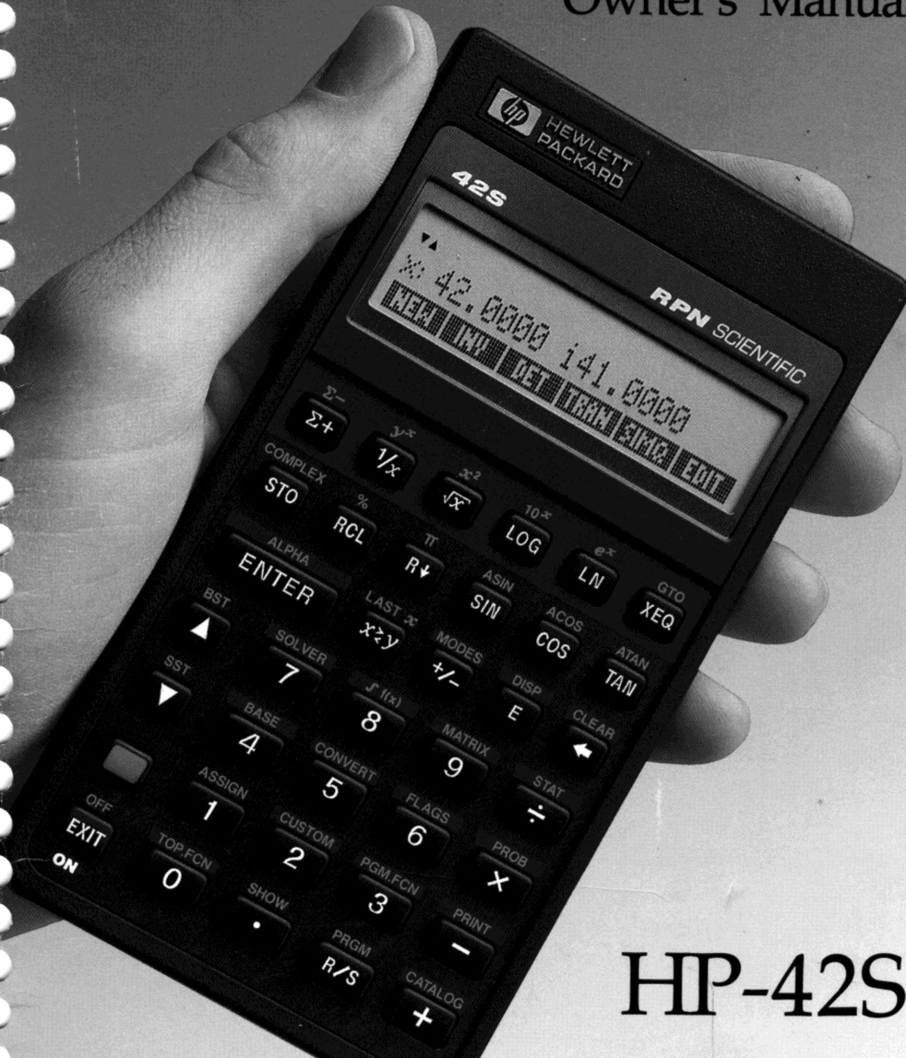
HEWLETT-PACKARD

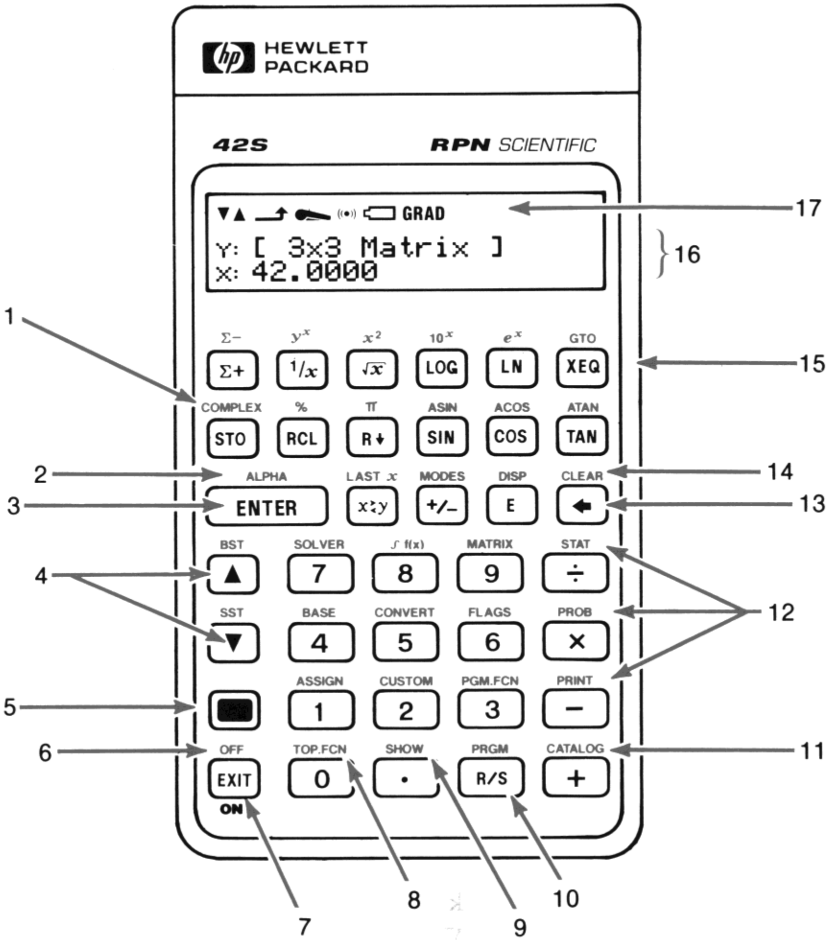
RPN Scientific Calculator

Owner's Manual



HP-42S

HEWLETT PACKARD



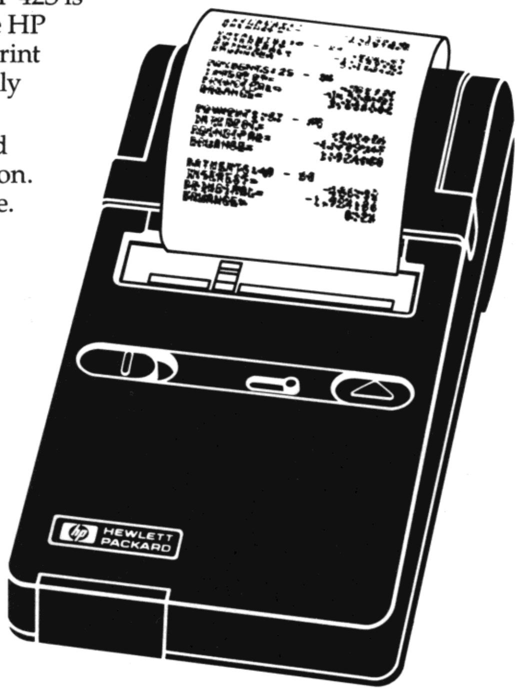
1. Converts to/from complex numbers.
2. Menu for typing characters.
3. Enters a number.
4. Moves up/down through a menu or program.
5. Shift key.
6. Calculator OFF.
7. Exits current menu or mode.
8. Top-row functions.
9. Shows full precision of number.
10. Run/stop program.
11. Catalogs of functions, programs, and variables.
12. Menu-selection keys.
13. Backspace.
14. Functions for clearing.
15. Menu keys (top row).
16. Two-line display.
17. Display annunciators.

HEWLETT-PACKARD

For Your HP-42S: An Invisible Link for Visible Results

An added bonus to using the HP-42S is seeing your work on paper. The HP Infrared Printer (82240A) will print all your steps as you work or only what you tell it to.

The printer is cordless – infrared signals make the print connection. No cords clutter your workspace. Four AA alkaline batteries give the HP Infrared Printer go-anywhere portability. Or, to extend battery life, plug in the optional AC adapter.



Accessories

Printer Adapters

U.S./Canada	82241A
Japan	82241AJ
Europe	82241AB
UK	82241AU
Australia	82241AG

Thermal Printing Paper

Black printing, 6 rolls per box	82175A

Leather Cases

For the HP-42S		*For the HP Infrared Printer*	
Black	92169K	Black	92169G
Brown	92169L	Brown	92169H
Burgundy	92169M	Burgundy	92169J

Application Book for the HP-42S

Programming Examples and Techniques (00042-90020)

- Solving problems in science, engineering and business with these built-in applications:
  - the equation-solving function
  - integration
  - matrices
  - statistics
- Using the equation-solving and integration functions in programs
- Enhancing HP-41 programs for the HP-42S (with examples)
- Building plots and graphics with the HP-42S
- Printing plots and graphics with the HP Infrared Printer

For More Information

In the U.S.A., visit your nearest HP dealer for additional information on calculator accessories or a demonstration of Hewlett-Packard professional calculators. For the location and number of the dealer nearest you, call toll-free 800-752-0900.

For items your local dealer does not carry, and for information about availability and prices of accessories for calculators which are no longer in production, call toll-free 800-538-8787. Please refer to Call Code P280 when ordering. MasterCard, Visa and American Express cards are welcome.

Outside the U.S.A., contact your local HP sales office. It is listed in your telephone directory.

U.S.A.:
Hewlett-Packard Co.
Corvallis Division
1000 NE Circle Blvd.
Corvallis, OR 97330

Canada:
Hewlett-Packard (Canada) Ltd.
6877 Goreway Drive
Mississauga, Ontario
L4V 1M8

Europe, North Africa, Middle East:
Hewlett-Packard S.A.
7, rue du Bois-du Lan
P.O. Box 364
CH-1217 Meyrin 1
Geneva
Switzerland

Other countries:
Hewlett-Packard Intercontinental
3495 Deer Creek Road
Palo Alto, California 94304
U.S.A.

Hewlett-Packard Corporate Offices
3000 Hanover Street
Palo Alto, California 94304
U.S.A.

Note: Product availability subject to change without notice.

HEWLETT PACKARD

Printed in U.S.A. 8/88

HP-42S

Owner's Manual

HEWLETT
PACKARD

Edition 1 June 1988
Reorder Number 00042-90001

Notice

For warranty and regulatory information for this calculator, see pages 262 and 265.

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

Printing History

Edition 1 June 1988 Mfg. No. 00042-90002

Welcome to the HP-42S

Your HP-42S reflects the superior quality and attention to detail in engineering and manufacturing that have distinguished Hewlett-Packard products for nearly 50 years. Hewlett-Packard stands behind this calculator: we offer accessories, worldwide service, and expertise to support its use (see inside the back cover).

Hewlett-Packard Quality

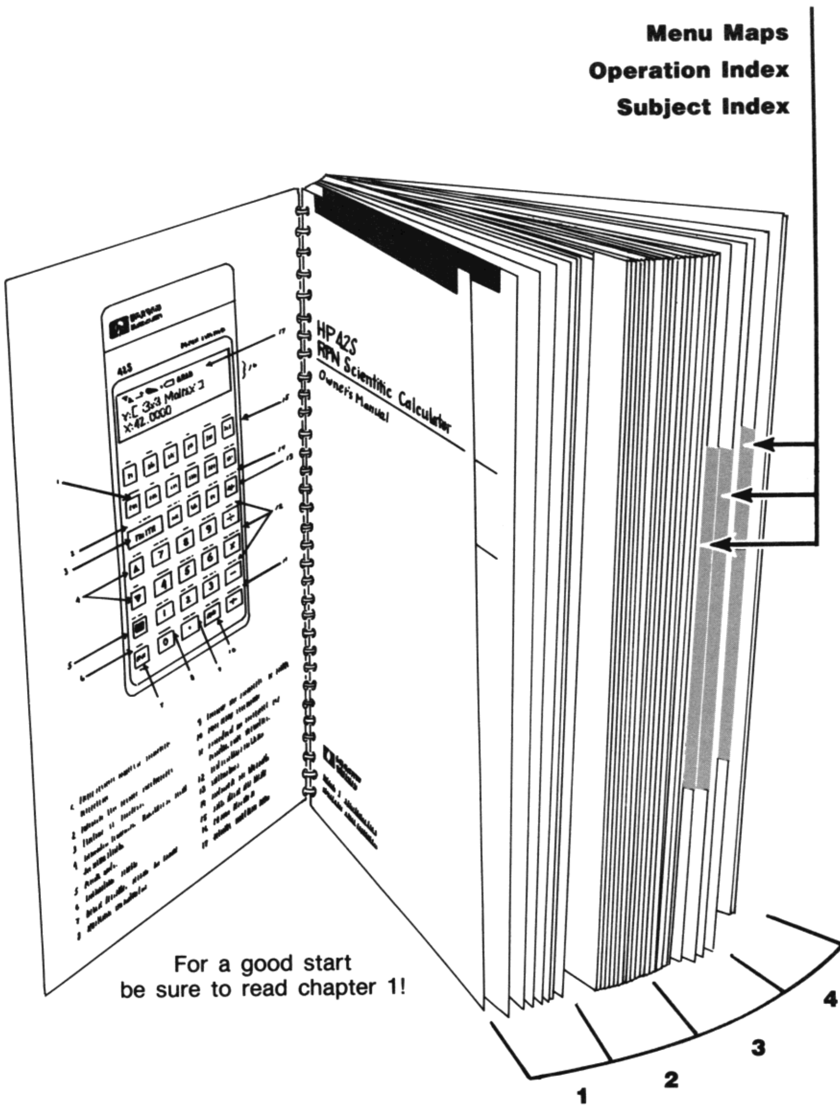
Our calculators are made to excel, to last, and to be easy to use.

- This calculator is designed to withstand the usual drops, vibrations, pollutants (smog, ozone), temperature extremes, and humidity variations that it may encounter in normal, everyday worklife.
- The calculator and its manual have been designed and tested for ease of use. We selected spiral binding to let the manual stay open to any page, and we added many examples to highlight the varied uses of this calculator.
- Advanced materials and permanent, molded key lettering provide a long keyboard life and a positive feel to the keyboard.
- CMOS (low-power) electronics and the liquid-crystal display allow the HP-42S to retain data while it is off and let the batteries last a long time.
- The microprocessor has been optimized for fast and reliable computations. The calculator uses 15 digits internally, then rounds to 12 digits for precise results.
- Extensive research has created a design that has minimized the adverse effects of static electricity, a potential cause of malfunctions and data loss in calculators.

Features

The feature set of this calculator reflects needs and wishes we solicited from customers. The HP-42S features:

- Built-in applications:
  - A solver (root finder) that can solve for any variable in an equation.
  - Numeric integration for calculating definite integrals.
  - Matrix operations, including a Matrix Editor, a solver for simultaneous linear equations, and many other useful matrix functions.
  - Statistical operations, including curve fitting and forecasting.
  - Base conversions, integer arithmetic, and binary manipulation of hexadecimal, decimal, octal, and binary numbers.
- Complex numbers and vector functions.
- Graphic display control functions.
- Menus that can be customized.
- The ability to run programs written for the HP-41C and HP-41CV calculators.
- Over 7,200 bytes of memory for storing programs and data.
- An infrared printer port for printing calculations, programs, data, and graphics using the HP 82240A Infrared Printer.
- Catalogs for reviewing and using items stored in memory.
- An easy-to-use menu system that uses the bottom line of the display to label the top row of keys.
- Reverse Polish Notation (*RPN*) operating logic for the most efficient solutions to complicated problems.
- Keystroke programming with branching, looping, tests, and flags.
- A two-line, 22-character, alphanumeric display with adjustable contrast.



Part 1: Basic Operation

Part 2: Programming

Part 3: Built-In Applications

Part 4: Appendixes and Reference

Contents

Part 1: Basic Operation

Part 2: Programming

Part 3: Built-In Applications

Part 4: Appendixes and Reference

Part 1

Basic Operation

1

Getting Started

This chapter provides you with a detailed orientation to the HP-42S. You'll learn how to:

- Use menus to access calculator functions.
- Clear information from calculator memory.
- Key in numbers and do arithmetic.
- Change the way numbers are displayed.
- Key in alphanumeric data with the ALPHA menu.
- Use catalogs to review the contents of calculator memory.

Important Preliminaries

Power On and Off; Continuous Memory

To turn on the HP-42S, press [EXIT]. Notice that **ON** is printed below the key.

To turn the calculator off, press ■ [OFF]. That is, press and release the shift key, ■, then press [EXIT] (which has OFF printed above it). Since the calculator has *Continuous Memory*, turning it off does not affect any information you've stored.

After about 10 minutes of inactivity, the calculator turns itself off to conserve battery power. When you turn the calculator on again, you can resume working right where you left off.

Under most conditions, calculator batteries last well over a year. If you see the low-battery symbol (▭) in the display, replace the batteries as soon as possible. Refer to appendix A for details and instructions.

Regular and Shifted Keystrokes

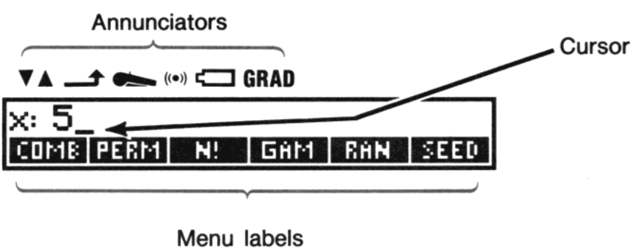
Each key has two functions: one printed on its face, and a *shifted* function printed in color above the key. For example, OFF is the shifted function of the [EXIT] key (written as ■ [OFF]). To execute a shifted function, press ■, then press the key.

Pressing ■ turns on the shift annunciator (⮥), which remains on until you press the next key. To cancel ⮥, just press ■ again.

The ⮥ remains active for as long as you hold down the ■ key. To execute several consecutive shifted functions, hold ■ down and press the appropriate keys.

Annunciators

The calculator uses seven *annunciators* at the top of the display to indicate various conditions.



Annunciator	Meaning
▼▲	The [▼] and [▲] keys are active for moving through a multirow menu (page 23).
⮥	Shift (■) is active.
🖨	The calculator is sending information to the printer (page 100).
((•))	The calculator is busy executing a function or a program.
▭	Battery power is low.
RAD	Radians angular mode is set (page 80).
GRAD	Grads angular mode is set (page 80).

Adjusting the Display Contrast

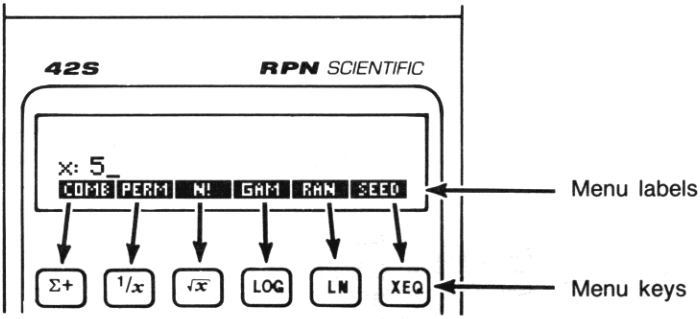
To adjust the contrast of the display for various viewing angles and lighting conditions:

1. Press and hold [EXIT].
2. Press [+] to darken the display, [−] to lighten the display.
3. Release [EXIT].

You can use this sequence at any time without disrupting any other calculator operation.

Using Menus

The top row of keys is very special. In addition to the standard functions printed on the keyboard, these six keys can be redefined by *menu labels* in the display. To execute a function in a menu, press the key directly below the corresponding menu label.



Example: Using a Menu. Use the N! (*factorial*) function in the menu shown above to calculate the factorial of 5 (that is, 5!). Key in 5 and display the PROB (*probability*) menu.

5 ■ [PROB]

```
x: 5.0000
COMB PERM N! GAM RAN SEED
```

To execute the N! function, press the key directly below the menu label ([√x]). This is written as:

[N!]

```
Y: 0.0000
x: 120.0000
```

Thus, 5! = 120.

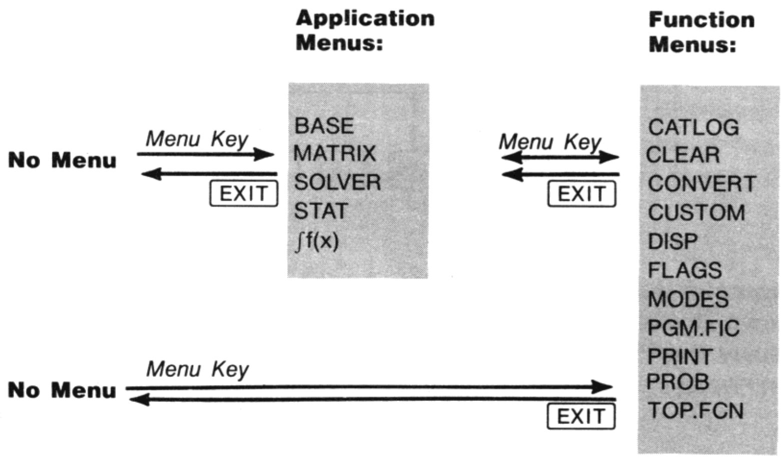
Displaying a Menu

Notice that some of the shifted functions are printed on the keyboard in shaded boxes. These are keys that select menus. When you select a menu with one of these keys, the first row of the menu is immediately displayed.

Application Menus. There are five menu-driven *applications* in the HP-42S. (See the illustration below.) Application menus have top priority among all of the menus. To exit from an application, press [EXIT] or select another application.

Function Menus. The HP-42S has over 350 built-in functions. The most frequently used functions are grouped into *function menus*. In the example above, you used a function menu (■ [PROB]) to execute the N! function.

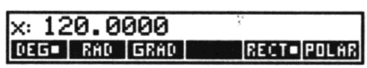
If you select a function menu while in an application, the calculator remembers the application menu and displays it again when you exit the function menu.



Disabling Automatic Exit. Function menus automatically exit as soon as you execute one of the functions in the menu. If you want to use a function menu repeatedly, you can disable automatic exiting by selecting the menu twice. For example, if you press ■ [PROB] ■ [PROB], the PROB menu stays in the display until you press [EXIT] or select another menu.

Menu Labels Marked With "■". There are a variety of modes and settings in the HP-42S. If a menu label contains the ■ character, then that mode or setting is currently selected. For example, display the MODES menu:

■ [MODES]



The menu in this display shows that Degrees (DEG■) and Rectangular (RECT■) modes are selected. (These modes are explained in chapter 5.)

The ALPHA Menu. The ALPHA menu (■ [ALPHA]) is neither an application nor a function menu. It is an extension of the keyboard that allows you to type characters (alphabetic and others) that don't appear on the keyboard. Instructions for using the ALPHA menu are on page 37.

The TOP.FCN Menu. Pressing ■ [TOP.FCN] (*top-row functions*) displays a menu containing the functions (shifted and unshifted) on the six top-row keys:

$\Sigma-$	y^x	x^2	10^x	e^x	GTO
[Σ+]	[1/x]	[$\sqrt{x}$]	[LOG]	[LN]	[XEQ]

Use the TOP.FCN menu when you want to use one of these functions without exiting from the current application menu.

Multirow Menus (▼▲)

Menus with more than six labels are divided into *rows*. If a menu has more than one row, the ▼▲ annunciator appears, indicating that the [▼] and [▲] keys can be used to display the other rows.

For example, the CLEAR menu has two rows. Press ■ [CLEAR] to see the first row:

CLΣ CLP CLV CLST CLA CLX

Press [▼] to display the second row:

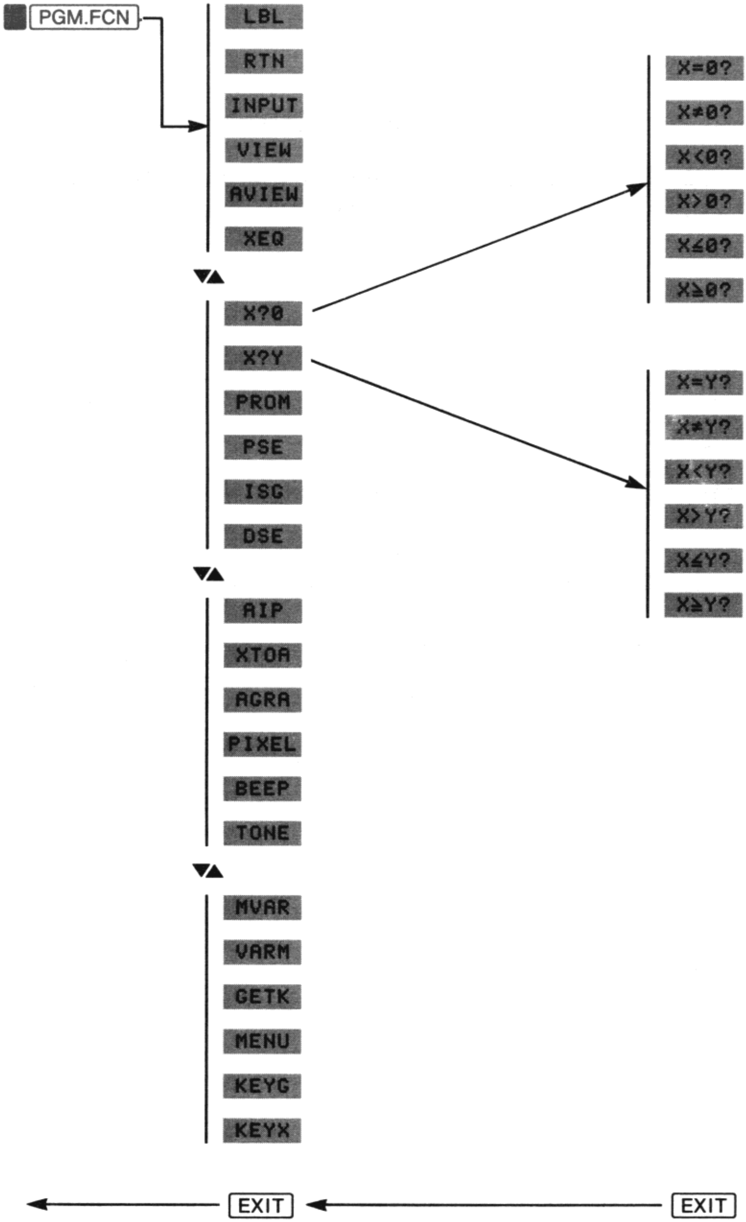
CLRG DEL CLKY CLLCD CLMN CLALL

Because menus are *circular*, pressing [▼] again returns to the first row.

Submenus and [EXIT]

Some menu keys lead to other menus, called *nested* menus or *submenus*. The *menu map* below shows:

- Pressing ■ [PGM.FCN] displays the first of four rows in the PGM.FCN menu.
- Pressing [▼] or [▲] displays the next or previous row (▼▲ is displayed).
- Pressing X?0 or X?Y displays a corresponding submenu.
- Pressing [EXIT] exits the current menu. If it is a submenu, then the previous menu is displayed.


PGM.FCN
LBL
RTN
INPUT
VIEW
AVIEW
XEQ
X?0
X?Y
PROM
PSE
ISG
DSE
AIP
XTOA
AGRA
PIXEL
BEEP
TONE
MVAR
VARM
GETK
MENU
KEYG
KEYX
X=0?
X≠0?
X<0?
X>0?
X≤0?
X≥0?
X=Y?
X≠Y?
X<Y?
X>Y?
X≤Y?
X≥Y?
EXIT
EXIT

Example: Displaying the X?0 Submenu. Display the second row of the PGM.FCN menu.

■ [PGM.FCN]

```
x: 120.0000
LBL  RTN  INPUT VIEW AVIEW XEQ
```

[▼]

```
x: 120.0000
X?0  X?Y  PROM  PSE  ISG  DSE
```

Now display the X?0 submenu.

X?0

```
x: 120.0000
X=0? X≠0? X<0? X>0? X≤0? X≥0?
```

When you exit the submenu, the calculator displays the second row of PGM.FCN again.

[EXIT]

```
x: 120.0000
X?0  X?Y  PROM  PSE  ISG  DSE
```

Press [EXIT] again and the PGM.FCN menu disappears.

[EXIT]

```
y: 0.0000
x: 120.0000
```

Clearing the Calculator

There are several ways to clear information from the calculator. You can clear characters, numbers, variables, programs, or even all of calculator memory with a single operation.

Using the [◆] Key

The [◆] key is a backspace and delete key. The calculator's response when you press [◆] depends on what is in the display.

- If a cursor is present (_), [◆] backspaces and deletes the preceding digit or character.
- If a message is displayed, [◆] clears the message.
- If a number (or other data) is displayed *without a cursor,* [◆] clears the entire number to zero.
- If program lines are displayed, [◆] deletes the current program line. (Program-entry mode is explained in chapter 8.)

The CLEAR Menu

The CLEAR menu contains 12 functions for clearing information from the calculator.

■ [CLEAR] →

CLΣ	*Clear statistics.*
CLP	*Clear program.*
CLV	*Clear variable.*
CLST	*Clear stack.*
CLA	*Clear Alpha register.*
CLX	*Clear X-register* to zero.

▼▲

CLRG	*Clear storage registers.*
DEL	*Delete* program lines.
CLKY	*Clear keys.*
CLLCD	*Clear liquid crystal display.*
CLMN	*Clear MENU.*
CLALL	*Clear all* programs and data.

Clearing All Programs and Data

The CLALL (*clear all*) function clears *all* programs and data from calculator memory but leaves display formats and other settings intact.

1. Press ■ [CLEAR] [▼] CLALL .
2. Press YES to confirm; or any other key to cancel.

A special key sequence can be used to clear all of memory (including modes and flags). Refer to "Clearing All Memory" in appendix B.

Errors and Messages

Whenever you attempt an operation that the calculator cannot complete, it displays a message that specifies the problem. If you're not sure what you've done wrong, refer to appendix D, "Messages."

You do not have to clear the message to continue working—the message disappears as soon as you press a key. If you want to clear the message without altering anything else, press [◄].

Keying In Numbers

If you make a mistake while keying in a number, press [◄] to backspace and delete the last digit, or press ■ [CLEAR] CLX (*clear X-register*) to clear the entire number.

Making Numbers Negative

The [+/-] (*change sign*) key changes the sign of a number.

- To key in a negative number, type the number, then press [+/-].
- To change the sign of a number *already* displayed, just press [+/-].

Exponents of Ten

Numbers with exponents of ten are shown in the display with an E to separate the nonexponent part of the number from the exponent. A number too large or too small for the current display format is automatically displayed in exponential form. For example, the number 123,000,000,000,000 (1.23×10^{14}) is displayed as 1.2300E14.

To key in a number with an exponent:

1. Key in the nonexponent part of the number. If this part is negative, press [+/-].
2. Press [E]. Notice that the cursor follows the E.

3. Key in the exponent. If it is negative, press [+/-]. The largest possible exponent you can key in is ±499 (with one digit to the left of the decimal point).

For example, to key in Planck's constant, 6.6262×10^{-34}, you would press: 6.6262 [E] 34 [+/-].

For a power of ten without a multiplier, such as 10^{34}, just press [E] 34. The calculator automatically inserts a "1" before the exponent: 1E34_.

Other Exponent Functions. To specify an exponent of ten *while entering a number,* use [E]. To *calculate* an exponent of ten (the base 10 antilogarithm), use ■ [10^x]. To *calculate* the result of *any* number raised to a power, use ■ [y^x]. Numeric functions (including ■ [10^x] and ■ [y^x]) are covered in chapter 5.

Understanding Digit Entry

As you key in a number, the *cursor* (_) appears in the display. The cursor shows you where the next digit will go and indicates that the number is not completed yet. That is, when a cursor is present, *digit entry is not terminated.*

- If digit entry *is not* terminated, then [◆] backspaces to erase the last digit.
- If digit entry *is* terminated (no cursor), then [◆] clears the entire number (which is equivalent to ■ [CLEAR] CLX).

Simple Arithmetic

All numeric functions follow one simple rule: *when you press a function key, the calculator immediately executes the function*. Therefore, all operands must be present *before* you execute a function.

Arithmetic can be broken down into two types of functions: one-number functions (such as square root) and two-number functions (such as addition).

Note Many of the displays shown in this manual assume that you've worked the preceding example. Unless indicated otherwise, previous results and the contents of your calculator are irrelevant to the current example.

One-Number Functions

One-number functions operate on the the value in the display (x: *value*). To use a one-number function:

1. Key in the number. (*If the number is already displayed, you can skip this step.*)
2. Press the function key. (The function may be on a normal or shifted key or in a menu.)

For example, to calculate 1/32, key in 32 ...

32

```
Y: 120.0000
X: 32_
```

... then press the function key:

[1/x]

```
Y: 120.0000
X: 0.0313
```

The result (to four decimal places) is 0.0313.

Now calculate $\sqrt{1.5129}$.

1.5129 [√x]

```
Y: 0.0313
X: 1.2300
```

If a number is already in the display, you don't have to key it in again. Calculate the square of 1.23.

■ [x²]

```
Y: 0.0313
X: 1.5129
```

Remember, you can make a number negative at any time with the [+/-] key. Notice that only the number in the bottom line changes.

[+/-]

```
Y: 0.0313
X: -1.5129
```

One-number functions also include the logarithmic functions, the trigonometric functions, the parts-of-numbers functions, and the hyperbolic functions; they are covered in chapter 5.

Two-Number Functions

To use a two-number function (such as [+], [−], [×], or [÷]):

1. Key in the first number.
2. Press [ENTER] to separate the first number from the second.
3. Key in the second number. (Do not press [ENTER] again.)
4. Press the function key.

Remember, both numbers must be present before executing the function.

For example:

To Calculate:	Press:	Result:
12 + 3	12 [ENTER] 3 [+]	15.0000
12 − 3	12 [ENTER] 3 [−]	9.0000
12 × 3	12 [ENTER] 3 [×]	36.0000
12 ÷ 3	12 [ENTER] 3 [÷]	4.0000

The order of entry is essential for noncommutative functions (such as [−] and [÷]). If the numbers have been entered in the wrong order, you can still get the correct answer without reentering the numbers. *Swap* the order of the numbers by pressing [x≷y] (*x exchange y*), then perform the intended function. (Refer also to "Exchanging *x* and *y*" in chapter 2.)

Chain Calculations

The speed and simplicity of calculating with the HP-42S is apparent during *chain calculations* (calculations with more than one operation). Even during the longest of calculations, *you still work with only one or two numbers at a time*—the automatic memory stack stores intermediate results until you need them. (The stack is explained in chapter 2.) The process of working through a problem is the same as working it out on paper, but the calculator does the hard part.

Example: A Chain Calculation. Solve (12 + 3) × 7. To work this problem on paper, you would first calculate the intermediate result of (12 + 3). That is, you would start *inside* the parentheses and work outward.

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

Then you would multiply the intermediate result by 7 to get the final answer.

$$15 \times 7 = 105$$

Solving the problem on the HP-42S uses the same logic. Start inside the parentheses:

12 [ENTER] 3 [+]

```
Y: 4.0000
X: 15.0000
```

This intermediate result is saved automatically—you don't need to press [ENTER]. Simply multiply it by seven.

7 [×]

```
Y: 4.0000
X: 105.0000
```

Example: Another Chain Calculation. Problems that have multiple parentheses can be solved in the same simple manner because intermediate results are automatically remembered. For example, to solve (2 + 3) × (4 + 5) on paper, you would first calculate the values inside parentheses, and then you would multiply them together.

$$\overset{5}{(\cancel{2 + 3})} \times \overset{9}{(\cancel{4 + 5})}$$

Again, working the problem on the HP-42S involves the same logical steps:

2 [ENTER] 3 [+]

```
Y: 105.0000
X: 5.0000
```

4 [ENTER] 5 [+]

```
Y: 5.0000
X: 9.0000
```

Notice that the two intermediate results in the display are the same ones you calculated on paper. Press [×] to multiply them.

[×]

```
Y: 105.0000
X: 45.0000
```

Remember: This method of entering numbers, called Reverse Polish Notation (*RPN*), is unambiguous and therefore does not need parentheses. It has the following advantages:

- You never work with more than two numbers at a time.
- Pressing a function key immediately executes that function so there is no need for an [=] key.
- Intermediate results appear as they are calculated, so you can check each step as you go.
- Intermediate results are automatically stored. They reappear as they are needed for the calculation—the last result stored is the first to come back out.
- You can calculate in the same order as you would with pencil and paper.
- If you make a mistake during a complicated calculation, you don't have to start over. (Correcting mistakes is covered in chapter 2.)
- Calculations with other types of data (such as complex numbers and matrices) follow the same rules.
- Calculations in programs follow the same steps as when you execute them manually.

Exercises: Calculations for Practice

The following calculations exercise the methods you've learned for simple arithmetic. Work each problem in the same order as you would work it on paper. (There may be more than one way to work each problem.) Remember, use [ENTER] only for separating two numbers entered *sequentially*.

Calculate: (2 + 3) ÷ 10
Answer: 0.5000

A Solution: 2 [ENTER] 3 [+] 10 [÷]

Calculate: 2 ÷ (3 + 10)
Answer: 0.1538

A Solution: 3 [ENTER] 10 [+] 2 [x≷y] [÷]

Another Solution: 2 [ENTER] 3 [ENTER] 10 [+] [÷]

Calculate: (14 + 7 + 3 − 2) ÷ 4
Answer: 5.5000

A Solution: 14 [ENTER] 7 [+] 3 [+] 2 [−] 4 [÷]

Calculate: 4 ÷ (14 + (7 × 3) − 2)
Answer: 0.1212

A Solution: 7 [ENTER] 3 [×] 14 [+] 2 [−] 4 [x≷y] [÷]

Another Solution: 4 [ENTER] 14 [ENTER] 7 [ENTER] 3 [×] [+] 2 [−] [÷]

Range of Numbers

The HP-42S is capable of representing numbers as large as $9.99999999999 \times 10^{499}$ and as small as 1×10^{-499}. If you attempt to execute a function that returns a result larger than $9.99999999999 \times 10^{499}$, the calculator displays the `Out of Range` error message. The operation you attempted is ignored, and the message disappears when you press the next key.

If you attempt an arithmetic function that returns a number whose magnitude is smaller than 1×10^{-499}, the calculator automatically substitutes the number zero.

Changing the Display Format

Internally, the HP-42S *always* saves numbers with full 12-digit accuracy plus a three-digit exponent of ten.

Even though numbers are stored with full precision, the way they're displayed depends on the current display format. There are two general ways to display numbers:

- Round the number to a specified number of digits. There are three formats that do this: FIX (*fixed-decimal notation*), SCI (*scientific notation*), and ENG (*engineering notation*).
- Show all of the digits in a number (except trailing zeros). This is the ALL format.

In addition to controlling how digits are displayed, you can select the character used as the decimal point—called the *radix*. The radix may be a period (default) or a comma.

Functions for changing the display format are in the DISP (*display*) menu:

■ DISP →

FIX	*Fixed-decimal* notation.
SCI	*Scientific* notation.
ENG	*Engineering* notation.
ALL	*All* notation.
RDX.	*Radix period.*
RDX,	*Radix comma.*

Number of Decimal Places

The default display format is FIX 4. (The calculator displays numbers rounded to four places to the right of the decimal.)

To change the number of decimal places:

1. Press ■ [DISP].
2. Press: FIX, SCI, ENG, or ALL.
3. For FIX, SCI, and ENG, specify the number of digits (0 through 11):
   - Key in two digits (such as 02).
   - Or, key in a single digit followed by [ENTER] (such as 2 [ENTER]).

Example: Changing the Display Format. Key in the numbers 2.46×10^7 and 1234567.89, and then change the display format to ENG 2.

Keys	Display
2.46 [E] 7 [ENTER] 1234567.89	Y: 24,600,000.0000 X: 1,234,567.89_
■ [DISP] ENG 2 [ENTER]	Y: 24.6E6 X: 1.23E6

Now change to the ALL display format.

Keys	Display
■ [DISP] ALL	Y: 24,600,000 X: 1,234,567.89

Now return to the default setting (FIX 4).

Keys	Display
■ [DISP] FIX 4 [ENTER]	Y: 24,600,000.0000 X: 1,234,567.8900

Fixed-Decimal Notation (FIX). In FIX notation, the calculator displays numbers rounded to the specified number of decimal places. Exponents of 10 are used only if the number is too large or too small to display using the current display format. (Example: 3.1416.)

Scientific Notation (SCI). In SCI notation, the calculator displays numbers with one digit to the left of the decimal point and the specified number of digits to the right. An exponent of 10 is always shown; even if it is zero. (Example: 6.0220E26.)

Engineering Notation (ENG). In ENG notation, the calculator displays numbers in a format similar to SCI except the exponent of 10 is always a multiple of three. This means that more than one digit may appear to the left of the decimal point. The number of digits you specify indicates how many digits to display after the first digit. (Example: `10.423E-3`.)

All Notation (ALL). In ALL notation, the calculator displays numbers using full precision. That is, all significant digits to the right of the decimal point are shown. (Example: `4.17359249`.)

Selecting the Radix Mark (Comma vs. Period)

To change the radix mark to a comma, press ■ [DISP] RDX, . When the radix is a comma, periods are used to separate digits.

`1.234.567,8900`

To change the decimal point back to a period, press ■ [DISP] RDX. .

`1,234,567.8900`

You can remove the digit separators by clearing flag 29 (page 276).

Showing All 12 Digits

When you press and hold the ■ [SHOW] key, the calculator displays the contents of the X-register using the ALL format—that is, all significant digits are shown. When you release the key, the display returns to the current display format.

1.23456789012 [ENTER]	Y: 1.2346 X: 1.2346
■ [SHOW] *(hold down)*	1.23456789012
(release)	Y: 1.2346 X: 1.2346

The ■ [SHOW] key can also be used to show the entire contents of the Alpha register (page 40), a long program line (page 111), or the first element in a matrix (page 207).

Keying In Alphanumeric Data

Alphabetic and other characters are typed into the HP-42S using the ALPHA menu, which contains all the letters in the alphabet (uppercase and lowercase) and many other characters.

One or more characters typed with the ALPHA menu form an *Alpha string*.

Using the ALPHA Menu

To type a string of characters into the Alpha register:

1. Press ■ [ALPHA] to select the ALPHA menu.
2. Press an ALPHA menu key to select a *group* of letters or characters.
3. Press a menu key to type a character. To type a lowercase letter, press ■ before typing the letter.

Repeat steps 2 and 3 for each letter or character. You can also use the following keys to type Alpha characters: ■ [%], ■ [π], [E], [÷], [×], [−], [+], [0], [1], [2], [3], [4], [5], [6], [7], [8], [9], and [·].

Example. The keystrokes for typing the string `The HP-42S.` are:

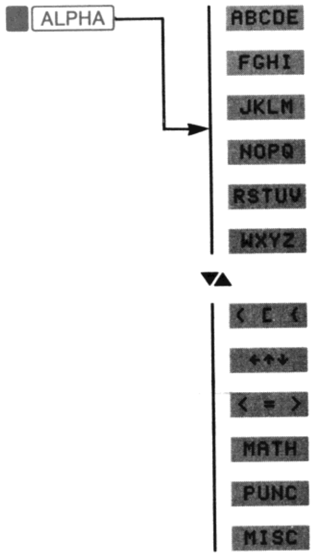
■ [ALPHA] [RSTUV] [T] ■ (*hold down*) [FGHI] [H] [ABCDE] [E] (*release* ■) [WXYZ] [ ] [FGHI] [H] [NOPQ] [P] [−] [4] [2] [RSTUV] [S] [·]

For simplicity, this manual shows these keystrokes as:

■ [ALPHA] The HP−42S.

ALPHA Typing Tips:

- Any blank menu key in the ALPHA menu can be used to type a space character. A rapid sequence for typing a space is [XEQ] [XEQ] (that is [WXYZ] [ ] or [MISC] [ ]).
- To type several lowercase letters, hold down the shift key (■) while typing.



The characters in each of the submenus are shown in the menu maps beginning on page 292.

The Alpha Display and the Alpha Register

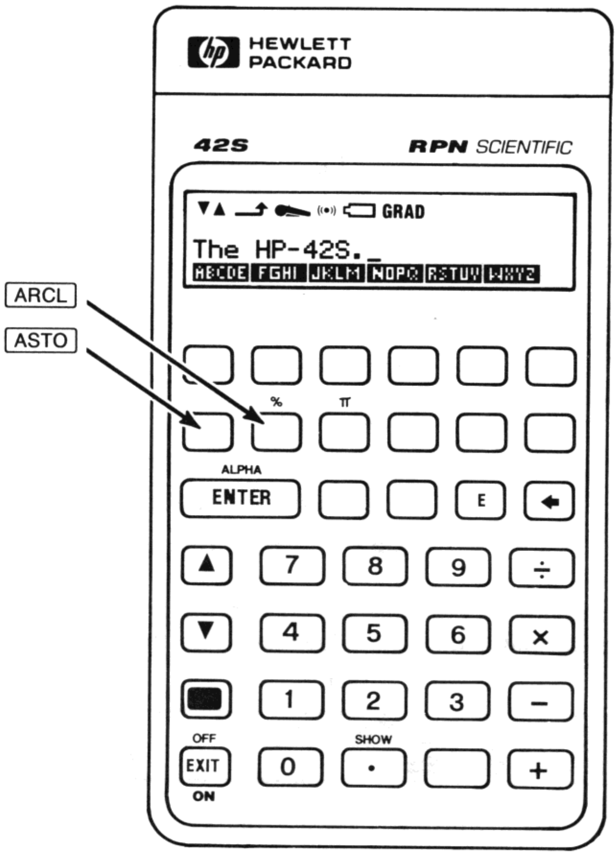
Alpha strings can be typed only when the ALPHA menu is displayed. How strings are used or where they are stored, however, depends on other circumstances. Alpha strings can be:

- Typed directly into the Alpha register.
- Used as a function parameter to specify a variable name or program label (page 73).
- Entered as program instructions (page 130).

Alpha Mode: Entering Characters Into the Alpha Register. In the previous example, the Alpha characters were entered into the *Alpha register*. When you press ■ [ALPHA], the calculator displays the ALPHA menu *and* the Alpha register—this is *Alpha mode.*

If there are characters in the Alpha register, they are displayed when you enter Alpha mode. The Alpha register is cleared when you begin typing. To append characters to the current contents of Alpha, press [ENTER] to turn the cursor on before you begin to type.

The following illustration shows the keys that are active in Alpha mode.



Capacity of the Alpha Register. The Alpha register can hold up to 44 characters. The calculator beeps when the Alpha register gets full. The beep warns you that each additional character you type will push the first (left-most) character out of the Alpha register.

If the display overflows, the … character indicates there are some characters you can't see.

To display the entire Alpha register:

- While in Alpha mode, press and hold ■[SHOW].
- While not in Alpha mode, press ■[PGM.FCN] AVIEW (*Alpha view*).

Printing the Alpha Register. To print the contents of the Alpha register, press ■[PRINT] PRA (*print Alpha*). For more information on printing, refer to chapter 7.

Catalogs

Catalogs are used to view the contents of calculator memory. You can also use a catalog to execute functions or programs or recall variables.

■[CATALOG] →

FCN	*Functions.*
PGM	*Programs.*
REAL	*Real variables.*
CPX	*Complex variables.*
MAT	*Matrix variables.*
MEM	*Memory available.*

To display the amount of available memory, press and hold the MEM key. The calculator displays a message like this:

```
Available Memory:
6836 Bytes
```

The message disappears when you release the key.

An Introduction to Flags

Throughout the rest of this manual there are references to numbered *flags*. A flag has two states, *set* and *clear*. If you are unfamiliar with flags, simply think of them as switches that are either on or off.

The HP-42S has 100 flags (numbered 00–99); most of them have special purposes inside the calculator. To set, clear, and test the status of flags, use the functions in the FLAGS menu:

■ [FLAGS] →

SF	*Set flag.*
CF	*Clear flag.*
FS?	*Flag set test.*
FC?	*Flag clear test.*
FS?C	*Flag set test and clear.*
FC?C	*Flag clear test and clear.*

For more information on flags, refer to appendix C.

2

The Automatic Memory Stack

This chapter explains how calculations take place in Hewlett-Packard's automatic memory stack and how it minimizes the number of keystrokes required to do complicated calculations.

More specifically, you will learn:

- What the stack is.
- How the stack automatically remembers results from previous calculations.
- What is meant by *stack lift* and *stack drop*.
- How to view and manipulate the contents of the stack.
- How to save keystrokes and correct mistakes with ■ [LASTx].

You do not need to read and understand this chapter to use the HP-42S. However, you'll find that understanding this material will greatly enhance your use of the calculator. In programs, efficient use of the stack saves memory by reducing the number of program steps needed to solve a problem.

What the Stack Is

Automatic storage of intermediate results is the reason the HP-42S easily processes complex calculations, and does so without parentheses. The key to automatic storage is the *automatic, RPN memory stack.**

* HP's operating logic is based on a mathematical logic known as "Polish Notation," developed by the Polish logician Jan Łukasiewicz (1878—1956). While conventional algebraic notation places the operators *between* the relevant numbers or variables, Łukasiewicz's notation places them *before* the numbers or variables. For optimal efficiency of the stack, we have modified that notation to specify the operators *after* the numbers. Hence the term *Reverse Polish Notation*, or *RPN*.

The stack consists of four storage locations, called *registers*, which are "stacked" on top of each other. It is a work area for calculations. These registers—labeled X, Y, Z, and T—store and manipulate four current numbers. The "oldest" number is the one in the T-register (*top*).

T	0.0000
Z	0.0000
Y	0.0000
X	0.0000

The most "recent" number is in the X-register and is usually displayed.

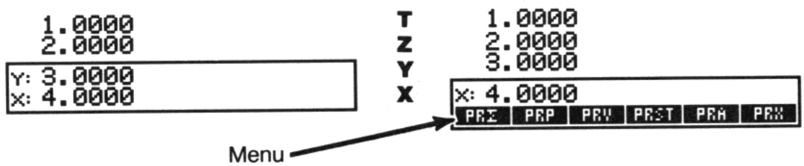
You might have noticed that several functions' names include an *x* or *y*. These letters refer to the values in the X- and Y-registers. For example, ■ [y^x] raises the number in the Y-register to the power of the number in the X-register.

To clear all four of the stack registers to zero, press ■ [CLEAR] **CLST** .

Note Each stack register can hold any type of data (a real number, Alpha string, complex number, or matrix). Examples in this chapter use real numbers; however, the stack works the same regardless of the type of data it contains.

