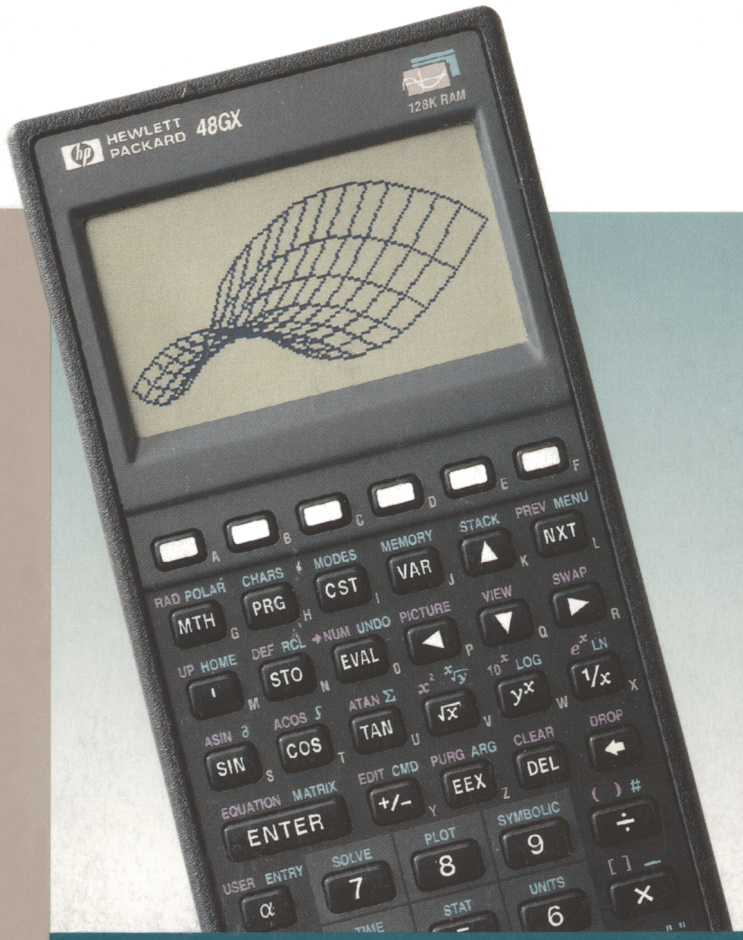
HEWLETT PACKARD

HP 48G Series
User's Guide



Regulatory Information

U.S.A.

The HP 48 generates and uses radio frequency energy and may interfere with radio and television reception. The HP 48 complies with the limits for a Class B computing device as specified in 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 unit off and on), try the following:

- Reorienting or relocating the receiving antenna.
- Relocating the HP 48 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.

Europe

Declaration of Conformity (according to ISO/IEC Guide 22 and EN 45014)

Manufacturer's name:	Hewlett-Packard Co.	Hewlett-Packard Co.
Manufacturer's address:	Corvallis Division 1000 NE Circle Blvd. Corvallis, OR 97330	Singapore (PTE) Ltd. 72 Bendemeer Rd. 01/01-07/07 Singapore 1233
	declares that the following products:	
Product name:	HP 48G Series calculators	
	conform to the following product specifications:	
EMC:	CISPR 22:1985 / EN 55022 (1988): Class B, IEC 801-2:1991 / prEN 55024-2 (1992): 3 kV CD, 8 kV AD, IEC 801-3:1984 / prEN 55024-3 (1991): 3 V/m	
Safety:	IEC 950 (1986)+A1,A2/EN 60950 (1988)+A1,A2	

Quality Department
Hewlett-Packard Company
Corvallis Division

HP 48G Series User's Guide

HP Part No. 00048-90126
Printed in Singapore

Notice

This manual and any examples contained herein are provided "as is" and are subject to change without notice. **Hewlett-Packard Company makes no warranty of any kind with regard to this manual, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose.** Hewlett-Packard Co. shall not be liable for any errors or for incidental or consequential damages in connection with the furnishing, performance, or use of this manual or the examples herein.

© Copyright Hewlett-Packard Company 1993. All rights reserved. Reproduction, adaptation, or translation of this manual is prohibited without prior written permission of Hewlett-Packard Company, except as allowed under the copyright laws.

The programs that control this product are copyrighted and all rights are reserved. Reproduction, adaptation, or translation of those programs without prior written permission of Hewlett-Packard Co. is also prohibited.

© Trustees of Columbia University in the City of New York, 1989. Permission is granted to any individual or institution to use, copy, or redistribute Kermit software so long as it is not sold for profit, provided this copyright notice is retained.

Hewlett-Packard Company
Corvallis Division
1000 N.E. Circle Blvd.
Corvallis, OR 97330, U.S.A.

Acknowledgements

Hewlett-Packard gratefully acknowledges the members of the Education Advisory Committee (Dr. Thomas Dick, Dr. Lynn Garner, Dr. John Kenelly, Dr. Don LaTorre, Dr. Jerold Mathews, and Dr. Gil Proctor) for their assistance in the development of this product. Special thanks are also due to Donald R. Asmus, Scott Burke, Bhushan Gupta and his students at the Oregon Institute of Technology, and Carla Randall and her AP Calculus students.

Edition History

Edition 1 ..... May 1993
Edition 2 ..... June 1993
Edition 3 ..... July 1993
Edition 4 ..... August 1993
Edition 5 ..... October 1993
Edition 6 ..... January 1994
Edition 7 ..... March 1994
Edition 8 ..... December 1994

Contents

1. The Keyboard and Display

Organization of the Display 1-1
The Status Area, Annunciators, and Messages . . . 1-1
The Stack 1-3
The Command Line 1-4
Menu Labels 1-4
Organization of the Keyboard 1-4
Applications and Command Menus 1-6
Cursor Keys 1-7
The CANCEL Key 1-8
Menus: Extending the Keyboard 1-8
Working with Menus 1-9

2. Entering and Editing Objects

Keying In Numbers 2-1
Keying In Characters (the Alpha Keyboard) 2-2
Keying in Special Characters 2-4
Keying In Objects with Delimiters 2-5
Using the Command Line 2-7
Accumulating Data in the Command Line 2-7
Selecting Command-Line Entry Modes 2-8
Recovering Previous Command Lines 2-10
Viewing and Editing Objects 2-10
Using the EDIT Menu 2-12

3. Stack

Using the Stack for Calculations 3-1
Making Calculations 3-1
Manipulating the Stack 3-4
Recalling the Last Arguments 3-5
Restoring the Last Stack (UNDO) 3-6
The Interactive Stack 3-6

Stack Command Menu 3-11

4. Modes
Using the MODES Application 4-1
Setting the Display Mode 4-2
Setting the Angle Mode 4-3
Setting the Coordinate Mode 4-4
Setting the Beep 4-6
Setting the Clock Display 4-6
Setting the Fraction Mark 4-6
Using System Flags 4-7
Using the Flag Browser 4-7
Using the FLAG Command Submenu 4-8
User Flags 4-9
MODES Submenus 4-9

5. Memory
HOME: Variables and Directories 5-3
Where to Store Variables 5-4
Using The Variable Browser Application 5-5
Creating New Variables 5-5
Selecting, Editing, and Recalling Variables 5-7
Copying, Moving, and Purging Variables 5-9
Determining the Size of Variables 5-10
Using Variables: The VAR Menu 5-11
Defining Variables 5-13
Evaluating Variables 5-13
Quoted Variable Names and Formal Variables . . . 5-14
Special Memory Operations 5-16
System Halt 5-16
Memory Reset 5-17
Responding to Low-Memory Conditions 5-18

6. Input Forms and Choose Lists
Input Forms 6-1
Selecting Fields in Input Forms 6-2
Entering Data in Input Forms 6-3
Selecting Options in Input Forms 6-4
Other Input Form Operations 6-4
When You Finish Entering Data into an Input Form 6-7
Input Form Commands 6-7

7. The EquationWriter
How the EquationWriter Application Is Organized . . 7-2
Constructing Equations 7-3
Entering Equations 7-3
Controlling Implicit Parentheses 7-6
EquationWriter Examples 7-8
Editing Equations 7-9
Editing with Subexpressions 7-10
Summary of EquationWriter Operations 7-13

8. The MatrixWriter
How the HP 48 Displays Arrays 8-1
Entering Arrays 8-2
Editing Arrays 8-5
MatrixWriter Operations 8-5

9. Graphic Objects
The PICTURE Environment 9-2
Using the Picture Editor 9-2
Turning Pixels On and Off 9-3
Adding Elements Using the Graphics Environment . 9-3
Editing and Erasing a Picture 9-4
Saving and Viewing Graphic Objects 9-6
Graphic Object Coordinates 9-7
Graphic Object Commands 9-8

10. Unit Objects
Overview of the Units Application 10-1
Units and Unit Objects 10-1
The UNITS Catalog Menu 10-2
Creating a Unit Object 10-3
Unit Prefixes 10-4
Converting Units 10-5
Using the UNITS Catalog Menu 10-6
Using CONVERT 10-6
Using UBASE (for SI Base Units) 10-6
Converting Angular Units 10-7
Calculating with Units 10-7
Factoring Unit Expressions 10-9
Using Unit Objects in Algebraics 10-9
Working with Temperature Units 10-10
Converting Temperature Units 10-10

Calculating with Temperature Units 10-11
Creating User-Defined Units 10-14
Additional Commands for Unit Objects 10-14

11. Using Mathematical Functions
Built-In Functions and Commands 11-1
Expressing Functions: Algebraic Syntax 11-2
Expressing Functions: Stack Syntax 11-3
Expressions and Equations 11-4
Symbolic Constants 11-4
Controlling How Symbolic Constants are Evaluated . 11-5
Using Built-In Math Functions 11-5
User-Defined Functions 11-6
Creating a User-Defined Function 11-7
Executing a User-Defined Function 11-7
Nesting User-Defined Functions 11-8

12. Functions of Real and Complex Numbers
Math Functions on the Main Keyboard 12-1
Arithmetic and General Math Functions 12-1
Exponential and Logarithmic Functions 12-2
Trigonometric Functions 12-2
Hyperbolic Functions 12-3
Probability and Test Statistics 12-4
Calculating Test Statistics 12-4
Real Number Functions 12-7
Angle Conversion Functions 12-7
Percent Functions 12-9
Other Real-Number Functions 12-9
Complex Numbers 12-11
Displaying Complex Numbers 12-11
Entering Complex Numbers 12-12
Real Calculations with Complex Results 12-13
Other Complex-Number Commands 12-13

13. Vectors and Transforms

Displaying 2D and 3D Vectors 13-1
Entering 2D and 3D Vectors 13-3
Vector Math Commands 13-4
Examples: Calculating with 2D and 3D Vectors . . . 13-5
Fast Fourier Transforms 13-7

14. Matrices and Linear Algebra

Creating and Assembling Matrices 14-1
Disassembling Matrices 14-4
Inserting Rows and Columns 14-5
Extracting Rows and Columns 14-6
Swapping Rows and Columns 14-6
Extracting and Replacing Elements of Matrices 14-7
Characterizing Matrices 14-7
Transforming Matrices 14-10
Calculating with Matrix Elements 14-11
Using Arrays and Array Elements in Algebraic Expressions 14-12
Transforming Complex Matrices 14-13
Matrix Solutions to Systems of Linear Equations . . . 14-14
Ill-Conditioned and Singular Matrices 14-16
Determining the Accuracy of a Matrix Solution 14-17
Gaussian Elimination and Elementary Row Operations 14-18
Additional Linear Algebra Topics 14-20

15. Binary Arithmetic and Number Bases

Binary Integers and Bases 15-1
Using Boolean Operators 15-4
Manipulating Bits and Bytes 15-4

16. Date, Time and Fraction Arithmetic

Calculating with Dates 16-1
Calculating with Times 16-3
Calculating with Fractions 16-5

17. Lists and Sequences
Creating Lists 17-1
List Processing 17-2
Multiple-Argument Commands with Lists 17-3
Applying a Function or Program to a List (DOLIST) 17-4
Applying a Function Recursively to a List 17-5
List Manipulations 17-6
Sequences 17-7

18. Solving Equations
Solving an Equation for an Unknown Variable 18-1
Interpreting Results 18-3
Solving Options 18-5
SOLVR: An Alternative Solving Environment 18-6
Additional Solving Options in SOLVR 18-8
Finding All Roots of a Polynomial 18-10
Solving a System of Linear Equations: 18-11
Using the Finance Solver 18-13
Calculating Amortizations 18-19

19. Differential Equations
Solving Differential Equations 19-1
Solving a Standard Initial-Value Problem 19-2
Solving a Stiff Initial-Value Problem 19-3
Solving a Vector-Value Differential Equation 19-5
Plotting Solutions to Differential Equations 19-7
Plotting a Stiff Differential Equation 19-9
Plotting a Phase Plane for a Vector-Valued Solution . 19-12

20. Calculus and Symbolic Manipulation
Integration 20-1
Numeric Integration 20-1
The Accuracy Factor and the Uncertainty of Numerical Integration 20-6
Symbolic Integration 20-7
Differentiation 20-9
Creating User-Defined Derivatives 20-11
Implicit Differentiation 20-12
Taylor Polynomial Approximation 20-12
Finding Symbolic Solutions to Equations 20-14
Isolating a Single Variable 20-15
Solving Quadratic Equations 20-15

Getting General and Principal Solutions 20-16
Showing Hidden Variables 20-17
Rearranging Symbolic Expressions 20-17
Manipulating Whole Expressions 20-17
Manipulating Subexpressions 20-19
Making User-Defined Transformations 20-27
Symbolic Integration Patterns 20-30

21. Statistics and Data Analysis
Entering Statistical Data 21-1
Editing Statistical Data 21-4
Calculating Single-Variable Statistics 21-6
Generating Frequencies 21-8
Fitting a Model to a Set of Data 21-9
Calculating Summary Statistics 21-11
Using the PAR Reserved Variable 21-12

22. Plotting
Using the PLOT Application 22-1
Cursor Coordinates: Standard and TRACE Modes . . 22-4
Keyboard Operations in the PICTURE Environment . 22-5
Using Zoom Operations 22-6
Setting Zoom Defaults 22-7
Selecting a Zoom 22-7
Analyzing Functions 22-9
Understanding the PLOT Reserved Variables 22-12
EQ 22-12
DAT 22-12
ZPAR 22-13
PPAR 22-13
VPAR 22-15
PAR 22-16

23. Plot Types
Function Plots 23-1
Polar Plots 23-4
Parametric Plots 23-7
Differential Equation Plots 23-11
Conic Plots 23-11
Truth Plots 23-14
Statistical Plots 23-17
Histogram Plots 23-18

Bar Plots 23-19
Scatter Plots 23-20
Plotting Functions of Two Variables 23-22
Sampling Grid 23-22
Output Grid 23-22
Slopefield Plots 23-25
Wireframe Plots 23-28
Pseudo-Contour Plots 23-30
Y-Slice Plots 23-32
Gridmap Plots 23-34
Parametric Surface Plots 23-35

24. Advanced Plot Options
Labeling and Locating the Axes 24-1
Plotting Programs and User-Defined Functions 24-2
Plotting Range vs. Display Range 24-3
Saving and Restoring Plots 24-6

25. The Equation Library
Solving a Problem with the Equation Library 25-1
Using the Solver 25-2
Using the Menu Keys 25-3
Browsing in the Equation Library 25-4
Viewing Equations 25-4
Viewing Variables and Selecting Units 25-5
Viewing the Picture 25-5
Using the Multiple-Equation Solver 25-6
Defining a Set of Equations 25-8
Interpreting Results from the Multiple Equation Solver 25-10
Using the Constants Library 25-12
Minehunt Game 25-14
User-Defined Units 25-15

26. Time Management
Using the Clock (Date and Time) 26-1
Setting Alarms 26-2
Responding to Alarms 26-4
Viewing and Editing Alarms 26-5

27. Transmitting and Printing Data

Transferring Data Between Two HP 48s 27-1
Printing 27-2
Setting Up the Printer 27-3
Printing Tasks 27-3
Transferring Data Between the HP 48 and a Computer 27-7
Preparing the Computer and the HP 48 27-7
Using Kermit 27-9
Transferring Variables with Kermit 27-9
Choosing and Using File Names 27-11
Backing Up HP 48 Memory 27-12
Sending Kermit Commands 27-13
Using XMODEM 27-14
Using Other Serial Protocols 27-15

28. Libraries, Ports and Plug-In Cards

Port Memory and Plug-In Card Slots 28-1
Port 0 28-2
Card Slot 1 28-2
Card Slot 2 28-2
Using Backup Objects 28-3
Backing Up All Memory 28-5
Using Libraries 28-6
Installing and Removing Plug-In Cards 28-9
Expanding User Memory with Plug-In RAM Cards . . 28-15

29. Programming the HP 48

Understanding Programming 29-1
The Contents of a Program 29-2
Calculations in a Program 29-4
Structured Programming 29-4
Entering and Executing Programs 29-5
Viewing, Debugging, and Editing Programs 29-8
Using Programming Structures 29-10
Conditional Structures 29-10
Loop Structures 29-12
Error Trapping Structures 29-15
Using Local Variables 29-16
Creating Local Variables 29-16
Evaluating Local Names 29-17
Using Local Variables within Subroutines 29-18
Local Variables and User-Defined Functions 29-18

Exploring the Programs in the EXAMPLES Directory . 29-19
Using HP 48S/SX programs with the HP 48G/GX . . 29-20
Where to Find More Information 29-21

30. Customizing the HP 48
Customizing Menus 30-1
Enhancing Custom Menus 30-3
Customizing the Keyboard 30-4
User Modes 30-5
Assigning and Unassigning User Keys 30-5
Disabling User Keys 30-7
Recalling and Editing User Key Assignments 30-7

A. Support, Batteries, and Service
Answers to Common Questions A-1
Environmental Limits A-4
When to Replace Batteries A-4
Changing Batteries A-5
Testing Calculator Operation A-9
Self-Test A-10
Keyboard Test A-11
Port RAM Test A-12
IR Loop-Back Test A-13
Serial Loop-Back Test A-14
Limited One-Year Warranty A-15
If the Calculator Requires Service A-16

B. Error Messages

C. Menus

D. System Flags

E. Table of Units

F. Table of Built-In Equations

G. Operation Index

H. Stack Diagrams for Selected Commands

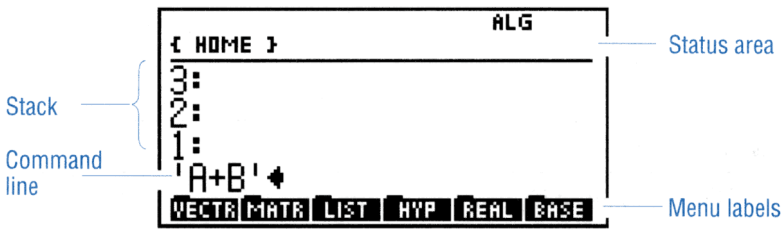
Index

1

The Keyboard and Display

Organization of the Display

For most operations, the display is divided into three sections, as shown below. This configuration is called the *stack display*. The following topics describe each of these sections.



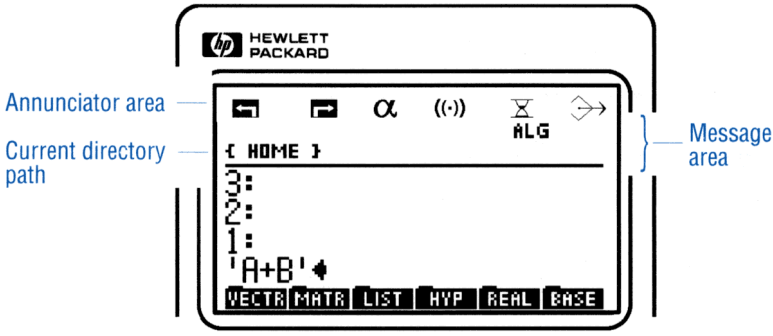
The Status Area, Annunciators, and Messages

The status area displays the following:

- **Annunciators.** These indicate the status of the calculator.
- **The current directory path.** When you turn the calculator on for the first time, the current directory path is { HOME }. Directories divide memory into parts, much as files do in a file cabinet. (Directories are covered in chapter 5.)
- **Messages.** These inform you when an error has occurred, or provide other information to help you use the calculator more effectively.

The table that follows describes the annunciators. The first six appear at the top of the display, and the rest (along with the directory path) share their "territory" with messages. Any message replaces the

annunciators and directory path; when you clear the message, the directory path and annunciators reappear.



Annunciators

Symbol	Meaning
⮌	Left-shift is active (you pressed ⮌).
⮎	Right-shift is active (you pressed ⮎).
α	The alpha keyboard is active (you can type letters and other characters).
((•))	(Alert.) An appointment has come due, or a low battery condition has been detected. See the message in the status area for information. (If no message appears, turn the calculator off and on. A message describing the cause of the alert will appear.)
⧖	Busy—not ready to process new input. (However, the calculator can remember up to 15 keystrokes while busy and then process them when free.)
⇛	Transmitting data to an external device.

Annunciators (continued)

Symbol	Meaning
RAD	Radians angle mode is active.
GRAD	Grads angle mode is active.
R∠Z	Polar/Cylindrical coordinates mode is active.
R∠∠	Polar/Spherical coordinates mode is active.
HALT	Program execution has been halted.
1 2 3 4 5	The indicated user flags are set.
1USR	The user keyboard is active for one operation.
USER	The user keyboard is active until you press ← USER.
ALG	Algebraic-entry mode is active.
PRG	Program-entry mode is active.

The Stack

The stack is a series of memory storage locations for numbers and other *objects*. The locations are called levels 1, 2, 3, etc. The number of levels changes according to how many objects are stored on the stack—from none to hundreds.

As you enter new numbers or other objects on the stack, the stack grows to accommodate them: the new data moves into level 1, and older data is "bumped" to higher levels. As you use data from the stack, the number of levels decreases as the data moves down to lower levels.

The stack display shows level 1 and up to three additional levels. Any additional levels are maintained in memory, but normally aren't displayed.

For more information about the stack and command line, see "Using the Stack for Calculations" on page 3-1.

The Command Line

The command line appears whenever you start keying in or editing text. The stack lines move up to make room. If you type more than 21 characters, information scrolls off the left side of the display, and an ellipsis (...) appears to tell you there is more information "in that direction."

The command line is closely tied to the stack. You use it to key in (or edit) text and then to process it, transferring the results to the stack.

After you finish using the command line, the stack display moves back down into the command line area.

For more information about the stack and command line, see "Using the Command Line" on page 2-7.

Menu Labels

Menu labels across the bottom of the display show the operations associated with the six white menu keys across the top of the keyboard. These labels change depending on the menu you select. See "Working with Menus" on page 1-9 for information about using menus.

Organization of the Keyboard

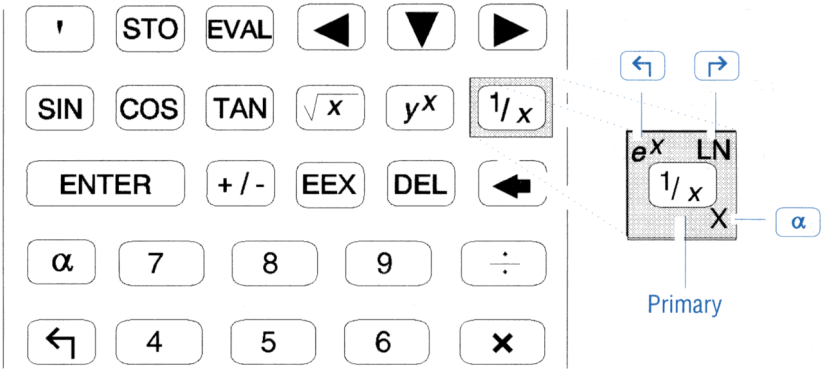
The HP 48 keyboard has six levels (or "layers") of functions, each containing a different set of keys:

- **Primary keyboard**. Represented by the labels on the key faces; for example, [+], [7], [ENTER], [TAN], and [▲] are on the primary keyboard.
- **Left-shift keyboard**. Activated by pressing the purple [←] key. Left-shift keys are labeled in purple and located above and to the left of their associated primary keys. To execute ASIN, for example, press the [←] key followed by the [SIN] key.
- **Right-shift keyboard**. Activated by pressing the green [→] key. Right-shift keys are labeled in green and located above and to the

right of their associated primary keys. To execute LN, for example, press the [→] key followed by the [1/x] key.

- **Alpha keyboard**. Activated by pressing the [α] key. Alpha keys are labeled in white and located to the right of their associated primary keys. Alpha keys are all capital letters. To generate "N", for example, press [α] followed by the [STO] key. Notice that when the alpha keyboard is active, the α annunciator appears. Notice also that the number pad still generates numbers.
- **Alpha left-shift keyboard**. Activated by pressing [α] and then [←]. Alpha left-shift characters include lowercase letters, along with some special characters. (Alpha left-shift characters are not shown on the keyboard.) To type "n", for example, press [α], then [←], and then [STO].
- **Alpha right-shift keyboard**. Activated by pressing [α] and then [→]. Alpha right-shift characters include Greek letters and other special characters. (Alpha right-shift characters are not shown on the keyboard.) To generate λ, for example, press [α], then [→], and then [NXT].

The unshifted and shifted Alpha keyboards are shown on page 2-3. Note also that you can easily access all displayable characters on the HP48 using the CHARS application (see page 2-4).



When you press ◀ (left-shift) or ▶ (right-shift), the ◀ or ▶ annunciator appears in the display.

To cancel a shift key:

- To clear the shift key, press it again.
- To change to the *other* shift key, press the other shift key.

Applications and Command Menus

Some keys have both right- and left-shifted labels printed above, but many have only one of the two.

Keys that have only labels printed in green represent *applications*. Each of these keys launches an application with specially designed user-interfaces that make it easier for you to *interact* with that application. The HP 48 has twelve application keys:

▶ CHARS	Displays a catalog of all 256 characters used by the HP 48 (see chapter 2).
▶ EQ LIB	Provides access to over 300 scientific equations, accompanying diagrams and variable sets, 40 physical constants, and the Multiple Equation Solver (see chapter 25).
▶ I/O	Facilitates transfer of data between the HP 48 and printers, computers, and other HP 48s (see chapter 27).
▶ LIBRARY	Allows you to access commands and programs on plug-in cards and port memory (see chapter 28).
▶ MEMORY	Provides access to the Variable Browser application for organizing and managing stored variables (see chapter 5).
▶ MODES	Provides access to the Calculator Modes screen and the Flag Browser (see chapter 4).
▶ PLOT	Provides access to the PLOT application and its 15 plot types (see chapters 22, 23, and 24).
▶ SOLVE	Provides access to the SOLVE application and its 5 kinds of equation-solving functions (see chapter 18).
▶ STACK	Accesses the Interactive Stack application (see chapter 3).
▶ STAT	Accesses the STAT (statistics) application and its data analysis and curve-fitting operations (see chapter 21).

(→)(SYMBOLIC) Accesses the symbolic algebra and calculus capabilities of the HP 48 (see chapter 20).

(→)(TIME) Accesses the HP 48's Alarm Browser and clock-setting features (see chapter 26).

Each of these applications also has a left-shifted version that displays the companion *command menu* for that application. For example, pressing (←)(STAT) displays a menu of commands that pertains to statistical analysis.

Command menus provide convenient access to commands for inclusion in programs, or when using functions directly from the stack display instead of from an application.

Cursor Keys

The six cursor keys differ from other keys because their behavior depends on whether a *cursor* is currently displayed. Their behavior when a cursor is being displayed is summarized below:

The Behavior of Cursor Keys

Key	No Shift	Right Shift
(◀)	Moves the cursor left.	Moves the cursor to the beginning.
(▶)	Moves the cursor right.	Moves the cursor to the end.
(▼)	Moves the cursor down.	Moves the cursor to the bottom (or end).
(▲)	Moves the cursor up.	Moves the cursor to the top (or beginning).
(DEL)	Deletes the current character.	Deletes all characters to the end.
(←)	Deletes the previous character.	Deletes all previous characters to the beginning.

Whenever there is no cursor being displayed, pressing any of these six keys executes the operation indicated by the colored label above them:

- ◀ (or PICTURE) displays the current picture.
- ▶ (or SWAP) swaps the objects on levels 1 and 2 of the stack.
- ▲ (or STACK) begins the Interactive Stack application.
- ▼ (or VIEW) puts the object on level 1 of the stack into its "best" viewing mode (see page 2-10).
- DEL (or CLEAR) clears the stack.
- ← (or DROP) drops the object on level 1 from the stack.

The CANCEL Key

When the HP 48 is on, ON becomes the CANCEL key. Generally, CANCEL halts the current activity—so you can immediately start your next task or recover from an unexpected situation.

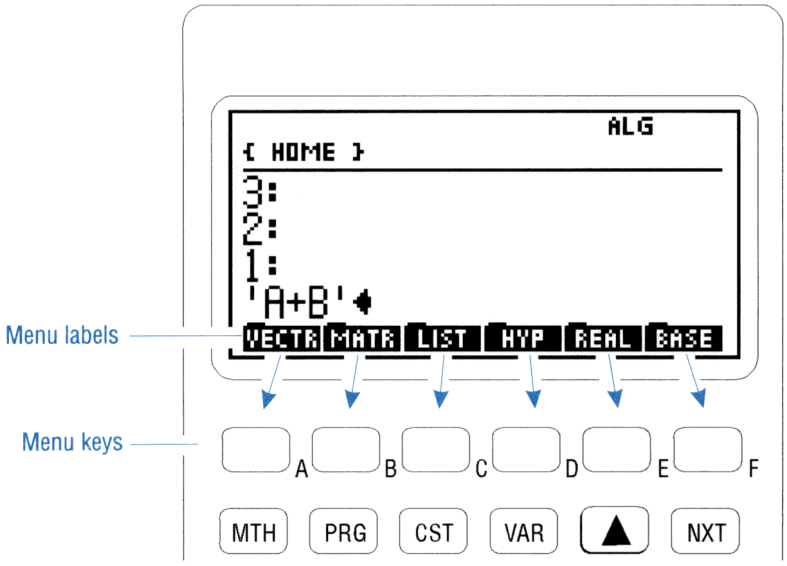
To stop the calculator:

- To delete the command line, press CANCEL.
- To cancel a special environment and restore the stack display, press CANCEL.
- To cancel a running program, press CANCEL.

Menus: Extending the Keyboard

The HP 48 uses menus extensively to accommodate its hundreds of built-in commands and functions.

A menu is a set of operations defined for the six blank *menu keys* at the top of the keyboard. The current operations are described by the six *menu labels* at the bottom of the display.



Some menus have multiple sets of labels, called *pages*. If a menu label has a tab over its left corner like a file folder, it selects another menu called a *submenu*.

Working with Menus

To display a menu:

1. Press the key or keys corresponding to the menu you want.
2. Menus with more than six entries are extended over two or more pages. If necessary, change to the menu page you want:
   - To move to the next page, press NXT.
   - To move to the previous page, press ← PREV.

   Note that you will eventually cycle through and return to the first page.

When you want to go to another menu, simply press the keys for that menu—you don't "get out" or "back out" of one menu to go to another—you just go to the new one.

To display the previous menu:

- Press ↱ MENU.

Occasionally you may be working primarily with a particular menu, but need to use commands in another menu. For example, you may need to briefly leave the second page of the SYMBOLIC menu to use a command in the second page of the MTH PROB menu.

When you switch from one menu to another, the HP 48 stores the identity and page number of the last menu you were in. Pressing ↱ MENU (found over the NXT key) returns you to that menu. Menus that list *only* additional menus (such as the MTH and PRG menus) aren't stored as the last menu.

To select a function from a menu:

- Press the menu key below the label of the operation.

2

Entering and Editing Objects

The basic items of information the HP 48 uses are called *objects*. For example, a real number, an equation, and a program are each an object. An object occupies a single level on the stack, and can be stored in a variable.

The HP 48 can store and manipulate many types of objects, including real and complex numbers, binary integers, arrays, algebraic expressions, programs, graphics, text strings, and lists. Many of the HP 48's operations are the same for all types of objects, while others only apply to particular types of objects.

Keying In Numbers

To key in a simple number:

1. Press the appropriate number and (·) keys.
2. If the number is negative, press (+/–).

To correct a typing mistake:

- Press (←) (the backspace key) to erase the mistake, then retype it correctly.

To delete the entire number from the command line:

- Press (CANCEL).

Example: Enter the number −123.4 in the command line.

Step 1: Key in the digits.

123 . 4

Step 2: Make the number negative.

+/–

Press CANCEL (the ON key) to delete the command line.

To key in a number as a mantissa and an exponent:

1. Key in the mantissa. If it's negative, press +/– to change its sign.
2. Press EEX. (It types an E for "exponent.")
3. Key in the exponent—the power of 10. If it's negative, press +/–.

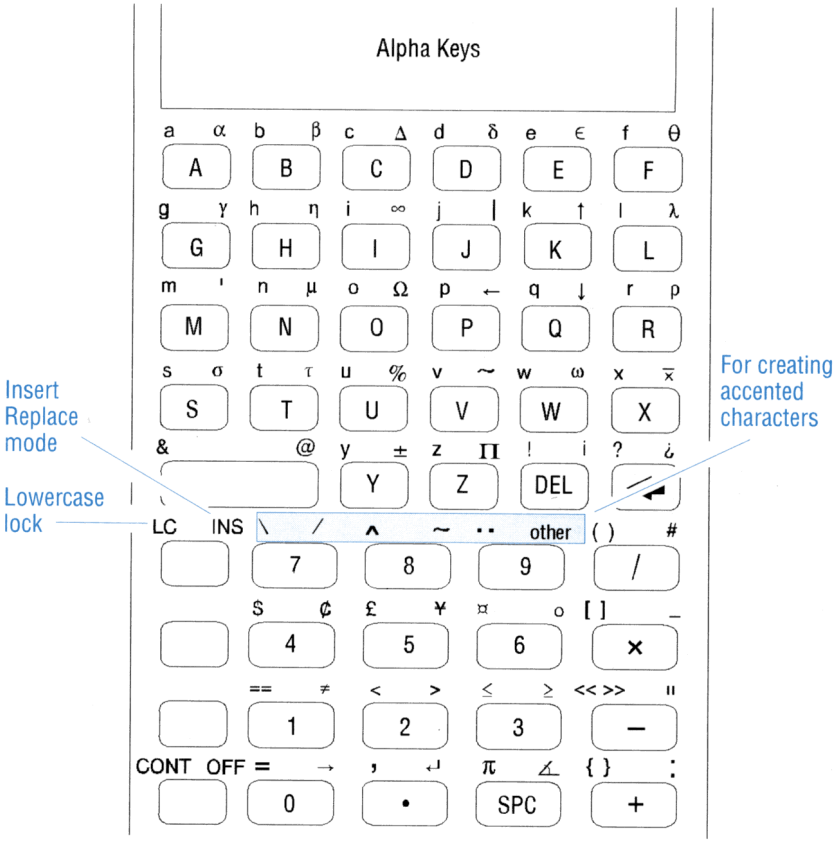
Keying In Characters (the Alpha Keyboard)

The HP 48 features an "alpha" keyboard that lets you type letters and other characters into your data. Activate the alpha keyboard using the α key. (The α annunciator turns on while the alpha keyboard is active.)

When you press the α key, you can type in capital letters. The letters available are printed in white to the lower right of each key. In addition, the left-shift and right-shift keys provide additional characters:

- The left-shifted alpha keyboard types in lowercase letters.
- The right-shifted alpha keyboard types in Greek letters and miscellaneous symbols.

To keep the HP 48 keyboard from appearing too cluttered, most of the alpha left- and right-shift keys are not shown on it. For your reference, the next illustration shows how keys are interpreted when the α annunciator is on.



To key in a single character:

- Press (α) and key in the character.
  or
- Hold down (α), key in the character, then release (α).

To key in several characters:

- Press (α)(α), key in the characters, then press (α) again.
  or
- Hold down (α), key in the characters, then release (α).

Pressing [α] *once* activates Alpha-entry mode for one character only. Pressing [α] *twice* locks Alpha-entry mode. Alpha-entry mode then remains active until you press [α] again or press [ENTER] (or [CANCEL]). You can also press and hold down [α] while you type several characters in a row. If you prefer, set Flag −60 to enable a single press of [α] to lock Alpha-entry mode.

To lock or unlock the lowercase keyboard:

- If α is locked on, press [↰][α] to lock lowercase mode.
- If α is off, press [α][α][↰][α] to lock lowercase mode.
- To unlock lowercase, press [↰][α]. Also, terminating the entry process—by pressing [ENTER] or [CANCEL], or by executing a command—automatically unlocks lowercase mode.

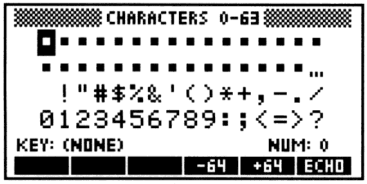
While in lowercase Alpha-entry mode, you must use [↰] to get uppercase letters. Lowercase mode automatically unlocks when you press [ENTER] or [CANCEL], or execute a command.

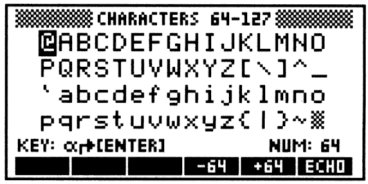
Keying in Special Characters

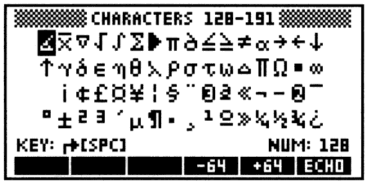
Although most of the 256 characters that can be displayed on the HP 48 are available on the Alpha keyboard, it is easy to forget the particular sequence of keystrokes needed for the less-frequently used characters.

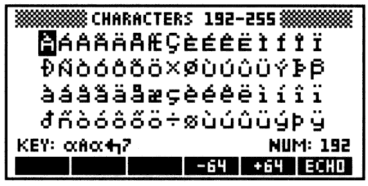
The CHARS application is designed to avoid this problem by enabling you to select characters directly from the display and insert them at the cursor point. CHARS displays the HP 48's characters 64 at a time, along with the number of each character and the keystroke used to type it in from the alpha keyboard.

The Four CHARS Screens









To use CHARS to view or key in characters:

1. Press ↱ CHARS. A display of 64 characters appears.
2. Use -64 and +64 to cycle through the pages of characters.
3. Use the arrow keys (◀, ▶, ▲, and ▼) to select a character. Notice that the character's number is shown at lower right, and its keystroke is shown at lower left.
4. To insert the selected character at the cursor point, press ECHO.
5. Repeat steps 2, 3, and 4 to insert additional characters.
6. When you've finished, press ENTER or CANCEL to exit CHARS.

Keying In Objects with Delimiters

Real numbers represent one type of object. Most other types of objects require special *delimiters* to indicate which type of object they are.

Here's a partial list of different types of objects and their corresponding delimiters.

Objects	Delimiters	Keys	Examples
Real number	none		14.75
Complex number	( )	◁ ( )	(8.25,12.1)
String	" "	▷ " "	"Hello"
Array	[ ]	◁ [ ]	[4.8 -1.3 2.1]
Unit	_	▷ _	11.5_ft
Program	« »	◁ « »	« ∫ DUP NEG » *or* « → a b 'a*b' »
Algebraic	' '	'	'A-B'
List	{ }	◁ { }	{ 6.8 5 "FIVE" }
Built-in command	none		FIX
Name	' '	'	VOL *or* 'VOL'

To key in an object using delimiters:

- To key in data within opening and closing delimiters, press the delimiter key, then key in the data. (The delimiter key types *both* delimiters.)
- To insert a single delimiter within the data, press the delimiter key where required, then delete the one you don't want.

Even large-scale objects such as algebraics and arrays can be entered in the command line.

To key in an algebraic object using the command line:

1. Press ' to type the delimiters.
2. Key in the numbers, variables, operators, and parentheses in the expression or equation in left-to-right order. Press ▶ to skip past right parentheses.

To key in a matrix using the command line:

1. Press ◁ [ ] to begin the array and ◁ [ ] to begin the first row.
2. Key in the first row. Press SPC between each element.
3. Press ▶ to move the cursor past the] row delimiter.

4. Optional: Press (→)(↵) (new line) to start a new row in the display.
5. Key in the rest of the matrix. You don't need to add [ ] delimiters for subsequent rows—they're added automatically later.
6. Press (ENTER).

To key in a vector using the command line:

1. Press (←)([]) to begin the array. Because a vector is equivalent to a *single-column matrix*, there is no need to group elements into rows using additional delimiters unless you specifically want to create a *row-vector*.
2. Key in the vector elements. Press (SPC) to separate the elements.
3. Press (ENTER).

The HP 48 also provides special entry environments for algebraics and arrays that use visually intuitive methods to key in these large objects. See chapter 7, "The EquationWriter," and chapter 8, "The MatrixWriter," for information.

Using the Command Line

The command line is essentially a workspace for keying in and editing the objects you will enter into the HP 48. The command line appears whenever you enter or edit text (except when you're using the EquationWriter application).

Accumulating Data in the Command Line

You can key any number of characters into the command line, using up to half of the available memory. To enter more than one object in the command line, use spaces, newlines ((→)(↵)), or delimiters to separate objects. For example, you can key in 12 (SPC) 34 to enter two numbers.

If you enter an ℗ character outside of a string in the command line, both the ℗ character and the adjacent text are treated as a "comment" and are stripped away when you press (ENTER).

When you type in the command line, characters are normally *inserted* at the cursor position, and any trailing characters move to the

right. In addition, you can use the following keys to edit data in the command line:

Command Line Operations

Key	Description
◀ ▶	Move the cursor left and right in the command line. (→ ◀ and → ▶ move the cursor to the far left and far right.)
▲ ▼	If the command line has more than one line, move the cursor up and down one line. (→ ▲ and → ▼ move the cursor to the first and last line.)
	If the command line has only one line, ▲ selects the Interactive Stack, and ▼ displays the EDIT menu.
⌫	Erases the character to the left of the cursor.
DEL	Deletes the character beneath the cursor.
↩ EDIT	Displays the EDIT menu, which contains additional editing operations.
→ ENTRY	Changes the command-line entry mode to Program-entry mode or Algebraic/Program-entry mode, as described below.
ENTER	Processes the text in the command line—moves objects to the stack and executes commands.
CANCEL	Discards the entire command line.

Selecting Command-Line Entry Modes

The HP 48 has four command-line entry modes that make it easier for you to key in objects.

- **Immediate-Entry Mode.** (Activated automatically, and indicated by no entry-mode annunciator.) In Immediate-entry mode, the contents of the command line are entered and processed immediately when you press a function or command key (such as +, SIN, or STO). Immediate-entry mode is the default mode.
- **Algebraic-Entry Mode.** (Activated when you press ', and indicated by the ALG annunciator.) Algebraic-entry mode is used primarily for keying in names and algebraic expressions for immediate use. In Algebraic-entry mode, function keys act as

typing aids (for example, SIN types SIN( ). Other commands *are* executed immediately (for example, STO or ◀ PURGE).

- **Program-Entry Mode.** (Activated when you press ◀ « » or ◀ { }, and indicated by the PRG annunciator.) Program-entry mode is used primarily for entering programs and lists. It's also used for command-line editing (◀ EDIT). In Program-entry mode, function keys and command keys act as typing aids (for example, SIN types SIN, and STO types STO). Only non-programmable operations are executed when you press a key (for example, ENTER, VAR, or ▶ ENTRY).
- **Algebraic/Program-Entry Mode.** (Activated when you press ' while in Program-entry mode, and indicated by the ALG and PRG annunciators.) Algebraic/Program-entry mode is used for keying algebraic objects into programs.

To change entry modes manually:

- Press ▶ ENTRY.

Pressing ▶ ENTRY switches from Immediate-entry to Program-entry mode, and between Program-entry and Algebraic/Program-entry modes.

▶ ENTRY allows you to accumulate commands in the command line for later execution. For example, you can manually invoke Program-entry mode to enter 4 5 + √ into the command line, and then press ENTER to calculate $\sqrt{4+5}$. ▶ ENTRY also makes it easier to edit algebraic objects in programs.

Example: Calculate 12 − log(100) by including the LOG command in the command line.

Step 1: Enter the command line.

