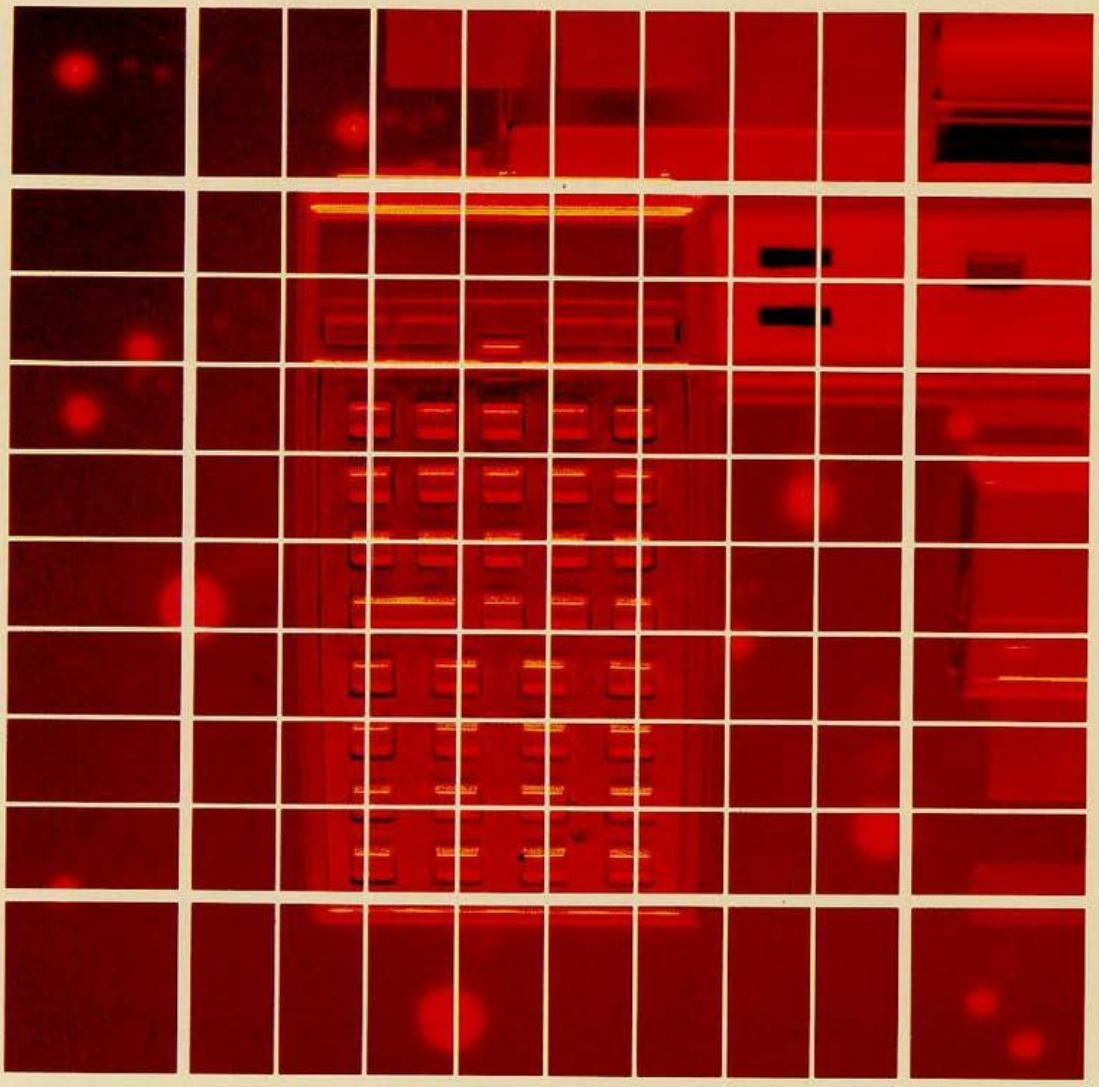
HEWLETT-PACKARD

HP-41CX
OWNER'S MANUAL

VOLUME 1: BASIC OPERATION



Summary of Conventions Used in This Manual

Notation (Example)	Description
[STO]	*Black keybox.* Primary keyboard function.
[10ˣ]	*Gold keybox.* Shifted keyboard function. Press and release the shift key ([f]) 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.
[T] [f] [T] [g] [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 [f].
parameter	*The type of parameter* required for a function.

For a full description, refer to "How This Manual Represents Keystrokes," page 16.

HEWLETT PACKARD

HP-41CX
Owner's Manual

Volume 1
Basic Operation

August 1983

00041-90474

Printed in Singapore

Introducing the HP-41CX

The HP-41CX is an advanced model of the HP-41 family of compact, handheld computers. It combines a rich instruction set (over 200 built-in functions) with 319 registers of main memory for data storage, program lines, alarms, and user-defined keys, and 124 registers of extended memory for storing data files, text files, and program files.

The HP-41CX features:

- A powerful set of numeric and mathematical functions.
- Time, calendar, alarm, and stopwatch functions.
- Program, data, and text file operations in extended memory, including an easy-to-use text editor.
- An alphabetic character set for creating messages.
- 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-41CX 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/CV (including those in plug-in modules) can be used with the HP-41CX without modification. Programs written for the HP-41CX, however, will not necessarily run on the HP-41C/CV, since the HP-41CX has so many more functions.

Volume 1: Basic Operation

Contents

List of Diagrams and Tables

Contents of Volume 2: Operation in Detail

Part II: Fundamentals In Detail

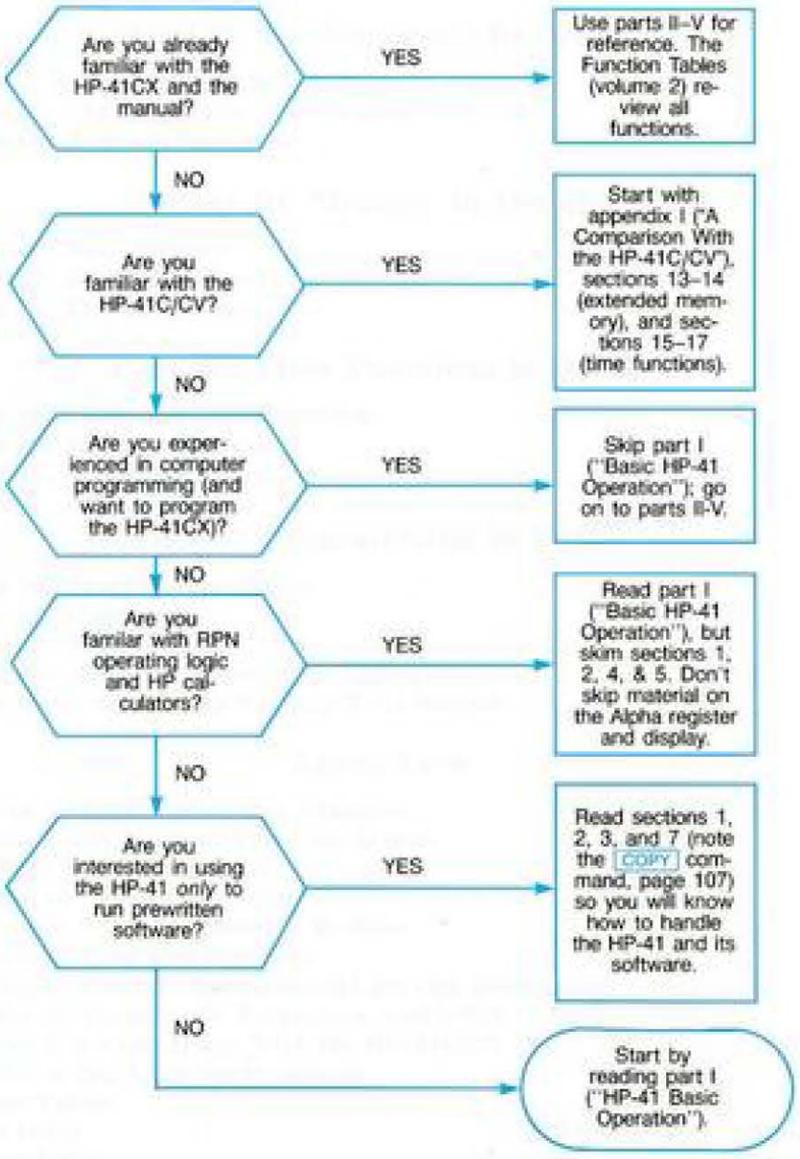
Part III: Memory in Detail

Part IV: Time Functions in Detail

Part V: Programming in Detail

Appendices

Where to Start



How to Use This Manual

The HP-41 is a powerful handheld computer with a broad range of functions.* Each person comes to the HP-41 with different problems to solve, and each person has a different level of experience with computers. The *HP-41CX Owner's Manual* was written to accommodate these differences: the two volumes of the manual address different topics *as well as* different levels of experience and need. We recommend that you start reading volume 1 (part I), "Basic Operation", and read or skim as much as is not familiar to you. This will teach you how to use the HP-41. Then, when you want to explore a specific topic or problem, turn to volume 2, "Operation in Detail", containing parts II through V.

- Volume 1 (part I) is an introduction to most aspects of operation. It does not cover everything there is to know about the HP-41, because you don't need to know everything to start. (In fact, you might not ever need to know everything.) This simplifies learning to use the HP-41, and then you can decide later if you want to learn more about a particular subject.
- Volume 2 (parts II through V) comprehensively covers operation in a detailed manner. This is good for referencing a specific operation or topic in depth.
- The Function Tables at the back of volume 2 (the blue-edged pages) provide a complete and easy-to-reference summary of all HP-41 functions.
- Both volumes have a function index on the inside of the back cover. This lists every function in the HP-41CX and its main reference in volume 1, volume 2, and the Function Tables.
- Appendix I, "A Comparison With the HP-41C/CV," is for those people already familiar with the HP-41C/CV. It outlines the features in the HP-41CX that are different, and refers to where those features are discussed in the text.

For First-Time Users

Volume 1, "Basic Operation," is the place to start if you are not familiar with the HP-41 or other computers. This contains part I, "Basic HP-41 Operation", which is an introduction to HP-41 logic, keyboard and display layout, standard functions, time functions, basic programming, and text and program files. This will teach 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 part I, you should feel well acquainted and comfortable with the HP-41. This volume also includes a List of Errors, a Subject Index, and a Function Index.

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

As your understanding and use of the HP-41 increase, you might want to know more about its operation, including those functions not discussed in volume 1. Then it's time to turn to volume 2.

For More Experienced Users

If you are an experienced computer user, you can skip part I and start with volume 2, "Operation in Detail", which contains parts II through V and the appendices. Part II, "Fundamentals in Detail," part III, "Memory in Detail," part IV, "Programming in Detail," and part V, "Time Functions in Detail," comprehensively cover the use and operation of the HP-41 computer. Once you are familiar with the HP-41, you can refer to these parts to find out more about a topic.

Some of the topics in these parts duplicate topics in volume 1, but they are discussed in much more detail. Other topics—the memory stack, control alarms, many aspects of files, and advanced programming—are not discussed in volume 1.

If you are already familiar with the HP-41C/CV, the first thing you should read is appendix I, "A Comparison With the HP-41C/CV." This also discusses the new terminology used in this manual. The new conventions for notation are summarized on the inside of the front covers in both volumes.

Reference

For reference use, this manual includes a comprehensive subject index (at the very back of both volumes), an alphabetical function index (on the inside of the back cover in both volumes), a comprehensive set of function tables (the blue-edged pages preceding the subject index in volume 2), a summary of the notation conventions used in this manual (inside the front cover of both volumes), and extensive appendix information in volume 2—including error conditions (appendix A), printer considerations (appendix D), time specifications (appendix F), and a comparison of the HP-41CX with the HP-41C/CV (appendix I).

Part I
Basic HP-41 Operation

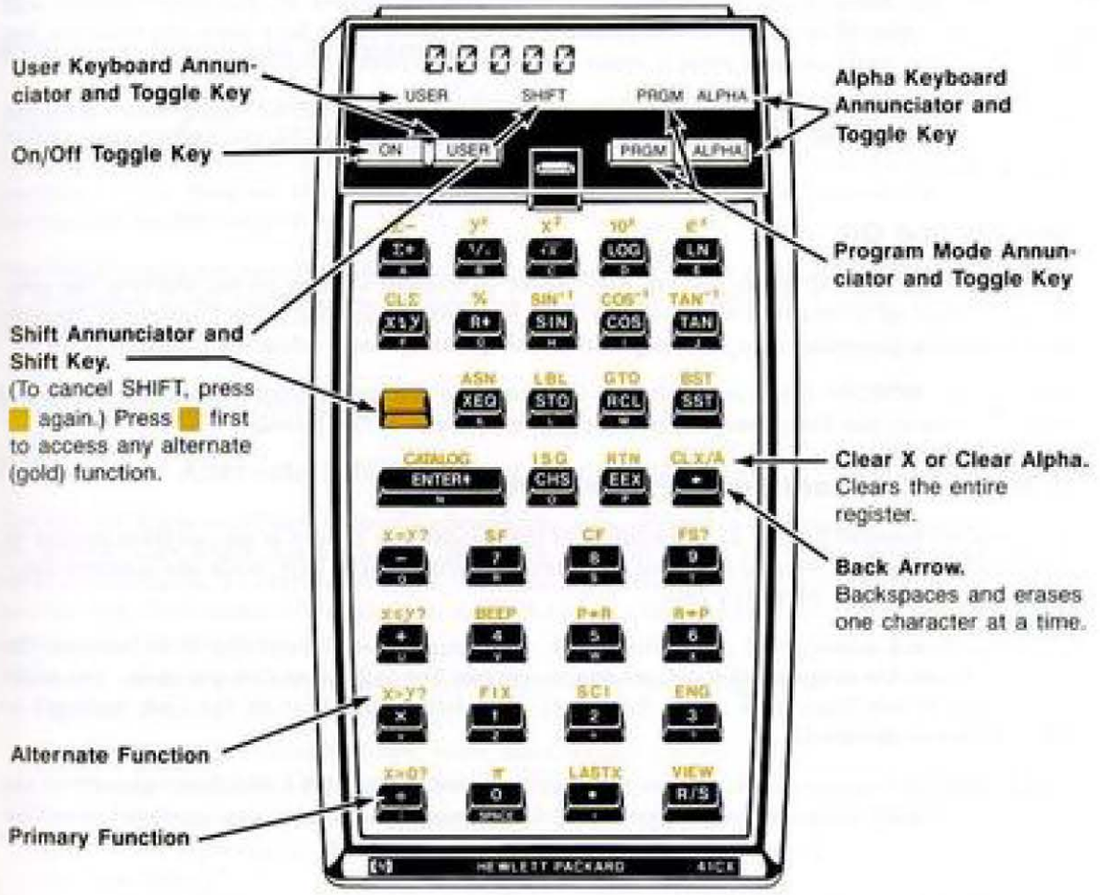
Section 1

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 25, 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 on page 24.)

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 7.)

Keyboard Conventions

Primary and Alternate (Shifted) Keyboard Functions

The HP-41CX has six different sets of keyboard functions, referred to as *keyboards*. These keyboards—the Normal, User, Alpha, Alarm Catalog, Stopwatch, and Text Editor 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 25 and in the Quick Reference Guide. Most of the Alpha keyboard functions are shown on the Text Editor keyboard and 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 the Alarm Catalog function [T] (for *time*).

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: [T] for alarm time.

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 19.) 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.
[√x]	6.7082	[√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'orech*) (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	
[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 part II, section 10, "The Automatic Memory Stack." 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 7.

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 ←.

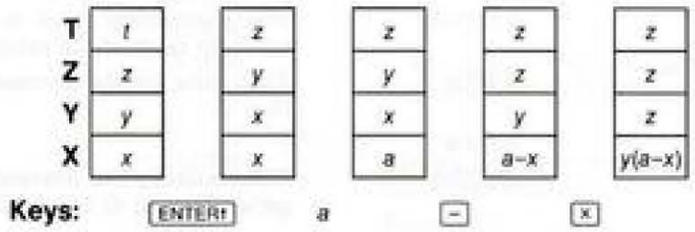
Clearing MEMORY LOST and Error Messages. If MEMORY LOST or any other error message appears and remains in the display, you can clear it by pressing ← or any key (except ■ and USER). ← will restore the previous display; any other key will execute its own function. (Error messages and their meanings are covered on page 34 in section 2, "The Display.")

Doing Longer Calculations

The use of RPN (page 17) and the "automatic memory stack" (section 10) 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.)



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 part II, section 10 ("The Automatic Memory Stack") 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 ÷ [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 ÷ (−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 ÷ [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 ÷ (−61).

Using Constants in Arithmetic Calculations

The LAST X register is a register related to the memory stack. It is described in section 10, "The Automatic Memory Stack." 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 $\frac{96.704 + 52.394706}{52.394706}$.

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 Rigil 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 Rigil 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 34.)

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.

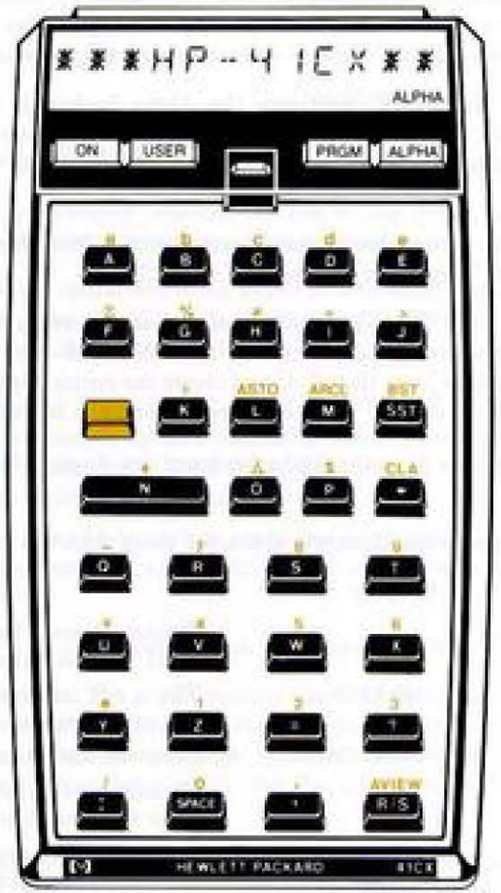
Entering Alphabetic Characters

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 14 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," page 55.

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 [CLΣ] 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-41CV	HP-41CV_	The _ cue indicates that character entry is not yet terminated.
[←] X	HP-41CX_	Removes last character; adds X.
[ALPHA]		Terminates character entry and restores the prior numeric display.
[ALPHA]	HP-41CX	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.

* [CLΣ], the Normal keyboard function, and [CLA], the Alpha keyboard function, are printed on the keyboard in combined form as [CLΣ/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-41CX_	Notice the _ cue reappears. This signals that the next entry will be appended to the previous one.
/CV	HP-41CX/CV_	
[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.)

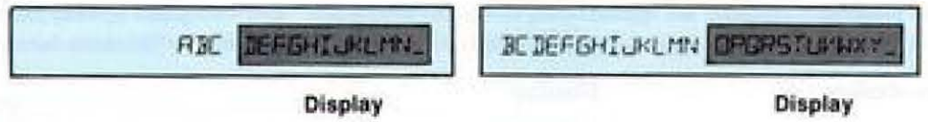
Alpha entries are ***not*** *put into the Alpha register in the following two important conditions:*

- 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.



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).
- Time and date settings.
- Alarm settings.
- The current file and the current position in that file (extended memory).

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 G 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.

This will not reset the time. (For a summary of other effects, refer to "Clearing Memory" in appendix G.)

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 19-20.)

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.)

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.

---

* "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. Refer to section 11. 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.

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).

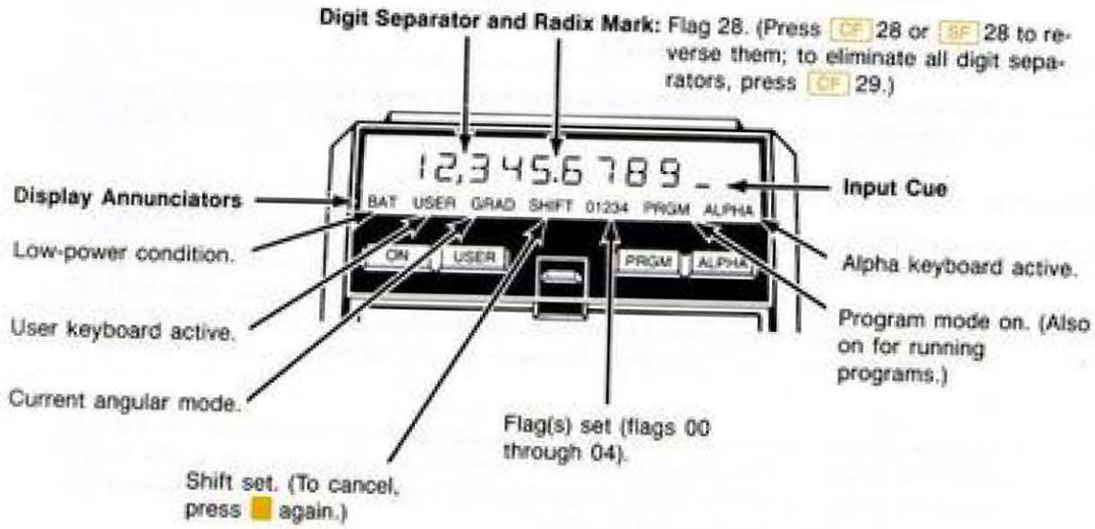
Note: In program instruction lines and in printer listings, numbers with exponents are printed with an E before the exponent. For example, **1.234568 02** would look like **1.234568 E02**. If you are trying to copy or enter such a line yourself, just use the EEX key, as explained on page 18. Do not try 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 in this volume:

BAT	Low power indication, page 34.
USER	User keyboard, pages 14 and 46.
GRAD and **RAD**	Angular modes, page 53.
SHIFT	Shift for alternate functions, page 15.
0 1 2 3 4	Flags 00, 01, 02, 03, 04. Refer to part III.
PRGM	Program mode or running program, pages 15 and 84.
ALPHA	Alpha keyboard, pages 14 and 24.

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 G 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 brief list of error messages at the back of this volume (page 130), before the Subject Index. Volume 2 has a complete list of error messages and their causes in appendix A, "Error and Status Messages."

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 29) 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 part V ("Programming in Detail"), sections 19 and 20.

Aspects of printer formatting controlled by flags are in appendix D, "Printer Operation."

Section 3

Storing and Recalling Numbers

Contents

The HP-41 stores information—data, Alpha strings, User keyboard function assignments, alarms, 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.

(The HP-41CX also has an *extended* memory for special text, data, and program storage. These registers are not explained here. Extended memory is covered in section 8 and in part III, section 13.)

Storage Registers and Computer Memory

Main Memory Distribution

When you first turn the computer on or when you reset Continuous Memory (page 29), there are:

100 *data storage* registers (numbered R_{00} to R_{99}), which are directly accessible.*

219 *uncommitted* registers available in a common pool for programs, alarms, and User-function assignments.

* If you have more than 100 data storage register, those above R_{99} are only *indirectly accessible*. This is explained in section 12.

These registers are convertible, so, for instance, you can allocate up to 319 registers for data storage. The means of changing the allocation of memory (and accessing data registers with numbers greater than 99) are discussed in part III, section 12, "Main Memory." You don't need to worry about reallocating memory unless you run out of available registers.

The figure on page 195 in section 12 is a representation of the main memory of the HP-41, showing the distribution of registers between data storage and the uncommitted register pool.

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.)
- You can clear a specific block of registers by entering the six-digit number *bbb.eee* and executing CLRGX. The portion *bbb* (*beginning*) is the address of the first storage register you want cleared, and *eee* (*ending*) is the address of the last register to be cleared. All registers in sequence from *bbb* through *eee* will be cleared.

Keystrokes	Display	
XEQ	XEQ __	Execute what?
ALPHA CLRG ALPHA		Clears all data registers (this is R_{00} to R_{99} 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 (see also page 22).

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

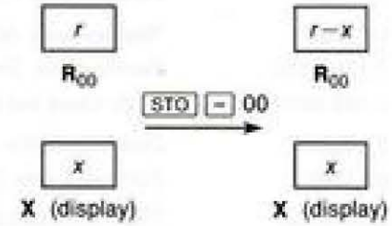
Suppose you not only wanted to store a number, but perform an arithmetic operation with it and store the result in the same register. You can do this directly—without using [RCL]—by doing storage arithmetic, as follows:

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:

	55 (25 + 27 + 19 + 23) − 2% [55 (25 + 27 + 19 + 23)]	Monday and Tuesday
+	57.5 (26 + 28) − 3% [57.5 (26 + 28)]	Wednesday
	Total Net Income	

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 $9.99999999 \times 10^{99}$ 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 24.

Section 4

How to Execute HP-41 Functions

Contents

You might not realize it, but the HP-41CX has over 200 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 46-48.)

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 volume and the Function Tables at the back of volume 2. 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: [√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 27, 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.†‡

* You can write a program to assign and re-assign User-function definitions. Refer to the use of PASN (*programmable assign*) in section 9.

† 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. Refer to the diagram on page 167.

‡ 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 36.

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

Cancelling One Redefined Key. 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).

Alternatively, you can use catalog 6 (CATALOG 6), as described on page 48.*

Cancelling All Redefined Keys. Execute CLKEYS (*clear keys*) to clear all User-key definitions in main memory.

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.

* The HP-41CX has six different catalogs. They are summarized together in section 9.

† 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

Viewing Individual Function Assignments (Function Preview). 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 ←.

Viewing All User-Function Assignments (CATALOG 6). Pressing CATALOG 6 will list, one at a time, the names and keycodes of all function assignments you have made. You can stop and restart the listing with the R/S (*run/stop*) key.

Once the listing has been stopped, you can move forward in the catalog one step at a time by pressing SST (*single step*), or backwards one step by pressing BST (*back step*). Pressing ■ C cancels the currently displayed user-function assignment.

> 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) and the Function Tables (at the back of volume 2).

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	x²	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 x²	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 16.)

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
$\sqrt{2}$ or $2^{1/2}$	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

For ease of calculation, the HP-41 works with degrees with decimal fractions. However, you can translate between decimal degrees and degrees-minutes-seconds using the time conversion functions HMS (*to hours-minutes-seconds*) and HR (*to hours*). Refer to "Converting Between Forms for Hours and Degrees," section 6, page 65.

In addition, the time functions HMS+ (*hours-minutes-seconds add*) and HMS− (*hours-minutes-seconds subtract*) will add and subtract degrees in the degrees-minutes-seconds format. Refer to "Adding and Subtracting Time Values," page 65.

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

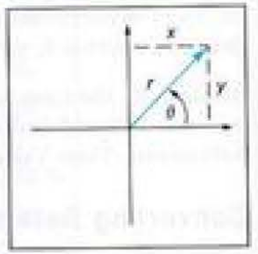
To Calculate	Press	Or Execute
Sine of x	SIN	SIN
Cosine of x	COS	COS
Tangent of x	TAN	TAN
Arc sine of x	SIN^{-1}	ASIN
Arc cosine of x	COS^{-1}	ACOS
Arc tangent of x	TAN^{-1}	ATAN

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.

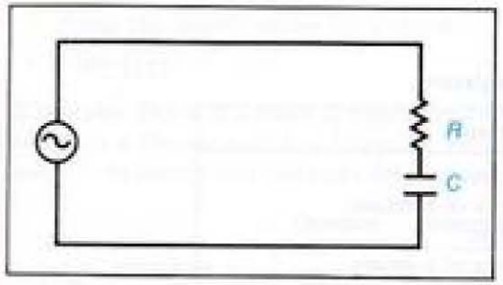
To Convert	Key In	Press	Result
(x,y) to (r,θ)	y-value	ENTER↑	
	x-value	R→P	r-value
		x⇄y	θ-value
(r,θ) to (x,y)	θ-value	ENTER↑	
	r-value	P→R	x-value
		x⇄y	y-value

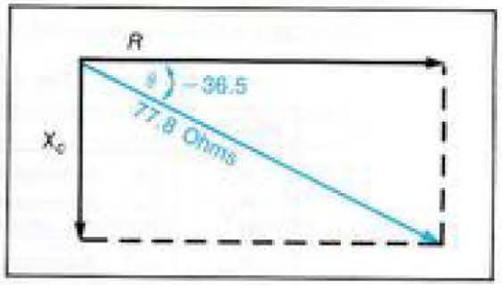
Both R→P and P→R require two numeric inputs and return you two outputs. For R→P, be sure to enter y before x. The resulting display will show r; press x⇄y to see θ.

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 formulas used are shown in section 11. 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.* The formulas used are shown in section 11.

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]. Refer also to section 11.

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 109 in section 7.

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

The Time Functions

Contents

The HP-41CX contains an internal timer, allowing you to use the computer as a clock, a calendar, an alarm clock, a stopwatch, and a time-based system controller (with time-controlled program execution). This section covers most of the clock and calendar functions of the HP-41, as well as basic alarm and stopwatch operation. A more complete and detailed discussion of alarm and stopwatch capabilities is in part IV, "Time Functions in Detail".

The Clock

Displaying the Clock (ON , CLOCK)

- Press ON or execute CLOCK to display the clock time.
- Press ← to cancel the time display and return to the X-register.
- As long as the clock time is displayed, the HP-41 *will not automatically turn off.*

Note: The clock and stopwatch displays consume more than normal battery power. These displays will not operate if the **BAT** annunciator is lit. Refer to appendix G, "Battery, Warranty, and Service Information."

The clock display is a type of message display—it cannot be used as a numeric value for calculations.

Clock Display Format (CLKT , CLKTD , CLK12 , CLK24)

The clock can display either the time alone or the time and date. The time can be in either a 12-hour or a 24-hour format. At initial turn-on, the clock shows only the time, and it uses a 12-hour display format.

Note: These formats affect the clock display only, and not the format you use for entering time values (time input), nor the format used for the numeric results of time functions (time output). (See the chart on page 63.)

- CLKT (*clock time*) sets the time-only clock format.
- CLKTD (*clock time and date*) sets the time/date clock format.
- CLK12 (*clock, 12 hour*) sets a 12-hour clock display.
- CLK24 (*clock, 24 hour*) sets a 24-hour clock display.

Clock Display Formats

Format	CLKT	CLKTD
CLK12	*(H)H:MM:SS* **AM** *(H)H:MM:SS* **PM**	*(H)H:MM* **AM** *date* *(H)H:MM* **PM** *date*
CLK24	*HH:MM:SS*	*HH:MM date*

Date Format (MDY, DMY)

When the date is displayed—whether by itself or with the time (in the clock display)—it can appear with the month first or the day first. At initial turn-on or memory reset, the format is month/day/year.

- MDY (*month/day/year*) sets the date format to month/day/year.
- DMY (*day/month/year*) sets the date format to day/month/year.

The date format (unlike the clock display format) affects all date inputs (what you enter), outputs (results), and displays.

Date Formats

Setting	Display and Printer	Numeric Value for Date
MDY	*MM/DD* or *MM/DD/YY*	*(M)M.DDYYYY*
DMY	*DD.MM* or *DD.MM.YY*	*(D)D.MMYYYY*

Setting the Date (SETDATE)

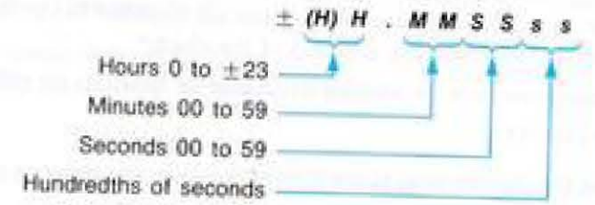
Enter the numeric value you wish to set as the current date, then execute SETDATE.

To set the date, key in either ***MM.DDYYYY*** or ***DD.MMYYYY***, as appropriate. You can omit any leading and trailing zeros. Any date from January 1, 1900 to December 31, 2199 is valid.

	MDY Format	DMY Format
Numeric Value for Date	5 . 1 0 1 9 9	1 0 . 0 5 1 9 9
May 10, 1990	Month Day Year	Day Month Year

Setting the Time (SETIME)

SETIME will set the clock time using the time you specify. Use the following format to input clock time (regardless of the clock display format):



The time input can be any ***HH.MMSSss*** value from −23.595999 to 23.595999, and you can omit leading and trailing zeros.

The table to the right shows how negative numbers can be used for p.m. input.

(The **ss** values will not show in the clock display; they are used for other time calculations.)

1. Specify the desired time in the display, according to the ***±HH.MMSSss*** format shown here.
2. Execute SETIME.

Example: To set the time to 3:30:10 (a.m.), and then to 3:30:10 p.m. (15.30.10):

Keystrokes	Display	
3.301 SETIME	3.3010	Sets the time to 3:30:10 a.m.
15.301 SETIME or 3.301 CHS SETIME	15.3010 -3.3010	Sets the time to 3:30:10 p.m. (=15:30:10).
CLOCK (or ■ ON)	3:30:++ PM	Clock time in 12-hour, time-only format.*

Setting Time

Setting	Clock Time
0	Midnight
1	1 (a.m.)
2	2 (a.m.)
⋮	⋮
11	11 (a.m.)
±12	Noon
±13 or −1	13 = 1 p.m.
±14 or −2	14 = 2 p.m.
⋮	⋮
±23 or −11	23 = 11 p.m.

*The ++ symbols represent continuously changing digits.

Adjusting the Clock Time (T+X)

Once you have set the time, you shouldn't have to use SETIME again. (The time is not affected when Continuous Memory is cleared.) Should you need to adjust the time due to a time change or your own lack of precision in setting the time, use T+X (*time plus X*). This function increments (positive number) or decrements (negative number) the time by the amount specified in the display (X-register). You should not use this function to correct the accuracy of the clock.*

1. Enter the hours-minutes-seconds (as necessary) change in time, up to ±***HHHH.MMSSss***.
2. Execute T+X (*time plus X*).

If the time change moves the current time to a different day than the previous clock time, the date will also change.

Example: Adjust for a 1.75 second slow timesetting error. Then decrement the current clock time by 1 hour for a time zone change.

Keystrokes	Display	
.000175	.000175_	
T+X	0.0002	Increments the clock time by 1.75 seconds. (The display has been rounded to four places, but this does not affect the calculation.†)
1 CHS	-1_	
T+X	-1.0000	Decrements the clock time by 1 hour.

Calculations With Time Values

Recalling a Discrete Time Value (TIME)

If you need a discrete time value, as for a calculation or for use by a program, then you can recall a numeric value for the time by executing TIME. (Do not use the CLOCK function or ■ ON.)

- During the execution of a program, TIME will place the 24-hour clock time in the X-register in the format ***HH.MMSSss***. (Midnight is zero.)
- If TIME is executed manually (that is, not in a program), you will see a message display showing the time. This display is like the clock display, and is given in ***HH:MM:SS*** format, using the CLK12 or CLK24 format in effect. Pressing ← erases the message and displays the time in the X-register, using the same time format shown in the preceding paragraph.

To see all six decimal places in the X-register, press FIX 6.

* The CORRECT function described in appendix F, "Time Specifications," will correct accumulated error in the clock.

† You can retain all of these numbers in the display if you press FIX 6 to change the display format. Display formats are discussed in section 2.

Adding and Subtracting Time Values (HMS+ , HMS−)

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.)

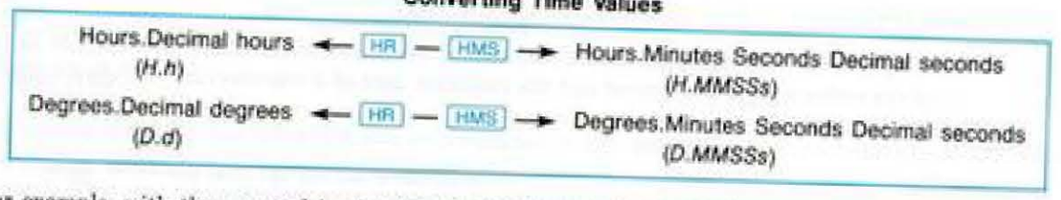
Keystrokes	Display	
FIX 6		
20.165655 ENTER↑	20.165655	Display shows previous result.
11.230712 HMS−	8.534943	Result.

Another example with HMS+ is in program TR in section 22, "Programs for Keeping Time Records."

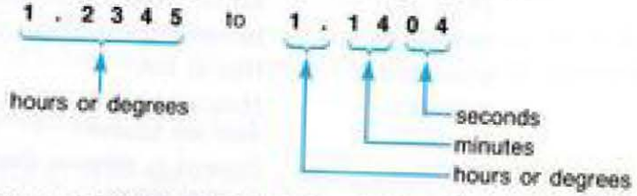
Converting Between Forms for Hours and Degrees (HR , HMS)

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



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



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

Another example with HR is in program TR in section 22, "Programs for Keeping Time Records."

Calendar Functions

Any date from October 15, 1582 (the beginning of the Gregorian calendar) through September 10, 4320 is valid for calculations with dates.

Recalling a Date Value (DATE)

Executing DATE will recall the date.

- If executed as a program instruction, DATE recalls for use (into the X-register) a numeric value in the form ***MM.DDYYYY*** or ***DD.MMYYYY***.
- If executed manually (that is, not in a program), the display will show instead ***MM/DD/YY DAY*** or ***DD.MM.YY DAY*** (where ***DAY*** is a three-letter abbreviation for the day of the week).*

Pressing ← cancels this display and shows ***MM.DDYYYY*** or ***DD.MMYYYY*** in the X-register.

Adding Days to a Date (DATE+)

Given a date and a number of days by which to change (increment or decrement) the date, the function DATE+ (*date plus*) will find the new date.

1. Enter the date (***MM.DDYYYY*** or ***DD.MMYYYY***). Press ENTER↑.
2. Enter the increment (positive) or decrement (negative) value.
3. Execute DATE+.

Example: A proposed bicycle trip from San Francisco to Montreal is estimated to take 135 days. Given this, find the estimated date of arrival and the midpoint date of a trip starting on July 17, 1983.

Keystrokes	Display	
FIX 6		Will show all six decimal places of date.
7.171983 ENTER↑	7.171983	Starting date in MDY format.
135 DATE+	11.291983	Arrive: November 29, 1983.
LASTx	135.000000	Retrieves the number 135.
2 ÷	67.500000	Half of 135.
CHS DATE+	9.231983	Midpoint: September 23, 1983. (Fractional days are ignored.)
FIX 4	9.2320	Returns to previous display format of four decimal places.

* If the year is 2000 to 2199, all four digits will be displayed: ***MM/DD/YYYY:DA*** or ***DD.MM.YYYY DAY*** (where ***DA*** is a two-letter abbreviation).

Finding the Number of Days Between Dates (DDAYS)

Given two dates, the function DDAYS (*delta days*) will calculate the number of days between them. Enter the earlier date (the lower number) first, then the later date. (If you enter the later date first, the result will be a negative number.)

Finding the Day of the Week (DOW)

Given a date (***MM.DDYYYY*** or ***DD.MMYYYY***) in the display (X-register), DOW (*day of week*) finds the day of the week for that date.

- When executed as a program instruction, DOW returns a number from 0 (Sunday) through 6 (Saturday) into the X-register.
- When executed manually, the display will show the name of the day.

Pressing ← will cancel this display and show the X-register value of 0 through 6.

Example: Calculate the day of the week on which the total solar eclipse of July 31, 1981 took place (MDY format).

Keystrokes	Display	
7.311981	7.311981_	The date.
DOW	FRI	The day of the week.
←	5.0000	The value in the X-register (5=Friday).

Alarms

Generally speaking, the HP-41CX can set two types of alarms: message alarms, which beep and remind you of an appointment, and control alarms, which execute programs (or functions from peripheral devices). This section only discusses message alarms; for control alarms, refer to part IV, "Time Functions in Detail."

Like the clock itself, all alarms operate whether the computer is on or off. If an alarm comes due during the execution of a function, it will activate when the execution of that function is completed.

Basic Alarm Operation (XYZALM)

Each execution of XYZALM (*XYZ alarm*) sets a separate alarm using up to four different parameters. Three of these parameters are numbers—repeat interval, date, and time—and one is an Alpha entry (message) in the Alpha register. (The message can be blank.)

Setting Message Alarms

You can set an alarm to display—when it goes off—either a message or the time and date. In addition, you can set an alarm to repeat.

When an alarm is set, it is stored in main memory in the uncommitted registers (page 36).

The steps for setting a message alarm are:

1. Put a message of up to 24 characters in the Alpha register.

   If you do not want a message, clear the Alpha register. (Otherwise, whatever is in the Alpha register when you set the alarm becomes the alarm's message.) The alarm will then display its time and date when it goes off.

2. *For an alarm that you want to repeat periodically,* enter the alarm repeat interval, a numeric time value in the form ***HHHH.MMSSs***. You can omit leading and trailing zeros; the minimum repeat interval is 1 second. (WARNING: It can be difficult to delete a repeating alarm of less than 10 seconds. Refer to "Clearing Repeating Control Alarms," section 16, before experimenting!)

   For a one-time only alarm, enter zero. (This is necessary so the alarm won't repeat.)

   Press [ENTER↑]. (This will be in the Z-register when done.)

3. Enter the alarm date in the format ***MM.DDYYYY*** or ***DD.MMYYYY***. You can omit leading and trailing zeros, *but you must include the year.*

   If the alarm date is the same as the current date, enter zero.

   Press [ENTER↑]. (This will be in the Y-register when done.)

4. Enter the alarm time in the format ***HH.MMSSs*** (using the values on page 63 for setting time). (This is in the X-register.)

5. Execute [XYZALM]. The alarm is now set.

To make alarm-setting even easier, there is a program in section 16 that you can key into program memory and execute whenever you want to set an alarm. The program prompts you for the necessary information. See "Using a Program to Set an Alarm," in section 16.

Alarms That Come Due

When a message alarm comes due, it sounds two tones and displays the first 12 characters of your message. If there is no message, the time and date are shown instead. The alarm then pauses (for about 1 second), starts flashing the display, and sounds up to 16 more pairs of tones.

If a program is running when a message alarm comes due, the program is temporarily interrupted while the alarm activates. The program then resumes from where it was stopped.

The alarm can be acknowledged—that is, stopped and (in most cases) erased from memory—any time while the display is flashing. It cannot be acknowledged during the 1-second pause. If you do not acknowledge the alarm while it is sounding and flashing, the alarm becomes "past due". (Past-due alarms are discussed more in section 16.)

Acknowledging and Deleting Message Alarms as They Go Off

To halt a current, flashing alarm, press any key except [STO]. (Alarms that are not currently active, including past-due alarms, must be cleared using the Alarm Catalog keyboard. This will be explained on page 71.)

Nonrepeating Alarms. Active, nonrepeating alarms are stopped and deleted from memory by pressing any key except [STO].

- Pressing [←] or [ON] will halt the alarm, clear the alarm display, and delete the alarm from memory.
- Pressing any other key except [STO] will halt the alarm, show the alarm display without blinking for about 3 seconds, and then delete the alarm from memory.
- Pressing [STO] halts the alarm, but preserves it in memory as a past-due alarm (in case, for example, you want to save the information in its message). Past-due alarms are discussed in section 16.

Repeating Alarms. If the alarm is a repeat alarm, acknowledging it will *not delete* it from memory, but will *reset* the alarm to its next activation time. (Except with [STO], which does not reset an alarm.)

Press [■] [C] to halt and delete an active, repeating alarm. (This does not use the Alpha keyboard.)

Programmable Alarm-Clearing ([CLRALMS]). There are several programmable functions to identify and clear one or all alarms from within a running program. They are discussed in section 16. [CLRALMS] will clear all alarms.

Examples

The following examples set two message alarms—with and without a message, with and without a repeat interval, for a different day, and for the same day.

> **Note:** So that you can try out examples to set real alarms that will go off within a minute, the first example uses the [TIME] function to recall the current time, add one minute to it, and use that time for the alarm time. Instead of using a real time value, therefore, the displays that follow will use the ***HH:MM:SS*** or ***HH.MMSS*** form to indicate whatever current time value you get.

Example: Set a nonrepeating alarm with no message for 1 minute from now.

Keystrokes	Display	
[ALPHA] [←] [ALPHA]		Clears Alpha register. (Display shows previous result.)
0 [ENTER↑] [ENTER↑]	0.0000	Enters zero twice: for no repeat interval and for current date.
[TIME]	*HH:MM:SS*	Current time display.
[←]	*HH.MMSS*	Clears current time display; returns to X-register, which contains time value. (This step is for clarity but is not necessary because the X-register contents would be used in the next calculation anyway.)
.01 [HMS+]	*HH.MMSS*	Adds 1 minute to time value.
[XYZALM]	*HH.MMSS*	Sets alarm for today, 1 minute from when [TIME] was executed.
	HH:MM MM / DD	Activated alarm flashes.
	or *HH:MM DD.MM*	(Display format depends on your setting.)
[←] (or any key except [STO])	*HH.MMSS*	Acknowledges alarm; returns display to X-register.

Example: Set a repeating alarm for 15-minute intervals starting at 3:15 p.m. on August 31, 1993. Assume [MDY] and [CLK12] formats. Include the message, "Check pH".

Keystrokes	Display	
[MDY]		Set if necessary.
[ALPHA] CHECK PH [ALPHA]		Enters message.
.15 [ENTER↑]	0.1500	15-minute repeat interval.
8.311993 [ENTER↑]	8.3120	Alarm date. (Only four decimal places displayed—this does not affect the internally held value.)
3.15 [CHS] (or 15.15)	−3.15_ (or 15.15_)	3:15 p.m.
[XYZALM]	−3.1500 (or 15.1500)	Sets alarm.

This alarm should now exist in the computer's memory, since it should not have gone off (and been acknowledged) yet. The following topic on the alarm catalog describes how to keep track of all alarms, including unactivated ones, and how to delete any alarm from memory.

Catalog 5: The Alarm Catalog and Alarm Catalog Keyboard

The alarm catalog and the Alarm Catalog keyboard are part of the [ALMCAT] function, also executable as [CATALOG] 5.* ([ALMCAT] is programmable; [CATALOG] 5 is not.)

- Executing [ALMCAT] or pressing [CATALOG] 5 starts a display of all alarms currently in memory, listed in order of alarm time (from earliest to latest).
- The time and date for each alarm are shown first, then any message. If there are no alarms in memory, the catalog shows CAT EMPTY.

Keystrokes	Display	
[CATALOG] 5 (or [ALMCAT])	3:15PM 08 / 31 CHECK PH	Displays the one alarm in memory, with its message.
		Display changes back to X-register; Alarm Catalog keyboard inactive.

The Alarm Catalog keyboard further enables you to:

- Stop, restart, and step through the catalog listing.
- Examine the time, date, repeat interval, or message of any alarm.
- Reset a repeating alarm to its next activation time.
- View the current time (to compare to an alarm time).
- Delete any alarm.

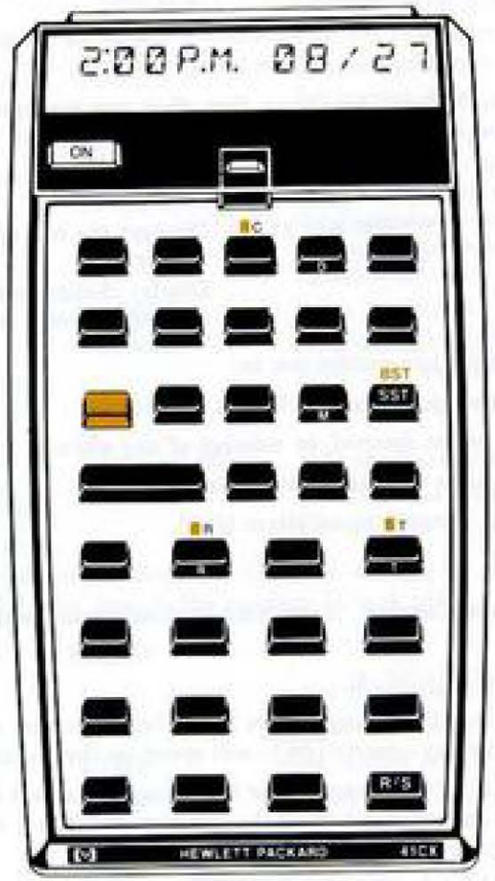
Stepping Through the Alarm Catalog. If you want to closely examine the catalog, delete an alarm, or reset a repeating alarm:

- Start the catalog listing ([CATALOG] 5).
- Press [R/S] (*run/stop*) to stop the listing (before it has finished on its own). (While the catalog is running, pressing any other key—except [ON]—will speed up the listing.)
- Use [SST] (*single step*) or [BST] (*back step*) while the listing is stopped to step through the alarms listed, one at a time. You cannot step beyond the end (using [SST]) or the beginning (using [BST]) of the catalog; the display merely blinks.
- To resume the interrupted catalog listing, press [R/S].
- To cancel (exit) the catalog before it is finished, press [←] *while the listing is stopped*. The display returns to the X-register.

When using a printer, the catalogs only print in Trace mode.

* The HP-41CX has six different catalogs. They are identified together in section 9.

The Active Keys on the Alarm Catalog Keyboard



Note: The Alarm Catalog keys represent a temporary redefinition of the keyboard. The letter keys used *are not Alpha characters*; they are used so the letters can remind you of the catalog function. *Do not press* [ALPHA].

Acting on Alarms in the Alarm Catalog. Once you've stopped the catalog listing (prior to its own completion), you can carry out the following operations on the alarm currently displayed.

The Active Keys on the Alarm Catalog Keyboard

Key(s)	Operation
[T]	Displays the alarm time of the currently listed alarm.
[f] [T]	Displays the current time. (This can be compared to the alarm time.)
[D]	Displays the alarm date.
[R]	Displays the repeat interval.
[f] [R]	Resets the alarm to the next occurrence as determined by the repeat interval, and displays the resulting time and date.*
[M]	Displays the alarm message, if any.
[f] [C]	Deletes the alarm from memory. This is also the way to clear past-due alarms. (You can also use [f] [C]—while the Alarm Catalog is not on—to clear a currently activating alarm.)

* If the alarm is past due, it will reset to the next future occurence.

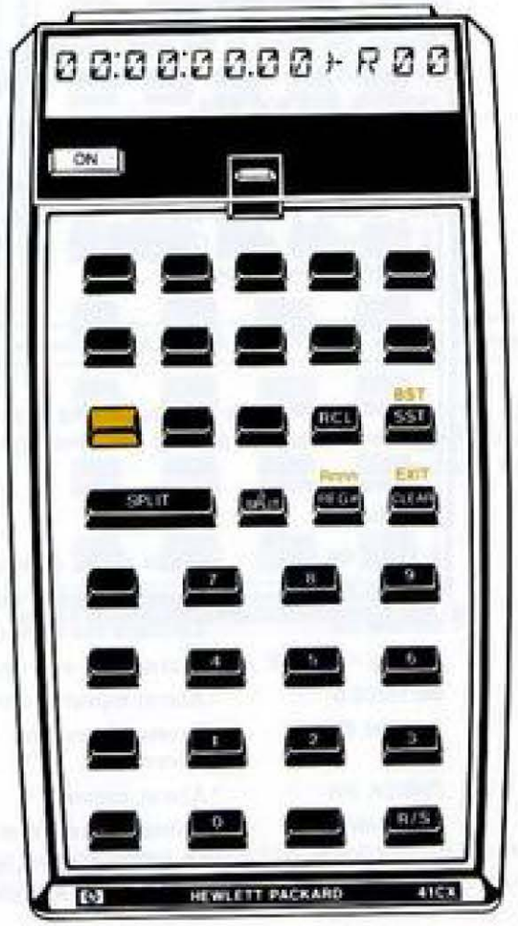
All other keys, except [ON], are deactivated while the Alarm Catalog keyboard is active. If the catalog listing has been halted with [R/S], and no key is pressed for about 2 minutes, the Alarm Catalog keyboard will automatically deactivate.

Keystrokes	Display	
[CATALOG] 5 [R/S]	3:15PM 08/31	Halts alarm catalog listing.
[T]	3:15:00.0 PM	Displays all the digits of the alarm time.
[f] [T]	*HH:MM:SS*	Displays the current time.
[D]	08/31/93 TUE	Alarm date and day.
[R]	00:15:00.0	Alarm repeat interval.
[f] [R]	3:30PM 08/31	Resets alarm time according to repeat interval.
[M]	CHECK PH	Alarm message.
[f] [C]	CAT EMPTY	Deletes alarm from memory.
[←]		Deactivates the Alarm Catalog keyboard and displays the X-register.

Stopwatch Functions

The stopwatch's timer is separate from the clock. The SW function activates the Stopwatch keyboard with its digital stopwatch display, which is set up for taking and reviewing "splits" (timings of events). The stopwatch can be running whether or not it is displayed, the Stopwatch keyboard is active, or the HP-41 is on, so the stopwatch can also be used as an internal timer during program execution. More stopwatch operations are covered in section 17.

The Active Keys on the Stopwatch Keyboard



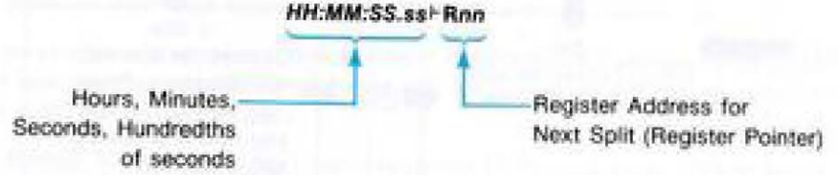
Note: The stopwatch display consumes as much power as a running program. Refer to appendix F, "Time Specifications."

The Stopwatch Keyboard and Display (SW)

Your HP-41CX comes with a keyboard overlay that you can put on your keyboard to mark the functions active for the Stopwatch keyboard. All other functions (except ON)—those not marked on the overlay—cannot be used from the Stopwatch *keyboard*. None of the operations on the Stopwatch keyboard are programmable, although SW is (as are several other stopwatch functions given in section 17).

The computer *will not automatically turn off* while the stopwatch display and keyboard are active.

Activating the Stopwatch Keyboard and Display. To activate the Stopwatch keyboard, execute SW (*stopwatch*). The display shows the time held by the stopwatch and the number of the data storage register in which the next split, when taken, will be stored.



Executing SW sets both register pointers (there is one for storage and one for recall) to 00.

Deactivating the Stopwatch Keyboard and Display. Pressing EXIT (←) deactivates the Stopwatch keyboard and returns the display to the X-register. (This operation is not programmable.)

Note: If the stopwatch is running when you deactivate the Stopwatch keyboard, the stopwatch timer will continue to run, even though it is not displayed.

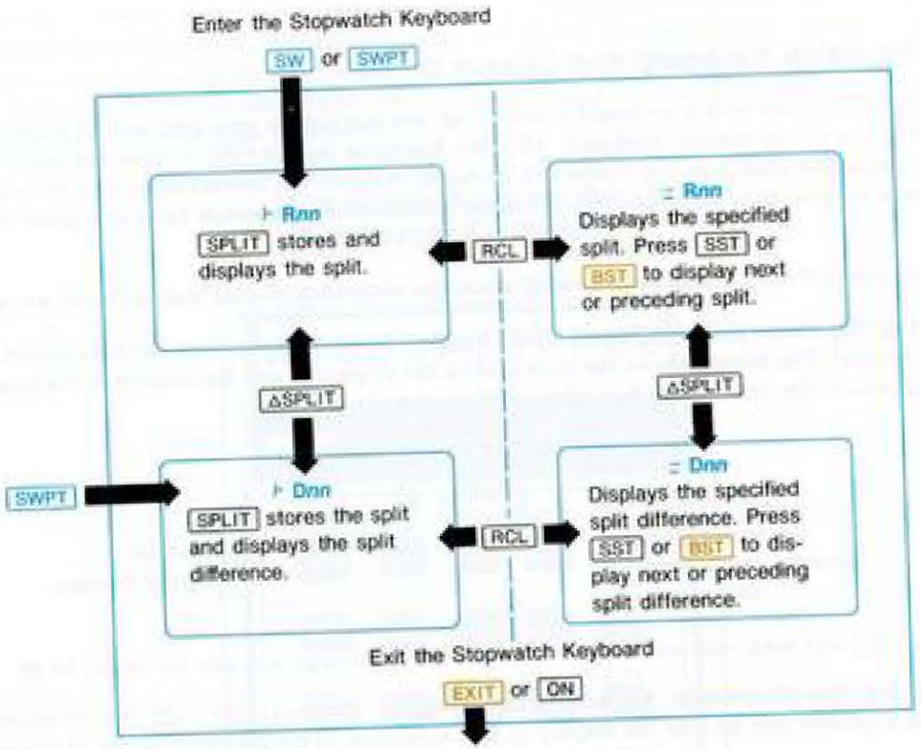
Starting, Halting, and Clearing the Stopwatch (R/S, CLEAR)

Starting and Stopping. The R/S (*run/stop*) key will both run and stop the stopwatch. Stopping and restarting the stopwatch will not automatically reset it to zero.

While the stopwatch is running, you can temporarily freeze the display (not the stopwatch itself) by pressing any nonstopwatch key.

Clearing. Pressing CLEAR (when the stopwatch is stopped) will reset the stopwatch time to zero. If the stopwatch is not cleared, it can run to 99h 59m 59.99s before returning to zero. (CLEAR does not reset the register pointers.)

Modes of Stopwatch Operation



Taking Splits (Timings) ([SPLIT])

Taking a split involves storing the currently displayed time of the running stopwatch into the register specified in the display. You can take as many splits as you have storage registers available in main memory. (If you don't change the memory allocation, R_{00} through R_{99} are available.)

Press [SPLIT] to take a split. While the key is held down, the display—showing the exact time of the split—freezes. The stopwatch itself does not stop. When the key is released, the register pointer increments one number and the running stopwatch display resumes. This stores sequential splits in sequential storage registers.

You can move the register pointer by pressing [SST] (*single step*), [BST] (*back step*), or the digit keys that specify the register you want. The next split will then be stored in the specified register instead.

Recall Mode for Splits ([RCL])

Whether the stopwatch is running or not, pressing [RCL] changes the display from split Storage mode to Recall mode. The register pointer changes from ⊢Rnn to :⊢Rnn, and the display shows the time (split) that was stored in that register in ***HH:MM:SS.ss*** format.*

Using Stored Splits in Other Calculations. The splits you've taken (stored) remain in their respective storage registers even when the stopwatch is not active. The values are stored in the form ***HH.MMSSss***, which is a number you can then use in other calculations. This would allow you, for instance, to recall a split time (without having to re-enter it) and find the difference between it and any other time value to which you might want to compare it.

Viewing Different Registers. Pressing [RCL] displays the contents of R_{00} or the last register whose split was recalled. To view other registers' contents, press [SST] or [BST] to move sequentially to other registers, or press the digit keys specifying the register you want.

The register pointer for taking splits (⊢Rnn or ⊢Dnn) and the register pointer for recalling splits (:⊢Rnn or :⊢Dnn) are separately maintained. This means that you can store splits, then recall splits, then resume storing splits where you left off.

Cancelling Recall Mode. Press [RCL] again to return to Storage mode. (When you deactivate the Stopwatch keyboard and display, all stopwatch register pointers reset to zero.)

In addition, pressing [R/S] or [SPLIT] cancels Recall mode and executes the specified operation ([R/S] or [SPLIT]). [CLEAR] cancels Recall mode.

Example: The following keystrokes store a series of random splits in R_{00} through R_{02} and in R_{10}. (The time values shown are only for illustration; yours will be different. The + symbols indicate changing digits.) If you hold the [SPLIT] key, you will see a frozen display of the split time.

Keystrokes	Display		
[SW]	00:00:00.00	⊢R00	Sets Stopwatch keyboard and display.
[R/S]	00:00:0+.+	⊢R00	Starts stopwatch running.
[SPLIT] (hold)	00:00:02.27	⊢R00	Stores a split in R_{00}.
(release)	00:00:0+.+	⊢R01	The running stopwatch; register pointer incremented.

* If you recall a value that was not stored as a split (that is, it was not stored from the Stopwatch keyboard) it will still appear in ***HH:MM:SS.ss*** format. ERROR :⊢Rnn results if the integer portion is more than two digits.

Keystrokes	Display		
[SPLIT] (hold)	00:00:02.43	⊢R01	Stores a split in R_{01}.
(release)	00:00:0•.•	⊢R02	
[SPLIT] (hold)	00:00:04.78	⊢R02	Stores a split in R_{02}.
(release)	00:00:0•.•	⊢R03	
10	00:00:••.•	⊢R10	Changes register pointer to R_{10}.
[SPLIT] (hold)	00:00:10.44	⊢R10	Stores a split in R_{10}.
(release)	00:00:••.•	⊢R11	
[R/S]	00:00:13.22	⊢R11	Stops stopwatch.

Now set Recall mode and review the splits taken.

Keystrokes	Display		
[RCL]	00:00:02.27	⊣R00	Recalls split value from R_{00}.
[SST]	00:00:02.43	⊣R01	From R_{01}.
[SST]	00:00:04.78	⊣R02	From R_{02}.
10	00:00:10.44	⊣R10	Recalls value from R_{10}.
[RCL]	00:00:13.22	⊢R11	Returns to stopwatch display.

Running Out of Storage Registers

While taking splits, a tone will sound when there is only one register left in memory for split storage. The tone will sound again when there are no registers left. If you still try to take another split, the error message NONEXISTENT will appear and the Stopwatch keyboard will be deactivated—*without* halting the stopwatch timer itself.

(If you want to take more splits, you can do so by allocating more registers to data storage. This is explained in section 12.)

Viewing Split Differences ([ΔSPLIT])

The [ΔSPLIT] (*delta split*) function sets Delta Split mode, allowing you to see the time difference between successive splits, while storing the actual split times themselves. The procedure is:

1. Press [ΔSPLIT] to establish Delta Split mode. The display changes to ⊢Dnn (or ⊣Dnn).
2. Proceed just as with regular splits: press [SPLIT] to record a split time. The display will show the delta-split time while the [SPLIT] key is held down.

3. Press [RCL] to set Recall mode. Recall mode in Delta Split mode recalls and displays the delta split-value (the time difference) between the indicated register and the preceding one. The display shows ⊣Dnn.
4. Press [ΔSPLIT] again to cancel Delta Split mode and return to Regular Split mode, the regular split display. Display shows ⊢Rnn or ⊣Rnn.

Interpreting the Pointer in the Stopwatch Display

Display	Meaning
⊢Rnn	Store split, display split.
⊣Rnn	Recall split.
⊢Dnn	Store split, display split difference.
⊣Dnn	Find split difference.

Note: You can find delta-split values (by pressing [ΔSPLIT] and [RCL]) for any stored split value, whether the split was actually taken during Delta Split mode or not. In the same way, using [SPLIT] always *stores* the accumulated split time, whether the split is actually taken during Regular Split or Delta Split mode. *Only the display* of the time you see while pressing [SPLIT] is affected by whether or not Delta Split mode is in effect. The actual stored values will be the same.

Example: With the Stopwatch keyboard still on, set Delta Split mode and take three running splits. If you hold the [SPLIT] key down, you will be able to read the delta-split values. (Your numbers will differ from those used below.)

Keystrokes	Display		
[CLEAR]	00:00:00.00	⊢R11	Clears stopwatch; register pointer assumed still at R_{11} from last example.
[ΔSPLIT]	00:00:00.00	⊢D11	Changes to Delta Split mode.
[R/S]	00:00:••.•	⊢D11	Starts stopwatch running.
[SPLIT] (hold)	00:00:03.45	⊢D11	First split.
(release)	00:00:0•.•	⊢D12	Running display. Register pointer incremented.
[SPLIT] (hold)	00:00:04.10	⊢D12	Second split. Shows R_{12}–R_{11} split difference.
(release)	00:00:0•.•	⊢D13	Running display of accumulated time.
[SPLIT] (hold)	00:00:03.51	⊢D13	Third split. Shows R_{13}–R_{12}.
(release)	00:00:••.•	⊢D14	
[R/S]	00:00:14.66	⊢D14	Halts stopwatch.
[RCL]	00:00:08.36	⊣D10	Recall-pointer is at 10 from last example.

Keystrokes	Display		
SST	ERROR	⌐D11	Split difference for R_{11}–R_{10} is negative. (This error is explained after this example.)
SST	00:00:04.10	⌐D12	Split difference R_{12}–R_{11}.
SST	00:00:03.51	⌐D13	Split difference R_{13}–R_{12}.
ΔSPLIT	00:00:11.06	⌐R13	Returns to Regular Split mode in Recall mode. Shows the actual, accumulated split time in R_{13}.
BST	00:00:07.55	⌐R12	Recalls split time in R_{12}.
BST	00:00:03.45	⌐R11	Recalls split in R_{11}.
EXIT			Cancels Stopwatch keyboard. Display shows X-register.

Invalid Results and Error Displays. When the HP-41 is operating in Recall mode, the message ERROR ⌐D*nn* will result if the split difference is a negative number. (The smaller-numbered register is subtracted from the larger-numbered register.) This is what happened above for ⌐D11, since the split in R_{11} is smaller than the split in R_{10}.

Also, attempting to recall (Recall mode) the split difference involving a register whose contents are not in the form ***HH.MMSSss*** will cause ERROR ⌐D*nn*. Recalling the contents of a register whose contents are 100 or more will cause ERROR ⌐R*nn* or ERROR ⌐D*nn*. These errors can result if the register contents were not stored as a stopwatch time (nonstopwatch data).

If these errors result, just set the pointer to a different register. (Do not press ←, which will switch the stopwatch mode from Recall mode to Storage mode. This is indicated in the display as a change from ⌐D to ⊦D.)

Section 7

Elementary Programming

Contents

The HP-41 operations covered in sections 1 through 6 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 89, 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 90.
3. "Copying a Program from an Application Module," page 107, 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 (section 10) 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 part V, "Programming 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

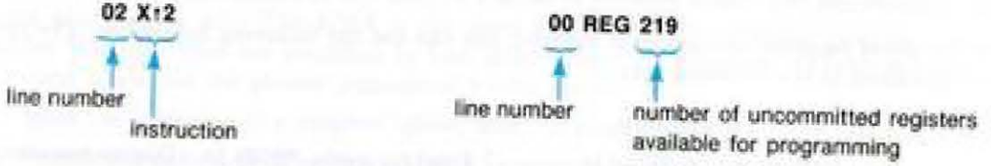
One *register* of memory can provide up to seven *lines* of program instruction. If there is not enough uncommitted register space available to store an instruction you are adding in Program mode, the HP-41 will "pack" its memory and then display TRY AGAIN. (*Packing* memory means to arrange the program 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. How to do this is explained in section 12, "Main Memory." (You can also make registers available by clearing other programs, clearing any alarms, or clearing User-function assignments.)

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 part III, section 12, "Main Memory," 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 100 registers for data storage, 219 registers for all else.

The Pointer in Program Memory

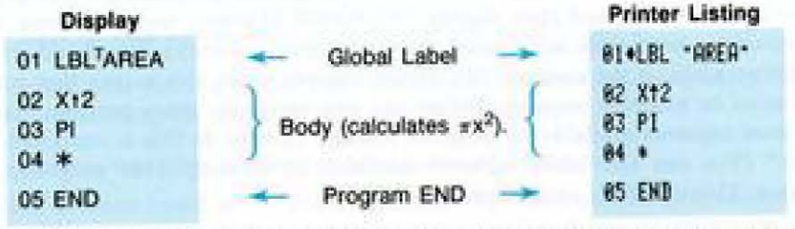
When you press [PRGM] and switch to Program mode, the display will show you a two- (or three-) digit line number (on the left) and the *current program line*. If your program memory is empty, you will see 00 REG 219. The displayed line—the current line in the current program—is determined by the *program 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:



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 45.)

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 219	Program mode; **PRGM** lit. (Display assumes no other programs present.)
[GTO] [.] [.]	00 REG 219	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"AREA	The global label heads the program. Remember to use the Alpha keyboard to enter Alpha characters.
[↵]	02 X↑2	All functions appear as their Alpha names.
[−]	03 PI	
[×]	04 *	
[GTO] [.] [.]	00 REG 217	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 on page 98.)

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 98.) 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. The technique of program branching is covered in section 20.

- 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 section 18 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 92.)
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 pages 86-87), 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] [y^x] [X]	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 98.)

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 94).

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 105.

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-41CX, 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"AREA
02 X↑2
03 PSE
04 PI
05 *
06 END
```

In the display, the " ("raised T") always precedes an Alpha string.‡

Display shows result of r^2.

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. See also appendix D, "Printer Operation."

† 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
07 END
```

Sounds tones when program is done.

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 page 93.)

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"AREA
02"AREA OF CIRCLE
03 AVIEW
04 PSE
05 X↑2
06 PI
07 *
08 CLD
09 END
```

The " 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.

Clears the message display so that the result in the X-register will show.

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 86), 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"AREA	
02"AREA OF CIRCLE	
03 AVIEW	
04 PSE	
05"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 86) 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 217	Packs memory and inserts END after previous routine, if needed.
LBL	01 LBL _ _	Cues for label.
ALPHA CIRCLE ALPHA	1 LBL"CIRCLE	New global label for new program.
ALPHA AREA OF CIRCLE	A OF CIRCLE _	Scrolls through display.
AVIEW ALPHA	03 AVIEW	
	04 PSE	
ALPHA RADIUS? ALPHA	05"RADIUS?	Use XEQ or a User key.
PROMPT	06 PROMPT	Alpha message.
x^2	07 X↑2	Use XEQ or a User key.
*	08 PI	
X	09 *	
ALPHA AREA=	10"AREA=	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 210	(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 91).

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 $\sqrt{x}$ * X 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 100.

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 91) 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 each END is a "byte count," that is, the number of bytes of program memory used by that program.* 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 ENDs. 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. Such considerations are discussed in more detail in section 12, "Main Memory."

If you keyed in the AREA program (from page 86) and the CIRCLE program (from page 97), then program memory looks like this:

```
01 LBL"AREA
02 X↑2
03 PI
04 *
05 END
01 LBL"CIRCLE
02"AREA OF CIRCLE
03 AVIEW
04 PSE
05"RADIUS?
06 PROMPT
07 X↑2
08 PI
09 *
10"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"AREA	After specifying 1, immediately press [R/S] so you can see and read each entry.
[SST]	END 14	This routine took 14 bytes.*
[SST]	LBL"CIRCLE	
[SST]	END 51	Number of bytes in CIRCLE is 51.*
[SST]	.END. REG 210	210 uncommitted registers still available for programming.*
[SST]		Going past the permanent .END. cancels the catalog operation and returns the X-register display.

* This figure can vary if program memory is not packed. It will vary if you have made corrections to program lines, unless you do [GTO] [.] [.] afterwards. This is because making alterations uses extra memory until you pack memory.

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 ENDs, 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 87-88.)

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 91.)

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 CL← *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"CIRCLE	Program mode: displays first line of CIRCLE.
SST	EA OF CIRCLE	Moves to second line (display scrolls).
←	1 LBL"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 99.) 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"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 86-87) 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^TAREA	In Program mode, use . with GTO. Position is at line 01.
ALPHA RADIUS? ALPHA	02^TRADIUS?	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^TAREA=	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 or Programs (CLP, PCLPS)

Clearing One Program. 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 LBL^TAREA
02^TRADIUS?
03 PROMPT
04 X↑2
05 PI
06 *
07^TAREA=
08 ARCL X
09 AVIEW
10 END
.END. REG 215
```

Clearing Several Programs. To clear more than one program at a time, use PCLPS (*programmable clear programs*). PCLPS *is* programmable (so if you use it in Program mode you will store it), and it is *not* a parameter function (so the global label should aleady be in the Alpha register when you execute PCLPS).

To use PCLPS, place in the Alpha register the global label of the first program you want deleted, then execute PCLPS. (If the Alpha register is empty, the current program is considered the first one to be deleted.) This clears the named program and all subsequent programs, up to but excluding the permanent .END.

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, but a complete explanation of the characteristics and operation of conditional tests awaits part V, "Programming in Detail," sections 19 and 20.

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 216	
[LBL] [ALPHA] QUAD [ALPHA]	01 LBL"QUAD	Global label for program.
[ALPHA] a=? [ALPHA]	02"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"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"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 203	(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 (see pages 90-91 for methods of execution). 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, plus all time and extended memory functions (see "Cataloguing the New Functions" in appendix I). 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 100.)

  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 (refer to page 35 and section 19) 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. Refer to appendix D, "Printer Operation," for a discussion of the effect of the printer on program operation.

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 203	
GTO · ·		
LBL ALPHA VECTOR ALPHA	1 LBL"VECTOR	
CLΣ	02 CLΣ	
ALPHA THETA1=? ALPHA	03"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"R1=?	
PROMPT	06 PROMPT	
P→R	07 P→R	
Σ+	08 Σ+	
ALPHA THETA2=? ALPHA	09"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"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: Σθ in Y-register, Σr in X-register.
ALPHA ΣTHTA=	18"ΣTHTA=	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 Σθ is in the Y-register; Σr is in the X-register.
ARCL . Y ALPHA	19 ARCL Y	
PROMPT	20 PROMPT	
ALPHA ΣR=	21"ΣR=	
ARCL . X	22 ARCL X	
AVIEW ALPHA	23 AVIEW	
		The PROMPT function is used to display the Alpha register (with Σθ) and interrupt the program.
GTO . .	00 REG 192	(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_	This assigns VECTOR to the LN key on the User keyboard.
	SN VECTOR 15	
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 Σθ.
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.)

Section 8

Storing Text, Data, and Programs in Files

Contents

Files are units of information storage outside of main memory.* They can (but don't have to) be used or manipulated by programs. Storing files in the HP-41 is like keeping information on paper and storing that information in labeled files kept in a file cabinet—except that with the HP-41 it's easier to modify a file's contents and length, it's easy to keep these files private, and they don't take up space in your desk. This section explains how to create and gain access to text, data, and program files. The emphasis is on text files and the Text Editor, because storing text in the HP-41 is unique to text files. Data files and program files are akin to the data storage registers and programs in main memory with which you are already familiar.

Storing Text With Text Files

The HP-41 is a machine that functions on many different levels—doing calculations, programming instructions for automatic calculations, and using alphabetic messages and codes (the Alpha keyboard). There is one more level to the HP-41CX: text editing. By creating files of text, you can store and edit written material. Text files store words, lists, and messages—using any characters that you can produce using the Alpha keyboard.† With the Text Editor, it is very easy to produce, read, update, and erase text files.

How a Text File is Organized

Every file is identified by a name (as programs are identified by labels) and a type (in this case, the type is text file). Each text file is composed of *records*, which are composed of *characters*. The information about a file's name, type, size, and *pointer* is stored at the beginning of the file itself, in a *header*.

* The files themselves are stored in the extended memory, an area in the HP-41 memory that can hold different kinds of information than that held in main memory. These concepts are covered in part III, section 13, "Extended Memory." Extended memory can be expanded using extended memory modules. Refer to section 13 and appendix E.

† All Alpha characters and digits are encoded in the computer according to the American Standard Code for Information Interchange (ASCII, pronounced "as'kee"), so these files are also called "ASCII files".

The diagram below shows a simplified version of the structure of a text file named SECRETS with six records. (There is a more detailed and more exact depiction of file structure in section 13.)

Registers and Memory Requirements. When you create a text file, you must allocate to it a certain amount of memory, that is, a certain number of *registers* in memory. It's easiest just to estimate roughly how many registers to allocate for the text file, and then change the file size later if you need more (or less) room.

A text file uses roughly one register of memory for every seven characters you put in a text file. Each record you create and the file itself also need "overhead" space in memory equivalent to one character each.* A more detailed description of file memory requirements is given in section 13.

For instance, the above example has 149 characters (including spaces) and 6 records. That makes a total of 149 + 6 + 1 = 156 characters. 156/7 means at least 23 whole registers (if no more characters were added) are needed for this text file.

Records and Characters. You can think of records as being your individual entries in a list or text. (Putting one entry per record, as in the SECRETS file, makes reading and gaining access to individual entries easy.) The length of a record depends on how many characters you put in it, with a maximum of 254 characters. So, every record in a file can have a different length.

* Each file also uses an additional two registers for the header. However, these two registers are not included in the file size that you specify and that the computer displays—these two registers are taken up automatically.

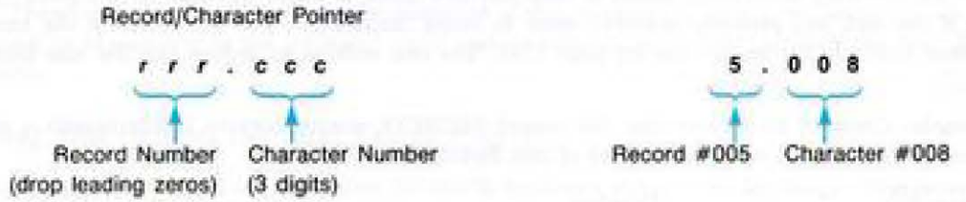
The File Pointer and the Current File

As program memory uses a program pointer to indicate the current program in program memory, so extended memory uses a file pointer to indicate the *current file* in extended memory. The file pointer is often moved (as part of a specific function) by indicating in the Alpha register the name of the desired file.

The current file is the one upon which file operations act. Some file functions use the contents of the Alpha register to determine which file to operate on, and make that file the current file; if the Alpha register is empty, then such functions act on whichever file is already the current file.

The Record/Character Pointer

The record/character pointer is a two-part number (or address) located in the header of all text files. This pointer indicates which *record* (*rrr*) and which *character* (*.ccc*) are current, that is, which character in which record will be affected by the next operation meant to alter a character. In the SECRETS text file, the pointer is 5.008, meaning the current character is #008 in record #005. *Numbering starts with #000,* because the first record or character is 000, not 001.



Locating Files and Adjusting Pointers (SEEKPTA)

Some file functions act only on whichever file is already the current file. To change the current file, change the file pointer address by using SEEKPTA (*seek pointer using the Alpha register*).

1. Put the file name in the Alpha register.
2. Put the address for the record/character pointer (*rrr.ccc*) in the X-register. (A zero will locate the first character in the first record.)
3. Execute SEEKPTA.

Creating and Clearing Text Files

The simplest way to work on text files is by using the Text Editor. First, however, each new, empty file must be created by name.

Creating Text Files (CRFLAS)

Before entering and storing text material, first set up a text file for it. In order to create a text file, the computer must know two things:

- The name to give the file.
- The size (number of registers) to give to the file.

Creating a file also makes it the current file. To create a file:

1. Put a name in the Alpha register (up to seven characters).
2. Enter the estimated number of registers into the X-register (the normal display).
3. Execute CRFLAS (*create file-ASCII**).

Since you probably won't know ahead of time just how much material will end up in a given file (and, even if you did, you probably wouldn't want to count characters), you can estimate the number of registers that will be needed (as on page 114). You can enlarge or reduce the file size later using RESZFL.

Example: Create a 23-register text file named SECRETS, preparatory to filling it with a series of personal numbers and codes at the end of this section.

Keystrokes	Display	
ALPHA SECRETS ALPHA		Remember to set the Alpha keyboard to type Alpha characters.
23	23_	
CRFLAS	23.0000	File SECRETS now exists (empty).

Changing the Size of a Text File (RESZFL)

You can change the size of the current file by specifying the new file size (in the X-register) and then executing RESZFL (*resize file*).

Keystrokes	Display	
30 RESZFL	30.0000	Enlarges the current file (SECRETS) to 30 registers in extended memory (even though there are no records and no characters in it).

* Refer to the second footnote on page 113.

To make a file the current file, use SEEKPTA (page 115).

Clearing and Purging Text Files (CLFL, PURFL)

- To *clear* a file's contents only—leaving its name and size allocation—put the file name in the Alpha register and execute CLFL (*clear file*). The cleared file becomes the current file.
- To *purge* a file—removing its contents, name, and space—put the file name in the Alpha register and execute PURFL (*purge file*).

Entering and Editing Text With the Text Editor

The HP-41CX is equipped with a special function called ED—the Text Editor. The Text Editor creates a different operating state, wherein the keyboard is redefined (that is, the keys have different operations) and the display is reorganized. Instead of operating with numbers and calculations and the X-register, the Text Editor operates with Alpha character strings within records (entries) in named files. (A string is a series of characters that does not represent an instruction or a parameter.)

All of the editing functions are nonprogrammable, except for the function that activates the Editor (ED). Therefore, the Text Editor can be activated within a program, so you or another user (who would need to be monitoring the program) could make a text file entry, and then exit the Text Editor when done.

Entering and Exiting the Text Editor (ED)

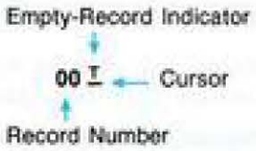
While the Text Editor is active, the entire HP-41CX keyboard is redefined for text editing functions. The key symbols and placement of the text editing functions are shown on the back of the HP-41CX, as well as in the diagram and discussion below. (This information is also in the Quick Reference Guide.)

- To enter the Text Editor, put the name of the desired file in the Alpha register and execute ED. An empty Alpha register specifies the current file. (The blinking character in the display indicates the position of the character pointer, and is explained under the next heading.)
- To exit the Text Editor, press EXIT—the same key as the ON key. (This will not turn the computer off.)
- After a few minutes of inactivity, the computer automatically deactivates the Text Editor.

The only HP-41 annunciators that operate with the Text Editor are: **BAT** (low battery power), **SHIFT**, **ALPHA** (Alpha keyboard active), and **1**. The **1** is used to indicate Insert mode (explained below). When you exit the Editor, the status of all annunciators is restored to what it was when you entered the Editor.

The Display and the Cursor

Upon entering an empty file, the display will show



When you enter the Text Editor, you will see the blinking cursor (an underscore) alternating its display with whatever character it is under. The cursor represents the point of activity; that is, it shows you where a new character will be entered, or an old one deleted or overwritten. In an empty record, the cursor alternates with the raised-T, empty-record indicator (ᵀ). At the beginning of a record, a two-digit record number appears at the left side of the display.

When you enter a text file, that file's record/character pointer determines where the cursor will be, that is, which record and what portion of that record will be displayed. Unless you alter the pointer (with SEEKPTA), the point at which you enter a file (executing ED) will be the same as the point from which you last exited it. This is so you can leave the Editor and later return to it without losing your place.

More detailed aspects of the display of characters are included in section 14, "The Text Editor."

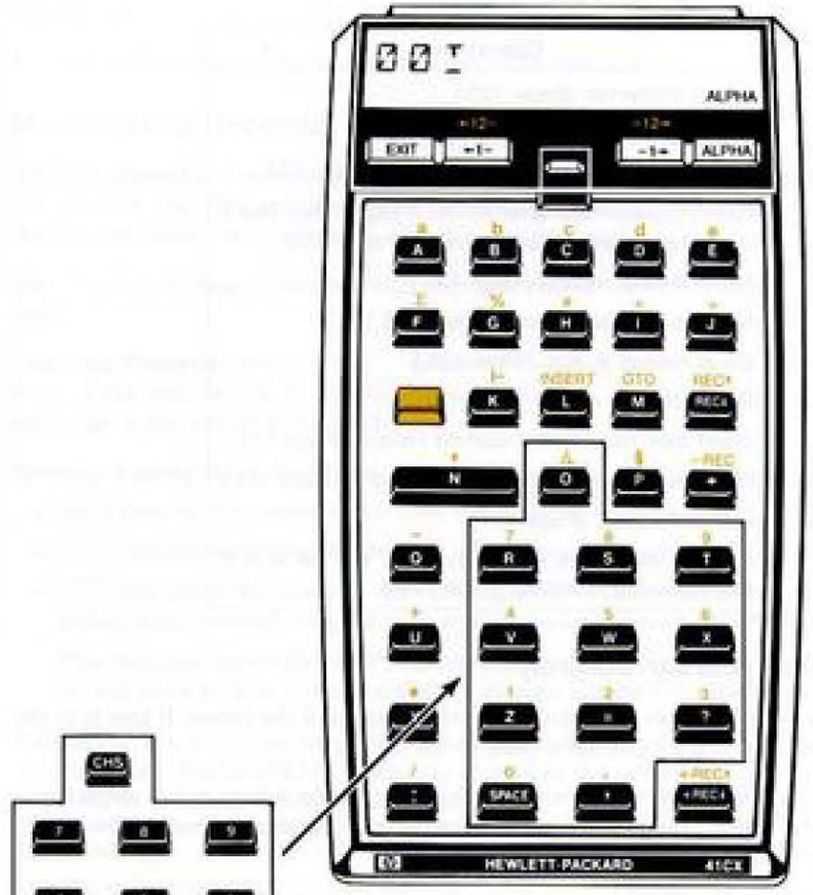
The Text Editor Keyboard

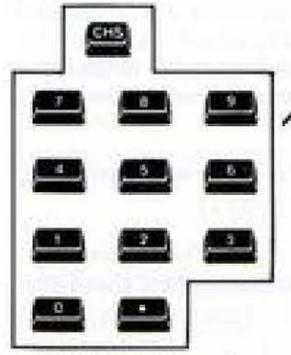
The Text Editor keyboard, as shown in the diagram below and on the backplate of the HP-41CX, is a superset of the Alpha keyboard: it includes all the Alpha characters (in blue on the actual keyboard) and the shifted Alpha characters (the digits plus the gold characters shown on the diagrams). In addition to this Alpha character set, which you need for writing text, are the *control* keys, which control the cursor, record formation, and character insertion/deletion.

The Numeric Keypad. The Alpha keyboard is automatically activated when you execute ED manually, but not when ED is executed in a program. The Text Editor uses either the Alpha keyboard or the numeric keypad. The numeric keypad is shown in the inset on the keyboard diagram (the digits, decimal point, and CHS).* The numeric keypad is useful if you will be entering solely digits. It is activated by pressing ALPHA to deactivate the Alpha keyboard. This makes the digits become the unshifted functions, and eliminates access to the Alpha characters. Pressing ALPHA a second time activates the Alpha keyboard.

* If flag 28, the radix mark flag, is set, then . on the numeric keypad will register a ",".

The Text Editor Keyboard





Numeric Keypad
(ALPHA off)

The Control Keys. The control functions of the Text Editor are listed in the table below.

The Text Editor Keyboard

Key	Operation
←	*Delete character.* (Page 120.)
←1, 1→	*Move cursor left or right.* (Page 120.)
←12, 12→	*Move cursor 12 spaces left or right.* (Page 120.)
INSERT	*Insert/Replace character mode.* A toggle key. The **1** is lit when Insert mode is active. (Page 120.)
REC↓	*Move to next record.* (Page 121.)
REC↑	*Move to previous record.* (Page 121.)
GTO	*Go to record # nnn.* (Page 121.)
−REC	*Delete current record.* (Page 121.)
+REC↓	*Insert new record after current record.* (Page 121.)
+REC↑	*Insert new record before current record.* (Page 121.)
EXIT	*Exit Text Editor.* (Page 117.)
ALPHA	*Activate/deactivate Alpha keyboard.* **ALPHA** is lit if this keyboard is active. (Page 118.)

Writing and Editing Text (Cursor Control)

Deleting Characters. The ← key deletes the character at the position of the cursor. If this is in the middle of a record, the characters to the right move back to fill the gap.

Moving the Cursor. ←1 and 1→ move the cursor one character to the left or right, respectively, without altering the text. The cursor cannot move beyond the current beginning or end of the record (which is one position after the last character).

←12 and 12→ move the cursor 12 characters (one display-length) to the left or right, respectively. (For long records, you can get to the beginning quickly by pressing REC↓, then REC↑.)

Insert and Replace Mode. Normally, by default, the Editor is in *Replace mode*. That is, any characters you enter will write over any character previously at that position. The cursor moves ahead after each character entry.

Pressing INSERT sets *Insert mode*, so any character you type is inserted just ahead of the cursor. The **1** annunciator is displayed while Insert mode is on. To return to Replace mode, press INSERT again, like a toggle key.

You can add characters to the end of a record in either mode.

Manipulating Records

Adding Records. Pressing +REC↓ inserts a new, empty record *after* the current record, and moves the cursor to the first position in the new record. Pressing +REC↑ inserts a new, empty record *before* the current record, and moves the cursor to its first postion.

The ᵀ appears in empty records. With each new record, the Editor automatically re-establishes Replace mode.

Deleting Records. Pressing −REC deletes the current record and moves the cursor to the first character of the next record. If the deleted record was the last record, then the cursor moves to the first character of the previous record instead.

Moving Among Records. There are three operations for moving among records:

- REC↓ moves the cursor to the first character of the next record, if there is one.
- REC↑ moves the cursor to the first character of the previous record, if there is one.
- GTO will move the cursor to the first character of the specified record. It is a parameter function, which, when pressed, cues for a three-digit record number: GTO ___.

  This function works like other parameter functions: it is executed when all the input cues are filled in, and prior to that the input digits and even the function itself can be corrected using ←.

Example: The following keystroke sequence creates six records of personal numbers and codes into the empty text file SECRETS, which was created in the example on page 116. The text for each record is shown in the diagram of the file on page 114.

Keystrokes	Display	
ALPHA SECRETS ALPHA		Type SECRETS into Alpha register if it isn't still in there.
ED	00 ᵀ	The ᵀ and the underscore alternate in the display. The Alpha keyboard is automatically set.
BANK ACCT, SWISS	ACCT, SWISS_	Add one space before digits.
ALPHA 85463 ALPHA	SWISS 85463_	Use numeric keypad to enter digits faster.
+REC↓	01 ᵀ	Creates second record (01).
BICYCLE, SERIAL NO.	SERIAL NO._	
GND 1472	NO. GND 1472_	The end of the display, record 01.
+REC↓	02 ᵀ	

Keystrokes	Display	
LOCKER COMBO 33 48 29	BO 33 48 29_	End of record 02. There is no space key on the numeric keypad; hyphens could be substituted ([CHS]) if numeric keypad were used.
[+REC▮]	03 T	
PASSWORD, COMPUTER DMR	OMPUTER DMR	End of record 03.
[+REC▮]	04 T	
PHONE CREDIT CARD 555 000 1234 5678	0 1234 5678_	End of record 04.
[+REC▮]	05 T	
SSN	SSN_	Record 05, the sixth and last record.
[ALPHA] 000-88-4424	000-88-4424_	Use numeric keypad to enter digits. For hyphens on numeric keypad, use [CHS].
[EXIT]		Exits the Editor; display returns to X-register.

Suppose that, to shorten a record length, you want to go back to record 04 and abbreviate CREDIT CARD to CR. CD.

Keystrokes	Display	
[ALPHA] [ALPHA]		Check to make sure the file name SECRETS is still in the Alpha register.
[ED]	000-88-4424_	Cursor is at same position it was at last.
[REC↑]	04 PHONE CRE	Moves to previous record, 04.
[1➡] [1➡]...	NE CREDIT CA	Move cursor to the "E".
[.] [←] [←] [←]	NE CR._CARD 5	Replaces "E" with "." and deletes three characters.
[1➡] [1➡]	CR. CARD 555	
[➡] [➡] [1➡]	CR. CD._555 00	
[INSERT] [.]	CR. CD._555 00	The **1** annunciator in the display indicates Insert mode.
[EXIT]		

Storing Data With Data Files

The contents of data storage registers in main memory can be copied into data files in extended memory, freeing the registers in main memory for other data. When you want access to the data saved in files, you copy the data from the files back into registers (or one register at a time) in main memory. Explained here are only the simplest data-file operations; refer to section 13 for a complete discussion.

Saving Data Register Contents in Files ([CRFLD])

Just as with text files, you must create a data file—giving it a name and a size—before storing information in it.

1. Put the name of the file in the Alpha register (up to seven Alpha characters; no commas).
2. Put the size of the data file in the X-register. The size of the file is the *number of registers* needed, based on the number of data registers you will want to save.*
3. Execute [CRFLD] (*create file, data*).

Saving All Registers ([SAVER]). The function [SAVER] (*save registers*) will save the data in all main memory registers by copying all values into the corresponding register number in a given data file.

1. Put the name of the data file in the Alpha register.
2. Execute [SAVER].

There must be at least as many registers in the data file as there are data registers in main memory.

Saving One Register at a Time ([SAVEX]). The function [SAVEX] (*save X*) copies ("saves") the contents of the X-register into the specified register *n* of a data file.

1. Put the name of the data file in the Alpha register.
2. Put *n* in the X-register and execute [SEEKPTA]. This sets the data-file pointer to *n*.
3. Execute [SAVEX].

The value from the X-register is copied into register *n* in the data file.

If you want to save data into sequential registers, then you can skip step 2 after saving the first value. This is because the function [SAVEX] automatically increments the data-file pointer by one.

Recalling Data From Files

Recalling All Registers ([GETR]). The function [GETR] (*get registers*) copies the contents of all registers in a given data file into the corresponding registers in main memory. (This is the reverse of [SAVER].)

1. Put the name of the data file in the Alpha register.
2. Execute [GETR].

The data from register 000 in the data file is copied into R_{00} in main memory, and so on. If there are fewer registers available in main memory than there are registers in the data file, as many registers as possible will be copied.

* Data files, like text files, use two registers for a header, where file name, type, size, and pointer information are stored. However, the header registers are not included in the file size.

Recalling From One Register at a Time (GETX). The function GETX (*get X*) copies the contents from the specified data-file register *n* into the X-register. (This is the reverse of SAVEX.)

1. Put the data-file name in the Alpha register.
2. Put *n* in the X-register and execute SEEKPTA. This sets the data-file pointer to register *n*.
3. Execute GETX.

If you want to recall data from sequential registers, then you can skip step 2 after recalling the first register. This is because the function GETX automatically increments the data-file pointer by one.

Storing Programs With Program Files

An exact copy of a program in main memory can be saved in a program file, freeing that space in main memory for other programs you want to create or edit. When you want access to that saved program again, you copy it back into main memory. Described below is the simplest way to save a program; there are other ways described in section 13.

You do not have to create a program file before saving a given program. A program is *identified* by a global label. A program is *defined* as all program lines following one **END** (or the beginning of program memory) through the next **END**.

Saving a Program in a File (SAVEP).

1. Put the global label contained in that program into the Alpha register.
2. Execute SAVEP (*save program*). This creates a program file of the *same name* as the source program. (If a program file of the same name *already* existed, that program file would be *overwritten*.)
3. Execute CLP *global label* to clear the source program from main memory. (Optional.)

Recalling a Program From a File (GETP). This procedure is for recalling a program manually *only*. If you want to do this in a running program, refer to section 13.

1. Press GTO . . to save the last program in program memory. (If you don't do this, the last program in main memory will be overwritten.)
2. Put the name of the program file into the Alpha register.
3. Execute GETP (*get program*). This program is now the current program in program memory.
4. You can press R/S to execute this program. It will start at the first line in that program.

Example: Move the program labeled AREA to a program file. AREA was created in section 7, on pages 86-87.

Keystrokes	Display	
ALPHA AREA ALPHA		Puts the global label in the Alpha register.
SAVEP		An exact copy of AREA is now stored in extended memory as a program file named AREA.
CLP	CLP _	
ALPHA AREA ALPHA	PACKING	Clears AREA from main memory and packs main memory. Now the only copy of AREA is the program file.

Catalog 4: The Catalog of Files

Catalog 4 displays a list of all the files in extended memory. Along with each file name is an indication of the file type (A for ASCII/text, D for data, P for program) and the file size (in registers). The function can be executed as either CATALOG 4 or as EMDIR (*extended memory directory*), with EMDIR (but not CATALOG 4) being programmable.

Catalog 4 operates similarly to catalog 1, the program directory:

- Pressing CATALOG 4 or executing EMDIR starts a listing—in order of creation—of the names of all the files, with type and size, in extended memory. The size is the one you allocated for your information in the file.
- R/S will stop and restart this listing (if it hasn't already finished).
- When the catalog is stopped, SST will single-step forward through the listing and BST will backstep one entry at a time through the listing.
- While the catalog is stopped, pressing ← aborts the catalog and returns the X-register to the display. The file that was being displayed becomes the current file. (The only other key that is active during the catalog is ON.)
- When the catalog runs to completion, the number of registers available in extended memory for another file is returned to the X-register and displayed. (This figure is two less than the total number of registers still available in extended memory, because the two registers needed for the next file header are not counted.)

If you are using a printer, the catalog prints only in Trace mode.

Keystrokes	Display	
CATALOG	CAT _	*Which catalog?*
4	SECRETS A030	Text file SECRETS; 30 registers allocated.*
	AREA P005	Program file AREA; 5 registers allocated.*
	85.0000	85 registers available for the information in a new file.

How Catalog 4 Changes the File Pointer. Catalog 4 changes the file pointer's address (and therefore the current file) as it lists each file name. You can use catalog 4 to change your current file by stopping the catalog listing at the desired file name, then using [←] to exit the catalog. If the catalog runs to completion, the current file will be the same one as before the catalog was run.

DIR EMPTY. If the extended memory directory is empty, executing catalog 4 results in the message DIR EMPTY. When you clear the display (press [←]), the X-register will show 124 registers available for a file.

Keystrokes	Display	
ALPHA SECRETS ALPHA		Type in SECRETS (using the Alpha keyboard).
PURFL		Purges file SECRETS.
ALPHA AREA ALPHA		Use the Alpha keyboard.
PURFL		Purges file AREA.
CATALOG 4	DIR EMPTY	Shows number of extended memory registers now available for your next file.†
←	124.0000	

* File SECRETS with its header takes up 32 registers total and the file AREA with its header takes up 7 registers. The figure 85 is two registers less than the total number remaining since two registers will be taken for a file header for the next file to come.

† This number is higher if you have added HP 82181 Extended Memory Modules.

Possible Error Conditions

- DUP FL occurs if you try to create a new file using a file name that already exists (*duplicate file*).
- FL NOT FOUND occurs if the file specified in the Alpha register does not exist. This error often arises if you mean to access the current file, but forget to clear the Alpha register. The computer will try to use whatever happens to be in the Alpha register.
- FL TYPE ERR if the current file is not the right type (text, data, or program) for the function executed.
- NAME ERR occurs if no file name is given in the Alpha register, or the specified file does not exist. (If a file name is too long, it will simply be truncated to the maximum seven characters.)
- NO ROOM results, and a tone sounds, if you are working in the Text Editor and there is not enough memory space in the text file to add more characters or more records. Refer to the discussion of RESZFL on page 116 to "resize" the file. RESZFL will cause NO ROOM if there is not enough room in extended memory to enlarge a file.
- REC TOO LONG results if you try to type in a record that is longer than the limit of 254 characters. This causes an automatic exit from the Text Editor.

List of Errors

List of Errors

Following is a simplified description of each HP-41CX error message. For complete descriptions of the error conditions, refer to appendix A in volume 2.

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

Error	Meaning
ALPHA DATA	Nonnumeric data used.
CHKSUM ERR	Part of file lost.
DATA ERROR	Illegal operand.
DUP FL	A file of that name already exists.
END OF FL	Pointer at end of file.
END OF REC	Pointer at end of record.
ERROR::Dnn	Number not in time format.
ERROR::Rnn	Number greater than 99.
FL NOT FOUND	Specified file does not exist.
FL SIZE ERR	Invalid file size.
FL TYPE ERR	Invalid file type.
KEYCODE ERR	Nonassignable keycode.
MEMORY LOST	Continuous Memory has been cleared and reset.
NAME ERR	Invalid file name.
NO DRIVE	The necessary device absent.
NONEXISTENT	The register, label, or function specified does not exist.
NO ROOM	Not enough room in memory.
NO SUCH ALM	Alarm 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.
REC TOO LONG	Record too long.
ROM	You cannot access a program in ROM.

Indexes

Subject Index

Page numbers in **bold** type indicate primary references; page numbers in regular type indicate secondary references. Also, page numbers in *italic* type are in volume 1, while page numbers *not* in italic type are in volume 2.

D

E

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.)

Each function has up to three page references. The first one, in *italics*, is for volume 1. The second one is for volume 2. The third one, in **boldface**, is for the Function Tables, a summary in volume 2 of all functions.

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

00041-90474 English - 10K-3.84 - IPC

Printed in Singapore