The Stack and the Display

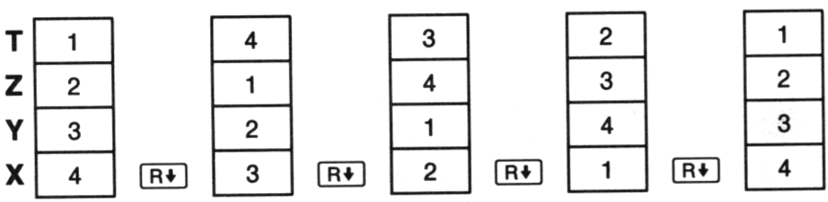
Since the HP-42S has a two-line display, it is capable of displaying two numbers (*x* and *y*) or one number (*x*) and a menu.



Reviewing the Stack ([R↓])

The [R↓] (*roll down*) key lets you review the entire contents of the stack by "rolling" the contents downward, one register at a time.

Suppose the stack is filled with 1, 2, 3, 4 (press 1 [ENTER] 2 [ENTER] 3 [ENTER] 4). Pressing [R↓] four times rolls the numbers all the way around and back to where they started:



Notice that the *contents* of the registers are shifted—the registers themselves maintain their positions.

Exchanging x and y ([x⇄y])

Another key for manipulating the contents of the stack is [x⇄y] (*x exchange y*). It swaps the contents of the X- and Y-registers without affecting the rest of the stack. The [x⇄y] function is generally used to swap the order of numbers for a calculation.

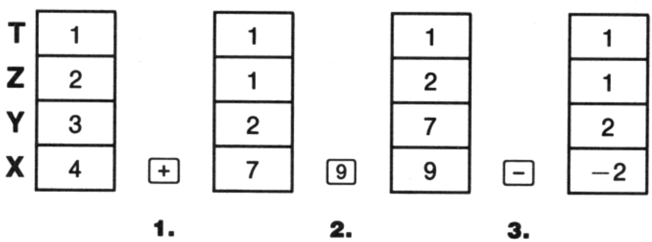
To calculate 9 ÷ (13 + 8) you might press 13 [ENTER] 8 [+] 9 [x≷y] [÷]. The [x≷y] function swaps the two numbers so they are in the correct order for division.

Arithmetic—How the Stack Does It

The contents of the stack move up automatically as new numbers enter the X-register (*lifting the stack*). The contents of the stack automatically move down when a function replaces two numbers (x and y) with a single result in the X-register (*dropping the stack*).

Suppose the stack is still filled with the numbers 1, 2, 3, and 4. See how the contents of the stack lift and drop while calculating

$$3 + 4 - 9.$$



1. The stack "drops" its contents. (The top register replicates its contents.)

2. The stack "lifts" its contents. (The top contents are "lost".)

3. The stack drops.

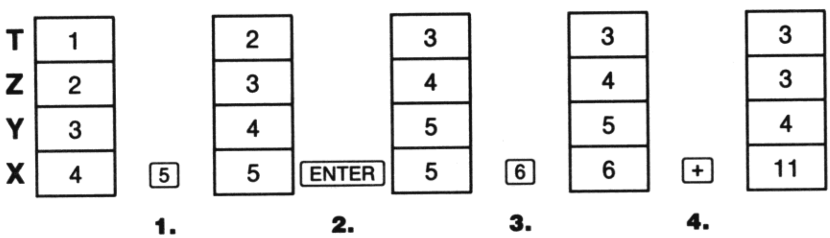
- Notice that when the stack lifts, numbers are pushed off the top of the stack (out of the T-register) and are lost. Therefore, the stack memory is limited to four numbers.
- Because of the automatic movement in the stack, you do not need to clear the display before starting a new calculation. "Old" results are just pushed up the stack.

■ Generally, keying in a number causes the stack to lift. However, there are four functions that specifically *disable stack lift*. They are [ENTER], CLX *, [Σ+], and ■[Σ−]. That is, a number keyed in immediately after one of these functions *replaces* the number in the X-register rather than pushing it up.

How [ENTER] Works

In chapter 1 you learned that [ENTER] separates two numbers keyed in one after the other. In terms of the stack, how does it do this? Suppose the stack is again filled with 1, 2, 3, and 4. Now enter and add two new numbers:

$$5 + 6$$



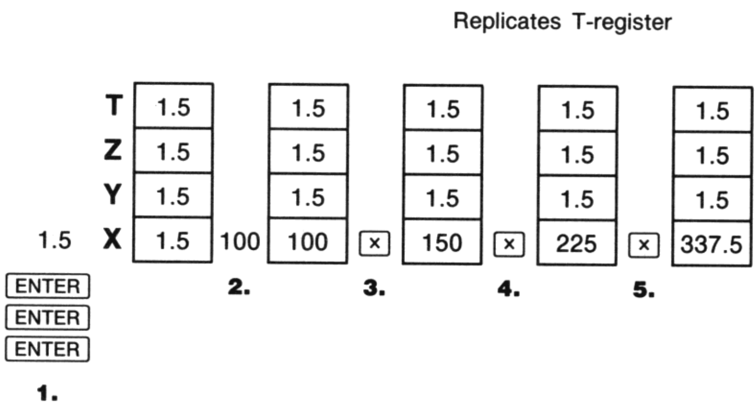
1. Lifts the stack.
2. Lifts the stack and replicates the X-register.
3. Does *not* lift the stack.
4. Drops the stack and replicates the T-register.

[ENTER] copies the contents of the X-register into the Y-register and *disables stack lift* so that the second number you enter *writes over* the copy of the first number in the X-register. The effect is simply to separate two numbers entered sequentially.

* Remember, the [←] key sometimes functions as CLX. Refer to "Using the [←] Key" on page 25.

Filling the Stack With a Constant. Whenever the stack drops, the number in the T-register is duplicated in the Z-register. Therefore, you can completely fill the stack with a constant number and use that number repeatedly in calculations. Every time the stack drops, the constant is duplicated at the top of the stack.

Example: Constant, Cumulative Growth. Given a bacterial culture with a growth rate of 50% per day, how large would a population of 100 be at the end of 3 days?



1. Fills the stack with the growth rate.
2. Keys in the initial population.
3. Calculates the population after 1 day.
4. Calculates the population after 2 days.
5. Calculates the population after 3 days.

Other Uses of the [ENTER] Key. The primary purpose of the [ENTER] key is to separate two numbers entered sequentially for a calculation. [ENTER] can also be used to:

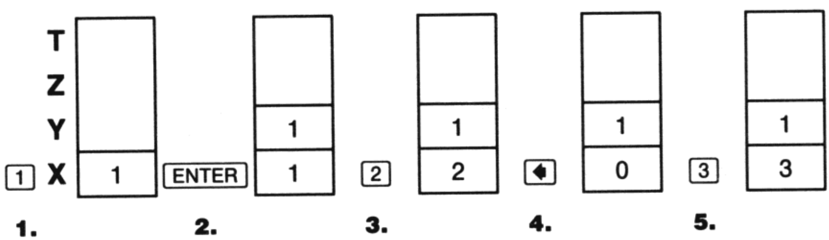
- Turn the cursor on or off in Alpha mode.
- Select the ALPHA menu when a function is prompting for a parameter.
- Complete an instruction after keying in a parameter.

How CLX Works

To prevent an unwanted zero from being added to the stack, the CLX function (and [◄] when it clears the X-register) disables stack lift. That is, CLX puts a zero in the X-register, but the next number entered writes over the zero.

This feature lets you correct mistakes without interfering with the current calculation. Since stack lift does not occur, the contents of the Y-, Z-, and T-registers are left unchanged.

For example, suppose you wanted to enter 1 and 3 but mistakenly entered 1 and 2. This is what you would do:



1. Lifts the stack.
2. Lifts the stack and replicates the X-register.
3. Overwrites the X-register.
4. Clears *x* by overwriting it with zero.
5. Overwrites *x* (replaces the zero.)

The LAST X Register

The LAST X register is a companion to the stack—it holds the contents of the X-register used in the most recent numeric function. Pressing ■[LASTx] recalls this value into the X-register. This ability to retrieve the "last *x*" has two main uses: correcting mistakes and reusing a number in a calculation.

Using ■ [LASTx] To Correct Mistakes

Wrong One-Number Function. If you execute the wrong one-number function, use ■ [LASTx] to retrieve the number so you can execute the correct function.

If you are in the middle of a chain calculation when you make the mistake, clear the X-register ([◆]) before executing ■ [LASTx]. This clears the incorrect result and disables stack lift so that intermediate results in the stack are not lost.

Example. Suppose that you had just calculated $4.7839^3 \times (3.879 \times 10^5)$ and wanted to find its square root ([√x]) but pressed [LOG] by mistake. You don't have to start over! To find the correct result, just press [◆] ■ [LASTx] [√x]. ([◆] is needed only if you want to prevent the incorrect result from being lifted into the Y-register.)

Mistakes With Two-Number Functions. If you make a mistake with a two-number function, you can correct it by using ■ [LASTx] and the *inverse* of the two-number function.

For mistakes with the *wrong function* or *wrong second number:*

1. Press ■ [LASTx] to recover the second number (the one in the X-register just before the operation).
2. Execute the inverse operation. (For example, [−] is the inverse of [+] and [÷] is the inverse of [×].) This returns the number that was originally first. The second number is still in the LAST X register.
3. Execute the correct calculation:
   - If you had used the *wrong function*, press ■ [LASTx] again to restore the original stack contents. Now execute the correct function.
   - If you had used the *wrong second number*, key in the correct one and then execute the function.

For mistakes with the *wrong first number:*

1. Key in the correct first number.
2. Press ■ [LASTx].
3. Execute the function again.

If the contents of the other stack registers are important, clear the X-register *first* to prevent the incorrect result from being lifted into the stack.

Example. Suppose you made an error while calculating

$$16 \times 19 = 304.$$

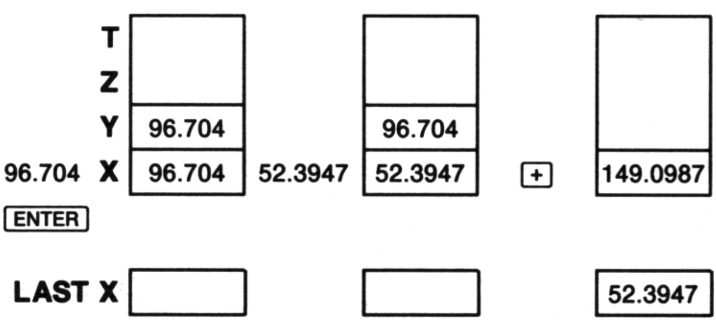
There are three kinds of mistakes you could have made:

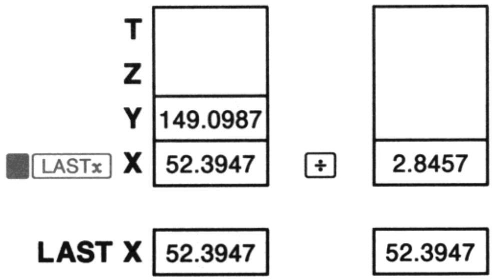
Wrong Calculation	Mistake	Correction
16 [ENTER] 19 [−]	Wrong function.	■ [LASTx] [+] ■ [LASTx] [×]
16 [ENTER] 18 [×]	Wrong second number.	■ [LASTx] [÷] 19 [×]
15 [ENTER] 19 [×]	Wrong first number.	16 ■ [LASTx] [×]

Using ■ [LASTx] To Reuse Numbers

Recovering and reusing a number can be useful in short calculations that use the same number more than once. Since ■ [LASTx] recovers the last value that was used in a calculation, you can reuse the same number. Often, pressing ■ [LASTx] is quicker than keying the number in again.

Example. Calculate (96.704 + 52.3947) ÷ 52.3947. Remember to enter 52.3947 second so it can be reused.





96.704 [ENTER]	Y: 96.7040 X: 96.7040
52.3947 [+]	Y: 0.0000 X: 149.0987
■ [LASTx]	Y: 149.0987 X: 52.3947
[÷]	Y: 0.0000 X: 2.8457

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

Enter the distance to Rigel Centaurus and multiply by the speed of light.

4.3 [ENTER] 9.5 [E] 15 [×]	Y: 2.8457 X: 4.0850E16

The distance to Rigel Centaurus is 4.085×10^{16} meters.

Now, enter the distance to Sirius and recall the speed of light from the LAST X register.

8.7 ■ [LASTx]

```
Y: 8.7000
X: 9.5000E15
```

Multiply to get the distance.

[×]

```
Y: 4.0850E16
X: 8.2650E16
```

The distance to Sirius is 8.265×10^{16} meters.

Chain Calculations

The automatic lifting and dropping of the stack's contents let you retain intermediate results without storing or reentering them, and without using parentheses.

Order of Calculation

In chapter 1 we recommended solving chain calculations by working from the innermost parentheses outward. You may choose to work problems in a left-to-right order. (However, since the stack can only hold four numbers at a time, some expressions may be too long to calculate from left to right.)

For example, in chapter 1 you calculated:

$$4 \div [14 + (7 \times 3) - 2]$$

by starting with the innermost parentheses (7×3) and working outward—just as you would with pencil and paper. The keystrokes were:

7 [ENTER] 3 [×] 14 [+] 2 [−] 4 [x≷y] [÷].

Working the problem from left-to-right, the solution would be:

4 [ENTER] 14 [ENTER] 7 [ENTER] 3 [×] [+] 2 [−] [÷],

which takes one additional keystroke. Notice that the first intermediate result is still the innermost parentheses: (7 × 3). The advantage of working a problem from left-to-right is that you don't have to use [x≷y] to reposition operands for noncommutative functions ([−] and [÷]).

The first method (starting with the innermost parentheses) is often preferred because:

- It takes fewer keystrokes.
- It requires fewer registers in the stack.

In summary, the stack gives you the flexibility to work problems in an order that best fits *your* needs.

Exercises: More RPN Calculations

Here are some additional problems that you can work for more practice using RPN. As demonstrated above, there's more than one way to solve most problems. Therefore, the solutions shown below are not necessarily unique.

Calculate: (14 + 12) × (18 − 12) ÷ (9 − 7)
Answer: 78.0000

A Solution: 14 [ENTER] 12 [+] 18 [ENTER] 12 [−] [×] 9 [ENTER] 7 [−] [÷]

Another Solution: 14 [ENTER] 12 [+] 18 ■ [LASTx] [−] [×] 9 [ENTER] 7 [−] [÷]

Calculate: $23^2 - (13 \times 9) + 1/7$
Answer: 412.1429

A Solution: 23 ■ [x^2] 13 [ENTER] 9 [×] [−] 7 [1/x] [+]

Another Solution: 23 [ENTER] [×] 13 [ENTER] 9 [×] [−] 7 [1/x] [+]

Calculate: $\sqrt{(5.4 \times 0.8) \div (12.5 - 0.7^3)}$
Answer: 0.5961

A Solution: 5.4 [ENTER] .8 [×] .7 [ENTER] 3 ■ [y^x] 12.5 [x≷y] [−] [÷] [√x]

Another Solution: 5.4 [ENTER] .8 [×] 12.5 [ENTER] .7 [ENTER] 3 ■ [y^x] [−] [÷] [√x]

Calculate: $\sqrt{\dfrac{8.33 \times (4 - 5.2) \div [(8.33 - 7.46) \times 0.32]}{4.3 \times (3.15 - 2.75) - (1.71 \times 2.01)}}$

Answer: 4.5728

A Solution: 4 [ENTER] 5.2 [−] 8.33 [×] ■ [LASTx] 7.46 [−] .32 [×] [÷] 3.15 [ENTER] 2.75 [−] 4.3 [×] 1.71 [ENTER] 2.01 [×] [−] [÷] [√x]

3

Variables and Storage Registers

In the previous chapter, you learned how the calculator's stack provides temporary storage during calculations. For more permanent data storage, you can use variables and storage registers. In this chapter you will learn how to use [STO] (*store*) and [RCL] (*recall*) to:

- Copy data between the stack and variables or storage registers.
- Perform arithmetic with variables and registers.
- Directly access each of the stack registers.

In addition, you'll see how the [ASTO] (*Alpha store*) and [ARCL] (*Alpha recall*) functions are used to copy data between the Alpha register and variables or registers.

Storing and Recalling Data

The X-register is used in all store and recall operations. [STO] copies data *from* the X-register into a variable or register. [RCL] recalls data *into* the X-register from a variable or register.

When you press [STO] or [RCL], the calculator displays a prompt (STO __ or RCL __) and a menu of variable names. To complete the instruction, you must supply one of the following parameters to indicate what you want to store or recall:

- A variable name.
- A storage register number.
- A stack register.

Variables

Variables are *named* storage locations. Each variable can hold any type of data, from a single number to a large, two-dimensional matrix of complex numbers. The number of variables stored in the calculator is limited only by the amount of memory available.

To store data into a variable:

1. Press [STO].
2. Select the variable from the catalog (automatically displayed), or type the variable name using the ALPHA menu:

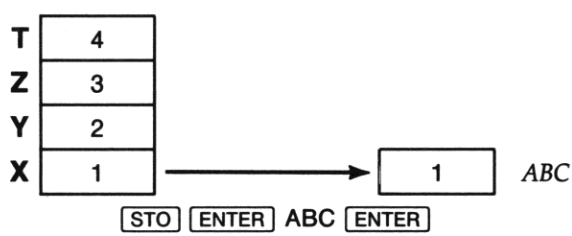
   - *Using the variable catalog:* If the variable name you want already exists, press the corresponding menu key. Data previously stored in the variable is overwritten with the new data.
   - *Using the ALPHA menu:*

     a. Press [ENTER] or ■[ALPHA] to select the ALPHA menu.

     b. Type the variable name (one to seven characters).*

     c. Press [ENTER] or ■[ALPHA] to complete the name.

For example, to store a copy of the X-register into a variable named *ABC*, press [STO] [ENTER] ABC [ENTER]. If *ABC* already exists, press [STO] ABC .

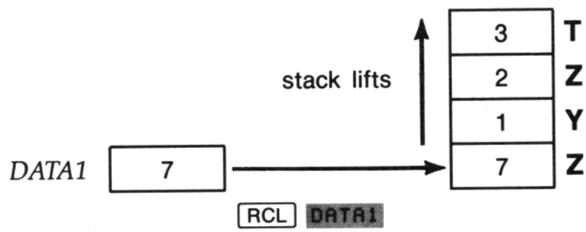


To recall data from a variable:

1. Press [RCL].
2. Select the variable from the catalog, or type the variable name using the ALPHA menu. (Refer to step 2 above.)

* Instructions for using the ALPHA menu are on page 37.

For example, to recall a copy of the data in the variable *DATA1*, press [RCL] DATA1 (assuming *DATA1* already exists).



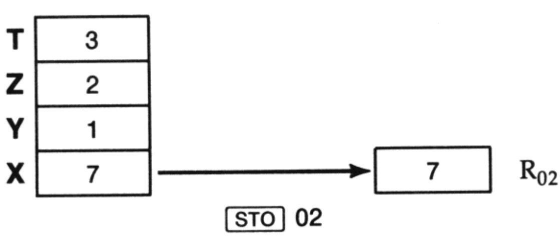
Storage Registers

Storage registers are numbered storage locations that each hold a single number. Initially, the HP-42S has 25 storage registers (designated R_{00}–R_{24}), each containing a zero. You can change the number of storage registers with the SIZE function (page 64).

To store data into a storage register:

1. Press [STO].
2. Key in the register number: two digits *or* a single digit followed by [ENTER]. Data previously stored in the register is overwritten with the new data.

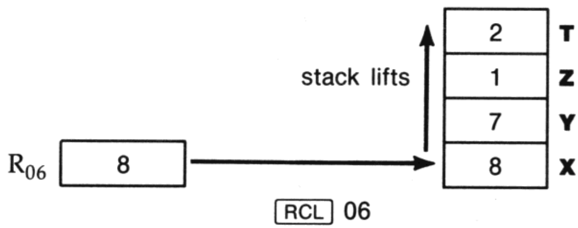
For example, to store a copy of the number in the X-register into R_{02}, press [STO] 02 *or* [STO] 2 [ENTER].



To recall data from a storage register:

1. Press [RCL].
2. Key in the register number: two digits *or* a single digit followed by [ENTER].

For example, to recall a copy of the number in R_{06}, press [RCL] 06 *or* [RCL] 6 [ENTER].



Storing and Recalling Stack Registers

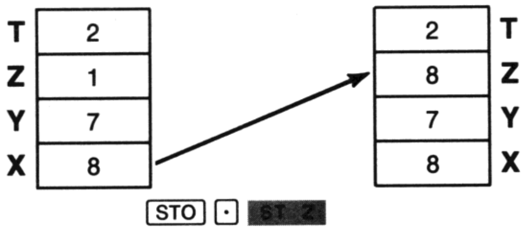
You can store and recall data directly to registers in the stack using *stack addressing*.

To store data directly into a stack register:

1. Press [STO].
2. Press [·] to display the stack menu.
3. Press *one* of the following menu keys:
   - ST L to copy the data into the LAST X register.
   - ST X to copy the data into the X-register.*
   - ST Y to copy the data into the Y-register.
   - ST Z to copy the data into the Z-register.
   - ST T to copy the data into the T-register.

* Although [STO] [·] ST X is a valid instruction, storing a copy of the X-register into itself is of little value.

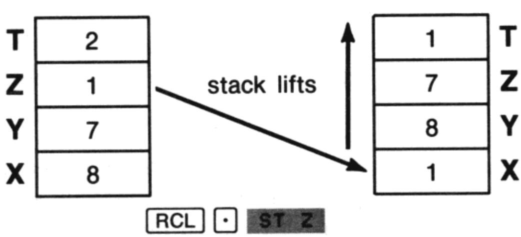
For example, to copy the data in the X-register into the Z-register, press [STO] [·] ST Z.



To recall data directly from a stack register:

1. Press [RCL].
2. Press [·] to display the stack menu.
3. Press *one* of the following menu keys:
   - ST L to copy data from the LAST X register (equivalent to executing ■ [LASTx].)
   - ST X to copy data from the X-register. (This is similar to executing [ENTER] except that stack lift is enabled.)
   - ST Y to copy data from the Y-register
   - ST Z to copy data from the Z-register.
   - ST T to copy data from the T-register (equivalent to executing the R↑ function).

For example, to recall the data in the Z-register into the X-register, press [RCL] [·] ST Z.



Data Types

The HP-42S uses four data types. You can identify a type of data by the way it is displayed:

- *Real numbers* are displayed using the current display format. Some real numbers are displayed with exponents of 10.

  Examples: `1,024.0000`
  `3.1600E4`

- *Complex numbers* are displayed in two parts, separated with `i` or `∡` (depending on the current coordinate mode). If a complex number is too long to be displayed in the current display mode, it is automatically displayed in ENG 2 format.

  Examples: `12.1314 i15.1617` (Rectangular mode)
  `55.0300 ∡90.0000` (Polar mode)

- *Alpha strings* (in the stack) are displayed with surrounding quotation marks. The quotation marks are not part of the string.

  Examples: `"String"`
  `"JIM"`

- *Matrices* are displayed with brackets (`[` and `]`). The dimensions of the matrix are shown (*rows* × *columns*) and complex matrices are indicated with `Cpx`.

  Examples: `[ 3×2 Matrix ]`
  `[ 5×7 Cpx Matrix]`

Where Data Can Be Stored. You can store any type of data into a stack register (X, Y, Z, T, or LAST X) or variable. However, individual storage registers may only contain a single number. That is, you cannot store a matrix into a storage register. Further, you cannot store a complex number into a storage register unless the entire set of registers is converted to complex (page 98).

An Alpha string (up to six characters) can be stored into a variable, stack register, or storage register. Each element in a real matrix may also contain an Alpha string. (Alpha strings are not allowed in complex matrices.)

Arithmetic With [STO] and [RCL]

By combining [STO] and [RCL] with the basic arithmetic operators ([+], [−], [×], and [÷]) you can do arithmetic using stored values without first recalling them to the stack.

- Arithmetic with the [STO] function changes only the contents of the variable or register; the stack is not affected.

  For example, you could triple the value in the variable *ABC* by pressing 3 [STO] [×] ABC.

- Arithmetic with the [RCL] function calculates the result in the X-register. The contents of the variable or register and the other stack registers are not affected.

  For example, you could subtract the number in R_{12} from the number in the X-register by pressing [RCL] [−] 12.

Instruction	Result	Location of Result
[STO][+] *destination*	*destination* + *x*	*destination*
[STO][−] *destination*	*destination* − *x*	*destination*
[STO][×] *destination*	*destination* × *x*	*destination*
[STO][÷] *destination*	*destination* ÷ *x*	*destination*
[RCL][+] *source*	*x* + *source*	X-register
[RCL][−] *source*	*x* − *source*	X-register
[RCL][×] *source*	*x* × *source*	X-register
[RCL][÷] *source*	*x* ÷ *source*	X-register

Note that the *destination* and *source* may be any stack register, storage register, or variable. *x* denotes the contents of the X-register.

Recall Arithmetic and LAST X. [RCL] arithmetic saves the *x*-value in the LAST X register just as one-number functions do. Note how a normal recall instruction followed by arithmetic compares to recall arithmetic:

■ 100 [RCL] 03 [÷] recalls the contents of R_{03} and then divides that value into 100. The divisor, R_{03}, is saved in the LAST X register. Since the stack lifts when you execute [RCL], the value in the T-register is lost.

■ 100 [RCL][÷] 03 calculates the same result. However, the contents of LAST X are different. The numerator, 100, is saved in LAST X because it was the last x-value used in a calculation. The source, R_{03}, is never recalled to the stack. Since the stack does not lift, the value in the T-register is not lost.

Managing Variables

Clearing Variables

To clear a variable from memory:

1. Press ■ [CLEAR] CLV.
2. Select the variable from the catalog, or type the variable name using the ALPHA menu.

Using the Variable Catalogs

When you create a variable, the HP-42S adds that variable's name to the appropriate catalog. You can think of each catalog as a file holding variables of the same data type. To display a catalog, press ■ [CATALOG] and then:

- REAL for variables containing real numbers or Alpha strings.
- CPX for variables containing complex numbers.
- MAT for variables containing matrices.

To recall a variable from a catalog, select the catalog, and then press the corresponding menu key.

Printing Variables

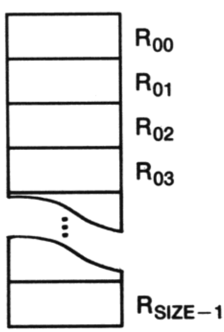
To print the contents of a single variable:

1. Press ■ [PRINT] PRV.
2. Select the variable from the catalog, or type the variable name using the ALPHA menu.

To print a complete list of variable names: Press ■ [PRINT] [▼] PRUSR (*print user*). The PRUSR function prints all variable names and global program labels. The variable names are printed first, so if you're not interested in the program labels, press [R/S] to stop the listing.

Managing Storage Registers

The storage registers are maintained in the HP-42S as a matrix named *REGS*. Each element in the matrix is a single storage register that, as you've already seen, can be stored to or recalled from with [STO] and [RCL]. Because *REGS* is a variable, you can manipulate the entire set of storage registers as a single matrix. (Refer to chapter 14 for more information on matrix operations.)



Changing the Number of Storage Registers (SIZE)

The SIZE function changes the number of storage registers available. The default size is 25 registers (R_{00}–R_{24}). The maximum number of storage registers is limited by the amount of available memory. However, the [STO] and [RCL] functions can only *directly* access registers R_{00} through R_{99}. To store and recall data in registers numbered above 99, you must use *indirect addressing* (page 74).

To change the SIZE:

1. Press ■ [MODES] [▼] SIZE.
2. Key in the number of registers. Use one, two, or three digits followed by [ENTER] or key in all four digits.

For example, to set the SIZE to 10 registers, press: ■ [MODES] [▼] SIZE 10 [ENTER].

You can also change the number of storage registers by redimensioning the *REGS* matrix. Refer to "Redimensioning a Matrix" in chapter 14.

Clearing Storage Registers

To clear all of the storage registers to zero, press ■ [CLEAR] [▼] CLRG.

To clear a single storage register to zero, store zero in it. For example, to clear R_{10}, press 0 [STO] 10.

Printing Storage Registers

To print all of the storage registers, press ■ [PRINT] PRV REGS. You can stop the listing at any time by pressing [R/S]. Note that the registers are printed as a matrix—element 1:1 corresponds to R_{00}.

For more information, refer to chapter 7, "Printing."

Storing and Recalling Alpha Data

When the calculator is in Alpha mode, the [STO] and [RCL] keys are redefined as [ASTO] (*Alpha store*) and [ARCL] (*Alpha recall*). These Alpha functions are used to copy data to and from the Alpha register in the same way [STO] and [RCL] are used to move data to and from the X-register.

There are several other functions for working with Alpha data. Refer to "Working With Alpha Data" in chapter 9.

Storing Alpha Data (ASTO)

The ASTO function copies the six left-most characters in the Alpha register into a variable or register. Variables containing Alpha strings are located in the real-variable catalog (■ [CATALOG] REAL).

Example: Storing Alpha Data. Type a string of characters into the Alpha register and store the string (the first six characters) into R_{03}.

Turn Alpha mode on. (If you worked the last example in chapter 1, the string `The HP-42S.` may still be in the Alpha register. It disappears as soon as you start typing a new string.)

■ [ALPHA]

```
The HP-42S.
ABCDE FGHI JKLM NOPQ RSTUV WXYZ
```

Type in the string `RESULT=`. (The keystrokes are RSTUV R ABCDE E RSTUV S RSTUV U JKLM L RSTUV T [▼] < = > = .)

RESULT=

```
RESULT=_
( [ { ←↑↓ < = > MATH PUNC MISC
```

Now, store the string into R_{03}. (Remember, to execute the ASTO function, press [STO] when Alpha mode is on.)

[ASTO] 03

```
RESULT=
( [ { ←↑↓ < = > MATH PUNC MISC
```

Exit Alpha mode and recall R_{03} into the X-register.

[EXIT] [RCL] 03

```
Y: 0.0000
X: "RESULT"
```

This is what an Alpha string looks like when it is in the stack. The = character is not included because strings stored in variables and registers are limited to six characters.

Recalling Alpha Data (ARCL)

The ARCL function copies data in a variable or register into the Alpha register. If the Alpha register already contains a string, the recalled data is appended to it.

If you recall a number into the Alpha register, the ARCL function converts it into Alpha characters using the current display format.

Example: Recalling Data to the Alpha Register. Calculate 5^3 and append the result to the Alpha register (which should contain RESULT= from the previous example). Remember, to execute the ARCL function, press [RCL] when Alpha mode is on.

5 [ENTER] 3 ■ [y^x] ■ [ALPHA]

```
RESULT=
ABCDE FGHI JKLM NOPQ RSTUV WXYZ
```

[ARCL] [·] ST X

```
RESULT=125.0000_
ABCDE FGHI JKLM NOPQ RSTUV WXYZ
```

Display the contents of the Alpha register using the AVIEW function.

■ [PGM.FCN] AVIEW

```
RESULT=125.0000
X: 125.0000
```

The viewed information can be cleared from the display like any other message.

[◆]

```
Y: "RESULT"
X: 125.0000
```

4

Executing Functions

The HP-42S has over 350 built-in functions—far too many to fit them all on the keyboard. Because of this, there are several ways to execute functions. You've already learned how to execute functions that appear on the keyboard and in menus. In this chapter you will learn three additional ways to execute functions:

- *Using the function catalog.* Press ■ [CATALOG] FCN to display a menu containing all of the calculator's functions. The functions are arranged in alphabetical order with special characters at the end.
- *Using the CUSTOM menu.* You can create a menu containing the functions, programs, and variables you use most often.
- *Using the* [XEQ] *(execute) key.* You can execute any calculator function by pressing [XEQ] and then typing the function name using the ALPHA menu.

You will also learn how to:

- Specify a parameter when a function prompts for additional information.
- *Preview* an instruction by holding down a key.

Using the Function Catalog

To execute a function using the function catalog:

1. Press ■ [CATALOG] FCN . (If you are planning on executing more than one function, you can prevent automatic exiting by selecting the CATALOG menu twice: ■ [CATALOG] ■ [CATALOG] FCN .)

2. Find the function you want to execute:

   - Use the [▲] and [▼] keys to move up and down through the menu. If you hold either of these keys down, they repeat so you can scroll quickly through the menu.
   - To return to the top of the catalog, press [EXIT] FCN.

3. To execute a function, press the corresponding menu key.

Example: Using the Function Catalog. Use the ASINH (*hyperbolic arc sine*) function to determine the hyperbolic arc sine of 15.

Keys	Display
15	Y: 0.0000 X: 15_
■ [CATALOG] FCN	X: 15.0000 ABS ACOS ACOSH ADV AGRA AIP

Use the [▼] key to search the catalog until you find ASINH.

Keys	Display
[▼] [▼]	X: 15.0000 ARCL AROT ASHF ASIN ASINH ASGN
ASINH	Y: 0.0000 X: 3.4023

The hyperbolic arc sine of 15 is 3.4023 (to four decimal places).

Using the CUSTOM Menu

The CUSTOM menu contains 18 blank menu labels. Each label can be redefined by assigning the name of a function, program, or variable. Therefore, you can tailor your own menu to include functions, programs, and variables you use most often.

Making CUSTOM Menu Key Assignments

To make a key assignment:

1. Press ■ [ASSIGN].

2. Use a catalog or the ALPHA menu to specify the function, program, or variable you want to assign:
   - *Using a catalog:*
     a. Press FCN, PGM, REAL, CPX, or MAT.
     b. Press the menu key corresponding to the item you want to assign.
   - *Using the ALPHA menu:*
     a. Press [ENTER] or ■[ALPHA] to select the ALPHA menu.
     b. Type the name of the function, program, or variable.
     c. Press [ENTER] or ■[ALPHA] to complete the name.
3. Press the menu key for the label to be assigned. There are 18 menu labels in the CUSTOM menu (numbered 01 through 18). Press [▼] to display the second row (labels 07 through 12). Press [▼] again to display the third row (labels 13 through 18). If you press a key for a label that already has an assignment, the new assignment replaces the previous one.

Example: Using the CUSTOM Menu. Assign the ACOSH (*hyperbolic arc cosine*) function to the first key in the CUSTOM menu and calculate the arc hyperbolic cosine of 27.

■[ASSIGN] FCN

```
ASSIGN "_
ABS ACOS ACOSH ADV AGRA AIP
```

The ACOSH function is in the first row of the function catalog.

ACOSH

```
ASSIGN "ACOSH" TO _
```

Now press the first key in the CUSTOM menu ([Σ+]).

[Σ+]

```
x: 3.4023
ACOSH
```

The assignment is ready to use.

27 ACOSH

```
x: 3.9886
ACOSH
```

Thus, the arc hyperbolic cosine of 27 is 3.9886 (to four decimal places).

Unlike other function menus, CUSTOM does not automatically exit after each use. Press [EXIT].

Clearing CUSTOM Menu Key Assignments

To clear a single key assignment:

1. Press ■[ASSIGN].
2. Press [ENTER] [ENTER] or ■[ALPHA] ■[ALPHA]. This terminates the prompt for a name.
3. Press the CUSTOM menu key for the label whose assignment you want to clear.

To clear all key assignments:

1. Press ■[CLEAR] [▼] to select the second row of the CLEAR menu.
2. Press CLKY.

Using the [XEQ] Key

To execute a function with [XEQ]:

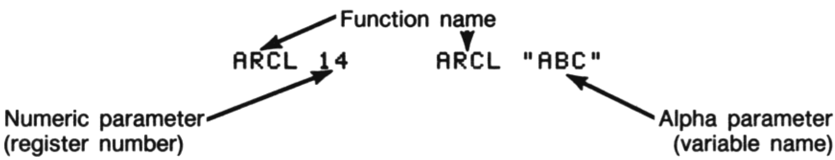
1. Press [XEQ].
2. Press [ENTER] or ■[ALPHA] to select the ALPHA menu.
3. Type the name of the function.
4. Press [ENTER] or ■[ALPHA] to complete the entry.

For example, you can execute the BEEP function by pressing [XEQ] [ENTER] BEEP [ENTER].*

* If you are not sure how to type BEEP, refer to the instructions for using the ALPHA menu on page 37.

Specifying Parameters

Many functions require a parameter to specify exactly what the function is to do. For example, the ARCL function interprets a numeric parameter as a register number and an Alpha parameter as a variable name. Refer to the table below.



Functions That Require One Parameter

Functions	Numeric Parameter	Alpha Parameter
ARCL, ASTO, DSE, INPUT, ISG, RCL, STO, VIEW, X<>	Register number.*	Variable name.
ΣREG	Register number.*	-
CLV, DIM, EDITN, INDEX, INTEG, MVAR, PRV, SOLVE	-	Variable name.
CF, FC?, FC?C, FS?, FS?C, SF	Flag number.	-
ENG, FIX, SCI	Number of digits.	-
GTO, LBL†	Numeric program label.	Alpha program label.
XEQ	Numeric program label.	Function name or Alpha program label.
CLP,† PGMINT, PGMSLV, PRP†	-	Alpha program label.

* Functions that accept register numbers also accept stack registers as parameters. Refer to "Specifying Stack Registers as Parameters" below.

† Indirect addressing cannot be used with this function.

Functions That Require One Parameter (Continued)

Functions	Numeric Parameter	Alpha Parameter
DEL,† LIST†	Number of program lines.	-
SIZE†	Number of storage registers.	-
TONE	Tone number.	-

* Functions that accept register numbers also accept stack registers as parameters. Refer to "Specifying Stack Registers as Parameters" below.

† Indirect addressing cannot be used with this function.

Functions That Require Two Parameters

Functions	First Parameter	Second Parameter
ASSIGN	Function name, Alpha program label, or variable name.†	Key number (01 through 18).†
KEYG, KEYX	Key number (1 through 9).†	Program label (local or global).

† Cannot be specified using indirect addressing.

Numeric Parameters

Functions that accept numeric parameters display a cursor for each digit expected. For example, the FIX function prompts you with `FIX __`, which takes two digits.

To complete an instruction with a numeric parameter:

- Key a digit for each position marked by a cursor. Include leading zeros if there are any.
- *Or,* key in fewer digits and complete the function by pressing [ENTER].

For example, you can set the SIZE to 9 storage registers by pressing ■ [MODES] [▼] SIZE followed by 9 [ENTER] *or* 0009.

Alpha Parameters

If the function accepts Alpha parameters, you can select the ALPHA menu by pressing [ENTER] or ■[ALPHA]. After typing the parameter, press [ENTER] or ■[ALPHA] to complete the instruction. Digits typed while the ALPHA menu is displayed are treated as Alpha characters.

Many functions that take Alpha parameters automatically display an appropriate catalog menu. If the parameter you need already exists, select it by pressing the corresponding menu key.

For example, when you execute [STO] the calculator displays a catalog of all variables currently stored in the calculator. If there are more than six entries in the catalog, the ▼▲ annunciator indicates that you can use [▼] and [▲] to display the additional rows of the catalog menu.

Specifying Stack Registers as Parameters

Any function that uses a numbered storage register can also access any of the stack registers (X, Y, Z, T, and LAST X).

To specify a stack register as a parameter:

1. Execute the function. (For example, press [STO].)
2. Press [·].
3. Specify which register you want to address:
   - ST L for the LAST X register.
   - ST X for the X-register.
   - ST Y for the Y-register.
   - ST Z for the Z-register.
   - ST T for the T-register.

Refer to page 59 for examples using stack parameters.

Indirect Addressing—Parameters Stored Elsewhere

Parameters for many functions can be specified using *indirect addressing*. That is, rather than entering the parameter itself as part of the instruction, you supply the variable, storage register, or stack register that contains the actual parameter.

Indirect addressing is particularly useful in programs when the parameter for a function is calculated.

To specify a parameter using indirect addressing:

1. Execute the function.
2. Press [·]. If the calculator displays IND __ after the function name, skip to step 4.
3. Press IND.
4. Specify where the actual parameter is located:
   - *In a variable.* Press a menu key to select the variable (the real-variable catalog is automatically displayed if there are any real variables) or type the name of the variable using the ALPHA menu.
   - *In a storage register.* Key in the register number (two digits or a single digit followed by [ENTER]).
   - *In a stack register.* Press [·] followed by ST L, ST X, ST Y, ST Z, or ST T.

Note Alpha parameters specified indirectly are limited to six characters because Alpha strings stored in variables and registers are limited to six characters.

Example: Indirect Addressing Using a Variable. Store 3 into *ABC*. Then store $\sqrt{7}$ in R_{03} using indirect addressing.

3 [STO] [ENTER] ABC [ENTER]	Y: 3.9886 X: 3.0000
7 [√x]	Y: 3.0000 X: 2.6458

[STO] [.] IND ABC

```
Y: 3.0000
X: 2.6458
```

To see that the instruction was successful, recall the contents of R_{03}.

[RCL] 03

```
Y: 2.6458
X: 2.6458
```

Exercises: Specifying Parameters

Task: Set the display format to two decimal places.

Keystrokes: ■ [DISP] FIX 02

Task: Set the display format to engineering notation using the number of digits specified in the X-register.

Keystrokes: ■ [DISP] ENG [.] [.] ST X

Task: Store a copy of the X-register into the variable or storage register specified in the Y-register.

Keystrokes: [STO] [.] IND [.] ST Y

Task: Copy the first six characters of the Alpha register into the X-register. (In Alpha mode, the [STO] key executes the ASTO function.)

Keystrokes: ■ [ALPHA] [ASTO] [.] ST X

Task: Append a copy of the data in the T-register to the contents of the Alpha register. (In Alpha mode, the [RCL] key executes the ARCL function.)

Keystrokes: ■ [ALPHA] [ARCL] [.] ST T

Task: Test the flag specified by the number in the variable *F* (assuming *F* already exists).

Keystrokes: ■ [FLAGS] FS? [.] F

Function Preview and NULL

When you hold down a key that executes a function, the name of the function is displayed. This is a *preview* of the function.

If you hold the key down for about a second, the word NULL replaces the function name in the display and the function is not executed. If you release the key before NULL is displayed, the instruction is executed.

For example, press and hold the [TAN] key.

[TAN] *(hold down)*	TAN x: 2.6458
	NULL x: 2.6458

The NULL message remains in the display until you release the key and the TAN function is not executed.

(release)	Y: 2.6458 x: 2.6458

You can preview instructions that include parameters by holding down the last key in the sequence that executes the instruction.

15 [STO] 02 (*hold down the* [2] *key*)	STO 02 x: 15.0000
	NULL x: 15.0000
(release)	Y: 2.6458 x: 15.0000

Since the instruction was aborted, the data in R_{02} was not overwritten.

5

Numeric Functions

Most of the functions built into the HP-42S are for numeric calculations. This chapter describes numeric functions for:

- General mathematics.
- Percent and percent change.
- Trigonometric calculations and conversions.
- Altering parts of numbers.
- Probability.
- Hyperbolics.

Many of the functions presented in this chapter do not appear on the HP-42S keyboard. The preceding chapter, "Executing Functions," describes how to execute functions that are not on the keyboard or in a menu.

Remember, there are two types of numeric functions:

- *One-number* functions, which replace the number in the X-register with a result (page 29).
- *Two-number* functions, which replace the numbers in the X- and Y-registers with a result and drop the stack (page 30).

General Mathematical Functions

The following table summarizes the general mathematical functions on the HP-42S keyboard. The Alpha name for each function is displayed when you hold down the key or when the function is entered into a program.

One-Number Functions

To Calculate	Press	Alpha Name
Negative of x.	[+/-]	+/−
Reciprocal of x.	[1/x]	1/X
Square root of x.	[√x]	SQRT
Square of x.	■ [x^2]	X↑2
Common logarithm of x.	[LOG]	LOG
Common exponential of x.	■ [10^x]	10↑X
Natural logarithm of x.	[LN]	LN
Natural exponential of x.	■ [e^x]	E↑X

Two-Number Functions

To Calculate	Press	Alpha Name
Sum of x and y $(x + y)$.	[+]	+
Difference of x and y $(y - x)$.	[−]	−
Product of x and y $(x \times y)$.	[×]	×
Quotient of x and y $(y \div x)$.	[÷]	÷
y to the x (y^x).	[y^x]	Y↑X

Example: Calculating a Cube Root. Calculate $\sqrt[3]{14}$. Since this can be expressed as an exponent ($14^{1/3}$), use the ■ [y^x] function.

14 [ENTER] 3

```
Y: 14.0000
X: 3_
```

[1/x]

```
Y: 14.0000
X: 0.3333
```

■ [y^x]

```
Y: 0.0000
X: 2.4101
```

The cube root of 14 is 2.4101 (to four decimal places).

Percentages

The percentage functions are special two-number functions because, unlike other two-number functions, the stack does not drop when the result is returned to the X-register.

Simple Percent

The percent function (■ [%]) calculates x% of y. For example, to calculate 12% of 300:

300 [ENTER] 12 ■ [%]

```
Y: 300.0000
X: 36.0000
```

Since the original value is preserved in the Y-register, you can easily calculate another percentage of the same number. Clear the X-register and calculate 25% of 300.

[◆] 25 ■ [%]

```
Y: 300.0000
X: 75.0000
```

The preservation of the y-value is also useful if you want to add it to the calculated percentage.

[+]

```
Y: 2.4101
X: 375.0000
```

This result is 300 plus 25% of 300 (or 125% of 300).

Percent Change

The %CH (*percent change*) function calculates the percentage of change from y to x.

Example: Calculating a Percent Change. The cost of blouses at Sonja's Boutique recently rose from $24.99 to $26.99. What was the percentage increase?

24.99 [ENTER] 26.99

```
Y: 24.9900
X: 26.99_
```

■ [CATALOG] FCN [▲] %CH

```
Y: 24.9900
X: 8.0032
```

The price increased slightly more than 8%.

Trigonometry

Setting Trigonometric Modes

The first row of the MODES menu (■ [MODES]) shows two sets of modes:

- The *angular mode* tells the HP-42S what unit of measure to assume for numbers used with trigonometric functions.

  360 degrees = 2π radians = 400 grads

- The *coordinate mode* indicates how complex numbers are displayed—rectangular or polar notation. Refer to chapter 6 for a complete description of complex numbers.

To change a mode, press the corresponding menu key.

■ [MODES] →

DEG	*Degrees* (no annunciator).*
RAD	*Radians* (**RAD** annunciator).
GRAD	*Grads* (**GRAD** annunciator).
RECT	*Rectangular.**
POLAR	*Polar.*

Trigonometric Functions

To calculate the sine, cosine, or tangent of an angle, use the trigonometric functions on the keyboard. For example, to calculate the sine of 30°, press 30 [SIN].

* Default setting.

To calculate an angle, use the inverse trigonometric functions on the keyboard. For example, to calculate the angle that produces a sine of 0.866, press .866 ■ [ASIN] (*arc sine*).

The trigonometric functions (including the inverse functions) observe the current angular mode for all calculations.

Example: Using the COS Function. Show that the cosine of $(5/7)\pi$ radians and the cosine of 128.57° are the same (to four decimal places). Start by setting Radians mode (**RAD** turns on).

■ [MODES] **RAD**

```
Y: 24.9900
X: 8.0032
```

Calculate $(5/7)\pi$.

5 [ENTER] 7 [÷] ■ [π] [×]

```
Y: 8.0032
X: 2.2440
```

Calculate the cosine of $(5/7)\pi$.

[COS]

```
Y: 8.0032
X: -0.6235
```

Now, switch to Degrees mode (**RAD** turns off).

■ [MODES] **DEG**

```
Y: 8.0032
X: -0.6235
```

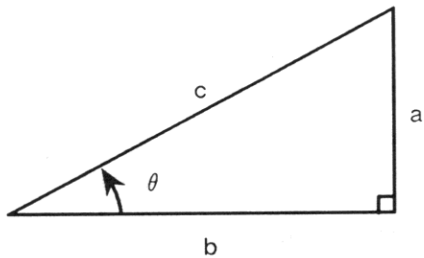
Calculate the cosine of 128.57°.

128.57 [COS]

```
Y: -0.6235
X: -0.6235
```

When you're done, both results are in the display for you to compare.

Example: Calculating an Angle. The angle θ in the following triangle can be determined by using the *arc* (inverse) trigonometric functions.



$$\theta = \text{arc sine } (^a/_c) = \text{arc cosine } (^b/_c) = \text{arc tangent } (^a/_b)$$

Suppose $a = 4$ and $c = 8$. What is θ?

4 [ENTER] 8 [÷]

```
Y: -0.6235
X: 0.5000
```

■ [ASIN]

```
Y: -0.6235
X: 30.0000
```

The angle θ is 30°.

The Conversion Functions

The first row of the CONVERT menu (■ [CONVERT]) contains six functions for converting trigonometric units or coordinates.

■ [CONVERT] →

→DEG	*Radians to degrees.*
→RAD	*Degrees to radians.*
→HR	*Hours-minutes-seconds to hours.*
→HMS	*Hours to hours-minutes-seconds.*
→REC	*Polar to rectangular.*
→POL	*Rectangular to polar.*

Converting Between Degrees and Radians

The →DEG (*to degrees*) function converts a real number in the X-register from radians into decimal degrees. Conversely, the →RAD (*to radians*) function converts a real number in the X-register from decimal degrees into radians. (The current angular mode is ignored by these two functions.)

For example, convert 0.5 radians to degrees.

.5 ■ [CONVERT] →DEG

```
Y: 30.0000
X: 28.6479
```

Convert 30° to radians.

30 ■ [CONVERT] →RAD

```
Y: 28.6479
X: 0.5236
```

Using the Hours-Minutes-Seconds Format

The HP-42S has four functions for working with numbers expressed in hours-minutes-seconds format. You may use this format for time values (*H.MMSSss*) or angles expressed in degrees (*D.MMSSss*). For example, the following numbers could represent the time 15:25:18.98 or the angle 15°25′18.98″:

15.251898 — 15: Hours or degrees; 25: Minutes; 1898: Seconds (18.98)

15.4219 — Hours or degrees in decimal format (rounded to four decimal places)

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

For example, convert 1.25 hours to hours-minutes-seconds format.

1.25 ■ [CONVERT] →HMS

```
Y: 0.5236
X: 1.1500
```

Executing →HR would change 1.1500 (that is, 1:15:00 or 1°15′00″) back to 1.2500.

Arithmetic With Minutes and Seconds. To add and subtract time (or angle) values in hours-minutes-seconds form, use the HMS+ (*hours-minutes-seconds, add*) and HMS− (*hours-minutes-seconds, subtract*) functions.