12 SPC 100 ▶ ENTRY ▶ LOG

Step 2: Process the command line to complete the calculation.

ENTER −

Recovering Previous Command Lines

The HP 48 automatically saves a copy of the four most recently executed command lines.

To retrieve a recent command line:

1. Press → CMD (found over the +/− key).
2. Select the command you want to retrieve, using the ▲ and ▼ keys, and press OK.

Viewing and Editing Objects

You can't always see all the objects on the stack—you can see only the beginning of large objects, and you can't see objects that have changed levels and scrolled off the display.

To enable you to view any object on the stack, the HP 48 gives you a choice of *environments* for viewing and editing objects. An environment defines a particular display and keyboard behavior—it determines how you see and change the object.

To view or edit an object:

1. Depending on the location of the object and the desired environment, press the keys listed in the table below.
2. View or edit the object according to the rules of the environment.
3. Exit the environment:
   - To exit after viewing, press CANCEL.
   - To save changes you've made, press ENTER.
   - To discard changes you've made, press CANCEL.

Viewing or Editing an Object

Location of Object	Viewing/Editing Environment	Keystrokes to View or Edit
Level 1	Command line	◀ EDIT
	Best (see below)	▼
Level *n*	Interactive Stack	▲ to level *n*, VIEW
Variable *name*	Command line	' *name* ◀ EDIT
	Best	' *name* ▶ RCL ▼

The command line is the simplest viewing and editing environment:

- The EDIT menu is displayed, which provides operations that make it easier to edit large objects. (See "Using the EDIT Menu" below.)
- Real and complex numbers are displayed with full precision (standard format), regardless of the current display mode.
- Programs, lists, algebraics, units, directories, and matrices are formatted onto multiple lines.
- All the digits of binary numbers, all the characters in strings, and entire algebraic expressions are displayed.

The "best" editing environment is the one that the HP 48 determines is most appropriate based on the type of object:

- Algebraic objects and unit objects are copied into the EquationWriter environment in scrolling mode. To edit the equation, enter selection mode by pressing ◀ (see chapter 7).
- Matrices are copied into the MatrixWriter environment (see chapter 8).
- All other object types are copied into the command line.

The Interactive Stack is an environment for viewing, editing, and manipulating all objects on the stack. (See "The Interactive Stack" on page 3-6.)

2 Using the EDIT Menu

Whenever the command line is present, you can press ◀EDIT to get the EDIT menu. The EDIT menu is also displayed whenever you perform a viewing or editing operation as described in the previous section.

Certain operations in the EDIT menu use the concept of a *word*—a series of characters between spaces or newlines. For example, pressing ←SKIP skips to the beginning of a *word*. The following table lists the operations available in the EDIT menu:

EDIT Menu Operations

Key	Description
◀EDIT:	
←SKIP	Moves the cursor to the beginning of the current word.
SKIP→	Moves the cursor to the beginning of the next word.
←DEL	Deletes characters from the beginning of the word to the cursor.
DEL→	Deletes characters from the cursor to the end of the word.
▶ ←DEL	Deletes characters from the beginning of the line to the cursor.
▶ DEL→	Deletes all characters from the cursor to the end of the line.
INS	Switches the command-line entry mode between *Insert* mode (◆ cursor) and *Replace* mode (▮ cursor). A ■ in the menu label indicates Insert mode is active.
↑STK	Activates the Interactive Stack. (See "The Interactive Stack" in chapter 3.)

3

Stack

The stack is a series of storage locations for numbers and other objects. In general, you will use the HP 48 by entering numbers and other objects onto the stack, then executing commands that operate on the data.

Using the Stack for Calculations

You ordinarily do calculations by entering objects onto the stack and then executing the appropriate functions and commands. The fundamental concepts of stack operations are these:

- A command that requires *arguments* (objects the command acts upon) takes its arguments from the stack. Therefore, the arguments must be present *before* you execute the command.
- The arguments for a command are removed from the stack when the command is executed.
- Results are returned to the stack so that you can see them and use them in other operations.

Making Calculations

When you execute a command, any arguments in the command line are automatically put onto the stack *before* the command is executed. This means you don't always have to press ENTER to put the arguments on the stack—you can leave one or more arguments in the command line when you execute the command. (You should still think of the arguments as being on the stack, though.)

To use a one-argument command:

1. Enter the argument into level 1 (or into the command line).
2. Execute the command.

Example: Use the one-argument commands LN (→ LN) and INV (1/x) to calculate 1/ln 3.7.

3.7 → LN
1/x

To use a two-argument command:

1. Enter the first argument and then the second argument. The first argument should be in level 2, and the second in level 1 (or in the command line).
2. Execute the command.

A two-argument command acts on the arguments (objects) in levels 1 and 2, and returns the result to level 1. The rest of the stack *drops* one level—for example, the previous contents of level 3 move to level 2. The arithmetic functions (+, −, ×, /, and ^) and percent calculations (%, %CH, and %T) are examples of two-argument commands.

Example: Calculate 85 − 31.

85 ENTER 31 −

Example: Calculate $\sqrt{45} \times 12$.

45 √x 12 ×

Example: Calculate $4.7^{2.1}$.

4.7 ENTER 2.1 y^x

To enter more than one argument in the command line:

- Press SPC to separate arguments.

Example: Calculate $\sqrt[4]{2401}$.

2401 SPC 4 ▶ $\sqrt[x]{y}$

```
1:                          7
VECTR MATR LIST HYP REAL BASE
```

Since the HP 48's stack retains previous results, it makes it especially easy to perform chain calculations.

To use previous results (chain calculation):

1. If necessary, move the previous results to the proper stack level for the command (see "Manipulating the Stack," below).
2. Execute the command.

Example: Calculate $(12 + 3) \times (7 + 9)$.

Step 1: Perform the additions.

12 ENTER 3 +
7 ENTER 9 +

```
2:                         15
1:                         16
VECTR MATR LIST HYP REAL BASE
```

Step 2: Notice that the two intermediate results remain on the stack. Now, multiply them.

×

```
1:                        240
VECTR MATR LIST HYP REAL BASE
```

Example: Calculate $23^2 - (13 \times 9) + \frac{5}{7}$.

Step 1: First, calculate 23^2 and the product 13×9.

23 ◀ x^2
13 ENTER 9 ×

```
2:                        529
1:                        117
VECTR MATR LIST HYP REAL BASE
```

Step 2: Subtract the two intermediate results and calculate $\frac{5}{7}$.

−
5 ENTER 7 ÷

```
2:                        412
1:                .714285714286
VECTR MATR LIST HYP REAL BASE
```

Step 3: Add the two results.

+

```
1:              412.714285714
VECTR MATR LIST HYP REAL BASE
```

3 Manipulating the Stack

The HP 48 enables you to reorder, duplicate, and delete specific objects on the stack.

To swap the objects in levels 1 and 2:

- Press ◀ SWAP (or ▶ when no command line is present).

The SWAP command is useful with commands where the order is important, such as −, /, and ^.

Example: Use ◀ SWAP to help calculate $\frac{9}{\sqrt{13+8}}$.

Step 1: First, calculate $\sqrt{13+8}$.

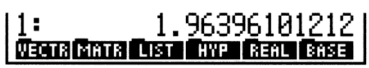
13 ENTER 8 + √x

Step 2: Enter 9 and swap levels 1 and 2.

9 ◀ SWAP

Step 3: Divide the two values.

÷



To duplicate the object in level 1:

- Press ◀ STACK NXT DUP (or press ENTER if no command line is present).

The DUP command duplicates the contents of level 1 and bumps the rest of the stack up one level.

Example: Calculate $\frac{1}{47.5} + \left(\frac{1}{47.5}\right)^4$.

Step 1: First, calculate the inverse of 47.5 and duplicate the value.

47.5 1/x ENTER

Step 2: Raise the value to the 4th power.

4 y^x

Step 3: Add the result to the original value.

+

To delete the object in level 1:

- Press ← DROP (or ← when no command line is present).

When you execute the DROP command, the remaining objects on the stack drop down one level.

To clear the entire stack:

- Press ← CLEAR (or DEL when no command line is present).

Recalling the Last Arguments

The LASTARG command (→ ARG) places the arguments of the most recently executed command on the stack so that you can use them again. This is particularly useful for more complicated arguments, such as algebraics and matrices.

To recall the arguments of the last command:

- Press → ARG.

Example: Use → ARG to help calculate $ln2.3031 + 2.3031$.

Step 1: Calculate ln 2.3031, then retrieve the argument of LN. (ARG is above the EEX key.)

2.3031 → LN
→ ARG

Step 2: Add the two numbers.

⊞

Restoring the Last Stack (UNDO)

The UNDO command ↷ UNDO restores the stack to the way it was before you executed the most recent command.

To restore the stack to its previous state:

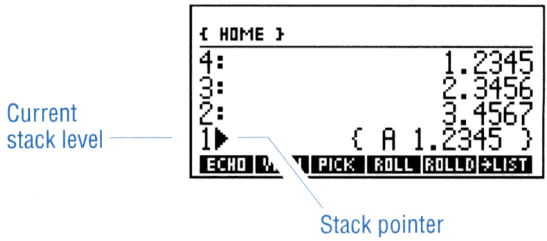
- Press ↷ UNDO.

The Interactive Stack

The normal stack display is a "window" that shows level 1 and as many higher levels as will fit in the display. The HP 48 also features the *Interactive Stack*, a special environment in which the keyboard is redefined for a specific set of stack-manipulation operations. The Interactive Stack lets you do the following:

- Move the window to see the rest of the stack.
- Move and copy objects to different levels.
- Copy the contents of any stack level to the command line.
- Delete objects from the stack.
- Edit stack objects.
- View stack objects in an appropriate environment.

When you activate the Interactive Stack, the *stack pointer* turns on (pointing to the *current stack level*), the keyboard is redefined, and the Interactive Stack menu is displayed. You must exit the Interactive Stack before you can execute any other calculator operations.



To use the Interactive Stack:

1. Press ▶ STACK (or +STK in the EDIT menu) to activate the Interactive Stack. (If no command line is present, press ▲.) The stack pointer appears on level 1.
2. Use the keys described in the following table to view or manipulate the stack.
3. Press ENTER (or CANCEL) to leave the Interactive Stack and show the changed stack.
4. Optional: To cancel changes made to the stack in the Interactive Stack, press ▶ UNDO.

If a command line is present when you select the Interactive Stack, only the ECHO key appears in the menu because all you can do is copy (*echo*) an object from a higher stack level to the cursor position in the command line.

Interactive-Stack Operations

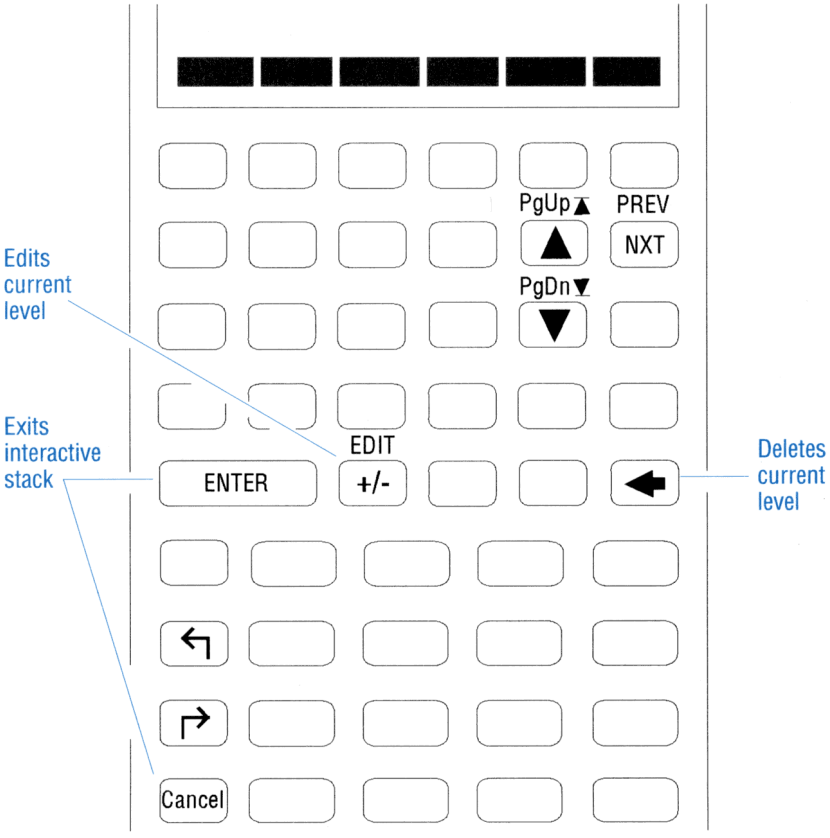
Key	Description
▶ STACK (or ▲ when no command line is present):	
ECHO	Copies the contents of the current level into the command line at the cursor position.
VIEW	Views or edits the object in the current level using the best environment. Press ENTER when finished editing (or CANCEL to abort).
▶ VIEW	Views or edits the object specified by the name or level number using the best environment. Press ENTER when finished editing (or CANCEL to abort).
PICK	Copies the contents of the current level to level 1 (equivalent to *n* PICK).

Interactive-Stack Operations (continued)

Key	Description
ROLL	Moves the contents of the current level to level 1, and rolls upwards the portion of the stack beneath the current level (equivalent to *n* ROLL).
ROLLD	Moves the contents of level 1 to the current level, and rolls downwards the portion of the stack beneath the current level (equivalent to *n* ROLLD).
→LIST	Creates a list containing all the objects in levels 1 through the current level (equivalent to *n* →LIST).
DUPN	Duplicates levels 1 through the current level (equivalent to *n* DUPN). For example, if the pointer is at level 3, levels 1, 2, and 3 are copied to levels 4, 5, and 6.
DRPN	Deletes levels 1 through the current level (equivalent to *n* DROPN).
KEEP	Clears all levels above the current level.
LEVEL	Enters the current level number into level 1.
▲	Moves the stack pointer up one level. When preceded by ◀, moves the stack pointer up four levels (◀ PgUp in the following keyboard illustration); when preceded by ▶, moves the stack pointer to the top of the stack (▶ ⏫ in the following keyboard illustration).
▼	Moves the stack pointer down one level. When preceded by ◀, moves the stack pointer down four levels (◀ PgDn in the following keyboard illustration); when preceded by ▶, moves the stack pointer to the bottom of the stack (▶ ⏬ in the following keyboard illustration).
◀ EDIT	Copies the object in the current level into the command line for editing. Press ENTER when finished editing (or CANCEL to abort).
⌫	Deletes the object in the current level.
NXT	Selects the next page of Interactive-Stack operations.
ENTER	Exits the Interactive Stack.
CANCEL	Exits the Interactive Stack.

Most of the operations in the Interactive Stack menu have equivalent programmable commands (see "Stack Command Menu" on page 3-11).

The redefined keyboard for the Interactive Stack looks like this:



To copy an object from the stack into the command line:

1. Put the cursor at the point in the command line where you want the object placed.
2. Press ◀ EDIT +STK .
3. Press ▲ and ▼ to move the Interactive Stack pointer to the desired object and press ECHO .
4. Press ENTER (or CANCEL) to leave the Interactive Stack.

Example: Use the Interactive Stack to insert the number 1.2345 into the command line, creating the list { A 1.2345 }.

Step 1: Put these numbers on the stack.

1.2345 ENTER
2.3456 ENTER
3.4567 ENTER

Step 2: Start entering the list.

◀ {} A

Step 3: Select the Interactive Stack.

◀ EDIT ↑STK

Step 4: Move the pointer to level 3, echo the object, and leave the Interactive Stack.

▲ ▲ ECHO ENTER

Step 5: Put the list on the stack.

ENTER

Stack Command Menu

The following table describes the programmable commands that manipulate the stack. These commands are available from the ◀ STACK command menu.

Command/Description	Example	
	Input	Output
DEPTH Returns the number of objects on the stack.	3: 2: 16 1: 'X1'	3: 16 2: 'X1' 1: 2
DROP2 Removes the objects in levels 1 and 2.	3: 12 2: 10 1: 8	3: 2: 1: 12
DROPN Removes the first $n + 1$ objects from the stack (n is in level 1). Shown in menu as DRPN.	4: 123 3: 456 2: 789 1: 2	4: 3: 2: 1: 123
DUP Duplicates the object in level 1.	3: 2: 232 1: 543	3: 232 2: 543 1: 543
DUP2 Duplicates the objects in levels 1 and 2.	4: 3: 2: 'A' 1: (2,3)	4: 'A' 3: (2,3) 2: 'A' 1: (2,3)
DUPN Duplicates n objects on the stack, starting at level 2 (n is in level 1).	6: 5: 4: 123 3: 456 2: 789 1: 3	6: 123 5: 456 4: 789 3: 123 2: 456 1: 789

Command/Description	Example	
	Input	Output
OVER Returns a copy of the object in level 2.	3: 2: 'AB' 1: 1234	3: 'AB' 2: 1234 1: 'AB'
PICK Returns a copy of the object in level $n + 1$ to level 1 (n is in level 1).	4: 123 3: 456 2: 789 1: 3	4: 123 3: 456 2: 789 1: 123
ROLL Moves object in level $n + 1$ to level 1 (n is in level 1).	5: 555 4: 444 3: 333 2: 222 1: 4	5: 4: 444 3: 333 2: 222 1: 555
ROLLD Rolls down a portion of the stack between level 2 and level $n + 1$ (n is in level 1).	6: 12 5: 34 4: 56 3: 78 2: 90 1: 4	6: 5: 12 4: 90 3: 34 2: 56 1: 78
ROT Rotates the first three objects on the stack (equivalent to 3 ROLL).	3: 12 2: 34 1: 56	3: 34 2: 56 1: 12

4

Modes

The HP 48 operates using many different *modes*, depending on the nature of the operation it's performing. Many of these modes are controlled automatically by the commands you select; others are determined by settings that you control.

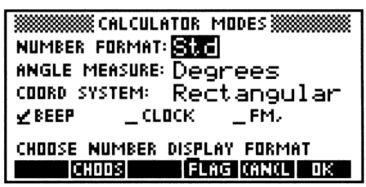
The MODES application and its companion command menu give you access to those modes you can control.

Using the MODES Application

The MODES application provides a convenient way for you to control the modes the HP 48 uses.

To use the MODES application:

- Press ↱ (MODES).



The Calculator Modes Screen

This screen allows you to set the following calculator modes:

- Display mode for the number format
- Angle mode
- Coordinate mode

- Beep mode
- Clock display mode
- Fraction mark mode

Setting the Display Mode

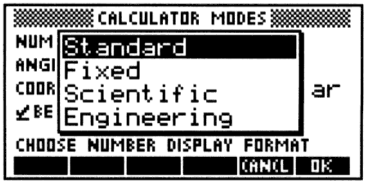
The *display mode* controls how the HP 48 displays numbers. (Note that numbers can be *displayed* differently than they are *stored*. Regardless of the display mode, numbers are always stored as signed, 12-digit mantissas with signed, 3-digit exponents.)

The HP 48 has four display modes:

- **Standard mode** (Std)—Displays numbers using full precision. All significant digits to the right of the decimal point are shown, up to 12 digits.
- **Fix mode** (Fix)—Displays numbers rounded to a specified number of decimal places. Real numbers on the stack are displayed with digit separators (separating digits into groups of three): commas (if a period is used as the decimal point) or periods (if a comma is used as the decimal point).
- **Scientific mode** (Sci)—Displays a number as a mantissa (with one digit to the left of the decimal point and a specified number of decimal places) and an exponent.
- **Engineering mode** (Eng)—Displays a number as a mantissa with a specified number of digits, followed by an exponent that is a multiple of 3.

To set the display mode:

- Press → MODES.
- Highlight the NUMBER FORMAT: field.
- Press CHOOS. (Or, press +/- repeatedly to cycle through the fixed list of options, and stop when your choice appears in the field.)



The Number Format Choose Box

- Select a number format and press OK .
- If the format is Fix, Sci, or Eng, press ▶, type the number of display digits, and press ENTER.
- Press OK .

Setting the Angle Mode

The angle mode determines how the calculator interprets angle arguments and how it returns angle results.

Angle Modes

Mode	Definition	Annunciator
Degrees	$^1/_{360}$ of a circle	(none)
Radians	$^1/_{2\pi}$ of a circle	RAD
Grads	$^1/_{400}$ of a circle	GRAD

To set the angle mode from the MODES application:

1. Press ↱ MODES.
2. Use the arrow keys to highlight the ANGLE MEASURE: field.
3. Do one of the following:
   - Press CHOOS to display the list of choices, select your choice, and press OK .
   - Press +/- repeatedly until your choice appears in the field.
4. Press OK to confirm your choice, or CANCL to cancel it.

To set the angle mode directly from the keyboard:

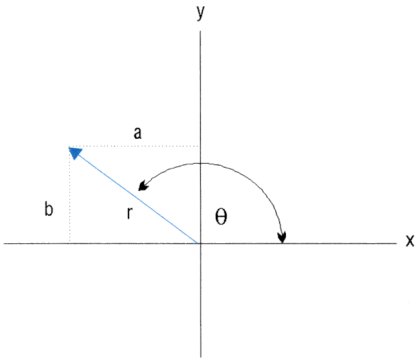
- Press ↰ RAD to switch between Radians mode and Degrees mode. (If Grads mode had been previously selected in the MODES application, this keystroke switches between Radians mode and Grads mode instead.)

Setting the Coordinate Mode

The coordinate mode affects how complex numbers and vectors are displayed. Complex numbers and two-dimensional vectors can be displayed in either *rectangular* mode (`(X,Y)` or `[ X Y ]`) or *polar* mode (`(R,∠)` or `[ R ∠ ]`).

Three-dimensional vectors can be displayed in *rectangular* mode (`[ X Y Z ]`), *cylindrical* mode (`[ R ∠ Z ]`), or *spherical* mode (`[ R ∠ ∠ ]`).

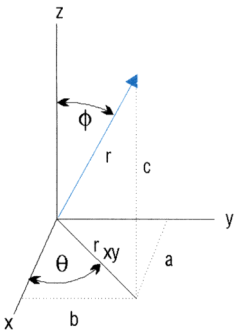
Note that no matter how these objects are *displayed*, they are always stored in *rectangular* mode, and calculations are based on this internal rectangular representation.



Two-Dimensional Display Modes

Rectangular	Polar
[a b]	[r ∠θ]

Coordinate Systems for Complex Numbers and 2D Vectors



Three-Dimensional Display Modes

Rectangular	Cylindrical	Spherical
[a b c]	[r_{xy} ∠θ c]	[r ∠θ ∠ϕ]

Coordinate Systems for 3D Vectors

To set the coordinate mode from the MODES application:

1. Press (→)(MODES).
2. Use the arrow keys to highlight the COORD SYSTEM: field.
3. Do one of the following:
   - Press CHOOS to display the list of choices, select one, and press OK.
   - Press (+/−) repeatedly until your choice appears in the field. Note that Polar means "polar cylindrical" when 3D vectors are displayed.
4. Press OK to confirm your choice, or CANCL to cancel it.

To change the coordinate mode directly from the keyboard:

- Press (→)(POLAR) to switch between Rectangular mode and Polar (cylindrical) mode. If Spherical mode had been previously selected in the MODES application, this keystroke switches between Rectangular mode and Spherical mode instead.

Setting the Beep

By default, the HP 48 sounds a "beep" whenever an error occurs. You can turn this beeper on and off.

To set the beep using the MODES application:

1. Press (→)(MODES).
2. Highlight the BEEP field and press ✓CHK (or (+/−)) until the desired setting is shown (checked—beep is on; unchecked—beep is off).
3. Press OK to confirm your choice, or CANCL to cancel it.

Setting the Clock Display

The HP 48 can display a clock that includes the date and time.

To display the clock:

1. Press (→)(MODES).
2. Highlight the CLOCK field and press ✓CHK (or (+/−)) until the desired setting is shown (checked—clock displayed; unchecked—no clock displayed).
3. Press OK to confirm your choice, or CANCL to cancel it.

Setting the Fraction Mark

A *fraction mark* is the displayed punctuation separating the integer and fractional parts of a real number (the "decimal point"). Because different countries use different fraction marks, the HP 48 allows two different kinds of fraction marks: the period (.) and the comma (,). As the table below indicates, the fraction mark used also changes the punctuation used to separate digits and arguments:

Fraction Mark	Digit Separator	Argument Separator
. (3.456)	, (34,300.54)	, (3,4)
, (3,456)	. (34.300,54)	; (3;4)

To set the fraction mark:

1. Press ↱(MODES).
2. Highlight the FM, field, and press ✓CHK (or +/−) until the desired setting is shown (checked—fraction mark is comma; unchecked—fraction mark is period).
3. Press OK to confirm your choice, or CANCL to cancel it.

Using System Flags

Most modes are controlled by *system flags*. The HP 48 has 64 system flags, numbered −1 through −64. Each flag can have two states: set (value of 1) or clear (value of 0). The system flags and the modes they control are described in appendix D.

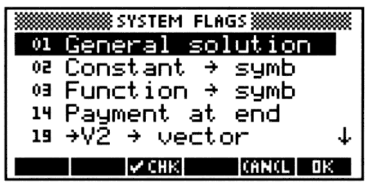
You can control modes by directly manipulating the system flags. You can access the flags by using the Flag Browser, part of the MODES application, or by using the FLAG command submenu.

Using the Flag Browser

The HP 48 uses two kinds of system flags: flags that single-handedly determine a mode, and flags that work with others to determine a mode. The Flag Browser lets you view and set the "single-handed" flags.

To view or change flag settings using the Flag Browser:

1. Press ↱(MODES).
2. Press FLAG to begin the Flag Browser.



The Flag Browser

3. Use the arrow keys to scroll through the flags. A check-mark to the left of the flag number indicates that the flag is *set*. The text describes how the flag's setting affects the calculator's function.
4. Press ✓CHK to change the flag setting. Note that the description changes to reflect the new setting.
5. When you finish, press OK to confirm your changes (if any), or CANCL to cancel them.

Using the FLAG Command Submenu

The commands for setting, clearing, and testing flags are in the MODES FLAGS menu (↰ MODES FLAG). (They are also duplicated in the PRG TEST menu.) These commands take flag numbers as arguments.

To use a flag command:

1. Enter the number of the flag (negative for a system flag).
2. Execute the command (see the table below).

Flag Commands

Key	Programmable Command	Description
↰ MODES FLAG or PRG TEST (NXT NXT):		
SF	SF	Sets the flag.
CF	CF	Clears the flag.
FS?	FS?	Returns true (1) if the flag is set and false (0) if the flag is clear.
FC?	FC?	Returns true (1) if the flag is clear and false (0) if the flag is set.
FS?C	FS?C	Tests the flag (returns true (1) if set and false (0) if clear), then clears it.
FC?C	FC?C	Tests the flag (returns true (1) if clear and false (0) if set), then clears it.

Example: Set the Automatic Alpha Lock to activate after only a single press of α (instead of two presses). To do this, set system flag −60, which controls the alpha lock: 60 +/− ↰ MODES FLAG SF.

To set or clear several flags at once:

1. From the stack, enter onto level 1 a list of the flag numbers you wish to set or clear.
2. Do one of the following:
   - To set the flags, press ◀ MODES FLAG SF .
   - To clear the flags, press ◀ MODES FLAG CF .

To recall all flag settings to the stack:

- Press ◀ MODES FLAG NXT RCLF .

This command returns a list containing two 64-bit binary integers representing the current states of the system and user flags. The rightmost (least significant) bit corresponds to system flag -1 or user flag 1.

To reset all flags to their defaults:

- Press ◀ MODES FLAGS NXT RESET.

User Flags

User flags are numbered 1 through 64 and are *not* used by the system. What they mean depends on how a user's program uses them. You can set, clear, and test them the same way as with system flags.

User flags 1 through 5 appear as numbers in the display when they are set.

MODES Submenus

The MODES command menu contains three submenus whose commands act as keyboard shortcuts for changing the states of particular modes, and as programmable commands. Each of these submenus contains special menu labels that indicate the state of the modes represented. When a menu label has a ■ in it, that mode is active.

MODES Operations

Key	Description
Number Format Modes (◀ MODES FMT)	
STD	Sets the display mode to Standard.
FIX	Sets the display mode to Fix using the number on level 1 for the number of decimal places.
SCI	Sets the display mode to Scientific using the number on level 1 for the number of decimal places.
ENG	Sets the display mode to Engineering using the number on level 1 for the number of mantissa digits to be displayed after the first significant digit.
FM,	Switches the fraction mark between period and comma.
ML	Switches between displaying a multiline level 1 as multiple lines (■ in label) and as a single-line followed by an ellipsis.
Angle Measure Modes (◀ MODES ANGL)	
DEG	Sets the angle mode to Degrees.
RAD	Sets the angle mode to Radians.
GRAD	Sets the angle mode to Grads.
RECT	Sets the coordinate mode to Rectangular.
CYLIN	Sets the coordinate mode to Cylindrical.
SPHER	Sets the coordinate mode to Spherical.
Miscellaneous Modes (◀ MODES MISC).	
BEEP	Switches between errors beeping (■ in label) and not beeping.
CLK	Switches between displaying (■ in label) and not displaying a clock.
SYM	Switches between symbolic (■ in label) and numeric evaluation of symbolic expressions.
STK	Switches between saving (■ in label) and not saving the last stack. Affects the action of ▶ UNDO.
ARG	Switches between saving (■ in label) and not saving the last arguments. Affects the action of ▶ ARG.
CMD	Switches between saving (■ in label) and not saving in memory the last command line. Affects the action of ▶ CMD.
INFO?	Switches between displaying (■ in label) and not displaying prompt messages and data automatically.

5

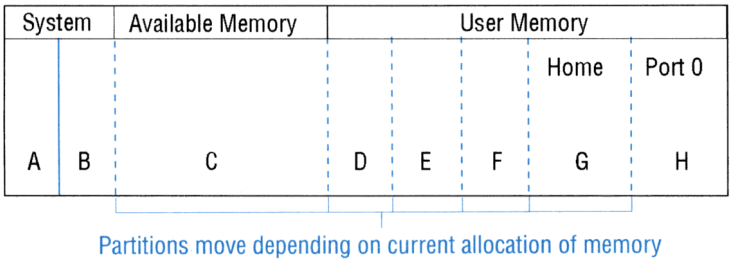
Memory

The HP 48 has two types of memory:

- **Read-only memory (ROM).** ROM is memory that's dedicated to specific operations and cannot be altered. The HP 48 has 512 KB (kilobytes) of built-in ROM, which contains its command set. Except for the HP 48G model, you can expand your calculator's ROM by installing plug-in application cards (described in chapter 28).
- **Random-access memory (RAM).** RAM is memory that you can modify. You can store data into RAM, modify its contents, and purge it. Except for the HP 48G model, you can increase your calculator's RAM by adding memory cards (described in chapter 28).

RAM is also called *user memory* because it's memory that you (the user) have access to. You use or manipulate user memory when you enter an object on the stack, save an object in a variable, delete a variable, create an equation or matrix, run a program, etc. In addition, the HP 48 periodically performs a system cleanup to free memory for use.

The figure on the next page illustrates how the built-in RAM in the HP 48 is organized. Note that the figure is not to scale.



Schematic view of Built-In RAM in the HP 48

After a full memory reset, the calculator memory is returned to the state it was in when shipped from the factory. The only items stored in memory at that point are the built-in system variables (section A). All other memory is available memory (C).

As you work with the calculator, available memory is allocated automatically into the various regions shown in the figure above and described below:

- System Memory: This space is reserved by the RPL system. You have no direct control of this space. It is divided into non-expandable and expandable sections:
  - System Variable Storage (A): Non-expandable section containing the values of all RPL system variables (such as *PICT*) and the current locations of the "boundaries" between the remaining expandable sections of RAM.
  - Temporary System Storage (B): Expandable section containing temporary copies of objects being manipulated and the "return stack" (a list of currently pending operations).
- Available Memory (C): Expandable section containing all non-allocated RAM remaining after System and User Memory is subtracted from the total configured memory.
- User Memory: The memory available for your use. User memory is further divided into five expandable sections:
  - The Stack (D): Contains the objects currently on the stack.

- LAST Variable Storage (E): Contains the three temporary variables—LAST CMD, LAST STACK, and LAST ARG—that save copies of the previous commands, stack, and arguments so that they can be restored if needed. To save memory, you can disable these variables (see page 4-10).
- Local Variables Storage (F): Contains any local variables created by currently-executing programs. *Local* variables exist only while the program executes.
- HOME (G): Contains all named (stored) objects. You can organize and control HOME using the Variable Browser (see page 5-5). Most of the remainder of this chapter describes HOME.
- Port 0 (H): Contains backup objects and libraries that you have stored in Port 0.

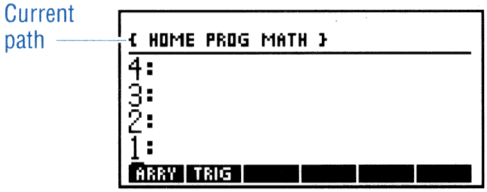
HOME: Variables and Directories

The HOME section of memory works much like a disk does on a personal computer. Each named object in HOME, or HOME *variable*, is analogous to a file on a computer disk.

Like files, HOME variables let you store and retrieve information using meaningful names. For example, you can store the acceleration of gravity, 9.81 m/s^2, into a variable named *G* and then use the name to refer to the variable's contents. Unless you specify otherwise, all variables you create (by name) will be HOME variables.

Also, like files, HOME variables can be arranged hierarchically into *directories*, and organized to suit your particular needs. Note that directory names are stored in variables.

Only one directory (the *current directory*) can be active at a time. The master directory (or *root* directory) for the HP 48 is called the *HOME* directory and is, until you change it, the current directory. The path of the current directory (the *current path*) is shown in the status area of the display.



You always have immediate keyboard access to the variables stored in the current directory. Pressing the VAR key brings up a menu (the VAR menu) of the last six variables you stored into the current directory. (Pressing NXT steps you through additional "pages" of variables.) Since directory names are normally stored in variables, they also appear in the VAR menu, with bars over the top-left corner ("tabs") of their menu labels to show that they are directories.

The *HOME* directory is the only directory that exists when the calculator is turned on for the first time. You create other directories as needed using the Variable Browser.

Where to Store Variables

When you evaluate a variable, the HP 48 searches the current directory for the name of the variable. If the name isn't there, the HP 48 searches up the path until it finds the variable or searches the *HOME* directory. Note that it will evaluate the *first* variable it finds with the designated name, which may or may not be the one you had in mind. This suggests several approaches to organizing your variables:

- Put variables that you want to access from any directory into the *HOME* directory.
- Put variables that you don't want to be able to access from all directories into a directory *not on the current path*.
- You can use duplicate variable names, as long as they are not in the same directory.

Using The Variable Browser Application

The Variable Browser is an application designed to facilitate the way you view and organize the *HOME* memory section and the directories, subdirectories, and variables in it.

To select the Variable Browser, press (→)(MEMORY):

The Variable Browser—Main Screen

This section describes the tasks you can perform using the Variable Browser:

- Creating new variables.
- Selecting variables.
- Editing variables.
- Copying and moving variables.
- Purging variables.
- Finding the size of variables.

Creating New Variables

Variable names can contain up to 127 characters, and can contain letters, digits, and any characters except these:

- Characters that separate objects: space, period, comma, ℮
- Object delimiters # [ ] " ' ⟨ ⟩ ( ) « » : _
- Mathematical function symbols: + − × / ^ √ = < > ≤ ≥ ≠ ∂ ∫ !

Note that uppercase and lowercase letters are *not* equivalent, even though they appear the same in menu labels.

The variable names you use must also follow these restrictions:

- Names can't begin with a digit.
- You can't use the names of commands (for example, SIN, i, or π).

- You can't use the name PICT, which is used by the HP 48 to contain the current graphics object.
- Certain names are legal variable names, but are used by the HP 48 for specific purposes. You can use these names, but remember that certain commands use them as implicit arguments: if you alter their contents, those commands may not execute properly. These variables are called *reserved variables*:
  - *EQ* refers to the equation currently used by the SOLVE and PLOT applications.
  - *CST* contains data for custom menus.
  - *ΣDAT* contains the current statistical matrix.
  - *ALRMDAT* contains the data for an alarm being built or edited.
  - *ΣPAR* contains a list of parameters used by STAT commands.
  - *PPAR* contains a list of parameters used by PLOT commands.
  - *VPAR* contains a list of parameters used by 3D PLOT commands.
  - *PRTPAR* contains a list of parameters used by PRINT commands.
  - *IOPAR* contains a list of parameters used by IO commands.
  - *s1*, *s2*, ... , are created by ISOL and QUAD to represent arbitrary signs obtained in symbolic solutions.
  - *n1*, *n2*, ... , are created by ISOL to represent arbitrary integers obtained in symbolic solutions.
  - Names beginning with "der" refer to user-defined derivatives.

To create a new variable from the Variable Browser:

1. Press (→) MEMORY.
2. Select NEW from the menu.

The NEW VARIABLE screen

3. Enter the new object in the OBJECT: field. You can do this several ways:
   - Key the object into the command line and press ENTER.
   - Use the Equation Writer environment (see chapter 7) to enter an algebraic object.
   - Use the Matrix Writer environment (see chapter 8) to enter an array object.
   - Press CHOOS and select an object.
   - Press NXT CALC, put the desired object on level 1 of the stack, and press OK (see chapter 3).
4. Enter a name into the NAME: field (with or without quotes).
5. Press OK.

Note that if the variable name you use is too long to fit in a menu label, only the beginning of the name will appear in its menu label.

To create a new subdirectory in the current directory:

1. Press → MEMORY.
2. Press NEW.
3. Press ▼ and enter a name into the NAME: field.
4. Highlight the _DIRECTORY check field, and press ✓CHK (or +/–).
5. Press OK.

Selecting, Editing, and Recalling Variables

The Variable Browser makes it easy to select one or more variables from any directory, and to perform operations on several variables at a time. You can also edit existing variables, and recall variables to the stack.

To select a single variable in the current directory:

- Press → MEMORY.
- Use the ▲ and ▼ keys to highlight the desired variable.

To select a group of variables in the current directory:

1. Press → MEMORY.
2. Use the ▲ and ▼ keys to highlight one of the variables. Or, press α and a letter key to highlight the next variable in the current

directory that starts with that letter and repeat until the desired variable is highlighted.

3. Press ✓CHK (or press +/- to include the variable in the group you're selecting).
4. Repeat steps 2 and 3 for each additional variable you want.

Once you've marked the variables you want, you can perform an operation on the entire group at once.

To select variables in a different directory (change current directory):

1. Press → MEMORY.
2. Press CHOOSE to bring up the Directory Browser, which shows the directories and subdirectories in the HOME memory section.

A Typical Directory Browser

3. Use the ▲ and ▼ keys (or α and the first letter of the subdirectory) to highlight the subdirectory you want, and press OK.
4. Select the desired variable or variables.

To edit a variable:

1. Press → MEMORY.
2. Select the variable you want to edit.
3. Press EDIT EDIT.
4. Edit the object using the Edit environment, and press OK OK when you finish.

To recall a variable to the stack:

1. Press (→)(MEMORY).
2. Select the variable you want to recall.
3. Press (NXT) RCL .
4. Exit the Browser when done (press (CANCEL)).

Copying, Moving, and Purging Variables

The Variable Browser also makes it convenient to arrange your variables the way you need.

To copy variables:

1. Press (→)(MEMORY).
2. Select the variable or variables you want to copy.
3. Press COPY .

The Copy Variable Screen

4. Enter one of the following into the COPY TO: field:
   - A new variable name (to store a copy of the selected variable in a new name)
   - An existing variable name (to replace the contents of the named variable with the selected object)
   - A directory path list (to store a copy of the selected variable under the same name but in a different directory)
5. Press OK .

To move a variable:

1. Press ▶ MEMORY.
2. Select the variable(s) you want to move.
3. Press MOVE.

The Move Variable Screen

4. Enter one of the following into the MOVE TO: field:
   - A new variable name (to rename the selected object)
   - An existing variable name (to replace the contents of the named variable with the selected object, and delete the selected object)
   - A directory path list (to move the selected variable to a different directory)
5. Press OK.

To purge variables:

1. Press ▶ MEMORY.
2. Select the variable or variables you want to purge.
3. Press NXT PURG.

Determining the Size of Variables

The Variable Browser lets you find how much storage memory a variable uses.

To determine the size of variables:

1. Press ▶ MEMORY.
2. Select the variable or variables that you want to "measure."
3. Press SIZE from the second page of the menu (press NXT if necessary). You will see a message box similar to the one shown below:

The SIZE Message Box

4. Press OK to remove the message box.

Using Variables: The VAR Menu

The VAR menu provides access to the global variables you've created in the current directory.

While the Variable Browser is most useful for organizing and manipulating the variables you've created, the VAR menu is handy for using variables in calculations, incorporating them in equations, and as a shortcut for a number of common memory tasks:

- **Create a new variable.** Put the desired object on level 1 of the stack, type the name of the variable, and press STO. The new variable is placed in the current directory and displayed in the VAR menu.
- **Evaluate a variable.** Press the variable's VAR menu key.
- **Recall the contents of a variable to the stack.** Press ▶ followed by the variable's VAR menu key.
- **Recall the name of a variable to the stack.** Press ' followed by the variable's menu key.

- **Update the contents of a variable.** Put the newly edited contents of the variable on stack level 1 and press ◀ followed by the variable's menu key.
- **Purge a variable (name and contents) from memory.** Recall the name of the variable to the stack and then press ◀ PURG.
- **Purge a group of variables at once.** Put a list (with { } delimiters) containing the *unquoted* names of the variables to be purged on stack level 1, and press ◀ PURG.
- **Include the name of a variable in an algebraic or program.** Assuming that the appropriate quote marks are already entered (tick-marks for algebraics or guillemots for programs), press the variable's VAR menu key.
- **Switch to the HOME directory.** Press ▶ HOME.
- **Switch to the parent directory.** Press ◀ UP.

Example: Explore the use of the VAR menu using a variable named *OPTION* containing 6.05.

Step 1: Create the variable, *OPTION* and display the VAR menu.

6.05 ENTER ' α α
OPTION α STO VAR

Step 2: Recall the value of the variable.

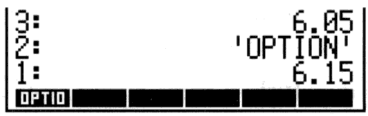
OPTIO

Step 3: Recall the name of the variable.

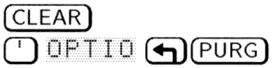
' OPTIO ENTER

Step 4: Change the value in *OPTION* to 6.15. Recall the contents again to confirm the change.

6.15 ◀ OPTIO
OPTIO



Step 5: Clear the stack and purge *OPTION* from memory.



Defining Variables

The HP 48's DEFINE command can create variables from equations (see chapter 7 for information about creating equations). If stack level 1 has an equation with this form, '*name* = *expression*', executing DEFINE stores that expression in that name.

To create a variable from a symbolic definition:

1. Enter an equation of the form '*name* = *expression*'.
2. Press ← DEF (the DEFINE command).

Example: Use DEFINE to store M*C^2 in the variable *E*.

Step 1: Press ' α α E ← = M × C α y^x 2 ENTER.

Step 2: Press ← DEF.

Note that if flag −3 is clear (its default state), DEFINE stores the expression without evaluation. If you've set flag −3, the expression to be stored is evaluated to a number, if possible, before it's stored. For example, the keystrokes 'A=10+10' ← DEF create variable *A* and store '10+10' in it if flag −3 is clear, and 20 if flag −3 is set.

Evaluating Variables

To use a variable's *contents* in a calculation, the variable must be *evaluated*. This is accomplished by pressing the variable's menu key in the VAR menu.

Evaluating a variable name calls the object stored in the variable:

- **Name.** The name is evaluated (calling *its* object).
- **Program.** The program runs.
- **Directory.** The directory becomes the current directory.
- **Other Object.** A copy of the object is returned to the stack.

Example: Suppose that there are four variables in the current directory—*A* containing 2, *B* containing 5, *ALG* containing the expression 'A+B', and *ADD2* containing

the short program « + + ». Evaluate them from the VAR menu.

Step 1: From the stack, display the VAR menu.

VAR

Step 2: Evaluate *ALG*, *B*, and *A*. Because none of these variables contain programs or directories, their contents are put on the stack.

ALG
B
A

Step 3: Evaluate *ADD2*. Notice that the program is executed, not merely put onto the stack.

ADD2

Quoted Variable Names and Formal Variables

The ' delimiter is very important when you enter a variable name: it determines whether the name is automatically evaluated when you press ENTER. If the ' delimiter is present, the name is not evaluated.

To enter a variable name onto the stack:

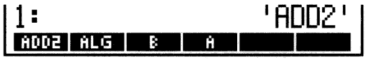
- If a variable with that name exists (or might exist), press ', then type the name or press its VAR menu key. Variable names included as a part of an algebraic expression are within quotes and are not evaluated until the algebraic is evaluated.
- If no variable with that name exists, type in the unquoted name and press ENTER. Because the HP 48 can't find an object associated with the new name, the name is treated as a *formal* variable and put on the stack *with quotes*. Evaluating a formal variable simply returns the formal variable name again.

Although they appear identical on the stack, there are two important differences between formal variable names (that have no associated objects) and quoted variable names (that have associated objects):

- Evaluating a formal variable doesn't appear to do anything, because the formal variable is returned again to the stack. Evaluating the quoted name of a variable containing an object evaluates that object.
- Formal variables never appear in the VAR menu. All variables that appear in the VAR menu have an associated object. However, you *can* store a formal variable name in a VAR variable with a *different* name.

Example: Enter the name ADD2 onto the stack using single-quotes.

' α α ADD2 ENTER



Example: Enter the formal variable name, C, onto the stack by using the unquoted name. If an actual variable, C exists somewhere in the current path, you will see its contents instead of the variable name.

α C ENTER

Example: Store the formal variable 'C' in the variable *C2*. Then evaluate C2 using the VAR menu.

Step 1: Store 'C' in *C2*.

← CLEAR → MEMORY
NEW α C ENTER α C2
ENTER OK NXT OK

Step 2: Evaluate C2 using the VAR menu. Confirm that 'C' is a formal variable by pressing EVAL.

VAR C2

Special Memory Operations

Occasionally the HP 48 may seem to freeze while executing something, and won't respond to the CANCEL key. This can happen if the memory becomes corrupted, or if the system becomes "confused" during execution.

If this happens, you have two ways to try to correct it: a *system halt* and a *memory reset*.

Caution

If you must reset your calculator, always attempt a system halt first. Consider a memory reset only if the system halt fails.

System Halt

A system halt does the following:

- Interrupts and cancels all executing programs and system operations.
- Clears the stack, all local variables, the three LAST variables, the PICTURE display, and the temporary system storage section of memory.
- Turns off the user keyboard (clears flag -62).
- Detaches all libraries from the HOME directory, and reconfigures all libraries in all available ports (see "Using Libraries" in chapter 28 for details).
- Makes the HOME directory the current directory.
- Activates the main MTH menu.

Note that a system halt does *not* affect the stored objects in HOME and Port 0.

To halt the system from the keyboard:

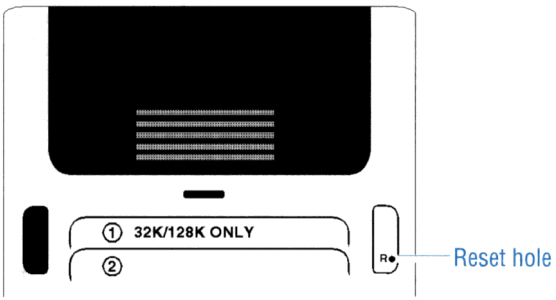
1. Press and hold ON.
2. Press the C menu key.
3. Release both keys.

A system halt is also performed automatically when you turn on the calculator *if* you have added, removed, or changed the write-protect switch position of any plug-in card since you last turned the calculator on.

Sometimes the HP 48 may freeze up and *won't respond to* ON-C because it isn't accepting input from the keyboard. At these times, you need to execute the system halt directly, without using the keyboard.

To halt the system without using the keyboard:

1. Turn the calculator over and remove the top right (as you face the back side of the machine) rubber foot. You should see a small hole with the letter R next to it.



2. Insert the end of an ordinary metal paper clip into the hole as far as it goes. Hold for one second and remove.
3. Press ON.
4. If necessary, press ON-C. If this still doesn't work, you need to try a memory reset.

Memory Reset

A *memory reset* can reset the HP 48 to its factory-default state, *erasing all information you've stored. Use this function very carefully.*

To perform a memory reset:

1. Press and hold down these three keys simultaneously: ON, the A menu key, and the F menu key.
2. Release the two menu keys, but continue to hold ON:
   - If you want to *continue* with the memory reset, release ON.
   - If you want to *cancel* the memory reset, press the B menu key, then release ON.

Once you've initiated the memory reset, the calculator beeps and displays the following screen:

Memory Reset Prompt

3. Press YES if you want to try and recover the variables you have stored in HOME and in Port 0. There is no guarantee that all of the variables can be recovered. Press NO to perform a complete memory reset. This restores the HP 48 to its factory-default state and clears all user memory.

Responding to Low-Memory Conditions

HP 48 operations share memory with the objects you create. This means that the calculator may operate slowly or even fail if user memory becomes too full. If this happens, the HP 48 returns one of a series of low memory warnings. These messages are described below in order of increasing severity.

- No Room for Last Stack—If there's not enough memory to save a copy of the current stack, this message appears when ENTER is executed. The UNDO operation is disabled when this message appears.

  Remedy: Delete unused variables or unnecessary objects from the stack.

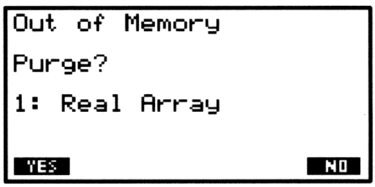
- Insufficient Memory—This appears if there isn't enough memory to completely execute an operation. If the LASTARG command is enabled (flag −55 is clear), the original arguments are restored to the stack. If LAST ARG is disabled (flag −55 is set), the arguments are lost.

  Remedy: Delete unused variables or unnecessary objects from the stack.

- No Room To Show Stack—This message appears when the HP 48 completes all pending operations but doesn't have enough free memory to display the stack. When this occurs, the stack displays objects only by type: Real Number, Algebraic, and so on. The amount of memory required to display a stack object varies with the object type.

  Remedy: Delete unnecessary variables and unneeded objects from the stack, or store stack objects in variables so they don't have to be displayed.

- Out of Memory—In extreme cases, the calculator runs completely out of memory and is unable to do anything. In this situation you *must* clear some memory before you can continue. The HP 48 activates a special procedure to do this, and shows this display:



When the procedure starts, the HP 48 asks whether you want to purge the object (described by object type) in level 1 (a real array in the above figure). If you delete it, the calculator then asks about the new level 1 object. This continues until the stack is empty or you respond to the display by pressing NO. The calculator then asks if you want to discard the contents of LAST CMD, and then asks in the following order if you want to delete other items:

1. Stack level 1 (repeated)
2. The contents of LAST CMD
3. The contents of LAST STACK (if active)
4. The contents of LAST ARG (if active)
5. The variable *PICT* (if present)
6. Any user-key assignments
7. Any alarms
8. The entire stack (unless already empty)
9. Each global variable by name
10. Each port 0 object by tagged name

To respond to the "Out Of Memory" prompts:

- To delete the indicated object, press YES .
- To keep the indicated object, press NO .
- To stop the procedure and see if the condition is fixed, press CANCEL.

Note

The purge sequence may begin with the command line and *then* cycle through the stack, the contents of LAST CMD, etc. If you answer NO to the purge prompt for the command line, you'll return to the command line when you terminate the Out of Memory procedure.

The prompts for global variables start with the newest object in the *HOME* directory and proceed to successively older objects. If the variable to be purged is an empty directory, YES purges it. If the directory is not empty, YES cycles the variable-purge sequence through the variables (from newest to oldest) in that directory.

Whenever you like, you can try to terminate the Out of Memory procedure by pressing CANCEL. If sufficient memory is available, the calculator returns to the normal display; otherwise, the calculator beeps and continues with the purge sequence. After cycling once through the choices, the HP 48 attempts to return to normal operation. If there still is not enough free memory, the procedure starts over.

6

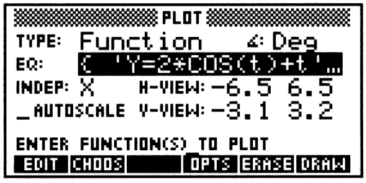
Input Forms and Choose Lists

Although the HP 48 screen is small compared to a standard computer screen, it is approximately the size of the average "dialog box." *Input forms* are the HP 48 equivalent of these dialog boxes.

Most of the HP 48's applications have input forms associated with them that make it easy to remember the information you need to enter, and to set any options you want.

Input Forms

All input forms have a similar appearance. The figure below uses the main input form from the PLOT application to illustrate the main components.



A Sample Input Form: PLOT

Each input form has a *title*, a set of *fields* (some with *labels*) a *prompt line* (just above the menu), and a menu that displays choices relevant to the currently selected field (the prompt line also displays a message pertaining to the current field). As you switch fields, the prompt and the menu change to reflect the newly-selected field.

Input forms use four basic kinds of fields:

- **Data fields.** These accept data of a particular kind directly from the keyboard. The fields labeled INDEP:, H-VIEW:, and V-VIEW in the PLOT form are examples of data fields.
- **Extended data fields.** These fields extend the abilities of data fields by also allowing you to enter an object you have previously stored (as long as it is an allowable type for that field). The EQ: field in the PLOT form is an example of an extended data field.
- **List fields.** These fields have a limited, pre-determined set of possible values, of which you must choose one. The TYPE: and ∠: fields in the PLOT form are examples of list fields.
- **Check fields.** These fields control various options in the applications (a check in the field makes the option active). The AUTOSCALE field in the PLOT form is an example of a check field.

Selecting Fields in Input Forms

The cursor keys are active in input forms and are the primary means of selecting fields:

▶ Selects the next field, moving from left to right and from top to bottom. From the last field on the form, ▶ "wraps around" and selects the first field at the top of the form.

◀ Selects the previous field. From the first field on the form, ◀ "wraps around" and selects the last field at the bottom of the form.

▲ Selects the corresponding field on the previous line. From a field on the top line of the form, ▲ "wraps around" and selects the corresponding field on the bottom line of the form.

▼ Selects the corresponding field on the next line. From a field on the bottom line of the form, ▼ "wraps around" and selects the corresponding field on the top line of the form.

↱◀ Selects the first field on the form.

↱▲ Selects the first field on the form.

↱▶ Selects the last field on the form.

↱▼ Selects the last field on the form.

When you press ENTER or OK to enter data that has been typed into the command line, the next field is automatically selected. Otherwise, you must move the selection bar using the cursor keys.

Entering Data in Input Forms

The HP 48 provides several ways to enter data into input forms.

To enter information into a data field:

1. Select the data (or extended data) field.
2. Type in the object. The command line is available for all kinds of objects (remember to use the proper delimiters, however). You can also use the EquationWriter for algebraic objects (see chapter 7), or the MatrixWriter for arrays (see chapter 8). To change to the Equation or Matrix Writer, see "To make a side-trip to a second input form" on page 6-5.
3. Press ENTER or OK.

To enter a previously stored object into an extended data field:

1. Select the extended data field.
2. Press CHOOS. A miniature version of the Variable Browser appears, containing all variables in the current directory that can be used by the selected field.
3. Use the ▲ and ▼ arrows to highlight the object you want.
4. Press ENTER or OK.

Some extended data fields also allow you to enter multiple objects, grouped into a list.

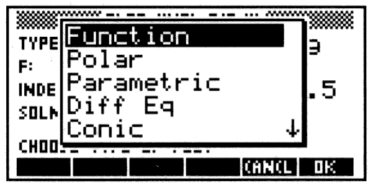
To enter a list of objects into an extended data field:

1. Select the extended data field that accepts list objects.
2. Press CHOOS. A miniature version of the Variable Browser appears, containing all variables in the current directory that can be used by the selected field.
3. Use the ▲ and ▼ arrows to select an object that belongs in the list.
4. Press ✓CHK to display a check-mark next to the object.
5. Repeat steps 3 and 4 for the other objects in the list.
6. Once all objects in the list have checks, press ENTER or OK.

Selecting Options in Input Forms

To select an option from a list field:

1. Select the list field.
2. Select an option for that field using one of the following methods:
   - Use a *pop-up list*.
     a. Press CHOOS to bring up a *pop-up list* of available options.



A Sample Pop-Up List: PLOT Types

     b. Use the ▲ and ▼ arrow keys to select an option.
     c. Press ENTER or OK.
   - Press +/- repeatedly to cycle through the options. Stop when the option you want appears.
   - Press α followed by the first letter of the desired option. The next option beginning with that letter appears. If more than one option begins with the same letter, you may have to repeat the procedure once or twice until the option you want appears.

To select an option in a check field:

1. Select the check field.
2. Do one of the following:
   - Press ✓CHK once or twice to check or uncheck the field.
   - Press +/- once or twice to check or uncheck the field.

Other Input Form Operations

To edit a data field:

1. Select the data (or extended data) field.
2. Press EDIT (or ↰ EDIT). This copies the object into the command line.

3. Edit the object using the standard command line editing procedures.
4. Press (ENTER) or OK .

To make a "side" calculation while in the input form:

1. Select the data (or extended data) field.
2. Press (NXT) CALC . A version of the stack appears (notice that both the form title and the field prompt are still visible), and any object that was in the selected field is now on level 1. If desired, press STS to show or hide the status line.
3. Perform the stack calculation you need, entering additional objects or selecting commands from other menus. The result you want to enter into the data field must be on level 1 when you finish.
4. If OK isn't visible on the menu (because you have used other menus), press (◀)(CONT) to make it visible again.
5. Press OK to enter the calculated result into the selected data field, or CANCL to return without entering the result.

To make a "side-trip" to a second input form:

1. Select a data (or extended data) field.
2. Press (NXT) CALC . A version of the stack appears (notice that both the form title and field prompt are still visible), and any object that was in the selected field is now on level 1.
3. Open the second input form.
4. Complete your task(s) in the second input form and exit it by pressing OK or CANCL, or by executing a task that exits that particular form.
5. If OK isn't visible on the menu (because you have used other command menus), press (◀)(CONT) to make it visible again.
6. Make sure the object on level 1 of the stack is what you want stored in the selected data field of the original form. (It may have changed, depending on what you did in the second input form.)
7. Press OK to return to the original input form and enter the level 1 object into the selected data field in the input form, or CANCL to return without entering the level 1 object.

To reset the value of a field to its default:

1. Select the field.
2. Do one of the following:
   - Press (NXT) RESET.
   - Press (DEL).
3. Select Delete value (or Reset Value) from the pop-up list.
4. Press (ENTER) or OK.

To reset the values of all fields to their defaults:

1. Select a field.
2. Do one of the following:
   - Press (NXT) RESET.
   - Press (DEL).
3. Select Reset all from the pop-up list. In some applications (PLOT is one) the phrasing will differ slightly, because one or more fields must be reset individually (as a precaution against unintentional data loss).
4. Press (ENTER) or OK.

To determine which object types are valid for a data field:

1. Select the data (or extended data) field.
2. Press (NXT) TYPES. A pop-up window display the types of objects you can use in that field.

A Sample TYPES Pop-Up List

3. Do one of the following:
   - To begin entering a particular data type, select the type in the pop-up list and press NEW. The appropriate delimiters appear in the command line.
   - If you don't want to enter data, press OK.

When You Finish Entering Data into an Input Form

Input forms are designed to help you enter data and prepare to execute a larger task. The data you enter and options you select may either be used only within the context of the particular input form and its tasks, or they may cause *global* changes throughout all applications. Changes to reserved variables (such as *EQ*) and system flags are examples of global changes.

Whether or not these global changes are saved depends on how you exit the input form. The following procedures illustrate your options:

To execute the main action of an input form:

1. Make sure the necessary data is entered and proper options checked.
2. Press the appropriate action key (particular to each form). Global changes are saved and the action executed, displaying the appropriate screen. This does not usually cause you to exit the input form.

To exit the input form after saving any global changes:

- Press OK from the menu. Sometimes it is shown only on the second page (press NXT), if necessary.)

To exit the input form and discard any global changes:

- Press CANCEL or CANCL. Sometimes CANCL is shown only on the second page (press NXT), if necessary.)

To exit SOLVE input form and open PLOT input form (and vice-versa):

- Open the new input form. Global changes are saved and all non-global data is discarded before the current input form is exited and the new input form opened.

Input Form Commands

The HP 48 has several programmable commands you can use to create your own customized input forms. These commmands, located in the PRG IN command menu, are discussed briefly in appendix G, and in detail in the *HP 48G Series Advanced User's Reference*.

To create an input form:

1. Enter a title string for the input form (use → " ").
2. Enter a list of field specifications. If more than one field is specified, enclose each field specification in brackets.
3. Enter a list of format options.
4. Enter a list of reset values (values that appear when RESET is pressed).
5. Enter a list of default values.
6. Execute the INFORM command.

Example: Create an input form.

Enter a title, a field specification, format options, a blank list for reset values, and a default value.

→ " " α α FIRST SPC
ONE ENTER ← {} → " "
α α N ← α ame → ::
ENTER ← {} 1 SPC 5
ENTER ← {} ENTER ← {}
→ " " α α WENDY
ENTER
PRG NXT IN INFOR

7

The EquationWriter

The HP 48 features the EquationWriter application, which makes it easy to enter and review algebraic expressions and equations in the form most familiar to you—the way you usually see them in print, and the way you write them with pencil and paper.

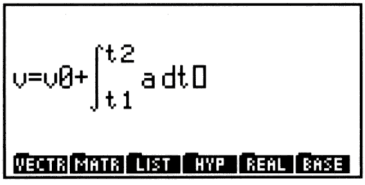
For example, here's a physics equation:

$$v = v_0 + \int_{t_1}^{t_2} a \; dt$$

Here's how the equation would look on the stack:

```
'v=v0+∫(t1,t2,a,t)'
```

Now, here's the same equation keyed in using the EquationWriter application:



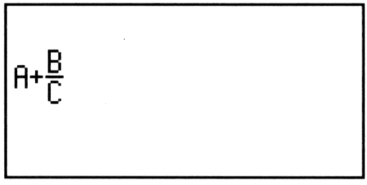
How the EquationWriter Application Is Organized

In the EquationWriter application, keys corresponding to algebraic functions enter the function name or graphical function symbol into the equation. For example, pressing [√x] draws a square root sign. You can display any command menu—however, only those keys that correspond to algebraic functions are active. Like the function keys on the keyboard, the menu keys don't execute the corresponding function—they simply enter the function name into the equation.

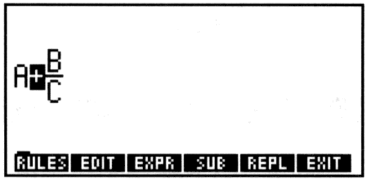
The EquationWriter application consists of three modes, each with a special purpose:

- **Entry mode**—for entering and editing equations
- **Scrolling mode**—for viewing large equations
- **Selection mode**—for editing expressions within equations

Entry Mode



Scrolling Mode



Selection Mode

Constructing Equations

To start the EquationWriter application:

- Press ◀ EQUATION. The EquationWriter is accessible from the stack or from any field in an input form that can accept algebraic objects.

After you start the EquationWriter application, you can enter an equation or expression (or unit object, number, or name) using the operations available in this environment. See "Entering Equations" below.

To exit the EquationWriter application:

- To put the equation on the stack and exit, press ENTER.
- To discard the current equation and exit, press CANCEL.

Entering Equations

Occasionally the EquationWriter may not be able to display your equation as fast as you type it. However, you can continue typing, and the HP 48 will remember up to 15 keystrokes and display them when it "catches up."

To enter numbers and names:

- Key numbers and names in exactly as you would into the command line. You can also use the menu keys in the VAR menu as typing aids for variable names.

To include addition, subtraction, and multiplication:

- To enter +, −, and ·, press +, −, and ×.
- To do *implied* multiplication, don't press ×. You can do implied multiplication (without pressing ×) in some situations—a multiply sign (·) is automatically inserted between the following:
  - A number followed by an alpha character, a parenthesis, or a prefix function (a function whose argument(s) appear after its name)—for example, press 6 SIN.
  - An alpha character and a prefix function—for example, A ◀ x^2.
  - A right parenthesis followed by a left parenthesis.

- A number or alpha character and the divide bar, square-root symbol, or xth root term—for example, B ▲.

Note

All multiplication (even implied multiplication) should display a multiplication operator (* or ·). In particular, an expression such as X(Y+Z) contains *no* multiplication. The form, X( ), is a user-defined function (see page 11-6), whose parentheses contain its argument. By contrast, expressions such as X*(Y+Z) or X·(Y+Z) do include valid multiplications.

7

To include division and fractions:

1. Press ▲ to start the numerator.
2. Press ▶ to end the numerator and start the denominator (▼ works too).
3. Press ▶ to end the denominator.

Here's another way to type in fractions whose numerator consists of either *one* term or a sequence of terms with operators of precedence greater than or equal to that of division:

1. Type the numerator (without pressing ▲).
2. Press ÷ to start the denominator.
3. Press ▶ to end the denominator (▼ works too).

To include exponents:

1. Press y^x to start the exponent.
2. Press ▶ to end the exponent (▼ works too).

To include roots:

- To include a square root, press $\sqrt{x}$ to draw the √ symbol and start the term, then press ▶ to end the term.
- To include an xth root, press ↱ $\sqrt[x]{y}$ to start the x term (outside the √ symbol), press ▶ to draw the √ symbol and start the y term inside the √ symbol, then press ▶ to end the xth root term.

To include functions with parenthetical arguments:

1. Press the function key, or type the name and press ↰ ( ).

2. Press ▶ to end the argument and display the right parenthesis.

To include parenthesized terms:

1. Press ↰ ( ) to display the left parenthesis.
2. Press ▶ to end the term and display right parenthesis.

To include powers of 10:

1. Press EEX to display E.
2. If the power is negative, press +/– to display –.
3. Key in the digits of the power.
4. Press any function key to end the power.

To include derivatives:

1. Press ↱ ∂ to display $\frac{\partial}{\partial}$.
2. Key in the variable of differentiation, then press ▶ to end the differentiation term and display the left parenthesis.
3. Key in the expression.
4. Press ▶ to end the expression and display the right parenthesis.

To include integrals:

1. Press ↱ ∫ to display the integral symbol ∫ with the cursor positioned at the lower limit.
2. Key in the lower limit and press ▶.
3. Key in the upper limit and press ▶.
4. Key in the integrand and press ▶ to display d.
5. Key in the variable of integration.
6. Press ▶ to complete the integral.

To include summations:

1. Press ↱ Σ to display the summation symbol Σ with the cursor positioned below.
2. Key in the summation index.
3. Press ▶ (or ↰ =) to key in the equal sign.
4. Key in the initial value of the index and press ▶.
5. Key in the final value of the index and press ▶.
6. Key in the summand.
7. Press ▶ to end the summation.

To include units:

1. Key in the number part.
2. Press ▶ _ to start the unit expression.
3. Key in the unit expression.
4. Press ▶ to end the expression.

7 You can also build unit objects (described in chapter 10) in the EquationWriter application. For compound units, press × or ÷ to separate each individual unit in the unit expression. You can key in unit names in one keystroke by pressing the corresponding menu key in the UNITS Catalog menu.

To include | (where) functions:

1. Key in a parenthetic expression with symbolic arguments.
2. Press ◀ SYMBOLIC NXT | to display |. The cursor is positioned at the bottom right of the symbol.
3. Key in the defining equation for each argument, pressing ▶ or ◀ = to key in =, and SPC to key in the separator between each equation.
4. Press ▶ to end the function.

The | (where) function substitutes values for names in expressions. It's described under "Showing Hidden Variables" on page 20-17.

Controlling Implicit Parentheses

Implicit parentheses are turned on whenever you start the EquationWriter application. This means the arguments for ÷, $\sqrt{x}$, and y^x are normally enclosed in "invisible" parentheses, so that only ▶ (or ▼) ends the argument.

If you turn off implicit parentheses, the argument ends when you enter the next function—or press ▶.

To turn implicit parentheses on or off:

- Press ◀ { }. A message briefly displays the current state.

Disabling implicit parentheses is convenient for entering polynomials, for example, where exponents are completed when you enter the function that starts the next term.

Leaving and then restarting the EquationWriter application turns implicit parentheses on. If you turn off implicit parentheses after keying in ÷, √x, or y^x, but before supplying the argument, implicit parentheses are *not* applied to those arguments.

Example: Key in the expression $X^3 + 2X^2 - \frac{1}{X}$, first with implicit parentheses and then without.

Step 1: Key in the expression with implicit parentheses on (the default).

↰ EQUATION α X y^x 3 ▶
\+ 2 α X y^x 2 ▶
− 1 ÷ α X

Step 2: Clear the display and turn off implicit parentheses.

↰ CLEAR
↰ {}

Step 3: Key in the expression again.

α X y^x 3 \+ 2 α X y^x 2 −
1 ÷ α X

Press ↰ {} to turn on implicit parentheses again.

EquationWriter Examples

At the end of each of the following examples, you can either press ENTER to put the equation on the stack, or press ◀ CLEAR to clear the display for the next example. If you do the latter, ignore the ◀ EQUATION instruction at the start of each new example.

7

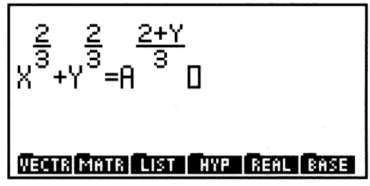
If you make a mistake while you're keying in an equation, press ◀ to backspace to the error, or press ▶ CLEAR and start again.

Example: Key in this equation:

$$X^{\frac{2}{3}} + Y^{\frac{2}{3}} = A^{\frac{2+Y}{3}}$$

Step 1: Key in the equation.

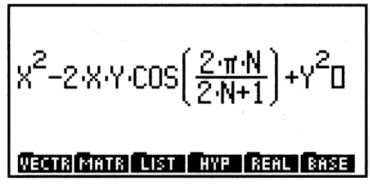
◀ EQUATION
α X y^x 2 ÷ 3 ▶ ▶ +
α Y y^x 2 ÷ 3 ▶ ▶ ◀ =
α A y^x ▲ 2 + α Y ▶ 3
▶ ▶



Example: Key in this expression:

$$X^2 - 2XY\cos\frac{2\pi N}{2N+1} + Y^2$$

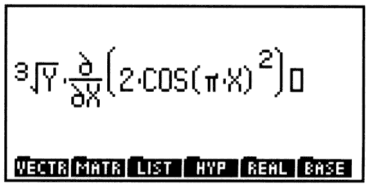
◀ EQUATION
α X y^x 2 ▶ −
2 α X × α Y COS
2 ◀ π × α N ÷
2 α N + 1 ▶ ▶
+ α Y y^x 2 ▶



Example: Key in this expression:

$$\sqrt[3]{Y}\,\frac{d}{dX}2\cos^2(\pi X)$$

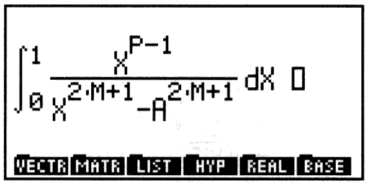
EQUATION
$\sqrt[x]{y}$ 3 ▶ α Y ▶
∂ α X ▶
2 COS ◀ π × α X ▶
y^x 2 ▶ ▶



Example: Key in this expression:

$$\int_0^1 \frac{X^{P-1}}{X^{2M+1} - A^{2M+1}} dx$$

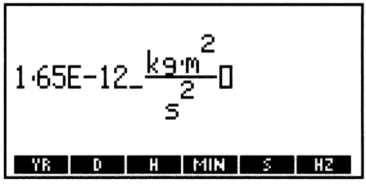
EQUATION
∫ 0 ▶ 1 ▶
α X y^x α P − 1 ▶ ÷
α X y^x 2 α M + 1 ▶
− α A y^x 2 α M + 1 ▶
▶ ▶ α X ▶



Example: Key in this expression:

$$1.65 \times 10^{-12} \frac{\text{kg} \cdot \text{m}^2}{\text{s}^2}$$

EQUATION
1.65 EEX +/− 12 ▶ _
▶ UNITS MASS KG ×
▶ UNITS AREA M^2 ▶
÷ ▶ UNITS TIME S
y^x 2 ▶ ▶



Editing Equations

The EquationWriter application provides several options for editing equations:

- Backspace editing
- Editing a full expression in the command line
- Editing a subexpression in the command line
- Inserting an object (subexpression) from the stack into the equation
- Replacing a subexpression with an algebraic from the stack

To edit by backspacing:

1. Press ◀ until you delete the error.
2. Complete the expression correctly.

To edit the full equation:

7

1. If the equation ends in an incomplete subexpression, complete it.
2. Press ↰ EDIT.
3. Edit the equation in the command line.
4. Press ENTER to save the changes (or press CANCEL to discard them) and return to the EquationWriter application.

To view a large equation or unit object:

1. Press ↰ PICTURE to activate scrolling mode.
2. Press ◀ ▶ ▲ ▼ to move the viewing "window."
3. Press ↰ PICTURE to return to the previous mode.

Editing with Subexpressions

The *Selection environment* is a special part of the EquationWriter application used to specify a subexpression in the equation.

A *subexpression* consists of a function and its arguments. The function that defines a subexpression is called the *top-level* function for that subexpression. The top-level function is essentially the *last* function to be evaluated following the usual rules of algebraic precedence.

For example, in the expression `'A+B*C/D'`, the top level function for the subexpression `'B*C'` is `*`, the top-level function for `'B*C/D'` is `/`, and the top level function for `'A+B*C/D'` is `+`. You can specify an individual object (a name, for example) as the subexpression.

You can also use the Selection environment to specify a subexpression to rearrange using the Rules transformations—see "Manipulating Subexpressions" on page 20-19.

To edit a subexpression of an equation:

1. If the equation ends in an incomplete subexpression, complete it.
2. Press ◀ to activate the Selection environment.
3. Press ▲ ▼ ◀ ▶ to move the selection cursor to the *top-level function* for the subexpression you want to edit.

4. Optional: Press EXPR at any time to highlight the current subexpression. (Press it again to turn off the highlight.)
5. Press EDIT to put the current subexpression in the command line.
6. Edit the subexpression in the command line.
7. Press ENTER to enter the revised subexpression into the equation (or press CANCEL to discard it).
8. Press EXIT to leave the Selection environment. (If EXIT isn't displayed, press ◀ to return to the Selection menu.)

To insert an object from level 1 into an equation:

1. Create the object to be inserted and put it on level 1. The object can be a name, a real number, a complex number, an algebraic, or a string.
2. Open the EquationWriter and begin creating the equation.
3. Press ▶ RCL to insert the level 1 object at the cursor in the EquationWriter expression.

Example: Enter this expression:

$$\int_0^{10} x^2 - y\, dx + \frac{x^2 - y}{2}$$

Step 1: Enter the expression `'X^2-Y'` into level 1 and duplicate it.

' α X y^x 2 − α Y ENTER
ENTER

Step 2: Select the EquationWriter application and key in the integral sign and limits of integration.

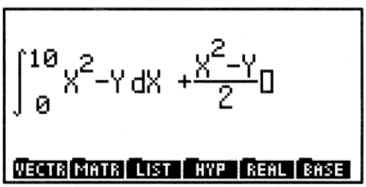
◀ EQUATION
▶ ∫ 0 ▶ 10 ▶

Step 3: Insert the integrand into the expression.

→ RCL

Step 4: Complete the subexpression. Then key in the remainder of the expression, inserting the second term from the stack.

▶ α X ▶
+ ▲ → RCL ▶ 2 ▶



To replace a subexpression with an algebraic from level 1:

1. If the equation ends in an incomplete subexpression, complete it.
2. Press ◀ to activate the Selection environment.
3. Press ▲ ▼ ◀ ▶ to move the selection cursor to the *top-level function* for the subexpression you want to replace. (See "Editing with Subexpressions" on page 7-10.)
4. Optional: Press EXPR any time to highlight the associated subexpression (press EXPR again to turn the highlight off).
5. Press REPL.
6. Press EXIT to leave the Selection environment.

The algebraic is deleted from the stack.

Summary of EquationWriter Operations

Operations in the EquationWriter Application

Key	Description
▲	Starts a numerator.
▶ or ▼	Ends a subexpression. →▶ or →▼ ends all pending subexpressions.
◀	Invokes *selection* mode, in which the Selection environment is active.
←( )	Starts a parenthesized term. ▶ or ▼ ends the term.
SPC	Enters the current separator (, or ;) for multiple parenthetical arguments of functions and the terms of complex numbers.
EVAL	Exits the EquationWriter application and evaluates the equation.
ENTER	Returns the equation to the stack and exits the EquationWriter application.
CANCEL	Exits the EquationWriter application without saving the equation.
←PICTURE	Toggles *scrolling* mode. In scrolling mode, the menu keys are erased; if the equation is larger than the display, ▲ ▼ ◀ ▶ scroll the display window over the equation in the indicated direction. Press ←PICTURE again (or CANCEL) to return to the previous mode. (Exception: Pressing ▼ with an algebraic on the stack, starts the EquationWriter in scrolling mode and when you exit—with CANCEL or ←PICTURE—you invoke the selection mode.)
←EDIT	In entry mode, returns the equation to the command line for editing.
STO	Returns the equation to the stack as a *graphic object*. (See chapter 9 for more details about graphic objects.)

Operations in the EquationWriter Application (continued)

Key	Description
◀ CLEAR	Erases the display without leaving the EquationWriter application.
▶ RCL	Inserts the level 1 object into the equation at the cursor position. (See "Editing Equations" on page 7-9.)
◀ {}	Turns *implicit parentheses* mode off. Press ◀ {} again to turn implicit parentheses mode back on. (See "Controlling Implicit Parentheses" on page 7-6.)
▶ " "	Returns the equation to the stack as a string.

8

The MatrixWriter

The HP 48's MatrixWriter application provides you with extensive capabilities for entering and manipulating arrays (both one-dimensional matrices and two-dimensional matrices).

How the HP 48 Displays Arrays

The stack displays arrays as numbers within nested [ ] delimiters. A pair of [ ] delimiters enclose the entire array, and additional pairs enclose each row within the matrix. For example, here's a 3 × 3 matrix as it might appear on the stack:

```
[[ 1 2 3 ]
 [ 3 4 5 ]
 [ 7 8 9 ]]
```

Vectors (also called column vectors or one-column matrices) appear on the stack as numbers within a single level of [ ] delimiters:

```
[ 2 4 6 8 ]
```

Row vectors (one-*row* matrices) appear on the stack as numbers within *two* pairs of [ ] delimiters:

```
[ [ 1 3 5 7 9 ] ]
```

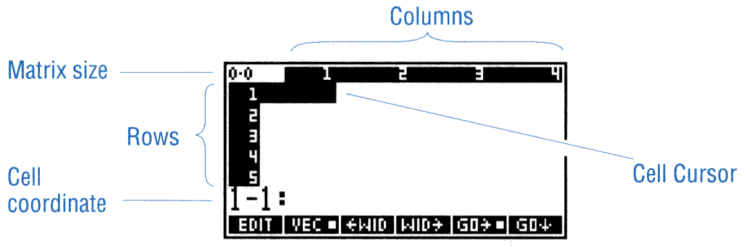
In addition, the current coordinate mode and angle mode affect how 2-dimensional and 3-dimensional vectors are displayed. See "Displaying 2D and 3D Vectors" on page 13-1 for details.

Entering Arrays

The MatrixWriter application provides a special environment for entering, viewing, and editing arrays. You can access the Matrix Writer by selecting it from the stack or from any field in an input form that accepts array objects.

8

The MatrixWriter display shows array elements in individual cells arranged in rows and columns.



To enter a matrix using the MatrixWriter application:

1. Press (→) (MATRIX).
2. Key in the numbers in the first row, and press (ENTER) after each.
3. Press (▼) to mark the end of the first row.
4. Key in the numbers in the rest of the matrix, and press (ENTER) after each. Notice that when you enter the last number in the row, the cursor automatically moves to the beginning of the next row.
5. After you've entered all the numbers in the matrix, press (ENTER) to put the matrix on the stack.

Example: Enter this matrix:

$$\begin{bmatrix} 2 & -2 & 0 \\ 1 & 0 & 3 \\ -3 & 5 & 1 \end{bmatrix}$$

Step 1: Select the MatrixWriter application, and key in the first element (cell 1-1):

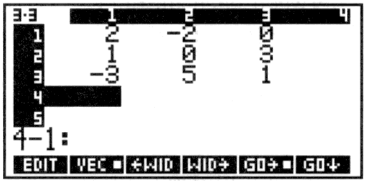
→ MATRIX 2

Step 2: Enter the first element and the rest of the first row.

ENTER 2 +/– ENTER 0
ENTER

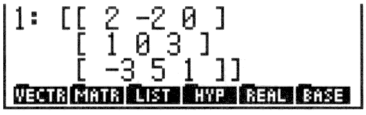
Step 3: Use ▼ to end the first row, and enter rest of the matrix.

▼
1 ENTER 0 ENTER 3 ENTER
3 +/– ENTER 5 ENTER 1
ENTER



Step 4: Enter the matrix onto the stack.

ENTER



While you're entering a number, the cell coordinate is replaced by the command line. When you press ENTER to store the value in the cell, the cell cursor normally advances to the next cell.

When you press ▼ at the end of the first row, it sets the number of columns in the matrix and moves the cursor to the beginning of the next row. You don't have to press ▼ again—the cell cursor automatically wraps to each new row.

If the displayed number is wider than the cell width, an ellipsis indicates "more to the right" (as in 1.2…). The default cell width is four characters.

Note the two uses of ENTER: While you're using the command line for data entry, ENTER enters data into a cell. When a cell coordinate is displayed, ENTER enters the entire matrix onto the stack.

8

To enter a vector using the MatrixWriter application:

1. Press → MATRIX to display the MatrixWriter screen and menu.
2. Key in the numbers in the vector, and press ENTER after each.
3. After you've entered all the numbers in the vector, press ENTER to put the vector on the stack.

Vectors normally use only one row of data, so you don't need to press ▼.

To enter numbers into more than one cell at a time:

1. Enter the series of numbers into the command line, pressing SPC between each number.
2. Press ENTER to enter the numbers.

To compute elements in the command line as you enter them:

1. Enter arguments and press command keys as required to execute the computation (press SPC to separate arguments).
2. Press ENTER to finish the computation and enter the result into the current cell.

Example: Enter 2.2^4 in a cell.

2.2 SPC 4 y^x ENTER

Editing Arrays

MatrixWriter provides functions that make it simple to edit the arrays you've entered.

To edit an array you're viewing with the MatrixWriter application:

1. Press ◀ ▶ ▲ ▼ to move the cell cursor. (Use with → to move the cursor to the far end.)
2. Use the MatrixWriter operations listed below to add or edit cells.
3. Press ENTER to save the changes (or press CANCEL to discard them) and return to the stack.

MatrixWriter Operations

To edit the contents of a cell:

1. Move the cursor to the cell to be edited.
2. Press EDIT.
3. Optional: Press ← EDIT to use the regular EDIT menu (see page 2-12). Press → MATRIX to restore the MatrixWriter menu.
4. Make the desired changes and press ENTER to save them (or CANCEL to discard them).

To make the displayed cells narrower or wider:

- Press +WID to make the cells narrower and display one additional column.
- Press WID+ to make the cells wider and display one less column.

To control how the cursor advances after an entry:

- To make the cursor move to the next *column* after entry, press GO→ so that the ■ is visible.
- To make the cursor move to the next *row* after entry, press GO↓ so that the ■ is visible.
- To keep the cursor from advancing at all after entry, press GO→ and GO↓ until neither shows a ■.

To insert a column:

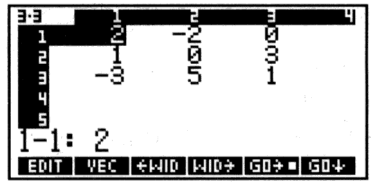
1. Move the cursor to the column where you want the new column inserted.
2. Press +COL. A column of zeros is inserted.

Example: Change the matrix in the first example of this chapter

$$\text{from } \begin{bmatrix} 2 & -2 & 0 \\ 1 & 0 & 3 \\ -3 & 5 & 1 \end{bmatrix} \quad \text{to} \quad \begin{bmatrix} 2 & -2 & 4 & 0 \\ 1 & 0 & 1 & 3.1 \\ -3 & 5 & 3 & 1 \end{bmatrix}$$

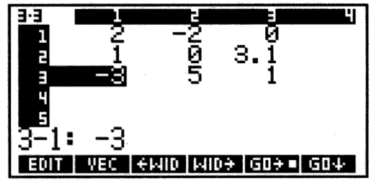
Step 1: If the matrix is on the stack, bring it into level 1—otherwise, enter the matrix into level 1. Then view the matrix in the MatrixWriter environment. (This example assumes GO→■ is active.)

▼ (or enter the matrix)



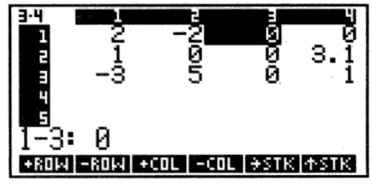
Step 2: Edit element 2-3:

▼ ▶ ▶
EDIT ▶ .1 ENTER



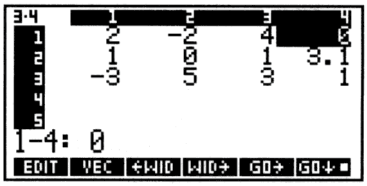
Step 3: Insert a new column in front of column 3, and move the cell cursor to the top of the new column.

▶ ▶ NXT +COL ▲ ▲



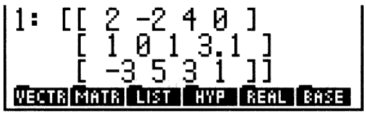
Step 4: Set top-to-bottom entry mode. Fill in the new column.

NXT GO↓
4 SPC 1 SPC 3 ENTER



Step 5: Restore left-to-right entry mode, then enter the edited matrix.

GO→ ENTER



To delete a column:

1. Move the cursor to the column you want to delete.
2. Press -COL.

To add a column to the right of the last column:

1. Move the cursor to the right of the last column.
2. Enter a value. The rest of the column fills with zeros.

To insert a row:

1. Move the cursor to the row where you want the new row inserted.
2. Press +ROW. A row of zeros is inserted.

To delete a row:

1. Move the cursor to the row you want to delete.
2. Press -ROW.

To add a row below the bottom row:

1. Move the cursor below the bottom row.
2. Enter a value. The rest of the row fills with zeros.

Summary of MatrixWriter Operations

Key	Description
EDIT	Places contents of the current cell in the data entry line for editing. (Press ◀ EDIT to get the EDIT menu.) Press ENTER to save the changes, or CANCEL to discard them.
VEC	For one-row arrays, toggles between vector entry and matrix entry. If this key is "on" (VEC■), one-row arrays are entered into the command line as vectors (example: [1 2 3]); if it's "off" (VEC), one-row arrays are entered as matrices (example: [[1 2 3]]).
←WID	Narrows all cells so that one more column appears.
WID→	Widens all cells so that one fewer column appears.
GO→	Sets left-to-right entry mode. The cell cursor moves to the next *column* after data entry.
GO↓	Sets top-to-bottom entry mode. The cell cursor moves to the next *row* after data entry.
+ROW	Inserts a row of zeros at the current cursor position.
-ROW	Deletes the current row.
+COL	Inserts a column of zeros at the current cursor position.
-COL	Deletes the current column.
→STK	Copies the current cell to level 1 of the stack.
↑STK	Activates the Interactive Stack, which can copy (echo) stack objects to the command line.
▶ MATRIX	Restores the MatrixWriter menu if another menu is displayed.

9

Graphic Objects

Graphic objects (or *grobs*) encode the data for HP 48 "pictures," including plots of mathematical data, custom graphical images, and representations of the stack display itself. The HP 48 provides a PICTURE environment for viewing and editing graphic objects.

Like all objects on the HP 48, graphic objects can be put on the stack and stored in variables. On the stack, a grob is displayed as

Graphic $n \times m$

where n and m are the width and height in *pixels*. (A pixel is one picture element, or "dot," in the display.)

The HP 48 uses two kinds of graphic objects:

- **Plots.** These are graphical representations of functions, equations, and sets of data that are automatically generated by the PLOT application. The HP 48 can zoom plots in any of 15 ways, and can numerically analyze function plots.
- **Pictures.** These are free-form grobs created "pixel-by-pixel"—either automatically using "snapshot" commands, or manually using the Picture Editor.

The PICTURE Environment

To enter the PICTURE environment directly:

- From the stack, press ◀ PICTURE.

9

The Default PICTURE Display

To exit the PICTURE environment:

- Press CANCEL. Note that this does *not* discard the displayed graphics object, but just returns the display to what you were viewing before entering the PICTURE environment.

Using the Picture Editor

The Picture Editor allows you to create and modify graphics using defined elements (lines, boxes, and circles) or pixel by pixel at a time. It also enables you to copy or delete some or all of a picture, and to superimpose one picture onto another.

To begin the Picture Editor:

- From the stack, press ◀ PICTURE EDIT.

To return to the main PICTURE environment from the Editor:

- Press PICT in the third page of the Picture Editor, or press ▶ MENU.

To return to the stack from the Picture Editor:

- Press CANCEL.

Turning Pixels On and Off

Two operations, DOT+ and DOT- , allow you to selectively turn pixels on and off. If one of these keys is active, it displays a "bullet" (■) in its label.

- If there is a ■ in the DOT+ label, pixels under the cursor are turned on.
- If there is a ■ in the DOT- label, pixels under the cursor are turned off.

Adding Elements Using the Graphics Environment

The Picture Editor allows you to add three geometric elements—line segments, boxes, and circles—to your grob:

Each of these elements requires *two* cursor positions. This means that you must tell the Editor to remember the first cursor position while you move to the second. You do this by *marking* the first position.

To mark the current cursor position:

- Press MARK from the second page of the Picture Editor menu or ⊗. Press MARK or ⊗ again to remove the mark. Also, any operation that requires a mark will *create* a mark when you first press its key, and then execute the operation when you press its key a second time.

To draw a line segment on the current grob:

1. From within the Picture Editor, move the cursor to where you want one endpoint of the segment.
2. Press ⊗ (or MARK or LINE).
3. Move the cursor to the other endpoint and press LINE .

To draw a box on the current grob:

1. From within the Picture Editor, move the cursor to where you want one corner of the box.
2. Press ⊗ (or MARK or BOX).
3. Move the cursor to the *opposite* corner of the desired box and press BOX .

To draw a circle on the current grob:

1. From within the Picture Editor, move the cursor to where you want the center of the circle.
2. Press ⊠ (or MARK or CIRCL).
3. Move the cursor to any point on the perimeter of the desired circle and press CIRCL.

9

To toggle a line segment on the current grob:

1. From within the Picture Editor, move the cursor to one endpoint of the segment.
2. Press ⊠ (or MARK or TLINE).
3. Move the cursor to the other endpoint and press TLINE. All pixels between the mark and the cursor are toggled—those that were on are now off, and vice versa.

Editing and Erasing a Picture

To erase an entire picture:

- While viewing the picture press NXT ERASE (or ◀ CLEAR as a shortcut).

To erase a rectangular area of the picture:

1. Move the cursor to one corner of the rectangular area to be erased, and press ⊠ (or DEL) to mark it.
2. Move the cursor to the opposite corner of the rectangular area.
3. Press NXT DEL (or DEL as a shortcut).

To copy a rectangular area of the picture to the stack:

1. Move the cursor to one corner of the rectangular area you want to copy, and press ⊠ (or SUB) to mark it.
2. Move the cursor to the opposite corner of the region.
3. Press NXT NXT SUB . The area is copied to level 1, and the picture remains in the display.

To superimpose a second grob on top of the current one:

1. Put the second grob on stack level 1.

2. Open the Picture Editor (◀ PICTURE EDIT) and move the cursor to the upper left corner of the rectangular region where you want to superimpose the grob.
3. Press NXT NXT REPL .

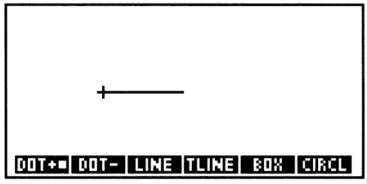
To copy the entire picture (the contents of *PICT*) to the stack:

- While viewing the picture, press STO (or PICT→). A copy of *PICT* is entered on level 1, and the picture remains in the display.

Example: Create and edit a small picture. This will illustrate some of the Picture Editor operations discussed above.

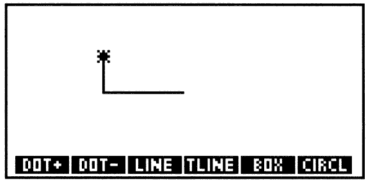
Step 1: Begin the Picture Editor and erase *PICT*. Then use DOT+ to draw a horizontal line from the center halfway to the left edge.

◀ PICTURE
◀ CLEAR EDIT
DOT+
◀ (hold down)



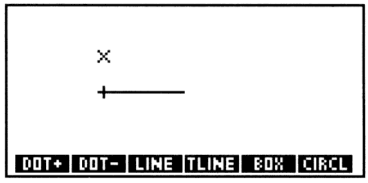
Step 2: Turn off line-drawing, then use LINE to draw a vertical line from the current cursor position halfway to the top edge.

DOT+■
⊠ (to mark)
▲ (hold down)
LINE



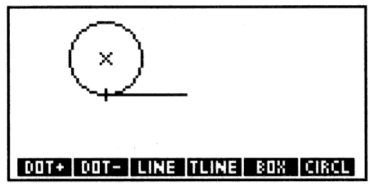
Step 3: Move the cursor to the lower end of the line, and toggle the line off.

▼ (hold down)
TLINE



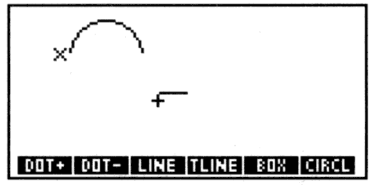
Step 4: Draw a circle using the existing mark and the current cursor position.

CIRCL



Step 5: Delete the lower semicircle.

◀ (move to left of circle)
▲ (move halfway up the circle)
× (to mark)
▼ (move below the circle)
▶ (move right of the circle)
DEL



Saving and Viewing Graphic Objects

The PICTURE environment displays and uses one grob at a time. The *current* grob is always stored in the reserved variable *PICT*. Think of *PICT* as the HP 48's built-in "chalkboard," where functions are plotted and pictures are drawn. Grobs (plots and pictures) can be stored using any valid name of your choice, but to be viewed the evaluated name must be copied into *PICT*.

To save the currently displayed grob:

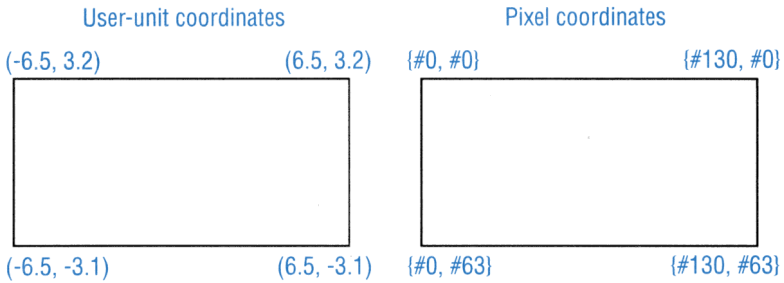
1. While viewing the grob in the PICTURE environment, press STO. This copies the grob to stack level 1.
2. Press CANCEL one or more times to exit the PICTURE environment and return to the stack.
3. Using ' delimiters, type in a name.
4. Press STO. The grob is stored in the current directory.

To view a grob that is not currently displayed:

1. Save the currently displayed grob (see above) if you want to keep it.
2. Recall the desired grob (do not use ' delimiters) to level 1.
3. Type `PICT` into the command line (do not use ' delimiters).
4. Press [STO].
5. Press [←][PICTURE].

Graphic Object Coordinates

Pixels in a grob can be specified with either *pixel coordinates* or *user-unit* coordinates.



User-Unit Coordinates vs. Pixel Coordinates

Pixel coordinates (the default) in a standard size *PICT* grob are numbered from `{ #0 #0 }` in the upper-left corner to `{ #130 #63 }` in the lower-right corner. Notice that pixel coordinates are given as a list containing two binary integers—the first designating the column and the second designating the row. Pixel coordinates are the most convenient to use when manipulating pictures.

User-unit coordinates depend on the current settings in *PPAR* (see chapter 22) but their default values range from `(-6.5, 3.2)` in the upper-left corner to `(6.5, -3.1)` in the lower-right corner. User-unit coordinates are given as a complex number (ordered pair), with the real part representing the horizontal coordinate and the imaginary part representing the vertical coordinate. User-unit coordinates are most convenient for use when plotting.

Graphic Object Commands

The PRG command menus contain two submenus, GROB and PICT, that contain programmable commands useful for manipulating pictures and picture elements.

9

Graphics Object Commands

Key	Programmable Command	Description
(PRG) PICT:		
PICT	PICT	Puts the name PICT on the stack so you can access the *PICT* graphics object as if it were stored in a variable.
PDIM	PDIM	Redimensions *PICT* to the dimensions given in levels 2 and 1. Dimensions are either the width and height (given in pixels) or the minimum and maximum coordinates (given in user-units).
LINE	LINE	Draws a line in *PICT* between the coordinates in levels 2 and 1.
TLINE	TLINE	Same as LINE except that pixels along the line are toggled on or off, rather than turned on.
BOX	BOX	Draws a box in *PICT* using two coordinate arguments as opposite corners.
ARC	ARC	Draws an arc in *PICT* centered at a coordinate (in level 4) with a given radius (in level 3) counterclockwise from θ_1 (in level 2) to θ_2 (in level 1). (The coordinate and radius must both use user-units or pixels.)
PIXON	PIXON	Turns on the pixel in *PICT* specified in level 1.
PIXOF	PIXOFF	Turns off the pixel in *PICT* specified in level 1.

Graphics Object Commands (continued)

Key	Programmable Command	Description
PIX?	PIX?	Returns 1 if the pixel specified by the coordinate in level 1 is on, or 0 if the pixel is off.
PVIEW	PVIEW	Displays *PICT* with the specified coordinate at the upper left corner of the graphics display.
PX→C	PX→C	Converts a pixel coordinate { #n_x #n_y } to a user-unit coordinate (x, y).
C→PX	C→PX	Converts a user-unit coordinate (x, y) to a pixel coordinate { #n_x #n_y }.
PRG GROB :		
→GRO	→GROB	(To graphics object.) Converts an object (level 2) into a graphics object using real number n (0 to 3 from level 1) to specify the character size. The resultant graphics object is a string of small (n=1), medium (n=2), or large (n=3) characters. For n=0, the character size is the same as for n=3, except that for algebraic and unit objects, the resulting graphics object is the EquationWriter picture.
BLAN	BLANK	Creates a blank graphics object on the stack of size #n_x (in level 2) by #n_y (in level 1).
GOR	GOR	(Graphics-object OR.) Superimposes the level 1 graphics object onto the level 3 graphics object. The upper left corner of the level 1 graphics object is positioned at coordinates specified in level 2.
GXOR	GXOR	(Graphics-object XOR.) Same as GOR except that the level 1 graphics object appears normal on a light background and inverse on a dark background.

Graphics Object Commands (continued)

Key	Programmable Command	Description
SUB	SUB	(Subset.) Extracts and returns to the stack a portion of a graphics object (level 3) defined by two coordinates (levels 2 and 1) that mark the diagonal corners of the rectangle to be extracted.
REPL	REPL	(Replace.) Same as GOR except that the level 1 graphics object *overwrites* the level 3 graphics object where the level 1 graphics object is located.
→LCD	→LCD	(Stack to LCD.) Displays the graphics object from level 1 in the *stack* display, with its upper left pixel in the upper left corner of the display. It overwrites all of the display except the menu labels.
LCD→	LCD→	(LCD to stack.) Returns to level 1 a graphics object representing the current stack display.
SIZE	SIZE	For the graphics object in level 1, returns the width (level 2) and height (level 1) in pixels.
ANIM	ANIMATE	Takes from levels 2 through $n + 1$ a sequence of grobs and from level 1 either: a) the number of grobs (n) or b) a list containing four items: the number of grobs (n), a list containing the pixel coordinates ({ #n_x #n_y }) of the upper left corner of region where the animation is to be located, the time delay (in seconds) between each "frame" of the animation, and the number of times to repeat the animation sequence (0 = repeat indefinitely until interrupted by a keystroke). It then displays each grob in sequence in the specified location for the specified number of seconds.

10

Unit Objects

The Units application contains a catalog of 147 units that you can combine with real numbers to create *unit objects*. The Units application lets you do the following:

- Convert units. For example, you can convert the unit object `10_ft` to `120_in` or `3.048_m`.
- Factor units. For example, you can factor `20_W` with respect to `1_N` and return `20_N*m/s`.
- Calculate with units. For example, you can add `10_ft/s` to `10_mph` and return `24.67_ft/s`.

Overview of the Units Application

The Units application consists of two menus:

- The UNITS Catalog menu (→ UNITS), which contains the HP 48 units organized by subject. This menu enables you to create unit objects, and to convert between related units in the catalog.
- The UNITS Command menu (← UNITS), which contains commands for converting units and for managing unit objects.

Units and Unit Objects

The Units application is based on the International System of Units (SI). The International System specifies seven *base* units: `m` (meter), `kg` (kilogram), `s` (second), `A` (ampere), `K` (kelvins), `cd` (candela), and `mol` (mole). The HP 48 makes use of two additional base units: `r` (radian) and `sr` (steradian). The UNITS Catalog menu contains these nine base units, and 141 *compound* units derived from the base units.

For example, `in` (inch) is defined as .0254 `m`, and `Fdy` (Faraday) is defined as 96487 `A*s`. (See appendix E for a complete listing of the built-in units and their SI values.)

A *unit object* has two parts: a *number* (a real number) and a *unit expression* (a single unit or multiplicative combination of units). The two parts are linked by the underscore character _. For example, `2_in` (2 inches) and `8.303_gal/h` (8.303 US gallons per hour) are unit objects. Like other object types, a unit object can be placed on the stack, stored in a variable, and used in algebraic expressions and programs.

When you perform a *unit conversion*, the HP 48 replaces the old unit expression with a new unit expression you specify, and automatically multiplies the number by the appropriate conversion factor.

Operators in unit objects follow this precedence order:

1. ( ) (highest precedence)
2. ^
3. * and /

For example, `7_m/s^2` is 7 meters per second squared, and `7_(m/s)^2` is 7 square meters per second squared.

The UNITS Catalog Menu

The UNITS Catalog menu (→ UNITS) displays a three-page menu of "subject" keys, each of which, when pressed, displays a submenu of related units. For example, → UNITS NXT PRESS displays a two-page menu of units for pressure.

The individual keys in each submenu behave differently than standard menu keys, as described throughout this chapter. In Immediate-entry mode, you can use the shift keys with the menu keys as follows:

- An unshifted menu key *creates* a unit object by combining the real number on level 1 with the unit expression that corresponds to that key. (In Algebraic- or Program-entry, the unshifted keys act as typing aids, echoing the corresponding name into the command line.)

- A left-shifted menu key *converts* the unit object in the command line or level 1 to the corresponding unit.
- A right-shifted menu key *divides by* the corresponding unit. This helps you create unit expressions with units in the denominator.

Creating a Unit Object

The UNITS Catalog menu provides a simple method for creating a unit object.

To create a unit object on the stack:

1. Key in the number part of the unit object.
2. Press → UNITS and select the appropriate subject menu.
3. Press the menu key for the unit you want. (If you want the *inverse* of the unit, press → and the menu key.)
4. For compound units, repeat steps 2 and 3 for each individual unit in the unit expression.

When you press a menu key in the UNITS Catalog menu, the HP 48 first enters a corresponding unit object on the stack with the number value 1. Then, for an unshifted key, it executes * (multiply), or, for a right-shifted key, it executes / (divide).

To create a unit object in the command line:

1. Key in the number.
2. Key in the _ character (press → _). This activates Algebraic-entry mode.
3. Key in the unit expression as you would an algebraic expression:
   - To key in a unit name, either press the corresponding menu key or spell the unit name.
   - To create compound units, press ×, ÷, y^x, and ← ( ) as required.

Note that unit names are case-sensitive. For example, Hz (hertz) must be typed with uppercase H and lowercase z. (For legibility, all letters in menu keys are uppercase. Don't confuse the menu-key representation of a unit with its proper name.)

By spelling unit names, you can create a unit object without switching between submenus in the UNITS Catalog menu. However, the menu keys eliminate errors resulting from incorrect spelling and incorrect use of uppercase or lowercase.

Example: Create the unit object `8_Btu/(ft^2*h*°F)` in the command line.

Step 1: Key in the number and the _ character. Then key in the unit expression using alpha characters and enter the unit object.

8 ▶ _
α α B ◀ T ◀ U ÷ ◀ ( )
◀ F ◀ T α y^x 2
× α ◀ H × α ▶ 6 α F
ENTER

To create a unit object using the EquationWriter application:

1. Press ◀ EQUATION.
2. Enter the number, press ▶ _, and enter the unit expression using standard EquationWriter notation.
3. Press ENTER.

The EquationWriter application lets you build algebraics that contain unit objects, showing you the unit expression as you would write it on paper. Inverse units are displayed in fractional form, and exponents are displayed as superscripts.

Unit Prefixes

You can also insert a unit prefix in front of a unit to indicate a power of ten. The following table lists the available prefixes. (To key in μ, press α ▶ N.)

Unit Prefixes

Prefix	Name	Exponent
Y	yotta	+24
Z	zetta	+21
E	exa	+18
P	peta	+15
T	tera	+12
G	giga	+9
M	mega	+6
k *or* K	kilo	+3
h *or* H	hecto	+2
D	deka	+1

Prefix	Name	Exponent
d	deci	−1
c	cent	−2
m	milli	−3
μ	micro	−6
n	nano	−9
p	pico	−12
f	femto	−15
a	atto	−18
z	zepto	−21
y	yocto	−24

Most prefixes used by the HP 48 correspond with standard SI notation, with one exception: "deka" is "D" in HP 48 notation and "da" in SI notation.

Note

You cannot use a prefix with a built-in unit if the resulting unit matches another built-in unit. For example, you cannot use `min` to indicate milli-inches, because `min` is a built-in unit indicating "minutes." Other combinations that match built-in units are `Pa`, `da`, `cd`, `ph`, `flam`, `nmi`, `mph`, `kph`, `ct`, `pt`, `ft`, `au`, `cu`, `yd`, `yr`.

Converting Units

The HP 48 provides several ways to convert unit objects to different units:

- The UNITS Catalog menu—converts to built-in units only.
- The CONVERT command—converts to any units.
- The UBASE (base units) command—converts to SI base units only.

If you're working with temperature units, see "Working with Temperature Units" on page 10-10.

Using the UNITS Catalog Menu

The UNITS Catalog menu lets you convert the unit object in stack level 1 to any dimensionally consistent unit in the menu.

To convert units to a built-in unit:

1. Enter the unit object with the original units.
2. Press ▶ UNITS and select the subject menu that contains the desired unit.
3. Press ◀ and the menu key for the desired unit.

Using CONVERT

You can use the CONVERT command to convert unit objects between any dimensionally consistent unit expressions.

To convert to any units:

1. Enter the unit object with the original units.
2. Enter any number (such as 1) and attach the units you want to convert to.
3. Press ◀ UNITS CONV.

CONVERT converts the level 2 unit object using the units from the level 1 object. It ignores the number part of the level 1 unit object.

Using UBASE (for SI Base Units)

The UBASE command converts a compound unit into its equivalent SI base units.

To convert units to SI base units:

1. Enter the unit object with the original units.
2. Press ◀ UNITS UBASE.

Converting Angular Units

Planar and solid angles are associated with actual units. However, although this distinguishes them from scalars (*dimensionless* numbers), the HP 48 allows you to convert between units of angle and scalars. The conversion will interpret the scalar according to the current angle mode setting (degrees, radians, or grads).

Unit	Symbol	Definition	Value
Arcmin	arcmin	$^1/_{21600}$ unit circle	$2.90888208666 \times 10^{-4}$ r
Arcsec	arcs	$^1/_{1296000}$ unit circle	$4.8481368111 \times 10^{-6}$ r
Degree	°	$^1/_{360}$ unit circle	$1.74532925199 \times 10^{-2}$ r
Grad	grad	$^1/_{400}$ unit circle	$1.57079632679 \times 10^{-2}$ r
Radian	r	$^1/_{2\pi}$ unit circle	1 r
Steradian	sr	$^1/_{4\pi}$ unit sphere	1 sr

Calculating with Units

The HP 48 lets you execute many arithmetic operations with unit objects, just as you would with real numbers:

- Addition and subtraction (dimensionally consistent units only)
- Multiplication and division
- Inversion
- Raising to a power
- Percentage calculations (dimensionally consistent units only)
- Comparisons of values (dimensionally consistent units only)
- Trigonometric operations (planar angular units only)

Several additional math operations are available, but work only on the number part of the unit object.

To calculate with unit objects:

1. Enter the unit objects.
2. Execute the commands.

Units are automatically converted and combined during the calculation, though certain operations require dimensionally consistent

units. Such operations convert results with units to the units from the object in level 1.

Temperature units require special note: see "Working with Temperature Units" on page 10-10.

The trigonometric operations SIN, COS, and TAN, operate only on unit objects with *planar angular* units: radians (r), degrees (°), grads (grad), arc-minutes (arcmin), or arc-seconds (arcs). The result is a dimensionless real number.

Example: **Subtraction.** Subtract 39_in from 4_ft.

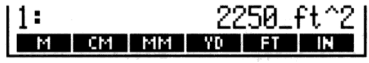
10

→ UNITS LENG
4 FT
39 IN
−

Example: **Unit Multiplication and Division.** Multiply 50_ft by 45_ft, then divide by 3.2_d (days).

Step 1: First, multiply the two unit objects.

→ UNITS LENG
50 FT
45 FT
×



Step 2: Enter the third unit object and divide.

→ UNITS TIME
3.2 D
÷

Example: **Powers.** Raise 2_ft/s to the sixth power. Find the square root of the result. Then find the cube root of that result.

Step 1: Enter the unit object and raise the unit object to the sixth power.

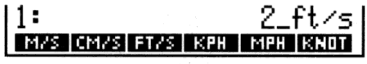
2 → UNITS SPEED FT/S
6 y^x

Step 2: Now find the square root of the result.

√x

Step 3: Find the cube root of the result.

3 ▶ ∛y



Example: **Percentage.** `4.2_cm^3` is what percent of `1_in^3`?

▶ UNITS `VOL` `1 IN^3` `4.2 CM^3`
MTH `REAL` `%T`

10

Factoring Unit Expressions

The UFACT command factors one unit within a unit expression, returning a unit object whose unit expression consists of the factored unit and the remaining SI base units.

To factor units within a unit expression:

1. Enter the unit object with the original units.
2. Enter any number (such as 1) and attach the units you want to factor out.
3. Press ▶ UNITS `UFACT`.

UFACT factors the units of the level 1 object from the level 2 unit object.

Using Unit Objects in Algebraics

Unit objects are allowed in algebraics—you enter them just as you enter them in the command line. In addition, the command line permits symbolic numbers instead of real numbers, converting `'Y_ft'`, for example, to `Y*1_ft` when entered on the stack.

+ and − are allowed in the number. However, the _ character takes precedence over + and −. Thus `'(4+5)_ft' EVAL` returns `9_ft`, but `'4+5_ft' EVAL` returns `+ Error: Inconsistent Units`.

Working with Temperature Units

The HP 48 lets you work with temperature units the same ways you work with other units—*except* you must recognize and anticipate the difference between temperature *level* and temperature *difference*. For example, a temperature *level* of 0 °C means "freezing," but a temperature *difference* of 0 °C means "no change."

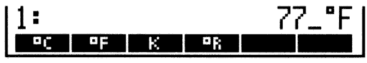
When °C or °F represents a temperature *level*, then the temperature is a unit with an additive constant: 0 °C = 273.15 K, and 0 °F = 459.67 °R. But when °C or °F represents a temperature *difference*, then the temperature is a unit with no additive constant: 1 °C = 1 K, and 1 °F = 1 °R.

Converting Temperature Units

Conversions between the four temperature scales (K, °C, °F, and °R) involve additive constants as well as multiplicative factors. The additive constants are *included* in a conversion when the temperature units reflect actual temperature *levels*, and are *ignored* when the temperature units reflect temperature *differences*:

- **Pure temperature units (levels).** If both unit expressions consist of a single, unprefixed temperature unit with no exponent, the UNITS Catalog menu or CONVERT performs an *absolute* temperature scale conversion, which includes the additive constants.
- **Combined temperature units (differences).** If either unit expression includes a prefix, an exponent, or any unit other than a temperature unit, CONVERT performs a *relative* temperature unit conversion, which ignores the additive constants.

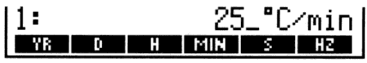
Example: Convert 25_°C to °F.



Example: Convert 25_°C/min to °F/min.

Step 1: First, create the unit object 25_°C/min.

→ UNITS NXT TEMP
25 °C
→ UNITS TIME → MIN



Step 2: Enter a unit object containing the new units.

→ MENU 1 °F
→ MENU → MIN

Step 3: Perform the conversion.

← UNITS CONV

Calculating with Temperature Units

Temperature units are automatically converted and combined during calculations.

- **Pure temperature units (levels or differences).** The relational operators ($<$, $>$, $\leq$, $\geq$, $==$, $\neq$) interpret pure temperatures as temperature *levels relative to absolute zero* for all temperature scales. Before making the calculation, the HP 48 converts any Celsius or Fahrenheit temperature to absolute temperatures.

  The operators + and − and the functions %CH and %T require that pure temperature arguments either both be absolute temperatures (K or °R), both °C, or both °F. This ensures that such operations retain their correct algebraic properties.

  For all other functions, pure temperature units are interpreted as temperature *differences*—they're not converted before the calculation.

- **Combined temperature units (differences).** Temperature units with prefixes, exponents, or other units are interpreted as temperature *differences*—they're not converted before the calculation.

Example: Determine if 12 °C is greater than 52 °F. (The > operator interprets temperatures as levels.)

→ UNITS NXT TEMP
12 °C
52 °F
PRG TEST >

The result shows the test is true (12 °C is greater than 52 °F).

Example: Calculate the final temperature for an increase of 18 °F from the current temperature of 74 °F.

→ UNITS NXT TEMP
18 °F 74 °F +

Example: For a coefficient of linear expansion α of 20×10^{-6} 1/°C and a temperature change ΔT of 44 °C, calculate the fractional change of length given by $\alpha \Delta T$. (The × command interprets temperatures as differences.)

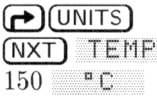
→ UNITS NXT TEMP
20 EEX +/- 6 → °C
44 °C ×

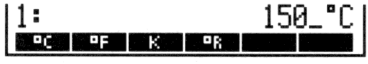
Whenever you must use *absolute* temperatures in a compound unit or expression, be sure to enter the temperatures using absolute scale. The HP 48 will not convert from °C or °F to the absolute scale correctly once the temperature has become a part of a combined expression.

Example: The ideal gas equation of state is $PV = nRT$, where P is the pressure exerted by the gas (in atmospheres), V is the volume of the gas (in liters), n is the amount of the gas (in moles), R is the ideal gas constant (0.082057 liter-atmospheres/kelvin-mole), and T is the temperature of the gas (in kelvins).

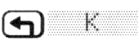
Assuming ideal gas behavior, calculate the pressure exerted by 0.305 mole of oxygen in a volume of 0.950 liter at 150 °C.

Step 1: First, enter the temperature.

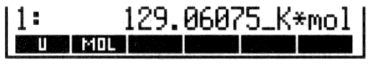




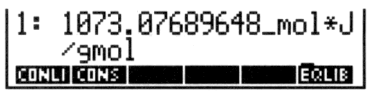
Step 2: Convert the units to kelvins. This conversion works correctly at this point because the temperature is still "pure," and not yet part of a compound unit.



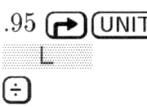
Step 3: Multiply T (already in level 1) by n (0.305 mole).



Step 4: Multiply nT by R, the ideal gas constant. Retrieve R from the Constants Library before multiplying.



Step 5: Divide by V (0.950 liter) to calculate P.



Step 6: Convert the pressure units to atmospheres.

Step 7: Convert the pressure (in atmospheres) to SI base units.

Creating User-Defined Units

If you use a unit that's not contained in the UNITS Catalog menu, you can create a *user-defined* unit that behaves just like a built-in unit. (The Equation Library contains four user-defined units—see page 25-15.)

To create a user-defined unit:

1. Enter a unit object using built-in or previously defined units that equals value of 1 new unit.
2. Store the unit object in a variable—the variable name is used as the name of the new unit.
3. Optional: Add a unit object having the user-defined unit to the CST menu—see below. The number part is ignored. (Custom menus are described on page 30-1.)

You *can't* use the unit key in the VAR menu like unit keys in the UNITS menus—because VAR menu keys store and recall objects. However, if you add the user-defined unit to the CST menu, you can use the CST menu key to enter and convert your user-defined units—just like UNITS menu keys.

Example: Use the built-in unit `d` (day) to create the user-defined unit `WEEK`. To do this, store the unit object `7_d` in variable WEEK. Enter a list containing an object with the new units: `{1_WEEK}`. Store the list in the custom menu and display the menu—press ↩ (MODES) `MENU MENU`.

Additional Commands for Unit Objects

Key	Programmable Command	Description
↪ (UNITS):		
`UVAL`	UVAL	Returns the number part of the level 1 unit object to level 1.
`→UNIT`	→UNIT	Combines a number from level 2 with a unit object from level 1, ignoring the number part of the level 1 object, to form a unit object in level 1.

11

Using Mathematical Functions

Built-In Functions and Commands

Built-in *functions* and built-in *commands* are subsets of HP 48 *operations*. An operation is any action the calculator can perform. (Every time you press a key, you execute an operation.) But not all operations are equivalent to one another. They are classified into the following categories:

- **Operation.** Any built-in action represented by a name or key.
- **Command.** Any programmable operation.
- **Function.** Any command that can be included in algebraic objects.
- **Analytic function.** Any function for which the HP 48 provides an inverse and derivative.

Analytic functions are a subset of functions—functions are a subset of commands—and commands are a subset of operations.

SIN, for example, is an analytic function—it has an inverse and derivative, can be included in an algebraic object, and is programmable. SWAP (the command to swap stack levels 1 and 2), however, is a command—it can be included in a program, but it can't go in an algebraic and has no derivative or inverse.

The operation index in appendix G tells you how each operation is classified. Also, throughout the manual, HP 48 activities are referred to as operations, commands, functions, or analytic functions where appropriate.

Built-in function and built-in command objects describe the HP 48 command set. You can think of them as built-in program objects. (Operations that aren't commands are not objects—you can't include them in programs.)

Expressing Functions: Algebraic Syntax

The difference between functions and other commands is that functions can be included in algebraic expressions. The syntax used by a function determines how it interprets its inputs (or *arguments*). Functions can be classified into three types based on their syntax:

- **Prefix functions.** Functions such as 'SIN(X)' and 'MAX(X,Y)' whose name (or *operator*) comes *before* its argument(s) (which are listed in parentheses and separated by commas).
- **Infix functions.** Functions such as + and ≥ that come *between* their two arguments.
- **Postfix functions.** Functions such as ! (factorial) that come *after* their argument.

11

Note

In the expression 'A(B*C)', A is treated as a prefix function and not as a multiplicative argument. The HP 48 interprets the expression as "apply function A to the product of B and C," instead of "multiply A by the product of B and C." If multiplication is intended, be sure to include the multiplication operator, * (or · in the EquationWriter).

Algebraic objects use algebraic syntax and thus use the normal rules of algebraic precedence to determine the order in which the functions are executed. Functions with higher precedence are performed first, and functions with the same precedence are performed from left to right. The HP 48's functions have the following algebraic precedence, from highest (1) to lowest (11):

1. Expressions within parentheses. Expressions within nested parentheses are evaluated from inner to outer.
2. Prefix functions (such as SIN, INV, or LOG).
3. Postfix functions (such as !).
4. Power (^).
5. Negation (−), multiplication (*), and division (/).
6. Addition (+) and subtraction (−).
7. Comparison operators (==, ≠, <, >, ≤, or ≥).
8. Logical operators AND and NOT.
9. Logical operators OR and XOR.
10. The left argument for | (where).
11. Equals (=).

Example:

`'A^3+B'`	Cubes A, then adds B to that quantity, since ^ has a higher precedence than +.
`'A^(3+B)'`	Raises A to the power $3+B$, since an expression within parentheses has a higher precedence than ^.

Expressing Functions: Stack Syntax

Although there are very few postfix algebraic functions, all functions on the HP 48 can be executed in postfix form using the stack. *Stack syntax* is postfix syntax, where the arguments are entered first, followed by the command or function name. Postfix syntax is often a more efficient means of using a series of functions than is standard algebriac syntax.

Thus the HP 48 allows you to use functions in two distinct ways: algebraic syntax within algebraic objects or in postfix syntax executed directly on the stack. For example, the sine function can be used either as `'SIN(X)'` or `'X' SIN` and addition can be either `'X+Y'` or `'X' 'Y' +`.

Remember: Unless you surround functions in single-quotes (the ' ' tick-mark delimiters), the HP 48 assumes that you are using postfix syntax whenever you invoke the name of a function and thus uses (or attempts to use) objects from the stack as the arguments for the function.

Expressions and Equations

An *expression* is an algebraic that does not contain an = function. An *equation* is an algebraic that does contain an = function. For example, 'SIN(X)-ATAN(2*X)+6*X' is an expression, and 'Y=ATAN(2*X)+6*X' is an equation.

When you use an equation as the argument of a function, the function is applied to both sides, and the result is also an equation. For example, 'X=Y' SIN returns 'SIN(X)=SIN(Y)'.

In the HP 48, = generally means equating two expressions. The DEFINE command (◀ DEF) interprets = differently—it *stores* the expression on the right side of the equal sign in the name on the left side (see 11-7 for more details).

Symbolic Constants

The HP 48 has five built-in constants that can be included in algebraic expressions either as symbolic constants or as 12-digit numerical approximations. The five constants are these:

- π (3.14159265359), the ratio of a circle's perimeter to its diameter.
- e (2.71828182846), the natural logarithm base.
- i (0,1), the square root of (-1).
- *MAXR* (9.99999999999E499), the largest positive real number representable by the HP 48.
- *MINR* (1.E-499), the smallest positive real number representable by the HP 48.

The five constants are available in both symbolic and numeric forms in the MTH CONSTANTS menu, accessed by pressing MTH NXT CONS. Three of the constants can also be entered directly from the main keyboard:

- Press ◀ π to obtain π.
- Press α ◀ E to obtain e.
- Press α ◀ I to obtain i.

The HP 48 also provides 40 physical constants (with their units) through its Constants Library. The CONST function allows these constants to be used in symbolic form. See page 25-14 for details.

Controlling How Symbolic Constants are Evaluated

System flags −2 (Symbolic Constants) and −3 (Symbolic Results) control whether evaluating symbolic constants return symbolic or numeric results. The default setting for both flags is clear.

To control the evaluation of symbolic constants:

- To leave a symbolic constant unchanged during evaluation, clear flags −3 and −2 (their default states).
- To replace a symbolic constant with its numeric value during evaluation, set flag −3.
- To replace a symbolic constant with its numeric value *except* when it's the argument of a function, clear flag −3 and set flag −2. Pressing EVAL causes the numeric value to be used, but the execution of another function (/, SIN, LOG, and so forth) does not.
- To force all constants to evaluate numerically regardless of the flag settings, press ◀ →NUM.

Using Built-In Math Functions

The next six chapters (chapters 12-17) are devoted to the built-in math functions available on the HP 48. The functions are grouped into chapters and divided into sections.

Most of the common arithmetic and scientific functions are located on the main keyboard. But there are many, many more that are located in the submenus accessed using the MTH key. The following table describes how to find each group of functions on the HP 48 and where in the manual it is discussed.

Finding Mathematical Functions on the HP 48

Topic or Group	Access	Reference
Arithmetic	Keyboard	Chapter 12
Exponential Functions	Keyboard	Chapter 12
Logarithmic Functions	Keyboard	Chapter 12
Trigonometric Functions	Keyboard	Chapter 12
Hyperbolic Functions	MTH HYP	Chapter 12
Probability Functions	MTH PROB	Chapter 12
Percentages	MTH REAL	Chapter 12
Built-In Constants	MTH NXT CONS	Chapter 12
	◀ EQLIB COLIB	Chapter 25
Real-Number Functions	MTH REAL	Chapter 12
Complex Functions	MTH NXT CMPL	Chapter 12
Vector Functions	MTH VECTR	Chapter 13
Fourier Transforms	MTH NXT FFT	Chapter 13
Matrix Functions	MTH MATR	Chapter 14
Linear Algebra	MTH MATR	Chapter 14
Number Base Conversions	MTH BASE	Chapter 15
Binary Arithmetic	MTH BASE	Chapter 15
Boolean Logic Operations	MTH BASE LOGIC	Chapter 15
Date and Time Arithmetic	◀ TIME	Chapter 16
Fraction Arithmetic	◀ SYMBOLIC	Chapter 16
Applying Functions to Lists	MTH LIST Keyboard	Chapter 17
Sequences and Series	MTH LIST	Chapter 17
Recursive List Procedures	PRG LIST	Chapter 17

User-Defined Functions

You can add your own *user-defined functions*. A user-defined function behaves like a built-in function is several ways:

- It takes its arguments from the stack or in algebraic syntax.
- It takes symbolic arguments.
- It can be differentiated.

Creating a User-Defined Function

The DEFINE command lets you create a user-defined function directly from an equation. The equation must have the form '*name*(*arguments*)=*expression*'.

To create a user-defined function:

1. Enter an equation that specifies the function name and its arguments on the left side, and the expression that defines the calculation on the right side. On the left side, use commas to separate multiple arguments.
2. Press ◀ DEF (the DEFINE command).

Example: Use DEFINE to create *CMB*, a user-defined function that calculates the number of combinations C of n different items taken 1, 2, 3, ... n at a time: $C = 2^n - 1$.

Step 1: Enter the equation for *CMB*.

' α α CMB ◀ ( ) ◀ N
α ▶
◀ = 2 y^x α ◀ N − 1
ENTER

Step 2: Execute DEFINE. Select the VAR menu and note that it now contains the user-defined function *CMB*.

◀ DEF
VAR

Executing a User-Defined Function

A user-defined function is executed just like a built-in function—it can take numeric or symbolic arguments, either from the stack or in algebraic syntax.

To execute a user-defined function:

- To use the stack, put the arguments on the stack in the same order they appear in the left side of the function definition (the last argument should be in stack level 1), then press the function key in the VAR menu (or type the function name and press ENTER).
- To use algebraic syntax, press ', press the function key in the VAR menu (or type the function name), press ◀()), enter the algebraic arguments in their proper order and separated by commas, then press ENTER (or press EVAL to evaluate the expression).

Example: Execute the user-defined function *CMB* from the earlier example to make the following calculations.

Step 1: Calculate the total number of ways to combine one or more of four items ($n = 4$).

4 CMB

Step 2: For the same value of n, calculate the combinations in algebraic syntax.

' CMB ◀() 4
EVAL

Step 3: Calculate *CMB(Z)* in algebraic syntax, where Z is a *formal variable*. (Purge Z to make sure is doesn't contain an object.)

' α Z ◀ PURG
' CMB ◀() α Z
EVAL

Nesting User-Defined Functions

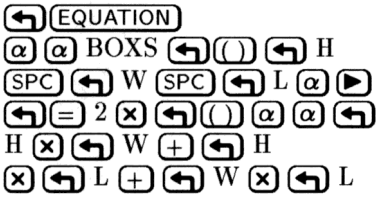
Just like built-in functions, user-defined functions can be included in the defining expression of a user-defined function.

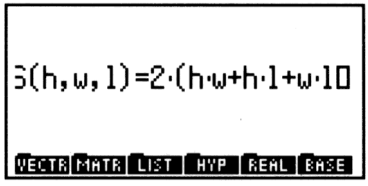
Example: Write a user-defined function to calculate the ratio of surface area to volume of a box. The formula for this calculation is

$$\frac{A}{V} = \frac{2(hw + hl + wl)}{hwl}$$

where h, w, and l are the height, width, and length of the box.

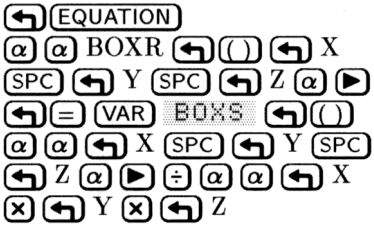
Step 1: First, create a user-defined function *BOXS* to calculate the surface area of the box. Use the EquationWriter application to key in the equation.

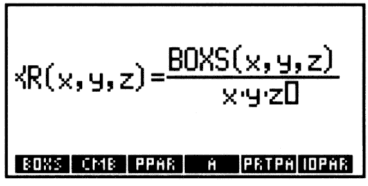




Step 2: Enter the equation and create the user-defined function.

Step 3: Now create a user-defined function *BOXR* to calculate the ratio of surface area to volume. Use the EquationWriter application to key in the equation.





Step 4: Enter the equation and create the user-defined function.

Step 5: Use *BOXR* to calculate the ratio of surface area to volume for a box 9 inches high, 18 inches wide, and 21 inches long. Enter the height, width, and length, then execute *BOXR*.

9 ENTER 18 ENTER 21
VAR BOXR

Note that *BOXS* was defined using h, w, and l as variables, and that *BOXS* takes x, y, and z as arguments in the definition for *BOXR*. It makes no difference if the variables in the two definitions match—each set of variables is independent of the other.

12

Functions of Real and Complex Numbers

Math Functions on the Main Keyboard

The following tables describe built-in commands that appear on keyboard.

Arithmetic and General Math Functions

Key	Programmable Command	Description
[1/x]	INV	*Prefix.* Inverse (reciprocal).
[√x]	√	*Prefix.* Square root.
[←] [x²]	SQ	*Prefix.* Square.
[+/−]	NEG	*Infix.* Change sign. Changes the sign of the number in the command line. When no command line is present, [+/−] executes a NEG command (changes the sign of the argument in level 1).
[+]	+	*Infix.* Level 2 + level 1.
[−]	−	*Infix.* Level 2 − level 1.
[×]	*	*Infix.* Level 2 × level 1.
[÷]	/	*Infix.* Level 2 ÷ level 1.
[y^x]	^	*Infix.* Level 2 raised to the level 1 power. The algebraic syntax for the ^ command is '*y*^*x*'.
[ˣ√y]	XROOT	*Prefix.* The *x*th (in level 1) root of a real value in level 2. The algebraic syntax for the XROOT command is 'XROOT(*x*,*y*)'.

Exponential and Logarithmic Functions

Key	Programmable Command	Description
↰ 10^x	ALOG	*Prefix.* Common (base 10) antilogarithm.
↱ LOG	LOG	*Prefix.* Base 10 logarithm.
↰ e^x	EXP	*Prefix.* Natural (base e) antilogarithm.
↱ LN	LN	*Prefix.* Natural (base e) logarithm.

12

Trigonometric Functions

For trigonometric functions, the angle arguments and results are interpreted as degrees, radians, or grads, depending on the current angle mode.

Trigonometric Functions

Key	Programmable Command	Description
SIN	SIN	*Prefix.* Sine.
↰ ASIN	ASIN	*Prefix.* Arc sine.
COS	COS	*Prefix.* Cosine.
↰ ACOS	ACOS	*Prefix.* Arc cosine.
TAN	TAN	*Prefix.* Tangent.
↰ ATAN	ATAN	*Prefix.* Arc tangent.

Hyperbolic Functions

Hyperbolic functions are found in the MTH HYP menu (MTH HYP).

Key	Programmable Command	Description
MTH HYP :		
SINH	SINH	*Prefix.* Hyperbolic sine: $(e^x - e^{-x})/2$.
ASINH	ASINH	*Prefix.* Inverse hyperbolic sine: $\sinh^{-1} x$.
COSH	COSH	*Prefix.* Hyperbolic cosine: $(e^x + e^{-x})/2$.
ACOSH	ACOSH	*Prefix.* Inverse hyperbolic cosine: $\cosh^{-1} x$.
TANH	TANH	*Prefix.* Hyperbolic tangent: $\sinh x/\cosh x$.
ATAN	ATANH	*Prefix.* Inverse hyperbolic tangent: $\sinh^{-1}(x/\sqrt{1-x^2})$.
EXPM	EXPM	*Prefix.* $e^x - 1$. Argument x is in level 1. (EXPM is more accurate than EXP when the argument to e^x is close to 0.)
LNP1	LNP1	*Prefix.* $\ln(x + 1)$. Argument x is in level 1. (LNP1, ln *plus* 1, is more accurate than LN when the argument to ln is close to 1.)

Probability and Test Statistics

Use the commands in the PROB (probability) menu (MTH NXT PROB) to calculate combinations, permutations, factorials, random numbers, and upper-tail probabilities of various test statistics.

Probability Commands

Key	Programmable Command	Description
MTH NXT PROB :		
COMB	COMB	*Prefix.* Number of combinations of n (in level 2) items taken m (in level 1) at a time.
PERM	PERM	*Prefix.* Number of permutations of n (in level 2) items taken m (in level 1) at a time.
!	!	*Postfix.* Factorial of a positive integer. For non-integers, ! returns $\Gamma(x + 1)$.
RAND	RAND	*Command.* Returns the next real number n ($0 \leq n < 1$) in a pseudo-random number sequence. Each random number becomes the seed for the next random number.
RDZ	RDZ	*Command.* Takes a real number from level 1 as a seed for the next random number (from RAND). 0 in level 1 creates a seed based on the clock time. A sequence of random numbers can be repeated by starting with the same nonzero seed.

Calculating Test Statistics

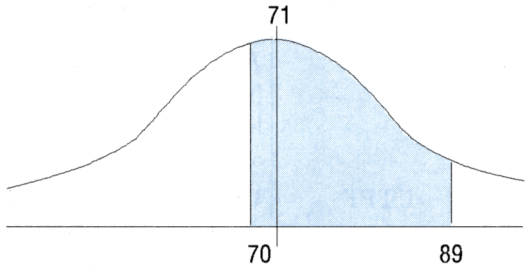
Test statistics are calculated using values you enter on the stack—they do *not* use the statistical data stored in ΣDAT in the STAT application.

Test Statistics Commands

Keys	Programmable Command	Description
(MTH) (NXT) PROB (NXT):		
UTPC	UTPC	**Upper-tail chi-square distribution.** *Prefix*. Takes the degrees of freedom from level 2 and a real number (x) from level 1, and returns the probability that a χ^2 random variable is greater than x.
UTPF	UTPF	**Upper-tail f distribution.** *Prefix*. Takes the numerator degrees of freedom from level 3, the denominator degrees of freedom from level 2, and a real number (x) from level 1, and returns the probability that a Snedecor's F random variable is greater than x.
UTPN	UTPN	**Upper-tail normal distribution.** *Prefix*. Takes the mean from level 3, the variance from level 2, and a real number (x) from level 1, and returns the probability that a normal random variable is greater than x for a normal distribution.
UTPT	UTPT	**Upper-tail t distribution.** *Prefix*. Takes the degrees of freedom from level 2 and a real number (x) from level 1, and returns the probability that the Student's t random variable is greater than x.
NDIST	NDIST	**Normal distribution.** *Prefix*. Takes the mean from level 3, the variance from level 2, and a real number (x) from level 1, and returns the density function at x for a normal distribution.

Note that, when used as an argument for these commands, the number of degrees of freedom must be between 0 and 499. Also, in the calculations, the degrees of freedom are rounded to the nearest integer.

Example: The scores on a final exam approximate a normal curve with a mean of 71 and standard deviation of 11. What percentage of the students scored between 70 and 89?



Step 1: First, calculate the probability that a student chosen at random obtained a score greater than 70. (Square the standard deviation to get the variance.)

(MTH) (NXT) PROB (NXT)
71 (ENTER)
11 (←) (x^2)
70 UTPN

```
1:              .536217586697
UTPC UTPF UTPN UTPT NDIST
```

Step 2: Now, do the same calculation for a score of 89, after recalling the last argument used.

(→) (ARG) (←)
89 UTPN

```
2:              .536217586697
1:          5.08817524756E-2
UTPC UTPF UTPN UTPT NDIST
```

Step 3: Subtract the two values. About 49% of the students scored between 70 and 89.

(−)

```
1:              .485335834221
UTPC UTPF UTPN UTPT NDIST
```

Real Number Functions

Some functions can only use real numbers as arguments. Among these are angle conversions, percentages, and several functions that round, truncate, or extract parts of real numbers.

Angle Conversion Functions

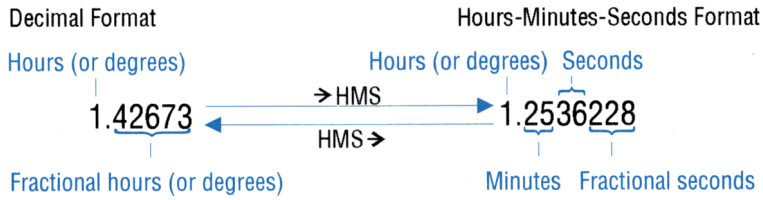
Two commands in the MTH REAL menu convert values between decimal degrees and radians. Two other commands in the TIME menu let you do degrees-minutes-seconds calculations using hours-minutes-seconds (HMS) format.

In Degrees mode, angle arguments and results use decimal degrees.

Angle Conversion Functions

Key	Programmable Command	Description
(MTH) REAL (NXT) (NXT):		
D→R	D→R	*Prefix*. Degrees to radians. Converts a number from a decimal degree value to its radian equivalent.
R→D	R→D	*Prefix*. Radians to degrees. Converts a number from a radian value to its decimal degree equivalent.
(↶) (TIME) (NXT):		
→HMS	→HMS	Decimal to HMS. Converts a number from decimal degrees to HMS format.
HMS→	HMS→	HMS to decimal. Converts a number from HMS format to decimal degrees.
HMS+	HMS+	Adds two angles in HMS format.
HMS−	HMS−	Subtracts two angles in HMS format.

The following illustrates the conversion to and from HMS format:



12

Example: Convert 1.79π radians to degrees.

Step 1: First, enter 1.79π.

1.79 ENTER
◀ π ×

Step 2: Use the R→D function. (The function acts independently of the current angle mode.)

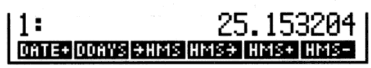
MTH REAL NXT NXT
R→D

Step 3: Use →NUM to obtain a numeric result.

◀ →NUM

Example: Convert 25.2589 degrees to degrees, minutes, and seconds.

25.2589 ◀ TIME NXT
→HMS



Percent Functions

Percent Functions

Key	Programmable Command	Description
(MTH) REAL :		
%	%	*Prefix.* A percent of B, or B percent of A (A is in level 2, B is in level 1): $(A \times B)/100$.
%CH	%CH	*Prefix.* The percent change from A to B, as a percentage of A (A is in level 2, B is in level 1): $((B - A)/A) \times 100$.
%T	%T	*Prefix.* The percent of total (the total, A, is in level 2 and the value, B, is in level 1): $(B/A) \times 100$.

Other Real-Number Functions

The functions in the following table are found in the MTH REAL menu ((MTH) REAL).

Command/Description	Example	
	Input	Output
ABS *Prefix.* Absolute value.	1: -12	1: 12
CEIL *Prefix.* Smallest integer greater than or equal to the argument.	1: -3.5	1: -3
	1: 3.5	1: 4
FLOOR *Prefix.* Greatest integer less than or equal to the argument.	1: 6.9	1: 6
	1: -6.9	1: -7
FP *Prefix.* Fractional part of the argument.	1: 5.234	1: .234
	1: -5.234	1: -.234

Command/Description	Example	
	Input	Output
IP *Prefix.* Integer part of the argument.	1: -5.234 1: 5.234	1: -5 1: 5
MANT *Prefix.* Mantissa of the argument.	1: 1.23E12	1: 1.23
MAX *Prefix.* Maximum; the greater of two arguments.	2: 5 1: -6	1: 5
MIN *Prefix.* Minimum; the lesser of two arguments.	2: 5 1: -6	1: -6
MOD *Prefix.* Modulo; remainder of $^{A}/_{B}$. *A* MOD *B* = *A* − *B* FLOOR ($^{A}/_{B}$).	2: 6 1: 4	1: 2
RND *Prefix.* Rounds number according to argument: *n* = 0 to 11 rounds to *n* FIX, *n* = −11 to −1 rounds to *n* significant digits, and *n* = 12 rounds to current display format.	2: 1.2345678 1: 5 2: 1.2345678 1: -5	1: 1.23457 1: 1.2346
SIGN *Prefix.* Returns +1 for positive arguments, −1 for negative arguments, and 0 for arguments of 0.	1: -2.7	1: -1
TRNC *Prefix.* Truncates number according to argument: *n* = 0 to 11 truncates to *n* FIX, *n* = −11 to −1 truncates to *n* significant digits, and *n* = 12 truncates to current display format.	2: 1.2345678 1: 5 2: 1.2345678 1: -5	1: 1.23456 1: 1.2345
XPON *Prefix.* Exponent of the argument.	1: 1.23E45	1: 45

Complex Numbers

Most functions that work with real numbers also work with complex numbers. So, the way you use complex numbers is similar to the way you use real numbers.

The examples in this section assume the calculator is set to Degrees mode. (Press ← MODES ANGL DEG to set Degrees mode.)

Displaying Complex Numbers

You can display complex numbers as either rectangular coordinates or polar coordinates—in Rectangular mode or in Polar mode.

To display rectangular coordinates for complex numbers:

- Press → POLAR until *no* coordinate annunciator is on.

  or

- Press → MODES ▼ ▼, then press +/- until Rectangular is displayed in the COORD SYSTEM: field. Confirm your selection by pressing OK .

To display polar coordinates for complex numbers:

- Press → POLAR until the R∠Z or R∠∠ coordinate annunciator is on.

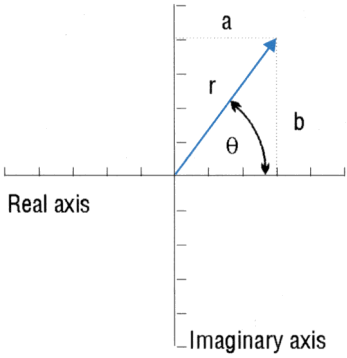
  or

- Press → MODES ▼ ▼, then press +/- until Polar is displayed in the COORD SYSTEM: field. Confirm your selection by pressing OK .

Even though only two coordinate modes are needed for complex numbers, three coordinate modes are available on the HP 48 (to provide for three-dimensional vectors)—Rectangular mode, Polar (cylindrical) mode, and Spherical mode.

Complex numbers are displayed inside parentheses. In rectangular form, the real and imaginary parts are separated by a comma. (If the Fraction Mark is set to comma, they're separated by a semicolon

instead.) In polar form, the magnitude and phase angle are separated by a comma and angle sign (∠). (The angle is based on the current angle mode: Degrees, Radians, or Grads.) Regardless of how complex numbers are displayed, the HP 48 stores them internally in *rectangular* form.



Display Modes

Rectangular	Polar
(a,b)	(r, ∠θ)

Entering Complex Numbers

You can enter complex numbers using either rectangular coordinates or polar coordinates.

To enter a complex number:

- To enter rectangular coordinates, press ◀ (), enter the coordinates separated by SPC or ◀ ., and press ENTER.
- To enter polar coordinates, press ◀ (), enter the coordinates separated by ▶ ∠, and press ENTER.

The internal rectangular representation of all complex numbers has the following effects on polar numbers:

- θ is normalized to the range $\pm 180°$ ($\pm\pi$ radians, ± 200 grads).
- If you key in a negative r, the value is made positive, and θ is increased by 180° and normalized.
- If you key in an r of 0, θ is also reduced to 0.

Real Calculations with Complex Results

The complex-number capabilities of the HP 48 can affect the results of real-number operations. Certain calculations that would result in an error on most calculators yield valid complex results on the HP 48. For example, the HP 48 returns a complex number for the square root of −4. Also, the arcsine of 5 yields a complex result.

You'll find that for most calculations, the HP 48 gives you the type of result (real or complex) you expect. However, if you find that you get complex results when you expect real results, check your program or keystrokes for these potential causes:

- The data you supplied to the calculator may be outside the range of the formula you are calculating.
- The formula (or its execution) may be incorrect.
- A rounding error at a critical point in the formula may have compromised the computation.
- A complex result may be unexpected, but correct, for your problem.

Other Complex-Number Commands

Most commands that operate on real numbers also operate on complex numbers (such as SIN, INV, ^, and LN). The following table describes additional commands that are especially useful for complex numbers.

The remaining commands are found in the MTH CMPL menu (press MTH NXT CMPL).

Command/Description	Example	
	Input	Output
ABS *Prefix.* Absolute value; $\sqrt{x^2+y^2}$.	1: (3,4)	1: 5
ARG *Prefix.* Polar angle of a complex number.	1: (1,1)	1: 45
CONJ *Prefix.* Complex conjugate of a complex number.	1: (2,3)	1: (2,-3)

Command/Description	Example	
	Input	Output
C→R *Command.* Complex to real; separates a complex number into two real numbers, the rectangular coordinates x and y.	1: (2,3)	2: 2 1: 3
IM *Prefix.* Imaginary (y) part of a complex number.	1: (4,-3)	1: -3
NEG *Infix.* Negative of its argument.	1: (2,-1)	1: (-2,1)
RE *Prefix.* Real (x) part of a complex number.	1: (4,-3)	1: 4
R→C *Command.* Real to complex; combines two real numbers into a complex number (x,y).	2: -7 1: -2	1: (-7,-2)
SIGN *Prefix.* Unit vector in the direction of the complex number argument; $(\frac{x}{\sqrt{x^2+y^2}}, \frac{y}{\sqrt{x^2+y^2}})$	1: (3,4)	1: (.6,.8)

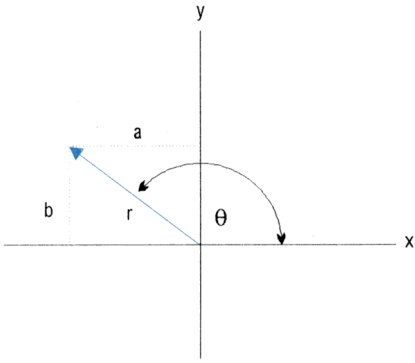
13

Vectors and Transforms

All vectors are array objects. The general case of n-dimensional vectors is covered in chapter 14, "Matrices and Linear Algebra"—this chapter deals primarily with 2D and 3D vectors.

Displaying 2D and 3D Vectors

You can display 2D vectors as either rectangular components ([*X Y*]) or polar components ([*R* ∡]) —in *Rectangular mode* or in *Polar mode*.

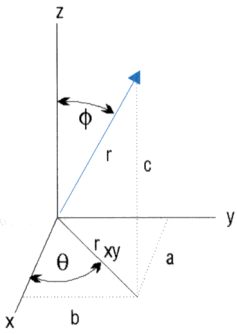


Two-Dimensional Display Modes

Rectangular	Polar
[a b]	[r ∠θ]

2D Vector Components

You can display 3D vectors as rectangular components ([*X Y Z*]), cylindrical components ([*R* ∡ *Z*]), or spherical components ([*R* ∡ ∡]) —in *Rectangular mode*, in *Cylindrical mode*, or in *Spherical mode*.



Three-Dimensional Display Modes

Rectangular	Cylindrical	Spherical
[a b c]	[rxy ∠θ c]	[r ∠θ ∠ϕ]

13

3D Vector Components

Polar mode is actually two modes—Cylindrical mode and Spherical mode. For 2D vectors, Cylindrical and Spherical modes are interchangeable—both give the same two-dimensional results.

To display rectangular components:

- Press (→)(POLAR) until *no* coordinate annunciator is on.
  or
- Press (MTH) VECTR (NXT) RECT.

To display polar (cylindrical or spherical) components:

- Press (→)(POLAR) until the R∠Z or R∠∠ coordinate annunciator is on.
  or
- Press (MTH) VECTR (NXT) CYLIN (for cylindrical/polar) or SPHER (for spherical/polar).

The ▪ in the menu label, and the coordinate annunciator, indicate the active coordinate mode:

- Rectangular mode: RECT▪, no annunciator
- Cylindrical mode: CYLI▪, R∠Z annunciator
- Spherical mode: SPHE▪, R∠∠ annunciator

Vectors are displayed inside [ ] delimiters. In rectangular form, the components are separated by spaces. In polar (cylindrical or spherical) form, angles are preceded by an angle sign (∠). (The angle is based on the current angle mode: Degrees, Radians, or Grads.) Regardless

of how vectors are displayed, the HP 48 stores them internally in rectangular form.

If you enter one type of coordinates, you can simply change the coordinate mode to convert vectors to the new coordinate mode.

Entering 2D and 3D Vectors

You can enter 2D and 3D vector components using rectangular, cylindrical/polar, or spherical/polar form.

To enter a 2D or 3D vector:

- To enter specific components, press ◀ [ ], enter the components separated by SPC or ▶ ∠, and press ENTER. (Press ▶ ∠ just before each angular component.)
- To use the current coordinate mode, enter the two or three component values and press MTH VECTR →V2 or →V3. (Don't enter ∠.)

The internal rectangular representation of all vectors has the following effects on displayed polar (cylindrical and spherical) vectors:

- θ is normalized to within $\pm 180°$ ($\pm\pi$ radians, ± 200 grads).
- ϕ is normalized to within 0 to 180° (0 to π radians, 0 to 200 grads).
- If you key in a negative r, the value is made positive; θ is increased by 180°, ϕ is subtracted from 180°, and both are normalized.
- If ϕ is 0° or 180°, θ is reduced to 0°.
- If you key in an r of 0, θ and ϕ are reduced to 0°.

To assemble a 2D or 3D vector from components on the stack:

- For a 2D vector, enter a component on level 1 and level 2, press MTH VECTR →V2. The components are interpreted according to the current coordinate mode.
- For a 3D vector, enter a component on level 1, level 2, and level 3, press MTH VECTR →V3. The components are interpreted according to the current coordinate mode.

To take apart a 2D or 3D vector on the stack:

- Press (MTH) VECTR V→ . The returned values are the same as the displayed components.

Vector Math Commands

A vector, like a real number, is a single object. So you can use vectors as arguments for commands. You can add and subtract vectors—you can multiply and divide vectors by scalars—and you can execute special vector commands (DOT, CROSS, and ABS). These special commands interpret their arguments and return results using the current coordinate mode and can be found in the MTH VECTR menu((MTH) VECTR).

Command/Description	Example	
	Input	Output
ABS *Prefix.* Returns the scalar magnitude of vector, calculated as the Frobenius norm—defined as the square root of the sum of the squares of the *absolute value* of each element.	1: [2 -3 4]	1: 5.3851648
DOT *Command.* Returns the inner or dot product (a scalar) of two vectors of equal dimensions.	2: [2 -3 4] 1: [-1 2 8]	1: 24
CROSS *Command.* Returns the cross product (a 3D-vector) of two vectors. Two element vectors will have a third zero element added during execution.	2: [2 3 4] 1: [-1 2 1]	1: [-5 -6 7]

Additional commands for manipulating vectors and elements of vectors are covered in chapter 14.

Examples: Calculating with 2D and 3D Vectors

To calculate with vectors:

- Enter the vectors on the stack, then execute the command.

Example: **Finding the Unit Vector.** A unit vector parallel to a given vector is found by dividing a vector by its magnitude:

$$\mathbf{u} = \frac{\mathbf{v}}{|\mathbf{v}|}$$

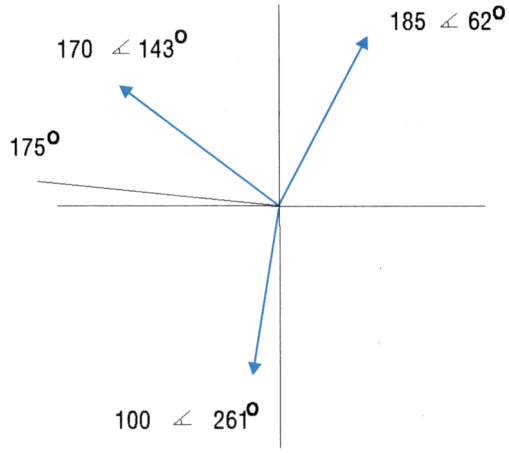
1. Enter the vector.
2. Duplicate the vector (press ENTER a second time).
3. Press MTH VECTR ABS to compute the magnitude of the vector.
4. Press ÷ to divide the vector by its magnitude to get the unit vector.

Example: **Finding the Angle between Vectors.** The angle between two vectors is given by

$$angle = \cos^{-1}\left[\frac{\mathbf{V1} \cdot \mathbf{V2}}{|\mathbf{V1}|\,|\mathbf{V2}|}\right]$$

1. Enter the two vectors onto the stack.
2. Press MTH VECTR DOT to take the dot (inner) product.
3. Press → ARG to return the two vectors to the stack.
4. Press ABS ← SWAP ABS to find the magnitude of each vector.
5. Press × to multiply the magnitudes.
6. Press ÷ to divide the product of the magnitudes into the dot product.
7. Press ← ACOS to find the angle between the vectors.

Example: **Finding the Component in One Direction.** The following diagram represents three 2D vectors. Find their sum, and then use DOT to resolve them along the 175° line. (This example assumes Degrees mode is active.)



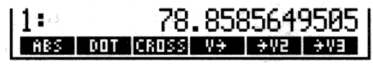
13

Step 1: Set Polar-cylindrical mode, enter the three vectors, and find their sum.

MTH VECTR NXT CYLIN
[ ] 170 ∠ 143 ENTER
[ ] 185 ∠ 62 ENTER
[ ] 100 ∠ 261 + +

Step 2: Enter the unit vector of 175°, and find the magnitude of the resultant vector along the 175° line.

[ ] 1 ∠ 175 ENTER
MTH VECTR DOT



Fast Fourier Transforms

A physical process can be described in two distinct ways:

- The change of a quantity, h, as a function of time, t ($h(t)$).
- The change of an amplitude, H, as a function of frequency, f ($H(f)$).

For many situations, it helps to consider $h(t)$ and $H(f)$ as two different representations of the *same* function. *Fourier transforms* are used to switch between these representations, or *domains.*

The HP 48 can perform *discrete* Fourier transforms, whereby a sequence of discretely sampled data can be transformed into the "other" domain. The HP 48 performs "Fast" Fourier transforms, which make use of computational efficiencies that require that the number of rows and the number of columns in the sample set to be a integral power of 2.

Fast Fourier transforms are most commonly used in analyzing one-dimensional signals or two-dimensional images. The HP 48 commands can handle both cases. In the first case the data should be entered as a vector of N elements where N is an integral power of 2 (2, 4, 8, 16, 32, ...). In the second case, the data should be entered as a matrix of M rows by N columns where both M and N are integral powers of 2.

The "forward" transformation (FFT) maps an array of M×N real or complex numbers (h_k) in the time domain to an array of M×N real or complex numbers (H_n) in the frequency domain:

$$H_k \equiv \sum_{n=0}^{N-1} h_n e^{-2\pi ikn/N}$$

The "inverse" transformation (IFFT) maps an array of M×N real or complex numbers (H_n) in the frequency domain to an array of M×N real or complex numbers (h_k) in the time domain:

$$h_n \equiv \frac{1}{N} \sum_{k=0}^{N-1} H_k e^{2\pi ikn/N}$$

To prepare an array for fast Fourier transforms:

1. Put the array of data on the stack.
2. If necessary, add zeros to the array so that all dimensions are equal to an integral power of two. See chapter 14 for efficient ways to add columns or rows of zeros to matrices.

To use a fast Fourier transform:

1. Enter the array of data to be transformed (or its name) onto the stack. Make sure its dimensions are integral powers of 2 (see above).
2. Press (MTH) (NXT) FFT FFT to transform the data from the time domain to the frequency domain.
   or
   Press (MTH) (NXT) FFT IFFT to transform the data from the frequency domain to the time domain.

Example: **Using FFT and IFFT for forward and inverse fast Fourier transforms.** The example uses the elements of a random vector to represent a sampled signal.

1. Create a 16-element random vector on the stack: enter ‹16› RANM.
2. Compute the one-dimensional discrete Fourier transform of this signal: execute FFT. The elements of the resulting vector represent the frequency components of the original signal.
3. Reconstruct the original signal by computing the one-dimensional inverse discrete Fourier transform: execute IFFT. The result is the same as the original signal, subject to small rounding errors.

You can compute two-dimensional Fourier transforms using matrices as arguments. For instance, use a random 16×16 matrix in the above example: ‹16 16› RANM.

14

Matrices and Linear Algebra

The HP 48 has extensive capabilities for entering and manipulating arrays. Array objects represent both vectors and matrices. Many of the operations described in this chapter also apply to vectors. Wherever this is the case, the more general term *array* is used instead of *matrix*.

Creating and Assembling Matrices

You can enter a matrix two ways:

- **MatrixWriter application.** A visual method of entering, viewing, and editing array elements.
- **Command line.** The basic object-entry method.

To enter a new matrix using the MatrixWriter:

1. Press (→)(MATRIX) to display the MatrixWriter screen and menu.
2. For each element in the first row, do one of the following:
   - Type the real or complex number and press (ENTER). You cannot mix real and complex elements in the same matrix.
   - Compute the element using the command line and press (ENTER). To compute an element, type in the arguments (press (SPC) to separate arguments) and press the desired function keys.
3. Press (▼) to mark the end of the first row (which specifies the number of columns in the matrix).
4. For each element in the rest of the matrix, either type in a value or compute it in the command line and then press (ENTER). Or, if you wish, enter numbers into more than one cell at a time by typing them all into the command line (pressing (SPC) to separate each number), and then pressing (ENTER) once to enter them all.

5. After you've entered all of the numbers in the matrix, press ENTER to put the matrix on the stack.

For more details about using the MatrixWriter, see chapter 8.

To enter a matrix using the command line:

1. Press ◀ [and ◀ [to type the delimiters for the matrix and for the first row.
2. Key in the first row. Press SPC to separate the elements.
3. Press ▶ to move the cursor past the] row delimiter.
4. Optional: Press ▶ ↵ (new line) to start a new row in the display.
5. Key in the rest of the matrix. You don't need [ ] delimiters for subsequent rows—they're added automatically later.
6. Press ENTER.

The HP 48 has built-in commands to automatically create special matrices that are often used in conjunction with matrices you create element by element.

To create an array filled with a given constant:

1. Enter one of the following onto the stack:
   - A list containing the dimensions of the desired constant array: { *rows columns* }.
   - Any array whose elements you don't mind changing.
2. Enter the constant that you want in the array.
3. Press MTH MATR MAKE CON. This returns an array of the same dimensions as you entered (or that the argument array had) filled with your chosen constant.

To create an identity matrix:

1. Enter one of the following onto the stack:
   - A real number representing the number of rows and columns you want in the square identity matrix (fractional values are rounded).
   - Any square matrix whose elements you don't mind changing.
2. Press MTH MATR MAKE IDN. Returns a square identity matrix of the given dimensions.

To create an array filled with random integers:

1. Enter one of the following onto the stack:
   - A list containing the dimensions of the desired random array: { *rows columns* }.
   - Any array whose elements you don't mind changing.
2. Press (MTH) MATR MAKE RANM . Returns a random array have the dimension specified by the list or array argument. The array elements are all integers in the range [-9 9]. Each integer is as likely as any other except for 0, which is twice as likely as the others.

To assemble a matrix by rows from a series of vectors:

1. Enter each vector onto the stack in the order you want them to appear in the matrix. Enter the row 1 vector first, then the row 2 vector, and so forth, entering the bottom row vector last.
2. Enter the number of rows in the desired matrix.
3. Press (MTH) MATR ROW ROW→ to assemble the vectors into a matrix.

To assemble a matrix by columns from a series of vectors:

1. Enter each vector onto the stack in the order you want them to appear in the matrix. Enter the column 1 vector first, then the column 2 vector, and so forth, entering the right-most column vector last.
2. Enter the number of columns in the desired matrix.
3. Press (MTH) MATR COL COL→ to assemble the vectors into a matrix.

To assemble a matrix with a particular diagonal from a vector:

1. Enter the vector containing the diagonal elements.
2. Enter one of the following:
   - A list containing the dimensions of the desired matrix: { *rows columns* }.
   - A real number representing the number of rows and columns in the desired square matrix.
3. Press (MTH) MATR (NXT) DIAG→ to create a matrix of the desired dimensions using the elements of the diagonal vector as the diagonal elements of the matrix. If the vector contains more diagonal elements than are needed to create the matrix, the extra

elements are discarded. If the vector does not contain enough elements to complete the matrix, the undefined diagonal elements are set to zero.

Disassembling Matrices

The HP 48 assembles and disassembles the elements of a two-dimensional matrix according to *row-major* order. Beginning with the first element (the element in row 1 and column 1), row-major order assumes that the "next" element is the next one in the *row*. If there are no more elements in the row, then the "next" element is the first element in the next row. The row-major convention thus works much like a left-to-right word processor that fills up (or removes) one line before "wrapping around" to the beginning of the next line.

To disassemble a matrix into its elements:

1. Enter the matrix onto the stack.
2. Press (PRG) TYPE OBJ→ . The matrix is disassembled in row-major order, leaving each element on its own stack level. Level 1 contains a list of the original dimensions of the matrix.

To assemble a matrix from a sequence of elements:

1. Enter the elements onto the stack in row-major order.
2. Enter a list containing the dimensions of the desired matrix: { *rows columns* }.
3. Press (PRG) TYPE →ARR to assemble the matrix.

To disassemble a matrix into row vectors:

1. Enter the matrix onto the stack.
2. Press (MTH) MATR ROW →ROW . The matrix is disassembled into row vectors (first row to last). Level 1 of the stack contains a real number representing the number of rows in the original matrix.

To disassemble a matrix into column vectors:

1. Enter the matrix onto the stack.
2. Press (MTH) MATR COL →COL . The matrix is disassembled into column vectors (first column to last). Level 1 of the stack

contains a real number representing the number of columns in the original matrix.

To extract the vector of diagonals from a matrix:

1. Enter the matrix onto the stack.
2. Press (MTH) MATR (NXT) →DIAG to extract the diagonal elements as a vector.

Inserting Rows and Columns

To insert one or more new rows into a matrix:

1. Enter the target array—the one you wish to modify—onto the stack.
2. Enter the vector, matrix, or element (when the target array is a vector) you want to insert. An inserted array must have the same number of columns as the target matrix.
3. Enter the row number where you want the first (or only) inserted row to be. The elements currently located in that row, and in all rows below, will be shifted down to accommodate the insertion. Row numbers start at 1, not 0.
4. Press (MTH) MATR ROW ROW+ to insert the new row(s).

To insert one or more new columns into an array:

1. Enter the target array—the one you wish to modify—onto the stack.
2. Enter the vector, matrix, or element (when the target array is a vector) that you wish to insert. An inserted array must have the same number of rows as the target array.
3. Enter the column number where you want the first (or only) inserted column to be. The elements currently located in that column, and in all columns to the right, will be shifted right to accommodate the insertion. Column numbers start at 1, not 0.
4. Press (MTH) MATR COL COL+ to insert the new columns.

Extracting Rows and Columns

To extract a particular row from an array:

1. Enter the array onto the stack.
2. Enter the row number (or element number if the array is a vector) that you wish to extract.
3. Press (MTH) MATR ROW ROW- . The extracted row vector (or element) is returned to level 1 and the contracted array—with the row or element deleted—is returned to level 2.

To extract a particular column from an array:

1. Enter the array onto the stack.
2. Enter the column number (or element number if the array is a vector) that you wish to extract.
3. Press (MTH) MATR COL COL- . The extracted column vector (or element) is returned to level 1 and the contracted array—with the column (or element) deleted—is returned to level 2.

Swapping Rows and Columns

To swap the location of two rows in an array:

1. Enter the array onto the stack. If the array is a vector, it is considered to be a column vector.
2. Enter the two row numbers that are to be swapped.
3. Press (MTH) MATR ROW (NXT) RSWP . The modified array is returned to level 1.

To swap the location of two columns in an array:

1. Enter the array onto the stack. If the array is a vector, it is considered to be a row vector.
2. Enter the two column numbers that are to be swapped.
3. Press (MTH) MATR COL CSWP . The modified array is returned to level 1.

Extracting and Replacing Elements of Matrices

To extract the array element at a specified position:

1. Enter the array onto the stack.
2. Enter one of the following:
   - A list containing the row and column number of the element you wish to extract: { *row column* }.
   - The position number of the element you wish to extract. (For matrices, this is interpreted as row-major order.)
3. Press (MTH) MATR MAKE (NXT) GET to extract the specified array element.

To replace an array element at a specified position:

1. Enter the array onto the stack.
2. Enter one of the following:
   - A list containing the row and column number of the element you wish to replace: { *row column* }.
   - The position number of the element you wish to replace. (For matrices, this is interpreted as row-major order.)
3. Enter the new replacement element.
4. Press (MTH) MATR MAKE (NXT) PUT to replace the element at the specified location with the new element.

Characterizing Matrices

Matrix calculations are often sensitive to special characteristics of the matrices used. The HP 48 has a number of commands that return characteristics of matrices. Note that some commands are only defined for *square* matrices, some for any rectangular matrix.

Commands for Characterizing Matrices

Key	Programmable Command	Description
(MTH) MATR MAKE :		
SIZE	SIZE	*Command.* Returns the dimensions of the array on level 1 of the stack.
(MTH) MATR NORM :		
ABS	ABS	*Prefix.* Returns the Frobenius norm of a matrix and the Euclidean length of a vector: the square root of the sums of the squares of the absolute values of the elements.
SNRM	SRNM	*Command.* Returns the spectral norm of an array. The spectral norm of a matrix is equal to the largest singular value of the matrix. Same as ABS for a vector.
RNRM	RNRM	*Command.* Returns the row norm of an array. The row norm of a matrix is the maximum value (over all rows) of the sums of the absolute values of all elements in a row. The row norm of a vector is the maximum absolute value of its elements.
CNRM	CNRM	*Command.* Returns the column norm of an array. The column norm of a matrix is the maximum value (over all columns) of the sums of the absolute values of all elements in a column. The column norm of a vector is the sum of the absolute values of its elements.

Commands for Characterizing Matrices (continued)

Key	Programmable Command	Description
SRAD	SRAD	*Command.* Returns the spectral radius of a square matrix. The spectral radius is the absolute value of the largest eigenvalue of the matrix.
COND	COND	*Command.* Returns the column-norm condition number of a square matrix. The condition number is defined to be the product of the column norm of a square matrix and the column norm of its inverse.
RANK	RANK	*Command.* Returns an estimate of the rank of a matrix. The rank of a matrix is equal to the number of nonzero singular values of the matrix. If Flag −54 is clear (default), RANK treats any computed singular value less than 10^{-14} times the size of the largest computed singular value as zero. If Flag −54 is set, RANK counts all nonzero singular values no matter their size.
DET	DET	*Command.* Returns the determinant of a square matrix. DET checks flag −54, and will refine its computed value only if −54 is clear (default).
TRACE	TRACE	*Command.* Returns the trace of a square matrix. The trace of a matrix is equal to the sum of the diagonal elements and also equal to the sum of the eigenvalues of the matrix.

Transforming Matrices

To transpose a matrix:

1. Enter the matrix onto the stack.
2. Press (MTH) MATR MAKE TRN to transpose the matrix. The first row of the original matrix is now the first column, the original second row is now the second column and so forth. This computes the conjugate transpose of complex matrices.

To invert a square matrix:

1. Enter the square matrix.
2. Press (1/x) to invert the matrix. Note that matrix inversion can produce erroneous results if you are using ill-conditioned matrices. See "Ill-Conditioned and Singular Matrices" on page 14-16.

To change the dimensions of an array:

1. Enter the array onto the stack.
2. Enter a list containing the new dimensions of the array: { *rows columns* }.
3. Press (MTH) MATR MAKE RDM to redimension the array. Elements from the original array are placed in row-major order into the newly-dimensioned array. If there are fewer elements in the new array than in the original one, the excess elements are dropped. If there are more elements in the new array than in the original one, the missing elements are filled with zeros (or (0,0) if the array is complex).

Calculating with Matrix Elements

To add or subtract two matrices:

1. Enter the two matrices onto the stack in the same order as you would to add or subtract real numbers. They must have the same dimensions.
2. Press [+] to add or [−] to subtract. The single resulting matrix has elements that are the sum or difference of the corresponding elements in the original argument matrices.

To multiply or divide a matrix by a scalar:

1. Enter the matrix onto the stack.
2. Enter the scalar (a real or complex number).
3. Press [×] or [÷]. Each element of the resulting matrix is the product or quotient of the scalar and the corresponding element in the original argument matrix.

To change the sign of each element in a matrix:

1. Enter the matrix onto the stack.
2. Press [+/−] to change the sign of each element in the matrix.

To find the matrix product (AB) of two matrices:

1. Enter the two matrices onto the stack. Be careful about the entry order because matrix multiplication is *not* commutative. Enter matrix **A** first and matrix **B** second. Remember also that the number of columns in **A** must equal the number of rows in **B**.
2. Press [×]. The result will be a matrix with the same number of rows as **A** and the same number of columns as **B**.

To multiply a matrix and vector:

1. Enter the matrix onto the stack.
2. Enter the vector. The number of elements in the vector must equal the number of columns in the matrix.
3. Press [×]. The result will be a vector with as many elements as the number of rows in the original matrix.

To "divide" an array by a square matrix:

1. Enter the array onto the stack.
2. Enter the square matrix. The number of rows in the matrix must equal the number of rows in the array (elements in the vector).
3. Press ÷ to calculate $\mathbf{Y} \cdot \mathbf{X}^{-1}$. The result will be a vector of the same size as the original. Note that array division can produce erroneous results if you are using ill-conditioned matrices. See "Ill-Conditioned and Singular Matrices" on page 14-16 .

14

Using Arrays and Array Elements in Algebraic Expressions

You can perform calculations with array elements using algebraic syntax. The array must be represented by a name in the symbolic expression or equation.

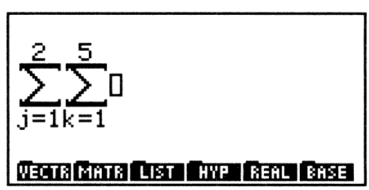
To use a matrix element in an algebraic expression:

1. Make sure that the array is stored in a named variable.
2. Create the algebraic expression and, at the point where the matrix element is to be used, type the array name and press ◀ ( ).
3. Enter the subscripts for the element:
   - For a vector, enter one subscript (position number of the element).
   - For a matrix, enter two subscripts separated by ◀ . (row and column numbers of the element).

Example: Enter a symbolic expression for the sum of all elements of a 2 × 5 matrix stored in *MATR*.

Step 1: Begin the expression.

◀ EQUATION
▶ Σ α ◀ J ▶ 1 ▶ 2 ▶
▶ Σ α ◀ K ▶ 1 ▶ 5 ▶



Step 2: Enter the name of the matrix and the subscripts.

(α)-(hold)MATR (release)
◀ ( ) α ◀ J ◀ , α ◀
K ▶

Step 3: Press ENTER to put the expression on the stack. Assuming that the 2 × 5 matrix is already stored in *MATR*, press EVAL to compute the sum of its elements.

To apply a mathematical function to each element in an array:

1. Enter an array.
2. Enter a program that contains a function. The program must take one argument and give one result.
3. Type TEACH and press ENTER.
4. Press VAR EXAM PRGS APLY. The function is applied to each element, and the result replaces that element. If the applied function returns an algebraic for one of the elements, then the array is returned in list format.

Transforming Complex Matrices

To combine two matrices into a complex matrix:

1. Enter the real matrix that will become the real part of the complex matrix.
2. Enter the real matrix that will become the imaginary part of the complex matrix.
3. Press MTH NXT CMPL R→C to combine the two real matrices into a complex matrix.

To split a complex matrix into two real matrices:

1. Enter the complex matrix on the stack.
2. Press MTH NXT CMPL C→R to split the complex matrix into its real and imaginary parts.

To conjugate each element of a complex matrix:

1. Enter the complex matrix onto the stack.
2. Press (MTH) (NXT) CMPL (NXT) CONJ to conjugate each complex element of the matrix.

To extract a matrix of real parts from a complex matrix:

1. Enter the complex matrix onto the stack.
2. Press (MTH) (NXT) CMPL RE to return a matrix containing just the real parts of each element of the original complex matrix.

To extract a matrix of imaginary parts from a complex matrix:

1. Enter the complex matrix onto the stack.
2. Press (MTH) (NXT) CMPL IM to return a matrix containing just the imaginary parts of each element of the original complex matrix.

Matrix Solutions to Systems of Linear Equations

Systems of linear equations fall into three categories:

- **Over-determined systems.** These systems have more linearly independent equations than independent variables. There is no exact solution to over-determined systems, so the "best" (least-squares) solution is sought.
- **Under-determined systems.** These systems have more independent variables than linearly independent equations. There are either no solutions or an infinite number of solutions for under-determined systems. If a solution exists, you want to find the solution with the minimum Euclidean norm; otherwise, you want to find a minimum norm least-squares solution.
- **Exactly-determined systems.** These systems have an equal number of independent variables and equations. Usually (but not always), there is a single exact solution for exactly-determined systems. (See "Ill-Conditioned and Singular Matrices" on page 14-16.)

To calculate the "best" solution to any system of linear equations:

1. Press → SOLVE ▲ ▲ OK to open the SOLVE SYSTEM A·X=B input form.
2. Enter the matrix of coefficients into field A:.
3. Enter the array (vector or matrix) of constants into field B:.
4. Press SOLVE to calculate the "best" solution and display it in the X field. If the system being solved is exactly-determined, the solution is a 12-digit approximation of the exact solution; if it is over- or under-determined, then the solution is the minimum norm least-squares solution (to 12 digits).

To estimate the solution to an over-determined system of linear equations:

1. Enter the array (vector or matrix) of constants onto the stack.
2. Enter the matrix of coefficients. It will typically have more rows than columns.
3. Press MTH MATR LSQ to calculate the "least-squares solution" (**X**) that minimizes the residual (**AX**−**B**) (the minimized Euclidean norm of the columns).

To estimate the solution to an under-determined system of linear equations:

1. Enter the array (vector or matrix) of constants onto the stack.
2. Enter the matrix of coefficients. It will typically have more columns than rows.
3. Press MTH MATR LSQ to calculate the one "least-squares solution" (**X**) of many possible least-square solutions that has the smallest Frobenius norm.

To solve an exactly-determined system of linear equations:

1. Enter the vector of constants onto the stack.
2. Enter the square matrix of coefficients. The number of columns ("variables") in the matrix must equal the number of elements in the vector.
3. Press ÷. The result will be a solution vector of the same size as the constants vector. Note that array division can produce erroneous results if you are using ill-conditioned matrices. See "Ill-Conditioned and Singular Matrices" below.

Ill-Conditioned and Singular Matrices

A *singular* matrix is a square matrix that doesn't have an inverse. You normally get an error if you use [1/x] to find the inverse of a singular matrix—or use [÷] to solve a system of linear equations having a singular coefficient matrix.

The most common cause of singular matrices are equations within a system of linear equations that are *linear combinations* of one another. That is, the coefficients of one equation can be computed exactly from the coefficients of the other. Two equations thus related are *linearly dependent* and the set of equations as a whole referred to as *dependent*.

If a set of equations is independent, but small changes in their coefficients would make them dependent, then the set of equations (and their corresponding matrix **A**) are said to be *ill-conditioned*.

To determine if a matrix is ill-conditioned:

1. Enter the matrix onto the stack.
2. Compute its condition number: Press [MTH] MATR NORM COND. If it is large, then it is ill-conditioned. If the condition number is on the order of 10^{12}, the HP 48 may not be able to distinguish it from a singular matrix.

To use ill-conditioned matrices in solving systems of linear equations:

1. Set flag -22: Press [←][MODES] 22 [+/-] FLAG SF. This is the Infinite Result Exception flag that will keep you from getting an error using a singular matrix.
2. Solve the system of linear equations. The HP 48 perturbs the singular matrix by an amount that's usually small compared to the rounding error. The calculated result corresponds to that for a matrix close to the original, ill-conditioned matrix.
3. Determine the accuracy of the computed solution using the condition number as you would for any ill-conditioned matrix (see "Determining the Accuracy of a Matrix Solution" below.
4. Compute the residual to test your results.
5. Resolve the system of linear equations using LSQ.

Determining the Accuracy of a Matrix Solution

There are two approaches to evaluating the accuracy of a computed matrix solution when you suspect that you may be using singular or ill-conditioned matrices:

- **Compute the residual array.** This array is the result of substituting the computed solution back into the original equation. The closer the residual array is to being an array of zero elements, the more accurate the solution.
- **Use the condition number.** The condition number can be used to estimate the number of accurate digits that can be expected using a given matrix.

To find the residuals for a computed solution to a system of linear equations (AX=B):

1. Enter the array (vector or matrix) of constants (**B**) onto the stack.
2. Enter the matrix of coefficients (**A**).
3. Enter the computed solution array (must be of the same type and dimensions as the constants array)(**X**).
4. Press either MTH MATR NXT RSD (or ↶ SOLVE SYS RSD). The resulting array of residuals (**AX-B**) shows how close the computed solution was to an actual solution—the smaller the absolute value of the elements, the better the solution.

To approximate the number of accurate digits in a computed solution:

1. If the elements in the matrix **A** are exact, enter 15, the maximum number of digits computed internally by the HP 48, onto the stack. If the elements in matrix **A** were rounded to 12 digits (from previous computations, for example), then enter 12.
2. Enter the matrix of coefficients (**A**).
3. Press MTH MATR NORM COND to find the condition number of the matrix.
4. Press ↷ LOG − to find the approximate number of accurate digits in a solution computed using the given matrix of coefficients. This is merely a rough, rule-of-thumb estimate of a solution's accuracy and not a precise computation of it.

Gaussian Elimination and Elementary Row Operations

The systematic process, known as *Gaussian elimination* is one of the most common approaches to solving systems of linear equations and to inverting matrices. It uses the *augmented* matrix of the system of equations, which is formed by including the vector (or vectors) of constants ($[b_1 \ldots b_m]$) as the right-most column (or columns) of the matrix of coefficients ($[a_{11} \ldots a_{mn}]$):

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \ldots & a_{1n} & b_1 \\ a_{21} & a_{22} & a_{23} & \ldots & a_{2n} & b_2 \\ a_{31} & a_{32} & a_{33} & \ldots & a_{3n} & b_3 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \ldots & a_{mn} & b_m \end{bmatrix}$$

To create an augmented matrix:

1. Enter the matrix to be augmented (the matrix of coefficients in the context of Gaussian-elimination).
2. Enter the array to be inserted (the array of constants in the context of Gaussian-elimination). It must have the same number of rows as the matrix.
3. Enter the last column number, n, of the matrix to be augmented in order to indicate where to insert the array.
4. Press (MTH) MATR COL COL+ .

Once you have an augmented matrix representing a system of linear equations, then you can proceed with the Gaussian-elimination process. The process seeks to systematically eliminate variables from equations (by reducing their coefficients to zero) so that the augmented matrix is transformed into an equivalent matrix, from which the solution can be easily computed.

Each coefficient-elimination step depends on three elementary row operations for matrices:

- Interchanging two rows.
- Multiplying one row by a nonzero constant.
- Addition of a constant multiple of one row to another row.

Gaussian elimination uses elementary row operations to convert the augmented matrix to the equivalent *row-reduced echelon* matrix, from which the solution can be computed by back-substitution.

The HP 48 offers commands for each of these elementary row operations by themselves and one that uses these row operations repeatedly until the row-reduced echelon form is generated:

- **RSWP** interchanges two rows of a matrix.
- **RCI** multiplies each element in a given row of the augmented matrix by a scalar of your choosing.
- **RCIJ** multiplies each element in a given row by a scalar *and* adds the result to another row of the matrix.
- **RREF** converts an augmented matrix into the equivalent row-reduced echelon form.

To interchange two rows of a matrix:

1. Enter the matrix onto the stack.
2. Enter the numbers of the two rows to be interchanged.
3. Press (MTH) MATR ROW (NXT) RSWP .

To multiply the elements of a row of a matrix by a nonzero factor:

1. Enter the array onto the stack.
2. Enter the nonzero factor.
3. Enter the number of the row to be multiplied.
4. Press (MTH) MATR ROW RCI .

To add the product of a nonzero factor and one row of an array to another row:

1. Enter the array onto the stack.
2. Enter the nonzero factor.
3. Enter the number of the row to be multiplied by the factor.
4. Enter the number of the row to which the product is to be added.
5. Press (MTH) MATR ROW RCIJ .

To compute the row-reduced echelon form of a matrix:

1. Enter the matrix onto the stack. If you are attempting to solve a system of linear equations, the matrix should be the augmented matrix representation of the system (see discussion above).
2. Optional: Set flag −54 if you do *not* want "tiny" elements replaced with zero during the computation: Press ◀ MODES FLAG 54 +/− SF . "Tiny" elements, which can be generated due to roundoff errors during computation, are those elements that are less than 1×10^{-14} times the size of the largest element in their column. Unless you set flag −54, the HP 48 replaces any "tiny" elements with zero.
3. Press MTH MATR FACTR RREF .

Additional Linear Algebra Topics

The HP 48 offers a selection of other linear algebra commands that give you additional problem-solving power and flexibility.

Eigenvalues and Eigenvectors

A square $(n \times n)$ matrix $\mathbf{A}$ is said to have an *eigenvalue* λ and a corresponding *eigenvector* $\mathbf{x}$ if $\mathbf{Ax} = \lambda\mathbf{x}$.

Eigenvalues are the roots of the *characteristic equation*, $\det(\mathbf{A} - \lambda\mathbf{I}) = 0$, which is a polynomial of degree n. Thus, $\mathbf{A}$ has n eigenvalues, although they are not always distinct. Each eigenvalue has a corresponding eigenvector.

The HP 48 allows you to compute either the eigenvalues only (a faster computation) or both the eigenvalues and their corresponding eigenvectors.

To compute the eigenvalues for a square matrix:

1. Enter the square $(n \times n)$ matrix onto the stack.
2. Press MTH MATR NXT EGVL to compute a vector of n eigenvalues.

To compute the eigenvalues and eigenvectors for a square matrix:

1. Enter the square ($n \times n$) matrix onto the stack.
2. Press (MTH) MATR (NXT) EGV . An $n \times n$ matrix of eigenvectors is returned to level 2 and an n-element vector of eigenvalues is returned to level 1. The columns of the matrix on level 2 represent eigenvectors corresponding to the eigenvalues in level 1.

To decompose or factor a matrix:

The HP 48 offers a set of matrix decompositions and factorizations that you may use either alone or in programmed routines to solve specialized problems. These factorizations are all found in the (MTH) MATR FACTR menu:

LU — **Crout LU Decomposition.** This procedure is used in the process of solving an exactly-determined system of linear equations, inverting a matrix, and computing the determinant of a square matrix. It factors the square matrix (**A**) into a lower-triangular matrix **L** (returned on level 3) an upper-triangular matrix **U** containing ones on its diagonal (returned on level 2), and a permutation matrix **P** (returned to level 1) such that **PA** = **LU**.

LQ — **LQ Factorization.** This procedure factors a $m \times n$ matrix **A** into an $m \times n$ lower trapezoidal matrix **L** (returned to level 3), an $n \times n$ orthogonal matrix **Q** (returned to level 2), and an $m \times m$ permutation matrix **P** (returned to level 1) such that **PA** = **LQ**.

QR — **QR Factorization.** This procedure factors a $m \times n$ matrix **A** into an $m \times m$ orthogonal matrix **Q** (returned to level 3), an $m \times n$ upper trapezoidal matrix **R** (returned to level 2), and an $n \times n$ permutation matrix **P** (returned to level 1) such that **AP** = **QR**.

SCHUR **Schur Decomposition.** This procedure factors a square matrix **A** into an orthogonal matrix **Q** (returned to level 2) and an upper-triangular matrix (or, if **A** is real-valued, the upper quasi-triangular matrix) **U** (returned to level 1) such that: $\mathbf{A} = \mathbf{QUQ}^T$ ($\mathbf{Q}^T$ is the transpose of matrix **Q**).

SVD **Singular Value Decomposition.** This procedure factors a $m \times n$ matrix **A** into an $m \times m$ orthogonal matrix **U** (returned to level 3), a $n \times n$ orthogonal matrix **V** (returned to level 2), and a vector **S** of the singular values of **A** such that: $\mathbf{A} = \mathbf{US'V}$ (**S'** is the $m \times n$ matrix formed by using the elements of **S** as its diagonal elements).

To compute the singular values of a matrix:

1. Enter the matrix onto the stack.
2. Press (MTH) MATR FACTR (NXT) SVL to return a real vector of the singular values, arranged in non-increasing order.

To reconstruct a matrix from its singular values and orthogonal factor matrices:

1. Enter the orthogonal matrix **U** onto the stack.
2. Enter the vector **S**.
3. Enter the demensions of the matrix { m n }.
4. Press (MTH) MATR (NXT) DIAG→ to construct a matrix using the singular values as its diagonal elements.
5. Press (×).
6. Enter the orthogonal factor matrix (**V**) with the same number of columns as the original matrix.
7. Press (×) to recompute the original matrix. The degree to which the recomputed matrix matches the original matrix reflects the accuracy of the decomposition.

15

Binary Arithmetic and Number Bases

The HP 48 lets you to do binary arithmetic—operations that work with binary integers.

Binary Integers and Bases

On the HP 48, binary integer objects contain from 1 to 64 bits, depending on the current *wordsize*. You can enter and display binary integers in decimal (base 10), hexadecimal (base 16), octal (base 8), or binary (base 2). The *current base* determines which base is used to display binary integers on the stack.

The # delimiter precedes a binary integer. A d, h, o, or b following the binary integer indicates its base—for example, # 182d, # B6h, # 266o, or # 10110110b.

To set the current base:

1. Press (MTH) BASE .
2. Press one of the following keys: HEX (hexadecimal), DEC (decimal), OCT (octal), or BIN (binary).

HEX, DEC, OCT, and BIN are programmable. The settings for flags −11 and −12 correspond to the current base. (For more information on flags −11 and −12, see appendix D, "System Flags.")

The choice of current base has no effect on the internal representation of binary integers.

To view the current base:

1. Press (MTH) BASE .
2. View the menu labels. The ▪ in one of the menu labels identifies the current base.

15 The choice of current base has no effect on the internal representation of binary integers.

To set the wordsize:

1. Key in a number from 1 to 64.
2. Press (MTH) BASE (NXT) STWS (the STWS command). (A fractional number is rounded to the nearest integer.)

To recall the current wordsize:

- Press (MTH) BASE (NXT) RCWS (the RCWS command).

To enter a binary integer:

1. Press (→)(#).
2. Enter the value of the binary integer—valid characters depend on the base you're using.
3. Optional: To specify the base, type a base marker: d, h, o, or b. (Otherwise, the current base is used.)
4. Press (ENTER).

Note

If a binary integer argument exceeds the current wordsize, the excess most significant (leading) bits are dropped before the command is executed. If necessary, results are also truncated. If a calculation produces a remainder, only the integer portion of the result is retained.

To add or subtract two binary integers:

1. Enter the binary integer objects.
2. Press [+] or [−].

To find the negative of a binary integer:

1. Enter the binary integer onto the stack.
2. Press [+/−] to find the "negative" of a binary number. The negative of a binary number is its *two's complement* (all bits inverted and 1 added), since there are no "negative" binary integers in the same sense that there are negative real integers. Subtracting a binary integer is the same as adding its two's complement.

To multiply or divide two binary integers:

1. Enter the two binary integers.
2. Press [×] or [÷]. Remember that any remainder from a division is lost and the answer is truncated to be an integer.

To convert a binary integer to a different number base:

1. Enter the binary integer onto the stack.
2. Press [MTH] BASE and press the menu key corresponding to the desired number base.

To convert a binary integer to a real number:

1. Enter the binary integer onto the stack. It can be in any of the four number bases.
2. Press [MTH] BASE B→R to convert the integer to a real decimal integer.

To convert a real number to a binary integer:

1. Enter the real number onto the stack.
2. Press [MTH] BASE R→B to convert the real number to a binary integer. If necessary, the real number is rounded first to an integer before converting. Negative real numbers are converted to # 0 and real numbers $\geq 1.84467440738 \times 10^{19}$ are converted to the largest binary integer (# FFFFFFFFFFFFFFFFh, for example).

Using Boolean Operators

The following table contains commands from the MTH BASE LOGIC menu (MTH BASE NXT LOGIC) that perform Boolean operations on binary integers. Unless otherwise stated, each example assumes the wordsize is set to 24.

Command/Description	Example	
	Input	Output
AND Logical bit-by-bit AND of two arguments.	2: # 1100b 1: # 1010b	1: # 1000b
NOT Returns the one's complement of the argument. Each bit in the result is the complement of the corresponding bit in the argument.	1: # FF00FFh	1: # FF00h
OR Logical bit-by-bit OR of two arguments.	2: # 1100b 1: # 1010b	1: # 1110b
XOR Logical bit-by-bit exclusive OR of two arguments.	2: # 1101b 1: # 1011b	1: # 110b

Manipulating Bits and Bytes

The following table contains commands from the MTH BASE BIT and MTH BASE BYTE menus (MTH BASE NXT BIT and ... BYTE) that are useful for manipulating binary integers either one bit or one byte at a time. Unless otherwise stated, each example assumes the wordsize is set to 24.

Command/Description	Example	
	Input	Output
ASR Arithmetic Shift Right. Performs 1 bit arithmetic right shift. The most significant bit is regenerated.	1:# 1100010b 1: # 800000h	1: # 110001b 1: # C00000h
RL Rotate Left. Binary integer rotates left one bit. (Example assumes wordsize=4.)	1: # 1100b	1: # 1001b
RLB Rotate Left Byte. Binary integer rotates left one byte.	1: # FFFFh	1: # FFFF00h
RR Rotate Right. Binary integer rotates right one bit. (Example assumes wordsize=4.)	1: # 1101b	1: # 1110b
RRB Rotate Right Byte. Binary integer rotates right one byte.	1: # A0B0C0h	1: # C0A0B0h
SL Shift Left. Binary integer shifts left one bit.	1: # 1101b	1: # 11010b
SLB Shift Left Byte. Binary integer shifts left one byte.	1: # A0B0h	1: # A0B000h
SR Shift Right. Binary integer shifts right one bit.	1: # 11011b	1: # 1101b
SRB Shift Right Byte. Binary integer shifts right one byte.	1: # A0B0C0h	1: # A0B0h

16

Date, Time and Fraction Arithmetic

The HP 48 has a sophisticated built-in clock and calendar. The HP 48 uses the Gregorian calendar which reformed the Julian calendar on October 15, 1582 CE. Dates prior to that (or after December 31, 9999) are not valid.

Calculating with Dates

The TIME command menu contains special commands that allow you to calculate calendar and clock intervals.

The following table illustrates the time and date formats available on the HP 48. The time and date illustrated is 4:31 PM on February 21, 1992.

Clock Display	Format	Number Form
Date:		
02/21/1992	Month/day/year format	2.211992
21.02.1992	Day.month.year format	21.021992
Time:		
04:31:04P	12-hour format	16.3104
16:31:04	24-hour format	16.3104

To set or change the current date format:

1. Press → TIME ▲ OK .
2. Press → ▶ to highlight the date format field.
3. Press +/– until the desired date format is displayed
4. Press OK to confirm the selection.

To put the current date on the stack (in number form):

- Press ◀ TIME DATE .

To add or subtract a given number of days from a given date:

1. Enter a date in the number form corresponding to the current date format.
2. Enter a real number representing the numbers of days you want to add or subtract. Use negative numbers if you want to subtract days.
3. Press ◀ TIME NXT DATE+. The adjusted date is returned in the number form of the current date format.

16

Example: Find the date 239 days from today. (This example assumes the current date is April 30, 1993.)

Step 1: Put the current date in level 1.

◀ TIME DATE

Step 2: Enter the number of days and calculate the future date. The result is December 25, 1993.

239 NXT DATE+

To determine the number of days between two dates:

1. Enter the first date number onto the stack.
2. Enter the second date number onto the stack.
3. Press ◀ TIME NXT DDAYS.

Example: Find the number of days between April 20, 1982 and August 2, 1986.

4.201982 ENTER 8.021986
◀ TIME NXT DDAYS

Calculating with Times

To set or change the current time format:

1. Press → TIME ▲ OK .
2. Press ▶ ▶ ▶ to highlight the time format field.
3. Press +/– one or more times until the desired format is displayed.
4. Press OK to confirm your choice.

To put the current time on the stack (in number form):

- Press ← TIME TIME .

A number with HMS (hours-minutes-seconds) format is represented as *H.MMSSs*:

H	Zero or more digits representing the number of hours.
MM	Two digits representing the number of minutes.
SS	Two digits representing the number of seconds.
s	Zero or more digits representing the decimal fraction part of seconds.

To convert a time number from decimal hours to HMS format:

1. Enter the time number in decimal form.
2. Press ← TIME NXT →HMS .

To convert a time number from HMS format to decimal hours:

1. Enter the time number in HMS form.
2. Press ← TIME NXT HMS→ .

To add two time numbers in HMS format:

1. Enter the two time numbers in HMS format.
2. Press ◀ TIME NXT HMS+ . The time number returned is also in HMS format.

To subtract two time numbers in HMS format:

1. Enter the two time numbers in HMS format.
2. Press ◀ TIME NXT HMS- . The time number returned is also in HMS format.

To convert a time and date into a text string:

1. Enter the date number onto the stack.
2. Enter the time number.
3. Press ◀ TIME NXT NXT TSTR . A text string will be returned showing the day of the week, the date (in current date format) and the time (in the current time format).

To put the system time on the stack:

1. Press ◀ TIME TICKS. System time is kept in "ticks" of the clock. Each tick is 1/8192 of a second in duration. The total number of ticks is reported as a binary integer. TICKS is useful for measuring elapsed time, and can be converted into a standard time number in either decimal or HMS format.

To convert a system time (ticks) into HMS format:

1. Enter the system time. It should be a binary number.
2. Press MTH BASE B→R to convert the system time to a real number.
3. Press 29491200 ÷ to convert to decimal hours.
4. Press ◀ TIME NXT →HMS to convert decimal hours to HMS format.

To calculate elapsed time in seconds:

- Press ◀ TIME TICKS to start timing.
- Press TICKS to stop timing.
- Press SWAP − to get the elapsed time in clock ticks.
- Press MTH BASE B→R 8192 ÷ to calculate the elapsed time in seconds.

Calculating with Fractions

A *fraction* is an algebraic representation of an arithmetic operation that hasn't yet been evaluated. For example, the fraction $\frac{4}{3}$ is a means of representing the result of a division operation $4 \div 3$ without actually dividing. The mixed fraction $4\frac{5}{6}$ represents the algebraic operation $4 + (5 \div 6)$. The HP 48 uses these algebraic representations to display fractions:

```
'4/3' '4+5/6'
```


To enter a fraction:

- Using the EquationWriter:
  1. Press ◀ EQUATION.
  2. Type in the fraction. Press ▲ to begin a numerator and ▼ (or ▶) to move from the numerator to the denominator.
  3. Press ENTER
- Using the command line:
  1. Press ' (because the fraction is an algebraic).
  2. Type in the fraction onto the command line.
  3. Press ENTER.

To add, subtract, multiply, and divide with fractions:

1. Enter the fractions onto the stack in the same order as you would if they were decimal numbers.
2. Execute the operation (+, −, ×, or ÷).
3. Press EVAL to convert the expression into a decimal answer.

To convert a decimal into a fraction:

1. Put the decimal on level 1 of the stack.
2. If necessary, change the display mode. The accuracy of the fractional approximation is dependent on the display mode. If the display mode is Std, the approximation is accurate to 11 significant digits. If the display mode is *n* Fix, the approximation is accurate to *n* significant digits.
3. Press ◀ SYMBOLIC NXT →Q.

To convert a fraction into a decimal:

- Put the fraction on level 1 of the stack.
- Press EVAL.

To convert a decimal into a fraction involving π:

1. Put the decimal on level 1 of the stack.
2. If necessary change the display mode to indicate the accuracy of the fractional approximation you desire.
3. Press ◀ SYMBOLIC NXT →Qπ. →Qπ computes both the fractional equivalent of the original number *and* the fractional equivalent of the original number divided by π, and then compares the denominators. It returns the fraction with the smallest denominator—this fraction might be the same fraction returned by →Q, or it might be a different fraction multiplied by π.

Example: Convert 7.896 to a pure fraction using →Q.

7.896 ◀ SYMBOLIC NXT →Q

17

Lists and Sequences

Creating Lists

To enter a list from the keyboard:

1. Use ◀ {} to indicate the beginning and end of a list.
2. Enter the elements of the list. Use SPC to separate each element.

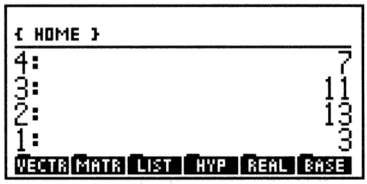
To assemble a series of elements into a list:

1. Enter the elements on the stack.
2. Enter the number of elements on the first level of the stack.
3. Use PRG LIST →LIST to convert the stack elements to a list.

Example: Create a list with the following elements 7 11 13 using →LIST.

Step 1: Enter the elements and number of elements on the stack.

7 ENTER 11 ENTER 13 ENTER 3 ENTER



Step 2: Convert the stack to a list.

PRG LIST →LIST

To append a new element to the beginning of a list:

1. Enter the new element.
2. Enter the list.
3. Press +.

To append a new element to the end of a list:

1. Enter the list.
2. Enter the new element.
3. Press [+].

List Processing

The way the calculator performs operations on lists is called list processing.

17

To apply a one-argument command to each element in a list:

1. Enter a list.
2. Execute the command.

Example: Find the factorial of 3, 4, and 5.

Step 1: Enter the numbers in a list.

[←] [{}] 3 [SPC] 4 [SPC] 5 [ENTER]

Step 2: Find the factorials of the elements.

[MTH] [NXT] PROB !

To perform a two-argument command using a list and a number:

1. Enter the list.
2. Enter the number.
3. Execute the command.

Note that you must use [MTH] LIST ADD instead of [+] to add a number to each element of a list.

Example: How many combinations can be made from 4 objects taken 3 at a time? How about 5 objects? And 6 objects?

Step 1: Enter a list that contains the objects, and enter the number of objects taken at a time on level 1.

◀{} 4 SPC 5 SPC 6 ENTER 3 ENTER

Step 2: Find the number of combinations.

MTH NXT PROB COMB

Multiple-Argument Commands with Lists

Operations that operate on two objects can also operate on individual corresponding elements of two lists.

To add corresponding elements of two lists:

1. Enter both lists.
2. Execute the ADD command.

Example: Add { 3 2 1 } and { 4 5 6 }.

Step 1: Enter the two lists.

◀{} 3 SPC 2 SPC 1 ENTER ◀{} 4 SPC 5 SPC 6 ENTER

Step 2: Add the corresponding elements of the lists.

MTH LIST ADD

To concatenate two lists:

1. Enter the list whose elements will form the first part of the concatenated list.
2. Enter the list whose elements will form the latter part of the concatenated list.
3. Press +.

Example: Concatenate { 3 2 1 } and { 4 5 6 }.

Step 1: Enter the two lists.

◀{} 3 SPC 2 SPC 1 ENTER ◀{} 4 SPC 5 SPC 6 ENTER

Step 2: Concatenate the lists.

+

17

To subtract, multiply, or divide corresponding elements of two lists:

1. Enter both lists.
2. Execute the operation.

Applying a Function or Program to a List (DOLIST)

You can run programs or execute functions on groups of lists.

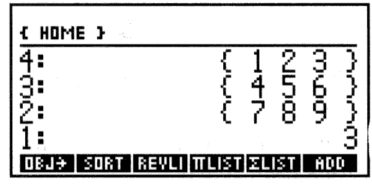
To run a program or execute functions on lists:

1. Enter the lists.
2. Enter the number of lists operated on. This number is actually the number of elements operated on with each iteration of the function or program.
3. Enter a program or function.
4. Execute DOLIST.

Example: Create 3 lists (a, b, and c) and execute a function that does the following to each element $a_n + (b_n \times c_n)$.

Step 1: Enter the lists and the number of lists to be operated on (3).

◀{} 1 SPC 2 SPC 3 ENTER ◀{} 4 SPC 5 SPC 6 ENTER ◀{} 7 SPC 8 SPC 9 ENTER 3 ENTER



Step 2: Enter the function and execute it.

◀ « » × + ENTER PRG
LIST PROC DOLIS

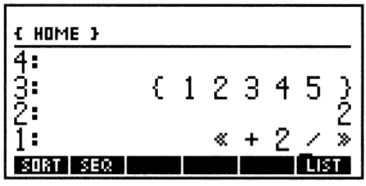
To apply a procedure sequentially to elements of a list:

1. Enter the list.
2. Enter the frame index. This is the number of elements affected by each iteration of the function. For example, entering 3 would take 3 elements from the list and use them as arguments for the function.
3. Enter the function.
4. Execute DOSUB.

Example: Find the 2-element moving average of { 1 2 3 4 5 }.

Step 1: Enter the list, the frame index, and the function.

◀ { } 1 SPC 2 SPC 3 SPC
4 SPC 5 ENTER 2 ENTER
◀ « » + 2 ÷ ENTER



Step 2: Execute DOSUB

PRG LIST PROC DOSUB

When writing programs with DOSUB the frame number (the position of the first object in the frame) is NSUB, and the number of frames is ENDSUB.

Applying a Function Recursively to a List

The STREAM command lets you apply a function recursively to every element of a list.

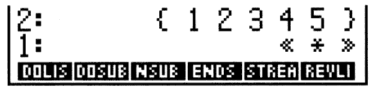
To execute a function on every element of a list:

1. Enter the list.
2. Enter a program or function. A program or function that takes two arguments and returns one result is ideal.

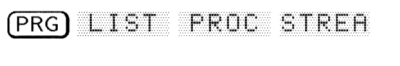
3. Execute STREAM. STREAM is designed to take the first two elements, excute the operation, then take the result and the next element and execute again. This process continues until all the elements have been used as arguments.

Example: Multiply all of the elements in { 1 2 3 4 5 } together.

Step 1: Enter the list and the function.



Step 2: Execute the function.



List Manipulations

The following functions provide ways to manipulate the elements of a list:

- MTH LIST SORT sorts the elements of a list in ascending order. The list must be on level 1.
- MTH LIST REVLI reverses the elements of a list. The list must be on level 1.
- + adds items to the begining or end of a list, or concatenates two lists. To add an item to begining of a list, enter the item, enter the list, and then press +. To add an item to the end of a list, enter the list, enter the item, and then press +.
- PRG LIST ELEM NXT HEAD replaces the list on level 1 with the list's first element.
- PRG LIST ELEM NXT TAIL replaces the list on level 1 with all but the first element of the list.
- PRG LIST ELEM GET replaces the list on level 2 and position index on level 1 with the element at that position within the list.

- PRG LIST ELEM GETI similar to GET, but it also increments a position index. The new index is put on level 2. The original list is put on level 3.
- PRG LIST ELEM PUT takes an object from level 1 and replaces an existing object within a list. You must supply a position index on level 2, and a list on level 3. The resulting list is on level 1.
- PRG LIST ELEM PUTI similar to PUT, but it also increments a position index. The new index is put on level 1. The new list is on level 2.
- PRG LIST ELEM SIZE replaces the list on level 1 with the number of elements in the list.
- PRG LIST ELEM POS replaces a list on level 2 and an element from that list on level 1 with a position index for the first occurrence of that element. If the element is not found, 0 is returned.
- PRG LIST OBJ→ puts each object from a list on level 1 onto the stack, plus puts the number of objects in level 1.
- PRG LIST SUB returns a list of elements from the list on level 3 specified by the starting and ending positions in levels 2 and 1.
- PRG LIST REPL replaces elements of a list on level 3 with elements from a list on level 1 starting at the element specified in level 2.

Sequences

Sequence commands automate the generation of a list from the repeated execution of a function or program.

To generate a sequence:

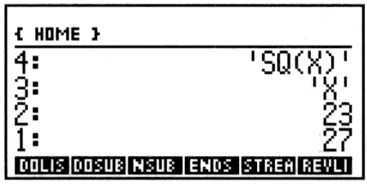
1. Enter the function or program (or its name).
2. Enter the index variable name.
3. Enter the initial value for the variable.
4. Enter the final value for the variable.

5. Enter the step size of the increment. The number of elements generated is the integer portion of $\frac{(final-initial)}{step} + 1$.
6. Execute SEQ.

Example: Generate a list of squares from 23 to 27.

Step 1: Enter the function, the variable name, the initial value, and the final value.

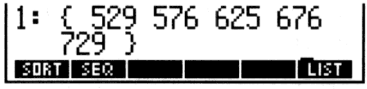
' ◀ x^2 α X ENTER ' α X ENTER 23 ENTER 27 ENTER



17

Step 2: Enter the step size, and generate the sequence.

1 ENTER PRG LIST PROC NXT SEQ



To find the sum of a finite sequence expressed as a list:

1. Enter the list.
2. Execute MTH LIST ΣLIST.

You can also find the sum of a finite sequence using the Σ function in an algebraic expression—see page 7-5.

To find the product of a finite sequence expressed as a list:

1. Enter the list.
2. Press MTH LIST πLIST.

To find the set of first differences of a finite sequence:

1. Enter the sequence as a list.
2. Press MTH LIST NXT ΔLIST.

The first differences for the list $\{\ x_1\ x_2\ \ldots\ x_n\ \}$ is defined as $\{\ x_2-x_1\ \ldots\ \ x_n-x_{n-1}\ \}$.

18

Solving Equations

Solving an Equation for an Unknown Variable

To solve an equation for numeric answers by hand, you might use the following general procedure:

1. Write down the equation you want to solve.
2. If possible, manipulate the equation to solve for the unknown variable.
3. Substitute known values for the given variables.
4. Calculate the value of the unknown variable.

When you use the SOLVE application, you follow a similar procedure—*except* you don't need to do step 2, and that simplifies the process.

To solve an equation for an unknown variable:

1. Press (→)(SOLVE) OK .
2. Enter or choose the equation you wish to solve.
3. Enter values for all of the known variables.
4. Optional: Enter a guess value for the unknown variable. This can make the search quicker or guide the root-finder to a particular one of several possible roots for a given equation.
5. Move the highlight to the unknown variable and press SOLVE.

The SOLVE application can solve for the numeric value of a variable in an equation, expression, or program:

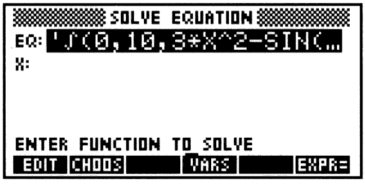
- **Equation.** An equation is an algebraic object containing = (for example, 'A+B=C'). A solution is a value of the unknown variable that causes both sides to have the same numeric value.
- **Expression.** An expression is an algebraic object not containing = (for example, 'A+B+C'). A solution is a *root* of the expression—a

value of the unknown variable for which the expression has a value of 0.

- **Program.** A program to be solved must return one real number. A solution is a value of the unknown variable for which the program returns 0.

To enter a new equation to solve:

1. Open the SOLVE application, if necessary, by pressing ▶ SOLVE. There may or may not already be an equation entered into the form.



2. Make sure the highlight is located in the EQ: field and do one of the following:
   - Type the equation, expression, or program (with proper delimiters) into the command line and press ENTER.
   - Press ◀ EQUATION, then type in the equation or expression in the EquationWriter, and then press ENTER.

To select a previously-created equation to solve:

1. Open the SOLVE application, if necessary, by pressing ▶ SOLVE.
2. Make sure the highlight is located in the EQ: field and press CHOOS.
3. Use the arrow keys to find the desired variable. If it isn't in the current directory, press CHOOS again, select the proper directory, and press OK. Then find the desired variable and press OK once more to enter the variable into the EQ: field.

Whenever an equation is entered into the EQ: field, there will also be variable names displayed as well. There will be a label for each variable in the current equation, except if a variable contains an algebraic object, in which case labels are included for the variables in the *algebraic* itself. For example, if the current equation is 'A=B+C',

and B contains the expression 'D+TAN(E)', you will see labels for A, D, E, and C.

Note

For equations that use a *placeholder* variable—such as an integral, summation, or derivative, the placeholder variable will have a label displayed. However, you will *not* be able to solve for a placeholder variable.

If one or more of the variables does not already exist, it will be created and added to the current directory when you solve the equation the first time.

To enter a value for a known variable:

1. Move the highlight to the field labeled with the name of the known variable.
2. Type in the value and press ENTER.

To store an estimated value for an unknown variable:

1. Move the highlight to the field labeled with the name of the unknown variable.
2. Type in the guess value and press ENTER.

To solve for an unknown variable:

1. Move the highlight to the field labeled with the name of the unknown variable.
2. Press SOLVE. The result will be displayed in the field and a labeled copy put on the stack.

Interpreting Results

The SOLVE application returns a message describing the result of the root-finding process. You can use this message and other information to judge whether the result is a root of your equation.

To interpret the computed result:

- After the result has been computed, press INFO. Press OK to clear it when you've finished viewing it.

The message is based on the *value of the equation*—the difference between the left and right sides of an equation, or the value returned by an expression or program.

If a root is found, the SOLVE application returns a message describing the root:

`Zero`	The SOLVE application found a point where the value of the equation is 0 within the calculator's 12-digit precision.
`Sign Reversal`	The SOLVE application found two points where the value of the equation has opposite signs, but it cannot find a point in between where the value is 0. This may be because: ■ The two points are neighbors (they differ by 1 in the 12th digit). ■ The equation is not real-valued between the two points. The SOLVE application returns the point where the value is closer to 0. If the value of the equation is a continuous real function, this point is the SOLVE application's best approximation of an actual root.
`Extremum`	One of the following occurred: ■ The SOLVE application found a point where the value of the equation approximates a local minimum (for positive values) or maximum (for negative values). The point may or may not represent a root. ■ The SOLVE application stopped searching at ±9.99999999999E499, the largest or smallest numbers in the calculator's range of numbers.

To obtain more information about the solution:

- Do any or all of the following:
  - Move the highlight to the `EQ` field and press `EXPR=` (ENTER). For an expression or program, the closer the result (labeled `Expr:`) is to 0, or for an equation, the closer the two results (labeled `Left:` and `Right:`) are to each other, the more likely it is that the SOLVE application found a root. You must use judgement in considering the results.

- Plot the expression or equation in the region of the answer. The PLOT application will show any local minimum, maximum, or discontinuity.
- Check the system flags that detect mathematical errors (see appendix B for a list of such errors). For example, flag −25 indicates whether overflow occurred.

If the SOLVE application can't return a result, it displays a message indicating the reason:

`Bad Guess(es)`	One or more of the initial guesses lie outside the domain of the equation—or units for the unknown variable aren't consistent with the units for the other variables. Therefore, when the equation was evaluated, it didn't return a real number or it generated an error.
`Constant?`	The value of the equation is the same value at every point sampled.

Solving Options

To display the root-finder at work:

1. Immediately after pressing `SOLVE` to begin the root-finder, press ENTER. You will see two intermediate guesses and the sign of the expression evaluated at each guess (displayed to the left of each guess).

```
- -.219330555745
+ -1.31111111149
```

   Watching the intermediate guesses can give you information about the root-finder's progress—whether the root-finder has found a sign reversal (the guesses have opposite signs), or if it is converging on a local minimum or maximum (the guesses have the same signs), or if it is not converging at all. In the latter case, you may want to halt the root-finder and restart with a new guess.

To halt and restart the root-finder:

1. While the root-finder is working, press CANCEL. The root-finder halts and displays current guess in the unknown variable's field.
2. To restart the root-finder, do one of the following:

- Press SOLVE to restart from where it left off.
- Enter a guess into the unknown variable's field and press SOLVE to restart the root-finder searching in a different region.

To use units while solving for an unknown variable:

1. To store a value *with units* in a variable, enter the unit object into the desired variable field. Note that all variables must contain a consistent set of units—*including the unknown variable*—before solving (or else you will generate a Bad Guess error).
2. To change a variable's value *and keep its old units*, enter the number only.
3. To return a solution with units attached, enter a guess for the unknown variable that includes *the desired units*, before pressing SOLVE.

To rearrange the order in which the variables appear:

1. From the main SOLVE EQUATION screen, press VARS and do one of the following:
   - Press EDIT, edit the variable list into the order that you want, and press ENTER.
   - Press ← {}, type the variables in the order you wish them to appear, and press ENTER.
2. Press OK to record the changes and return to the main SOLVE screen. You will see the variables in their new order.

SOLVR: An Alternative Solving Environment

There is an alternative solving environment available on the HP 48G Series calculators that works like the solving environment on their predecessors, the HP 48S and HP 48SX. This environment, SOLVR, makes use of the same built-in root-finder as the SOLVE application, but allows you to continue to see and use the stack while you are "in" the environment.

While the approach to solving equations is similar for both solving environments, the procedures used are somewhat different.

To enter the equation to be solved:

1. Put the equation (or properly-designed program) on stack level 1. You may type it in using the command line or the EquationWriter or recall it from a variable.
2. Press ↶ SOLVE ROOT .
3. Press ↶ EQ (or type STEQ and press ENTER) to make the equation on level 1 the current equation.

To enter the SOLVR root-finding environment:

- Press ↶ SOLVE ROOT SOLVR to enter the SOLVR environment. You will see the variables in the current equation displayed as *white* menu labels in the bottom row of the display and the current equation (stored in EQ:) displayed in the top row. For example:

To enter values for known variables using SOLVR:

1. If necessary, enter the SOLVR environment.
2. Type in the value of the known variable and press the menu key corresponding to the variable's white menu label.

To recall the value of a known variable:

- Press ↷ and then the menu key corresponding to the variable's white menu label.

To solve for an unknown variable in the SOLVR:

1. Make sure that all known variables have values stored in them.
2. Optional: Enter a guess estimate for the value of the unknown variable by typing in the guess and pressing the unknown variable's white menu key.

3. Press ◀ and then the unknown variable's white menu key. The labeled result will be returned to stack level 1 and a message interpreting the result is displayed at the top of the screen.
4. Optional: Press EXPR= to evaluate the current equation using the newly computed value for the unknown variable. See "Interpreting Results" on page 18-3 for more details about intrepreting the results of the root-finder.

Additional Solving Options in SOLVR

In addition to providing a different environment for finding roots of equations, expressions and programs, SOLVR allows you a few options not available in the SOLVE application.

To sequentially solve a series of equations using SOLVR:

1. Enter the equations that you want to use onto the stack in the order that you will probably want to solve them. Begin with the equation containing only one unknown variable. The other equations may begin with additional unknown variables, although they should have only one variable that remains unknown after each of the equations preceeding it have been solved.
2. Press ▲ repeatedly until the stack pointer indicates the first equation to be solved.
3. Press →LIST ENTER to gather the equations into a list.
4. Press ◀ SOLVE ROOT ◀ EQ to store the list into EQ as the current "equation."
5. Enter the SOLVR environment, enter values into the known variables, and solve for the unknown variable in the first equation as you would if there were only one equation.
6. Press NXEQ (you may need to press NXT one or more times if there are many variables) to make the "next" equation in the list into the current equation. The equations are actually rotated within the list, so that the first equation is now the last, the second equation now the first, the third now the second, and so forth.
7. Enter any additional known values and solve for the remaining unknown variable in the equation.
8. Repeat steps 6 and 7 until you have solved for all of the unknown variables in the series of equations.

You may prefer to use the Multiple Equation Solver instead of SOLVR for this kind of task (see page 25-6).

To create a customized SOLVR menu:

1. Enter a *solver-list* in level 1 of the stack. The syntax of a solver-list is { *equation* { *key-definitions* } }, where:

   equation — Specifies the equation. It can be an equation or expression (with ' delimiters), a program object (with « » delimiters), or the name of an equation, expression, or program.

   key-definitions — Specifies the menu keys—each entry defines one key. Each entry can be either a variable name or other type of object. Variable names will be displayed with white menu labels, other objects will be displayed with black menu labels. To include a program that you can execute, enter its *name* in the key-definition as a sublist of the form { "*label*" « *name* » }.

2. Press ◀ SOLVE ROOT ◀ EQ to make the equation in the solver-list the current equation and display the custom SOLVR menu according to the key definitions.

Example: The equation $I = 2\pi^2 f^2 \rho v a^2$ calculates the intensity of a sound wave. Suppose you always calculate the value of ρ and store it in the corresponding variable *prior* to using this equation, and so would like to suppress ρ from the SOLVR menu. Furthermore, suppose you want the IP command available in the SOLVR menu so that you can store integer values in the variables in the SOLVR menu. The following solver-list list includes two additional keys: a blank key and a key that executes IP (integer part) and suppresses ρ.

```
{ 'I=2*π^2*f^2*ρ*v*a^2' { I f v a { } IP } }
```

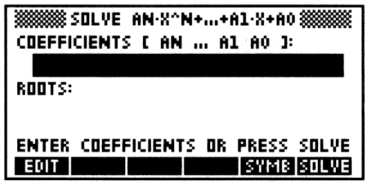
The list, when stored in *EQ*, creates this menu of variables and functions:

Finding All Roots of a Polynomial

A symbolic polynomial, such as $x^3 + 4x^2 - 7x + 9$, can also be expressed as a vector of its coefficients: [1 4 −7 9]. In this vector form, very efficient numeric techniques can be applied to find a vector of its roots.

To use the polynomial root-finder:

- Press ▶ SOLVE ▼ ▼ OK to display the following:



18

To find all roots of a polynomial:

1. Open the polynomial root-finder.
2. Move the highlight to the COEFFICIENTS field, if necessary.
3. Enter the polynomial in coefficient form. Use either the command line (don't forget the [ ] delimiters)or the MatrixWriter. Note that the first element in the vector should be the coefficient of the highest order term and the last element should be the constant term. Remember to include zeroes wherever "missing" terms are located in the polynomial.
4. With the highlight located in the ROOTS: field, press SOLVE. A complex array of roots is displayed in the ROOTS: field and a labeled copy sent to the stack.

To find a polynomial, given a set of roots:

1. Open the polynomial root-finder.
2. Move the highlight to the ROOTS: field, if necessary.
3. Enter the set of roots as a vector. Remember that if any of the roots are complex, then you must enter all roots as complex (real roots are entered as (*real*, 0)).
4. Move the highlight to the COEFFICIENTS field and press SOLVE.

To evaluate a polynomial at a given value:

1. Enter the polynomial in coefficient form onto the stack.
2. Enter the given value at which you want to evaluate the polynomial.
3. Press ◀ SOLVE POLY PEVAL.

To convert a polynomial from coefficient to algebraic form:

1. Open the polynomial root-finder.
2. If it doesn't already contain the polynomial in coefficient form, enter it into the COEFFICIENTS field.
3. With the highlight on the COEFFICIENTS field, press SYMB. The symbolic polynomial is sent to the stack using X as the variable.

Solving a System of Linear Equations:

The HP 48 can solve a system of linear equations. To create your system of equations, you can either choose them from among those you've stored, or enter them directly.

As you solve a system of equations, remember that a system of equations can be represented by a single matrix equation of the form **A·X = B**:

Equation Form → **Matrix Form**

$$\begin{aligned} ax + by + cz &= k_1 \\ dx + ey - fz &= k_2 \\ gx + hy + iz &= k_3 \end{aligned} \quad \rightarrow \quad \begin{bmatrix} a & b & c \\ d & e & -f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} k_1 \\ k_2 \\ k_3 \end{bmatrix}$$

The HP 48 makes use of this representation to solve systems of linear equations rapidly and efficiently.

To solve a system of linear equations:

1. Press ▶ SOLVE ▲ ▲ OK to begin the Linear System Solver.
2. Enter the matrix of coefficients in the A: field. You can use either the MatrixWriter or the command line.

3. Enter the array of constants in the B: field.
4. Move the highlight (if necessary) to the X: field and press SOLVE. The result array (with the same dimensions as the constants array, B) is displayed in the X: field and a labeled copy placed on the stack.
5. Optional: Press EDIT to view the result in the MatrixWriter.

The Linear System Solver will return a result array for any of the following systems:

- **Exactly-Determined Systems.** The number of equations equals the number of independent variables in the system. The result returned will be exact (to within the limits of precision of the HP 48), as long as the coefficient matrix is not ill-conditioned (see "Ill-Conditioned and Singular Matrices" on page 14-16).
- **Over-Determined Systems.** The number of equations is greater than the number of variables in the system. There is usually no exact solution to over-determined systems so the least-squares solution is returned.
- **Under-Determined Systems.** The number of equations is smaller than the number of independent variables in the system. There are usually an infinite number of solutions to under-determined systems so the one with the minimum Euclidean norm is returned.

Be sure to pay attention to the nature of the linear system you are solving because it will influence how you should interpret the result array. In some cases, you will want to test for ill-conditioning (see page 14-16) before accepting even an exact solution as a "true" solution.

Another approach to testing the validity of an answer is to find the residual of the solution ($\mathbf{A}\cdot\mathbf{X} - \mathbf{B}$). Accurate solutions have residuals close to zero.

To find the residual ($A\cdot X - B$) of a solution:

1. Make sure the computed solution array is on stack level 1, and press ENTER to duplicate it.
2. Enter the constants array (**B**) on the stack.
3. Enter the matrix of coefficients (**A**).
4. Press ← STACK ROT to move the solution array back to level 1.
5. Press ← SOLVE SYS RSD to compute the residual of the solution.

See 14-17 for additional approaches to testing the accuracy of a computed array solution.

Using the Finance Solver

The Finance Solver application provides time-value-of-money (TVM) and amortization capabilities. You can use it for compound-interest and amortization calculations.

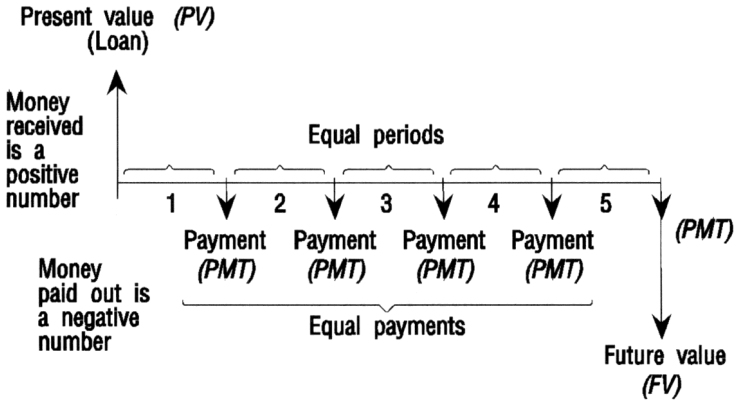
Compound interest occurs when earned interest is added to the principal at specified compounding periods, and then the combined amount earns interest. Many financial calculations are compound interest calculations—for example, savings accounts, mortgages, pension funds, leases, and annuities.

Time Value of Money calculations, as the name implies make use the notion that "time is money."—that a dollar now is worth more than a dollar at some time in the future. A dollar now can be invested and generate a return that the dollar-in-the-future cannot. This TVM principle underlies the notion of interest rates, compound interest and rates of return.

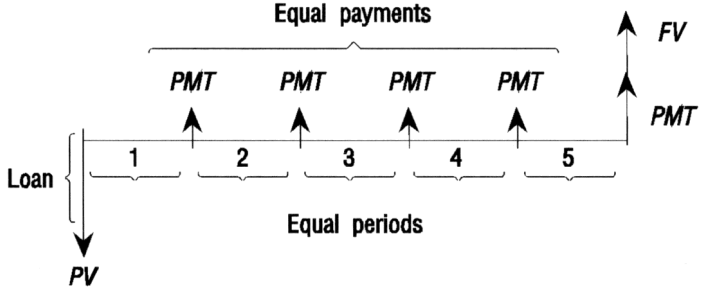
TVM transactions can be represented and understood by using *cash flow diagrams*. A cash flow diagram is a time line divided into equal segments representing the compounding periods. Arrows represent the cash flows. Money received is a positive value, and money paid out is a negative value.

The cash flow diagram for a transaction depends on the point of view you take in your problem statement. For example, a loan is an initial positive cash flow for the borrower, but it's an initial negative cash flow for the lender.

The following cash flow diagram shows a loan from a *borrower's* point of view.

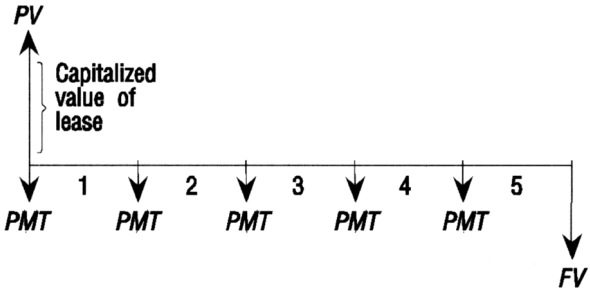


The following cash flow diagram shows a loan from a *lender's* point of view.

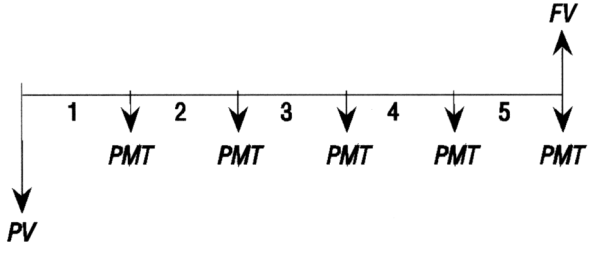


In addition, cash flow diagrams specify *when* payments occur relative to the compounding periods: at the *beginning* of each period or at the *end*. The Finance Solver application provides both of these payment modes: Begin mode and End mode.

The following cash flow diagram shows lease payments at the *beginning* of each period.



The following cash flow diagram shows deposits into an account at the *end* of each period.



As the foregoing cash-flow diagrams imply, there are five TVM variables:

N	The total number of compounding periods or payments.
I%YR	The nominal annual interest rate (or investment rate). This rate is divided by the number of payments per year (P/YR) to compute the nominal

interest rate *per compounding period*—which is the interest rate actually used in TVM calculations.

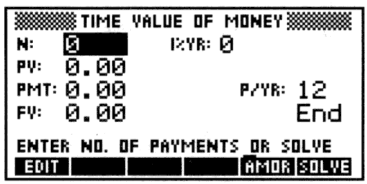
PV — The present value of the initial cash flow. To a lender or borrower, PV is the amount of the loan; to an investor, PV is the initial investment. PV always occurs at the beginning of the first period.

PMT — The periodic payment amount. The payments are the same amount each period and the TVM calculation assumes that no payments are skipped. Payments can occur at the beginning or the end of each compounding period—an option you control by setting the Payment mode to Beg or End.

FV — The future value of the transaction: the amount of the final cash flow or the compounded value of the series of previous cash flows. For a loan, this is the size of the final balloon payment (beyond any regular payment due). For an investment this is the cash value of an investment at the end of the investment period.

To perform a TVM calculation:

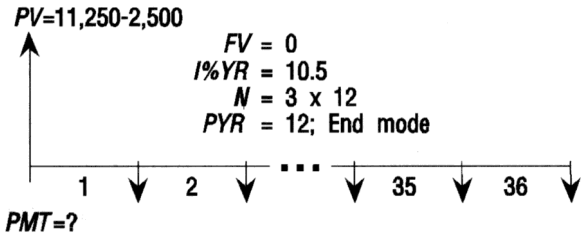
1. Press ▶ SOLVE ▲ OK to begin the Finance Solver application.



The Finance Solver

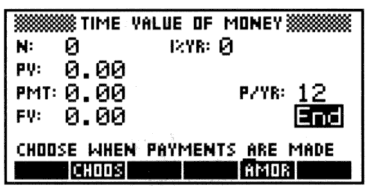
2. Move the highlight to a field labeled as a TVM variable, type the appropriate value and press ENTER. Be sure that values are entered for at least four of the five TVM variables.
3. If necessary, enter a different value for P/YR.
4. If necessary, press +/− to change the Payment mode (Beg or End) as required.
5. Move the highlight to the TVM variable you wish to solve for and press SOLVE.

Example: Otto Tailfin is financing the purchase of a car with a 3-year loan at 10.5% annual interest, compounded monthly. The purchase price of the car is $11,250, and his down payment is $2500. What are his monthly payments? What is the largest loan Otto can afford if his maximum monthly payment is $225? (Assume that payments start at the end of the first period.)



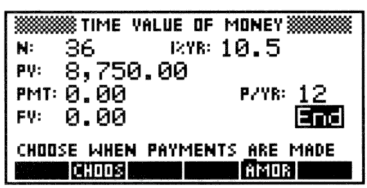
Step 1: Open the Finance Solver, and make sure there are 12 payments/year (monthly payments) and that payments are made at the end of each compounding period.

→ SOLVE ▲ OK
◀ ▲ 12 ENTER ▶ +/− (if necessary)



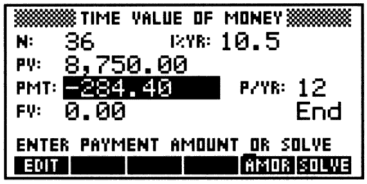
Step 2: Enter the known TVM variables. Make sure to set the FV to 0 because the loan is fully paid after 3 years (3 × 12 payments).

→ ◀ 36 ENTER 10.5 ENTER
NXT CALC 11250 ENTER
2500 − OK NXT ▼ ▼ 0
ENTER



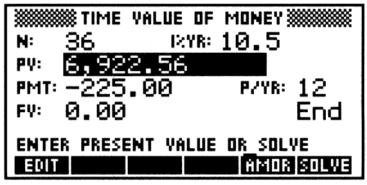
Step 3: Solve for the payment.

◀ ▲ SOLVE



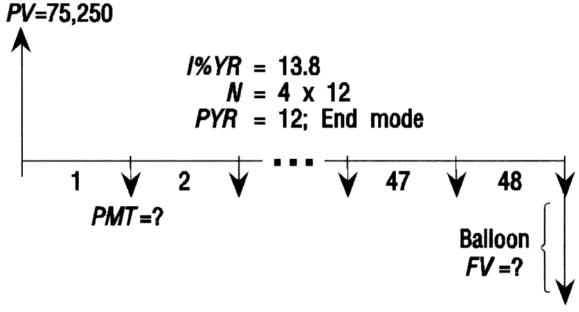
Step 4: Put in -225.00 for the payment and solve for PV to see how much Otto can afford to borrow.

225 +/− ENTER
▲ SOLVE



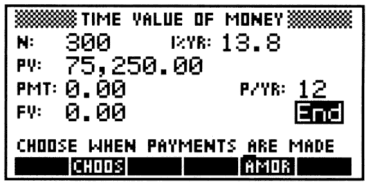
18

Example: **A Mortgage with a Balloon Payment.** Russ T. Pipes has taken out a 25-year, $75,250 house mortgage at 13.8% annual interest. He expects to sell the house in 4 years, repaying the loan in a balloon payment. Find the size of the balloon payment—the value of the mortgage after 4 years of payments.



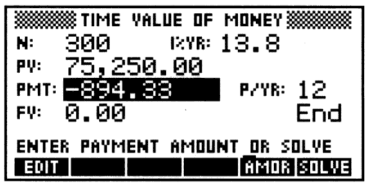
Step 1: Open the Finance Solver (if necessary) and enter values for the known TVM variables.

(→) SOLVE (▲) OK (if necessary) (→) (◀) 300 ENTER 13.8 ENTER 75250 ENTER (▼) 0 ENTER



Step 2: Find the monthly payment for the 25-year mortgage.

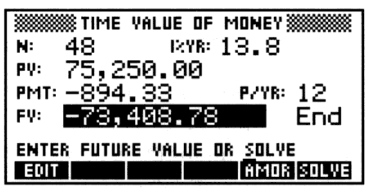
(◀) (▲) SOLVE



18

Step 3: Calculate the balloon payment needed after 4 years of payments.

(▲) (▲) 48 ENTER (▲) SOLVE



Calculating Amortizations

Amortization calculations, which also use the TVM variables, determine the amounts applied toward principal and interest in a payment or series of payments.

To calculate amortization:

1. Change the display mode to the accuracy you want, such as 2 Fix mode.
2. Start the Finance Solver.
3. Check and set these TVM conditions:
   - Number of payments per year.
   - Payments at beginning or end of periods.

4. Store values for four TVM variables: $I\%YR$, PV, PMT, and FV. These variables define the payment schedule. (You can calculate these using the TVM menu.)
5. Press AMOR and enter the number of payments to amortize in this batch.
6. Press AMOR to amortize the batch of payments. You will see the amount applied to interest, to principal, and the remaining balance after this set of payments have been amortized.

To continue amortizing the loan:

1. Press B→PV to store the new balance after the previous amortization as PV.
2. Enter the number of payments to amortize in the new group.
3. Press AMOR .
4. Repeat steps 1 through 3 as often as needed.

To amortize a series of future payments starting at payment p:

1. Calculate the balance of the loan at payment $p-1$.
2. Store the new balance in PV using B→PV .
3. Amortize the series of payments starting at the new PV.

The amortization operation reads the values from the TVM variables, rounds the numbers it gets from PV and PMT to the current display mode, then calculates the amortization rounded to the same setting. The original variables aren't changed, except for PV, which is updated by B→PV after each amortization.

19

Differential Equations

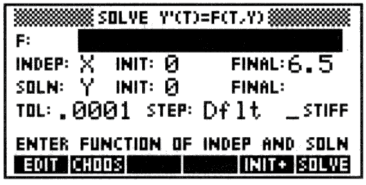
The HP 48 can find a solution, $y(t)$, to a differential equation expressed as $y'(t) = f(t,y)$, where the initial value of the solution is given as $y(t_0) = y_0$.

Solving Differential Equations

The differential equation solver is part of the SOLVE application.

To use the SOLVE differential equation solver:

1. Press ↱ SOLVE
2. Select `Solve diff eq...`.



This screen contains the following fields and menu keys:

`F:`	Contains the right-hand side of the differential equation you want to solve.
`INDEP:`	Specifies the independent variable. (Defaults to X.)
`INIT:`	Contains the independent variable's initial value (t_0). The initial value of the independent variable must correspond to the initial value of the solution variable: $y(t_0) = y_0$.

FINAL:	Contains the independent variable's final value, t_{FINAL}. You are trying to solve $y(t_{FINAL}) = (some\ unknown)$.
SOLN:	Specifies the solution variable. (Defaults to Y.)
INIT:	Contains the solution variable's initial value ($y_0 = y(t_0)$).
FINAL:	Contains the solution variable's final value. This is what you are trying to solve. You cannot enter a value in this field.
TOL:	Contains the acceptable level of absolute error. In a physical model, select the tolerance to suit the accuracy of the data. (Defaults to 0.0001.)
STEP:	Contains the initial step size used to compute the solution. The calculator uses the Runge-Kutta-Fehlberg method to calculate y_{FINAL}. This method computes its solution by automatically stepping from point to point, maintaining accuracy at each point.
_STIFF:	Selects the stiff solver.
EDIT	Lets you edit a field.
CHOOS	Lets you select a variable.
INIT+	Replaces the initial values with current final values. Use this to compute the solution at another point using the current solution as a starting point.
SOLVE	Solve the differential equation.

Solving a Standard Initial-Value Problem

Standard initial-value problems are differential equations that don't require the STIFF solver. To determine how stiff a differential equation is, try graphing it before solving it. Very slow graphing may indicate that the equation is stiff and the stiff solver should be used.

To solve a standard initial-value problem:

1. Enter an equation or press CHOOS to select an equation.
2. Specify the independent variable.
3. Enter the initial value for the independent variable.
4. Enter the final value for the independent variable.
5. Specify the solution variable.

6. Enter the initial value for the solution variable.
7. Enter an acceptable error tolerance.
8. (Optional:) Enter a step size. Normally, the solver computes an appropriate step size.
9. Press SOLVE.

Example: Solve this equation for $y(1)$ given that $y(0) = 2$:

$$y' = t + y$$

(→) SOLVE (▼) OK
(α) T (+) (α) Y ENTER
(α) T ENTER 0 ENTER 1
ENTER (▶) 2 ENTER SOLVE
EDIT

How accurate is the answer? The general solution to the differential equation

$$y' = t + y$$

is

$$y = ce^t - t - 1$$

Where c is an arbitrary constant. The given initial conditions were $2 = ce^0 - 0 - 1$. Solving for c and substituting back into the general solution, the solution equation is

$$y = 3e^t - t - 1$$

Solving for $y(1)$, returns $3e - 1 - 1 = 6.15484548538$. Comparing the results you can see there is an error of approximately 0.000068 which is well within the specified error tolerance of 0.0001.

Solving a Stiff Initial-Value Problem

Some differential equations may seem to take forever to solve. If this happens, the equations may be stiff. Use the stiff function to solve the equation.

To use the stiff function:

1. Press (→) SOLVE
2. Select Solve diff eq....

3. Highlight _STIFF and press ✓CHK

This screen contains the following additional fields:

∂F∂Y: The partial derivative with respect to y of the expression in F:.

∂F∂T: The partial derivative with respect to t of the expression in F:.

19

To solve a stiff initial-value problem:

1. Highlight STIFF and press ✓CHK.
2. Enter an equation or press CHOOS to select an equation already stored in memory.
3. Enter the partial derivatives of the equation with respect to y and t (or press CHOOS to select them if they are stored in memory).
4. Specify the independent variable.
5. Enter the initial value for the independent variable.
6. Enter the final value for the independent variable.
7. Specify the solution variable.
8. Enter the initial value for the solution variable.
9. Enter an acceptable error tolerance.
10. (Optional:) Enter a step size. It is usually best to accept the computed default step size.
11. Press SOLVE.

Example: Solve this equation for $y(1)$ given that $y(0) = 1$:

$$y' = -1000 * (y - sin(t)) + cos(t)$$

This example assumes that the calculator is set to radians.

→ SOLVE ▼ OK
◀ ✓CHK ▶
1000 +/- × ↶ ( ) α Y −
SIN α T ▶ ▶ + COS α
T ENTER 1000 +/- ENTER
1000 × COS α T ▶ − SIN
α T ENTER α T ENTER 0
ENTER 1 ENTER ▶ 1
ENTER SOLVE EDIT

The problem takes about a minute to solve. (If you had used the standard method, it would have taken over five minutes.)

How accurate is the answer? With the given initial conditions the solution equation is:

$$y = e^{-1000t} + sin(t)$$

Solving for $y(1)$ gives $e^{-1000} + \sin(1) = 0.841470984808$. Comparing the results, you can see there is an error of approximately 0.000098, which is within the specified error tolerance of 0.0001.

Solving a Vector-Value Differential Equation

You can use vector-valued equations to solve second-order (or higher) differential equations given two or more initial values.

Another way to write the second-order equation

$$y'' = a_1(t)y' + a_0(t)y + g(t)$$

is

$$\begin{bmatrix} y \\ y' \end{bmatrix}' = \begin{bmatrix} 0 & 1 \\ a_0(t) & a_1(t) \end{bmatrix} \begin{bmatrix} y \\ y' \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} g(t)$$

You can then substitute w for $\begin{bmatrix} y \\ y' \end{bmatrix}$, fw for $\begin{bmatrix} 0 & 1 \\ a_0(t) & a_1(t) \end{bmatrix}$, and c for $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ gives

$$w' = fw * w + c * g(t)$$

which is a first-order differential equation.

Example: Solve this equation for $w(1)$ given that $y(0) = 0$ and $y'(0) = 0$ ($w(0) = [\ 0\ 0\ ]$):

$$y'' = .5y' + .5y + .5t + 1$$

Step 1: Convert the equation into a first-order equation:

$$\begin{bmatrix} y \\ y' \end{bmatrix}' = \begin{bmatrix} 0 & 1 \\ .5 & .5 \end{bmatrix} \begin{bmatrix} y \\ y' \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} (.5t + 1)$$

Step 2: Store the values in fw ($\begin{bmatrix} 0 & 1 \\ .5 & .5 \end{bmatrix}$) and c ($\begin{bmatrix} 0 \\ 1 \end{bmatrix}$):

◀ [ ] ◀ [ ] 0 SPC 1
▶ .5 SPC .5 ENTER ' α α FW α STO
◀ [ ] 0 SPC 1 ENTER ' α C STO

19

Step 3: Enter the equation and initial values, set the solution variable to w, and solve for $w(1)$:

▶ SOLVE ▼ OK
α α FW × W + C α ×
◀ ( ) .5 × α T + 1
ENTER
α T ENTER 0 ENTER 1
ENTER α W ENTER ◀ [ ]
0 SPC 0 ENTER SOLVE

Press EDIT to view the result vector, $w(1)$, [.718262064225 1.71826206422]. The first value is $y(1)$, the second value is $y'(1)$.

How accurate is the answer? The original equations are

$$y = e^t - t - 1$$

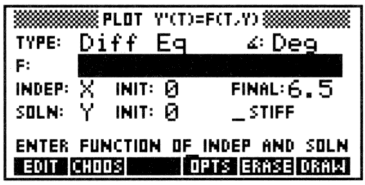
and

$$y' = t + y$$

Evaluating the equations at 1 and comparing the results you can see there is an error of approximately 0.0000198, which is well within the specified error tolerance of 0.0001.

Plotting Solutions to Differential Equations

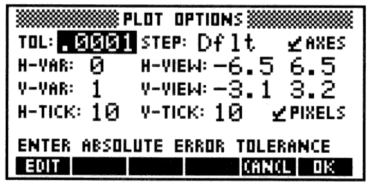
You can plot the solution to an initial-value by selecting the differential equation option of the PLOT dialog box.



This screen contains the following fields and menu keys:

TYPE:	The plot type (must be Diff Eq).
∠:	The angle mode.
F:	The right side of the equation you want to plot.
INDEP:	The independent variable. (Defaults to X.)
INIT:	The independent variable's initial value (t_0).
FINAL:	The independent variable's final value.
SOLN:	The solution variable. (Defaults to Y.)
INIT:	The solution variable's initial value.
STIFF:	Selects the stiff solver plot.
EDIT	Lets you edit a field.
CHOOS	Lets you select a variable.
OPTS	Lets you control plotting variables.
ERASE	Erases previous plots.
DRAW	Creates the plot.

Press OPTS and the following options appear:



This screen contains the following fields:

TOL:	The acceptable error tolerance.
STEP:	The step size.
_AXES	Whether or not axes are drawn.
H-VAR:	The variable plotted on the horizontal axis.
V-VAR:	The variable plotted on the vertical axis.
H-VIEW:	The portion of the horizontal axis you want to see.
V-VIEW:	The portion of the vertical axis you want to see.
H-TICK:	The horizontal tick-marks.
V-TICK:	The vertical tick-marks.
_PIXELS	Whether the tick-marks are spaced in user-units or pixels.

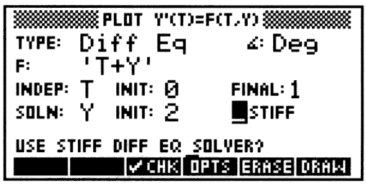
To plot a standard initial-value problem:

1. Enter an equation or press CHOOS to select an equation.
2. Specify the independent variable.
3. Enter the initial value for the independent variable.
4. Enter the final value for the independent variable.
5. Specify the solution variable.
6. Enter the initial value for the solution variable.
7. Set desired options and view parameters.
8. Press ERASE DRAW.

Example: Graph $y' = t + y$, $y(0) = 2$, over the interval [0, 1].

Step 1: Select PLOT Diff Eq mode, enter the equation, set the independent variable to T, and set the initial and final values:

↱ PLOT ▲ α D ▼
α T + α Y ENTER α T
ENTER 0 ENTER 1 ENTER
▶ 2 ENTER

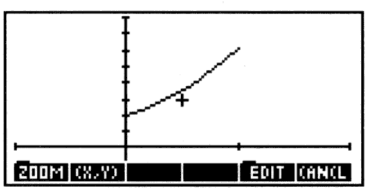


Step 2: Set the horizontal view to range from -1 to 2, set the vertical view to range from -2 to 8, and mark the axes every user-unit.

OPTS ▶ ▼ 1 +/- ENTER
2 ENTER ▶ 2 +/- ENTER 8
ENTER 1 ENTER 1 ENTER
✔CHK

Step 3: Draw the graph.

OK ERASE DRAW



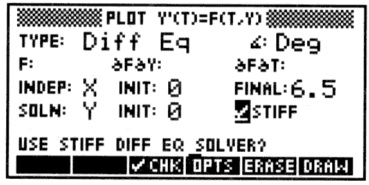
You can see that $y(1)$ is approximately 6. This coincides with the result from the first example in this chapter.

Plotting a Stiff Differential Equation

Use the stiff plot method when equations take a very long time to plot or when equations are plotted erratically. Plotting Stiff differential equations requires that you enter the partial derivatives of the equation.

To use the stiff plot function:

1. Press ▶ PLOT
2. Select Diff Eq.
3. Highlight _STIFF and press ✓CHK



This screen has the same elements that the standard plot has plus the following:

∂F∂Y: The partial derivative with respect to y of the expression in F:.

∂F∂T: The partial derivative with respect to t of the expression in F:.

To plot a stiff initial-value problem:

1. Select STIFF.
2. Enter an equation or press CHOOS to select an equation.
3. Enter the partial derivatives of the equation with respect to y and t (or press CHOOS to select them if they are stored in memory).
4. Specify the independent variable.
5. Enter the initial value for the independent variable.
6. Enter the final value for the independent variable.
7. Specify the solution variable.
8. Enter the initial value for the solution variable.
9. Set desired options and view parameters.
10. Press ERASE DRAW.

Example: Plot the following equation given that $y(0) = 1$:

$$y' = -1000 * (y - sin(t)) + cos(t)$$

Step 1: Select stiff, select radians, enter the function, the partial derivatives, and the initial values:

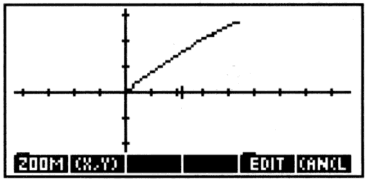
(↷) PLOT (▲) (α) D (▶) (α) R
(▲) ✓CHK (▶) (▼) 1000 (+/−) (×)
(↶) (()) (α) Y (−) SIN (α) T (▶)
(▶) (+) COS (α) T ENTER 1000
(+/−) ENTER 1000 (×) COS (α)
T (▶) (−) SIN (α) T ENTER (α)
T ENTER 0 ENTER 1 ENTER
(▶) 1 ENTER

Step 2: Set the horizontal view to range from -1 to 2, set the vertical view to range from -1 to 1, and mark the axes every 10 pixels:

OPTS (▶) (▼) 1 (+/−) ENTER
2 ENTER (▶) 1 (+/−) ENTER 1
ENTER 10 ENTER 10 ENTER
✓CHK

Step 3: Draw the graph.

OK ERASE DRAW



Plotting a Phase Plane for a Vector-Valued Solution

The HP 48 also lets you plot vector-valued equations and select which vector value is plotted on which axis. As described earlier, another way to write the second-order equation

$$y'' = a_1(t)y' + a_0(t)y + g(t)$$

is

$$w' = fw * w + c * g(t)$$

Where w is $\begin{bmatrix} y \\ y' \end{bmatrix}$, fw is $\begin{bmatrix} 0 & 1 \\ a_0(t) & a_1(t) \end{bmatrix}$, and c is $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$.

The initial condition $y'(t_0) = y_0$ and $y(t_0) = y_1$ can be written $w'(t_0) = [\ y_0\ y_1\ ]$. This is a vector-valued initial condition.

Example: Plot the following equation for $w(1)$ given that $y(0) = 0$ and $y'(0) = 0$ ($w(0) = [\ 0\ 0\ ]$):

19

$$y'' = .5y' + .5y + .5t + 1$$

where $y(0) = 0$ and $y'(0) = 0$ ($w(0) = [\ 0\ 0\ ]$).

Step 1: Convert the equation into a first-order equation.

$$\begin{bmatrix} y \\ y' \end{bmatrix}' = \begin{bmatrix} 0 & 1 \\ .5 & .5 \end{bmatrix} \begin{bmatrix} y \\ y' \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} (.5t + 1)$$

Step 2: Store the values in fw and c.

◀ [ ] ◀ [ ] 0 SPC 1
▶ ◀ [ ] .5 SPC .5 ENTER ' α α FW
α STO ◀ [ ] 0 SPC 1 ENTER ' α C
STO

Step 3: Enter the equation and initial values, set the solution variable to w.

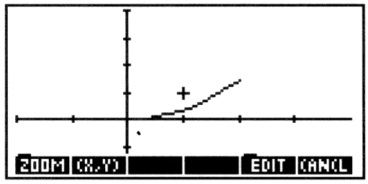
→ PLOT ▲ α D ▼
α α FW × W + C α ×
↰ () .5 × α T + 1
ENTER α T ENTER 0
ENTER 1 ENTER α W
ENTER ↰ [] 0 SPC 0
ENTER ✓CHK

Step 4: Set the horizontal view to range from -1 to 2, set the vertical view to range from -1 to 2, and mark the axes every 0.5 user-units.

OPTS ▶ ▼ 1 +/– ENTER
2 ENTER ▶ 1 +/– ENTER 2
ENTER .5 ENTER .5 ENTER
✓CHK

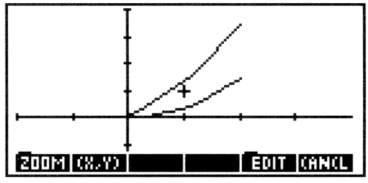
Step 5: Draw the graph.

OK ERASE DRAW



Step 6: Redraw the graph with the second vector value plotted on the vertical axis.

CANCL OPTS ▼ ▼ CHOOS
2 ENTER OK DRAW



20

Calculus and Symbolic Manipulation

Integration

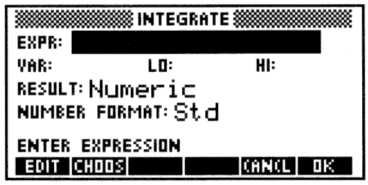
You can calculate *symbolic* integrals for expressions with known antiderivatives (indefinite integrals). You can also estimate the *numeric* value of those and other integrals.

Numeric Integration

Numeric integration lets you approximate a definite integral—even when symbolic integration can't generate a closed-form result. Numeric integration employs an iterative numeric procedure to obtain the approximation.

To find the value of an integral with numeric limits:

1. Press ↱ SYMBOLIC OK to open the INTEGRATE form.
2. Enter the expression to be integrated in the EXPR: field (without the integral sign).
3. Enter the variable of integration in the VAR: field.
4. Enter the limits of integration in the LO: and HI: fields. For numeric integration, the limits must be numbers or algebraic expressions that evaluate to numbers.
5. Make sure the RESULT field says Numeric (press +/−, if necessary.) You will see the NUMBER FORMAT field appear when the result type is Numeric. It is important because the number display format determines the *accuracy factor* for the numeric integration.



The Numeric INTEGRATE Screen.

6. Set the number display format to indicate the accuracy factor you desire for the computation. The `Std` format yields the highest accuracy factor (and consequently the longest calculation time) while `Fix 0` (or `Sci 0` or `Eng 0`) yields the lowest accuracy factor (and the shortest calculation time). See "The Accuracy Factor and the Uncertainty of Numerical Integration" on page 20-6.
7. Press `OK` to compute the integral.

20

Improper integrals are those integrals where one or both limits are infinity (∞). The HP 48 is a calculator with finite computational limits and thus must always use *finite* limits when computing numeric integrals. However, by using a transformation of variables, you can map an unbounded domain onto a bounded one.

One useful transformation, $y = \arctan x$, maps the entire real x-axis onto the bounded interval $\frac{-\pi}{2} \leq y \leq \frac{\pi}{2}$. This is the transformation:

$$\int_{-\infty}^{\infty} f(x)dx \longrightarrow \int_{arctan-\infty=\frac{-\pi}{2}}^{arctan\infty=\frac{\pi}{2}} f(\tan y) \cdot (1 + \tan^2 y)dy$$

To evaluate an improper integral:

1. Make sure that you are in Radian mode (press ◀ RAD), if necessary).
2. Press ▶ SYMBOLIC `OK` to open the `INTEGRATE` form.
3. Enter the integrand from the improper integral into the `EXPR:` field.
4. With the `EXPR:` field highlighted, press NXT `CALC` and enter the transformation expression onto the stack. For example, if the variable of integration for the improper integral is `X`, you would enter `TAN(Y)` to make the transformation ($x = \tan y$) shown above. Make a extra copy of transformation expression by pressing ENTER a second time.

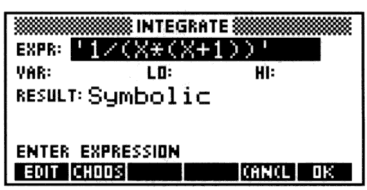
5. Enter the name of the original integration variable in the improper integral and press STO.
6. Enter the name of the new integration variable and press ▶ ∂ to compute the derivative of the transformation expression with respect to the new integration variable.
7. Press ✕ EVAL OK to compute the transformed integrand and return it to the EXPR: field.
8. Enter the new variable of integration into the VAR: field.
9. Enter the lower limit of integration into the LO: field. Use 'MAXR' wherever you need to include ∞.
10. Press CALC, transform the limit, and press OK to return. For the arctangent transformation above, $y = \arctan x$, you must find the arctangent of the limit to transform it. Note that this function is the inverse of that used to transform the expression above.
11. Repeat the previous two steps for the upper limit, beginning and ending in the HI: field.
12. Make sure that the result type is Numeric and set the desired number format.
13. Press OK to compute the numeric integral.

Example: Compute the following improper integral:

$$\int_1^{\infty} \frac{1}{x(x+1)}\,dx$$

Step 1: Open the INTEGRATE form and enter the integrand of the improper integral.

▶ SYMBOLIC OK
◀ EQUATION 1 ÷ α X ✕
◀ ( ) α X + 1 ENTER



Step 2: Use CALC to access the stack, enter and duplicate the transformation expression.

NXT CALC ' TAN α Y
ENTER ENTER

```
3:          '1/(X*(X+1))'
2:               'TAN(Y)'
1:               'TAN(Y)'
 STK                CANCL OK
```

Step 3: Transform the variable of integration, purge Y to ensure a symbolic result, and compute the derivative of the transformation expression.

' α X STO
' α Y ENTER ENTER
◀ PURG ▶ ∂

```
3:
2:          '1/(X*(X+1))'
1:          '1+TAN(Y)^2'
 STK                CANCL OK
```

Step 4: Multiply the derivative of the transformation expression by the integrand, evaluate to effect the transformation and return the result to the EXPR: field.

20

× EVAL OK

```
          INTEGRATE
EXPR: '1/(TAN(Y)*(TAN...
VAR:        LO:        HI:
RESULT: Symbolic

ENTER EXPRESSION
RESET CALC TYPES      CANCL OK
```

Step 5: Enter the new variable of integration, then compute and enter the transformed limits.

▼ α Y ENTER
CALC 1 ◀ ATAN OK
▶ CALC MTH NXT
CONS NXT MAXR
◀ ATAN ◀ CONT OK

```
          INTEGRATE
EXPR: '1/(TAN(Y)*(TAN...
VAR: Y      LO: .78... HI: 'AT...
RESULT: Symbolic

ENTER UPPER LIMIT
RESET CALC TYPES      CANCL OK
```

Step 6: Change the result type to Numeric, set the display to Std, and compute the integral.

▼ +/- ▼
+/- (until Std is displayed)
OK

```
1:          .693147180555
VECTR MATR LIST  HYP  REAL BASE
```

To evaluate a multiple integral numerically:

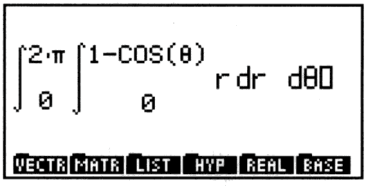
1. Press ◀ (EQUATION), key in the multiple integral (include all integral signs), and press (ENTER). All limits must evaluate to a number.
2. Set the number display format to reflect the desired accuracy.
3. Press ◀ (→NUM) to compute the result.

Example: Find the area of the region enclosed by the cardioid, $r = 1 - \cos\theta$. The region can be expressed by the double integral:

$$\int_0^{2\pi} \int_0^{1-\cos\theta} r \, dr \, d\theta$$

Step 1: Key in the double integral using the EquationWriter.

◀ (EQUATION) ▶ (∫) 0 ▶ 2
◀ (π) ▶ ▶ (∫) 0 ▶ 1 (−)
(COS) (α) ▶ F ▶ ▶ (α) ◀ R
▶ (α) ◀ R ▶ ▶ (α) ▶ F



Step 2: Enter the double integral onto the stack and set the display mode to Fix 3 and the angle mode to Radians.

(ENTER)
◀ (MODES) FMT 3 FIX
◀ (RAD) (if necessary)

Step 3: Evaluate the double integral. Then test your suspicion that π might be a factor in the result.

◀ (→NUM) (ENTER)
◀ (SYMBOLIC) (NXT) →Qπ

The Accuracy Factor and the Uncertainty of Numerical Integration

Numeric integration calculates the integral of a function $f(x)$ by computing a weighted average of the function's values at many values of x (sample points) within the interval of integration. The accuracy of the result depends on the number of sample points considered: generally, more sample points provide greater accuracy. There are two reasons why you might want to limit the accuracy of the integral:

- The length of time to calculate the integral increases as the number of sample points increases.
- There are inherent inaccuracies in each calculated value of $f(x)$:
  - Experimentally derived constants in $f(x)$ may be inaccurate. For example, if $f(x)$ contains experimentally derived constants that are accurate to only two decimal places, it is of little value to calculate the integral to the full (12-digit) precision of the calculator.
  - If $f(x)$ models a physical system, there may be inaccuracies in the model.
  - The calculator itself introduces round-off error into each computation of $f(x)$.

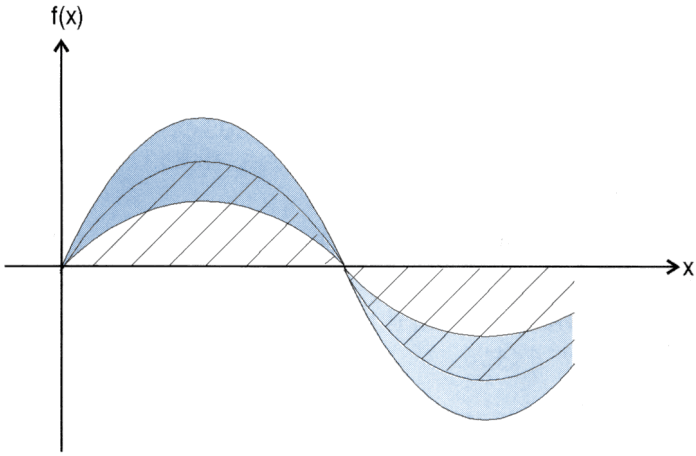
To indirectly limit the accuracy of the integral, you specify the *accuracy factor* of the *integrand* $f(x)$, defined as:

$$accuracy\ factor \leq \left| \frac{\text{true value of } f(x) - \text{computed value of } f(x)}{\text{computed value of } f(x)} \right|$$

The accuracy factor is your estimation in decimal form of the error in each computed value of $f(x)$. You specify the accuracy factor by setting the Display mode to n Fix. For example, if you set the display mode to 2 Fix, the accuracy factor is 0.01, or 1%. If you set the display mode to 5 Fix, the accuracy factor is 0.00001, or .001%.

The accuracy factor is related to the *uncertainty of integration* (a measurement of the accuracy of the *integral*) by:

$$uncertainty\ of\ integration \leq accuracy\ factor \times \int |f(x)|dx$$



The striped area is the value of the integral. The shaded area is the value of the uncertainty of integration. You can see that at any point x, the uncertainty of integration is proportional to $f(x)$.

The numeric integration algorithm uses an iterative method, doubling the number of sample points in each successive iteration. When the algorithm stops, the current value of the integral is returned to level 1, and the uncertainty of integration is stored in the variable *IERR*. The error in the final value will almost certainly be less than the uncertainty of integration.

To check the uncertainty of the numeric results:

- After computing the numeric results, press (VAR) IERR (you may need to press (NXT) one or more times before IERR is displayed in the menu.

Symbolic Integration

Symbolic integration means calculating an integral by finding a known antiderivative and then substituting specified limits of integration. The result is a symbolic expression.

The HP 48 can integrate the following patterns:

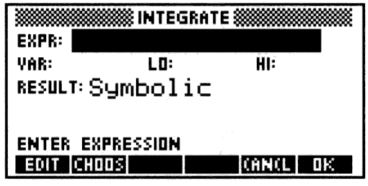
- All built-in functions whose antiderivatives contain only built-in functions (and whose arguments are linear). See the analytic

functions, labeled with "A" in appendix G. For example, 'SIN(X)' → 'COS(X)'.

- Sums, differences, negations, and other selected patterns of such functions. For example, 'SIN(X)-COS(X)' → '-SIN(X)-COS(X)', and '1/(COS(X)*SIN(X))' → 'LN(TAN(X))'.
- Derivatives of all built-in functions. For example, 'INV(1+X^2)' → 'ATAN(X)'.
- Polynomials whose base term is linear. For example, '(X-3)^3+6' → '6*X+(X-3)^4/4'.

To find the definite integral with symbolic limits:

1. Press ↱ SYMBOLIC OK to open the INTEGRATE form.



The Symbolic INTEGRATE Screen.

2. Enter the expression to be integrated in the EXPR: field (without the integral sign).
3. Enter the variable of integration in the VAR: field.
4. Enter the limits of integration in the LO: and HI: fields. If you want to use formal variables for limits, be sure that the variables do not exist in the current directory.
5. Make sure the RESULT field says Symbolic (press +/-, if necessary.)
6. Press OK to compute the integral. If the result is a *closed-form* expression—if there is no ∫ sign in the result—the symbolic integration was successful. If the result still contains ∫, you can try rearranging the expression and evaluating again. If rearranging fails to produce a closed-form result, you can estimate the answer with numeric integration or approximate the symbolic integral using a Taylor's polynomial (see "Taylor Polynomial Approximation," on page 20-12).

7. Press [EVAL] to simplify the closed-form result.

To find the indefinite integral of a function:

1. Press [→][SYMBOLIC] OK to open the INTEGRATE form.
2. Enter the expression to be integrated in the EXPR: field (without the integral sign).
3. Enter the variable of integration in the VAR: field. *Make sure that this variable is formal—that it doesn't exist in the current directory.*
4. Enter 0 as the lower limit and the variable of integration as the upper limit.
5. Make sure the RESULT field says Symbolic (press [+/–], if necessary.)
6. Press OK to compute the closed-form expression.
7. With the closed-form expression on level 1 of the stack, press [PRG] TYPE OBJ→ 3 [←][STACK] [NXT] DRPN to discard the lower limits.
8. Press [EVAL] to evaluate the result at the upper limit.

To symbolically integrate an expression that's not integrable:

1. Derive a Taylor's polynomial approximation to the integrand.
2. Find the symbolic integral of the Taylor's polynomial.

Differentiation

You can differentiate a symbolic expression either one step at a time, so you can see the substitutions—or completely in one step, so you can go right to the final result. If your expression contains only analytic functions (those labeled with "A" in appendix G), you get an explicit derivative.

To find the derivative of a function at a specified point:

1. Press [→][SYMBOLIC] [▼] OK to open the DIFFERENTIATE form.
2. Enter the function into the EXPR: field.
3. Enter the variable of differentiation into the VAR: field.
4. Press [+/–], if necessary, to change the result type to Numeric:

The Numeric DIFFERENTIATE Screen

5. Enter the value at which you want to compute the derivative into the VALUE: field.
6. Press OK .

To symbolically differentiate an expression completely in one step:

1. Press ↱ SYMBOLIC ▼ OK to open the DIFFERENTIATE form.

The Symbolic DIFFERENTIATE Screen

2. Enter the function into the EXPR: field.
3. Enter the variable of differentiation into the VAR: field.
4. Press OK .

To symbolically differentiate an expression step-by-step:

1. Press ↱ SYMBOLIC ▼ OK to open the DIFFERENTIATE form.
2. Enter the function into the EXPR: field.
3. Enter the variable of differentiation into the VAR: field.
4. Press STEP . The first step of the derivative will be computed and returned to the stack.
5. Press EVAL repeatedly to advance the evaluation of the derivative step-by-step.

Creating User-Defined Derivatives

If you execute ∂ for a function that has no built-in derivative, ∂ returns a *new* function whose name is *der* followed by the original function name. The new function has arguments that are the arguments of the original function, plus the arguments' derivatives. (You can differentiate further by creating a user-defined function to represent the new derivative function.)

If you execute ∂ for a formal user function (a name followed by arguments in parentheses, for which no user-defined function exists in user memory), ∂ returns a formal derivative whose name is *der* followed by the original user function name, plus the arguments and their derivatives.

Example: The HP 48 definition of % does not include a derivative. If you enter `'∂Z(%(X,Y))'` and press [EVAL], you get

```
'der%(X,Y,∂Z(X),∂Z(Y))'
```

Each argument of the % function results in two arguments for the der% function—`X` results in `X` and `∂Z(X)`, and `Y` results in `Y` and `∂Z(Y)`.

To define the derivative function for %, you can enter `'der%(x,y,dx,dy)=(x*dy+y*dx)/100'` and press [→][DEF]. `DER%` will appear in your VAR menu.

Now you can obtain the derivative of `'%(X,2*X)'` by entering the expression and the variable `'X'`, then pressing [→][∂] [←][ALGEBRA] `COLCT`. The result is `'.04*X'`.

Example: Enter the derivative of a formal user function, `'∂x(f(x1,x2,x3))'`. Then evaluate it by pressing [EVAL]. The result is this:

```
'derf(x1,x2,x3,∂x(x1),∂x(x2),∂x(x3))'
```

Implicit Differentiation

An implicit function of, say, x and y is a function in which one of the variables (y) is not directly expressed in terms of the other variable (f). This may be because it is either impossible, difficult, or not at all obvious how the expression may be solved for one variable in terms of the other. When this happens it is still possible to differentiate the expression using normal rules of differentiation (and the chain rule).

To perform implicit differentiation:

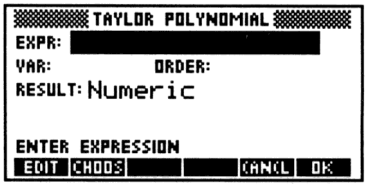
1. Enter the implicit function onto the stack. Instead of using two independent variables (such as x and y), make the second variable dependent on the first (such as x and $y(x)$). This links the two variables appropriately so that the differentiation treats the function as implicit, instead of eliminating one of the variables as a constant.
2. Enter the variable of differentiation onto the stack (using the ' delimiters).
3. Press (→)(∂) to compute the implicit derivative. You may see a user-defined derivative (such as derY(X,1)) included in the result. This is the way the HP 48 expresses the derivative of one variable with respect to the other (such as $\frac{dY}{dX}$).

Taylor Polynomial Approximation

For any mathematical function represented by a symbolic expression, you can compute a Taylor's polynomial approximation about $x = 0$, sometimes called a Maclaurin series. You can also specify the order of the polynomial.

To derive the Taylor's polynomial approximation about x = 0:

1. Press (→)[SYMBOLIC] (▼)(▼) OK to open the TAYLOR POLYNOMIAL form.



The TAYLOR POLYNOMIAL Screen

2. Enter the function you wish to approximate into the EXPR: field.
3. Enter the name of the variable to be used in the Taylor polynomial into the VAR: field.
4. Enter the order of the Taylor polynomial into the ORDER: field. Note that higher order polynomials require more time to compute.
5. Press OK to derive the Taylor polynomial approximation.

TAYLR always evaluates the function and its derivatives at zero. If you're interested in a function's behavior in a region away from zero, the Taylor's polynomial is more useful if you translate the point of evaluation to that region, as described below. Also, if the function has no derivative at zero, its Taylor's polynomial will be meaningless unless you translate the point of evaluation away from zero.

To derive the Taylor's polynomial approximation about x = a:

1. Press → SYMBOLIC ▼ ▼ OK to open the TAYLOR POLYNOMIAL form.
2. Enter the function you wish to approximate into the EXPR: field and press ENTER.
3. Press ▲ NXT CALC and enter '*Y*+*a*' on to the stack, where *a* is the point at which you are deriving the polynomial. Note that Y (or whatever name you wish to use instead) must not exist in the current directory path.
4. Press ' α X STO EVAL OK to store the translation, reevaluate the function using the translation, and return the result to the EXPR: field.
5. Enter the name of the new variable (Y) to be used in the Taylor's polynomial into the VAR: field.
6. Enter the order of the Taylor's polynomial into the ORDER: field. Note that higher order polynomials require more time to compute, but result in better approximations.

7. Press OK to derive the Taylor's polynomial approximation for the translated point.
8. Press VAR ' X ◀ PURG to purge the X variable.
9. Enter 'X-*a*' onto the stack and press ' α Y STO to store it in Y. (If you used a different dummy variable name, use it in place of Y here.)
10. Press EVAL to change the variable back to the original X. You may also wish to press ◀ SYMBOLIC COLCT to simplify the results.

Finding Symbolic Solutions to Equations

A common goal of algebraic manipulation of an expression or equation is to "solve for" a variable symbolically—that is, to express one variable in terms of the other variables and numbers in the expression or equation. You can solve symbolically using these commands:

20

- **ISOL.** Solves for a variable that appears only once in any type of expression or equation.
- **QUAD.** Solves for a variable that appears in a quadratic expression or equation.

Comparison of Commands for Symbolic Solutions

ISOL Command	QUAD Command
Variable appears only once. Variable can be any order. Variable can be the argument of a nonlinear function (such as SIN).	Variable can appear several times—no rearranging required. Variable must not be higher than second order for an exact solution.

Isolating a Single Variable

To solve for a variable that appears only once:

1. Press → SYMBOLIC ▼ ▼ ▼ OK to open the ISOLATE A VARIABLE form.
2. Enter the expression or equation to be solved into the EXPR: field. If the algebraic is an expression (it has no =), the expression is treated as an equation of the form '*expression*=0'.
3. Enter the variable you wish to solve for into the VAR: field. The variable to be isolated can be the argument of a function *only if the HP 48 has an inverse for that function*. Functions for which the HP 48 has inverses are called *analytic* functions in this manual. For example, you can isolate *X* in an algebraic containing TAN(X) or LN(X) because TAN and LN have inverses (ATAN and EXP). However, you cannot isolate *X* in an algebraic containing IP(X). The operations index in appendix H identifies the HP 48 analytic functions.
4. Optional: Select the result type you wish. (Numeric will attempt to compute a numerical solution and generate an error message if it fails.)
5. Optional: Check the PRINCIPAL field if you only want to see the principal solution (see "Getting General and Principal Solutions" on page 20-16).
6. Press OK to solve for the variable.

Solving Quadratic Equations

To solve for a variable in a quadratic:

1. Press → SYMBOLIC ▲ ▲ OK to open the SOLVE QUADRATIC form.
2. Enter the quadratic equation or expression you wish to solve in the EXPR: field. If the algebraic is an expression, the expression is treated as an equation of the form '*expression*=0'. If you supply an equation that is *not* first or second order in the variable to be solved for, it will be transformed into a second order polynomial *approximation* before being solved as a quadratic.
3. Enter the variable you wish to solve for in the VAR: field. If the algebraic contains other variables, they must *not* exist in the current directory if you want those variables to be included in the

solution as formal (symbolic) variables. If they exist in the current directory, they are evaluated when the quadratic is solved. (Purge a variable to make it formal.)

4. Optional: Select the result type you wish. (Numeric will attempt to compute a numerical solution and generate an error message if it fails.)
5. Optional: Check the PRINCIPAL field if you only want to see the principal solution (see below).
6. Press OK to solve the quadratic.

Getting General and Principal Solutions

HP 48 analytic functions always return one result—the *principal* solution. For example, $\sqrt{4}$ always returns +2, and ASIN(.5) always returns 30 degrees or 0.524 radians.

However, when you solve an algebraic for a variable, there may be more than one solution—and you may want to know what they are. So the ISOL and QUAD commands normally return a *general* solution. A general solution represents the multiple solutions by including special variables that can take on multiple values:

- *s1* represents an arbitrary + or − sign (+1 or −1). Additional arbitrary signs in the result are indicated by *s2*, *s3*, ... The "principal" value for arbitrary signs is +1.
- *n1* represents an arbitrary integer—0, ±1, ±2, ... Additional arbitrary integers are represented by *n2*, *n3*, ... The "principal" value for arbitrary integers is 0.

To specify general or principal solutions while viewing the stack:

1. Press ↷ MODES FLAG .
2. Press CHK until the desired option is displayed for Flag -1.

Example: Using ISOL to isolate x in the equation $y = \sin x^2$, gives the following results when general and principal solution options are chosen (in Radians mode):

Principal Solution: `'X=√ASIN(Y)'`.

General Solution: `'X=s1*√(ASIN(Y)*(-1)^n1+π*n1)'`

Showing Hidden Variables

Sometimes you may want to solve for a variable that's stored in another variable. To do this, you have to convert the algebraic so the hidden variable is visible.

Sometimes you may want to speed evaluation by converting an algebraic so all variables *except* certain ones are evaluated.

To evaluate only specified variables in an expression:

1. Enter the expression onto the stack.
2. Do one of the following:
   - Enter the name (with ' delimiters) of the variable in the expression that you *do* want to evaluate.
   - Enter a list containing the variable names in the expression that you do *not* want evaluated.
3. Press ◀ SYMBOLIC SHOW. The expression will be partially evaluated according to your wishes in step 2.

To evaluate an algebraic for temporary variable values:

1. Enter the algebraic onto the stack.
2. Enter a list that contains each variable name followed by the value to substitute. For example: { $name_1$ $expr_1$ ... $name_n$ $expr_n$ } where *expr* can be a number or a symbolic expression.
3. Press ◀ SYMBOLIC NXT | to perform the evaluation. If a variable named in the list currently exists (in the VAR menu), its contents are *not* changed by the | ("where") function.

Rearranging Symbolic Expressions

Manipulating Whole Expressions

You can sometimes simplify algebraics by expanding subexpressions or collecting like terms. For example, if a variable occurs more than once in an algebraic, you may be able to simplify it so the variable occurs only once—letting you use ISOL to solve for the variable.

A *subexpression* consists of a function and its arguments. The function that defines a subexpression is called the *top-level* function for that subexpression—it's the function that's executed last. For example, in the expression 'A+B*C/D', the top level function for the subexpression 'B*C' is *, the top-level function for 'B*C/D' is /, and the top level function for 'A+B*C/D' is +.

To collect like terms in an algebraic:

- Enter the expression onto the stack and press ↩ SYMBOLIC COLCT. COLCT simplifies an algebraic by doing the following:
  - Evaluates numerical subexpressions. For example, '1+2+LOG(10)' COLCT returns 4.
  - Collects numerical terms. For example, '1+X+2' COLCT returns '3+X'.
  - Orders factors (arguments of *) and combines like factors. For example, 'X^Z*Y*X^T*Y' COLCT returns 'X^(T+Z)*Y^2'.
  - Orders summands (arguments of + or −) and combines like terms differing only in a coefficient. For example, 'X+X+Y+3*X' COLCT returns '5*X+Y'.

  COLCT operates separately on the two sides of an equation, so like terms on the opposite sides of the equation are not combined.

To expand products and powers in an algebraic:

- Enter the expression onto the stack and press ↩ SYMBOLIC EXPA. EXPAN rewrites an algebraic by doing the following:
  - Distributes multiplication and division over addition. For example, 'A*(B+C)' EXPA returns 'A*B+A*C'.
  - Expands powers over sums. For example, 'A^(B+C)' EXPA returns 'A^B*A^C'.
  - Expands positive power integers. For example, 'X^5' EXPA returns 'X*X^4', and '(X+Y)^2' EXPA returns 'X^2+2*X*Y+Y^2'.

  EXPAN doesn't carry out all possible expansions of an algebraic in a single execution. Instead, EXPAN works down through the subexpression hierarchy, stopping in each branch of the hierarchy when it finds a subexpression that it can expand. It first examines the top-level subexpression (the top level subexpression is the algebraic itself). If it's suitable for expansion, EXPAN expands it and stops—otherwise, EXPAN examines all of the second-level

subexpressions. This process continues until an expansion occurs at some level—no lower levels are checked.

Manipulating Subexpressions

You can rearrange an algebraic in specific step-by-step stages, letting you get the result in the form you want. The Rules transformations are algebraic-rearrangement operations that are narrower in their scope than EXPAN and COLCT. The Rules transformations let you direct the path of an algebraic rearrangement.

To algebraically rearrange a particular subexpression:

1. Put the algebraic in the EquationWriter application:
   - To enter a new algebraic, press ◀ EQUATION and key it in.
   - To use an algebraic in level 1, press ▼.
   - To use an algebraic stored in a variable, press the VAR menu key associated with the variable and press ▼.
2. Get the Selection environment:
   - From entry mode, press ◀.
   - From scrolling mode, press ◀ PICTURE ◀.
3. Press ▲ ▼ ◀ ▶ to move the selection cursor to the *top-level function* for the subexpression you want to rearrange. (See below.)
4. Optional: Press EXPR at any time to highlight the current subexpression in its entirety (the highlight turns on or off).
5. Press RULES to get the RULES menu. (You can press − to return to the Selection menu.)
6. Press the menu key for the transformation you want (or just move the cursor to *not* do a transformation). Press ▶ before any transformation key to execute the transformation repeatedly until no further change occurs.
7. Repeat step 6 for each transformation you want. (If you move the cursor, you have to go back to step 3.)
8. Press ENTER to save the transformed algebraic (or press CANCEL to not save it).

In this section, the definition of *subexpression* in the previous section is expanded to include individual objects. For example, you can specify a name as the subexpression.

After you activate the Selection environment, you move the selection cursor—it specifies both an object in the algebraic and its corresponding subexpression.

Operations in the Selection Environment

Key	Description
RULES	Selects a menu of relevant rearrangement transformations for the specified subexpression.
EDIT	Returns the specified subexpression to the command line for editing.
EXPR	Highlights the specified subexpression.
SUB	Returns the specified subexpression to level 1 of the stack.
REPL	Replaces the specified subexpression with the algebraic in level 1 of the stack. (See "Replacing a Subexpression with an Algebraic Object" on page 7-12.)
EXIT	Exits the Selection environment, restoring the entry mode cursor at the end of the equation.
▲ ▼ ◀ ▶	Moves the selection cursor to the next object in the indicated direction. When prefixed with ↱, moves the selection cursor to the farthest object in the indicated direction.
+/-	Highlights the specified subexpression (just like EXPR), but is also active when the RULES menu is displayed.

The RULES menu may include transformations that aren't applicable to the specified subexpression—such menu keys produce a beep. After you execute a transformation, the selection cursor highlights the new top level object. The RULES menu is removed whenever you press any of the following keys: ◀ ▶ ▲ ▼, ← (to return to the Selection menu), ENTER, or CANCEL.

The tables on the next several pages describe the Rules transformations and show examples. However, the tables do *not* include all patterns for which transformations are applicable.

Note

The following tables include examples of transformations in the form

before → *after*

The before and after algebraics are shown in their *command-line form*—even though you execute Rules transformations in the *EquationWriter environment*. If you try an example, press ENTER to see the new expression in command-line form.

The RULES Menu—Universal Transformations

Key	Description
DNEG	Double-negate. A → --A
DINV	Double-invert. A → INV(INV(A))
*1	Multiply by 1. A → A*1 A+B/1 → A+B
^1	Raise to the power 1. A → A^1
/1	Divide by 1. A → A/1 A+B*1 → A+B
+1-1	Add 1 and subtract 1. A → A+1-1
COLCT	Collect. Executes a limited form of the COLCT command in the SYMBOLIC menu. Works only on the subexpression defined by the specified object and leaves the coefficients of collected terms as sums or differences. (2+3)*X → 5*X 2*X+3*X → (2+3)*X

The RULES Menu—Moving Terms

Key	Description
←T	Move-term-left. Moves the nearest neighbor to the right of the specified function over the nearest neighbor to the left of the function. A+B+(C+D) → A+C+(B+D) A+B+(C+D) → A+B+(D+C) A+(B+C)*1+D → A+D+(B+C)*1 A*B=C*D → A*B/C=D
T→	Move-term-right. Moves the nearest neighbor to the left of the specified function over the nearest neighbor to the right of the function. A+B=(D+E) → A=-B+(D+E) A*B=(X+Y) → A=INV(B)*(X+Y)

20

←T and T→ are used to move a *term* over its "nearest neighbor" to the left or right. A term is an argument of + or − (a summand), an argument of * or / (a factor), or an argument of =. Also, these two operations ignore parentheses—you can make them respect parentheses by executing *1 to make the parenthetical subexpression a term.

The RULES Menu—Building and Moving Parentheses

Key	Description
(())	Parenthesize-neighbors. Parenthesizes the nearest neighbors of + or *. Has no effect if the specified function is the first (or only) function in the expression, because these parentheses are already present, but hidden. A+B+C+D → A+(B+C)+D
(←	Expand-subexpression-left. Expands the subexpression associated with the specified function to include the next term to the left. Note that a matched pair of parentheses may disappear. A+B+(C+D)+E → A+(B+C+D)+E
→)	Expand-subexpression-right. Expands the subexpression associated with the specified function to include the next term to the right. A+(B+C)+D+E → A+(B+C+D)+E

The RULES Menu—Commuting, Associating, and Distributing

Key	Description
←→	Commute. Commutes the arguments of the specified function. A+B → B+A INV(A)*B → B/A
←A	Associate-left. A+(B+C) → A+B+C A*(B/C) → A*B/C A^(B*C) → A^B^C
A→	Associate-right. (A+B)+C → A+(B+C) (A*B)/C → A*(B/C) (A^B)^C → A^(B*C)
→()	Distribute-prefix-function. -(A+B) → -A-B INV(A/B) → INV(A)*B IM(A*B) → RE(A)*IM(B)+IM(A)*RE(B)

The RULES Menu—Commuting, Associating, and Distributing (continued)

Key	Description
←D	Distribute-left. (A+B)*C → A*C+B*C (A/B)^C → A^C/B^C
D→	Distribute-right. A*(B+C) → A*B+A*C A^(B-C) → A^B/A^C LN(A*B) → LN(A)+LN(B)
←M	Merge-factors-left. Merges arguments of +, −, *, and /, where the arguments have a common factor or a common single-argument function EXP, ALOG, LN, or LOG. For common factors, the ← indicates that the left-hand factors are common. Also merges sums where only one argument is a product. (A*B)+(A*C) → A*(B+C) EXP(A)*EXP(B) → EXP(A+B) A+A*B → A*(1+B)
M→	Merge-factors-right. Merges arguments of +, −, *, and /, where the arguments have a common factor. The → indicates that the right-hand factors are common. Also merges sums where only one argument is a product. (A*C)+(B*C) → (A+B)*C A*B+1*B → (A+1)*B
-()	Double-negate and distribute. Equivalent to DNEG followed by →() on the resulting inner negation. A+B → -(-A-B) LOG(INV(A)) → -LOG(A)
1/()	Double-invert and distribute. Equivalent to DINV followed by →() on the resulting inner inversion. A*B → INV(INV(A)/B) EXP(A) → INV(EXP(-A))

The RULES Menu—Rearranging Exponentials

Key	Description
L*	Replace log-of-power with product-of-log. LOG(A^B) → LOG(A)*B
L( )	Replace product-of-log with log-of-power. LN(A)*B → LN(A^B)
E^	Replace power-product with power-of-power. ALOG(A*B) → ALOG(A)^B
E( )	Replace power-of-power with power-product. EXP(A)^B → EXP(A*B)
→TRG	Replace exponential with trigonometric functions. (This example assumes Radians mode.) EXP(A) → COS(A/i)+SIN(A/i)*i

The RULES Menu—Adding Fractions

Key	Description
AF	Add fractions. Combines terms over a common denominator. (If the denominator is already common between two fractions, use M→ .) A+(B/C) → (A*C+B)/C (A/B)-C → (A-B*C)/B

The RULES Menu—Expanding Trigonometric Functions

Key	Description
→DEF	Expand-trigonometric-definition. Replaces trigonometric, hyperbolic, inverse trigonometric, and inverse hyperbolic functions with their definitions in terms of EXP and LN. (These examples assume Radians mode.) COS(X) → (EXP(X*i)+EXP(-(X*i)))/2 ASINH(U) → -LN(√(1+U^2)-U)
TRG*	Expand as product-of-trigonometric-functions. Expands trigonometric functions of sums and differences. SIN(X+Y) → SIN(X)*COS(Y)+COS(X)*SIN(Y)

The RULES Menu—Automatic Multiple Execution

Key	Description
↷ D→	Multiple-distribute-right.
↷ ←D	Multiple-distribute-left. (A+B+C)*D → A*D+B*D+C*D
↷ A→	Multiple-associate-right.
↷ ←A	Multiple-associate-left. A+(B+(C+D)) → A+B+C+D
↷ M→	Multiple-merge-factors-right. A*B+C*B+D*B → (A+C+D)*B
↷ ←M	Multiple-merge-factors-left.
↷ T→	Multiple-move-term-right. A+B+C+D=E → B+C+D=E-A
↷ ←T	Multiple-move-term-left.
↷ →)	Multiple-expand-subexpression right. A+(B+C)+D+E → A+(B+C+D+E)
↷ (←	Multiple-expand-subexpression-left.

20

Example: Solve for the variable x in the equation

$$ax = bx + c$$

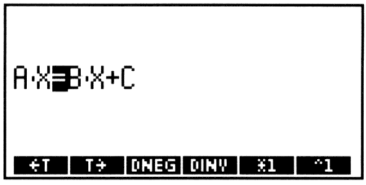
Do this by rearranging the equation so x appears only once, then using ISOL.

Step 1: Select the EquationWriter application and key in the expression.

↶ EQUATION
α α A ⊗ X ↶ =
B ⊗ X ⊕ C α

Step 2: Activate the Selection environment. Then move the selection cursor to the = sign and get the RULES menu.

◀
◀ (5 times)
RULES



Step 3: Move the term B·X to the left side of the = sign.

←T

Step 4: Merge the two terms on the left side of the = sign.

M→

Step 5: Now that x occurs only once in the equation, put the equation on the stack and isolate x.

ENTER ◀ SYMBOLIC ' α
X ISOL

Making User-Defined Transformations

If the built-in Rules transformations do not rearrange an algebraic in the form you want, you can make your own transformations. A "custom" transformation replaces occurrences of a pattern with a new pattern. The pattern can be specific or it can contain "wildcards" that match any subexpression and that you can reinsert in the replacement. You're informed whether or not a replacement was made.

You can also make conditional transformations—whether they occur depends on a condition you specify.

To replace a subexpression with a different subexpression:

1. Press (→)(SYMBOLIC)(▲) OK to display the MANIPULATE EXPRESSION form.
2. Press MATC to open the MATCH EXPRESSION form.
3. Enter or insert the expression you wish to modify in the EXPR: field. (You may insert an expression on level 1 of the stack by pressing (NXT) CALC OK (NXT).)
4. Enter the symbolic pattern that you wish to replace in the PATTERN: field. For generalized transformations, the search pattern can contain "wildcard" names that match *any* subexpressions. A wildcard name consists of an & character ((α)(←)(ENTER)), and a valid variable name (&A, &B, and &name are examples).
5. Enter the new, replacement symbolic expression in the REPLACEMENT: field. In general, if you used wildcards in the pattern expression, you should use wildcards in the replacement expression. You may not use a wildcard in the replacement expression that you did not use in the pattern expression.
6. Optional: Put a check-mark in the SUBEXPR FIRST field if you want the search-and-replace to begin at the lowest level of subexpressions and work "upwards" to the overall expression—a good option if the substitution will *simplify* the expression. Leave the field unchecked to begin the search with the overall expression and work "downwards" to the lower levels of subexpressions—a good option if the substitution will *expand* the expression. Note that a subexpression that already has been matched (and thus replaced) is not a candidate for any further matches, nor are any subexpressions whose arguments already have been matched.
7. Optional. Enter an expression representing a conditional test (such as '&A≤0'). The replacement will occur only if the test is *true*.
8. Press OK to execute the search-and-replace in the chosen "direction" and according to the conditional test, if any.

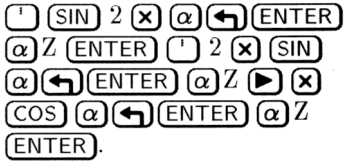
Example: An extension of the half-angle formula for sine is

$$\sin(2z) = 2\sin(z)\cos(z)$$

Create a transformation based on this formula and use it to transform the expression `'SIN(2*(X+1))'`.

Step 1: Open the MATCH EXPRESSION form and enter the target expression into the EXPR: field.

Step 2: Enter the pattern and replacement expressions, using a wildcard for z in the formulas.



Step 3: Leave SUBEXPR FIRST unchecked and do not include a conditional test. Execute the search-and-replace.

Symbolic Integration Patterns

This table lists the symbolic integration patterns used by the HP 48. These are the integrands that the HP 48 can integrate symbolically.

ϕ is a linear function of the variable of integration. The antiderivatives should be divided by the first-order coefficient in ϕ to reduce the expression to its simplest form. Also, patterns beginning with 1/ match INV: for example, $1/\phi$ is the same as INV(ϕ).

Symbolic Integration

Pattern	Antiderivative
ACOS(ϕ)	ϕ×ACOS(ϕ)−$\sqrt{}$(1−ϕ^2)
ALOG(ϕ)	.434294481904×ALOG(ϕ)
ASIN(ϕ)	ϕ×ASIN(ϕ)+$\sqrt{}$(1−ϕ^2)
ATAN(ϕ)	ϕ×ATAN(ϕ−LN(1+ϕ^2)/2
COS(ϕ)	SIN(ϕ)
1/(COS(ϕ)×SIN(ϕ))	LN(TAN(ϕ))
COSH(ϕ)	SINH(ϕ)
1/(COSH(ϕ)×SINH(ϕ))	LN(TANH(ϕ))
1/(COSH(ϕ)^2)	TANH(ϕ)
EXP(ϕ)	EXP(ϕ)
EXPM(ϕ)	EXP(ϕ)−ϕ
LN(ϕ)	ϕ×LN(ϕ)−ϕ
LOG(ϕ)	.434294481904×ϕ×LN(ϕ)−ϕ
SIGN(ϕ)	ABS(ϕ)
SIN(ϕ)	−COS(ϕ)
1/(SIN(ϕ)×COS(ϕ))	LN(TAN(ϕ))
1/(SIN(ϕ)×TAN(ϕ))	−INV(SIN(ϕ))
1/(SIN(ϕ)×TAN(ϕ))	−INV(SIN(ϕ))
1/(SIN(ϕ)^2)	−INV(TAN(ϕ))
SINH(ϕ)	COSH(ϕ)
1/(SINH(ϕ)×^{2}	−INV(SIN(ϕ))

Symbolic Integration (continued)

Pattern	Antiderivative
$1/(\text{SINH}(\phi)\times\text{COSH}(\phi))$	$\text{LN}(\text{TANH}(\phi))$
$1/(\text{SINH}(\phi)\times\text{TANH}(\phi))$	$-\text{INV}(\text{SINH}(\phi))$
$\text{SQ}(\phi)$	$\phi^3/3$
$\text{TAN}(\phi)^2$	$\text{TAN}(\phi)-\phi$
$\text{TAN}(\phi)$	$-\text{LN}(\text{COS}(\phi))$
$\text{TAN}(\phi)/\text{COS}(\phi)$	$\text{INV}(\text{COS}(\phi))$
$1/\text{TAN}(\phi)$	$\text{LN}(\text{SIN}(\phi))$
$1/\text{TAN}(\phi)\times\text{SIN}(\phi))$	$-\text{INV}(\text{SIN}(\phi))$
$\text{TANH}(\phi)$	$\text{LN}(\text{COSH}(\phi))$
$\text{TANH}(\phi)/\text{COSH}(\phi)$	$\text{INV}(\text{COSH}(\phi))$
$1/\text{TANH}(\phi)$	$\text{LN}(\text{SINH}(\phi))$
$1/\text{TANH}(\phi)\times\text{SINH}(\phi))$	$-\text{INV}(\text{SINH}(\phi))$
$\sqrt{\phi}$	$2\times\phi^{1.5}/3$
$1/\sqrt{\phi}$	$2\times\sqrt{\phi}$
$1/(2\times\sqrt{(\phi)})$	$2\times\sqrt{(\phi)}\times.5$
ϕ^z (z symbolic)	$\text{IFTE}(z==-1,\text{LN}(\phi),\phi^{(z+1)}/(z+1))$
ϕ^z (z real, $\neq 0,-1$)	$\phi^{(z+1)}/(z+1)$
ϕ^0	ϕ
ϕ^{-1}	$\text{LN}(\phi)$
$1/\phi$	$\text{LN}(\phi)$
$1/(1-\phi^2)$	$\text{ATANH}(\phi)$
$1/(1+\phi^2)$	$\text{ATAN}(\phi)$
$1/(\phi^2+1)$	$\text{ATAN}(\phi)$
$1/(\sqrt{(\phi-1)}\times\sqrt{(\phi+1)})$	$\text{ACOSH}(\phi)$
$1/\sqrt{(1-\phi^2)}$	$\text{ASIN}(\phi)$
$1/\sqrt{(1+\phi^2)}$	$\text{ASINH}(\phi)$
$1/\sqrt{(\phi^2+1)}$	$\text{ASINH}(\phi)$

21

Statistics and Data Analysis

Entering Statistical Data

Data can be accumulated on the HP 48 in two different kinds of objects: arrays and lists. In general, lists are best-suited for one-variable statistics, arrays for multi-variable statistics. Arrays may only contain numerical data; lists may contain any kind of data.

The built-in STAT application always uses arrays—specifically, it uses the data currently stored in the array variable named ΣDAT.

However, to apply programmed statistical functions other than those built into the STAT application, you may find lists are a more flexible object type than arrays.

To enter statistical data as a list:

1. Press ◀ {} to begin the list.
2. Type each datum followed by SPC. Press ENTER after you've typed the final datum.
3. Optional: Store the data list in a named variable in order to save it for later use. Be careful not to store the list in an HP reserved variable, such as ΣDAT.

To enter statistical data directly into ΣDAT:

1. Press ▶ STAT OK to open the SINGLE-VARIABLE STATISTICS form. (Actually, you may use any of the STAT application input forms.)
2. Optional: If there is already data in the ΣDAT: field either delete it (press DEL OK) or save the data by storing it in a variable (see next procedure) first and then deleting it.
3. With the highlight on the ΣDAT: field, press EDIT to begin the MatrixWriter (you may need to press NXT first).

4. Enter the data. Use a separate row for each individual *record* and a separate column for each variable within a record. For example, a data set containing the height, weight, and age for each of 100 people, would be entered as 100 rows of three columns each.
5. Press ENTER when you've finished. The data matrix is now temporarily stored in ΣDAT;. To confirm that you want it stored there, press OK ; to cancel the whole operation, press CANCL; to edit it further, press EDIT again.

To store the array in ΣDAT in a different variable:

1. Press → STAT OK to open the SINGLE-VARIABLE STATISTICS form. (Actually, you may use any of the STAT application input forms.) You should see the current statistics matrix partially displayed in the ΣDAT: field.
2. Press NXT CALC to bring up the stack.
3. Enter a name for the matrix onto level 1 (using the ' delimiters), and press STO.
4. Press OK to return to the SINGLE-VARIABLE STATISTICS.

To enter statistical data into a matrix:

1. Press → MATRIX to begin the MatrixWriter.
2. Enter the data. Use a separate row for each individual *record* and a separate column for each variable within a record. For example, a data set containing the height, weight, and age for each of 100 people, would be entered as 100 rows of three columns each.
3. Press ENTER when you've finished entering the data.
4. Enter a name for the data matrix onto level 1 of the stack, and press STO.

To designate a matrix the current statistics matrix:

1. Press → STAT OK to open the SINGLE-VARIABLE STATISTICS form. (Actually, you may use any of the STAT application input forms.)
2. Optional: If there is already data in the ΣDAT: field either delete it (press DEL OK) or save the data by storing it in a variable first and then deleting it.
3. With the highlight on the ΣDAT: field, press CHOOS and use the arrow keys to highlight the matrix you wish to make the current statistics matrix.

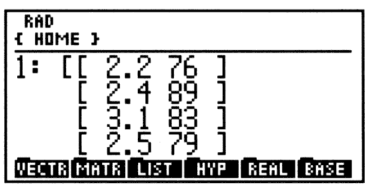
4. Press OK to temporarily store the matrix in ΣDAT. To confirm this action, press OK ; to cancel the action, press CANCL. You may also use the matrix in the current input form before pressing OK or CANCL.

Example: Enter the following data into a matrix and store it in the variable, *TEST*. Then make *TEST* the current statistical matrix. The data compares the undergraduate grade point averages of 12 corporate employees with their scores on a managerial trainee examination:

GPA	Exam Score
2.2	76
2.4	89
3.1	83
2.5	79
3.5	91
3.6	95
2.5	82
2.0	69
2.2	66
2.6	75
2.7	80
3.3	88

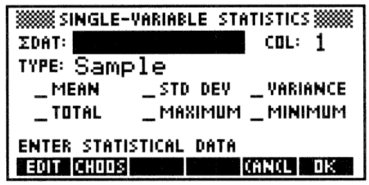
Step 1: Open the MatrixWriter and enter the data (12 rows, 2 columns)

(→) (MATRIX)
2.2 (ENTER) 76 (ENTER) (▼)
2.4 (ENTER) 89 (ENTER) 3.1
(ENTER) 83 (ENTER) 2.5 (ENTER)
79 (ENTER) 3.5 (ENTER) 91
(ENTER) 3.6 (ENTER) 95 (ENTER)
2.5 (ENTER) 82 (ENTER) 2.0
(ENTER) 69 (ENTER) 2.2 (ENTER)
66 (ENTER) 2.6 (ENTER) 75
(ENTER) 2.7 (ENTER) 80 (ENTER)
3.3 (ENTER) 88 (ENTER) (ENTER)



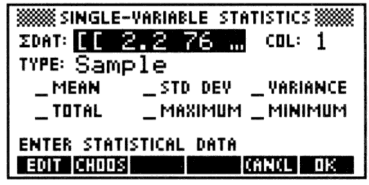
Step 2: Store the matrix as *TEST* and open the STAT application.

(hold) TEST (release)
STO
STAT OK



Step 3: Choose *TEST* as the current statistical matrix (ΣDAT).

CHOOS OK



21

Step 4: The *TEST* matrix overwrote any data stored in ΣDAT previously. Press OK to proceed or CANCL to remove *TEST* and restore the previous contents of ΣDAT.

Editing Statistical Data

To edit an element in the current statistical matrix:

1. Press STAT OK to open the SINGLE-VARIABLE STATISTICS form. (Actually, you may use any of the STAT application input forms.)
2. Press EDIT to bring the current statistical matrix into the MatrixWriter.
3. Use the arrow keys to highlight the element to be changed, type in its replacement, and press ENTER.
4. Press ENTER to save the change and return to the STAT application.

To transform a column of the current statistical matrix:

1. Press ▶ STAT OK to open the SINGLE-VARIABLE STATISTICS form. (Actually, you may use any of the STAT application input forms.)
2. Press NXT CALC to copy the matrix to the stack.
3. Enter the number of the column you wish to transform.
4. Press MTH MATR COL COL- to extract the designated column from the matrix.
5. Press PRG TYPE OBJ→ EVAL →LIST to convert the data to a list.
6. Perform the desired transformation on the data in the list. For example, to perform the transformation $x' = 3lnx$, press ▶ LN 3 ×.
7. Press PRG TYPE OBJ→ →ARR to convert the list to an array.
8. Enter the column number where the transformed variable is to be placed and press MTH MATR COL COL+.
9. Press ◀ CONT OK to return to the STAT application with the transformed matrix.

To transform a row, use ROW- and ROW+ in steps 4 and 8.

To add a new column to the current statistical matrix:

1. Press ▶ STAT OK to open the SINGLE-VARIABLE STATISTICS form. (Actually, you may use any of the STAT application input forms.)
2. Highlight the ΣDAT: field.
3. Press EDIT. This opens the MatrixWriter.
4. Move the highlight to the location of the new column.
5. Press NXT +COL. A column of zeros is inserted.
6. Press NXT GO↓. You can now replace the zeros with data.
7. Press ENTER to return the modified matrix to the STAT application.

To delete a column of the current statistical matrix:

1. Press ▶ STAT OK to open the SINGLE-VARIABLE STATISTICS form. (Actually, you may use any of the STAT application input forms.)
2. Highlight the ΣDAT: field.
3. Press EDIT. This opens the MatrixWriter.

4. Move the highlight to the column to be deleted.
5. Press NXT -COL. The column is deleted.
6. Press ENTER to return the modified matrix to the STAT application.

To mathematically transform the data in a list:

1. Put the data list onto the stack.
2. Perform the arithmetic necessary to transform each datum in the list. For example, to perform the transformation, $x' = 3 \ln x - 4$, press ↷ LN 3 × 4 −. (Remember to use MTH LIST ADD for list element addition instead of +.)

Calculating Single-Variable Statistics

21 If your statistical data measures a *sample of a population*, you calculate sample statistics. If, however, your data measures the *entire population*, you calculate population statistics.

The single-variable statistics built into the STAT application are:

MEAN	Returns the arithmetic mean of the data in the selected column.
STD DEV	Returns the standard deviation of the data in the selected column. It computes the version of the standard deviation indicated by the TYPE: field (sample or population).
VARIANCE	Returns the variance of the data in the selected column. It computes the version of the variance indicated by the TYPE: field (sample or population).
TOTAL	Returns the sum of the data in the selected column.
MAXIMUM	Returns the value of the largest datum in the selected column.
MINIMUM	Returns the value of the smallest datum in the selected column.

To compute a statistic for one variable:

1. Press ▶ STAT OK to open the SINGLE-VARIABLE STATISTICS form.
2. Enter or choose the data matrix containing the data for the variable.
3. Highlight the COL: field and enter the number of the column containing the data for the variable.
4. Select either Sample or Population in the TYPE: to indicate which version of the statistic you need to compute.
5. Put check-marks in one or more of the statistic check-fields.
6. Press OK. A labeled result for each statistic is placed on the stack.

To compute a statistic for all variables in the current data:

1. Press ◀ STAT 1VAR to display the single-variable statistics command menu.
2. Press the menu key corresponding to the statistic you wish to compute. For example, press MEAN to compute all of the means for each of the variables (columns) in the current statistics matrix. The result is a vector whose elements are the means of each column in the data matrix.

To compute a median for each variable in the current data:

1. Type TEACH and press ENTER to place a copy of the built-in EXAMPLES directory in the HOME directory.
2. Press VAR EXAM PRGS MEDIA. The result is a vector containing the medians for each variable (column) in the current statistics matrix.

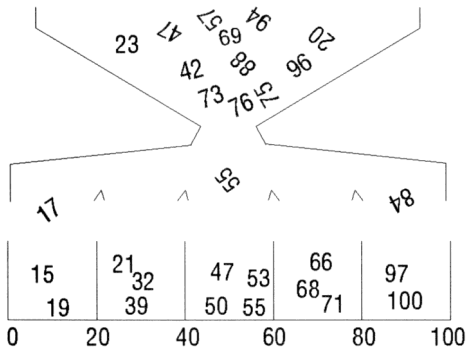
To draw a bar plot of the data for a variable:

1. Use the SINGLE-VARIABLE STATISTICS form to select the current statistics matrix and the column in that matrix containing the data you wish to plot.
2. Press NXT OK to accept the choices and return to the stack.
3. Press ◀ STAT PLOT BARPL to draw the bar plot using autoscaling (see 23-19 for more details.).

Generating Frequencies

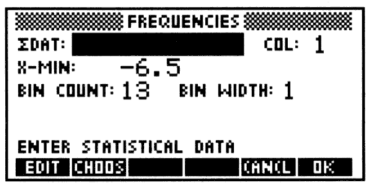
Many times, the most meaningful aspect of a data set is its distribution. Frequencies and frequency distributions are a common method for analyzing the distribution of a set of data.

Frequencies are created by dividing an interval (often that between the largest datum and the smallest) into an arbitrary number of equal subintervals, or *bins*, the number of which is suggested by the data and the precision with which one wants to study the distribution. The diagram below illustrates this.



To convert a data set into a set of frequencies:

1. Press ▶ STAT ▼ OK to open the FREQUENCIES form.



The FREQUENCIES Screen

2. Enter or choose the data matrix containing the data into the ΣDAT: field.

3. Enter the column number where the data to be converted is located.
4. Highlight the X-MIN: field and enter the minimum value that a datum can be and still be considered to lie within a bin. All lower values are considered *outliers*.
5. Enter the number of bins to use into the BIN COUNT: field.
6. Enter the width of each bin into the BIN WIDTH: field. All bins are of equal width.
7. Press OK to perform the conversion. On level 2 of the stack, you will see an array with integer elements, each one representing the number of data points that fell into each bin (in order from lowest to highest). On level 1 of the stack, you will see a two-element vector displaying the number of outliers. The first element represents the outliers *below* the lowest bin, and the second element represents the outliers *above* the highest bin.

To plot a histogram using the frequencies:

1. Convert the data set into frequencies as in the previous procedure.
2. Press ← to delete the outlier vector.
3. Press ↶ STAT DATA ↶ ΣDAT to store the frequency data in ΣDAT.
4. Press ↶ STAT PLOT BARPL to plot the frequencies.

Fitting a Model to a Set of Data

The HP 48 can use any of four general regression models in the attempt to quantify the relationship between the data in two columns from the current statistical matrix (ΣDAT):

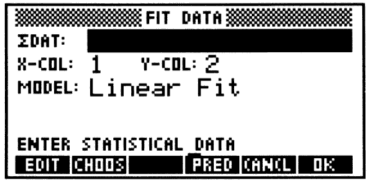
Linear Fit	$y = b + mx$
Logarithmic Fit	$y = b + m \ln x$
Exponential Fit	$y = be^{mx}$ or $\ln y = \ln b + mx$
Power Fit	$y = bx^m$ or $\ln y = \ln b + m \ln x$

For each of these general models the regression tool will find an intercept (b) and a slope (m) that corresponds to the least-squares fit

for that model. It also computes and returns the covariance (sample or population) and the correlation coefficient for the regression.

To perform a regression for two variables in the current data:

1. Press ↱ STAT ▼ ▼ OK to open the FIT DATA form.



The FIT DATA Screen

2. Enter or choose the data matrix containing the data you wish to fit.
3. Enter the independent variable in X-COL: and the dependent variable in Y-COL:.
4. Choose one of the four regression models (or Best Fit, which automatically selects the model with the correlation coefficient with the largest absolute value).
5. Press OK. You will see the computed regression model on level 3, the correlation coefficient on level 2, and the covariance on level 1.

To use the computed regression to predict the value of a variable:

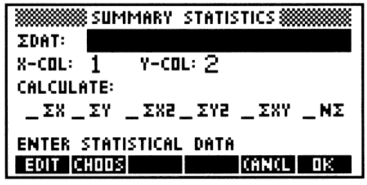
1. Press ↱ STAT ▼ ▼ OK to open the FIT DATA form.
2. Enter or choose the data matrix containing the data you wish to fit.
3. Enter the independent variable in X-COL: and the dependent variable in Y-COL:.
4. Choose one of the four regression models (or Best Fit, which automatically selects the model with the correlation coefficient with the largest absolute value.
5. Press PRED to display the PREDICT VALUES form.
6. Enter the assumed value in either the X: or the Y: field.
7. Move the highlight, if necessary, to the variable field whose value you wish to predict, and press PRED. The computed value is now displayed in the field. