER	Activates/deactivates User keyboard.		N	27	

Clearing Functions

To interpret this table, refer to "Explanation of Table Entries" on page 106.

Alpha Name	Keyboard Name	Description	Stack	Flags	Bytes
	◆	When input cue (_) is displayed, clears last digit or character entered. When digit or character entry is terminated, clears X-register or Alpha register in Execution mode; deletes displayed program line in Program mode.	*		
		When message is displayed, clears message.	N	50	
	◆ down, ON, ◆ up	Clears all of computer's memory.		00-55	
CLA	CLA	Clears Alpha register.	E		1
CLD		Clears message from display.	E	50	1
CLP *label*		Clears the program in main memory containing specified global label.	E		
CLRG		Clears all data storage registers in main memory.	E		1
CLΣ	CLΣ	Clears statistics registers.	E		1
CLST		Clears automatic memory stack.	E		1
CLX	CLx	Clears X-register.	D		1
DEL *nnn*		Deletes *nnn* program lines, starting with displayed line.	N		

* When pressing ◆ clears the X-register, stack lift is disabled. Otherwise, ◆ is neutral.

Stack/Data Register Functions

These functions manipulate the stack or the data storage registers, or take one of those registers as a parameter. To interpret this table, refer to "Explanation of Table Entries" on page 106.

Alpha Name	Keyboard Name	Description	IND	Stack	Flags	Bytes
ARCL *nn*	ARCL *nn*	Appends contents of R_{nn} to Alpha register.	I	E	28, 29, 36-41	2
ASTO *nn*	ASTO *nn*	Copies six leftmost characters in Alpha register into R_{nn}.	I	E		2
CLRG		Clears all data storage registers.		E		1
CLΣ	CLΣ	Clears statistics registers		E		1
CLST		Clears automatic memory stack.		E		1
CLX	CLx	Clears X-register.		D		1
DSE *nn*		For *iiiii.fffcc* in R_{nn}, decrements *iiiii* by *cc* and skips next program line if $iiiii - cc \leqslant fff$.	I	E		2
ENTER↑	ENTER↑	Copies number in X-register into Y-register and lifts stack.		↑, D		1
ISG *nn*	ISG *nn*	For *iiiii.fffcc* in R_{nn}, increments *iiiii* by *cc* and skips next program step if $iiiii + cc > fff$.	I	E		2
LASTX	LASTx	Recalls number in LAST X register.		↑, E		1
R↑		Rolls up stack.		E		1
RCL *nn*	RCL *nn*	Recalls contents of R_{nn}.	I	↑, E		*
RDN	R↓	Rolls down stack.		E		1
Σ+	Σ+	Accumulations for statistics.		L,D		1
Σ−	Σ−	Corrects statistics accumulations.		L,D		1
ΣREG *nn*		Assigns statistics registers to R_{nn} through R_{nn+5}.	I	E		2
SIZE *nnn*		Allocates *nnn* main memory registers for data storage.		E		

* If $00 \leqslant nn \leqslant 15$, requires 1 byte; otherwise, requires 2 bytes.

Stack/Data Register Functions (coninued)

Alpha Name	Keyboard Name	Description	IND	Stack	Flags	Bytes
ST+ *nn*	STO + *nn*	Adds number in X-register to number in R_{nn} and places result in R_{nn}.	I	E		2
ST− *nn*	STO − *nn*	Subtracts number in X-register from number in R_{nn} and places result in R_{nn}.	I	E		2
ST* *nn*	STO × *nn*	Multiplies number in X-register by number in R_{nn} and places result in R_{nn}.	I	E		2
ST/ *nn*	STO ÷ *nn*	Divides number in X-register into number in R_{nn} and places result in R_{nn}.	I	E		2
STO *nn*	STO *nn*	Copies contents of X-register into R_{nn}.	I	E		*
VIEW *nn*	VIEW *nn*	Displays contents of R_{nn}.	I	E	21,50, 55	2
X<> *nn*		Exchanges contents of X-register with contents of R_{nn}.	I	E		2
X<>Y	x≷y	Exchanges contents of X-register with contents of Y-register.		E		1

* If 00 ⩽ *nn* ⩽ 15, requires 1 byte; otherwise, requires 2 bytes.

Numeric Functions

All numeric functions are programmable, requiring one byte of program memory. The operation of trigonometric functions and rectangular/polar coordinate conversions depends on the angular mode (flags 42 and 43). To interpret this table, refer to "Explanation of Table Entries" on page 106.

Alpha Name	Keyboard Name	Description	Stack
+	+	$y + x$.	L,↓,E
−	−	$y - x$.	L,↓,E
*	×	$y \times x$.	L,↓,E
/	÷	y / x.	L,↓,E
1/X	1/x	Reciprocal.	L,E
10↑X	10^x	Common exponential.	L,E
ABS		$\lvert x \rvert$ (Absolute value).	L,E
ACOS	COS^{-1}	Arc (inverse) cosine.	L,E
ASIN	SIN^{-1}	Arc (inverse) sine.	L,E
ATAN	TAN^{-1}	Arc (inverse) tangent.	L,E
CHS	CHS	Change sign.	E
COS	COS	Cosine.	L,E
D−R		Degrees to radians conversion.	L,E
DEC		Octal to decimal conversion.	L,E
E↑X	e^x	Natural exponential.	L,E
E↑X−1		Natural exponential for arguments close to zero.	L,E
FACT		$x!$ (Factorial).	L,E
FRC		Fractional part.	L,E
HMS		Decimal hours to hours-minutes-seconds conversion.	L,E
HMS+		Hours-minutes-seconds add.	L,↓,E
HMS−		Hours-minutes-seconds subtract.	L,↓,E
HR		Hours-minutes-seconds to decimal hours conversion.	L,E
INT		Integer part.	L,E
LN	LN	Natural logarithm.	L,E
LN1+X		Natural logarithm for arguments close to 1.	L,E

Numeric Functions (continued)

Alpha Name	Keyboard Name	Description	Stack
LOG	LOG	Common logarithm.	L,E
MEAN		Means of accumulated *x*- and *y*-values.	L,E
MOD		*y* mod *x* (Remainder).	L,↓,E
OCT		Decimal to octal conversion.	L,E
P–R	P→R	Polar to rectangular conversion.	L,E
%	%	*x* percent of *y*.	L,E
%CH		Percent change from *y* to *x*.	L,E
PI	π	Pi (3.141592654).	↑,E
R–D		Radians to degrees conversion.	L,E
R–P	R→P	Rectangular to polar conversion.	L,E
RND		Round.	L,E
SDEV		Standard deviations of accumulated *x*- and *y*-values.	L,E
Σ+	Σ+	Accumulations for statistics.	L,D
Σ–	Σ–	Accumulations correction.	L,D
SIN	SIN	Sine.	L,E
SIGN		Sign of *x*.	L,E
SQRT	$\sqrt{x}$	Square root.	L,E
TAN	TAN	Tangent.	L,E
X↑2	x^2	Square.	L,E
Y↑X	y^x	*y* raised to the *x* power.	L,↓,E

Editing Functions

These are non-programmable functions that are executed in Program mode. They help you write or edit your programs. Like the toggle keys [ON], [USER], and [ALPHA], these functions don't require you to return to Execution mode for execution. To interpret this table, refer to "Explanation of Table Entries" on page 106.

Alpha Name	Keyboard Name	Description	Flags
	[←]	When input cue (_) is displayed, clears last digit or character entered; otherwise, clears displayed program line.	
[ASN]	[ASN]	Assigns specified function or global label to specified key on User keyboard.	
[BST]	[BST]	Displays preceding program line.	
[CAT] *n*	[CATALOG] *n*	Executes catalog *n*, $1 \leqslant n \leqslant 3$.	
[CLP] *label*		Clears program in main memory containing specified global label.	
[COPY] *label*		Copies ROM program containing specified global label to program memory.	
[DEL] *nnn*		Deletes *nnn* program lines, starting with displayed line.	
	[GTO] [.]	Goes to specified line number or global label.	
	[GTO] [.] [.]	Goes to bottom of program memory; packs program memory and creates null program.	
[ON]		Selects continuous on (disables time-out).	44
[PACK]		Packs program memory.	
[SIZE] *nnn*		Allocates *nnn* main memory registers for data storage.	
[SST]	[SST]	Displays next program line.	

Functions That Direct Program Execution

These are functions that can halt program execution or cause program lines to be executed other than sequentially. To interpret this table, refer to "Explanation of Table Entries" on page 106.

Alpha Name	Keyboard Name	Description	IND	Stack	Flags	Bytes
AVIEW	AVIEW	Displays contents of Alpha register; if flag 21 is set and flag 55 is clear, stops program execution.		E	21, 50, 55	1
DSE *nn*		For *iiiii.fffcc* in R_{nn}, decrements *iiiii* by *cc* and skips next program line if $iiiii - cc \leqslant fff$.	I	E		2
END		Marks end of program.		E		3
FC? *nn*		Tests flag *nn* ($00 \leqslant nn \leqslant 55$) and skips next program line unless flag *nn* is clear.	I	E	*nn*	2
FC?C *nn*		Tests flag *nn* ($00 \leqslant nn \leqslant 29$), clears flag *nn*, and then skips next program line unless flag *nn* was clear.	I	E	*nn*	2
FS? *nn*	FS? *nn*	Tests flag *nn* ($00 \leqslant nn \leqslant 55$) and skips next program line unless flag *nn* is set.	I	E	*nn*	2
FS?C *nn*		Tests flag *nn* ($00 \leqslant nn \leqslant 29$), clears flag *nn*, and then skips next program line unless flag was set.	I	E	*nn*	2
GTO *label*	GTO *label*	Transfers execution to specified global, numeric, or local Alpha label.	I	E		*
ISG *nn*	ISG *nn*	For *iiiii.fffcc* in R_{nn}, increments *iiiii* by *cc* and skips next program step if $iiiii + cc > fff$.	I	E		2
LBL	LBL	Global, numeric, or local Alpha label.		E		†
OFF		Turns off the computer.		N	11-26 44-55	1

* If $00 \leqslant nn \leqslant 14$ or parameter is indirectly specified, requires 2 bytes; if parameter is global label of *m* characters, requires $2 + m$ bytes; otherwise, requires 3 bytes.

† If $00 \leqslant nn \leqslant 14$, requires 1 byte; if parameter is global label of *m* characters, requires $4 + m$ bytes; otherwise, requires 2 bytes.

Functions That Direct Program Execution (continued)

Alpha Name	Keyboard Name	Description	IND	Stack	Flags	Bytes
PROMPT		Displays contents of the Alpha register and stops execution.		E	50	1
RTN	RTN	Returns execution to line following XEQ instruction that called this subroutine.		E		1
STOP	R/S	Stops execution.		E		1
VIEW *nn*	VIEW *nn*	Displays contents of R_{nn} and, if flag 21 is set and flag 55 is clear, stops execution.	I	E	21, 50, 55	2
X = 0?	$x = 0?$	Skips next instruction unless number in X-register = 0.		E		1
X ≠ 0?		Skips next instruction unless number in X-register ≠ 0.		E		1
X < 0?		Skips next instruction unless number in X-register < 0.		E		1
X <= 0?		Skips next instruction unless number in X-register ≤ 0.		E		1
X > 0?		Skips next instruction unless number in X-register > 0.		E		1
X = Y?	$x = y?$	Skips next instruction unless contents of X-register = contents of Y-register.		E		1
X ≠ Y?		Skips next instruction unless contents of X-register ≠ contents of Y-register.		E		1
X < Y?		Skips next instruction unless number in X-register < number in Y-register.		E		1
X <= Y?	$x \leq y?$	Skips next instruction unless number in X-register ≤ number in Y-register.		E		1

Functions That Direct Program Execution (continued)

Alpha Name	Keyboard Name	Description	IND	Stack	Flags	Bytes
X > Y?	x > y?	Skips next instruction unless number in X-register > number in Y-register.		E		1
XEQ *label*	XEQ *label*	Calls specified global, numeric, or local Alpha label as subroutine. (If you specify a function, refer to the table entry for that function.)	I	E		*

* If label is specified indirectly, requires 2 bytes; if local label is specified, requires 3 bytes; if global label of *m* characters is specified, requires 2 + *m* bytes.

Alpha Functions

These functions involve moving data into and out of the Alpha register, and manipulating the data in the Alpha register. Not included are functions that use the Alpha register for a file name. To interpret this table, refer to "Explanation of Table Entries" on page 106.

Alpha Name	Keyboard Name	Description	IND	Stack	Flags	Bytes
	⊢	Appends subsequent characters to Alpha register.		N		
AOFF		Deactivates Alpha keyboard.		E	48	1
AON		Activates Alpha keyboard.		E	48	1
ARCL *nn*	ARCL *nn*	Appends contents of R_{nn} to Alpha register.	I	E	28, 29, 36-41	2
ASHF		Shifts six leftmost characters out of the Alpha register.		E		1
ASTO *nn*	ASTO *nn*	Copies six leftmost characters in Alpha register into R_{nn}.	I	E		2
AVIEW	AVIEW	Displays contents of Alpha register.		E	21, 50, 55	1
CLA	CLA	Clears Alpha register.		E		1
PROMPT		Displays contents of Alpha register and stops program execution.		E	21, 50, 55	1

Interactive Functions

To interpret this table, refer to “Explanation of Table Entries” on page 106.

Alpha Name	Keyboard Name	Description	IND	Stack	Flags	Bytes
ADV		Advances paper (if printer is present).		E	21,55	1
BEEP	BEEP	Sounds four tones.		E	26	1
PROMPT		Displays contents of Alpha register and stops execution.		E	50	1
PSE		Delays execution for about one second.		E		1
TONE *n*		Sounds tone *n*, $0 \leqslant n \leqslant 9$.	I	E	26	2

Indexes

Subject Index

Page numbers in **bold** type indicate primary references; page numbers in regular type indicate secondary references.

R

S

T

U

V

W

X

Y

Z

Function Index

For each function, its Alpha name is given first (in blue), and its keyboard name follows (in black or gold), although not all functions have both an Alpha name and a keyboard name. (These conventions are explained on the inside of the front cover.) The page reference in normal type indicates the text; the **boldface** reference indicates the function tables.

Summary of Conventions Used in This Manual

Notation (Example)	Description
STO	*Black keybox.* Primary keyboard function.
10^x	*Gold keybox.* Shifted keyboard function. Press and release the shift key ( ) first. These can be on the Normal or Alpha keyboard.
DATE	*Blue keybox.* Nonkeyboard function. For Alpha execution: use XEQ followed by the Alpha name spelled out on the Alpha keyboard. For User-key execution: assign the function to the User keyboard. (Pages 45, 46.)
ABC	*Blue letters.* Alpha characters.
123	*Gold digits or characters.* Shifted Alpha characters.
X · T	*Black letters in keyboxes.* These are special functions, *not* Alpha characters, and are active only in special circumstances. If it is a shifted function, it is preceded by .
parameter	*The type of parameter* required for a function.

For a full description, refer to "How This Manual Represents Keystrokes," page 14.

HEWLETT PACKARD

Portable Computer Division
1000 N.E. Circle Blvd., Corvallis, OR 97330, U.S.A.

European Headquarters
150, Route du Nant-D'Avril
P.O. Box, CH-1217 Meyrin 2
Geneva-Switzerland

HP-United Kingdom
(Pinewood)
GB-Nine Mile Ride, Wokingham
Berkshire RG11 3LL

00041-90526 English

Printed in Singapore 5/84