For example, if a meeting started at 9:47 a.m. and adjourned at 1:02 p.m., how long was the meeting? Enter the two times in hours-minutes-seconds format. (Enter 1:02 p.m. as 13:02.)

13.02 [ENTER] 9.47

```
Y: 13.0200
X: 9.47_
```

Execute HMS− using the function catalog.

■ [CATALOG] FCN

Use [▼] and [▲] to find the HMS− function. (Remember, these keys repeat if you hold them down.) When you find the function, execute it by pressing the corresponding menu key.

HMS-

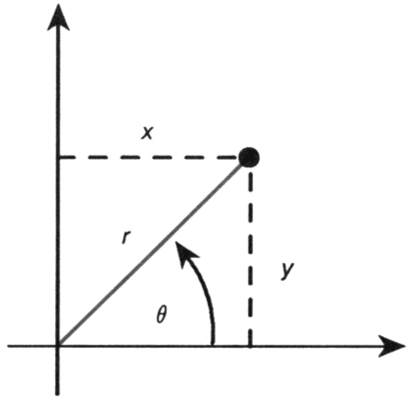
```
Y: 1.1500
X: 3.1500
```

The meeting lasted 3 hours and 15 minutes.

To multiply or divide using an hours-minutes-seconds value, first convert the number to decimal hours (■ [CONVERT] →HR), then perform the arithmetic. If you need the result expressed in hours-minutes-seconds format, convert it back (■ [CONVERT] →HMS).

Coordinate Conversions (Polar, Rectangular)

The coordinate conversion functions are →REC (*to rectangular*) and →POL (*to polar*). The rectangular coordinates (x,y) and the polar coordinates (r,θ) are measured as shown in the illustration below. The angle θ is measured in the units set by the current angular mode. (The current coordinate mode is ignored by these two functions.)



Before converting a set of coordinates, be sure the angular mode is set to the proper units for θ (page 80).

To convert rectangular to polar coordinates:

1. Key in the y-coordinate and press [ENTER].
2. Key in the x-coordinate.
3. Press ■[CONVERT] →POL. The polar coordinates (r and θ) replace x and y in the X- and Y-registers.

To convert polar to rectangular coordinates:

1. Key in θ and press [ENTER].
2. Key in the radius, r.
3. Press ■[CONVERT] →REC. The rectangular coordinates (x and y) replace r and θ in the X- and Y-registers.

Altering Parts of Numbers

The second row of the CONVERT menu contains the following functions:

■ [CONVERT] [▼]

IP	*Integer part.*
FP	*Fractional part.*
RND	*Round.*
ABS	*Absolute value.*
SIGN	*Sign.*
MOD	*Modulo.*

Integer Part (IP). The IP function removes the fractional part of a real number. For example, the integer part of 14.2300 is 14.0000.

Fractional Part (FP). The FP function removes the integer part of a real number. For example, the fractional part of 14.2300 is 0.2300.

Rounding Numbers (RND). The RND function rounds a real number to the number of digits specified by the current display format. For example, to round a dollar value to the nearest penny, set the display format to FIX 2 and then execute RND (■ [DISP] FIX 02 ■ [CONVERT] [▼] RND).

Absolute Value (ABS). The ABS function replaces the number in the X-register with its absolute value. If the X-register contains a complex number, ABS returns *r* (the radius).

The Sign of a Number (SIGN). The SIGN function tests a real number in the X-register and returns:

- 1 if x is a number greater than or equal to zero.
- -1 if x is a number less than zero.
- 0 if x is not a number.

If the X-register contains a complex number, SIGN returns the two-dimensional *unit vector* (which is also a complex number).

Modulo (MOD). The MOD function calculates the remainder of $y \div x$ (where x and y are real numbers).

Probability

The PROB (*probability*) menu contains the following functions:

[PROB]

COMB	*Combinations.*
PERM	*Permutations.*
N!	*Factorial.*
GAM	*Gamma.*
RAN	*Random number* ($0 \leq x < 1$).
SEED	*Seed.*

The Probability Functions

Combinations. The COMB (*combinations*) function calculates the number of possible *sets* of y different items taken in quantities of x items at a time. No item occurs more than once in a set, and different orders of the same x items are *not* counted separately. The formula is

$$C_{y,x} = \frac{y!}{x!(y - x)!}.$$

Permutations. The PERM (*permutations*) function calculates the number of possible different *arrangements* of y different items taken in quantities of x items at a time. No item occurs more than once in an arrangement, and different orders of the same x items *are* counted separately. The formula is

$$P_{y,x} = \frac{y!}{(y - x)!}.$$

Factorials. The N! (*factorial*) function calculates the factorial of the real number (integers only) in the X-register. For example, calculate 5!.

5 ■ [PROB] N!

```
Y: 3.1500
X: 120.0000
```

Gamma. The GAMMA function calculates $\Gamma(x)$. Key in x and then press ■ [PROB] GAM.

Generating a Random Number

To generate a random number: Press ■ [PROB] RAN. The RAN function returns a number in the range $0 \leqslant x < 1$.*

The calculator uses a *seed* to generate random numbers. Each random number generated becomes the seed for the next random number. Therefore, a sequence of random numbers can be repeated by starting with the same seed.

To store a new seed:

1. Key in any real number.
2. Press ■ [PROB] SEED.

Whenever Continuous Memory is reset, the seed is reset to zero. When the seed is equal to zero, the calculator generates a seed internally.

* The random number generator in the HP-42S actually returns a number that is part of a uniformly distributed pseudorandom number sequence. This sequence passes the spectral test (D. Knuth, *Seminumerical Algorithms*, vol. 2, London: Addison Wesley, 1981).

Hyperbolic Functions

To use a hyperbolic function, key in x, then execute the function.

To Calculate:	Execute:
Hyperbolic sine of x.	SINH
Hyperbolic cosine of x.	COSH
Hyperbolic tangent of x.	TANH
Hyperbolic arc sine of x.	ASINH
Hyperbolic arc cosine of x.	ACOSH
Hyperbolic arc tangent of x.	ATANH

6

Complex Numbers

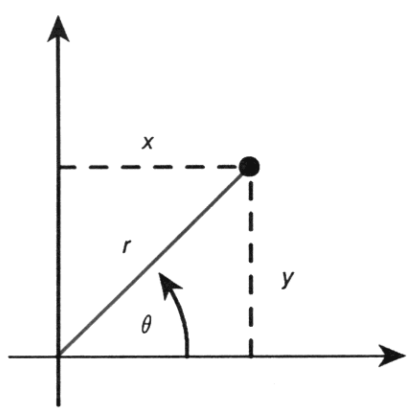
As mentioned in chapter 3, complex numbers are one of the four types of data used by the HP-42S. In this chapter you will learn:

- How to enter complex numbers.
- How complex numbers are stored and displayed.
- How to do arithmetic with complex numbers.
- How to convert the storage registers to hold complex numbers.

Entering Complex Numbers

There are two common notations for writing a complex number z:

- *Rectangular* form: $z = x + iy$.
- *Polar* form: $z = r \measuredangle \theta$.



The following relationships exist and define how the two forms are related.

$$x = r \cos \theta$$

$$y = r \sin \theta$$

$$r = \sqrt{x^2 + y^2}$$

$$i = \sqrt{-1}$$

There are two parts to a complex number: x and y, or r and θ. Each part may be any real number. The angle θ is expressed using the current angular mode (Degrees, Radians, or Grads).

To key in a complex number:

1. If necessary, set the correct coordinate and angular modes (using the MODES menu).
2. Key in the left-hand part (x or r); press [ENTER].
3. Key in the the right-hand part (y or θ).
4. Press ■[COMPLEX] to convert the two real numbers in the X- and Y-registers to a complex number in the X-register. Each part is displayed using the current display format.

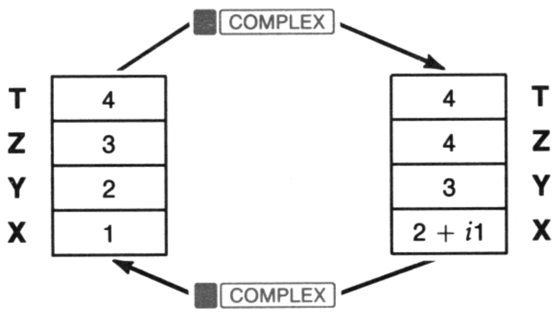
For example, to key in the complex number $2 + i1$, press 2 [ENTER] 1 ■[COMPLEX].

The coordinate mode (Rectangular or Polar) determines how the calculator interprets and displays complex numbers (as $x + iy$ or $r \measuredangle \theta$).

How ■[COMPLEX] Works:

- If the X- and Y-registers contain real numbers, executing ■[COMPLEX] combines them to form a complex number.

- If the X-register contains a complex number, executing ■[COMPLEX] separates the number into two real numbers. The left-hand part goes into the Y-register and the right-hand part stays in the X-register.

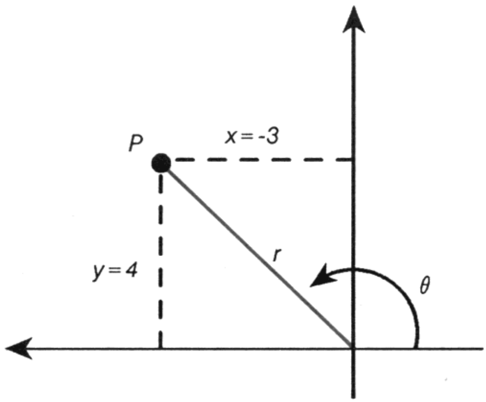


How Complex Numbers Are Displayed

Internally, the calculator always stores complex numbers in rectangular form. This has the following effects when Polar mode is used:

- The angle θ is always *normalized.* That is, the angle portion of a complex number is never larger than $\pm 180°$ ($\pm\pi$ radians).
- If a complex number is keyed in with a negative radius, the radius is made positive. The angle θ is increased by 180° (π radians) and then normalized.
- If a complex number is keyed in with a radius of zero, the angle portion of the number is also reduced to zero.

If either part of a complex number is too large or too small to display using the current display format, both parts are displayed using engineering notation (ENG 2). To view both parts of a complex number using full precision, press and hold ■[SHOW].



The following four complex numbers are equivalent representations of the point P shown above.

Coordinate Mode:	**Angular Mode:**	**Display:**
Rectangular	Any	`-3.0000 i4.0000`
Polar	Degrees	`5.0000 ∡126.8699`
Polar	Radians	`5.0000 ∡2.2143`
Polar	Grads	`5.0000 ∡140.9666`

Arithmetic With Complex Numbers

Most of the arithmetic functions in the previous chapter work with complex numbers as well as real numbers. For example, calculate the following expression:

$$(5 + i3) + (7 - i9).$$

Ensure the calculator is in Rectangular mode.

■ [MODES] **RECT**

```
Y: 0.0000
X: 0.0000
```

Enter the two numbers.

5 [ENTER] 3 ■[COMPLEX]
7 [ENTER] 9 [+/-] ■[COMPLEX]

```
Y: 5.0000 i3.0000
X: 7.0000 -i9.0000
```

And add them.

[+]

```
Y: 0.0000
X: 12.0000 -i6.0000
```

Complex Results Produced by Real-Number Functions. Some real-number functions can produce a complex number as a result. For example, calculating the square root of a negative number produces the appropriate complex number.

Multiply the result from the calculation above by $\sqrt{-25}$.*

25 [+/-] [$\sqrt{x}$]

```
Y: 12.0000 -i6.0000
X: 0.0000 i5.0000
```

[×]

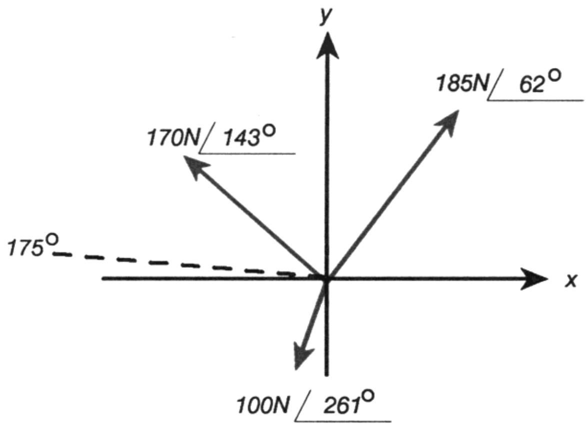
```
Y: 0.0000
X: 30.0000 i60.0000
```

Vector Operations Using Complex Numbers

A complex number can represent a vector in a two-dimensional plane. Using the vector functions in the second row of the MATRIX menu (page 220), you can perform vector operations with complex numbers.

Example: Dot Product of Complex Numbers. The figure below represents three two-dimensional force vectors. Use complex numbers and add the three vectors. Then use the DOT (*dot product*) function to find the component of the resulting vector along the 175° line.

* The calculator's ability to produce complex results with real-number functions can be disabled by pressing ■[MODES] [▼] RRES (*real results only*). To enable complex results (after they have been disabled with RRES), press ■[MODES] [▼] CRES (*complex-result enable*).



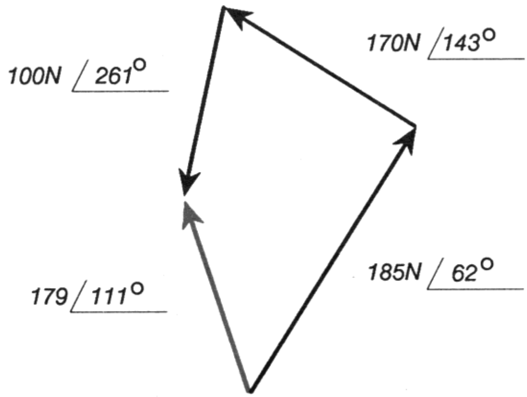
Select Degrees and Polar modes.

Keys	Display
■ [MODES] DEG ■ [MODES] POLAR	Y: 0.0000 X: 67.0820 ∡63.4349

Add the three vectors.

Keys	Display
185 [ENTER] 62 ■ [COMPLEX]	Y: 67.0820 ∡63.4349 X: 185.0000 ∡62.0000
170 [ENTER] 143 ■ [COMPLEX]	Y: 185.0000 ∡62.0000 X: 170.0000 ∡143.0000
[+]	Y: 67.0820 ∡63.4349 X: 270.1198 ∡100.4332
100 [ENTER] 261 ■ [COMPLEX]	Y: 270.1198 ∡100.4332 X: 100.0000 ∡-99.0000
[+]	Y: 67.0820 ∡63.4349 X: 178.9372 ∡111.1489

Thus, the resulting sum is a force of approximately 179 Newtons at 111°.



Now calculate the 175° component of this result.

1 [ENTER] 175 ■[COMPLEX]

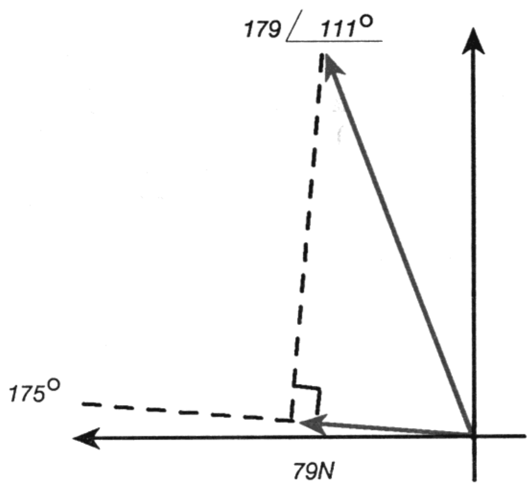
```
Y: 178.9372 ∡111.1489
X: 1.0000 ∡175.0000
```

■[MATRIX] [▼] DOT

```
X: 78.8586
 DOT  CROSS UVEC  DIM  INDEX EDITN
```

Thus, the resulting sum has a component of approximately 79 Newtons in the direction of 175°.

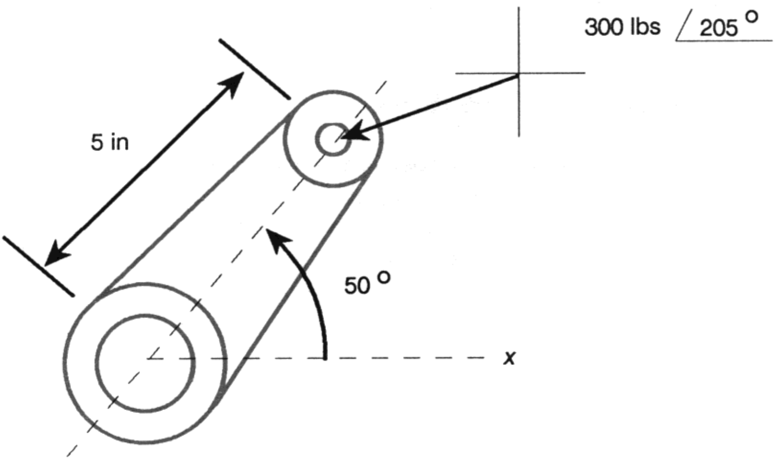
[EXIT]



Example: Computing Moments. To compute the moment of two vectors, use the CROSS (*cross product*) function. The cross product of two vectors is a third orthogonal vector. However, when two complex numbers are crossed, the HP-42S simply returns a real number that is equal to the signed magnitude of the resulting moment vector.

Find the moment generated by the force acting through the lever in the illustration below, where

$$\vec{M} = \vec{r} \times \vec{F}.$$



Select Degrees and Polar modes. (You can skip this step if you have already selected these modes.)

■ [MODES] DEC ■ [MODES] POLAR

```
Y: 67.0820 ∡63.4349
X: 78.8586
```

Key in the radius vector and the force vector.

5 [ENTER] 50 ■ [COMPLEX]

```
Y: 78.8586
X: 5.0000 ∡50.0000
```

300 [ENTER] 205 ■ [COMPLEX]

```
Y: 5.0000 ∡50.0000
X: 300.0000 ∡-155.0000
```

Calculate the cross product.

■ [MATRIX] [▼] CROSS

```
x: 633.9274
DOT CROSS UVEC DIM INDEX EDITN
```

The moment vector has a magnitude of 634 and, since the result is positive, the vector points up, perpendicular to the plane of this page.*

[EXIT]

Storing Complex Numbers

Complex-Number Variables

When you store a complex number into a variable, the variable name is added to the complex-variable catalog. To display a catalog menu containing all of the complex-number variables, press ■ [CATALOG] CPX . To recall a variable from the catalog, press the corresponding menu key. Refer to chapter 3 for details on using variables and catalogs.

Making the Storage Registers Complex

Normally, each storage register can hold only a real number or an Alpha string. However, you can change the type of the *REGS* matrix to complex so that each storage register holds a complex number.

* If the problem you're working requires a true (three dimensional) vector as a result, use a 1 × 3 matrix to represent each vector in three dimensions.

To make the storage registers complex:

1. Enter zero as a complex number: 0 [ENTER] ■ [COMPLEX].
2. Press [STO] [+] REGS to add the complex number (zero) to the *REGS* matrix.

Since the result of any arithmetic is complex if either operand is complex, this procedure makes the storage registers complex. The procedure will fail if any of the storage registers contain an Alpha string.

To make the storage registers real:

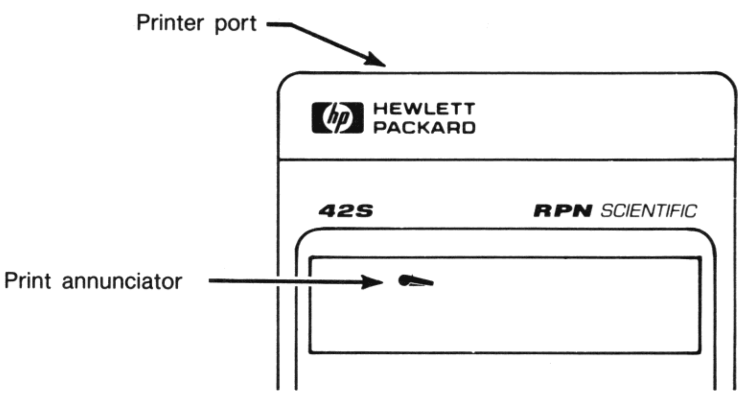
1. Press [RCL] REGS to recall a copy of the storage registers into the X-register.
2. Press ■ [COMPLEX] to separate the complex matrix into two real matrices.
3. Press [x≷y] to move the matrix of real-parts into the X-register.
4. Press [STO] REGS.

7

Printing

The HP-42S prints information using the HP 82240A Infrared Printer, which accepts the infrared signal generated by the calculator's printer port.

The print annunciator ( ) comes on whenever the calculator sends information through its printer port.



With a printer you can:

- Print intermediate and final results, including all types of data.
- Keep a running record of your keystrokes and calculations.
- List the names of programs and variables stored in the calculator.
- Print complete and partial program listings.
- Print a copy of the display.

Common Printing Operations

The first two rows of the PRINT menu contains these print functions:

■ [PRINT] →

PRΣ	*Print statistics.*
PRP	*Print program.*
PRV	*Print variable.*
PRST	*Print stack.*
PRA	*Print Alpha register.*
PRX	*Print X-register.*

▼▲

PRUSR	*Print user* (variables and programs).
LIST	*List* program lines.
ADV	*Advance* printer paper.
PRLCD	*Print LCD* (*liquid crystal display*).
DELAY	*Delay* time between lines.

Here are a few common printing tasks:

To enable printing: Press ■ [PRINT] [▲] PON (*printing on*). The PRON function sets flags 21 (printer enable) and 55 (printer existence).

The infrared printer port remains enabled until you disable it by pressing ■ [PRINT] [▲] POFF (*printing off*). The PROFF function clears flags 21 and 55.

To print the contents of the X-register: Press ■ [PRINT] PRX .

To print the contents of a variable:

1. Press ■ [PRINT] PRV .
2. Select the variable from the catalog, or type the variable name using the ALPHA menu.

For example, to print the contents of the storage registers (which are stored as a matrix named *REGS*), press ■ [PRINT] PRV REGS .

To print the contents of the Alpha register: Press ■ [PRINT] PRA .

Printing Modes

How and *when* information is sent to the printer depends on the current modes. The functions for controlling the printing modes are in the third row of the PRINT menu:

■ [PRINT] [▲] →

PON	*Printing on.*
POFF	*Printing off.*
MAN	*Manual mode.*
NORM	*Normal mode.*
TRACE	*Trace mode.*

To select a printing mode:

1. Press ■ [PRINT] [▲].
2. Press one of the following:
   - MAN (*Manual* mode). Use this mode when you want the calculator to print only when a print function is executed. The VIEW and AVIEW functions can also generate printer output in this mode. (This is the default mode.)
   - NORM (*Normal* mode). Use this mode when you want to print a record of prompts and keystrokes.
   - TRACE (*Trace* mode). Use this mode when you want to print a record of prompts, keystrokes, and results. If a program is running, each instruction is printed as it executes. This mode is primarily intended for testing and debugging programs.

Flags That Affect Printing

There are several flags that affect how and when information is printed. For example, to cause all printing to be double-wide, set flag 12 (■ [FLAGS] SF 12). To return to normal-width printing, clear flag 12 (■ [FLAGS] CF 12).

Flag(s)	Purpose	Page(s)
12	Double-wide printing.	274
13	Lowercase printing.	274
15 and 16	Printing mode.	274
21 and 55	Printer enable and printer existence.	131 and 132

Printing Speed and Delay Time

Since the HP-42S is capable of sending information faster than it can be printed by the HP 82240A Infrared Printer, the calculator uses a *delay* time to avoid losing information. To optimize printing speed, set the delay time slightly greater than the time it takes for your printer to print a single line of information.

To set the printing delay time:

1. Key the delay time into the X-register (in seconds). The longest delay time you can set is 1.9 seconds.
2. Press ■ [PRINT] [▼] DELAY.

If you're operating the printer without an AC adapter, printing speed will slow down as the batteries discharge. If you are using the longest delay time (1.9 seconds) and your printer is still too slow, replace the batteries or connect an AC adapter. Operating the printer with batteries this low (without an AC adapter) will likely result in poor infrared communication and may damage the printer.

Low Calculator Batteries

To conserve battery power, the HP-42S will not transmit data to the printer when the ▭ annunciator is on. If low battery power occurs after you've started printing, printing stops and the calculator displays `Batt Too Low To Print`. The calculator automatically returns to Manual mode.

Calculator Functions That Print

If printing is enabled (after executing PRON), the VIEW and AVIEW functions automatically generate printed output (in addition to performing their normal functions).

For more information on how these functions and flags 21 and 55 affect program execution, refer to chapter 9, "Program Input and Output."

Printing Graphics in the Display

The PRLCD (■ [PRINT] [▼] **PRLCD**) function copies the display to the printer, pixel for pixel. The primary purpose for this function is to print graphics that you create in the display using the PIXEL and AGRAPH functions (page 135).

The "PLOT" program on page 160 creates a plot in the display and then uses the PRLCD function to print it.

Printing Programs

To print an entire program:

1. Press ■ [PRINT] **PRP** (*print program*).
2. Select the program from the catalog, or type the program name (global label) using the ALPHA menu.

The PRP function prints the entire program, even if the global label you specify is not the first line of the program.

If you do not specify any label (■ [PRINT] PRP [ENTER] [ENTER]), the calculator prints the current program.

To print part of a program:

1. Position the program pointer to the line where you want to start the listing (page 111).
2. Press ■ [PRINT] [▼] LIST.
3. Key in the number of lines you want to print, *nnnn*. (If you enter fewer than four digits, complete the instruction by pressing [ENTER].)

The program listing begins with the current line and continues for *nnnn* lines or until an END instruction is encountered.

Character Sets

Some characters are not printed as you see them in the display. This is because the character set used in the HP-42S does not directly match the character set used in the HP 82240A Infrared Printer. Compare the character table in appendix E of this manual with the character set listed in the HP 82240A owner's manual.

Part 2

Programming

8

Simple Programming

Part 1 of this manual introduced you to several functions and operations that you can use *manually* (from the keyboard). In this chapter you will learn how *programs* can be used to store and execute a sequence of functions. More specifically, you will learn:

- How to key a program into memory.
- How to edit (change) a program.
- How to execute (run) a program.
- What happens when an error causes a program to stop.
- About the parts of a program.
- How to clear a program from memory.

The programming information in this manual (chapters 8, 9, and 10) should give you a good start at writing your own programs. If you're interested in more advanced programming information and techniques, refer to the *HP-42S Programming Examples and Techniques* manual (part number 00042-90020).

An Introduction to Keystroke Programming

The sequence of steps a program uses to perform a calculation are the same as the steps you would execute when solving the problem manually. By programming your calculator you can repeat operations or calculations without repeating the keystrokes every time.

For example, consider the formula for the area of a circle:

$$A = \pi r^2.$$

To calculate the area of a circle with a radius of 5, you would key in the radius, square it, and then multiply by π.

5 ■ [x^2] ■ [π] [×]

```
Y: 0.0000
X: 78.5398
```

The keystrokes ■ [x^2] ■ [π] [×] can be stored as a program and then executed any number of times for circles of different radii. Such a program might look like this:

```
01 LBL "AREA"
02 X↑2
03 PI
04 ×
05 END
```

This program assumes that the radius is in the X-register when the program runs. To calculate an area, you would key in a radius and then run the program. The result (the area of the circle) is left in the X-register when the program ends.

The label (line 01) identifies the program so you can refer to it by name. The END instruction (line 05) separates this program from the next program in memory.

Example: Keying In and Running a Program. To key a program into the calculator, press ■ [GTO] [·] [·] to move to a *new program space* (page 118), and then press ■ [PRGM] to select *Program-entry mode*.

■ [GTO] [·] [·] ■ [PRGM]

```
00▶{ 0-Byte Prgm }
01 .END.
```

Key in the program listed above.

■ [PGM.FCN] LBL AREA [ENTER]

```
00 { 8-Byte Prgm }
01▶LBL "AREA"
```

The next three lines are the *body* of the program—that is, the part that calculates the area of the circle. As you press the keys, the calculator records them and automatically numbers them as program steps.

■ [x²]

```
01 LBL "AREA"
02▸X↑2
```

■ [π]

```
02 X↑2
03▸PI
```

[×]

```
03 PI
04▸×
```

The calculator has automatically provided an END statement for you, so the program is complete. If you want to review the program before exiting Program-entry mode, use [▼] and [▲] to move up and down through the lines of the program.

[EXIT]

```
Y: 0.0000
X: 78.5398
```

Now you can use the program to calculate the area of any circle given the radius, *r*. Key in a radius of 5 and run the program.

5 [XEQ] AREA

```
Y: 78.5398
X: 78.5398
```

The result is the same as when you solved the problem manually.

Find the area of a circle with a radius of 2.5.

2.5 [XEQ] AREA

```
Y: 78.5398
X: 19.6350
```

Divide the two results.

[÷]

```
Y: 78.5398
X: 4.0000
```

The area of a circle with a radius of 5 is 4 times greater than the area of a circle with a radius of 2.5.

Program-Entry Mode

The ■ [PRGM] key toggles the calculator in and out of Program-entry mode. In Program-entry mode, functions and numbers you key in are saved as program instructions.

The Program Pointer

While working the example above, you may have noticed the ▸ character in the display. This is the *program pointer*. It points to the *current program line*. If the current program line is too long for the display, press and hold the ■ [SHOW] key to display the entire line.

Moving the Program Pointer

The following instructions are nonprogrammable, so you can execute them in or out of Program-entry mode to move the program pointer.

To move the program pointer to:	Press:
The next program line.	■ [SST] (or [▼] if no menu is displayed)
The previous program line.	■ [BST] (or [▲] if no menu is displayed)
Line number *nnnn* of the current program.	■ [GTO] [.] *nnnn*
A global label.	■ [GTO] [.] [ENTER] *label* [ENTER]
A new program space.	■ [GTO] [.] [.]

Inserting Program Lines

Instructions keyed into a program are inserted immediately *after* the current program line and the pointer advances to the new line. Therefore, to insert a program line between lines 04 and 05 of a program, you would move the pointer to line 04 and then key in the instruction.

Deleting Program Lines

To delete a program line, position the program pointer to the line you want to delete, and then press [◆]. When you delete a line, the program pointer moves to the previous line.

To delete several consecutive program lines, use the DEL (*delete*) function (page 120).

Executing Programs

In general, there are two ways to run programs:

- *Normal execution.* Program instructions continue to execute until an instruction that stops program execution is encountered (such as STOP, PROMPT, RTN, or END) or until you manually stop the program by pressing [R/S] or [EXIT].
- *Stepwise execution.* Program instructions are executed one at a time as you *step* through the program with the ■[SST] key. This method of executing a program is especially useful when you are *debugging* a program (testing for errors).

Remember, Program-entry mode must be *off* to run a program.

Normal Execution

To execute a program using the program catalog:

1. Press [XEQ] or ■[CATALOG] PGM .
2. Press the menu key corresponding to the program you want to execute.

This method is used in the example on page 110.

To assign a program to the CUSTOM menu:

1. Press ■[ASSIGN] PGM .
2. Press the menu key corresponding to the program you want to assign.

3. The CUSTOM menu has three rows; use [▼] or [▲] to display the row you want and then press the menu key where you want the program assigned.

For example, assign the "AREA" program to the CUSTOM menu.

■ [ASSIGN] PGM AREA	ASSIGN "AREA" TO _
[√x] *(the third menu key)*	x: 4.0000 — AREA

Now, each time you want to calculate the area of a circle, key in the radius and then press AREA.

5 AREA	x: 78.5398 — AREA
3.25 AREA	x: 33.1831 — AREA
[EXIT]	

Running a Program With [R/S]

To run the current program beginning with the current program line, press [R/S] *(run/stop)*. If you hold the [R/S] key down, the calculator displays the current program line (▶); that is, the line to be executed next. If you hold [R/S] down until NULL appears, the program does not begin running when you release the key.

You can position the program pointer to the top of the current program by executing the RTN function when Program-entry mode is off. Therefore, to run the current program (beginning with the first line), press ■ [PGM.FCN] RTN and then [R/S].

Stopping a Program

To stop a running program press [R/S] or [EXIT]. Execution halts after the current instruction is completed. To resume execution, press [R/S] again.

Testing and Debugging a Program

The HP-42S allows you to execute any program one step at a time with the ■[SST] key. This feature is particularly useful when you are trying to find a *bug* (error) in a program or when you just want to see how each instruction in a program works. (Note that if there is no menu displayed, the [▼] and [▲] keys can be used to execute the ■[SST] and ■[BST] functions.)

While testing or debugging a program, you may want to use Trace mode to print a running record of each program step as it is executed. To select Trace mode, press ■[PRINT] [▲] TRACE.

To execute a program one step at a time:

1. Position the program pointer to the label or line number where you want to start executing the program. If you skip this step, execution will begin with the current program line.
2. Be sure Program-entry mode is *off*. If any data is needed at the start of the program, enter it.
3. Press and hold ■[SST] to display the current program line. When you release ■[SST], the instruction is immediately executed and the program pointer advances.

   If you hold the ■[SST] key down too long, NULL appears and the program instruction is *not* executed when you release the key.

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

You can move the program pointer up (backwards) through a program with the ■[BST] key. The ■[BST] key:

- Moves the program pointer *without* executing program instructions.

- Repeats when you hold the key down.

Error Stops

If an error occurs while a program is running, program execution halts, and the appropriate error message is displayed. The error message disappears when you press a key.

The program pointer stops at the line that generated the error. To view the line, select Program-entry mode (■ [PRGM]).

A running program will ignore an error if flag 24 (*range ignore*) or flag 25 (*error ignore*) is set. Refer to appendix C for more information on these flags.

The Basic Parts of a Program

Program Lines and Program Memory

As you've already seen, when the HP-42S is in Program-entry mode, keystrokes you enter are not immediately executed, but are stored in program memory as instructions. Each instruction occupies a single program line, which is automatically numbered.

Types of Program Lines. Program lines are divided into several categories. A program line may contain:

- A program label (such as `LBL "AREA"`).
- A complete instruction (such as a simple numeric function, like [+], or an instruction that includes a parameter, like [STO] 14).
- A complete number (called a *numeric constant*).
- An Alpha string of up to 15 characters (called an *Alpha constant*).

Memory Requirements. Program steps may vary in size from 1 to 16 bytes. At the top of each program (line 00) the calculator displays the size of the current program in bytes.

If you run out of memory while trying to enter a program line, the calculator displays `Insufficient Memory`. Refer to appendix B, "Managing Calculator Memory."

Program Labels

A label is an identifier placed at the beginning of a series of program steps. Program labels may be used anywhere in a program. Generally, a program starts with a *global label*. Within a program, individual routines can be identified with *local labels*.

Global Labels. Global labels use Alpha characters and are distinguished by quotation marks around the label name.* For example, the program at the beginning of this chapter has a global label:

```
01 LBL "AREA"
```

Global labels may be one to seven characters long. The single-letter names `A` through `J` and `a` through `e` are reserved for local Alpha labels (which are displayed *without* quotation marks).

Global labels:

- Can be accessed no matter where the program pointer is located.
- Are listed in the program catalog (■ [CATALOG] PGM).
- Can be assigned to the CUSTOM menu.
- Should be unique within calculator memory to avoid confusing one program with another.

Local Labels. There are two types of local labels: *numeric* and *Alpha*.

- Numeric labels are identified by two digits, `LBL 00` through `LBL 99`. (`LBL 00` through `LBL 14` are called *short form* local labels because they use less memory.)
- Local Alpha labels use single Alpha characters, `LBL A` through `LBL J` and `LBL a` through `LBL e`.

* To key in a global label that begins with a digit character, select an ALPHA submenu and then type the digit. This forces the digit to become an Alpha character. For example, to key in LBL "1", press ■ [PGM.FCN] LBL ABCDE 1 [ENTER]. Without ABCDE, the label is interpreted as LBL 01.

Local labels are used to mark and provide access to various segments of a program. The primary purpose for local labels is to facilitate program *branching*. Refer to "Branching" in chapter 10.

Local labels can be:

- Accessed only within the current program.
- Duplicated in separate programs. That is, local labels do not need to be unique within calculator memory, but they should be unique within each program. (It is possible to use duplicate local labels within a single program if you consider the search patterns used to find local labels. Refer to page 148.)

The Body of a Program

The body of a program is where all the work is done. For example, the body of the "AREA" program is:

```
02 X↑2
03 PI
04 ×
```

This program contains two functions (X↑2 and ×) and a numeric constant (PI).

Constants

Numeric Constants. A numeric constant is simply a number in a program. When the line is executed the number is placed in the X-register, lifting the stack just as if you keyed the number in from the keyboard.

The PI function (■ [π]) operates like a numeric constant. Thus, the "AREA" program would return exactly the same result if line 03 looked like this:*

```
03 3.14159265359
```

* Although the program would run the same, keying in the 12-digit approximation for π takes 14 bytes of program memory; the PI function requires only 1 byte.

Consecutive Numeric Constants. Because numeric constants in programs are on different program lines, [ENTER] is not needed to separate them. Consider these two programs:

```
01 12                01 12
02 ENTER             02 17
03 17                03 ×
04 ×
```

Both programs produce the same result (12 × 17), however the one on the right is one line shorter and saves one byte of program memory. To key in the program lines on the right, press 12 [ENTER] [◆] 17 [×].

Program ENDs

Programs are separated from one another with END instructions. The last program in memory uses the *permanent* END, which appears in the display as `.END.`.

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. There are two ways to enter an END at the end of a program:

- Press ■[GTO] [·] [·]. This procedure automatically inserts an END after the last program in memory and positions the program pointer to the new program space at the bottom of program memory. This space contains the *null* program:

  ```
  00 { 0-Byte Prgm }
  01 .END.
  ```

- *Or,* manually execute the END function (press [XEQ] [ENTER] END [ENTER] or use the function catalog).

Because END instructions separate programs, deleting an END causes the two programs to be joined as one program. You cannot delete the permanent `.END.`.

```
LBL "PAM"
.
.
.
END
LBL "BRUCE"
.
.
.
.
END
LBL "CHRIS"
.
.
.
END
LBL "BOB"
.
.
.
END
LBL "DEX"
.
.
.END.
```

Clearing Programs

To clear an entire program from memory:

1. Press ■ [CLEAR] [CLP].
2. Specify the program you want to clear using *one* of the following:
   - Press the menu key corresponding to a global label in the program.
   - Use the ALPHA menu to type a global label ([ENTER] *label* [ENTER]).
   - *Or,* press [ENTER] [ENTER] to clear the *current program.*

To clear a portion of a program:

1. Press ■ [PRGM] to select Program-entry mode (if the calculator is not already in Program-entry mode).
2. Position the program pointer to the first line in the range of lines you want to delete.
3. Press ■ [CLEAR] [▼] DEL (*delete*).
4. Key in the number of lines you want to delete.

For example, to delete lines 14 through 22 of the current program, you would press: ■ [PRGM] (to select Program-entry mode), ■ [GTO] [·] 14 [ENTER] (to position the program pointer to line 14), ■ [CLEAR] [▼] DEL 9 [ENTER] (to delete 9 program lines).

The DEL function deletes program lines only if the calculator is in Program-entry mode.

9

Program Input and Output

An *interactive* program has two general characteristics:

- *Input*. The program prompts you to key in information or make a selection.
- *Output*. The program presents results in a meaningful format using the display or a printer.

This chapter covers functions and techniques for making programs easier to use. You'll learn about:

- Prompting for values and using variable menus.
- Displaying labeled output and messages.
- Printing during program execution.
- Working with Alpha data.
- Displaying graphics.

Using the INPUT Function

Using the INPUT function is one of the simplest ways for a program to prompt for data to be stored into a variable or register. When an INPUT instruction is executed:

- The current value of the variable or register is recalled into the X-register. If you use a new variable name, the INPUT function automatically creates the variable and assigns an initial value of zero.
- The normal label for the X-register (x:) is replaced with the name of the variable or register being input and a question mark.

■ Program execution halts, allowing you to key in or calculate a value.

When you press [R/S], the value in the X-register is automatically stored into the variable or register and program execution continues.

Pressing [EXIT] (if there is no menu displayed) cancels the INPUT function without storing any data. If you then press [R/S], the INPUT is resumed with the original value.

Example: Using INPUT. The formula for the surface area of a box is

Area = 2 ((*length* × *height*) + (*length* × *width*) + (*height* × *width*)).

The following program uses INPUT to prompt for the values of *L*, *H*, and *W* and then calculates the surface area.

Program	Comment
`01 LBL "SAREA"` `02 INPUT "L"` `03 INPUT "H"` `04 INPUT "W"`	Inputs each of the three variables.
`05 RCL× "L"`	Calculates *length* × *width*. The value of *W* is already in the X-register because it was the last value input.
`06 LASTX` `07 RCL× "H"`	Calculates *height* × *width*
`08 RCL "H"` `09 RCL× "L"`	Calculates *length* × *height*
`10 +` `11 +` `12 2` `13 ×` `14 END`	Calculates sum of the products, multiplies by 2, and leaves the result in the X-register.

Key the program into your calculator.

■ [GTO] [·] [·] ■ [PRGM]	00▶{ 0-Byte Prgm } 01 .END.
■ [PGM.FCN] ■ [PGM.FCN] LBL SAREA [ENTER]	01▶LBL "SAREA" LBL RTN INPUT VIEW AVIEW XEQ
INPUT [ENTER] L [ENTER]	02▶INPUT "L" LBL RTN INPUT VIEW AVIEW XEQ
INPUT [ENTER] H [ENTER]	03▶INPUT "H" LBL RTN INPUT VIEW AVIEW XEQ
INPUT [ENTER] W [ENTER] [EXIT]	03 INPUT "H" 04▶INPUT "W"
[RCL] [×] [ENTER] L [ENTER]	04 INPUT "W" 05▶RCL× "L"
■ [LASTx]	05 RCL× "L" 06▶LASTX
[RCL] [×] [ENTER] H [ENTER]	06 LASTX 07▶RCL× "H"
[RCL] [ENTER] H [ENTER]	07 RCL× "H" 08▶RCL "H"
[RCL] [×] [ENTER] L [ENTER]	08 RCL "H" 09▶RCL× "L"
[+]	09 RCL× "L" 10▶+
[+]	10 + 11▶+
2 [×]	12 2 13▶×
[EXIT]	

Run the program to calculate the surface area of a box that is 4 × 3 × 1.5 meters.

[XEQ] SAREA	Y: 0.0000 L?0.0000

The program is prompting for a value of *L*. Key in the length (4) and press [R/S].

4 [R/S]	Y: 4.0000 H?0.0000

Key in the height (3) and press [R/S].

3 [R/S]	Y: 3.0000 W?0.0000

Key in the width (1.5) and press [R/S].

1.5 [R/S]	Y: 0.0000 x: 45.0000

The surface area is 45 square meters.

What is the surface area of a box that is twice as long? Run the program again. This time multiply the length by 2 and leave the other values as they are.

[XEQ] SAREA	Y: 45.0000 L?4.0000
2 [×] [R/S]	Y: 8.0000 H?3.0000
[R/S]	Y: 3.0000 W?1.5000
[R/S]	Y: 3.0000 x: 81.0000

The surface area is 81 square meters.

Using a Variable Menu

Using a *variable menu* may be the most efficient way for a program to input values for several variables. The VARMENU (*variable menu*) function creates a menu containing variable names. When the program stops, the menu is displayed allowing you to store, recall, and view variables.

The VARMENU function requires a global program label as a parameter. When a program executes VARMENU, the calculator searches for the specified program label. It then builds the variable menu using the MVAR (*menu variable*) instructions immediately following the specified label. (The calculator ignores MVAR instructions except when they are being read by the VARMENU function.*)

To store a value into a menu variable:

1. Key in or calculate the value.
2. Press the corresponding menu key.

To recall the value of a menu variable:

1. Press [RCL].
2. Press the corresponding menu key.

To view a menu variable without recalling it:

1. Press ■ (shift).
2. Press *and hold* the corresponding menu key. The message disappears when you release the key.

* The Solver and Integration applications also use variable menus defined with MVAR instructions.

To continue program execution:

- Press a menu key.
- *Or,* press [R/S].

If you continue by pressing a menu key, the name of the corresponding variable is stored into the Alpha register. Your program can use this information to determine which key was pressed. If you continue by pressing [R/S], the Alpha register is not altered.

To exit from a variable menu:

- Press [EXIT].
- *Or,* select an application menu (■[SOLVER], ■[∫f(x)], ■[MATRIX], ■[STAT], or ■[BASE]).

Example: Using a Variable Menu. In the previous program, the INPUT function was used to prompt for three variables. By replacing lines 02, 03, and 04 with the following seven program lines, you can add a variable menu to the program.

02 MVAR "L" 03 MVAR "H" 04 MVAR "W"	Declares the menu variables following the global label.
05 VARMENU "SAREA" 06 STOP 07 EXITALL	Creates the variable menu and stops the program. When the program is restarted, the variable menu is exited.
08 RCL "W"	Since the variables in a variable menu can be entered in any order, there is no gaurantee that *W* will be in the X-register (as there was in the first program).

Edit the "SAREA" program. First delete lines 02, 03, and 04.

■[PRGM] ■[GTO] [·] 4 [ENTER]	03 INPUT "H" 04▸INPUT "W"
[◆] [◆] [◆]	01▸LBL "SAREA" 02 RCL× "L"

Now insert the new program lines.

Keys	Display
■ [PGM.FCN] ■ [PGM.FCN] [▲] MVAR L	02▶MVAR "L" MVAR VARM GETK MENU KEYG KEYX
MVAR H	03▶MVAR "H" MVAR VARM GETK MENU KEYG KEYX
MVAR W	04▶MVAR "W" MVAR VARM GETK MENU KEYG KEYX
VARM SAREA	05▶VARMENU "SAREA" MVAR VARM GETK MENU KEYG KEYX
[EXIT] [R/S]	05 VARMENU "SAREA" 06▶STOP
■ [CATALOG] FCN	06▶STOP ABS ACOS ACOSH ADV AGRA AIP

Use the arrow keys to find the EXITALL function in the catalog.

Keys	Display
[▼] ... [▼] EXITA	06 STOP 07▶EXITALL
[RCL] W	07 EXITALL 08▶RCL "W"
[EXIT]	

Now run the new version of the program.

Keys	Display
[XEQ] SAREA	x: 81.0000 L H W

The variable menu is displayed, ready to use. Calculate the surface area of a box that is 5.5 × 2 × 3.75 cm.

Keys	Display
5.5 L	L=5.5000 L H W

2 [W]

```
W=2.0000
 L  H  W
```

3.75 [H]

```
H=3.7500
 L  H  W
```

[R/S]

```
Y: 3.7500
X: 78.2500
```

The surface area is 78.25 cm^2.

[EXIT]

Displaying Labeled Results (VIEW)

To display the contents of a variable or register use the VIEW function. VIEW creates a message that includes the variable or register name, an equal sign, and the data stored there. (Also refer to "Printing With VIEW and AVIEW" on page 132.)

For example, add these two lines to the end of the "SAREA" program.

```
18 STO "SAREA"
19 VIEW "SAREA"
```

Line 18 stores the result into a variable named *SAREA*. Line 19 displays the contents of *SAREA*.

0 [STO] [ENTER] SAREA [ENTER]

```
Y: 78.2500
X: 0.0000
```

■ [PRGM] ■ [GTO] [·] 17 [ENTER]

```
16 2
17▸×
```

[STO] [SAREA]

```
17 ×
18▸STO "SAREA"
```

■ [PGM.FCN] [VIEW] [SAREA]

```
18 STO "SAREA"
19▸VIEW "SAREA"
```

[EXIT]

```
Y: 78.2500
X: 0.0000
```

Now, run the program again using dimensions of 2 × 3 × 4 m.

[XEQ] SAREA

```
X: 0.0000
L  H  W
```

2 L 3 W 4 H [R/S]

```
SAREA=52.0000
X: 52.0000
```

This time the answer is labeled for you. This technique is particularly useful when a program has several results.

Displaying Messages (AVIEW and PROMPT)

Messages are useful in programs to display descriptive prompts, output, and error conditions. For a program to display a message, it must:

1. Create the message in the Alpha register with an Alpha string.
2. Display the contents of the Alpha register.

To create a two line display, insert the *line feed* character (PUNC [▼] ␊) into the Alpha register as part of your message. When you execute AVIEW or PROMPT, characters following the line feed character are displayed on the second line of the display.

You can use more than one line feed character to produce multiline messages on the printer. However, since the calculator has a two-line display, anything following the second line feed character (within the same message) cannot be displayed.

The AVIEW Function. The AVIEW function displays the contents of the Alpha register. Depending on the status of flags 21 and 55, AVIEW may halt program execution or produce printer output. Refer to "Printing With VIEW and AVIEW" on page 132.

The PROMPT Function. The PROMPT function displays the contents of the Alpha register just as AVIEW does. However, PROMPT always halts program execution and only generates printer output in Normal and Trace printing modes.

Entering Alpha Strings Into Programs

An Alpha string entered as a program line—called an *Alpha constant*—is placed into the Alpha register when that line is executed. For a normal Alpha constant, like the one that follows, the Alpha string *replaces* the previous contents of the Alpha register.

```
01 "This is an"
```

If the *append symbol* precedes an Alpha string, the calculator appends the string to the current contents of the Alpha register.*

```
02 ⊢" Alpha String"
```

Append symbol

After executing these two program lines, the Alpha register contains:

```
This is an Alpha String
```

The "SMILE" program on page 139 uses program lines like this to create a special string in the Alpha register.

To key an Alpha string into a program:

1. Press ■ [ALPHA] to display the ALPHA menu.
2. Optional: press [ENTER] to insert the append symbol (⊢).
3. Type the string.
4. Press [ENTER] or ■ [ALPHA] to complete the string.

An Alpha string in a program may be up to 15 characters long. (The append symbol counts as a character.)

If the Alpha register fills up (44 characters), appending more characters pushes the left-most (oldest) characters out of the Alpha register.

* Note that some printers may not be able to print the append character.

This program displays three consecutive messages:

```
01 "Hello there,"
02 AVIEW
03 PSE
04 "this program"
05 AVIEW
06 PSE
07 "has 3 messages."
08 AVIEW
09 END
```

Without the PSE instructions (lines 03 and 06) the program would run too fast to see the first two messages. A PSE is not needed after the last AVIEW because the viewed information remains in the display after the program stops. Pressing a key during a PSE causes program execution to halt. Press [R/S] to resume program execution.

Printing During Program Execution

Printing is another important form of program output. For a complete description of print functions and modes, read chapter 7, "Printing."

Using Print Functions in Programs

When a print function (such as PRX, PRA, or PRV) is encountered in a running program, the calculator tests flags 21 and 55. In general, flag 21 (*printer enabled*) determines if printing is *desired* and flag 55 (*printer existence*) determines if printing is *possible*.

Flag 21	Flag 55	Result of Print Function
Clear	Set or clear	The print function is ignored and program execution continues with the next line.
Set	Clear	Program execution halts and displays `Printing Is Disabled`.
Set	Set	The print function is executed and the program continues.

Printing With VIEW and AVIEW

Like print functions, VIEW and AVIEW also test flags 21 and 55. In addition to performing their normal display functions, VIEW and AVIEW produce printed output if flags 21 and 55 are set.

To record results, set flag 21. If a program uses VIEW or AVIEW to display important results, set flag 21. Then if printing is enabled (flag 55 set), the information is printed.

If printing is disabled (flag 55 clear), the program stops so you can write down the displayed information. Press [R/S] to continue.

To display, but not record messages, clear flag 21. If flag 21 is clear, flag 55 is ignored by VIEW and AVIEW. The information is displayed and program execution continues.

Working With Alpha Data

This section describes the functions for manipulating data in the Alpha register. All of the techniques presented here can be executed manually; however, they are primarily meant for programming.

Moving Data Into and Out of the Alpha Register

In addition to keying data directly into the Alpha register or entering strings in programs, there are several ways to move data into and out of the Alpha register.

Storing Alpha Data. The ASTO (*Alpha store*) function copies the first six characters in the Alpha register into the specified variable or register. To execute the ASTO function:

1. If Alpha mode is not on, press ■ [ALPHA].
2. Press [ASTO]. (The [STO] key executes ASTO when Alpha mode is on.)

3. Specify where you want the string to be stored:
   - *In a storage register.* Key in the register number.
   - *In a variable.* Press a menu key to select the variable or use the ALPHA menu to type the name.
   - *In a stack register.* Press [·] followed by ST L, ST X, ST Y, ST Z, or ST T.

For example, to copy the first six characters of the Alpha register into the X-register, press [ALPHA] [ASTO] [·] ST X.

Recalling Data Into the Alpha Register. The ARCL (*Alpha recall*) function recalls data into the Alpha register, appending it to the current contents. To execute the ARCL function:

1. If Alpha mode is not on, press [ALPHA].
2. Press [ARCL]. (The [RCL] key executes ARCL when Alpha mode is on.)
3. Specify the storage register, variable, or stack register you want to recall. (Refer to step 3 above.)

If you recall a number into the Alpha register, it is converted to Alpha characters and formatted using the current display format. Recalling a matrix into the Alpha register recalls its descriptor (such as [2×3 Matrix]).

When the Alpha register fills up, characters at the left end of the register (the "oldest" characters) are lost to make room for the new data.

To recall an integer into the Alpha register:

1. Place the number in the X-register.
2. Press [PGM.FCN] [▼] [▼] AIP (*Alpha append integer part*). The AIP function appends the integer part of the number in the X-register to the current contents of the Alpha register.

You can produce a similar result by using FIX 0 display format, clearing flag 29 (to remove the decimal point), and recalling a number using ARCL. A number recalled this way, however, may be rounded if the fractional part of the number is greater than or equal to 0.5.

To translate a number into a character:

1. Key in the character code (the allowed range is 0 through 255). Appendix E lists all of the display characters and their character codes.
2. Press ■ [PGM.FCN] [▼] [▼] XTOA (*X to Alpha*).

If the X-register contains an Alpha string, the entire string is appended to the Alpha register.

If the X-register contains a matrix, the XTOA function uses each element in the matrix as a character code or Alpha string. XTOA begins with the first element (1:1) and continues rowwise (to the right) until it reaches the end of the matrix. If the Alpha register fills up, only the last 44 characters to be appended will remain.

The XTOA function is especially useful for building a graphics string in the Alpha register. Refer to the program on page 139.

To translate a character into its character code: Execute the ATOX (*Alpha to* X) function. ATOX converts the left-most character in the Alpha register into its character code (0 through 255) and returns the number to the X-register. The character is deleted from the Alpha register, shifting the rest of the string left one position. If the Alpha register is empty, ATOX returns zero.

For example, if the Alpha register contains `Janet`, executing ATOX deletes the `J` and returns its character code (74) to the X-register.

Searching the Alpha Register

To search the Alpha register for a character or string, use the POSA (*position in Alpha*) function. POSA searches the Alpha register for the *target* in the X-register. If a match is found, POSA returns the position number where the target was found (counting the left-most character as position 0). If a match is not found, POSA returns −1.

The target may be a character code or an Alpha string. POSA saves a copy of the target in the LAST X register.

Manipulating Alpha Strings

Once a string is in the Alpha register, there are several functions you can use to manipulate the data.

Finding the Length of an Alpha String. The ALENG (*Alpha length*) function returns to the X-register the number of characters in the Alpha register.

Shifting the Alpha Register. The ASHF (*Alpha shift*) function deletes the six left-most characters in the Alpha register. You may want to shift characters out of the Alpha register after using the ASTO function.

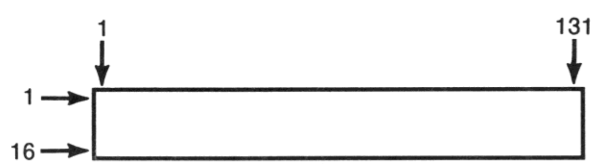
Rotating the Alpha Register. The AROT (*Alpha rotate*) function rotates the contents of the Alpha register by *n* characters (*n* is specified in the X-register). If *n* is positive, the rotation is to the left. If *n* is negative, the rotation is to the right.

Graphics

Using the functions PIXEL and AGRAPH (*Alpha graphics*), you can create graphics in the display of the HP–42S. The "DPLOT" and "PLOT" programs in the next chapter use the PIXEL function to produce graphs of functions (pages 156 and 160).

Turning On a Pixel in the Display

The PIXEL function turns on a pixel (one dot in the display) using the numbers in the X- and Y-registers. The *x*-value specifies the column (numbered from left to right; 1 through 131), and the *y*-value specifies the row (numbered from top to bottom; 1 through 16).



For a program to turn on a pixel in the display, it should:

1. Put the row number in the Y-register and the column number in the X-register.
2. Execute the PIXEL function (■ [PGM.FCN] [▼] [▼] PIXEL).

Executing PIXEL turns on the specified pixel and sets the message flags (flags 50 and 51). This allows subsequent PIXEL and AGRAPH instructions to add to the existing display.

To start with a clear display, execute CLLCD (*clear liquid crystal display*) before turning pixels on.

Drawing Lines in the Display

The PIXEL function can also be used to draw vertical and horizontal lines across the display. To draw a vertical line, use a negative *x*-value (−1 through −131). To draw a horizontal line, use a negative *y*-value (−1 through −16). If both numbers are negative, then PIXEL draws two lines—one vertical and one horizontal.

The plotting programs at the end of the next chapter use this feature of PIXEL to draw an *x*-axis.

Building a Graphics Image Using the Alpha Register

To create a graphics image in the display, a program should:

1. Create a string of characters in the Alpha register with each character specifying a column of eight pixels.
2. Specify where in the display the upper-left corner of the image should begin. Put that pixel-row number in the Y-register and the pixel-column number in the X-register.
3. Execute the AGRAPH function (■ [PGM.FCN] [▼] [▼] AGRA).

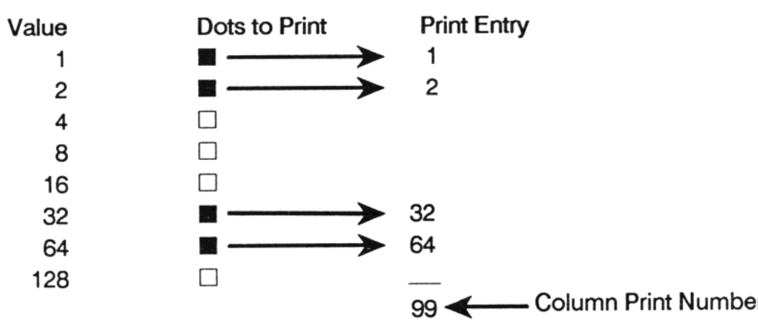
The status of flags 34 and 35 determine how the graphics image is displayed:

Flag 34	Flag 35	How the AGRAPH Image is Displayed
Clear*	Clear*	The image is merged with the existing display (logical OR).
Clear	Set	The image overwrites all pixels in that portion of the display.
Set	Clear	Duplicate "on" pixels get turned "off."
Set	Set	All pixels are reversed (logical XOR).

* Default setting.

Creating an Alpha String for AGRAPH. The AGRAPH function uses the character code of each character in the Alpha register as an eight-bit pattern for a column of pixels.

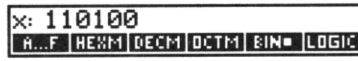
Each pixel in a column has a special value. Adding the values for all the pixels you want to display in a single column gives you the character code needed to produce that column.



To append the character to the Alpha register, key in the character code and then execute the XTOA function. You can type the character directly into the Alpha register if it is a typeable character. (Refer to the character table in appendix E.) For example, the character code 99 (calculated above) is the code for "c".

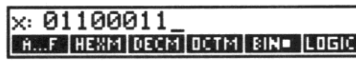
Example: Using Binary Mode to Calculate a Column Value. You can use the built-in Base application* to convert a column pattern into a character code. For example, select the Base application and Binary mode.

BASE BINM



Key in the column pattern above as a binary number. Start at the bottom, keying in a 0 for "off" pixels and a 1 for "on" pixels. (You can omit the leading zero if you want.)

01100011

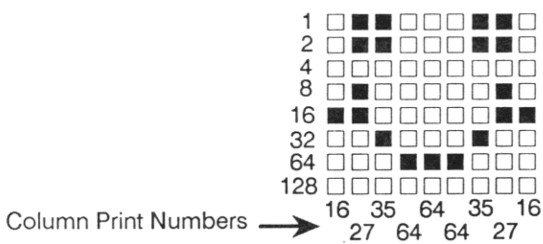


Display this number in Decimal mode.

DECM

You don't have to use Decimal mode to use this number. While still in Binary mode, you could append the character to the Alpha register using the XTOA function.

Example: Displaying a Happy Face. The program below creates this happy face in the display.



Use the character table in appendix E to look up these character codes. If the table does not have keystrokes for a particular character (in this case character number 27), then append it to the Alpha register with the XTOA funciton. Refer to lines 03, 04, and 06 in the following program.

* Refer to chapter 16 for more information on the Base application.

01 LBL "SMILE"	
02 "←"	Character number 16.
03 27 04 XTOA	Character number 27.
05 ⊦"#@@@#"	Character numbers 35, 64, 64, 64, and 35.
06 XTOA	Character 27 (X-register still contains 27).
07 ⊦"←"	Character number 16.
08 5 09 62	Specifies the location of the image: row 5, column 62. (To key in the two numbers press 5 [ENTER] [◆] 62.)
10 CLLCD 11 AGRAPH 12 END	Displays the image and stops.

Key in the "SMILE" program. (If you are still in the Base application from the previous example, press [EXIT].)

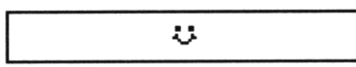
■ [GTO] [·] [·] ■ [PRGM]	00▶{ 0-Byte Prgm } 01 .END.
■ [PGM.FCN] LBL SMILE [ENTER]	00 { 9-Byte Prgm } 01▶LBL "SMILE"
■ [ALPHA] ← [ENTER]*	01 LBL "SMILE" 02▶"←"
27	02 "←" 03▶27_
■ [PGM.FCN] [▼] [▼] XTOA	03 27 04▶XTOA

* After displaying the ALPHA menu (■ [ALPHA]), the keystrokes to type ← are: [▼] ←↑↓ ← .

Keys	Display
■ [ALPHA] [ENTER] #@@@# *	05▶⊢"#@@@#_ ([{ ←↑↓ < = > MATH PUNC MISC
■ [PGM.FCN] [▼] [▼] XTOA	05 ⊢"#@@@#" 06▶XTOA
■ [ALPHA] [ENTER] ← [ENTER]	06 XTOA 07▶⊢"←"
5 [ENTER] [◀]	08▶5 09 .END.
62	08 5 09▶62_
■ [CLEAR] [▼] CLLCD	09 62 10▶CLLCD
■ [PGM.FCN] [▼] [▼] AGRA	10 CLLCD 11▶AGRAPH

Now exit from Program-entry mode and run the program.

[EXIT] [XEQ] SMILE



* After displaying the ALPHA menu and the append character (■ [ALPHA] [ENTER]), the keystrokes to type #@@@# are: [▼] MISC # MISC [▼] @ MISC [▼] @ MISC [▼] @ MISC #.

10

Programming Techniques

This chapter covers functions and techniques for writing more sophisticated programs. You'll learn how to use:

- GTO (*go to*) and XEQ (*execute*) instructions to cause program branching to execute subroutines and other programs.
- The programmable menu to create *menu-driven* programs.
- Conditional tests and counters to create program *loops* (routines that repeat themselves).
- Tests and comparisons to make decisions and cause program branching.

Branching

Branching occurs whenever the program pointer moves to a line other than the "next" line—that is, whenever program instructions are not executed sequentially. The two primary functions for branching are GTO and XEQ.

Often flag tests and comparisons are followed by branching instructions that are executed according to the result of the test or comparison.

Branching to a Label (GTO)

Labels can be considered *destinations* for branching instructions. As explained in chapter 8, global labels can be accessed from anywhere in memory and local labels can be accessed only from within their own program.

There are three programmable forms of GTO instructions:

- `GTO` *nn* for branching to a local numeric label (where *nn* is the label number).
- `GTO` *label* for branching to a local Alpha label (where *label* is a single letter `A` through `J` or `a` through `e`).
- `GTO "`*label*`"` for branching to a global label (where *label* is the Alpha label).

Here are a few examples:

Example Instruction:	**Description (Keys):**
`GTO 03`	Branches to `LBL 03` (■ GTO 03).
`GTO A`	Branches to `LBL A` (■ GTO ENTER A ENTER).
`GTO "AREA"`	Branches to `LBL "AREA"` (■ GTO AREA).

Executing GTO in a Program. In a running program, a GTO instruction causes program execution to branch to the specified label and continue running at that line.

Executing GTO From the Keyboard. Executing a GTO instruction from the keyboard moves the program pointer to the corresponding label. No program lines are executed.

Indirect Addressing With GTO. The following examples show how indirect addressing can be used with GTO instructions. That is, the label to be branched to is specified in a variable or register.

Example Instruction:	Description (Keys):
`GTO IND 12`	Branches to the label specified in storage register R_{12} (■ [GTO] [·] IND 12). For example, if R_{12} contains the string "AREA", then program execution branches to `LBL "AREA"`.
`GTO IND "ABC"`	Branches to the label specified in the variable *ABC* (■ [GTO] [·] IND ABC). For example, if *ABC* contains the number 17, then program execution branches to `LBL 17`.
`GTO IND ST X`	Branches to the label specified in the X-register (■ [GTO] [·] IND [·] ST X). For example, if the X-register contains the number 96, then program execution branches to `LBL 96`.

Calling Subroutines (XEQ and RTN)

The GTO function, described above, is used to make a simple program branch. XEQ is used in much the same way with one important difference: *after* an XEQ instruction has transferred execution to the specified label, the next RTN (*return*) or END instruction causes the program to branch *back* to the instruction that immediately follows the XEQ instruction.

XEQ instructions are *subroutine calls*. A subroutine call is not complete until a RTN or END has been executed to return program execution to the line following the XEQ instruction.

XEQ is also used to run programs from the keyboard ([XEQ]).

Example: GTO versus XEQ. Consider the following two programs. If you execute the first program ([XEQ] PRG1), `TONE 0` never executes because the GTO instruction branches to the second program. Program execution halts when the END is reached in the second program.

```
01 LBL "PRG1"
02 GTO "PRG2"
03 TONE 0
04 END
```

```
01 LBL "PRG2"
02 TONE 9
03 END
```

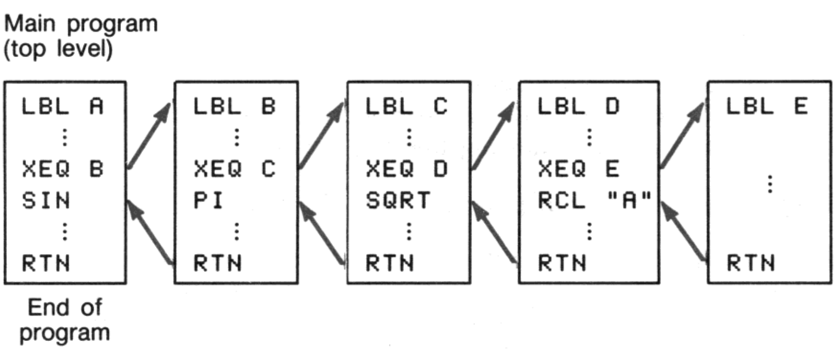
However, if you replace line 02 of the first program with an XEQ instruction (`XEQ "PRG2"`), both TONEs will sound. When the END is encountered in the second program, execution returns to the line immediately following the XEQ. Program execution halts at the END in the first program.

```
01 LBL "PRG1"
02 XEQ "PRG2"
03 TONE 0
04 END
```

```
01 LBL "PRG2"
02 TONE 9
03 END
```

Subroutine Return Locations. When an XEQ instruction calls a subroutine, the HP-42S remembers the location of that XEQ instruction so that execution can return there when the subroutine is completed.

For example, this illustration shows how the calculator *nests* subroutines by remembering return locations. The HP-42S can remember up to eight pending return locations.



Loss of Subroutine Returns. Pending return locations are lost under the following conditions:

- If there are already eight pending return locations when another subroutine or program is called with an XEQ, the first (oldest) return location is lost.* In this case, program execution never returns to the first XEQ that called a subroutine. Instead, execution halts when the first subroutine is finally completed because there are no further return locations.
- All pending return locations are lost when you execute any program from the keyboard or perform any other operation (while program execution is halted) that alters the program pointer. Pressing ■ [SST] or [R/S] does not cause return locations to be lost.

The Programmable Menu

The HP-42S has a programmable menu which is used to cause program branching. The MENU function selects the programmable menu. The menu is displayed when the program stops. You can define each key in the menu so that when the key is pressed, a particular GTO or XEQ instruction executes. You can even define [▲], [▼], and [EXIT].

To define a menu key:

1. Enter a string into the Alpha register. This is the text that appears in the menu label above the key. (The Alpha register is not used when defining [▲], [▼], or [EXIT].)
2. Execute KEYG (*on key, go to*) or KEYX (*on key, execute*). (These functions are in the last row of the PGM.FCN menu; press ■ [PGM.FCN] [▲].)
3. Specify which key you want to define:
   - Press [Σ+], [1/x], [√x], [LOG], [LN], [XEQ], [▲], [▼], or [EXIT].
   - *Or*, key in the key number, 1 through 9.

* The Solver and Integration applications also create return locations. If the calculator loses one of these returns, program execution stops and an error message is displayed.

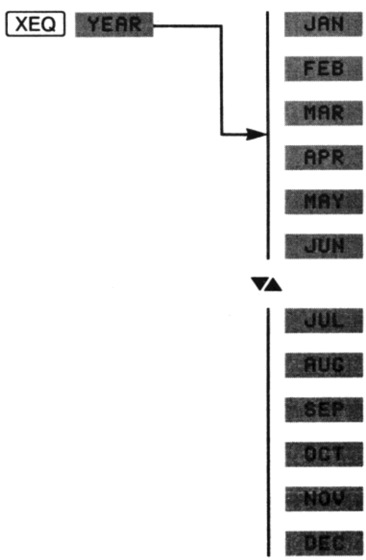
4. Specify a program label using *one* of these methods:
   - Select an existing global label by pressing the corresponding menu key.
   - Use the ALPHA menu to type an Alpha label (local or global): [ENTER] *label* [ENTER].
   - Key in a two-digit numeric label.

Repeat this procedure for each menu key you want to define. Defining a key replaces any previous definition that may exist for that key.

To display the programmable menu: Execute the MENU function (press ■ [PGM.FCN] [▲] MENU).

To clear all programmable menu key definitions: Execute the CLMENU (*clear menu*) function (press ■ [CLEAR] [▼] CLMN).

Example. The program segment listed below shows how the programmable menu can be used to emulate this menu:



```
01 LBL "YEAR"
02 LBL A
03 "JAN"
04 KEY 1 XEQ 01
05 "FEB"
06 KEY 2 XEQ 02
07 "MAR"
08 KEY 3 XEQ 03
09 "APR"
10 KEY 4 XEQ 04
11 "MAY"
12 KEY 5 XEQ 05
13 "JUN"
14 KEY 6 XEQ 06
```

Defines the first row of the "YEAR" menu. A different subroutine is executed for each month. The routines for the first six months are labeled with local labels 01 through 06.

```
15 KEY 7 GTO B
16 KEY 8 GTO B
17 KEY 9 GTO 99
```

Defines the [▲], [▼], and [EXIT] keys. The [▲] and [▼] keys are defined to go to the same program label (LBL B) because this is a two-row menu; either key should display the second row. The [EXIT] key is defined to cause a branch to a routine that exits the menu.

```
18 MENU
19 LBL 20
20 STOP
21 GTO 20
```

The programmable menu is selected and the program stops. Because of this little loop, pressing [R/S] keeps the program at line 20.

```
22 LBL B
23 "JUL"
24 KEY 1 XEQ 07
25 "AUG"
26 KEY 2 XEQ 08
27 "SEP"
28 KEY 3 XEQ 09
29 "OCT"
30 KEY 4 XEQ 10
31 "NOV"
32 KEY 5 XEQ 11
33 "DEC"
34 KEY 6 XEQ 12
```

Defines the menu keys for the second row of the "YEAR" menu.

`35 KEY 7 GTO A` `36 KEY 8 GTO A`	Defines the [▲] and [▼] to return to the first row of the menu. The [EXIT] key does not need to be define again. The definition made at line 17 is still in effect.
`37 LBL 21` `38 STOP` `39 GTO 21`	Stops the program. The programmable menu is still selected (line 18).
`40 LBL 99` `41 CLMENU` `42 EXITALL` `43 RTN`	The menu definitions are cleared and the menu is exited. If this program was called as a subroutine from another program, execution returns to that program.
`44 LBL 01` ⋮	The rest of the program consists of the subroutines for each month (LBL 01 ... RTN, LBL 02 ... RTN, and so on). For example, you might want to create a message in each of these subroutines that displays the full name of the month and the number of days in that month.

Many examples in the *HP-42S Programming Examples and Techniques* manual (part number 00042-90020) use the programmable menu.

Local Label Searches

Searches for local labels occur only within the current program. To find a local label, the calculator first searches sequentially downward through the current program, starting at the location of the program pointer. If the specified label is not found before reaching the end of the program, the calculator continues the search from the beginning of the program.

A local label search can consume a significant amount of time, depending on the length of the current program and the distance to the label. To minimize searching time, the calculator remembers the distance from the GTO or XEQ instruction to the specified local label.* This eliminates the searching time for subsequent executions of that same GTO or XEQ instruction.

Global Label Searches

When the calculator searches for a global label, the search begins with the *last* global label (bottom of program memory) and proceeds *upward,* stopping at the first label that matches the specified label. The search is in the same order as the labels are listed in the program catalog.

Conditional Functions

Flag tests and comparisons are *conditional functions.* They express a proposition that is either true or false depending on current conditions.

- Executing a conditional function from the keyboard generates a message: `Yes` if the proposition is currently true, or `No` if the proposition is currently false.
- Executing a conditional function in a program causes a program branch using the *do-if-true* rule. That is, the program line immediately following the conditional is executed *only* if the condition is true. If the condition is false, the next line is *skipped.* That is, DO the next instruction IF the condition is TRUE.

* The distance to the label is stored internally as part of the GTO or XEQ instruction. If this distance is greater than 4,096 bytes in either direction (128 bytes for short form lables; LBL 00 through LBL 14), the calculator cannot store the distance and a search must take place for each execution of the instruction.

Flag Tests

The following table shows the four flag test functions and how each causes program branching (a skipped line) based on the status of the flag being tested. (These functions are in the FLAGS menu.)

Flag Test	If Flag Is Set	If Flag Is Clear
FS?	Execute the next program line.	Skip the next program line.
FC?	Skip the next program line.	Execute the next program line.
FS?C*	Clear flag and execute the next program line.	Clear flag and skip the next program line.
FC?C*	Clear flag and skip the next program line.	Clear flag and execute the next program line.

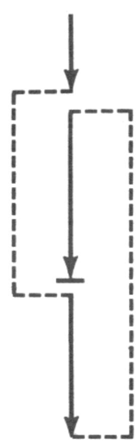
* This function can only be used with flags 00 through 35 and 81 through 99.

The following program demonstrates a subroutine call (line 03) and flag tests (lines 02 and 08). If flag 10 is clear, `FIRST` is displayed, and then `SECOND`. If flag 10 is set, the order of the messages is reversed.

Flag 10 Clear

```
01 LBL "FTEST"
02 FS? 10
03 XEQ B
04 LBL A
05 "FIRST"
06 AVIEW
07 PSE
08 FS? 10
09 STOP
10 LBL B
11 "SECOND"
12 AVIEW
13 PSE
14 END
```

Flag 10 Set



Comparisons

To compare the X-register with zero:

1. Press ■ [PGM.FCN] [▼] X?0 .
2. Press X=0? , X≠0? , X<0? , X>0? , X≤0? , or X≥0? .

To compare the X-register with the Y-register:

1. Press ■ [PGM.FCN] [▼] X?Y .
2. Press X=Y? , X≠Y? , X<Y? , X>Y? , X≤Y? , or X≥Y? .

If you execute one of these functions from the keyboard, the calculator displays Yes or No, indicating the result of the test. If a program executes one of these functions, the calculator follows the do-if-true rule.

Testing the Data Type

The following four functions test the type of data in the X-register. They also follow the do-if-true rule for program execution.

Function	Test Proposition
REAL?	Does the X-register contain a real number?
CPX?	Does the X-register contain a complex number?
MAT?	Does the X-register contain a matrix?
STR?	Does the X-register contain an Alpha string?

Bit Test

The BIT? (*bit test*) function tests a single bit of a number. If the x^{th} bit of y is a 1, then the test is true. Refer to chapter 16 for more information on the Base application and logic functions.

Looping

A loop is a sequence of program instructions that starts with a label and ends with a branch back to that label. An infinite loop is the simplest kind. Once started, this program runs until you stop it with [R/S] or [EXIT].

```
01 LBL "LOOP"
02 BEEP
03 GTO "LOOP"
04 END
```

Looping Using Conditional Functions

When you want to perfom an operation until a certain condition is met, but you don't know how many times the loop must repeat, you can create a loop with a conditional test and a GTO instruction.

For example, the following program loops until the RAN (*random number*) function returns a number that is at least 0.9. That is, the loop repeats if the random number is less than 0.9.

```
01 LBL "RANDOM"
02 LBL 01
03 0.9
04 RAN
05 X<Y?
06 GTO 01
07 END
```

Why does this program have two labels? Since the HP-42S only has to search for a local label once, the loop executes faster by branching to a local label. (Refer to "Local Label Searches" on page 148.) What's more, using a local label and corresponding GTO instruction (rather than branching to the global label) saves five bytes of program memory.

Loop-Control Functions

When you want to execute a loop a specific number of times, you can use special functions for that purpose—ISG (*increment, skip if greater*) and DSE (*decrement, skip if less than or equal*). Both functions (located in the PGM.FCN menu) take a parameter identifying the variable or register containing the number that controls the looping.

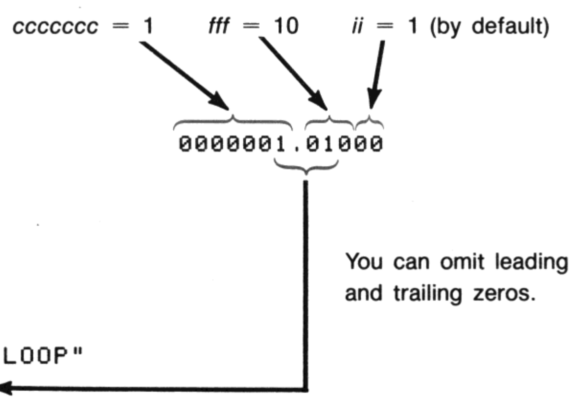
The format of the loop-control number is *ccccccc.fffii,* where:

- *ccccccc* is the current counter value. Executing ISG or DSE increments or decrements *ccccccc* by the value of *ii*.
- *fff* is the final counter value.
- *ii* is the increment/decrement value. If *ii* is 00 (or unspecified), the calculator uses a default value of 01.

Executing ISG increments *ccccccc* by *ii,* and then compares the resulting value of *ccccccc* with *fff*. If *ccccccc* is greater than *fff,* the next program instruction is skipped.

Executing DSE decrements *ccccccc* by *ii,* and then compares the resulting value of *ccccccc* with *fff*. If *ccccccc* is less than or equal to *fff,* the next program instruction is skipped.

Example: Using the ISG Function. The following program uses ISG to loop 10 times. The loop counter is stored into a variable named *COUNTER* and is interpreted by the ISG function like this:




```
01 LBL "LOOP"
02 1.01
03 STO "COUNTER"
```

```
04 LBL 01
05 VIEW "COUNTER"
06 PSE
07 ISG "COUNTER"
08 GTO 01
09 "DONE"
10 AVIEW
11 END
```

Controlling the CUSTOM Menu

If flag 27 is set when a program stops, the CUSTOM menu is displayed. Before displaying the menu, however, the calculator also checks flag 72.* If flag 72 is clear (indicated by KEY■ in the MODES menu), the CUSTOM menu displays menu assignments you have made. If flag 72 is set (indicated by LCLB■ in the MODES menu), the CUSTOM menu displays keys for executing local labels (page 301).

Example Programs

The programs in this section use many of the functions and techniques presented in chapters 8, 9, and 10. By examining them and using them, you should gain an even better understanding of programming. For many more programming examples, refer to the *HP-42S Programming Examples and Techniques* manual (part number 00042-90020).

The Display Plot Program ("DPLOT")

The "DPLOT" program plots a function in the display of the calculator. The function that you plot is entered into the calculator as a program. There are two general forms for a function program:

* The calculator also checks flag 72 when you use ■ [CUSTOM] to display the CUSTOM menu.

- *As f(x)*, where the program returns a value using an input value in the X-register. For example, to plot a sine curve ($f(x) = \sin x$), use a program like this:

```
01 LBL "SINE"
02 SIN
03 END
```

- *As a Solver program*. If the program uses menu variables, it is assumed to be written in the proper form for use with the Solver. Refer to "Writing a Program for the Solver" on page 179.

The name of the function is stored in a variable named *FCN*. Since Alpha strings stored in variables are limited to six characters, the global label that you use to identify the function cannot be longer than six characters.

You can determine what portion of the function is plotted by entering the limits of the plot:

YMIN = bottom of the display
YMAX = top of the display
XMIN = left end of the display
XMAX = right end of the display

You can also specify where you would like the x-axis to appear. Usually, the axis is at $y = 0$. If you don't want an axis, specify a y-value that is less than *YMIN* or greater than *YMAX*.

To use the "DPLOT" program:

1. Key the "DPLOT" program into your calculator. (The "DPLOT" program uses 234 bytes of program memory.)
2. Key in a program for the function you want to plot.
3. Press [XEQ] DPLOT. The program displays a variable menu containing *YMIN*, *YMAX*, *AXIS*, *XMIN*, and *XMAX*. Store a value into each variable: key in a number and then press the corresponding menu key.
4. Press [R/S]. The program displays the current function name stored in *FCN* (if there is one) along with the Alpha menu.
5. If necessary, type the name of the function you want to plot.

6. Press [R/S]. If the function does not use menu variables, plotting begins.
7. If the function does use menu variables, the program stops and displays the variable menu. Using the variable menu:
   a. Store a value into each of the known variables: key in a number and then press the corresponding menu key.
   b. Press a menu key to select the plot variable. Plotting begins.

When the plot is finished, the program prints a copy of the display (if printing is enabled).

The example on page 185 uses "DPLOT" to plot a function for the Solver.

Program:	**Comments:**
`01 LBL "DPLOT"` `02 MVAR "YMIN"` `03 MVAR "YMAX"` `04 MVAR "AXIS"` `05 MVAR "XMIN"` `06 MVAR "XMAX"`	Declares the menu variables.
`07 LBL A` `08 VARMENU "DPLOT"` `09 "Ready"` `10 PROMPT`	Selects the variable menu, displays a `Ready` message, and stops the program.
`11 CLA` `12 SF 25` `13 RCL "FCN"` `14 CF 25` `15 STR?` `16 ARCL ST X`	Recalls the current function name (if there is one) into the Alpha register.
`17 AON` `18 STOP`	Turns on the ALPHA menu and stops the program so a function name can be entered or changed.

```
AOFF
ALENG
X=0?
GTO A
ASTO "FCN"
```

Turns off the ALPHA menu and tests the length of the Alpha register. If the Alpha register is empty, execution returns to the first variable menu. Otherwise, the function name is stored into *FCN*.

```
CLA
CF 81
SF 25
VARMENU IND "FCN"
FC?C 25
SF 81
```

Selects the variable menu for the function. If there are no menu variables, flag 81 is set.

```
FC? 81
STOP
EXITALL
ALENG
X=0?
SF 81
ASTO 03
```

Stops to display the variable menu (if flag 81 is clear). Tests the Alpha register to see if a plot variable has been selected. If not, flag 81 is set. The variable name is stored into R_{03}.

```
15
RCL "YMAX"
RCL- "YMIN"
÷
STO 00
```

Calculates the *y*-value of one pixel.

```
RCL "XMIN"
STO 01
1.131
STO 02
```

Stores the first *x*-value and a loop counter. (There are 131 pixels across the display.)

```
CLLCD
XEQ "AXIS"
```

Clears the display and draws an axis.

```
LBL 01
RCL 01
FC? 81
STO IND 03
XEQ IND "FCN"
```

Recalls the current *x*-value. If flag 81 is clear, the *x*-value is stored into the plot variable. The function is then evaluated using the current *x*-value.

```
53 XEQ 02
54 RCL 02
55 PIXEL
```

The value of the function is converted into a pixel number.

```
56 RCL "XMAX"
57 RCL- "XMIN"
58 131
59 ÷
60 STO+ 01
```

The x-value is incremented.

```
61 ISG 02
62 GTO 01
63 PRLCD
64 RTN
65 GTO A
```

If the plot is done, the display is printed and the program stops. Line 65 allows the program to be restarted by pressing [R/S].

```
66 LBL 02
67 RCL- "YMIN"
68 RCL× 00
69 16
70 -
71 X>0?
72 CLX
73 ABS
74 RTN
```

Calculates a pixel number for the given y-value.

```
75 LBL "AXIS"
76 RCL "AXIS"
77 XEQ 02
78 +/-
79 1
80 PIXEL
81 END
```

Draws an x-axis.

The Printer Plot Program ("PLOT")

The "PLOT" program plots a function on the HP 82240A printer. The plot is created in sections. Each section is plotted in the display and then printed. The result is a continuous plot of the function on a strip of paper. (The x-axis runs lengthwise on the paper.)

Before plotting a function, you must write a program that expresses the function. The name of the function is stored into a variable named *FCN*. Since Alpha strings stored in variables are limited to six characters, the global label that you use to identify the function must be six or fewer characters.

You can determine what portion of the function is plotted by entering the limits of the plot:

YMIN = left edge of paper
YMAX = right edge of paper
XMIN = beginning x-value
XMAX = ending x-value
XINC = increment of x-values

The x-values are printed at increments determined by *XINC*. If you do not want these labels on your plot, set flag 00.

You can specify where you would like the x-axis to appear. Usually, the axis is at $y = 0$. If you do not want an axis, set flag 01.

To use the "PLOT" program:

1. Key the "PLOT" program into your calculator. (The "PLOT" program uses 337 bytes of program memory.)
2. Key in a program for the function you want to plot.
3. Press [XEQ] **PLOT**. The program displays a variable menu containing *YMIN*, *YMAX*, *AXIS*, *XMIN*, *XMAX*, and *XINC*. Store a value into each variable: key in a number and then press the corresponding menu key.
4. Press [R/S]. The program displays the current function name stored in *FCN* (if there is one) along with the Alpha menu.
5. If necessary, type the name of the function you want to plot.
6. Press [R/S] to begin the plot.

```
01 LBL "PLOT"
02 MVAR "YMIN"
03 MVAR "YMAX"
04 MVAR "AXIS"
05 MVAR "XMIN"
06 MVAR "XMAX"
07 MVAR "XINC"
```

Declares the menu variables.

```
08 LBL A
09 VARMENU "PLOT"
10 STOP
```

Selects the variable menu and stops the program.

```
11 EXITALL
12 XEQ 07
```

Exits from the variable menu and inputs a function name.

```
13 PRON
14 ADV
15 "Plot of:"
16 PRA
17 ADV
18 SF 12
19 CLA
20 ARCL "FCN"
21 PRA
22 ADV
23 CF 12
24 PRV "YMIN"
25 PRV "YMAX"
26 PRV "AXIS"
27 PRV "XMIN"
28 PRV "XMAX"
29 PRV "XINC"
30 ADV
31 "← YMIN"
32 ├"         YMAX →␊"
33 PRA
```

Prints the header information.

34 130 35 RCL "YMAX" 36 RCL- "YMIN" 37 ÷ 38 STO 00	Calculates the y-value of one pixel.
39 RCL "XMIN" 40 STO 01	Stores the first x-value.
41 LBL 00 42 CLLCD	Clears the display.
43 FC? 00 44 XEQ 05	Labels the x-increment if flag 00 is clear.
45 FC? 01 46 XEQ 06	Draws an axis if flag 01 is clear.
47 1.016 48 STO 02	Stores a loop counter into R_{02}. (There are 16 rows of pixels in the display.)
49 LBL 01 50 RCL "FCN" 51 STR? 52 XEQ 04	Plots the current point.
53 RCL "XINC" 54 16 55 ÷ 56 STO+ 01	Increments the x-value.
57 RCL "XMAX" 58 RCL 01 59 X>Y? 60 GTO 03	Goes to LBL 03 if the plot is done.
61 ISG 02 62 GTO 01 63 PRLCD 64 GTO 00	Prints the display if all 16 values have been plotted.

```
65 LBL 03
66 PRLCD
67 RTN
68 GTO A
```

Prints the final display. Line 68 allows the program to be restarted by pressing R/S.

```
69 LBL 04
70 RCL 01
71 XEQ IND ST Y
72 SF 24
73 RCL- "YMIN"
74 RCL× 00
75 1
76 +
77 CF 24
78 RCL 02
79 X<>Y
80 X>0?
81 PIXEL
82 RTN
```

Evaluates the function at the current x-value and then plots the appropriate pixel.

```
83 LBL 05
84 CF 21
85 CLA
86 ARCL 01
87 AVIEW
88 SF 21
89 RTN
```

Puts an x-value into the display to label the x-axis.

```
90 LBL 06
91 1
92 RCL "AXIS"
93 RCL- "YMIN"
94 RCL× 00
95 +/-
96 1
97 -
98 PIXEL
99 +/-
100 2
101 -
102 "×××××"
103 AGRAPH
104 RTN
```

Plots an x-axis. Note that line 102 is a string of multiply characters (■ ALPHA × × × × × ENTER).

```
105 LBL 07
106 CLA
107 SF 25
108 RCL "FCN"
109 CF 25
110 STR?
111 ARCL ST X
112 AON
113 STOP
114 AOFF
115 ASTO "FCN"
116 END
```

Recalls the current function name (if there is one) into the Alpha register. Turns on the ALPHA menu and stops the program. When the program continues (when [R/S] is pressed), the function name is stored into *FCN*.

Example: Using the Printer Plot Program. Key in the "PLOT" program listed above and the program "MISCFN" below. Plot the function with *YMIN* = −0.5, *YMAX* = 2, *AXIS* = 0, *XMIN* = −360, *XMAX* = 360, and *XINC* = 45.

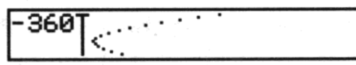
```
01 LBL "MISCFN"
02 ENTER
03 ENTER
04 360
05 ÷
06 X<>Y
07 3
08 ×
09 SIN
10 ×
11 1
12 +
13 END
```

■ [DISP] ALL [XEQ] PLOT

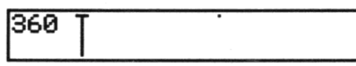
```
x: 0
YMIN YMAX AXIS XMIN XMAX XINC
```

.5 [+/-] YMIN

```
YMIN=-0.5
YMIN YMAX AXIS XMIN XMAX XINC
```

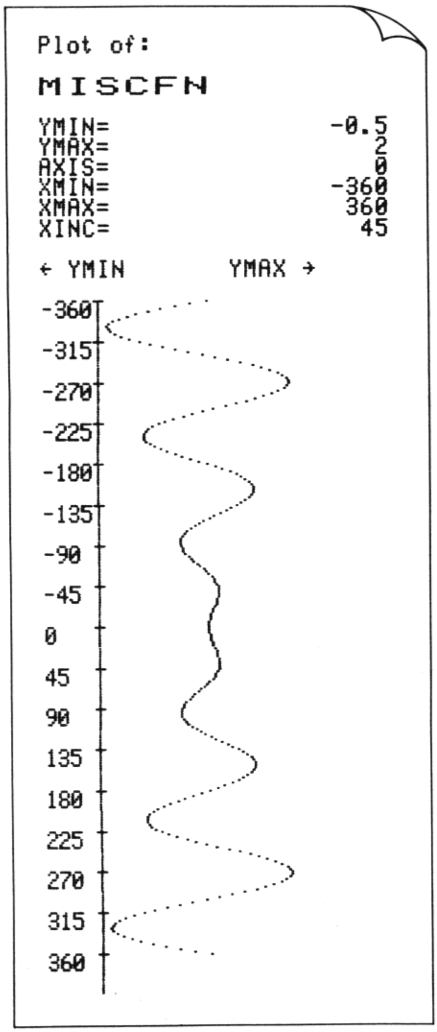
Keys:	Display:
2 YMAX	YMAX=2 YMIN YMAX AXIS XMIN XMAX XINC
0 AXIS	AXIS=0 YMIN YMAX AXIS XMIN XMAX XINC
360 XMAX	XMAX=360 YMIN YMAX AXIS XMIN XMAX XINC
[+/-] XMIN	XMIN=-360 YMIN YMAX AXIS XMIN XMAX XINC
45 XINC	XINC=45 YMIN YMAX AXIS XMIN XMAX XINC
[R/S]	ABCDE FGHI JKLM NOPQ RSTUV WXYZ
MISCFN [R/S]	

⋮



The printer output is shown on the following page.

Printer Output:



11

Using HP-41 Programs

All programmable functions of the HP-41C and HP-41CV calculators have been built into the HP-42S. This means that programs written for these HP-41* calculators will run on the HP-42S.

In addition to the HP-41C/CV function set, several new functions have been added to further enhance the programming capabilities of the HP-42S. As you become more familiar with programming, you will probably want to modify your favorite HP-41 programs to take advantage of the expanded function set of the HP-42S.

In this chapter you'll learn about:

- Special considerations you may need to make when running some HP-41 programs.
- Reading HP-41 program listings and keying programs into the HP-42S.
- Enhancing HP-41 programs.

Important Differences

While the HP-42S fully supports the function set of the HP-41C/CV calculators, there are some important differences that you should not overlook. Under most circumstances, these differences will add to the accuracy or capability of an existing HP-41 program. Some HP-42S operations, however, may need to be disabled so operation more closely emulates the HP-41.

* "HP-41" is used in this chapter to refer to the HP-41C and HP-41CV calculators. Not all extended functions built into the HP-41CX calculator are supported by the HP-42S.

HP-41 User Keyboard

The CUSTOM menu in the HP-42S provides capabilities that are similar to the User keyboard on the HP-41. That is, you can:

- Assign functions and programs to the CUSTOM menu.
- Use the CUSTOM menu to execute local labels in the current program.

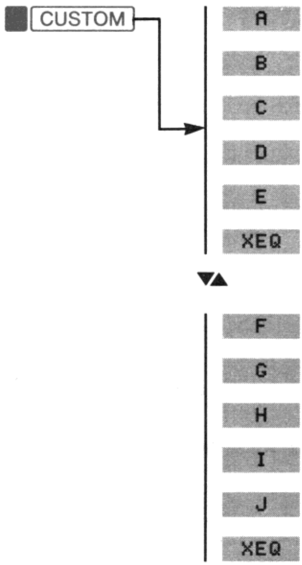
Flag 27, which is used on the HP-41 to control the User keyboard, is used to control the CUSTOM menu. In general, setting flag 27 is equivalent to pressing ■ [CUSTOM]. Clearing flag 27 is equivalent to pressing [EXIT] when the CUSTOM menu is displayed.

To use CUSTOM menu assignments:

1. If necessary, press ■ [MODES] [▼] KEY (*key assignments*) to select Key-assignment mode. The calculator selects this mode automatically each time you make an assignment to the CUSTOM menu (■ [ASSIGN]). The KEYASN function clears flag 72.
2. Press ■ [CUSTOM] or ■ [FLAGS] SF 27 to display the CUSTOM menu.

To use CUSTOM menu for executing local labels:

1. If necessary, press ■ [MODES] [▼] LCLBL (*local label*) to select *Local-label* mode. The LCLBL function sets flag 72.
2. Press ■ [CUSTOM] or ■ [FLAGS] SF 27 to display the CUSTOM menu.



Pressing A through J executes the instructions XEQ A through XEQ J. Use the shift key (■) to execute XEQ a through XEQ e (■ A through ■ E).

If you are using an HP-41 program that uses local Alpha labels, the instructions may say something like "Press [B]." When you're running the program, remember this means to press B. Similarly, if the instructions say "Press [b]," then press ■ B.

Statistical Operations

Statistical operations on the HP-42S have been expanded (beyond the capabilities of the HP-41) to include curve fitting and forecasting. These enhanced features require the use of seven more summation coefficients than the HP-41 uses.

To use only 6 summation coefficients (like the HP-41): Press ■ [STAT] [▼] LINΣ.

To use all 13 summation coefficients (default): Press ■ [STAT] [▼] ALLΣ.

Printer Interface

Because the HP-42S uses a one-way infrared printer interface, it cannot tell if a printer is receiving the infrared signal. It's up to you to tell the calculator if a printer is available.

To enable printing: Press ■[PRINT] [▲] PON .

To disable printing: Press ■[PRINT] [▲] POFF .

Refer to chapter 7, "Printing," for more information.

The Alpha Register

The Alpha register in the HP-42S is 44 characters long, which is 20 characters longer than the Alpha register in the HP-41. Programs that specifically require the Alpha register to be 24 characters long may not produce the desired output.

Range of Numbers

The HP-42S uses 15 digits (a 12-digit mantissa and a 3-digit exponent of ten) to represent all real numbers. The HP-41, however, uses a 10-digit mantissa and a 2-digit exponent. Therefore, because of this increased range, calculations that generate an "OUT OF RANGE" error on the HP-41 may not be out of range on the HP-42S.

Note that the HP-42S returns an Out of Range error for the tangent of 90°. The HP-41 returns 9.999999999 x 10^{99}.

Data Errors and the Real-Result Flag

Because of its complex-number capabilities, the HP-42S can return results for calculations that would not work on the HP-41. The HP-42S automatically returns a complex number for calculations such as:

- Square root of a negative number.
- Logarithm of a negative number.
- Arc sine or arc cosine of a number whose absolute value is greater than 1.

To disable complex results for real-number operations: Press ■ [MODES] [▼] RRES (*real results only*). This function sets flag 74, which prevents the calculator from producing a complex result. Attempting an operation that would normally return a complex number, displays Invalid Data.

Note that flag 74 is only observed if the inputs for a function are real numbers. That is, if one or more inputs for a function are already complex, the result will be complex, regardless of the state of flag 74.

To enable complex results for real-number operations: Press ■ [MODES] [▼] CRES (*complex-result enable*). This function clears flag 74 (default).

The Display

The HP-42S uses a two-line, 22-character display while the HP-41 uses a single-line, 12-character display. Therefore, programs that specifically format output for the HP-41 display may not produce the desired displays on the HP-42S.

The HP-42S does not *scroll* the display as the HP-41 does. The calculator indicates when a number is too large for the display by showing the … (ellipsis) character. Press and hold ■ [SHOW] to see the full-precision value for the number in the X-register.

Keystrokes

For the most part, the keystroke sequences on the HP-42S are similar to the HP-41. The following exceptions are worth noting:

- Alpha characters are typed with the ALPHA menu (page 37).
- Indirect addressing on the HP-41 uses the shift (■) key. The HP-42S, on the other hand, uses [·] or [·] IND to specify indirect parameters. (Refer to "Specifying Parameters" in chapter 4.)
- In addition to separating two numbers for calculations, the [ENTER] key has a few other uses. Refer to "Other Uses of the [ENTER] Key" on page 47.
- Pressing a key during a PSE (*pause*) causes program execution to stop. Press [R/S] to restart the program.

No Packing

If you're familiar with the HP-41, then you probably have seen the "PACKING" and "TRY AGAIN" messages. Packing removes any unused gaps in program memory. The HP-42S continuously keeps memory packed, so there is no need for a PACK function, and you'll never see a "PACKING" message.

Function Names

A number of function names used by the HP-42S are different from those on the HP-41, even though the functions work identically.

When keying in an HP-41 program, you can use *either* name for the functions in the following table. The calculator automatically converts each HP-41 function name to the corresponding HP-42S function. Note that HP-41 function names do not appear in the function catalog.

HP-41 Function Name	HP-42S Function Name
CHS	+/−
DEC	→DEC
D-R	→RAD
ENTER↑	ENTER
FACT	N!
FRC	FP
HMS	→HMS
HR	→HR
INT	IP
OCT	→OCT
P-R	→REC
RDN	R↓
R-D	→DEG
R-P	→POL

HP-41 Function Name	HP-42S Function Name
ST+	STO+
ST−	STO−
ST*	STO×
ST/	STO÷
X<=0?	X≤0?
X<=Y?	X≤Y?
*	×
/	÷

Stack Registers. The HP-42S distinguishes stack registers with `ST`. For example, the HP-41 instruction `10 VIEW X` is equivalent to the HP-42S instruction `10 VIEW ST X`.*

Alpha Strings. The HP-41 displays Alpha strings in programs with the ᵀ character. The HP-42S, however, surrounds Alpha strings with quotation marks. For example, the HP-41 program line 03 ᵀHELLO is equivalent to the HP-42S instruction `03 "HELLO"`. Similarly, 04 ᵀ⊢THERE is equivalent to `04 ⊢"THERE"`. (Note that some printers may not be able to print the append character.)

Example: Keying in an HP-41 Program. The following program was taken unaltered from the *HP-41CV Owner's Manual*. The program finds the roots of the equation $ax^2 + bx + c = 0$, where a, b, and c are constants. The solutions can be found using the quadratic formula, namely:

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

* It is *not* equivalent to the instruction `10 VIEW "X"`, which displays a variable named X.

Key the program into memory:

HP-41 Program Listing:	**HP-42S Keystrokes:**
	■ [GTO] [·] [·] ■ [PRGM]
01 LBL "QUAD"	■ [PGM.FCN] LBL QUAD [ENTER]
02 "a=?"	■ [ALPHA] a=?
03 PROMPT	■ [PGM.FCN] [▼] PROM
04 2	2
05 *	[×]
06 STO 00	[STO] 00
07 "b=?"	■ [ALPHA] b=?
08 PROMPT	■ [PGM.FCN] [▼] PROM
09 CHS	[+/-]
10 STO 01	[STO] 01
11 "c=?"	■ [ALPHA] c=?
12 PROMPT	■ [PGM.FCN] [▼] PROM
13 RCL 00	[RCL] 00
14 *	[×]
15 2	2
16 *	[×]
17 RCL 01	[RCL] 01
18 X↑2	■ [x^2]
19 X<>Y	[x≷y]
20 -	[−]
21 X<0?	■ [PGM.FCN] [▼] X?0 X<0?
22 GTO 01	■ [GTO] 01
23 SQRT	[$\sqrt{x}$]
24 STO 02	[STO] 02
25 RCL 01	[RCL] 01
26 +	[+]
27 RCL 00	[RCL] 00
28 /	[÷]
29 "ROOTS="	■ [ALPHA] ROOTS=
30 ARCL X	[ARCL] [·] ST X
31 AVIEW	■ [PGM.FCN] AVIEW
32 PSE	■ [PGM.FCN] [▼] PSE
33 RCL 01	[RCL] 01
34 RCL 02	[RCL] 02

```
35 -                      [−]
36 RCL 00                 [RCL] 00
37 /                      [÷]
38 "AND "                 ■ [ALPHA] AND (space)
39 ARCL X                 [ARCL] [·] ST X
40 AVIEW                  ■ [PGM.FCN] ■ [PGM.FCN] AVIEW
41 RTN                    RTN
42 LBL 01                 LBL 01
43 "ROOTS COMPLEX"        ■ [ALPHA] ROOTS COMPLEX [ENTER]
44 AVIEW                  AVIEW
45 .END.                  [EXIT]
```

After keying in the program, exit Program-entry mode and run the program for $a = 1$, $b = 7$, and $c = 12$.

Keys	Display
[EXIT] [XEQ] QUAD	a=? x: 0.0000
1 [R/S]	b=? x: 2.0000
7 [R/S]	c=? x: -7.0000
12 [R/S]	ROOTS=-3.0000 x: -3.0000
	AND -4.0000 x: -4.0000

Enhancing HP-41 Programs

The HP-42S has a number of functions that you may want to incorporate into existing HP-41 programs. The following list can help you start thinking about enhancements for your HP-41 programs:

- Use named variables instead of storage registers to make your programs easier to understand (chapter 3).
- Take advantage of automatic labeling by using the INPUT and VIEW functions (chapter 9).
- Create CUSTOM menu key assignments that aid in executing programs or routines within programs (pages 68 and 112).
- Modify messages to take advantage of the larger display (page 129).
- Use program-controlled menus to enhance the *user interface* of a program (pages 125 and 145).

The *HP-42S Programming Examples and Techniques* manual (part number 00042-90020) uses the "QUAD" program from the preceding example to demonstrate how to enhance an HP-41 program.

Part 3

Built-In Applications

12

The Solver

The built-in Solver application (■ SOLVER) is a special root finder that enables you to solve an equation for any of its variables. In this chapter, you'll learn how to:

- Solve for an unknown.
- Find the root(s) of an equation.
- Make initial guesses to help guide the Solver to a solution.
- Interpret the results returned by the Solver.
- Use the Solver in a program.

Additional examples using the Solver are included at the end of this chapter. They include the equation of motion for free-fall and the time value of money equation.

Using the Solver

The general procedure for solving is:

1. Enter a program that defines the function to be solved.
2. Press ■ SOLVER and then select the program you want to solve.
3. For each known variable, key in a value and then store the value by pressing the corresponding menu key.
4. Calculate the unknown variable by pressing the corresponding variable menu key.

Step 1: Writing a Program for the Solver

Before you use the Solver you must write a program or subroutine that evaluates *f(x)* for the function you want to solve. When you're writing the program, keep in mind that:

- The program must begin with a global label.
- The program must define the variables that will appear in the Solver variable menu.
- The Solver may execute your program many times to find a solution. Therefore, the length and efficiency of your program may affect the amount of time required to find a solution.

How the Solver Uses Your Program. The Solver executes your program using different values for the unknown variable. During each successive evaluation, the Solver moves closer to a solution. In most cases, the Solver eventually finds a value for the unknown variable that causes your function to evaluate to zero. This value is a solution.

Generally, the Solver finds a solution. However, it may encounter mathematical conditions in which a solution cannot be found. Refer to "How the Solver Works" on page 186.

Simplifying the Function. As with many mathematical procedures, the first step to solving a problem is simplification. Here you'll have to call upon your own expertise to simplify the equation. In general, you should attempt to combine like terms and constants, reducing the equation to the form

$$f(x) = 0$$

where *f(x)* is a function of one or more variables. For example, the equation for the volume of a box is given by

$$\text{Length} \times \text{Width} \times \text{Height} = \text{Volume.}$$

Rearranging terms gives

$$\text{Length} \times \text{Width} \times \text{Height} - \text{Volume} = 0.$$

Written as a program for the Solver, the function looks like this:

`01 LBL "VOL"`	The global label identifies the program.
`02 MVAR "L"` `03 MVAR "W"` `04 MVAR "H"` `05 MVAR "V"`	These lines identify the menu variables to appear in the Solver menu.
`06 RCL "L"` `07 RCL× "W"` `08 RCL× "H"` `09 RCL- "V"` `10 END`	This is the body of the program that calculates *f(x)*. (Recalling data and recall arithmetic are covered in chapter 3.)

Defining Menu Variables. MVAR (*menu variable*) instructions define which variables appear in the Solver variable menu. These definitions must be grouped together (sequential line numbers) and must immediately follow the global label. The calculator ignores MVAR instructions that occur anywhere else in the program.

Your program may use any number of variables; however, only those defined with MVAR appear in the Solver variable menu.

The Body of the Program. The main purpose of the program is to calculate the function, *f(x)*. Key in the instructions just as if you were solving the equation from the keyboard. Recall each variable as it is needed.

Example: Keying In a Solver Program. Key the "VOL" program into your calculator.

A helpful hint: programs that use variables are easier to key in if the variables already exist. Before keying in the program, create the variables *V*, *H*, *W*, and *L* by storing a zero into each one.

Keys	Display
0 [STO] [ENTER] V [ENTER]	Y: 0.0000 X: 0.0000
[STO] [ENTER] H [ENTER]	Y: 0.0000 X: 0.0000

[STO] [ENTER] W [ENTER]

```
Y: 0.0000
X: 0.0000
```

[STO] [ENTER] L [ENTER]

```
Y: 0.0000
X: 0.0000
```

Go to a new program space, select Program-entry mode, and key in the "VOL" program listed above.

■ [GTO] [·] [·]
■ [PRGM]

```
00▶{ 0-Byte Prgm }
01 .END.
```

■ [PGM.FCN] LBL VOL [ENTER]

```
00 { 7-Byte Prgm }
01▶LBL "VOL"
```

Pressing ■ [SOLVER] in Program-entry mode displays a menu containing the MVAR function.

■ [SOLVER] MVAR L

```
02▶MVAR "L"
MVAR | | | | PSLV | SOLVE
```

MVAR W

```
03▶MVAR "W"
MVAR | | | | PSLV | SOLVE
```

MVAR H

```
04▶MVAR "H"
MVAR | | | | PSLV | SOLVE
```

MVAR V [EXIT]

```
04 MVAR "H"
05▶MVAR "V"
```

[RCL] L

```
05 MVAR "V"
06▶RCL "L"
```

[RCL] [×] W

```
06 RCL "L"
07▶RCL× "W"
```

[RCL] [×] H

```
07 RCL× "W"
08▶RCL× "H"
```

[RCL] [−] V

```
08 RCL× "H"
09▶RCL- "V"
```

Press [EXIT] to exit Program-entry mode.

Step 2: Selecting a Program To Solve

When you execute the Solver from the keyboard (■ [SOLVER]), it prompts you to select a program. All global labels that are followed by MVAR instructions are displayed in a menu. Select a program by pressing the corresponding menu key. (If there are more than six labels, use [▲] or [▼] to find the program you're looking for.)

Example. Select the "VOL" program entered in the previous example. The Solver immediately displays the variable menu for "VOL".

■ [SOLVER] VOL

```
x: 0.0000
L  W  H  V
```

Step 3: Storing the Known Variables

When you select a program to solve, the calculator searches for menu variables used by the program and displays a variable menu. Use the variable menu to store values into the known variables. Refer to page 125 for more information about using variable menus.

Example. Store these dimensions: *length* = 5 cm, *width* = 7 cm, and *height* = 12 cm. Key in each value and then press the corresponding menu key.

5 L

```
L=5.0000
L  W  H  V
```

7 W

```
W=7.0000
L  W  H  V
```

12 H

```
H=12.0000
L  W  H  V
```

Step 4: Solving for the Unknown

After storing the known values, all that remains is to press the menu key for the unknown. The Solver immediately begins searching for a solution. During this process, the Solver displays two numbers. These numbers represent the two current estimates of the solution.

Example. Solve for the volume of a box using the dimensions entered in the previous example.

Keys	Display
[V]	V=420.0000 L W H V

The volume is 420 cm^3.

Using the same length and height, what is the width of a box if the volume is 400 cm^3? Store the known volume.

Keys	Display
400 [V]	V=400.0000 L W H V

Solve for the width.

Keys	Display
[W]	W=6.6667 L W H V

Choosing Initial Guesses

By entering guesses, you can control the initial estimates used in a search for a solution. Since the search starts in the range between the two initial estimates, entering guesses can reduce the number of iterations required to find a solution. Also, if more than one solution exists, guesses can help select the solution you desire.

A useful application of providing initial guesses is finding multiple roots of an equation. For example, the expression $(x - 3)(x - 2)$ has roots at $x = 3$ and $x = 2$. The root that the Solver finds depends on the starting point for its search. Initial guesses tell the Solver where to begin.

To enter guesses for the unknown variable:

1. Key in the first guess; press the menu key for the unknown variable.
2. Key in the second guess; press the menu key again.
3. Press the menu key a third time to begin solving.

Example: Finding Multiple Roots of an Equation. A solution for a single unknown, say x, is a root if $f(x) = 0$. Consider the following equation:

$$x^3 - 5x^2 - 10x = -20.$$

Rearranging terms gives

$$x^3 - 5x^2 - 10x + 20 = 0.$$

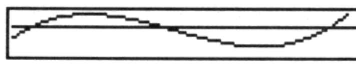
By factoring out an x, the equation is easier to write as a program.

$$x(x^2 - 5x - 10) + 20 = 0$$

Key in the following program:

`01 LBL "FNX"` `02 MVAR "X"`	The program defines a single menu variable, X.
`03 RCL "X"` `04 ENTER`	Recalls X and makes an extra copy.
`05 X↑2` `06 LASTX` `07 5` `08 ×` `09 -` `10 10` `11 -`	Calculates $(x^2 - 5x - 10)$.
`12 ×`	Calculates $x(x^2 - 5x - 10)$ using the extra copy of X made in line 04.
`13 20` `14 +` `15 END`	Completes $f(x) = x(x^2 - 5x - 10) + 20$.

If you have the "DPLOT" program in your calculator (page 156), you can plot $f(x) = x^3 - 5x^2 - 10x + 20$ in the display like this:

[XEQ] DPLOT	Ready YMIN YMAX AXIS XMIN XMAX
50 [+/-] YMIN 25 YMAX	YMAX=25.0000 YMIN YMAX AXIS XMIN XMAX
0 AXIS	AXIS=0.0000 YMIN YMAX AXIS XMIN XMAX
3 [+/-] XMIN 7 XMAX	XMAX=7.0000 YMIN YMAX AXIS XMIN XMAX
[R/S]	ABCDE FGHI JKLM NOPQ RSTUV WXYZ
FNX [R/S]	x: 3.0000 X
X	

By examining the plot, you can see that there are three roots (intersections with the x-axis). Use the Solver to find each root.

■ [SOLVER] FNX	x: 6.6667 X

Since X is the only variable declared in the program, it's the only one that appears in the Solver menu. By carefully choosing your guesses, you can zero in on each root. The graph shows that the first root is somewhere between $x = -3$ and $x = 0$. Enter the first guess.

3 [+/-] X	X=-3.0000 X

Enter the second guess and then solve for X.

0 [X] [X]

```
X=-2.4433
```

The first root is $x = -2.4433$. Now use the same procedure to find the second root, which from the graph appears to be between $x = 0$ and $x = 4$.

0 [X] 4 [X] [X]

```
X=1.3416
```

The second root is $x = 1.3416$. Calculate the third root, which appears to be between $x = 4$ and $x = 7$.

4 [X] 7 [X] [X]

```
X=6.1017
```

The third root is $x = 6.1017$.

How the Solver Works

The Solver uses an iterative (repetitive) process to search for a solution that sets the function equal to zero. The Solver starts with two initial estimates of the answer—your guesses, or numbers it generates. Using one of the estimates, the Solver evaluates your program. Then, the Solver repeats the calculation using the other estimate. If neither estimate produces a value of zero, the Solver produces two new estimates that appear to be closer to the answer. By repeating this process many times, the Solver approaches a solution.

```
6.10402112301  +
6.06268001092  -
```

During the search for a solution, the calculator displays the two current estimates for the unknown.* Next to each estimate, the calculator displays a sign (+ or −). Each sign indicates whether the function is positive or negative at that estimate.

* Estimates are not displayed when a program executes the Solver.

A question mark next to an estimate indicates the function cannot be evaluated at that estimate. Generally, this is because of some mathematical error, such as dividing by zero.

Halting and Restarting the Solver

Depending on the function you are solving, it can take several minutes to find a solution. You can halt the search by pressing [R/S] (or [EXIT]). To resume the search from where it left off, press [R/S] again.

If the estimates don't seem to be proceeding towards a number you judge to be a reasonable answer, halt the search (press [R/S]), and then enter new guesses and start over.

Interpreting the Results

There are several possible outcomes of an iterative search for a solution. The Solver returns data to the stack registers that can be used to help you interpret the results. For a more detailed description of these conditions, refer to the *HP-42S Programming Examples and Techniques* manual (part number 00042–90020).

Stack Register	Contents
T	An integer (0–4) indicating the condition that caused the Solver to stop. 0 = A solution has been found. 1 = A sign reversal has occurred. 2 = An extremum has been found. 3 = Bad guess(es) were used. 4 = The function may be a constant.
Z	The value of the function evaluated at the solution. If an actual root has been found, the Z-register contains a zero.
Y	The previous guess.
X	The solution (or the best guess if a solution was not found).

Solution Found. A solution has been found that may be a root. If you want to know if the result is an actual root, you can:

- Test the contents of the Z-register. If this number is equal to zero, then the solution is an actual root.
- Press the menu key to solve for the unknown again. If you get the same result (without a message), the solution is an actual root. If, on the other hand, you see the message `Sign Reversal`, then the result is only an approximation to a root.

Sign Reversal. A discontinuity or pole has been found. The Solver has found neighboring points for which the value of the function changes sign, but no point at which it evaluates to zero.

Extremum. The Solver has found an approximation to a local minimum or maximum of the numerical absolute value of the function. If the solution is $\pm 9.99999999999 \times 10^{499}$, it corresponds to an asymptotic extremum.

Bad Guess(es). If the Solver stops and displays `Bad Guess(es)`, one or both initial guesses lie outside of the domain of the function. That is, the function returns an error when evaluated at the values of the guesses.

Constant? If the Solver stops and displays `Constant?`, the function returns the same value at every point sampled by the Solver, suggesting that the function may be constant.

Using the Solver in a Program

To use the Solver in a program, the program must:

1. Select a program using the PGMSLV (*program to solve*) function.
2. Store the known variables.
3. Provide initial guesses for the unknown (optional). The first guess is stored into the variable. The second guess is taken from the X-register.
4. Solve for the unknown with the SOLVE function.

For example, the following program segment illustrates how the "VOL" program could be solved by another program. This program multiplies the current value of *L* by 3 and stores that value into *H*. That value is then multiplied by 3 again and stored into *V*. The program then solves for *W*.

`01 LBL "BOXSLV"`	
`02 PGMSLV "VOL"`	Selects "VOL" as the program to solve.
`03 RCL "L"` `04 3` `05 ×` `06 STO "H"` `07 3` `08 ×` `09 STO "V"`	Calculates new values for *H* and *V*.
`10 SOLVE "W"`	Solves for *W*.
`11 GTO IND ST T` ⋮	Branches to the subroutine specified by the code (0–4) in the T-register. That is, the program branches to LBL 00 if a solution is found, LBL 01 if a sign reversal occurred, LBL 02 if an extremum was found, LBL 03 if the guesses were bad, or LBL 04 if the function is a constant. (Refer to the table on page 187).

More Solver Examples

The Equation of Motion for Free-Fall

The equation of motion for a free-falling object is

$$Distance = v_0 t + \tfrac{1}{2} g t^2$$

where v_0 is the initial velocity, t is the time, and g is the acceleration due to gravity. The Solver enables you to solve for any of the variables, given values for the other variables.

Rearranging terms gives

$$0 = v_0 t + \tfrac{1}{2} g t^2 - Distance.$$

Written as a program for the Solver, the equation looks like this:

```
01 LBL "FREE"
02 MVAR "Dist"
03 MVAR "Vo"
04 MVAR "Time"
05 MVAR "g"
```

Defines the menu variables for the program.

```
06 RCL "Vo"
07 RCL "Time"
08 ×
```

Calculates $v_0 t$.

```
09 LASTX
10 X↑2
11 RCL× "g"
12 2
13 ÷
```

Calculates $\tfrac{1}{2} g t^2$.

```
14 +
```

Adds the two intermediate results: $v_0 t + \tfrac{1}{2} g t^2$.

```
15 RCL- "Dist"
16 END
```

Subtracts the distance, which completes *f(x)*.

Since the acceleration due to gravity, g, is a menu variable, you can change it to match the units of the problem you're working. It also allows you to calculate g based on experimental data.

Example. Calculate how far an object falls in 5 seconds (starting from rest). Before you begin, go to a new program space and key in the program listed above.

■ SOLVER FREE

```
x: 0.0000
DIST VO TIME G
```

The object is starting at rest, so $v_0 = 0$.

0 VO

```
Vo=0.0000
DIST VO TIME G
```

Store the appropriate acceleration constant. To get a final result in meters, use 9.8 m/s^2.

9.8 G

```
g=9.8000
DIST VO TIME G
```

Store the time (5 seconds).

5 TIME

```
Time=5.0000
DIST VO TIME G
```

Now solve for the distance.

DIST

```
Dist=122.5000
DIST VO TIME G
```

An object falls 122.5 meters in 5 seconds.

Try another calculation: how long does it take an object to fall 500 meters? Since v_0 and g are already stored, there's no need to store them again. Store the distance.

500 DIST

```
Dist=500.0000
DIST VO TIME G
```

Calculate the time.

TIME

```
Time=10.1015
DIST VO TIME G
```

It takes slightly more than 10 seconds for an object to fall 500 meters.

The Time Value of Money Equation

The time value of money equation

$$0 = PV + (1 + ip)\ PMT \left[1 - \frac{(1 + i)^{-N}}{i}\right] + FV\ (1 + i)^{-N}$$

establishes the relationships between the following variables:

N	The number of monthly payments or compounding periods.
I%YR	The annual interest rate as a fraction ($i = I\%YR \div 1200$).
PV	The present value. (This can also be an initial cash flow or a discounted value of a series of future cash flows.) *PV* always occurs at the beginning of the first month.
PMT	The monthly payment.
FV	The future value. (This can also be a final cash flow or a compounded value of a series of cash flows.) *FV* always occurs at the end of the *N*th month.

The value p indicates payment timing. If $p = 1$, then payments occur at the *beginning* of each month. If $p = 0$, then payments occur at the *end* of each month. The "TVM" program uses flag 00 to represent p. For payments at the beginning a each month, set flag 00. For payments at the end of each month, clear flag 00.

Here is how the equation can be written as a program for the Solver:

`01 LBL "TVM"` `02 MVAR "N"` `03 MVAR "I%YR"` `04 MVAR "PV"` `05 MVAR "PMT"` `06 MVAR "FV"`	Declares the menu variables.
`07 1` `08 ENTER` `09 ENTER` `10 RCL "I%YR"` `11 %` `12 12` `13 ÷` `14 STO ST T`	Calculates the monthly interest rate expressed as a decimal fraction, *i*.
`15 FC? 00` `16 CLX` `17 +`	If flag 00 is clear (End mode), calculates $(i + 0)$. If flag 00 is set (Begin mode), calculates $(i + 1)$.
`18 R↓` `19 +` `20 RCL "N"` `21 +/-` `22 Y↑X`	Calculates $(1 + i)^{-N}$.
`23 1` `24 X<>Y` `25 -`	Calculates $1 - (1 + i)^{-N}$.
`26 LASTX` `27 RCL× "FV"`	Calculates $FV\ (1 + i)^{-N}$.
`28 R↓` `29 X<>Y` `30 ÷`	Calculates $1 - \frac{(1 + i)^{-N}}{i}$.
`31 ×` `32 RCL× "PMT"` `33 +` `34 RCL+ "PV"` `35 END`	Completes the expression.

Example. Penny of Penny's Accounting wants to know what the monthly payments will be for a 3-year loan at 10.5% annual interest, compounded monthly. The amount financed is $5,750. Payments are made at the end of each period.

After keying in the program above, use the Solver to calculate the unknown information for Penny.

Keys	Display
■ [SOLVER] TVM	x: 0.0000 N I%YR PV PMT FV

Clear flag 00 and set the display format to FIX 2.

Keys	Display
■ [FLAGS] CF 00 ■ [DISP] FIX 02	x: 0.00 N I%YR PV PMT FV

Enter the known values: $PV = 5750$, $FV = 0$, $I\%YR = 10.5$, and $N = 3 \times 12$.

Keys	Display
5750 PV	PV=5,750.00 N I%YR PV PMT FV
0 FV	FV=0.00 N I%YR PV PMT FV
10.5 I%YR	I%YR=10.50 N I%YR PV PMT FV
3 [ENTER] 12 [×] N	N=36.00 N I%YR PV PMT FV

Now solve for the payment.

Keys	Display
PMT	PMT=-186.89 N I%YR PV PMT FV

The payment is negative because it is money to be *paid out*.

This is $10 higher than Penny's client can pay each month. What interest rate would reduce the monthly payments by $10? *Add* 10 to the negative payment that's already in the X-register and store the new value into *PMT*.

10 [+] PMT

```
PMT=-176.89
 N | I%YR | PV | PMT | FV |
```

Now, solve for the interest rate.

I%YR

```
I%YR=6.75
 N | I%YR | PV | PMT | FV |
```

Return to FIX 4 display mode and exit from the Solver.

■ [DISP] FIX 04 [EXIT] [EXIT]

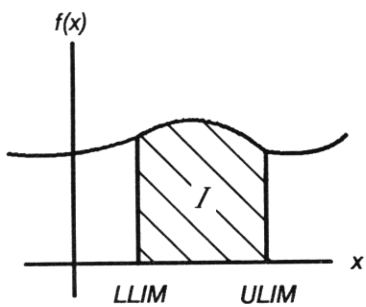
```
Y: 6.7509
X: 6.7509
```

13

Numerical Integration

Many problems in mathematics, science, and engineering require calculating the definite integral of a function. If the function is denoted by *f(x)* and the interval of integration is from the lower limit (*LLIM*) to the upper limit (*ULIM*), then the integral can be expressed mathematically as

$$I = \int_{LLIM}^{ULIM} f(x)\, dx.$$



The quantity I can be interpreted geometrically as the area of a region bounded by the graph of $f(x)$, the x-axis, and the limits $x = LLIM$ and $x = ULIM$ (provided that $f(x)$ is nonnegative throughout the interval of integration).

In this chapter, you'll learn how to use the HP-42S Integration application (■ [∫f(x)]) to calculate a definite integral.

Using Integration

The general procedure for calculating an integral is:

1. Enter a program that defines the function *f(x)* that you want to integrate.
2. Press ■ [∫f(x)] and then select the program you want to integrate.
3. For each constant used in *f(x)*, key in a value and then store the value by pressing the corresponding menu key.
4. Select a variable of integration by pressing the corresponding menu key.
5. Enter the limits of integration and an accuracy factor, and then press [∫] to calculate the integral.

Step 1: Writing a Program for Integration

Before you can calculate the definite integral of *f(x)*, you must write a program that evaluates *f(x)* given *x*. While writing the program, keep in mind:

- The program must begin with a global label.
- The program must define all variables you want to appear in the Integration variable menu.
- The Integration application may execute your program many times to find a solution. Therefore, the length and efficiency of your program affects the amount of time it takes to calculate the integral.

How an Integral Is Calculated. The HP-42S evaluates an integral by computing a weighted average of the function's values at many values of the variable of integration within the interval of integration. These are known as sample points.

The integration algorithm at first considers only a few sample points, yielding relatively inaccurate approximations. If these approximations are not yet as accurate as the accuracy of *f(x)* would permit, the algorithm is iterated (repeated) with a larger number of sample points. These iterations continue, using about twice as many sample points each time, until the resulting approximation is accurate as specified by the accuracy factor.

Depending on the number of iterations needed, it may take a few seconds to several minutes to calculate a result.

Defining Menu Variables. MVAR (*menu variable*) instructions define which variables appear in the Integration variable menu. These definitions must be grouped together (sequential line numbers) and must immediately follow the global label. The Integration application ignores MVAR instructions that occur anywhere else in the program.

Your program may use any number of variables; however, only those defined with MVAR appear in the Integration variable menu.

Example: Keying in a Program for Integration. The Bessel function of the first kind of order 0 can be expressed as

$$J_0(x) = {}^1/\pi \int_0^\pi \cos (x \sin t)\, dt.$$

Written as a program, this function looks like this:

01 LBL "BSSL" 02 MVAR "X" 03 MVAR "T"	Declares the menu variables.
04 RCL "T" 05 SIN 06 RCL× "X" 07 COS 08 END	Calculates $f(x) = \cos (x \sin t)$.

Create the variables and then key the program into the calculator.

0 [STO] [ENTER] T [ENTER] [STO] [ENTER] X [ENTER]	Y: 0.0000 X: 0.0000
■ [GTO] [·] [·] ■ [PRGM]	00▶{ 0-Byte Prgm } 01 .END.
■ [PGM.FCN] LBL BSSL [ENTER]	00 { 8-Byte Prgm } 01▶LBL "BSSL"

Keys	Display
■ [∫f(x)] MVAR X	02▶MVAR "X" MVAR PINT INTEG
MVAR T [EXIT]	02 MVAR "X" 03▶MVAR "T"
[RCL] T	03 MVAR "T" 04▶RCL "T"
[SIN]	04 RCL "T" 05▶SIN
[RCL] [×] X	05 SIN 06▶RCL× "X"
[COS]	06 RCL× "X" 07▶COS

Press [EXIT] to exit Program-entry mode.

Step 2: Selecting a Program To Integrate

When you select the Integration application (■ [∫f(x)]), it prompts you to select a program. All global labels that are followed by MVAR instructions are displayed in a menu. Select a program by pressing the corresponding menu key. (If there are more than six labels, use [▲] or [▼] to find the program you're looking for.)

Example. Select the "BSSL" program entered in the previous example. The Integration application immediately displays the variable menu for "BSSL".

Keys	Display
■ [∫f(x)] BSSL	Set Vars; Select ∫var X T

Step 3: Storing the Constants

The Integration application displays a variable menu for the function you selected. Use it to store each constant in the function:

1. Key in the constant value.
2. Press the corresponding menu key.

To view the contents of a variable without recalling it, press the shift key (■) and then hold down the corresponding menu key. The message disappears when you release the key.

Example. For the first evaluation of the Bessel integral, the constant X is 2.

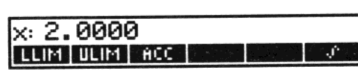
2 X

Step 4: Selecting a Variable of Integration

After storing the constants, press the menu key for the variable of integration. Do not key in a number (or alter the X-register in any way) before you press the key. If you do, the calculator assumes you are storing another constant. Press the key again. If the calculator displays a menu with the variables *LLIM*, *ULIM*, and *ACC*, then you have successfully selected a variable of integration. If you select the wrong variable, press EXIT and try again.

Example. Select *T* as the variable of integration for the Bessel function.

T



Step 5: Setting the Limits and Calculating the Integral

The menu displayed in the example above is used to store the limits of integration and an accuracy factor.

Lower Limit (*LLIM*). The *LLIM* variable specifies the left end of the x-range for the integral. To store a new value into *LLIM*, key in the value and then press **LLIM**.

Upper Limit (*ULIM*). The *ULIM* variable specifies the right end of the x-range for the integral. To store a new value into *ULIM*, key in the value and then press **ULIM**.

Accuracy Factor (*ACC*). The *ACC* variable specifies the accuracy factor to be used during the integration. The smaller the accuracy factor, the more accurate the integral calculation (which also increases execution time). To store a new value into *ACC*, key in the value and then press **ACC**.

Calculating the Integral. To calculate the integral, press **∫**. You can stop the calculation of the integral at any time by pressing [R/S] (or [EXIT]). To resume the calculation, press [R/S] again.

Example. Store the limits of integration to integrate the Bessel function from 0 to π radians.

Keys	Display
■ [MODES] **RAD**	x: 2.0000 LLIM ULIM ACC ∫
0 **LLIM**	LLIM=0.0000 LLIM ULIM ACC ∫
■ [π] **ULIM**	ULIM=3.1416 LLIM ULIM ACC ∫

Store an accuracy factor.

Keys	Display
.01 **ACC**	ACC=0.0100 LLIM ULIM ACC ∫

Now calculate the integral.

Keys	Display
∫	∫=0.7043 LLIM ULIM ACC ∫

Divide the result by π (the constant outside the integral).

■ [π] [÷]

```
x: 0.2242
LLIM ULIM ACC          ∫
```

Now change the constant, X, to 3 and calculate the integral again.

[EXIT] 3 [X]

```
X=3.0000
 X    T
```

[T] [∫]

```
∫=-0.8142
LLIM ULIM ACC          ∫
```

■ [π] [÷]

```
x: -0.2592
LLIM ULIM ACC          ∫
```

Exit from the Integration application.

[EXIT] [EXIT] [EXIT]

```
y: 0.0219
x: -0.2592
```

The value of the integral is in the X-register, and the *uncertainty of computation* (described below) is in the Y-register.

Accuracy of Integration

Since the calculator cannot compute the value of an integral exactly, it *approximates* it. The accuracy of this approximation depends on the accuracy of the integrand's function itself as calculated by your program.* This is affected by round-off error in the calculator and the accuracy of the empirical constants.

The Accuracy Factor. The accuracy factor (*ACC*) is a real number that specifies the relative error tolerance of the integration. The accuracy determines the spacing of the points, in the domain of the integration variable, at which the integrand is sampled for the approximation of the integral.

* While integrals of functions with certain characteristics such as spikes or rapid oscillations might be calculated inaccurately, these functions are rare.

The accuracy is specified as a fractional error, that is

$$ACC \geqslant \left| \frac{(true\ value\ -\ computed\ value)}{computed\ value} \right|$$

were *value* is the value of the integrand at any point in the integration interval. Even if your integrand is accurate to 12 significant digits, you may wish to use a larger accuracy factor to reduce integration time since the larger the accuracy factor, the fewer points that must be sampled.

Uncertainty of Computation. When an integral is calculated, the approximation of the integral is returned to the X-register and the *uncertainty of computation* is returned to the Y-register. That is, the integral is approximated to a value of x, $\pm\ y$.

For example, the uncertainty of computation returned in the example above is 0.0219. Dividing by π gives 0.0070. That means the approximation of the integral is -0.2592 ± 0.0070.

Using Integration in a Program

To calculate an integral in a running program, the program must:

1. Select a program using the PGMINT (*program to integrate*) function.
2. Store the constants (using [STO]).
3. Store the limits of integration and an accuracy factor.
4. Calculate the integral with the INTEG (*integrate*) function.

For example, the following program segment illustrates how these functions can be used to calculate an integral. In this example, the Bessel function is calculated again—this time using an x-value of 4.

```
          ⋮
73 PGMINT "BSSL"
```

Selects the Bessel function program to be integrated. (Refer to the example on page 198.)

```
74 CLX
75 STO "LLIM"
76 PI
77 STO "ULIM"
78 0.01
79 STO "ACC"
80 4
81 STO "X"
```

Stores the limits of integration, accuracy factor, and the constant X.

```
82 INTEG "T"
```

Calculates the integral with respect to the variable *T*. The result is returned to the X-register and the uncertainty is returned to the Y-register.

```
83 PI
84 ÷
          ⋮
```

Divides by the constant outside the integral (π).

The program could go on to interpret or display the results using the approximation of the integral in the X-register and the uncertainty in the Y-register.

14

Matrix Operations

A matrix is a rectangular array of numbers. In general, a matrix of *order* $m \times n$ has the following form:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & & & \cdot \\ \cdot & & & \cdot \\ \cdot & & & \cdot \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}.$$

In this chapter you will learn how to:

- Create and fill a matrix.
- Do matrix arithmetic and use the built-in matrix functions.
- Solve a system of simultaneous linear equations.
- Manipulate the contents of a matrix by indexing it and then using the matrix utility functions.

Matrices in the HP-42S

Matrices are one of the four data types used by the HP-42S. As such, a matrix can be manipulated in the calculator just like any other data. The first two rows of the MATRIX menu contain many of the functions needed for working with matrices.

■ MATRIX

NEW	*New matrix.*
INV	*Invert.*
DET	*Determinant.*
TRAN	*Transpose.*
SIMQ	*Simultaneous equations.*
EDIT	*Edit* matrix in X-register.
▼▲	
DOT	*Dot* product.
CROSS	*Cross* product.
UVEC	*Unit vector.*
DIM	*Dimension.*
INDEX	*Index.*
EDITN	*Edit named* matrix.

Creating and Filling a Matrix in the X-Register

To create a matrix in the X-register:

1. Key in the dimensions of the matrix: *rows* ENTER *columns*. (The maximum size of a matrix is limited only by the amount of memory available.)
2. Press ■ MATRIX NEW (*new matrix*).

To fill a matrix with data:

1. Press EDIT to activate the *Matrix Editor* on the matrix in the X-register.
2. Use ←, ↑, ↓, and → to move to the element you want to enter, and then key in the number. Repeat this step for each element of the matrix. (The Matrix Editor is explained in more detail on page 211.)
3. Press EXIT to exit the Matrix Editor and return the edited matrix to the X-register.

Example. Create the following matrix:

$$\begin{bmatrix} 7 & -5 \\ 4 & 9 \end{bmatrix}$$

2 [ENTER]	Y: 2.0000 X: 2.0000
■ [MATRIX] NEW	X: [2x2 Matrix] NEW INV DET TRAN SIMQ EDIT
EDIT	1:1=0.0000 ← OLD ↑ ↓ GOTO →

Fill the matrix rowwise. That is, start with the upper-left element and move left to right across each row.

7	1:1=7_ ← OLD ↑ ↓ GOTO →
→ 5 [+/-]	1:2=-5_ ← OLD ↑ ↓ GOTO →
→ 4	2:1=4_ ← OLD ↑ ↓ GOTO →
→ 9	2:2=9_ ← OLD ↑ ↓ GOTO →

Exit from the Editor to return the matrix to the X-register.

[EXIT]	X: [2x2 Matrix] NEW INV DET TRAN SIMQ EDIT

Pressing the ■ [SHOW] key when there is a matrix in the X-register displays the matrix descriptor and the first element.

■ [SHOW] *(hold down)*	[2x2 Matrix] 1:1=7

Store a copy of the matrix into the variable *MAT1* (refer to the note below).

[STO] [ENTER] MAT1 [ENTER]

```
x: [ 2x2 Matrix ]
NEW INV DET TRAN SIMQ EDIT
```

Exit from the MATRIX menu.

[EXIT]

```
y: 0.0000
x: [ 2x2 Matrix ]
```

Note Because matrices can be used to hold large amounts of data, it's recommended that you store copies of matrices (and other important data) into variables and then recall the data as it's needed. This saves you the trouble of keying the data in again if you inadvertently lose the matrix off the top of the stack during other calculations or while editing another matrix.

Creating and Filling a Named Matrix

A named matrix (that is, a matrix stored in a variable) can be created and filled directly in the variable. That is, you don't have to create the matrix in the stack and then store it.

To create a named matrix:

1. Key in the dimensions of the matrix: *rows* [ENTER] *columns*.
2. Press ■ [MATRIX] [▼] DIM .
3. Type the variable name for the new matrix: [ENTER] *name* [ENTER]. (If the variable already exists, the calculator redimensions it as the matrix you've specified.)

To edit a named matrix (without recalling it to the stack):

1. Press EDITN (*edit named matrix*).
2. Press a menu key to select the matrix you want to edit.

3. Use ←, ↑, ↓, and → to move to the element you want to enter, and then key in the number. Repeat this step for each element of the matrix.

4. Press [EXIT] to exit the Matrix Editor.

Example. Create a variable named *MAT2* and fill it with the following data:

$$\begin{bmatrix} -5 & 10 & 14 \\ 17 & 5 & -11 \end{bmatrix}$$

Display the second row of the MATRIX menu.

■ [MATRIX] [▼]

```
x: [ 2x2 Matrix ]
DOT CROSS UVEC DIM INDEX EDITN
```

Create the matrix.

2 [ENTER] 3 DIM [ENTER] MAT2 [ENTER]

```
x: 3.0000
DOT CROSS UVEC DIM INDEX EDITN
```

Fill *MAT2* using the Matrix Editor.

EDITN MAT2

```
1:1=0.0000
←  OLD  ↑  ↓  GOTO  →
```

5 [+/-]

```
1:1=-5_
←  OLD  ↑  ↓  GOTO  →
```

→ 10

```
1:2=10_
←  OLD  ↑  ↓  GOTO  →
```

→ 14

```
1:3=14_
←  OLD  ↑  ↓  GOTO  →
```

→ 17

```
2:1=17_
←  OLD  ↑  ↓  GOTO  →
```

→ 5

```
2:2=5_
←  OLD  ↑  ↓  GOTO  →
```

→ 11 [+/-]

```
2:3=-11_
 ←  OLD  ↑  ↓  GOTO  →
```

[EXIT]

```
x: -11.0000
 DOT CROSS UVEC DIM INDEX EDITN
```

Recall *MAT1* and *MAT2* and multiply them together.

[RCL] MAT1

```
x: [ 2x2 Matrix ]
 DOT CROSS UVEC DIM INDEX EDITN
```

[RCL] MAT2

```
x: [ 2x3 Matrix ]
 DOT CROSS UVEC DIM INDEX EDITN
```

[×]

```
x: [ 2x3 Matrix ]
 DOT CROSS UVEC DIM INDEX EDITN
```

Use the Editor to view the resulting matrix.

[▲] EDIT

```
1:1=-120.0000
 ←  OLD  ↑  ↓  GOTO  →
```

→

```
1:2=45.0000
 ←  OLD  ↑  ↓  GOTO  →
```

→

```
1:3=153.0000
 ←  OLD  ↑  ↓  GOTO  →
```

→

```
2:1=133.0000
 ←  OLD  ↑  ↓  GOTO  →
```

→

```
2:2=85.0000
 ←  OLD  ↑  ↓  GOTO  →
```

→

```
2:3=-43.0000
 ←  OLD  ↑  ↓  GOTO  →
```

Thus, *MAT1* × *MAT2* is:

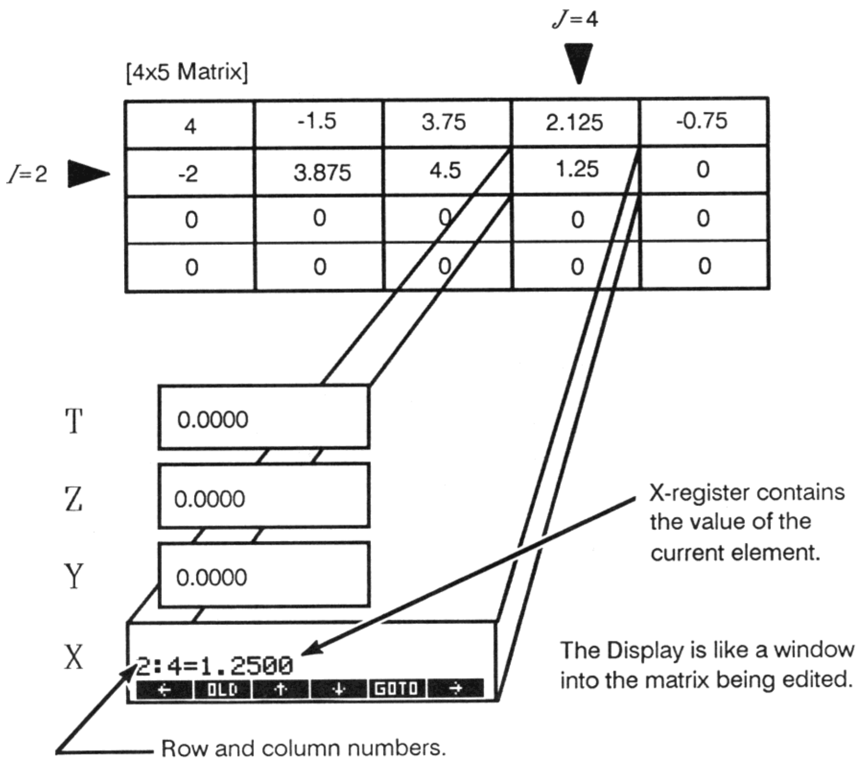
$$\begin{bmatrix} -120 & 45 & 153 \\ 133 & 85 & -43 \end{bmatrix}$$

[EXIT] [EXIT]

```
Y: -11.0000
x: [ 2x3 Matrix ]
```

The Matrix Editor

The Matrix Editor enables you to enter, view, and change any element in a matrix. When you activate the Editor, it recalls the contents of the first element into the X-register. As you move around in the matrix, the display shows the element number and its contents. To change an element, simply key in or calculate the new value.



The calculator's stack is a companion to the Matrix Editor. As you move from element to element, the stack moves right along with you, which means you can do calculations at any time.

■ MATRIX EDIT

←	*Move left.*
OLD	Restore *old* value of element.
↑	*Move up.*
↓	*Move down.*
GOTO	*Go to* element.
→	*Move right.*

▼▲

INSR	*Insert row.*
DELR	*Delete row.*
WRAP	*Wrap* mode.
GROW	*Grow* mode.

Since the Matrix Editor menu is part of the MATRIX menu (which is an application), you can select and use function menus while editing a matrix. However, if you select another application menu, the calculator automatically exits the Editor and the MATRIX menu.

How Elements Get Stored

Suppose you are editing a 5 × 5 matrix and the display shows `2:3=17.0000`. Pressing → does three things:

1. The value in the X-register (17) is stored into element 2:3 of the matrix.
2. The pointers are advanced to the next element (2:4).
3. The contents of element 2:4 are recalled to the X-register, *overwriting* the previous value (17).

This scheme allows you to use the Editor to view each element in a matrix without altering the data in the Y-, Z-, and T-registers.

The Editor allows you to recall any type of data into the X-register and do calculations. However, before you move to another element or exit from the Editor, *the X-register must contain data that can be stored into the matrix element*. A matrix cannot contain another matrix, a real matrix cannot contain a complex number, and a complex matrix cannot contain an Alpha string.

If the Matrix Editor displays `Invalid Type` when you press EXIT, the value in the X-register is not a valid element for the current matrix.

Matrices That Automatically Grow

In some circumstances you may want to create a matrix without knowing how big to make it. In *Grow* mode, the Matrix Editor allows you to keep adding rows to a matrix, regardless of its initial dimensions. Three things must happen for a matrix to automatically grow:

- Grow mode must be active. (Press GROW in the second row of the Matrix Editor menu.)
- The Editor must be positioned to the last (lower-right) element in the matrix.
- Press → to create the new row and move to the first element in that row. Each of the new elements is filled with a zero.

The example on page 241 shows how data can be entered into a matrix using Grow mode. To return to Wrap mode (default), press WRAP . The calculator automatically returns to Wrap mode when you enter or exit from the Matrix Editor.

Restoring the Old Value

Pressing OLD recalls the contents of the current element into the X-register. This is useful when you lose track of a calculation or change an element by mistake. You can also recall the current element by executing the RCLEL (*recall element*) function.

The "old" value is the number that was in the element when you first moved there. It is not replaced until you move to another element or exit from the Editor.

Inserting and Deleting Rows

While editing a matrix, you can insert and delete rows using functions in the second row of the Matrix Editor menu.

To insert a row into a matrix:

1. Move to any element in the row that will follow the new row.
2. Press INSR (*insert row*).

To delete a row of a matrix:

1. Move to any element in the row you want to delete.
2. Press DELR (*delete row*). You cannot use the DELR function if the matrix has only one row.

Complex Matrices

Before you can enter complex numbers into a matrix, the entire matrix must be made complex.

Creating Complex Matrices

To create a new complex matrix:

1. Create a real matrix using the procedure on page 206.
2. Before you enter any data into the matrix, press [ENTER] to make a copy of the matrix.
3. Press ■ [COMPLEX] to combine the two real matrices into a complex matrix. (For more information on the COMPLEX function, refer to page 91.)

Example. Create a new 3 × 4 complex matrix.

Keys	Display
3 [ENTER] 4	Y: 3.0000 X: 4_
■ [MATRIX] NEW	X: [3x4 Matrix] NEW INV DET TRAN SIMQ EDIT

ENTER ■ COMPLEX

x: [3x4 Cpx Matrix]
NEW | INV | DET | TRAN | SIMQ | EDIT

EXIT

To convert an existing matrix to complex:

1. Key in one of the following complex numbers:
   - 1 ENTER 0 ■ COMPLEX if you want the existing numbers in the matrix to become the real parts of the complex numbers.
   - 0 ENTER 1 ■ COMPLEX if you want the existing numbers in the matrix to become the imaginary parts of the complex numbers. (The calculator must be in Rectangular mode to enter this complex number.)
   - 0 ENTER ■ COMPLEX if you don't want to save any of the data in the existing matrix.
2. Multiply the matrix by the complex number.

For example, to convert *MAT1* (created in the example on page 207) to complex (saving the current data as real parts), press 1 ENTER 0 ■ COMPLEX STO × MAT1 .

Converting a Complex Matrix to Real

Pressing ■ COMPLEX converts the complex matrix in the X-register into two real matrices. The matrix containing the left-hand parts (x- or r-values) is left in the Y-register; the matrix containing the right-hand parts (y- or θ-values) is left in the X-register.

Filling a Complex Matrix

The Matrix Editor works with complex matrices just as it does for real matrices. When you're filling a matrix with data, key in complex numbers as described in chapter 6. If a number has a zero imaginary part, you can leave it off. (The calculator automatically converts the number to complex when it is stored into the matrix.)

Example. Calculate the determinant (DET) of the following complex matrix.

$$\begin{bmatrix} 10 + i16 & 4 + i9 \\ -4 & i17 \end{bmatrix}$$

Create a 2 × 2 real matrix.

Keys	Display
■ [MATRIX] 2 [ENTER] NEW	x: [2x2 Matrix] NEW | INV | DET | TRAN | SIMQ | EDIT

Make the matrix complex.

Keys	Display
[ENTER] ■ [COMPLEX]	x: [2x2 Cpx Matrix] NEW | INV | DET | TRAN | SIMQ | EDIT

Now edit the matrix. (Be sure your calculator is in Rectangular mode by pressing ■ [MODE] RECT .)

Keys	Display
EDIT 10 [ENTER] 16 ■ [COMPLEX]	1:1=10.0000 i16.0000 ← | OLD | ↑ | ↓ | GOTO | →
→ 4 [ENTER] 9 ■ [COMPLEX]	1:2=4.0000 i9.0000 ← | OLD | ↑ | ↓ | GOTO | →
→ 4 [+/-]	2:1=-4_ ← | OLD | ↑ | ↓ | GOTO | →
→ 0 [ENTER] 17 ■ [COMPLEX]	2:2=0.0000 i17.0000 ← | OLD | ↑ | ↓ | GOTO | →
[EXIT]	x: [2x2 Cpx Matrix] NEW | INV | DET | TRAN | SIMQ | EDIT
DET	x: -256.0000 i206.0000 NEW | INV | DET | TRAN | SIMQ | EDIT
[EXIT]	

Redimensioning a Matrix

To redimension a named matrix:

1. Enter the new dimensions: *rows* [ENTER] *columns*.
2. Press ■[MATRIX] [▼] DIM. The calculator displays a variable catalog of existing matrices.
3. Select a matrix by pressing the corresponding menu key or type the variable name using the ALPHA menu.

If the matrix does not exist, it is created using the dimensions and variable name that you've specified.

What Happens When a Matrix Is Redimensioned. Matrices are stored internally as a single sequence of elements. The elements fill the matrix *rowwise.*

When you redimension a matrix, the rowwise order of the elements is not changed. If you increase the size of the matrix, new elements are added at the end of the sequence. Similarly, if you reduce the number of elements in a matrix, the last elements (and the data stored in those elements) are lost.

[2×5 Matrix] redimensioned to [4×3 Matrix]

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \end{bmatrix} \qquad \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 0 & 0 \end{bmatrix}$$

You can recall the current dimensions of a matrix by recalling it into the X-register and then executing the DIM? (*dimensions?*) function. DIM? returns the number of rows to the Y-register, number of columns to the X-register, and saves a copy of the matrix in the LAST X register.

Matrix Arithmetic

Calculating with matrices is like calculating with numbers. You can manipulate a matrix on the stack using the same techniques you've already learned for working with numbers (chapter 2).

Scalar Arithmetic. Scalar arithmetic is defined as a matrix and a single number (the scalar) being combined with an arithmetic operation ([+], [−], [×], or [÷]). The arithmetic takes place on each element in the matrix.

Example: Scalar Arithmetic in the Stack. Recall the matrix *MAT1* (created in the first example in this chapter) and multiply it by 3.5. (Every element in *MAT1* is multiplied by 3.5.)

[RCL] MAT1	Y: -256.0000 i206.0000 X: [2x2 Matrix]
3.5	Y: [2x2 Matrix] X: 3.5_
[×]	Y: -256.0000 i206.0000 X: [2x2 Matrix]

Example: Scalar Arithmetic Combined With Variable Arithmetic. You can also use storage arithmetic to perform scalar arithmetic on a named matrix. Subtract 3 from each element in the matrix *MAT2*.

3 [STO] [−] MAT2	Y: [2x2 Matrix] X: 3.0000

Matrix Arithmetic Using One-Number Functions. Nearly all the one-number functions work on a matrix. For example, if you press ■ [x^2] when there is a matrix in the X-register, each element in the matrix is squared. To make a matrix negative (change the sign of each element), press [+/−].

Matrix Arithmetic Using Two-Number Functions. You can add, subtract, multiply, and divide matrices using [+], [−], [×], and [÷]. If either matrix is complex, the result is also complex.

Function	Inputs	Result
Addition ([+]) or Subtraction ([−])	Y: [m×n Matrix] X: [m×n Matrix]	X: [m×n Matrix]
Multiplication ([×])	Y: [m×n Matrix] X: [n×p Matrix]	X: [m×p Matrix]
Division ([÷])*	Y: [m×n Matrix] X: [m×m Matrix]	X: [m×n Matrix]

* Matrix division is defined as multiplying the numerator by the inverse of the denominator. Therefore, the X-register must contain a nonsingular (invertible) matrix.

Matrix Functions

Inverting a Matrix. Execute the INVRT function (■ [MATRIX] INV) to calculate the inverse of a square ($n \times n$) matrix in the X-register. A matrix multiplied by its inverse produces the *identity matrix* (a square matrix with 1's on the diagonal and 0's elsewhere).

Transposing a Matrix. Execute the TRANS function (■ [MATRIX] TRAN) to transpose a matrix in the X-register. The *transpose* of a matrix is obtained by *flipping* the matrix so that the rows become columns and the columns become rows.

Determinant. Execute the DET function (■ [MATRIX] DET) to calculate the determinant of a square matrix in the X-register.

Frobenius Norm. Execute the FNRM (*Frobenius norm*) function to calculate the Frobenius (Euclidean) norm of a matrix in the X-register. The Frobenius norm is defined as the square root of the sum of the squares of the absolute values of all of the elements.

Row Norm. Execute the RNRM (*row norm*) function to calculate the row norm (infinity norm) of a matrix in the X-register. The row norm is the maximum value (over all rows) of the sums of the absolute values of all elements in a row. For a vector, the row norm is the largest absolute value of any of the elements.

Row Sum. Execute the RSUM (*row sum*) function to calculate the sum of each row of a matrix in the X-register. RSUM returns an $m \times 1$ matrix filled with the row sums of the $m \times n$ input matrix.

Vector Operations

A single-row or single-column matrix is called a *vector*. The HP-42S performs the following vector operations.

Dot Product. Execute the DOT function (■ [MATRIX] [▼] DOT) to calculate the dot product of the matrices in the X- and Y-registers. The dot product is defined as the sum of the products of the corresponding elements in two matrices.

Cross Product. Execute the CROSS function (■ [MATRIX] [▼] CROSS) to calculate the cross product of the vectors in the X- and Y-registers. The two vectors must be two- or three-element matrices or complex numbers.

Unit Vector. Execute the UVEC function (■ [MATRIX] [▼] UVEC) to calculate the unit vector of the matrix in the X-register. That is, each element in the vector is adjusted so that the magnitude (Frobenius norm) is equal to 1.

Simultaneous Linear Equations

A system of linear equations

$$a_{11}x_1 + a_{12}x_2 = b_1$$

$$a_{21}x_1 + a_{22}x_2 = b_2$$

can be represented by the matrix equation $AX = B$, where

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, \quad X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad B = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}.$$

A is the *coefficient matrix*, *B* is the *constant* or *column matrix*, and *X* is the *solution matrix*.

To solve a system of simultaneous equations:

1. Specify the number of unknowns: Press ■ [MATRIX] **SIMQ** *nn*. The calculator automatically creates (if necessary) and dimensions three matrices: *MATA*, *MATB*, and *MATX*.
2. Enter the coefficient matrix: press **MATA**.
3. Enter the constant matrix: press **MATB**.
4. Calculate the solution matrix: press **MATX**. (For a large system of equations, this calculation can take several seconds to complete.)

To work another problem with the same number of unknowns, go to step 2 or 3. For a problem with a different number of unknowns, press [EXIT] and start over with step 1.

Example. Find the three unknowns in this system of simultaneous equations:

$$\begin{aligned} 7x + 2y - z &= 15 \\ x - y + 15z &= 112 \\ -9x + 2z &= -22 \end{aligned}$$

The coefficient matrix is:

$$\begin{bmatrix} 7 & 2 & -1 \\ 1 & -1 & 15 \\ -9 & 0 & 2 \end{bmatrix}$$

Create the appropriate matrices for three equations and three unknowns.

■ [MATRIX] **SIMQ** 03

```
x: 3.0000
MATA MATB MATX
```

Fill in the coefficient matrix.

Keys	Display
MATA	1:1=0.0000 ← OLD ↑ ↓ GOTO →
7 → 2 → 1 [+/-]	1:3=-1_ ← OLD ↑ ↓ GOTO →
→ 1 → 1 [+/-] → 15	2:3=15_ ← OLD ↑ ↓ GOTO →
→ 9 [+/-] → → 2	3:3=2_ ← OLD ↑ ↓ GOTO →
[EXIT]	x: 2.0000 MATA MATB MATX

Fill the constant matrix.

Keys	Display
MATB	1:1=0.0000 ← OLD ↑ ↓ GOTO →
15 ↓ 112 ↓ 22 [+/-]	3:1=-22_ ← OLD ↑ ↓ GOTO →
[EXIT]	x: -22.0000 MATA MATB MATX

Calculate and view the solution matrix.

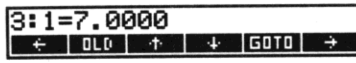
Keys	Display
MATX	1:1=4.0000 ← OLD ↑ ↓ GOTO →

The first unknown, *x*, is 4.

Keys	Display
↓	2:1=-3.0000 ← OLD ↑ ↓ GOTO →

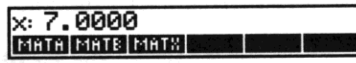
The second unknown, *y*, is −3.

↓



And the third unknown, *z*, is 7.

[EXIT]



Matrix Utility Functions (Indexing)

The functions in this section work on the currently *indexed* matrix. By indexing a matrix you can directly access and manipulate any element in a named matrix.

To index a matrix:

1. Press ■ [MATRIX] [▼] INDEX.
2. Specify a named matrix by pressing the corresponding menu key or typing the variable name with the ALPHA menu.

You can also index a matrix by editing it. After exiting the Matrix Editor, the matrix is no longer indexed.

Controlling the Index Pointers

Indexing a matrix establishes row and column pointers (*I* and *J*). These are the same pointers used by the Matrix Editor to identify the current element. When you index a matrix, the pointers are set to the first element. That is, $I = 1$ and $J = 1$. (Note that any operation that changes the dimensions of the indexed matrix also returns the index pointers to element 1:1.)

You can increment or decrement either pointer using the first four functions in the following table. If you attempt to move a pointer past the edge of a matrix (that is, outside its dimensions), the pointers automatically wrap around to the first element in the next column or row (or last element in the previous column or row).

To set the index pointers to a particular element, enter the pointer values into the X- and Y-registers (column and row, respectively), and then execute the STOIJ (*store IJ*) function.

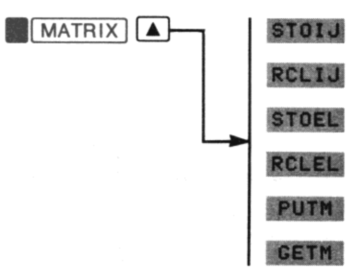
To recall the current pointer values to the X- and Y-registers, execute the RCLIJ (*recall IJ*) function.

Functions for Controlling the Index Pointers

Function	Description
I+	Increments the row pointer by one (down).*
I−	Decrements the row pointer by one (up).*
J+	Increments the column pointer by one (right).* If the calculator is in Grow mode, and the pointers are at the last element in the matrix, executing J+ creates a new row at the end of the matrix.
J−	Decrements the column pointer by one (left).*
STOIJ	Sets the index pointers to the numbers specified in the X- and Y-registers (x = column number; y = row number).
RCLIJ	Recalls the current values of the index pointers to the X- and Y-registers (x = column number; y = row number). If the pointers are both equal to zero, then there is currently no indexed matrix.

* Flags 76 and 77 are updated accordingly, indicating if a wrap has occurred. Refer to appendix C.

The third row of the MATRIX menu contains six of the most often used indexing functions.



Storing and Recalling Matrix Elements

The STOEL (*store element*) and RCLEL (*recall element*) functions are used to store and recall values in the indexed matrix. These functions do not alter the index pointers.

Function	Description
STOEL	Stores a copy of the value in the X-register into the indexed matrix at the current element, a_{ij}.
RCLEL	Recalls a copy of the current element, a_{ij}, into the X-register.

Programmable Matrix Editor Functions

Functions in the Matrix Editor menu (except GOTO) are programmable and work on the indexed matrix just as they do while using the Editor manually. For example, if you execute ← (*move left*), ↑ (*move up*), ↓ (*move down*), or → (*move right*), then:

1. The value in the X-register is stored into the indexed matrix at the current element.
2. The row and column pointers (*I* and *J*) are advanced to the next element—left, up, down, or right. (If the calculator is in Grow mode and the function is →, the matrix is enlarged by one complete row and the pointers are advanced to the first element in the new row.)
3. The value stored in the current element is recalled to the X-register, overwriting the previous value in the X-register.

The INSR, DELR, WRAP, and GROW functions (in the second row of the Editor menu) are also programmable. Refer to pages 212 through 214.

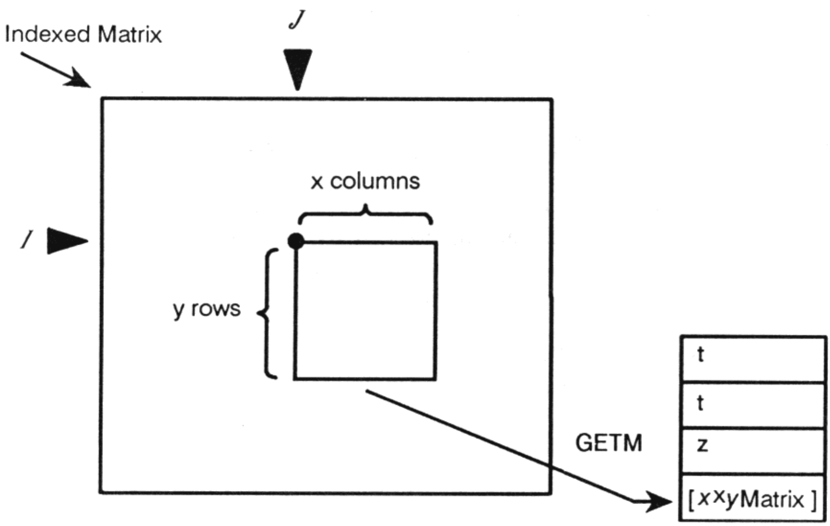
Swapping Rows

The R<>R (*row swap row*) function swaps the contents of two rows in the currently indexed matrix. Key the two row numbers into the X- and Y-registers, and then execute R<>R.

Submatrices

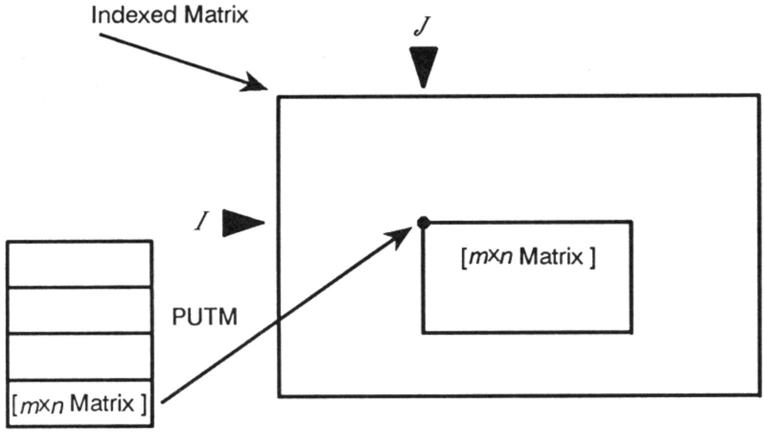
To get a submatrix from the indexed matrix:

1. Move the index pointers to the first element of the submatrix.
2. Enter the dimensions of the submatrix: number of rows in the Y-register and number of columns in the X-register.
3. Execute the GETM (*get matrix*) function (■ MATRIX ▲ GETM). GETM recalls the submatrix to the X-register.



To put a submatrix into the indexed matrix:

1. Move the index pointers to the element where you want the first element of the submatrix to go.
2. Execute the PUTM (*put matrix*) function (■ MATRIX ▲ PUTM). PUTM copies the matrix in the X-register, element for element, into the indexed matrix beginning at the current element.



Special Matrices in the HP-42S

In addition to the matrix variables that you create, there are several that are automatically created.

The Storage Registers *(REGS)*

The data storage registers are actually a special $m \times n$ matrix in calculator memory (where SIZE $= m \times n$). The name *REGS* is reserved for the storage registers matrix (and can only be used to store a matrix).

Matrices for Simultaneous Equations

The matrices *MATA*, *MATB*, and *MATX* are created (and redimensioned if necessary) whenever you execute SIMQ . The data stored in these three matrices remains until you work another problem or clear the variables.

15

Statistics

In this chapter you will learn how to:

- Enter statistical data into the HP-42S.
- Calculate statistical results based on accumulated data.
- Use statistical data stored in a matrix.
- Fit a curve to the data you've entered using one of four models.
- Forecast future values based on a curve fitted to your data.

Entering Statistical Data

Statistical data is saved with the [Σ+] (*summation plus*) key, which accumulates data into a block of storage registers containing *summation coefficients*. Executing [Σ+] adds *two* values to the statistical data: an x-value (from the X-register) and a y-value (from the Y-register). The number of accumulated data points, n, is returned to the X-register.

Clearing Statistical Data. Before you begin accumulating a new set of data, press ■[CLEAR] CLΣ (*clear statistics*) to clear data in the summation registers.

Two-Variable Statistics. To enter two-variable statistical data (x- and y-values):

1. Key in the y-value, and then press [ENTER].
2. Key in the x-value.
3. Press [Σ+].

Repeat these steps for each data pair in the data set.

One-Variable Statistics. To enter one-variable statistical data (that is, x-values only), first key in a 0 for the y-value (0 [ENTER]) and then, for *each* data point:

1. Key in an x-value.
2. Press [Σ+].

Uniformly Spaced Single-Variable Statistics. For some applications, you may want the accumulated y-values to be uniformly spaced integers. This allows you to use one-variable statistical data to do curve fitting and forecasting using the linear and logarithmic curve-fitting models. (The exponential and power models are invalid because the first y-value is zero.)

For the *first* x-value, press 0 [ENTER] *x-value* [Σ+]. For each subsequent x-value:

1. Press [ENTER] to lift n into the Y-register.
2. Key in the x-value.
3. Press [Σ+].

Example: Using Statistics. Below is a chart of maximum and minimum monthly winter (October–March) rainfall values in Corvallis, Oregon. Accumulate the values into the statistics registers.

	Oct.	Nov.	Dec.	Jan.	Feb.	Mar.
Y Maximum (inches of rain)	9.70	18.28	14.47	15.51	15.23	11.70
X Minimum (inches of rain)	0.10	0.22	2.33	1.99	0.12	0.43

Start by clearing any statistical data that may have been previously stored in the summation registers.

■ [CLEAR] CLΣ

```
Y: 0.0000
X: 0.0000
```

Enter the first data pair (remember, *y*-value first).

9.7 [ENTER] .1 [Σ+]

```
Y: 9.7000
X: 1.0000
```

Notice that the number in the Y-register did not move when you pressed [Σ+]. The *x*-value (0.10) was saved in LAST X and replaced in the X-register with *n*, the number of accumulations made so far (1).

Enter the remaining data.

18.28 [ENTER] .22 [Σ+]

```
Y: 18.2800
X: 2.0000
```

14.47 [ENTER] 2.33 [Σ+]

```
Y: 14.4700
X: 3.0000
```

15.51 [ENTER] 1.99 [Σ+]

```
Y: 15.5100
X: 4.0000
```

15.23 [ENTER] .12 [Σ+]

```
Y: 15.2300
X: 5.0000
```

11.7 [ENTER] .43 [Σ+]

```
Y: 11.7000
X: 6.0000
```

Now, calculate the average monthly minimum and maximum rainfall.

■ [STAT] MEAN

```
X: 0.8650
 Σ+ | SUM | MEAN | WMN | SDEV | CFIT
```

The average minimum monthly rainfall is 0.865 inches (average of the *x*-values).

[x≷y]

```
X: 14.1483
 Σ+ | SUM | MEAN | WMN | SDEV | CFIT
```

The average maximum monthly rainfall is 14.1483 inches (average of the *y*-values).

Statistical Functions

The SUM, MEAN, WMEAN, and SDEV functions (in the STAT menu) allow you to calculate results from statistical data you've entered.

■ [STAT]

Σ+	*Summation plus* (same as [Σ+]).
SUM	*Sum.*
MEAN	*Mean* (arithmetic average).
WMN	*Weighted mean.*
SDEV	*Standard deviation.*
CFIT	Displays the curve-fitting submenu.

▼▲

ALLΣ	*AllΣ (all statistics)* mode.
LINΣ	*Linear statistics* mode.
ΣREG	*Summation registers.*
ΣRG?	*Summation registers location.*

Sums

The SUM function returns the sum of accumulated x- and y-values into the X- and Y-registers, respectively.

Mean

As shown in the rainfall example above, the MEAN function returns the arithmetic mean (average) of the x- and y-values that have been stored with [Σ+]. The mean of x is returned to the X-register and the mean of y is returned to the Y-register.

Weighted Mean

The WMEAN function (■ [STAT] WMN) calculates the mean of x-values weighted by the y-values ($\Sigma xy \div \Sigma y$).

Standard Deviation

The SDEV (*standard deviation*) function computes the sample standard deviations,* s_x and s_y, of the data stored with [Σ+] and places them in the X- and Y-registers, respectively.

Example: Calculating Standard Deviations. If you worked the rainfall example earlier, calculate the standard deviations about the means. (If the STAT menu is not displayed, press ■[STAT].)

SDEV	x: 1.0156 Σ+ SUM MEAN WMN SDEV CFIT
[x≷y]	x: 3.0325 Σ+ SUM MEAN WMN SDEV CFIT
[EXIT]	

The standard deviations are $s_x = 1.0156$ and $s_y = 3.0325$.

Correcting Mistakes

If you discover that you have entered and accumulated incorrect data points, you can correct the mistake by using ■[Σ−] (*summation minus*).

Errors are corrected by reentering *both* the *x*- and *y*-values, pressing ■[Σ−], and then entering the correct data. Even if only one value of an (x,y) data pair is incorrect, you must delete and reenter *both* values.

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

1. Reenter the *incorrect* data pair into the X- and Y-registers.

* The SDEV function calculates the *sample standard deviation*, which assumes the data is a sampling of a larger, complete set of data. If your data constitutes the entire population of data, the *true population standard deviation* can be computed by calculating the mean of the original data, adding the mean to the statistical data using [Σ+], and then executing SDEV.

2. Press ■ [Σ−]. This function acts similarly to [Σ+] except that the results are subtracted from (rather than added to) the summation coefficients. The number of data pairs, n, is decremented by one.

3. Enter the correct data values: *y-value* [ENTER] *x-value*.

4. Press [Σ+].

The Summation Registers

The calculator uses a block of storage registers to save the summation coefficients. The current statistical mode determines how many coefficients are saved.

AllΣ mode	R_{11}	Σx	Linear mode*
AllΣ mode	R_{12}	Σx^2	Linear mode*
AllΣ mode	R_{13}	Σy	Linear mode*
AllΣ mode	R_{14}	Σy^2	Linear mode*
AllΣ mode	R_{15}	Σxy	Linear mode*
AllΣ mode	R_{16}	n	Linear mode*
AllΣ mode	R_{17}	$\Sigma \ln x$	
AllΣ mode	R_{18}	$\Sigma (\ln x)^2$	
AllΣ mode	R_{19}	$\Sigma \ln y$	
AllΣ mode	R_{20}	$\Sigma (\ln y)^2$	
AllΣ mode	R_{21}	$\Sigma \ln x \ln y$	
AllΣ mode	R_{22}	$\Sigma x \ln y$	
AllΣ mode	R_{23}	$\Sigma y \ln x$	

* These are the same six coefficients used for statistics on the HP-41 family of calculators. Before running an HP-41 program that uses statistical functions, you may need to select Linear mode to assure proper execution of the program.

To select AllΣ mode: Press ■ STAT ▼ ALLΣ. In AllΣ mode (the default), the calculator saves 13 summation coefficients. This allows you to do curve fitting and forecasting using four curve models (explained later in this chapter).

To select Linear mode: Press ■ STAT ▼ LINΣ. In Linear mode, the calculator saves only six summation coefficients. This is the minimum set of values needed to do curve fitting and forecasting using the linear model (linear regression).

Changing the Location of the Summation Registers. By default, the first summation register is R_{11}. However, you can change the location of the summation registers with the ΣREG (*summation registers*) function. Press ■ STAT ▼ ΣREG *nn*; where *nn* is the number of the first register.

For example, to relocate the statistical registers to R_{07}, press ■ STAT ▼ ΣREG 07.

Note The ΣREG function does not move any data; it only identifies which registers are used to accumulate the summation coefficients. If you want to move the location of the summation registers, do it *before* entering any data.

The ΣREG? (*summation registers location*) function returns the register number of the first summation register. To execute the ΣREG? function, press ■ STAT ▼ ΣRG?.

Nonexistent Summation Registers. After setting the number of summation registers (6 or 13), it is possible to reduce the SIZE such that one or more of the summation registers no longer exists. Statistical functions that directly access the summation registers will not operate unless *all* of the summation registers exist.

Example: Viewing the Summation Coefficients. George, owner of the Fish On Cafe, kept a record of Winter Steelhead fish caught by his customers in the nearby Wilson River. Enter the data below and then edit the storage registers to view the summation coefficients.

Weight of Fish (lbs.) (x-values)	Quantity (y-values)
6	8
7	12
8	24
9	23
10	15
11	9

Select AllΣ mode and then clear the summation registers.

■ [STAT] [▼] **ALLΣ** [EXIT]

■ [CLEAR] **CLΣ**

Keys	Display
8 [ENTER] 6 [Σ+]	Y: 8.0000 X: 1.0000
12 [ENTER] 7 [Σ+]	Y: 12.0000 X: 2.0000
24 [ENTER] 8 [Σ+]	Y: 24.0000 X: 3.0000
23 [ENTER] 9 [Σ+]	Y: 23.0000 X: 4.0000
15 [ENTER] 10 [Σ+]	Y: 15.0000 X: 5.0000
9 [ENTER] 11 [Σ+]	Y: 9.0000 X: 6.0000

Unless you've changed the location of the summation registers, the first coefficient (Σx) is stored in R_{11}. Use the Matrix Editor to view the *REGS* matrix.

Keys	Display
■ [MATRIX] [▼] **EDITN** **REGS**	1:1=0.0000 ← OLD ↑ ↓ GOTO →

Go to element 12:1 (which is R_{11}).

GOTO 12 [ENTER] 1 [ENTER]	12:1=51.0000 ← OLD ↑ ↓ GOTO →

This is the Σx coefficient. Using →, move through the registers and view all 13 coefficients. Compare them with the data above and the illustration on page 233.

→	13:1=451.0000
→	14:1=91.0000
→	15:1=1,619.0000
→	16:1=780.0000
→	17:1=6.0000
→	18:1=12.7148
→	19:1=27.2006
→	20:1=15.7832
→	21:1=42.5915
→	22:1=33.5635

→

```
23:1=134.7648
 ←  OLD  ↑  ↓  GOTO  →
```

→

```
24:1=194.2476
 ←  OLD  ↑  ↓  GOTO  →
```

[EXIT] [EXIT]

```
Y: 15.0000
X: 9.0000
```

Now, if you have a printer, print the summation coefficients using the PRΣ (*print statistics*) function. (If necessary, press ■ [PRINT] [▲] **PON** to enable printing.)

■ [PRINT] **PRΣ**

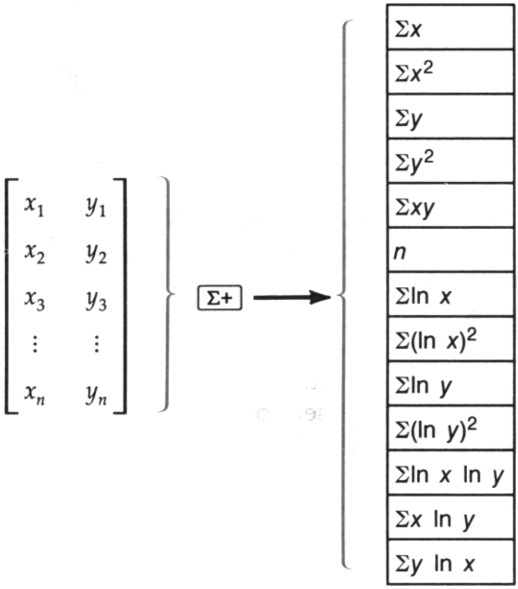
Limitations on Data Values

The calculator might be unable to perform some statistical calculations if your data values differ by a relatively small amount. To avoid this, you should normalize your data by entering the values as the difference from one value (such as the mean). This difference must then be added back to any calculations of the mean. For example, if your x-values were 776999, 777000, and 777001, you should enter the data as -1, 0, and 1; then add 777000 back to the relevant results.

If the [Σ+] function causes the contents of a register to exceed $\pm 9.99999999999 \times 10^{499}$, *there is no overflow error;* the overflowed register contains $\pm 9.99999999999 \times 10^{499}$.

Using Statistical Data Stored in a Matrix

You can enter statistical data into an $n \times 2$ matrix and then accumulate all of the data by pressing [Σ+] with the matrix in the X-register. The first column of the matrix contains the x-values, and the second column contains the y-values.



To use a matrix for statistical calculations:

1. Create a 1 × 2 named matrix. (Example: 1 ENTER 2 ■ MATRIX ▼ DIM ENTER ΣLIST ENTER.)
2. Activate the Matrix Editor. (Example: EDITN ΣLIST.)
3. Use Grow mode (▼ GROW ▲) so the matrix will grow as you enter each data pair.
4. Enter the first data pair into the matrix: *x-value* → *y-value*.
5. For each additional data pair:
   - **a.** Press → to grow the matrix by one row.
   - **b.** Enter the data pair: *x-value* → *y-value*.
6. Press ■ CLEAR CLΣ to clear the summation registers.
7. Place the matrix in the X-register. (Example: RCL ΣLIST.)
8. Press Σ+ to accumulate the data. The number of data pairs, *n*, is returned to the X-register and a copy of the matrix is saved in the LAST X register.

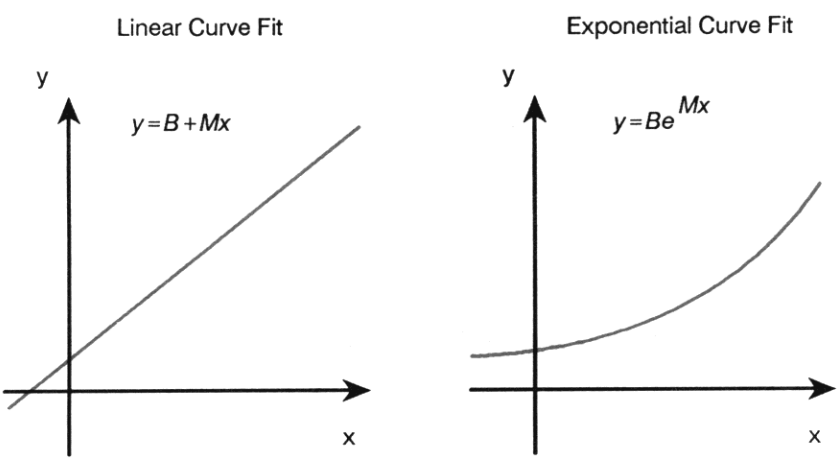
After the data is accumulated to the summation registers, you can work with it using any of the statistical functions.

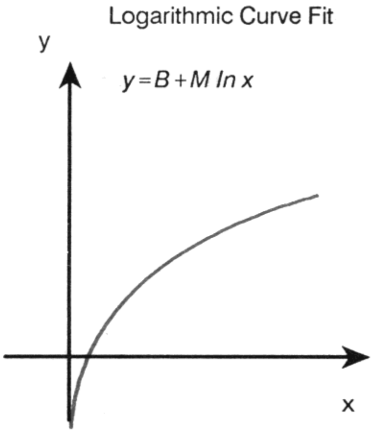
The example in the next section uses data stored in a matrix. The *HP-42S Programming Examples and Techniques* manual (part number 00042–90020) contains a utility program that makes data entry into a statistical matrix even easier.

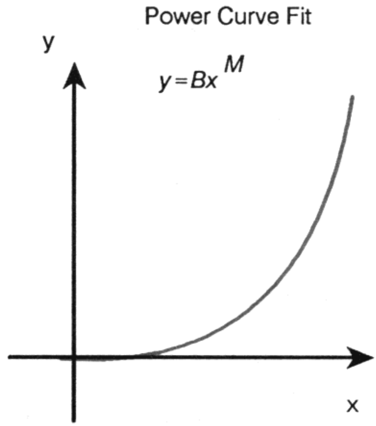
Curve Fitting and Forecasting

Curve fitting is a technique for finding a mathematical relationship between two variables, x and y. Based on this relationship, you can *forecast* a new value of y based on a given x-value, or a new value of x based on a given y-value.

To establish the relationships between the x- and y-values, you can select one of the four curve-fitting *models*:







To do curve fitting and forecasting:

1. If necessary, press ■ STAT ▼ ALLΣ to select AllΣ mode (enabling the use of all four curve-fitting models).
2. Accumulate the statistical data into the summation registers using Σ+ or Σ+ .
3. Select a curve-fitting model: ■ STAT CFIT MODL, and then LINF, LOGF, EXPF, or PWRF. (The menu label for the currently selected model is marked with a white box.)

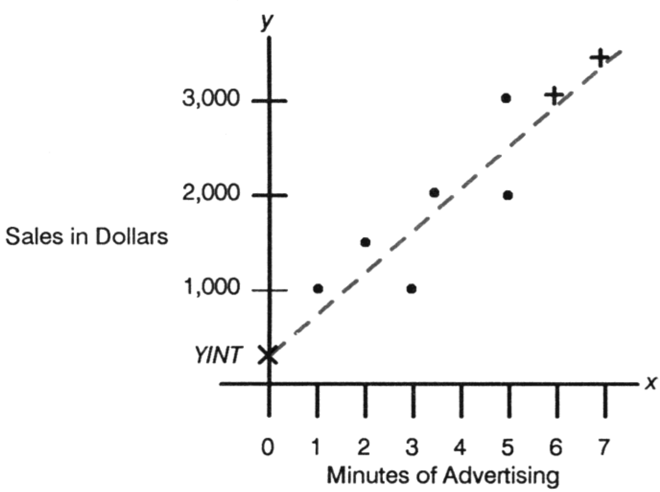
   Or, press BEST to have the calculator select a model for you. The BEST function examines the statistical data and selects the model that returns the highest correlaton coefficient.

   Press EXIT to return to the CFIT submenu.
4. Execute the function(s) you want:
   - FCSTX (*forecast x*). Key in a *y*-value and then press FCSTX.
   - FCSTY (*forecast y*). Key in an *x*-value and then press FCSTY.
   - SLOPE. Calculates the slope of the linear transformation for the current model.
   - YINT (*y-intercept*). Calculates the *y*-intercept of the linear transformation for the current model.
   - CORR (*correlation coefficient*). Calculates a coefficient ($-1 \leqslant x \leqslant 1$) that indicates how closely the accumulated data matches the current curve-fitting model.

Example: Forecasting. Smith's Moss Garden advertises on a local radio station. For the past six weeks, the manager has kept records of the number of minutes of advertising purchased and the sales per week.

	Number of Minutes of Radio Advertising (x-values)	Dollar Sales (y-values)
Week 1	2	$1,400
Week 2	1	$ 920
Week 3	3	$1,100
Week 4	5	$2,265
Week 5	5	$2,890
Week 6	4	$2,200

Smith's wants to determine whether there is a linear relationship between the amount of radio advertising and the weekly sales. If a strong relationship exists, Smith's wants to use the relationship to forecast sales. A graph of the data looks like this:



Set the display format to FIX 2 (for dollars and cents).

■ [DISP] FIX 02

```
Y: 15.00
X: 9.00
```

Enter the data from the table above into a matrix named *ΣLIST*. Start with a 1 × 2 matrix.

1 [ENTER] 2 ■ [MATRIX] [▼] DIM [ENTER] ΣLIST [ENTER]*

```
X: 2.00
DOT CROSS UVEC DIM INDEX EDITN
```

Activate the Editor on *ΣLIST* and select Grow mode so the matrix will grow to the necessary size as you enter the data.

EDITN ΣLIST [▼] GROW [▲]

```
1:1=0.00
← OLD ↑ ↓ GOTO →
```

Enter the data.

2 → 1400

```
1:2=1,400_
← OLD ↑ ↓ GOTO →
```

→ 1 → 920

```
2:2=920_
← OLD ↑ ↓ GOTO →
```

→ 3 → 1100

```
3:2=1,100_
← OLD ↑ ↓ GOTO →
```

→ 5 → 2265

```
4:2=2,265_
← OLD ↑ ↓ GOTO →
```

→ 5 → 2890

```
5:2=2,890_
← OLD ↑ ↓ GOTO →
```

→ 4 → 2200

```
6:2=2,200_
← OLD ↑ ↓ GOTO →
```

* To type ΣLIST, press [▼] MATH Σ [▲] JKLM L FGHI I RSTUV S RSTUV T .

Be sure the calculator is in AllΣ mode and then clear the summation registers.

■ [STAT] [▼] ALLΣ ■ [CLEAR] CLΣ

```
x: 2,200.00
ALL■ LINΣ        ΣREG ΣRG?
```

Accumulate the statistical data in *ΣLIST*.

[RCL] ΣLIST [▲] Σ+

```
x: 6.00
Σ+ SUM MEAN WMN SDEV CFIT
```

Select the linear curve-fitting model.

CFIT MODL LINF [EXIT]

```
x: 6.00
FCSTX FCSTY SLOPE YINT CORR MODL
```

Calculate the correlation coefficient. This number indicates how well the data conforms to the linear model.

CORR

```
x: 0.90
FCSTX FCSTY SLOPE YINT CORR MODL
```

This correlation coefficient is acceptable to Smith's. Using the linear model, estimate what the level of sales would be if the business purchased 7 minutes of advertising time per week. (That is, enter an x-value of 7 and forecast a y-value.)

7 FCSTY

```
x: 3,357.38
FCSTX FCSTY SLOPE YINT CORR MODL
```

How many minutes of advertising should Smith's buy if it wants to attain sales of $3,000? (Enter a y-value and forecast an x-value.)

3000 FCSTX

```
x: 6.16
FCSTX FCSTY SLOPE YINT CORR MODL
```

The business should buy about 6 minutes of advertising per week to increase sales to $3000.

How Curve Fitting Works

The exponential, logarithmic, and power models are calculated using transformations that allow the data to be fitted by standard linear regression. The equations for these transformations appear in the table below. The logarithmic model requires positive x-values, the exponential model requires positive y-values, and the power curve requires positive x- and y-values.

Transformation Equations

Model	Transformation
Logarithmic	$y = b + m \ln x$
Exponential	$\ln y = \ln b + mx$
Power	$\ln y = \ln b + m \ln x$

16

Base Operations

The HP-42S is capable of displaying numbers in four different bases: hexadecimal, decimal, octal, and binary. In this chapter you will learn how to:

- Select and use different number bases.
- Perform integer (base) arithmetic and use the logic functions.
- Use the programmable functions for selecting number bases.

Base Conversions

The BASE menu makes it easy to enter and display numbers in any of the four base modes.

■ [BASE] →

A…F	Selects Hexadecimal mode and displays A – F keys.
HEXM	*Hexadecimal mode.*
DECM	*Decimal mode.*
OCTM	*Octal mode.*
BINM	*Binary mode.*
LOGIC	Displays logic functions (page 250).

Select the Base application. The white box indicates that the current mode is Decimal (base 10).

■ [BASE]

```
x: 0.0000
A…F HEXM DEC■ OCTM BINM LOGIC
```

Key in a number and then switch to Hexadecimal mode (base 16).

31806 HEXM

```
x: 7C3E
A...F | HEX■ | DECM | OCTM | BINM | LOGIC
```

Switch to Octal mode to display the number in base 8.

OCTM

```
x: 76076
A...F | HEXM | DECM | OCT■ | BINM | LOGIC
```

Now, key in the hexadecimal number A14D. Pressing A...F automatically selects Hexadecimal mode and displays a submenu for keying the digits A through F.

A...F

```
x: 7C3E
A | B | C | D | E | F
```

A14D

```
x: A14D_
A | B | C | D | E | F
```

Display $A14D_{16}$ in Binary mode (base 2).

[EXIT] BINM

```
x: 1010000101001101
A...F | HEXM | DECM | OCTM | BIN■ | LOGIC
```

Change the sign of the number (which is the *2's complement*).

[+/-]

```
x: 11111111111111111111...
A...F | HEXM | DECM | OCTM | BIN■ | LOGIC
```

To view a binary number that is too large for the display, press and hold ■[SHOW].

■[SHOW] *(hold down)*

```
1111111111111111111111101
0111101011 0011
```

(release)

```
x: 11111111111111111111...
A...F | HEXM | DECM | OCTM | BIN■ | LOGIC
```

When you exit from the Base application, the calculator returns to Decimal mode.

[EXIT]

```
Y: 31,806.0000
x: -41,293.0000
```

Keying In Numbers of Different Bases. The current base mode determines which digit keys can be used to key in numbers:

- In Hexadecimal mode use the [0] – [9] and A – F keys (press A…F to select the A…F menu).
- In Decimal mode use the [0] – [9] keys.
- In Octal mode use the [0] – [7] keys.
- In Binary mode use the [0] and [1] keys.

The calculator will not allow you to key in nondecimal numbers that exceed the 36-bit word length. Refer to "Range of Numbers" below.

Base Arithmetic. The Base application redefines the arithmetic keys ([+], [−], [×], [÷], and [+/−]) to their corresponding integer arithmetic functions. For example, if you press [+], the calculator executes the BASE+ function instead of the normal addition function. Refer to "Integer Arithmetic" later in this chapter.

The Representation of Numbers

Base modes change the way real numbers are keyed in and displayed. Internally, however, real numbers are stored in decimal form regardless of the base mode.

In Hexadecimal, Octal, and Binary modes, numbers appear as integers. However, since the internal representation does not change, each number may have a nonzero fractional part. The calculator indicates that a nonzero fractional part exists by displaying a decimal point after the integer.

7C3E	7C3E.
This number *does not* have a fractional part internally.	This number *does* have a fractional part internally.

Negative Numbers

The left-most (most significant or "highest") bit of a number's binary representation is the sign bit; it is set (1) for negative numbers. If there are leading zeros (not displayed), then the sign bit is 0 (positive). A negative number is the 2's complement of its positive binary number.

Showing Numbers

The ■ SHOW key can be used in the Base application to:

- Display a hexadecimal, decimal, or octal number in full-precision *decimal* form.
- Display all 36 bits of a binary number.

Range of Numbers

The 36-bit word size determines the range of numbers that can be displayed in Hexadecimal, Octal, and Binary modes.

Range of Numbers for Base Conversions

Base	Largest Negative Number to Largest Positive Number
Hexadecimal	800000000 to 7FFFFFFFF
Decimal	-34,359,738,368 to 34,359,738,367
Octal	400000000000 to 377777777777
Binary	100000000000000000000000000000000000 to 011111111111111111111111111111111111

When you key in numbers, the maximum number of digits for each base. For example, if you attempt to key in a 10-digit hexadecimal number, digit entry halts after the ninth digit.

Numbers Too Big To Display. Nondecimal numbers outside the 36-bit range are displayed as `<Too Big>`. Don't mistake `<Too Big>` for an error message—it's merely the calculator's way of displaying a number that is too big to display in the current number base.

Integer Arithmetic

There are five functions for doing 36-bit integer arithmetic. These functions use only the integer portion of their operands and return only integer results. For example, if you add the numbers 15.7832 and 10.4859 using the BASE+ function, the result is 25.0000 because the fraction portion of each operand is ignored.

36-Bit Arithmetic Functions

Function	Description
BASE+	Integer addition.
BASE−	Integer subtraction.
BASE×	Integer multiplication.
BASE÷	Integer division.
BASE+/−	2's complement.

Note: When the BASE menu is displayed, these functions are automatically assigned to the [+], [−], [×], [÷], and [+/−] keys, respectively.

The calculator displays `Out of Range` if the result produced by any of these operations is greater than the 36-bit word size. If flag 24 (*range ignore*) is set, the calculator substitutes the maximum 36-bit number and does not report an error.

The Logic Functions

Pressing **LOGIC** in the BASE menu displays a submenu containing six logical functions. Like the integer arithmetic functions, the logic functions use only the integer portion of a number and return only integers as results.

■ BASE LOGIC →

AND	36-bit *AND.*
OR	36-bit *OR.*
XOR	36-bit *exclusive OR.*
NOT	36-bit *NOT.*
BIT?	*Bit test.*
ROTXY	*Rotate y* by *x* bits.

The Boolean Logic Functions. The AND, OR, and XOR functions are two-number functions. That is, they take two numbers (from the X- and Y-registers) and return a result to the X-register.

The NOT function returns the 36-bit logical NOT of the number in the X-register.

Bit Test. To test the x^{th} bit of the number in the Y-register, execute the BIT? function. Bits are numbered from 0 (the least significant bit) through 35 (the most significant bit). For example, the bits in this binary number* are numbered as shown:

```
101001111011011110101101001000011101
↑                                  ↑
Bit number 35                      Bit number 0
```

If BIT? is executed from the keyboard, the calculator displays `Yes` or `No` indicating whether the specified bit is set.

In a running program, the BIT? function follows the do-if-true rule—if the specified bit is set (1), the next program line is executed; if the bit is not set (0), the next program line is skipped.

* The decimal equivalent of this number is −23,698,157,027.

Rotating a 36-Bit Number. To rotate a 36-bit number by a specified number of bits, enter the number into the Y-register, the number of bits in the X-register, and then execute the ROTXY function. If the number of bits specified in the X-register is positive, the rotation is to the right. If the number of bits is negative, the rotation is to the left.

ROTXY returns the rotated number to the X-register and drops the stack.

Programming Information

To select a base mode in a program, execute HEXM, DECM, OCTM, or BINM. If a program stops after executing one of these instructions, the BASE menu is displayed and real numbers are input and displayed using the base mode that was selected. To exit the BASE menu, the program can then execute the EXITALL function.

You can also use the BASE menu for entering the base conversion and logic functions into a program. However, numbers entered directly into program lines are always entered and displayed in *decimal* form.

Example: A Program That Uses Base Operations. The following program prompts for an octal number and a binary number, adds them together, and displays the sum in Hexadecimal mode.

`01 LBL "OBH"`	Global label.
`02 OCTM` `03 INPUT 01`	Selects Octal mode and inputs the first number into R_{01}.
`04 BINM` `05 INPUT 02`	Selects Binary mode and inputs the second number into R_{02}.
`06 RCL 01` `07 BASE+`	Recalls a copy of the first number and adds to the second.
`08 HEXM` `09 VIEW ST X` `10 EXITALL` `11 END`	Selects Hexadecimal mode, displays the result, and exits the BASE menu.

Part 4

Appendixes and Reference

A

Assistance, Batteries, and Service

Obtaining Help in Operating the Calculator

Hewlett-Packard is committed to providing the owners of HP calculators with ongoing support. You can obtain answers to your questions about using the calculator from our Calculator Support department.

You should read the next section, "Answers to Common Questions," before contacting us. Past experience has shown that many of our customers have similar questions about our products. If you don't find an answer to your question, you can contact us using the address or phone number listed on the inside back cover.

Answers to Common Questions

Q: *I'm not sure if the calculator is malfunctioning or if I'm doing something incorrectly. How can I determine if the calculator is operating properly?*

A: Refer to page 261, which describes the diagnostic self-test.

Q: *My numbers contain commas as decimal points. How do I restore the periods?*

A: Press ■ [DISP] RDX. . Also check the status of flag 29 (page 276).

Q: *How do I change the number of decimal places the calculator displays?*

A: The procedure is described under "Number of Decimal Places" on page 34.

Q: *When I take the sine of π in radians mode, I get a small number (−2.06761537357 E-13) instead of zero. Why?*

A: The value returned is correct. While π has an infinite number of significant digits, the HP-42S uses the best possible 12-digit approximation of π. Given the inherent limitation of a finite number of input digits, the trigonometric functions provide the most accurate 12-digit results possible.

Q: *When I calculate* $\sqrt[3]{-27}$ (27 [+/-] [ENTER] 3 [1/x] ■ [y^x]) *I get a complex number (1.5000 i2.5981). Why?*

A: The value returned is correct. There are three possible answers, the HP-42S returns the root in the first quadrant. If you switch to Polar mode (■ [MODES] POLAR) you'll see that the number is 3 ∡60°.

To calculate the real-number cube root, use the following program:

```
01 LBL "CROOT"
02 SIGN
03 LASTX
04 3
05 1/X
06 Y↑X
07 ABS
08 ×
09 END
```

Q: *My calculator doesn't stop to display answers. They appear briefly and then computation resumes. How can I get the program to stop long enough to read the results?*

A: Set flag 21 (■ [FLAGS] SF 21). Flags 21 and 55 are used in conjunction to control the display and printer output. For more information on these flags, refer to pages 131 and 132.

Q: *How do I clear all or portions of memory?*

A: Press ■ [CLEAR] to display the CLEAR menu then execute the function you need. Refer to page 26.

Q: *What does an "E" in a number mean? (For example, 2.51E−13.)*

A: *Exponent* of ten (such as 2.51×10^{-13}). Refer to "Exponents of Ten" on page 27.

Q: *The calculator has displayed the message* `Insufficient Memory`. *What should I do?*

A: There is not enough memory to complete the operation that you attempted. Refer to appendix B, "Managing Calculator Memory."

Q: *Why isn't my calculator printing when I want it to?*

A: Printing is disabled. Press ■ [PRINT] [▲] PON to enable printing. Also refer to the owner's manual for the printer to see that you are positioning the calculator properly in front of the printer.

Q: *The calculator is operating slowly, and the* ▬ *annunciator is blinking. Why?*

A: The calculator is trace printing. Press ■ [PRINT] [▲] MAN to turn tracing off (page 102).

Q: *The beeper is not working. Why?*

A: The beeper has been disabled by executing the QUIET function or by clearing flag 26. Set the flag by pressing ■ [MODES] [▼] QUIE■ or ■ [FLAGS] SF 26.

Q: *How do I key in consecutive numbers in a program?*

A: Key in the first number, press [ENTER] [◆], and then key in the second number (page 118).

Q: *What is indirect addressing?*

A: It is used when a parameter for a particular function is stored in a variable or register. That variable or register is addressed (indirectly) by the function (page 74).

Q: *Why can't I get to the end of the matrix I'm editing? It appears to be much larger than when I created it.*

A: The Matrix Editor is in Grow mode. In the Matrix Editor menu, press [▼] WRAP to disable Grow mode (page 213).

Power and Batteries

The calculator is shipped with three mercury batteries. A fresh set of three mercury or silver oxide batteries provides approximately one year of normal use. (Alkaline batteries last about half as long.) However, expected battery life depends on how the calculator is used. Printing and long calculations require much more power than other operations.

Use only fresh button-cell batteries. Do not use rechargeable batteries. The following batteries are recommended for use. Not all batteries are available in all countries.

Mercury	Alkaline	Silver Oxide
Panasonic NP675	Panasonic LR44	Panasonic SR44W or SP357
Duracell MP675H	Varta VI3GA	Eveready 357
Toshiba NR44 or MR44	Eveready A76	Ray-O-Vac 357
Radio Shack NR44 or MR44	Duracell LR44	Varta V357
Eveready EP675E		

Low-Power Indications

When the low-battery annunciator (▭) comes on, replace the batteries as soon as possible.

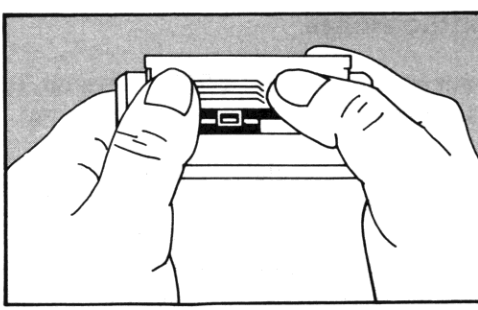
If you continue to use the calculator after the battery annunciator comes on, power can eventually drop to a level at which the calculator stops powering the display and keyboard. The calculator will require fresh batteries before it can be turned back on. When you turn the calculator on after fresh batteries have been installed, the calculator displays `Machine Reset` if your stored data is intact. If data has been lost, the calculator displays `Memory Clear`.

To conserve battery power, printing does not function when the battery annunciator is on. Printing might halt during a printing operation due to a borderline low-battery condition. The calculator can detect that there is insufficient power for printing before the battery annunciator comes on.

Installing Batteries

Once the batteries are removed, you must replace the batteries within one minute to prevent loss of Continuous Memory. Therefore, you should have the new batteries readily at hand before removing the batteries. Also, you must make sure the calculator is off during the entire process of changing batteries.

1. Have three fresh button-cell batteries at hand.
2. Make sure the calculator is *off*. **Do not press [EXIT] again until the entire procedure for changing batteries is completed. Changing batteries with the calculator on will erase the contents of Continuous Memory.**
3. Hold the calculator as shown. To remove the battery-compartment door, press down and outward on it until it slides off (away from the center).

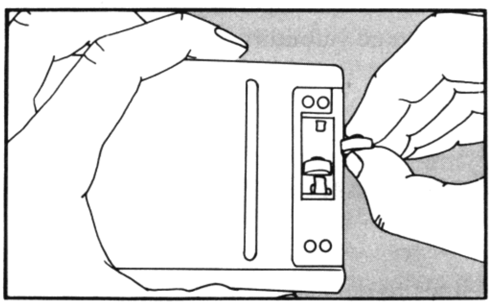


4. Turn the calculator over and shake the batteries out.

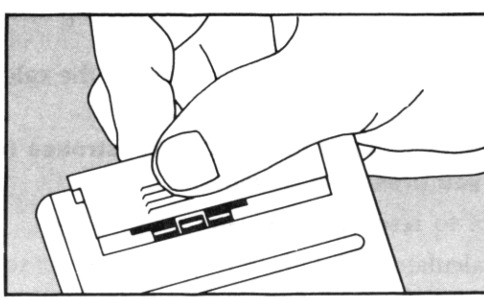
Warning **Do not mutilate, puncture, or dispose of batteries in fire. The batteries can burst or explode, releasing hazardous chemicals.**

---

5. Hold the calculator as shown and stack the batteries, one at a time, in the battery compartment. Orient the batteries according to the diagram inside the battery compartment. Be sure the raised and flat ends match the diagram.



6. Insert the tab of the battery-compartment door into the slot in the calculator case, as shown.



Now turn the calculator back on. If it does not function, you might have taken too long to change the batteries or inadvertently turned the calculator on while the batteries were out. *Remove the batteries* again and lightly press a coin against both battery contacts in the calculator *for a few seconds.* Put the batteries back in and turn the calculator on. It should display `Memory Clear`.

Environmental Limits

To maintain product reliability, observe the following limits:

- Operating temperature: 0° to 45°C (32° to 113°F).
- Storage temperature: −20° to 65°C (−4° to 149°F).
- Operating and storage humidity: 90% relative humidity at 40°C (104°F) maximum.

Determining if the Calculator Requires Service

Use these guidelines to determine if the calculator requires service. If it does, read "If the Calculator Requires Service" on page 263.

- **If the calculator won't turn on (nothing is visible in the display):**
  1. Attempt to reset the calculator (page 267).
  2. If the calculator fails to respond after step 1, replace the batteries (page 258).

  If steps 1 and 2 do not restore the display, the calculator requires service.
- **If the calculator doesn't respond to keystrokes (nothing happens when you press the keys):**
  1. Attempt to reset the calculator (page 267).
  2. If the calculator still fails to respond, attempt to clear all memory (page 268). This will erase all the information you've stored.

  If steps 1 and 2 do not restore calculator function, the calculator requires service.

- **If the calculator responds to keystrokes but you suspect that it is malfunctioning:**
  1. Do the self-test (described below). If the calculator fails the self-test, it requires service.
  2. If the calculator passes the self-test, it is quite likely you've made a mistake in operating the calculator. Try rereading portions of the manual, and check "Answers to Common Questions" on page 254.
  3. Contact the Calculator Support department. The address and phone number are listed on the inside back cover.

Confirming Calculator Operation—the Self-Test

If the display can be turned on, but it appears that the calculator is not operating properly, you can do a diagnostic self-test. The self-test runs continuously, repeating until you halt it.

To run the self-test:

1. Turn the calculator on.
2. If you have the optional infrared printer, turn it on. Certain diagnostic information is printed during the test.
3. To start the self-test, hold down [EXIT] while you press the [LN] key.* Once the self-test has begun, do not press any keys until you are ready to halt the test.
4. During the test, the calculator beeps periodically and displays various patterns and characters. Watch for one of two messages that are displayed before the test automatically repeats:
   - If the calculator passes the self-test, the calculator displays `OK-42S-E`.
   - If the calculator displays `FAIL` followed by a number, the calculator may require service.

* Pressing the [LOG] key starts another self-test that is used at the factory. If you accidently start this self-test, you can stop it by holding down the [EXIT] key while you press the [$\sqrt{x}$] key.

5. To halt the self-test, hold down [EXIT] while you press the [√x] key. The calculator displays `Machine Reset`. If you press any other key instead, the test halts and the calculator displays a `FAIL` message. *This message results from an incorrect key being pressed and does not mean that the calculator requires service.*
6. If the calculator failed the self-test, repeat steps 3 through 5 to verify the results. If you do not have a printer, write down the messages that are displayed in step 5 above.

Limited One-Year Warranty

What Is Covered

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

This warranty gives you specific legal rights, and you may also have other rights that vary from state to state, province to province, or country to country.

What Is Not Covered

Batteries, and damage caused by the batteries, are not covered by the Hewlett-Packard warranty. Check with the battery manufacturer about battery and battery leakage warranties.

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

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

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

Consumer Transactions in the United Kingdom

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

If the Calculator Requires Service

Hewlett-Packard maintains service centers in many countries. These centers will service a calculator whether it is under warranty or not. There is a charge for service after the warranty period. Calculators normally are serviced and reshipped within five working days of receipt.

Obtaining Service

- *In the United States:* Send the calculator to the Calculator Service Center listed on the inside back cover.

- *In Europe:* Contact your HP sales office or dealer or HP's European headquarters for the location of the nearest service center. *Do not ship the calculator for service without first contacting a Hewlett-Packard office.*

  Hewlett-Packard S.A.
  150, Route du Nant-d'Avril
  P.O. Box
  CH 1217 Meyrin 2
  Geneva, Switzerland
  Telephone: (022) 82 81 11

- *In other countries:* Contact your HP sales office or dealer or write to the U.S. Calculator Service Center (listed on the inside back cover) for the location of other service centers. If local service is unavailable, you can ship the calculator to the U.S. Calculator Service Center for repair.

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

Service Charge

There is a standard repair charge for out-of-warranty service. The Calculator Service Center (listed on the inside back cover) can tell you how much this charge is. The full charge is subject to the customer's local sales or value-added tax wherever applicable.

Calculator products damaged by accident or misuse are not covered by the fixed service charges. In these cases, charges are individually determined based on time and material.

Shipping Instructions

If your calculator requires service, ship it to the nearest authorized service center or collection point. Be sure to:

- Include your return address and description of the problem.
- Include proof of purchase date if the warranty has not expired.

- Include a purchase order, check, or credit card number plus expiration date (Visa or MasterCard) to cover the standard repair charge. In the United States and some other countries, the serviced calculator will be returned C.O.D. if you do not pay in advance.
- Ship the calculator in adequate protective packaging to prevent damage. Such damage is not covered by the warranty, so we recommend that you insure the shipment.
- Pay the shipping charges for delivery to the Hewlett-Packard service center, whether or not the calculator is under warranty.

Warranty on Service

Service is warranted against defects in materials and workmanship for 90 days from the date of service.

Service Agreements

In the U.S., a support agreement is available for repair and service. Refer to the form included with the manual. For additional information, contact the Calculator Service Center (see the inside back cover).

Radio Frequency Interference

U.S.A. The HP-42S generates and uses radio frequency energy and may interfere with radio and television reception. The calculator complies with the limits for a Class B computing device as specified in Subpart J of Part 15 of FCC Rules, which provide reasonable protection against such interference in a residential installation. In the unlikely event that there is interference to radio or television reception (which can be determined by turning the HP-42S off and on or by removing the batteries), try:

- Reorienting the receiving antenna.
- Relocating the calculator with respect to the receiver.

For more information, consult your dealer, an experienced radio/television technician, or the following booklet, prepared by the Federal Communications Commission: *How to Identify and Resolve Radio-TV Interference Problems*. This booklet is available from the U.S. Government Printing Office, Washington, D.C. 20402, Stock Number 004-000-00345-4. At the first printing of this manual, the telephone number was (202) 783-3238.

West Germany. The HP-42S and the HP 82240A printer comply with VFG 1046/84, VDE 0871B, and similar non-interference standards.

If you use equipment that is not authorized by Hewlett-Packard, that system configuration has to comply with the requirements of Paragraph 2 of the German Federal Gazette, Order (VFG) 1046/84, dated December 14, 1984.

B

Managing Calculator Memory

This appendix describes how calculator memory is organized and the techniques used internally by the calculator to conserve memory. You do not need to read and understand this material to use your calculator. However, you may find the information useful. For example, you can write your programs to take advantage of the HP-42S memory management scheme.

Resetting the Calculator

If the calculator doesn't respond to keystrokes or is otherwise behaving unusually, attempt to reset it. Resetting the calculator resets many conditions to their default states (such as turning Program-entry mode off and exiting all menus). Refer to appendix C, "Flags," for a table of flag settings at machine reset.

To reset the calculator, hold down the [EXIT] key while you press the [√x] key. Repeat this if necessary. The calculator displays `Machine Reset` to confirm that reset has occurred.

Clearing All Memory

There are two ways to clear calculator memory:

To clear all programs and data:

1. Press ■ [CLEAR] [▼] CLALL.
2. Press YES to confirm or any other key to cancel.

To clear all programs and data and reset flags:

1. Press and hold [EXIT] (lower-left corner of the keyboard).
2. Press and hold [Σ+] (upper-left corner of the keyboard).
3. Press and release [XEQ] (upper-right corner of the keyboard).
4. Release [Σ+].
5. Release [EXIT]. The calculator displays `Memory Clear`.

Continuous Memory can inadvertently be erased if the calculator is dropped or if power is interrupted.

Reclaiming Memory

If you see the message `Insufficient Memory`, there is not enough memory to complete the operation that you attempted.

To determine the amount of memory available, press and hold MEM in the CATALOG menu. To reclaim memory—that is, increase the amount of available memory—do one or more of the following:

- Reduce the number of storage registers by using a smaller SIZE (page 64).
- Clear variables that you no longer need (page 62).
- Clear programs that you no longer need (page 119).
- Clear the stack (page 43).

How the HP-42S Conserves Memory

As described in chapter 3, the HP-42S uses several types of data. Because data types can range in size (from a real number up to a large complex matrix), a sophisticated operating system has been developed that makes it easy for you to manipulate data using a consistent set of RPN rules. The techniques presented in chapter 2 for using the stack apply to all types of data. For example, when you press [ENTER], the stack is lifted and the data in the X-register is copied into the Y-register.

What Happens When Data Is Copied

Whenever you make a copy of data (with operations such as [ENTER], [STO], and [RCL]), internally the calculator *does not* actually make a complete copy even though it appears to.

Example: Observing the Conservation of Memory. To demonstrate this principle of copying objects, clear the stack and create a 10 × 10 matrix named *TEST*.

Create the matrix using the DIM function.

10 [ENTER] ■ [MATRIX] [▼] DIM [ENTER] TEST [ENTER] [EXIT]	Y: 10.0000 X: 10.0000
■ [CLEAR] CLST	Y: 0.0000 X: 0.0000

View the amount of memory available. (Note: the memory available in your calculator will differ from the numbers shown in this example.)

■ [CATALOG] MEM *(hold down)*	Available Memory: 6157 Bytes

Fill the stack with copies of *TEST*.

[RCL] TEST	X: [10x10 Matrix] FCN PGM REAL CPX MAT MEM
[ENTER] [ENTER] [ENTER]	X: [10x10 Matrix] FCN PGM REAL CPX MAT MEM

Now, there appear to be five complete matrices in the calculator: one stored in the variable *TEST* and four in the stack. But when you view the amount of memory available, you can see that making these "copies" did not use any additional memory.

■ [CATALOG] MEM *(hold down)*	Available Memory: 6157 Bytes

Internally, the HP-42S does not make copies of data until it is used. Add 2 to the matrix.

2 [+]

```
x: [ 10×10 Matrix ]
FCN | PGM | REAL | CPX | MAT | MEM
```

View the amount of memory available now.

MEM *(hold down)*

```
Available Memory:
5326 Bytes
```

The new copy required 831 additional bytes of memory (6,157 − 5,326 = 831).

Writing Memory-Efficient Programs

Use the Stack Efficiently. Review chapter 2 and remember the rules for RPN calculations. Many complicated mathematical expressions can be evaluated using only the stack. That is, you can often complete a calculation without using *additional* variables or storage registers. For example, refer to the "TVM" program on page 192.

Use Local Labels Whenever You Can. If you do a lot of programming, you can save a substantial amount of memory by using local labels whenever you can. Local labels only require 1 or 2 bytes each and branch instructions to local labels never require more than 3 bytes. What's more, the search for a local label is usually much faster than a search for a global label (page 148).

Global labels, on the other hand, require 4 bytes plus 1 byte for each character in the label. Each branch instruction to a global label (GTO and XEQ) requires 2 bytes plus 1 byte for each character in the label.

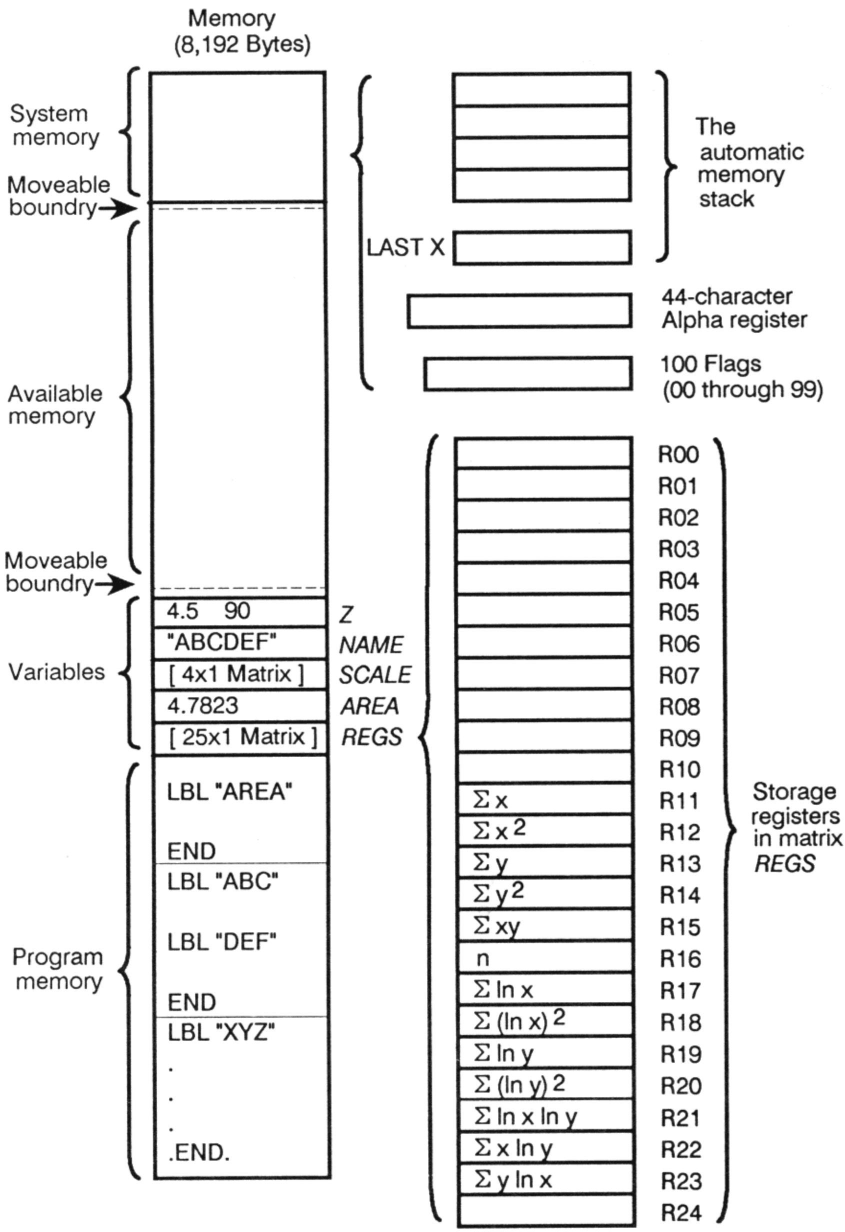
During Matrix Arithmetic. During some calculations with matrices, you can save memory by putting the smaller matrix or scalar in the X-register before executing a numeric function.

For example, if you are adding a scalar to a matrix, put the scalar in the X-register. Then, when you execute [+], the scalar (which uses less memory) gets saved in the LAST X register rather than the matrix.

Note that this technique does not reduce the amount of memory needed to perform the calculation—the calculator still uses a temporary workspace to calculate the result. However, it does increase the amount of memory available immediately after the calculation.

Memory Organization

The diagram on the next page illustrates how calculator memory is organized internally. *Available memory* is the unused portion of memory between the memory used for the stack (and other system memory) and the memory used to store variables and programs.



Memory
(8,192 Bytes)
System memory
Moveable boundry
Available memory
Moveable boundry
Variables
Program memory
4.5 90
"ABCDEF"
[4x1 Matrix]
4.7823
[25x1 Matrix]
Z
NAME
SCALE
AREA
REGS
LBL "AREA"
END
LBL "ABC"
LBL "DEF"
END
LBL "XYZ"
.
.
.
.END.
The automatic memory stack
LAST X
44-character Alpha register
100 Flags (00 through 99)
R00
R01
R02
R03
R04
R05
R06
R07
R08
R09
R10
Σx R11
Σx2 R12
Σy R13
Σy2 R14
Σxy R15
n R16
Σ ln x R17
Σ (ln x) 2 R18
Σ ln y R19
Σ (ln y) 2 R20
Σ ln x ln y R21
Σ x ln y R22
Σ y ln x R23
R24
Storage registers in matrix REGS

C

Flags

The HP-42S uses 100 flags (numbered 00–99) to keep track of various modes, settings, and conditions. A flag has only two states: *set* and *clear*. Flags are set, cleared, and tested using the functions in the FLAGS menu (page 41).

Flags that represent certain conditions may change in the course of operation. For example, when you press ■ [CUSTOM] to select the CUSTOM menu, flag 27 is set. When you exit from the CUSTOM menu, flag 27 is cleared.

Flags not listed in this appendix are either used internally or are reserved for future use.

User Flags (00 Through 10 and 81 Through 99)

The 30 *user flags* can be used to represent anything you want. For example, the "TVM" program on page 192 uses flag 00. If flag 00 is set, the program assumes payments are made at the beginning of each month; if flag 00 is clear, payments are made at the end of each month.

Control Flags (11 Through 35)

Control flags are used by the HP-42S to represent certain operating conditions. Some conditions are controlled only by altering flags, while others are changed by executing functions.

Flag 11: Automatic Execution. Flag 11 (if set before the calculator is turned off) allows a program to run automatically when the HP-42S is turned on. If flag 11 is set when you turn the HP-42S on, flag 11 is cleared, and program execution begins at the current program line.

Flag 12: Double-Wide Printer Output. If flag 12 is set, all printer output is printed double-wide.

Flag 13: Lowercase Printer Output. If flag 13 is set, the letters A through Z are printed in lowercase.

Flags 15 and 16: Print Mode. This table shows how flags 15 and 16 represent the current print mode.

Flag 15	Flag 16	Print Mode
clear	clear	Manual
clear	set	Normal
set	clear or set	Trace

Flag 21: Printer Enable. Flag 21 allows your program to control how the functions VIEW and AVIEW are executed. For more information, refer to "Printing With VIEW and AVIEW" on page 132.

Flags 22 and 23: Data Input. These flags allow a program that prompts for input to determine the user's response. Flag 22 is set whenever numbers are keyed into the X-register. Flag 23 is set whenever characters are keyed into the Alpha register.

If you intend to test these flags to determine if input has been made, you should clear them before prompting for the input.

Flags 24 and 25: Error Ignore. Normally, an error condition halts program execution. These flags allow you to avoid unnecessary program halts and to use error conditions as a programming tool.

- If flag 24 is set, the HP-42S ignores *all* range errors. `Out of Range` normally results from any calculation (except statistical accumulations) that produces a number x such that $|x| > 9.99999999999 \times 10^{499}$. If flag 24 is set, $\pm 9.99999999999 \times 10^{499}$ is returned as an approximation to the correct answer and program execution continues. Once you set flag 24, it remains set until you explicitly clear it.

  Flag 24 can also be used to ignore range errors produced by the 36-bit arithmetic functions (BASE+, BASE−, BASE×, and BASE÷), substituting the largest 36-bit number for an approximation to the correct answer (pages 248 and 249).

  You do not need to set flag 24 to prevent overflow errors when accumulating statistical data ([Σ+]) or using two-number functions on matrices. In these cases the calculator automatically returns $\pm 9.99999999999 \times 10^{499}$ when a result exceeds the range of the calculator.

- If flag 25 is set, the calculator ignores *only one* error of any kind and then clears flag 25. The instruction that caused the error is not executed.

  If both flags 24 and 25 are set, `Out of Range` is handled by flag 24—flag 25 is *not* cleared. Note that if flag 25 is set and flag 24 is clear, `Out of Range` does *not* cause $\pm 9.99999999999 \times 10^{499}$ to be placed in the appropriate register.

  You can detect an error by setting flag 25 just before an instruction and then testing the flag after the instruction to see if it has been cleared. (Generally, you should test *and clear* flag 25—you risk losing data if you choose to ignore unanticipated errors.) This enables a program to branch rather than to stop execution in case of an error.

Flag 26: Audio Enable. When flag 26 is set, the BEEP and TONE functions *will* produce audible tones. You can toggle flag 26 by executing the QUIET function in the MODES menu.

Flag 27: The CUSTOM Menu. Flag 27 is set whenever the CUSTOM menu is displayed. The status of flag 27 is not altered by turning the calculator off and on. Refer also to flag 72.

Flags 28 and 29: Display Punctuation. These flags control the use of periods and commas in numeric displays.

- If flag 28 is set (*default*), a period is used as the radix mark (to separate the integer portion of a number from the fractional part). If flag 28 is clear, a comma is used as the radix mark.
- If flag 29 is set (*default*), groups of digits in large numbers are separated. If flag 29 is clear, no digit separators are used. The character used to separate digits is a comma if the radix is a period, and a period if the radix is a comma.

  If the display format is set to FIX 0 and flag 29 is clear, the integer portion of the number is displayed without any punctuation.

Flag 30: Stack Lift Disable. This flag is cleared by nearly all functions. The functions that set flag 30 are ENTER, CLX, Σ+, and Σ−. If stack lift is disabled (flag 30 set), the next number keyed or recalled into the X-register writes over the current contents of the X-register (pages 45 through 46).

Flags 34 and 35: AGRAPH Control. The status of these two flags determines how a graphics image is displayed by the AGRAPH function. When both flags are clear (default), the image is merged with the existing contents of the display (logical OR). Refer to the table on page 137.

System Flags (36 Through 80)

The HP-42S uses system flags to keep track of a number of options and conditions. You cannot directly alter system flags. You can, however, test system flags, which might be useful in programs to detect particular options and conditions.

Flags That Represent Options

Flags 36 Through 41: Display Format. These flags represent the current display format. The calculator reads flags 36 through 39 as a four-bit binary number that specifies the number of digits to display. For example, the default format calls for 4 digits (flag 37 set; flags 36, 38, and 39 clear). That is, 0100 (binary) = 4 (decimal).

36	37	38	39
0	1	0	0

Flags 40 and 41 are used to represent the display format (FIX, SCI, ENG, or ALL).

Flag 40	Flag 41	Display Format
Clear	Clear	SCI
Clear	Set	ENG
Set	Clear	FIX (default)
Set	Set	ALL

Flags 42 and 43: Angular Mode. The status of flags 42 and 43 determines the angular mode (Degrees, Radians, or Grads). If flag 42 is set (**GRAD** on), the calculator is in Grads mode. If flag 43 is set (**RAD** on), the calculator is in Radians mode. If both flags are clear (default), the calculator is in Degrees mode.

Flags 56 Through 59: Curve-Fitting Model. These flags are used to indicate the current curve-fitting model. Only one of these four flags can be set at a time.

Flag	Curve-Fitting Model
56	Linear (default)
57	Logarithmic
58	Exponential
59	Power

Flag 60: AllΣ Mode. If flag 60 is set (AllΣ mode), the calculator uses all 13 summation coefficients for statistical calculations. If flag 60 is clear (Linear mode), the calculator uses only the six coefficients needed for linear curve fitting.

Flag 66: Grow Mode. If flag 66 is set, a matrix automatically grows by one complete row if you execute the → or J+ function while positioned at the last element in the matrix.

Flags 68 Through 71: Base Mode. If the current base mode is Decimal, all four of these flags are clear. In nondecimal modes, these flags are used as a four-bit number indicating the largest digit allowed in the current mode.

Base Mode	Flags 71	70	69	68	Largest Digit
Binary	0	0	0	1	1
Octal	0	1	1	1	7
Hexadecimal	1	1	1	1	F

Flag 72: Local Label Mode (CUSTOM). The calculator tests this flag before displaying the CUSTOM menu. (Refer to the menu maps on page 301). If flag 72 is set, the CUSTOM menu for executing local Alpha labels is displayed. If flag 72 is clear, the CUSTOM menu key assignments are displayed. Making a key assignment automatically clears flag 72.

To set flag 72, press ■ [MODES] [▼] LCLBL (*Local-label mode*). To clear flag 72, press ■ [MODES] [▼] KEY (*Key-assignment mode*).

Flag 73: Polar Mode. If flag 73 is set, the calculator displays complex numbers using polar notation.

Flag 74: Real-Result Only. If flag 74 is set, the calculator returns an error for functions that would turn a real-number input into a complex-number result (such as calculating the square root of a negative real number). Refer to page 169.

Flags That Represent Conditions

Flag 44: Continuous On. Flag 44 is set when the ON (*continuous on*) function is executed. The calculator automatically turns off after about 10 minutes of inactivity (no keys pressed) unless flag 44 is set.

Flag 45: Solving. Flag 45 is set only while the Solver is calculating a solution.

Flag 46: Integrating. Flag 46 is set only while the Integration applicaton is evaluating an integral.

Flag 47: Variable Menu. Flag 47 is only set when a variable menu is active (page 125).

Flag 48: Alpha Mode. Whenever the calculator is in Alpha mode (ALPHA menu and Alpha register displayed), flag 48 is set. You can control Alpha mode by executing AON (*Alpha on;* sets flag 48) and AOFF (*Alpha off;* clears flag 48).

Flag 49: Low Battery Power. Flag 49 is set and the annunciator is displayed when battery power is low. Refer to page 258 for information on replacing batteries.

Flags 50 and 51: Message. Flag 50 is set whenever a message is displayed. If the message uses both lines of the display, flag 51 is also set.

Flag 52: Program-Entry Mode. Whenever the calculator is in Program-entry mode, flag 52 is set.

Flag 53: INPUT. Flag 53 is set only while an INPUT is in progress (page 121). Note that the INPUT function cannot be executed from the keyboard.

Flag 55: Printer Existence. Executing the PRON (*printing on*) function enables printing by setting flags 21 and 55. Executing PROFF (*printing off*) disables printing and clears flags 21 and 55.

In general, flag 55 indicates if printing is *possible*. Flag 21 indicates if printing is *desired*.

Flags 61 Through 63: Invalid Models. These flags are used during entry of statistical data to identify which curve-fitting models are invalid.

Flag	Invalid Model (if set)
61	Logarithmic
62	Exponential
63	Power

Flag 65: Matrix Editor. Flag 65 is set if the Matrix Editor is in use.

Flag 75: Programmable Menu Selected. If flag 75 is set, the programmable menu (page 145) is selected. The MENU function sets flag 75.

Flags 76 and 77: Matrix Wrap. These flags are updated each time you execute any of the matrix functions that alter the row and column pointers.

- If the function causes the pointers to be wrapped from one edge of the matrix to the opposite edge (*edge wrap*), flag 76 is set. Otherwise the flag is cleared.
- If the function causes the pointers to be wrapped from the first element to the last or from the last element to the first (*end wrap*), then flag 77 is set. Otherwise, the flag is cleared.

Summary of HP-42S Flags

The following table lists all of the flags used by the HP-42S. *Status at Machine Reset* indicates if the flag is set or cleared when you reset the calculator. *Status at Memory Clear* indicates if the flag is set or cleared when you erase all of memory. An M indicates that the current status of the flag is *maintained* (not changed). A ? indicates that the status of the flag depends on other factors.

Flag Number	Flag Name	Status at Machine Reset	Status at Memory Clear
00–10	User Flags	M	Clear
11	Automatic Execution	Clear	Clear
12	Print Double-Wide	M	Clear
13	Print Lowercase	M	Clear
14	Reserved	M	Clear
15–16	Print Mode	M	Clear
17–18	Reserved	M	Clear

Flag Number	Flag Name	Status at Machine Reset	Status at Memory Clear
19–20	General Use	M	Clear
21	Printer Enable	M	Clear
22	Numeric Data Input	Clear	Clear
23	Alpha Data Input	Clear	Clear
24	Range Error Ignore	Clear	Clear
25	Error Ignore	Clear	Clear
26	Audio Enable	M	Clear
27	CUSTOM Menu	Clear	Clear
28	Radix Mark (. or ,)	M	Clear
29	Digit Separators	M	Clear
30	Stack Lift Disable	Clear	Clear
31–33	Reserved	?	?
34–35	AGRAPH Control	M	Clear
36–39	Number of Digits	M	4 Digits*
40–41	Display Format	M	FIX*
42	Grads Mode	M	Clear
43	Radians Mode	M	Clear
44	Continuous On	Clear	Clear
45	Solving	Clear	Clear
46	Integrating	Clear	Clear
47	Variable Menu	Clear	Clear
48	Alpha Mode	Clear	Clear
49	Low Battery	?	?
50	Message	Set	Set
51	Two-Line Message	Clear	Clear
52	Program-Entry	Clear	Clear
53	INPUT	Clear	Clear

* Refer to the description on page 276.

Flag Number	Flag Name	Status at Machine Reset	Status at Memory Clear
54	Reserved	Clear	Clear
55	Printer Existence	M	Clear
56	Linear Curve-Fitting Model	M	Set
57	Logarithmic Curve-Fitting Model	M	Clear
58	Exponential Curve-Fitting Model	M	Clear
59	Power Curve-Fitting Model	M	Clear
60	AllΣ Mode (statistics)	M	Set
61	Logarithmic Model Invalid	M	Clear
62	Exponential Model Invalid	M	Clear
63	Power Model Invalid	M	Clear
64	Reserved	M	Clear
65	Matrix Editor	Clear	Clear
66	Grow Mode	Clear	Clear
67	Reserved	Clear	Clear
68–71	Base Mode	Clear	Clear
72	Local-Label Mode (CUSTOM)	M	Clear
73	Polar Mode	M	Clear
74	Real-Result Only	M	Clear
75	Programmable Menu Active	Clear	Clear
76	Edge Wrap	M	Clear
77	End Wrap	M	Clear
78–80	Reserved	M	Clear
81–99	User Flags	M	Clear

D

Messages

The HP-42S displays messages to provide information and to warn you when you attempt an invalid operation. The message disappears when you press a key. To clear the message without altering anything else, press [◄].

`Alpha Data Is Invalid`
Attempted an operation using a variable, storage register, or stack register containing an Alpha string.

`Bad Guess(es)`
Guess(es) provided for the Solver are outside the domain of the function.

`Batt Too Low To Print`
The battery voltage is too low to power the calculator's infrared printer interface. Whenever the calculator displays this message, it also resets to Manual printing mode.

`Constant?`
The function returned the same value at every point sampled by the Solver.

`Dimension Error`

- Dimensions of two matrices are not compatible for matrix arithmetic.
- Attempted to calculate the determinant of a nonsquare matrix.
- Attempted to create a matrix with one or both dimensions less than or equal to zero.
- Attempted to move the index pointers beyond the dimensions of the indexed matrix.

`Divide by 0`
Attempted to divide by zero.

`Extremum`
Local minimum or maximum has been found by the Solver.

`Global Span`
Attempted to insert or delete a program line that would have left more than 3,584 bytes of program instructions between two global labels or a global label and an END.

`Insufficient Memory`
There is not enough memory to complete the attempted operation. In addition to the memory needed to complete the operation, the calculator always keeps some memory available as a system work space.

`Integ(Integ)`
Attempted to integrate a function while another integration was in progress.

`Integrating`
The calculator is calculating an integral (chapter 13).

`Interrupted`
A matrix operation has been interrupted by pressing [EXIT].

`Invalid Data`
Attempted a function using data outside the range of the function.

`Invalid Forecast Model`
The current statistical data is invalid or incomplete to use the selected curve-fitting model for forecasting.

`Invalid Type`
Data type does not match the type expected (real, complex, or matrix).

`Label Not Found`
Attempted an instruction that referenced a program label that does not exist.

`Machine Reset`
The calculator has been reset (page 267):

- All menus are exited.
- Program-entry mode is exited.
- All pending RTN locations are cleared.
- The display contrast is set to a middle setting.

`Memory Clear`
All of continuous memory has been cleared (page 268).

`No`
The proposition made by a test function executed from the keyboard is false. For example, the calculator displays `No` if you press ■ [FLAGS] `FS?` 03 when flag 03 is clear.

`No Complex Variables`
There are no variables in the complex-variable catalog.

`No Matrix Variables`
There are no variables in the matrix-variable catalog.

`No Menu Variables`
Attempted to display a variable menu with VARMENU, ■ [SOLVER], or ■ [∫f(x)] using a global label that is not followed by MVAR (*menu variable*) instructions.

`No Real Variables`
There are no variables in the real-variable catalog.

`No Variables`
Attempted to execute a function that requires a variable name as a parameter and there are no variables currently stored in the calculator.

`Nonexistent`

- Attempted to use a variable that does not exist.
- Attempted a matrix utility function when there is no indexed matrix.

`Out of Range`
The result of the attempted operation would exceed the numeric range of the calculator. You can use flag 24 to ignore this error.

`Printing Is Disabled`
A print operation was attempted from the keyboard with printing disabled (flag 55 clear). To enable printing, press ■ [PRINT] [▲] PON .

`Restricted Operation`

- Attempted to set or clear a flag in the range 36 through 80.
- Attempted to use a function from the keyboard that can only be used in programs.
- Attempted to enter a nonprogrammable function into a program.
- Attempted to store a number into *REGS*. The variable name *REGS* can only be used to store a matrix.
- Attempted to redimension, index, or clear the named matrix currently being edited.
- Attempted to execute the DEL (*delete*) function while not in Program-entry mode.
- Attempted to delete a row (DELR) in a matrix that has only one row.

`Sign Reversal`
An approximation to a solution has been found by the Solver but it may not be a normal solution.

`Size Error`

- Attempted to store or recall a storage register that does not exist.
- Attempted a statistical function when one or more of the summation registers do not exist.

`Solve/Integ RTN Lost`
The RTN (*return*) location for the Solver or Integration was lost. The calculator can remember up to eight pending return locations.

`Solve(Solve)`
Attempted to solve a function while another solve was in progress.

`Stat Math Error`
The statistical data is invalid or incomplete.

`Yes`
The proposition made by a test function executed from the keyboard is true. For example, the calculator displays `Yes` if you press ■ [FLAGS] FS? 03 when flag 03 is set.

Note The HP-42S uses `<Too Big>` in the Base application to display any number that is too big to display using a nondecimal base mode. That is, `<Too Big>` is a number, *not an error message*. Refer to page 249. To view a number that is displayed as `<Too Big>`, press and hold ■ [SHOW].

E

Character Table

The following table lists all of the Alpha characters used by the HP-42S. The keystrokes shown in the table assume that the first or second row of the ALPHA menu is displayed (■ [ALPHA] or ■ [ALPHA] [▼]).

Display Character	Character Code Dec	Hex	Keystrokes*
÷	0	00	[÷]
×	1	01	[×]
√	2	02	MATH √
∫	3	03	MATH ∫
▒	4	04	
Σ	5	05	MATH Σ
▶	6	06	
π	7	07	■ [π]
¿	8	08	PUNC [▼] ¿
≤	9	09	< = > ≤
␊	10	0A	PUNC [▼] ␊
≥	11	0B	< = > ≥
≠	12	0C	< = > ≠
↵	13	0D	
↓	14	0E	←↑↓ ↓
→	15	0F	←↑↓ →
←	16	10	←↑↓ ←
μ	17	11	MATH μ
£	18	12	MISC [▼] £
▪	19	13	MATH ▪

* If a character is not typable (no keystrokes) you can enter it into the Alpha register by keying the character code into the X-register and then executing the XTOA function.

Display Character	Character Code Dec	Character Code Hex	Keystrokes*
Å	20	14	ABCDE ▼ Å
Ñ	21	15	NOPQ ▼ Ñ
Ä	22	16	ABCDE ▼ Ä
∡	23	17	MATH ∡
ᴇ	24	18	E
Æ	25	19	ABCDE ▼ Æ
…	26	1A	PUNC ▼ …
␛	27	1B	
Ö	28	1C	NOPQ ▼ Ö
Ü	29	1D	RSTUV ▼ Ü
▒	30	1E	
■	31	1F	MISC ■
(space)	32	20	WXYZ
!	33	21	PUNC !
"	34	22	PUNC "
#	35	23	MISC #
$	36	24	MISC $
%	37	25	■ %
&	38	26	MISC ▼ &
'	39	27	PUNC ▼ '
(	40	28	([{ (
)	41	29	([{)
*	42	2A	MISC *
+	43	2B	+
,	44	2C	PUNC ,
–	45	2D	–
.	46	2E	.
/	47	2F	MISC /
0	48	30	0
1	49	31	1
2	50	32	2
3	51	33	3
4	52	34	4
5	53	35	5
6	54	36	6
7	55	37	7
8	56	38	8
9	57	39	9
:	58	3A	PUNC :

* If a character is not typable (no keystrokes) you can enter it into the Alpha register by keying the character code into the X-register and then executing the XTOA function.

Display Character	Character Code Dec	Hex	Keystrokes
;	59	3B	PUNC ;
<	60	3C	< = > <
=	61	3D	< = > =
>	62	3E	< = > >
?	63	3F	PUNC ?
@	64	40	MISC ▼ @
A	65	41	ABCDE A
B	66	42	ABCDE B
C	67	43	ABCDE C
D	68	44	ABCDE D
E	69	45	ABCDE E
F	70	46	FGHI F
G	71	47	FGHI G
H	72	48	FGHI H
I	73	49	FGHI I
J	74	4A	JKLM J
K	75	4B	JKLM K
L	76	4C	JKLM L
M	77	4D	JKLM M
N	78	4E	NOPQ N
O	79	4F	NOPQ O
P	80	50	NOPQ P
Q	81	51	NOPQ Q
R	82	52	RSTUV R
S	83	53	RSTUV S
T	84	54	RSTUV T
U	85	55	RSTUV U
V	86	56	RSTUV V
W	87	57	WXYZ W
X	88	58	WXYZ X
Y	89	59	WXYZ Y
Z	90	5A	WXYZ Z
[	91	5B	([{ [
\	92	5C	MISC ▼ \
]	93	5D	([{]
↑	94	5E	←↑↓ ↑
_	95	5F	PUNC ▼ _
`	96	60	PUNC ▼ `
a	97	61	ABCDE ■ A
b	98	62	ABCDE ■ B
c	99	63	ABCDE ■ C
d	100	64	ABCDE ■ D

Display Character	Character Code Dec	Hex	Keystrokes*
e	101	65	ABCDE ■ E
f	102	66	FGHI ■ F
g	103	67	FGHI ■ G
h	104	68	FGHI ■ H
i	105	69	FGHI ■ I
j	106	6A	JKLM ■ J
k	107	6B	JKLM ■ K
l	108	6C	JKLM ■ L
m	109	6D	JKLM ■ M
n	110	6E	NOPQ ■ N
o	111	6F	NOPQ ■ O
p	112	70	NOPQ ■ P
q	113	71	NOPQ ■ Q
r	114	72	RSTUV ■ R
s	115	73	RSTUV ■ S
t	116	74	RSTUV ■ T
u	117	75	RSTUV ■ U
v	118	76	RSTUV ■ V
w	119	77	WXYZ ■ W
x	120	78	WXYZ ■ X
y	121	79	WXYZ ■ Y
z	122	7A	WXYZ ■ Z
{	123	7B	([{ {
|	124	7C	MISC ▼ |
}	125	7D	([{ }
~	126	7E	MISC ▼ ~
⊢	127	7F	ENTER†
:	128	80	
Y	129	81	
▒	130–255	82–FF	

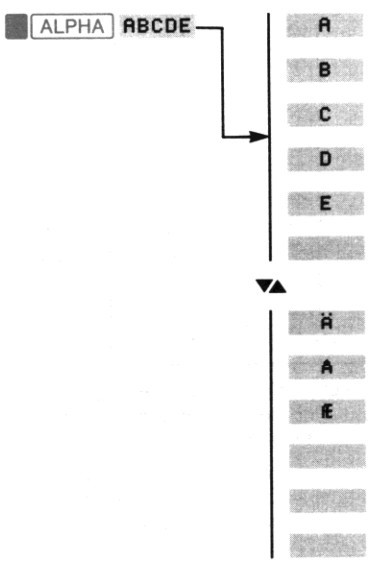
* If a character is not typable (no keystrokes) you can enter it into the Alpha register by keying the character code into the X-register and then executing the XTOA function.

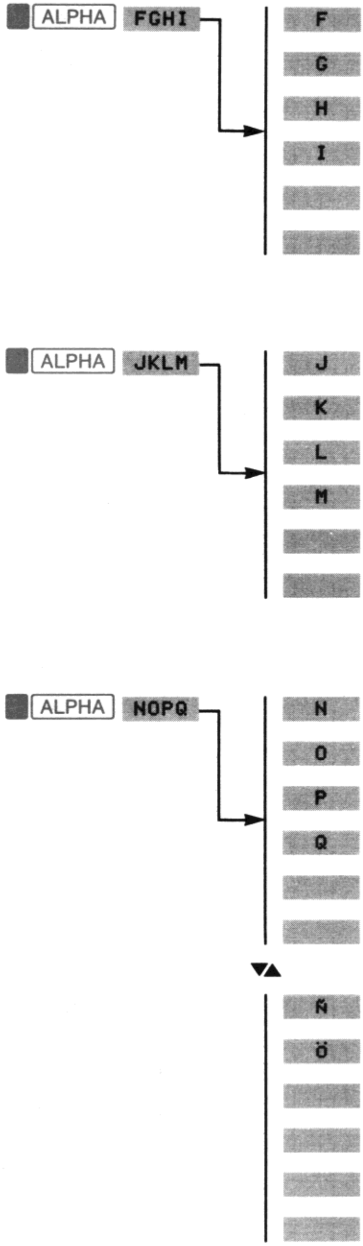
† The append character ⊢ cannot be typed directly into the Alpha register. However, in Program-entry mode this character can be entered to specify an *appended* Alpha string: press ■ ALPHA ENTER (page 130).

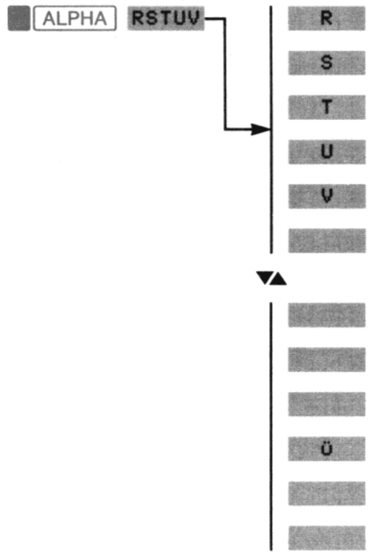
Menu Maps

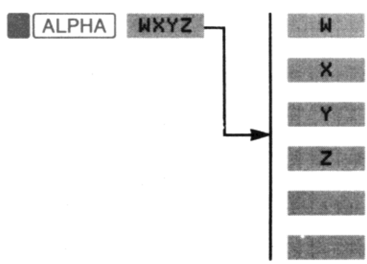
The ALPHA Submenus

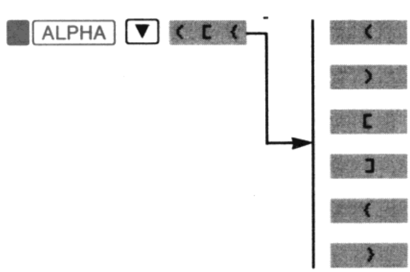
The following submenus are all part of the ALPHA menu. Refer to the menu map on page 38 for a more general view of the ALPHA menu.

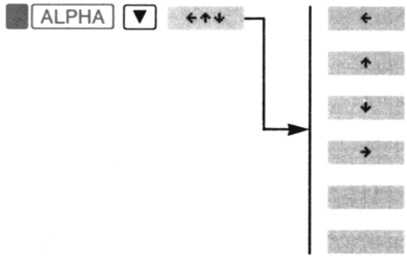


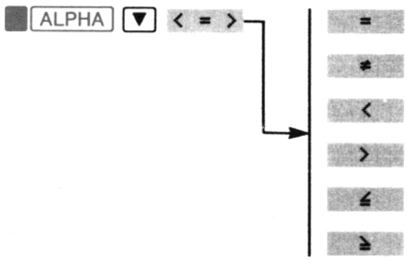

ALPHA
FGHI
F
G
H
I
ALPHA
JKLM
J
K
L
M
ALPHA
NOPQ
N
O
P
Q
Ñ
Ö

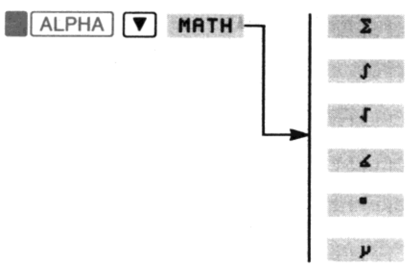

ALPHA
RSTUV
R
S
T
U
V
Ü


ALPHA
WXYZ
W
X
Y
Z


ALPHA
([{
(
)
[
]
{
}


ALPHA
←↑↓
←
↑
↓
→


ALPHA
< = >
=
≠
<
>
≤
≥


ALPHA
MATH
Σ
∫
∡
μ

■ ALPHA ▼ PUNC

- ,
- ;
- :
- !
- ?
- "

▼▲

- '
- _
- `
- ´
- ¿
- ␊

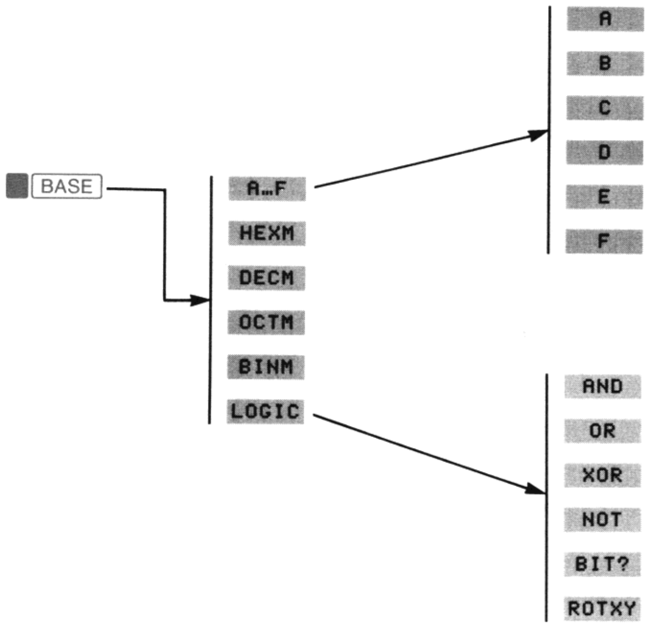
■ ALPHA ▼ MISC

- $
- *
- #
- /
- █
- (blank)

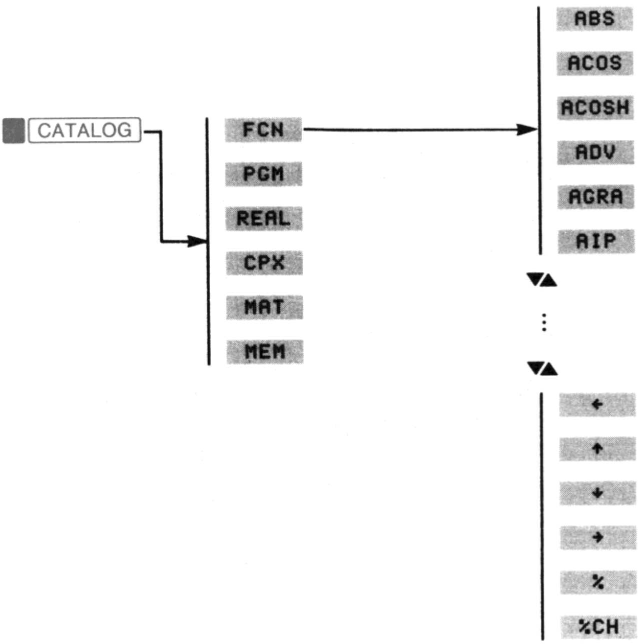
▼▲

- £
- &
- @
- \
- ~
- |

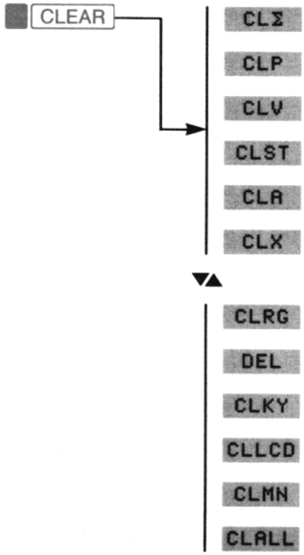
The BASE Menu



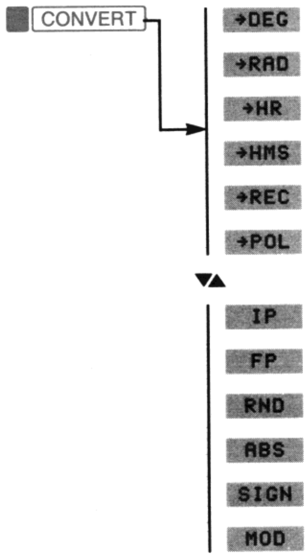
The CATALOG Menu



The CLEAR Menu

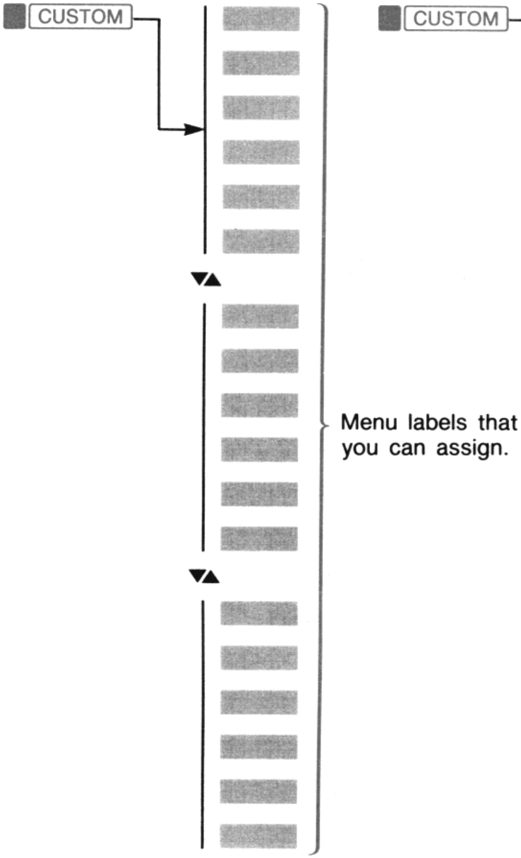


The CONVERT Menu

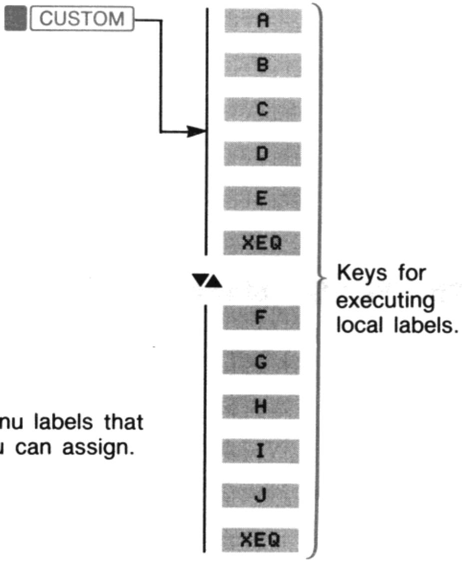


The CUSTOM Menu

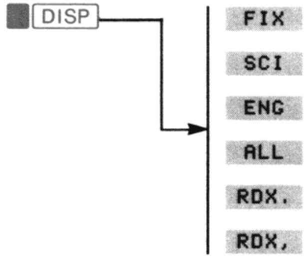
In Key-Assignments Mode



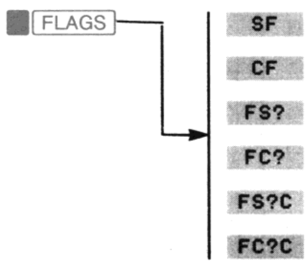
In Local-Label Mode



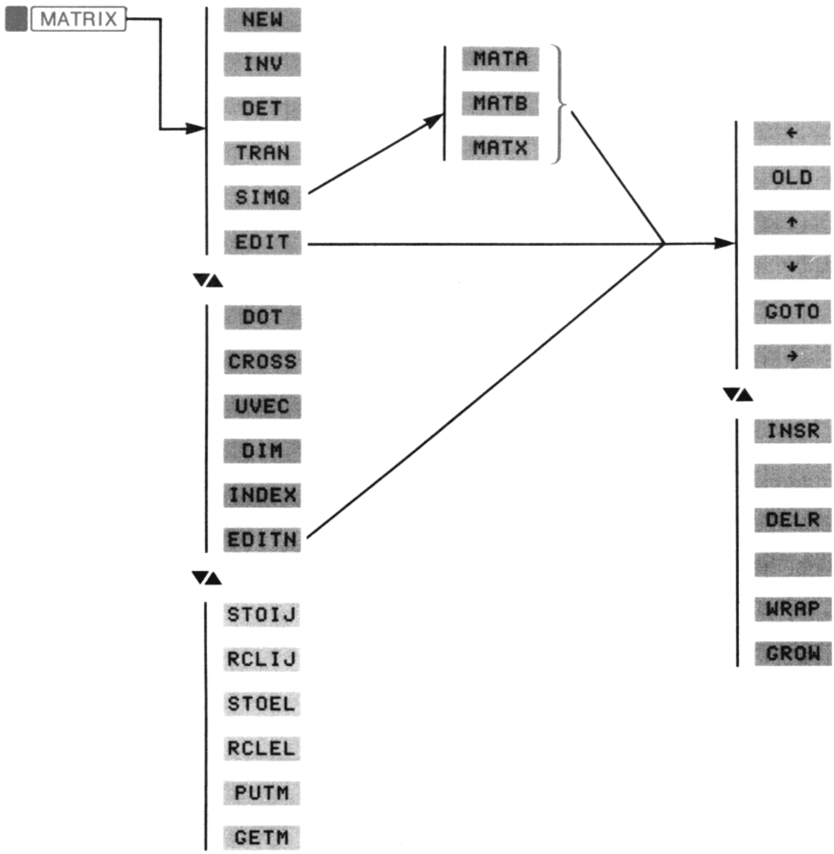
The DISP Menu



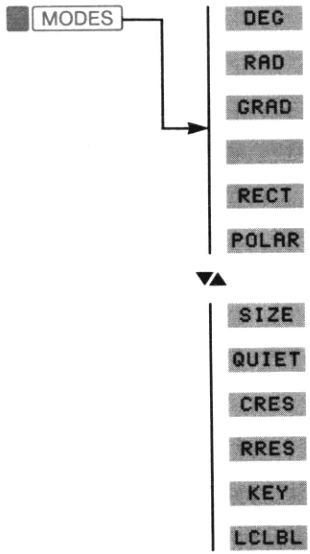
The FLAGS Menu



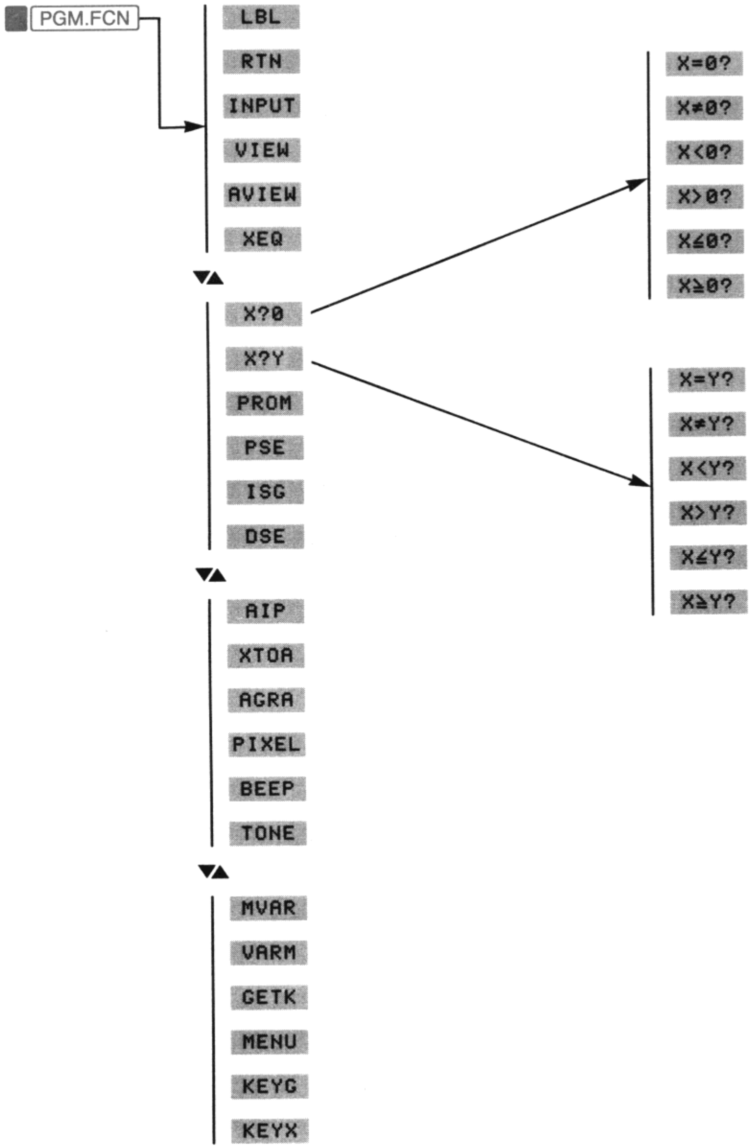
The MATRIX Menu



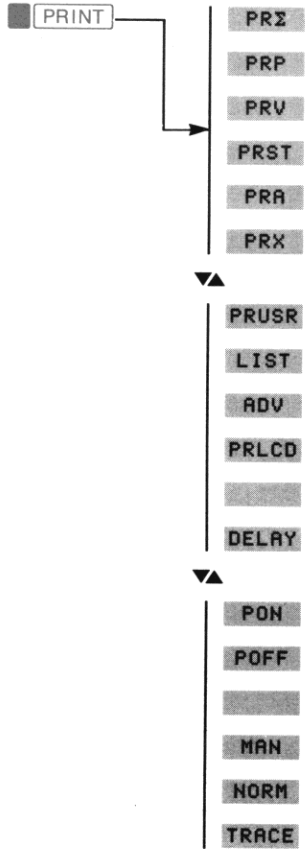
The MODES Menu



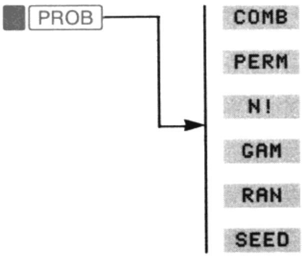
The PGM.FCN Menu



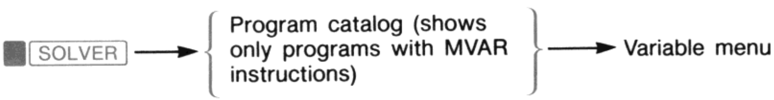
The PRINT Menu



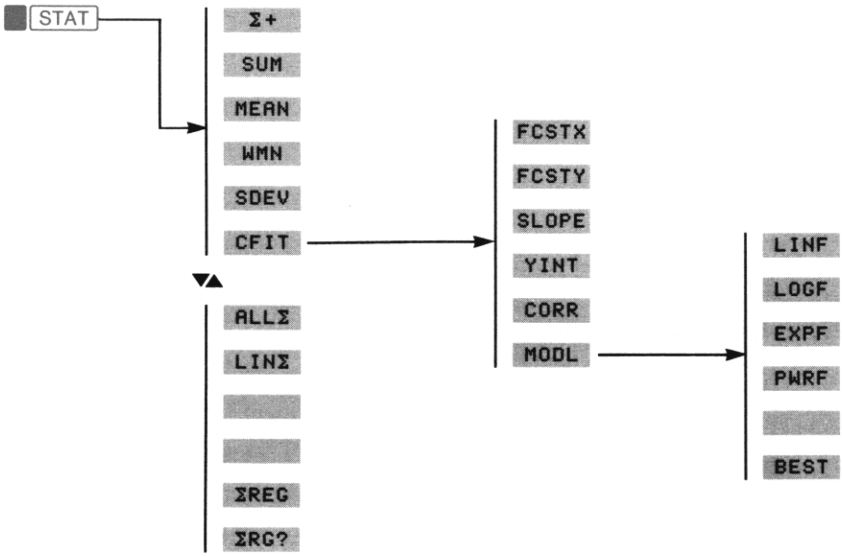
The PROB Menu



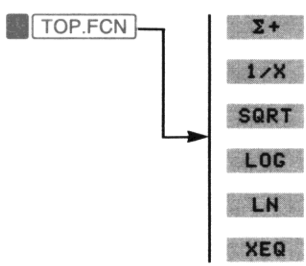
The SOLVER Menu



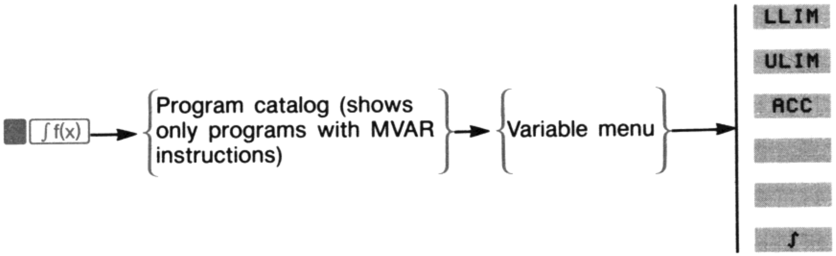
The STAT Menu



The TOP.FCN Menu



The ∫f(x) Menu


∫ f(x)
Program catalog (shows only programs with MVAR instructions)
Variable menu
LLIM
ULIM
ACC
∫

Operation Index

This index contains basic information and references for all HP-42S functions and keys.

Function Names. The entries in this index are listed alphabetically (with special characters at the end). This is the same order used in the function catalog.

Note that this index uses the *full Alpha name* for each function. Because menu labels are limited to five characters (or fewer), some function names are abbreviated when they appear in a menu label.

Keystrokes. Keystrokes are included for functions that are on the keyboard or in menus. If no keystrokes are shown for a particular function, use the function catalog (■ [CATALOG] FCN) or [XEQ] to execute the function (page 67).

Parameters. Parameters are described for those functions that require a parameter. The entry also indicates if the parameter can be specified using indirect addressing.

Name	Description, Keys, and Parameters	Page
ABS	*Absolute value*. Returns |x|. Keys: ■ [CONVERT] [▼] ABS	86
ACOS	*Arc cosine*. Returns $\cos^{-1} x$. Keys: ■ [ACOS]	82
ACOSH	*Arc hyperbolic cosine*. Returns $\cosh^{-1} x$.	89
ADV	*Advances* the printer paper one line. Keys: ■ [PRINT] [▼] ADV	101

Name	Description, Keys, and Parameters	Page
AGRAPH	*Alpha graphics*. Displays a graphics image. Each character in the Alpha register specifies an 8-dot column pattern. The X- and Y-registers specify the pixel location of the image. Keys: ■ [PGM.FCN] [▼] [▼] AGRA	136
AIP	*Appends integer part* of *x* to the Alpha register. Keys: ■ [PGM.FCN] [▼] [▼] AIP	133
ALENG	*Alpha length*. Returns the number of characters in the Alpha register.	135
ALL	Selects the *All* display format. Keys: ■ [DISP] ALL	36
ALLΣ	Selects *All*Σ (*All-statistics*) mode, which uses 13 summation coefficients. Keys: ■ [STAT] [▼] ALLΣ	233
■ [ALPHA]	Selects the ALPHA menu for typing characters.	37
AND	Logical *AND*. Returns *x* AND *y*. Keys: ■ [BASE] LOGIC AND	250
AOFF	*Alpha off*. Exits from the ALPHA menu.	157
AON	*Alpha on*. Selects the ALPHA menu.	156
ARCL	*Alpha recall*. Copies data into the Alpha register appending it to the current contents. Numbers are formatted using the current display format. Key: [RCL] *(when Alpha mode is on)* Parameter: register or variable Indirect: Yes	133
AROT	*Alpha rotate*. Rotates the Alpha register by the number of characters specified in the X-register.	135
ASHF	*Alpha shift*. Shifts the six left-most characters out of the Alpha register.	135
ASIN	*Arc sine*. Returns $\sin^{-1} x$. Keys: ■ [ASIN]	82

Name	Description, Keys, and Parameters	Page
ASINH	*Arc hyperbolic sine*. Returns $\sinh^{-1} x$.	89
ASSIGN	*Assigns* a function, program, or variable to a menu key in the CUSTOM menu. Keys: ■ [ASSIGN] Parameters: refer to the table on page 72.	68
ASTO	*Alpha store*. Copies the first six characters in the Alpha register into a register or variable. Key: [STO] *(when Alpha mode is on)* Parameter: register or variable Indirect: Yes	132
ATAN	*Arc tangent*. Returns $\tan^{-1} x$. Keys: ■ [ATAN]	82
ATANH	*Arc hyperbolic tangent*. Returns $\tanh^{-1} x$.	89
ATOX	*Alpha to X*. Converts the left-most character in the Alpha register to its character code (returned to the X-register) and deletes the character.	134
AVIEW	*Alpha view*. Displays the Alpha register. Keys: ■ [PGM.FCN] AVIEW	129
■ [BASE]	Selects the BASE menu.	245
BASE+	*Base addition*. Returns the 36-bit sum of $y + x$. Key: ■ [BASE] [+]	249
BASE−	*Base subtraction*. Returns the 36-bit difference of $y - x$. Key: ■ [BASE] [−]	249
BASE×	*Base multiplication*. Returns the 36-bit product of $y \times x$. Key: ■ [BASE] [×]	249
BASE÷	*Base division*. Returns the 36-bit quotient of $y \div x$. Key: ■ [BASE] [÷]	249

Name	Description, Keys, and Parameters	Page
BASE+/−	*Base change sign*. Returns the 36-bit 2's complement of *x*. Key: ■ [BASE] [+/−]	249
BEEP	Sounds a sequence of four tones. Keys: ■ [PGM.FCN] [▼] [▼] BEEP	24
BEST	Selects the *best* curve-fitting model for the current statistical data. Keys: ■ [STAT] CFIT MODL BEST	240
BINM	Selects *Binary mode* (base 2). Keys: ■ [BASE] BINM	245
BIT?	Tests the x^{th} bit of *y*. If the bit is set (1), executes the next program line; if the bit is clear (0), skips the next program line. Keys: ■ [BASE] LOGIC BIT?	250
BST	*Back step*. Moves the program pointer to the previous program line. (Not programmable.) Keys: ■ [BST] (or [▲] if no menu is displayed)	111
CF	*Clears flag nn* (00 ⩽ *nn* ⩽ 35; 81 ⩽ *nn* ⩽ 99). Keys: ■ [FLAGS] CF Parameter: flag number Indirect: Yes	41
■ [CATALOG]	Selects the CATALOG menu.	40
CLA	*Clears Alpha register*. If Alpha mode is on and character entry is terminated (no cursor displayed), then [◆] also executes the CLA function. Keys: ■ [CLEAR] CLA	26
CLALL	*Clear all*. Clears all stored programs and data. (Not Programmable.) Keys: ■ [CLEAR] [▼] CLALL YES	26
CLD	*Clear display*. Clears a message from the display.	26
■ [CLEAR]	Selects the CLEAR menu.	26

Name	Description, Keys, and Parameters	Page
CLKEYS	Clears all CUSTOM menu assignments. Keys: ■ [CLEAR] [▼] CLKY	70
CLLCD	*Clear LCD (liquid crystal display)*. Blanks the entire display. Keys: ■ [CLEAR] [▼] CLLCD	136
CLMENU	*Clear MENU*. Deletes all menu key definitions for the programmable menu. Keys: ■ [CLEAR] [▼] CLMN	146
CLP	Clears a program from memory. Keys: ■ [CLEAR] CLP Parameter: global label Indirect: No	119
CLRG	Clears all of the numbered storage registers to zero. Keys: ■ [CLEAR] [▼] CLRG	64
CLST	Clears the stack registers to zero. Keys: ■ [CLEAR] CLST	43
CLV	Clears a variable from memory. Keys: ■ [CLEAR] CLV Parameter: variable name Indirect: Yes	62
CLX	Clears X-register to zero. If digit entry is terminated (no cursor in the display), [◄] also executes CLX. Keys: ■ [CLEAR] CLX	48
CLΣ	*Clear statistics*. Clears the accumulated statistical data in the summation registers. Keys: ■ [CLEAR] CLΣ	228
COMB	*Combinations* of y items taken x at a time. Returns $y! \div (x!(y - x)!)$. Keys: ■ [PROB] COMB	87

Name	Description, Keys, and Parameters	Page
COMPLEX	Converts two real numbers (or matrices) into a complex number (or matrix). Converts a complex number (or matrix) into two real numbers (or matrices). Keys: ■ [COMPLEX]	91
■ [CONVERT]	Selects the CONVERT menu.	82
CORR	Returns a *correlation coefficient* using the current statistical data and curve-fitting model. Keys: ■ [STAT] CFIT CORR	240
COS	*Cosine*. Returns cos *x*. Key: [COS]	81
COSH	*Hyperbolic cosine*. Returns cosh *x*.	89
CPXRES	*Complex-results.* Enables the calculator to return a complex result, even if the inputs are real numbers. Keys: ■ [MODES] [▼] CRES	94
CPX?	If the X-register contains a complex number, executes the next program line; if the X-register does not contain a complex number, skips the next program line.	151
CROSS	Returns the *cross product* of two vectors (matrices or complex numbers). Keys: ■ [MATRIX] [▼] CROSS	220
■ [CUSTOM]	Selects the CUSTOM menu.	68
DECM	Selects *Decimal mode* (base 10). Keys: ■ [BASE] DECM	245
DEG	Selects the *Degrees* angular mode. Keys: ■ [MODES] DEG	80

Name	Description, Keys, and Parameters	Page
DEL	*Deletes* the specified number of lines from the current program. Program-entry mode must be on. (Not programmable.) Keys: ■ [CLEAR] [▼] DEL Parameter: number of lines Indirect: No	120
DELAY	Sets the print *delay* time to *x* seconds. Keys: ■ [PRINT] [▼] DELAY	103
DELR	*Delete row*. Deletes the current row from the indexed matrix. Keys: ■ [MATRIX] EDIT [▼] DELR	214
DET	Returns the *determinant* of the matrix in the X-register. Keys: ■ [MATRIX] DET	219
DIM	*Dimensions* a matrix to *x* columns and *y* rows. If the matrix does not exist, DIM creates it. Keys: ■ [MATRIX] [▼] DIM Parameter: variable name Indirect: Yes	217
DIM?	Returns the *dimensions of* the matrix in the X-register (*rows* to the Y-register and *columns* to the X-register).	217
■ [DISP]	Selects the DISP menu.	34
DOT	Returns the *dot product* of two vectors (matrices or complex numbers). Keys: ■ [MATRIX] [▼] DOT	220
DSE	*Decrement, Skip if* (less than or) *Equal*. Given *ccccccc.fffii* in a variable or register, decrements *ccccccc* by *ii* and skips the next program line if *ccccccc* is now ≤ *fff*. Keys: ■ [PGM.FCN] [▼] DSE Parameter: register or variable Indirect: Yes	153
[E]	*Enter exponent.* Adds "E" to the number being entered. Indicates that a power of ten follows.	27

Name	Description, Keys, and Parameters	Page
EDIT	*Edit* a matrix in the X-register. Keys: ■ [MATRIX] **EDIT**	206
EDITN	*Edit* a *named* matrix. Keys: ■ [MATRIX] [▼] **EDITN** Parameter: variable name Indirect: Yes	208
END	*End* of a program.	118
ENG	Selects *Engineering* display format. Keys: ■ [DISP] **ENG** Parameter: number of digits Indirect: Yes	36
ENTER	Separates two numbers keyed in sequentially; copies *x* into the Y-register, *y* into the Z-register, and *z* into the T-register, and loses *t*. Key: [ENTER]	46
[EXIT]	Exits the current menu. (Not programmable.)	23
EXITALL	*Exits all* menus.	
EXPF	Selects the *exponential* curve-fitting model. Keys: ■ [STAT] **CFIT** **MODL** **EXPF**	240
E↑X	*Natural exponential*. Returns e^x. Keys: ■ [e^x]	78
E↑X-1	*Natural exponential* for values of *x* which are close to zero. Returns e^x-1, which provides a much higher accuracy in the fractional part of the result.	
FC?	If the specified flag is clear, executes the next program line; if the flag is set, skips the next program line. Keys: ■ [FLAGS] **FC?** Parameter: flag number Indirect: Yes	41

Name	Description, Keys, and Parameters	Page
FC?C	If the specified flag is clear, executes the next program line; if the flag is set, skips the next program line. Cleared after the test is complete. (This function can be used only with flags 00 through 35 and 81 through 99.) Keys: ■ [FLAGS] **FC?C** Parameter: flag number Indirect: Yes	41
FCSTX	*Forecasts* an *x*-value given a *y*-value. Keys: ■ [STAT] **CFIT** **FCSTX**	240
FCSTY	*Forecasts* a *y*-value given an *x*-value. Keys: ■ [STAT] **CFIT** **FCSTY**	240
FIX	Selects *Fixed-decimal* display format. Keys: ■ [DISP] **FIX** Parameter: number of digits Indirect: Yes	35
■ [FLAGS]	Selects the FLAGS menu.	41
FNRM	Returns the *Frobenius norm* of the matrix in the X-register.	219
FP	Returns the *fractional part* of *x*. Keys: ■ [CONVERT] [▼] **FP**	86
FS?	If the specified flag is set, executes the next program line; if the flag is clear, skips the next program line. Keys: ■ [FLAGS] **FS?** Parameter: flag number Indirect: Yes	41
FS?C	If the specified flag is set, executes the next program line; if the flag is clear, skips the next program line. Clears the flag after the test is complete. (This function can be used only with flags 00 through 35 and 81 through 99.) Keys: ■ [FLAGS] **FS?C** Parameter: flag number Indirect: Yes	41

Name	Description, Keys, and Parameters	Page
GAMMA	*Gamma function*. Returns Γ(*x*). Keys: ■ [PROB] GAM	88
GETKEY	*Get key*. The calculator waits for you to press a key. When you do, the key number is returned to the X-register. Keys are numbered from 1 through 37 ([Σ+] through [+]) for normal keys and 38 through 74 (■ [Σ−] through ■ [CATALOG]) for shifted keys. Keys: ■ [PGM.FCN] [▲] GETK	
GETM	*Get matrix*. Copies a submatrix into the X-register from the indexed matrix. Keys: ■ [MATRIX] [▲] GETM	226
GRAD	Selects *Grads* angular mode. Keys: ■ [MODES] GRAD	80
GROW	Selects *Grow* mode. Executing → or J+ causes the matrix to grow by one new row if the index pointers are at the last (lower-right) element in the matrix. Keys: ■ [MATRIX] EDIT [▼] GROW	213
GTO	*Go to* label. From the keyboard, moves the program pointer to the specified label. In a running program, causes the program to branch to the specified label. Keys: ■ [GTO] Parameter: local or global label Indirect: Yes	141
■ [GTO] [·]	Moves the program pointer to a line number or global label. (Not programmable.)	111
■ [GTO] [·] [·]	Moves the program pointer to a new program space. (Not programmable.)	118
HEXM	Selects *Hexadecimal mode* (base 16). Keys: ■ [BASE] HEXM	245

Name	Description, Keys, and Parameters	Page
HMS+	Adds *x* and *y* using *H.MMSSss* (hours-minutes-seconds) format.	84
HMS−	Subtracts *x* from *y* using *H.MMSSss* format.	84
I+	Increments the row pointer in the indexed matrix.	224
I−	Decrements the row pointer in the indexed matrix.	224
INDEX	*Indexes* a named matrix. Keys: ■ [MATRIX] [▼] **INDEX** Parameter: variable name Indirect: Yes	223
INPUT	Recalls a register or variable to the X-register, displays the name of the register or variable along with the contents of the X-register, and halts program execution; pressing [R/S] (or ■ [SST]) stores *x* into the register or variable; pressing [EXIT] cancels. (Used only in programs.) Keys: ■ [PGM.FCN] **INPUT** Parameter: register or variable Indirect: Yes	121
INSR	*Inserts a row* in the indexed matrix. Keys: ■ [MATRIX] **EDIT** [▼] **INSR**	214
INTEG	*Integrates* the selected integration program with respect to the specified variable. Parameter: variable name Indirect: Yes	203
INVRT	Returns the *inverse* of the matrix in the X-register. Keys: ■ [MATRIX] **INV**	219
IP	Returns the *integer part* of *x*. Keys: ■ [CONVERT] [▼] **IP**	86
ISG	*Increment, Skip if Greater*. Given *ccccccc.fffii* in a variable or register, increments *ccccccc* by *ii* and skips the next program line if *ccccccc* is now > *fff*. Keys: ■ [PGM.FCN] [▼] **ISG** Parameter: register or variable Indirect: Yes	153

Name	Description, Keys, and Parameters	Page
J+	Increments the column pointer in the indexed matrix.	224
J−	Decrements the column pointer in the indexed matrix.	224
KEYASN	Selects *Key-assignments* mode for the CUSTOM menu. Keys: ■ [MODES] [▼] KEY	167
KEYG	On menu *key*, *go to*. Defines the label to be branched to when a particular menu key is pressed. Keys: ■ [PGM.FCN] [▲] KEYG Parameters: refer to the table on page 72.	145
KEYX	On menu *key*, *execute*. Defines the label to be executed (as a subroutine) when a particular menu key is pressed. Keys: ■ [PGM.FCN] [▲] KEYX Parameters: refer to the table on page 72.	145
LASTX	*Last x*. Recalls the last value of *x* used in a calculation. Keys: ■ [LASTx]	48
LBL	*Label*. Identifies programs and routines for execution and branching. Keys: ■ [PGM.FCN] LBL Parameter: local or global label Indirect: No	116
LCLBL	Selects *Local label* mode for the CUSTOM menu. Keys: ■ [MODES] [▼] LCLBL	167
LINF	Selects the *linear* curve-fitting model. Keys: ■ [STAT] CFIT MODL LINF	240
LINΣ	Selects *Linear statistics* mode, which uses six summation coefficients. Keys: ■ [STAT] [▼] LINΣ	233

Name	Description, Keys, and Parameters	Page
LIST	Prints a portion of a program listing. (Not programmable.) Keys: ■ [PRINT] [▼] LIST Parameter: number of lines Indirect: No	105
LN	*Natural logarithm*. Returns ln x. Key: [LN]	78
LN1+X	Natural logarithm for values close to zero. Returns ln(1 + x), which provides a much higher accuracy in the fractional part of the result.	
LOG	*Common logarithm*. Returns $\log_{10} x$. Key: [LOG]	78
LOGF	Selects the *logarithmic* curve-fitting model. Keys: ■ [STAT] CFIT MODL LOGF	240
MAN	Selects *Manual* print mode. Keys: ■ [PRINT] [▲] MAN	102
MAT?	If the X-register contains a matrix, executes the next program line; if the X-register does not contain a matrix, skips the next program line.	151
MEAN	*Mean*. Returns the mean of x-values ($\Sigma x \div n$) and the mean of y-values ($\Sigma y \div n$). Keys: ■ [STAT] MEAN	231
MENU	Selects the programmable menu. Keys: ■ [PGM.FCN] [▲] MENU	146
MOD	*Modulo*. Returns the remainder for $y \div x$. Keys: ■ [CONVERT] [▼] MOD	87
MVAR	Declares a *menu variable*. Keys: ■ [PGM.FCN] [▼] MVAR Parameter: variable name Indirect: No	125

Name	Description, Keys, and Parameters	Page
N!	*Factorial*. Returns *x*!. Keys: ■ [PROB] N!	87
NEWMAT	*New matrix*. Creates a *y* × *x* matrix in the X-register. Keys: ■ [MATRIX] NEW	206
NORM	Selects *Normal* print mode, which prints a record of keystrokes. Keys: ■ [PRINT] [▲] NORM	102
NOT	Logical *NOT*. Returns NOT *x*. Keys: ■ [BASE] LOGIC NOT	250
OCTM	Selects *Octal mode* (base 8). Keys: ■ [BASE] OCTM	245
■ [OFF]	Turns the calculator off. (Not programmable.)	18
OFF	Turns the calculator off (programmable). (Pressing ■ [OFF] does not execute the programmable OFF function.)	
OLD	Recalls the current element from the indexed matrix. (Equivalent to RCLEL.)	213
ON	*Continuous on*. Prevents the calculator from automatically turning off after ten minutes of inactivity.	
OR	Logical *OR*. Returns *x* OR *y*. Keys: ■ [BASE] LOGIC OR	250
■ [PGM.FCN]	Selects the PGM.FCN (*programming functions*) menu.	24
PERM	*Permutations* of *y* items taken *x* at a time. Returns *y*! ÷ (*y* − *x*)!. Keys: ■ [PROB] PERM	87
PGMINT	Selects a *program to integrate*. Keys: ■ [∫f(x)] PINT *(in Program-entry mode)* Parameter: global label Indirect: Yes	203

Name	Description, Keys, and Parameters	Page
PGMSLV	Selects a *program to solve*. Keys: ■ [SOLVER] PSLV *(in Program-entry mode)* *Parameter: global label* *Indirect: Yes*	189
PI	Recalls an approximation of π into the X-register (3.14159265359). Keys: ■ [π]	117
PIXEL	Turns on a single pixel (dot) in the display. The location of the pixel is given by the numbers in the X- and Y-registers. Keys: ■ [PGM.FCN] [▼] [▼] PIXEL	135
POLAR	Selects *Polar* coordinate mode for displaying complex numbers. Keys: ■ [MODES] POLAR	80
POSA	*Position in Alpha*. Searches the Alpha register for the target specified in the X-register. If found, returns the character position; if not found, returns −1.	134
PRA	*Print Alpha register.* Keys: ■ [PRINT] PRA	102
PRLCD	*Print LCD (liquid crystal display)*. Prints the entire display. Keys: ■ [PRINT] [▼] PRLCD	101
■ [PRGM]	Toggles the calculator in and out of *Program-entry mode*.	111
■ [PRINT]	Selects the PRINT menu.	101
■ [PROB]	Selects the PROB (*probability*) menu.	87
PROFF	*Printing off*. Clears flags 21 and 55. Keys: ■ [PRINT] [▲] POFF	101

Name	Description, Keys, and Parameters	Page
PROMPT	Displays the Alpha register and halts program execution. Keys: ■ [PGM.FCN] [▼] PROM	129
PRON	*Printing on*. Sets flags 21 and 55. Keys: ■ [PRINT] [▲] PON	101
PRP	*Print program*. If a label is not specified, prints the current program. (Not programmable.) Keys: ■ [PRINT] PRP Parameter: global label (optional) Indirect: No	104
PRSTK	*Print stack*. Prints the contents of the stack registers (X, Y, Z, and T). Keys: ■ [PRINT] PRST	101
PRUSR	*Prints user* variables and programs. Keys: ■ [PRINT] [▼] PRUSR	101
PRV	*Print variable*. Keys: ■ [PRINT] PRV Parameter: variable name Indirect: Yes	63
PRX	*Print X-register*. Keys: ■ [PRINT] PRX	101
PRΣ	*Print statistics*. Prints the contents of the summation registers. Keys: ■ [PRINT] PRΣ	237
PSE	*Pauses* program execution for about 1 second. Keys: ■ [PGM.FCN] [▼] PSE	131
PUTM	*Put matrix*. Stores the matrix in the X-register into the indexed matrix beginning at the current element. Keys: ■ [MATRIX] [▲] PUTM	226
PWRF	Selects the *power* curve-fitting model. Keys: ■ [STAT] CFIT MODL PWRF	240

Name	Description, Keys, and Parameters	Page
QUIET	Toggles flag 26 to disable/enable the beeper. (Not programmable.) Keys: ■ [MODES] [▼] QUIET	275
RAD	Selects *Radians* angular mode. Keys: ■ [MODES] RAD	80
RAN	Returns a *random* number ($0 \leqslant x < 1$). Keys: ■ [PROB] RAN	88
RCL	*Recalls* data into the X-register. Key: [RCL] Parameter: register or variable Indirect: Yes	55
RCL+	*Recall addition*. Recalls data and adds it to the contents of the X-register. Keys: [RCL] [+] Parameter: register or variable Indirect: Yes	61
RCL−	*Recall subtraction*. Recalls data and subtracts it from the contents of the X-register. Keys: [RCL] [−] Parameter: register or variable Indirect: Yes	61
RCL×	*Recall multiplication*. Recalls data and multiplies it by the contents of the X-register. Keys: [RCL] [×] Parameter: register or variable Indirect: Yes	61
RCL÷	*Recall division*. Recalls data and divides it into the contents of the X-register. Keys: [RCL] [÷] Parameter: register or variable Indirect: Yes	61
RCLEL	*Recall element*. Recalls the current matrix element from the indexed matrix. Keys: ■ [MATRIX] [▲] RCLEL	225

Name	Description, Keys, and Parameters	Page
RCLIJ	Recalls the row- and column-pointer values (*I* and *J*) for the indexed matrix. Keys: ■ [MATRIX] [▲] RCLIJ	224
RDX,	Selects a *comma* to be used as the radix mark (decimal point). Keys: ■ [DISP] RDX,	36
RDX.	Selects a *period* to be used as the radix mark (decimal point). Keys: ■ [DISP] RDX.	36
REALRES	*Real-results*. Disables the calculator's ability to return a complex result using real-number inputs. Keys: ■ [MODES] [▼] RRES	94
REAL?	If the X-register contains a real number, executes the next program line; if the X-register does not contain a real number, skips the next program line.	151
RECT	Selects *Rectangular* coordinate mode for displaying complex numbers. Keys: ■ [MODES] RECT	80
RND	*Rounds* the number in the X-register using the current display format. Keys: ■ [CONVERT] [▼] RND	86
RNRM	Returns the *row norm* of the matrix in the X-register.	219
ROTXY	*Rotates* the 36-bit number in the Y-register by *x* bits. Keys: ■ [BASE] LOGIC ROTXY	250
RSUM	Returns the *row sum* of each row of the matrix in the X-register and returns the sums in a column matrix.	220

Name	Description, Keys, and Parameters	Page
RTN	*Return*. In a running program, branches the program pointer back to the line following the most recent XEQ instruction. If there is no matching XEQ instruction, program execution halts. From the keyboard, RTN moves the program pointer to line 00 of the current program. Keys: ■ [PGM.FCN] RTN	143
R<>R	*Row swap row.* Swaps the elements in rows *x* and *y* in the indexed matrix.	225
R↑	*Rolls* the contents of the four stack registers *up* one position.	
R↓	*Rolls* the contents of the four stack registers *down* one position. Key: [R↓]	44
[R/S]	*Run/stop*. Runs a program (beginning at the current program line) or stops a running program. In Program-entry mode, inserts a STOP instruction into the program.	113
SCI	Selects *Scientific* display format. Keys: ■ [DISP] SCI Parameter: number of digits Indirect: Yes	35
SDEV	*Standard deviation*. Returns s_x and s_y using the current statistical data. Keys: ■ [STAT] SDEV	232
SEED	Stores a *seed* for the random number generator. Keys: ■ [PROB] SEED	88
SF	*Sets flag nn* (00 ⩽ *nn* ⩽ 35; 81 ⩽ *nn* ⩽ 99). Keys: ■ [FLAGS] SF Parameter: flag number Indirect: Yes	41
■ [SHOW]	*Shows* full precision of the number in the X-register, the entire Alpha register, or a complete program line.	36

Name	Description, Keys, and Parameters	Page
SIGN	*Sign*. Returns 1 for $x \geqslant 0$, -1 for $x < 0$, and 0 for non-numbers. Returns the unit vector of a complex number. Keys: ■ [CONVERT] [▼] SIGN	86
SIN	*Sine*. Returns sin *x*. Key: [SIN]	80
SINH	*Hyperbolic sine*. Returns sinh *x*.	89
SIZE	Sets the number of storage registers. Keys: ■ [MODES] [▼] SIZE Parameter: number of registers Indirect: No	64
SLOPE	Returns the *slope* of the linear transformation of the current curve-fitting model. Keys: ■ [STAT] CFIT SLOPE	240
SOLVE	*Solves* for an unknown variable. Keys: ■ [SOLVER] SOLVE *(in Program-entry mode)* Parameter: variable name Indirect: Yes	189
■ [SOLVER]	Selects the SOLVER menu.	178
SQRT	*Square root*. Returns $\sqrt{x}$. Key: [$\sqrt{x}$]	78
SST	*Single step*. Moves the program pointer to the next program line. (Not programmable.) Keys: ■ [SST] (or [▼] if no menu is displayed)	114
■ [STAT]	Selects the STAT (*statistics*) menu.	231
STO	*Stores* a copy of *x* into a destination register or variable. Key: [STO] Parameter: register or variable Indirect: Yes	55
STO+	*Store addition*. Adds *x* to an existing register or variable. Keys: [STO] [+] Parameter: register or variable Indirect: Yes	61

Name	Description, Keys, and Parameters	Page
STO−	*Store subtraction*. Subtracts x from an existing register or variable. Keys: [STO] [−] Parameter: register or variable Indirect: Yes	61
STO×	*Store multiplication*. Multiplies an existing register or variable by x. Keys: [STO] [×] Parameter: register or variable Indirect: Yes	61
STO÷	*Store division*. Divides an existing register or variable by x. Keys: [STO] [÷] Parameter: register or variable Indirect: Yes	61
STOEL	*Store element*. Stores a copy of x into the current element of the indexed matrix. Keys: ■ [MATRIX] [▲] STOEL	225
STOIJ	Moves the row- and column-pointers to $I = x$ and $J = y$ in the indexed matrix. Keys: ■ [MATRIX] [▲] STOIJ	224
STOP	*Stops* program execution. Key: [R/S] *(in Program-entry mode)*	114
STR?	If the X-regsiter contains an Alpha string, executes the next program line; if the X-register does not contain an Alpha string, skips the next program line.	151
SUM	Returns the *sums* Σx and Σy into the X- and Y-registers ■ [STAT] SUM	231
TAN	*Tangent*. Returns tan x. Key: [TAN]	
TANH	*Hyperbolic tangent*. Returns tanh x.	89

Name	Description, Keys, and Parameters	Page
TONE	Sounds a *tone*. Keys: ■ [PGM.FCN] [▼] [▼] TONE Parameter: tone number (0–9) Indirect: Yes	144
TRACE	Selects *Trace* printing mode, which prints a record of keystrokes and results. Keys: ■ [PRINT] [▲] TRACE	102
TRANS	Returns the *transpose* of the matrix in the X-register. Keys: ■ [MATRIX] TRAN	219
UVEC	*Unit vector*. Returns the unit vector for the matrix or complex number in the X-register. Keys: ■ [MATRIX] [▼] UVEC	220
VARMENU	Creates a *variable menu* using MVAR instructions following the specified global label. Keys: ■ [PGM.FCN] [▲] VARM Parameter: global program label Indirect: Yes	125
VIEW	*Views* the contents of a register or variable. Keys: ■ [PGM.FCN] VIEW Parameter: register or variable Indirect: Yes	128
WMEAN	*Weighted mean*. Returns the mean of *x*-values weighted by the *y*-values: $\Sigma xy \div \Sigma y$. Keys: ■ [STAT] WMN	231
WRAP	Selects *Wrap mode*, which prevents the indexed matrix from growing. Keys: ■ [MATRIX] EDIT [▼] WRAP	213
X<>	Swaps the contents of the X-register with another register or variable. Parameter: register or variable Indirect: Yes	
X<>Y	Swaps the contents of the X- and Y-registers. Key: [x≷y]	44

Name	Description, Keys, and Parameters	Page
X<0?	*X less than zero* test. Keys: ■ [PGM.FCN] [▼] X?0 X<0?	151
X<Y?	*X less than y* test. Keys: ■ [PGM.FCN] [▼] X?Y X<Y?	151
X≤0?	*X less than or equal to zero* test. Keys: ■ [PGM.FCN] [▼] X?0 X≤0?	151
X≤Y?	*X less than or equal to y* test. Keys: ■ [PGM.FCN] [▼] X?Y X≤Y?	151
X=0?	*X equal to zero* test. Keys: ■ [PGM.FCN] [▼] X?0 X=0?	151
X=Y?	*X equal to y* test. Keys: ■ [PGM.FCN] [▼] X?Y X=Y?	151
X≠0?	*X not equal to zero* test. Keys: ■ [PGM.FCN] [▼] X?0 X≠0?	151
X≠Y?	*X not equal to y* test. Keys: ■ [PGM.FCN] [▼] X?Y X≠Y?	151
X>0?	*X greater than zero* test. Keys: ■ [PGM.FCN] [▼] X?0 X>0?	151
X>Y?	*X greater than y* test. Keys: ■ [PGM.FCN] [▼] X?Y X>Y?	151
X≥0?	*X greater than or equal to zero* test. Keys: ■ [PGM.FCN] [▼] X?0 X≥0?	151
X≥Y?	*X greater than or equal to y* test. Keys: ■ [PGM.FCN] [▼] X?Y X≥Y?	151

Name	Description, Keys, and Parameters	Page
XEQ	*Execute* a function or program. Key: [XEQ] Parameter: function or label Indirect: Yes	143
XOR	Logical *XOR* (*exclusive OR*). Returns x XOR y. Keys: ■ [BASE] LOGIC XOR	250
XTOA	*X to Alpha*. Appends a character (specified by the code in the X-register) to the Alpha register. If the X-register contains an Alpha string, appends the entire string. Keys: ■ [PGM.FCN] [▼] [▼] XTOA	134
X↑2	*Square*. Returns x^2. Keys: ■ [x^2]	78
YINT	*Y intercept*. Returns the y-intercept of the curve fitted to the current statistical data. Keys: ■ [STAT] CFIT YINT	240
Y↑X	*Power*. Returns y^x. Keys: ■ [y^x]	78
■ [∫f(x)]	Selects the ∫f(x) menu.	197
1/X	*Reciprocal*. Returns $1 \div x$. Key: [1/x]	78
10↑X	*Common exponential*. Returns 10^x. Keys: ■ [10^x]	78
+	*Addition*. Returns $y + x$. Key: [+]	78
−	*Subtraction*. Returns $y - x$. Key: [−]	78

Name	Description, Keys, and Parameters	Page
×	*Multiplication*. Returns $x \times y$. Key: [×]	78
÷	*Division*. Returns $y \div x$. Key: [÷]	78
+/−	Changes the sign of the number in the X-register. While entering an exponent, can also be used to change the sign of the exponent. Key: [+/-]	78
Σ+	*Summation plus*. Accumulates a pair of *x*- and *y*-values into the summation registers. Key: [Σ+]	228
Σ−	*Summation minus*. Subrtacts a pair of *x*- and *y*-values from the summation registers. Keys: ■ [Σ−]	232
ΣREG	*Summation registers*. Defines which storage register begins the block of summation registers. Keys: ■ [STAT] [▼] **ΣREG** Parameter: register number Indirect: Yes	234
ΣREG?	Returns the register number of the first summation register.	234
→DEC	*To decimal*. Converts the octal (base 8) representation of a number to decimal (base 10). Note: This function is included to provide program compatability with the HP–41 (which uses the function name DEC) and is not related to the Base application (chapter 16).	171
→DEG	*To degrees*. Converts an angle-value from radians to degrees. Returns $(360/2\pi)x$. Keys: ■ [CONVERT] **→DEG**	83
→HMS	*To hours, minutes, and seconds*. Converts *x* from a decimal fraction to a minutes-seconds format. Keys: ■ [CONVERT] **→HMS**	83

Name	Description, Keys, and Parameters	Page
→HR	*To hours*. Converts x from a minutes-seconds format to a decimal fraction.	83
→OCT	*To octal*. Converts a decimal number to the octal representation. Note: This function is included to provide program compatability with the HP–41 (which uses the function name OCT) and is not related to the Base application (chapter 16).	171
→POL	*To polar*. Converts x and y to the corresponding polar coordinates r and θ. If the X-register contains a complex number, converts the two parts of the number to polar values. Keys: ■ [CONVERT] →POL	84
→RAD	*To radians*. Converts a angle value in degrees to radians. Returns $(2\pi/360)x$. Keys: ■ [CONVERT] →RAD	83
→REC	*To rectangular*. Converts r (in the X-register) and θ (in the Y-register) to the corresponding rectangular coordinates, x and y. If the X-register contains a complex number, converts the two parts of the number to rectangular values. Keys: ■ [CONVERT] →REC	84
[◆]	Backspaces or clears X-register. In Program-entry mode, deletes the current program line.	25
←	Moves *left* one element in the indexed matrix.	212
↑	Moves *up* one element in the indexed matrix.	212
↓	Moves *down* one element in the indexed matrix.	212
→	Moves *right* one element in the indexed matrix.	212
%	*Percent*. Returns $(x \times y) \div 100$. (Leaves the y-value in the Y-register.) Keys: ■ [%]	79
%CH	*Percent change*. Returns $(x - y)(100 \div y)$.	79

Subject Index

Page numbers in **bold** type indicate primary references. *To look up functions by name, use the "Operation Index"* (pages 310 through 335). Special (nonalphabetic) characters and symbols are listed at the end of this index.

A

B

C

D

E

F

G

H

I

J

K

L

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

Special Characters

Contacting Hewlett-Packard

For Information About Using the Calculator. If you have questions about how to use the calculator, first check the table of contents, the subject index, and "Answers to Common Questions" in appendix A. If you can't find an answer in the manual, you can contact the Calculator Support department:

Hewlett-Packard
Calculator Support
1000 N.E. Circle Blvd.
Corvallis, OR 97330, U.S.A.

(503) 757-2004
8:00 a.m. to 3:00 p.m. Pacific time
Monday through Friday

For Service. If your calculator doesn't seem to work properly, refer to appendix A to determine if the calculator requires service. The appendix also contains important information about obtaining service. If your calculator requires service, mail it to the Calculator Service Center:

Hewlett-Packard
Calculator Service Center
1030 N.E. Circle Blvd.
Corvallis, OR 97330, U.S.A.
(503) 757-2002

For Information About Hewlett-Packard Dealers, Products, and Prices. Call the following toll-free number:

(800) 752-0900

Contents

HEWLETT PACKARD

Reorder Number
00042-90001

00042-90002
Printed in U.S.A. 6/88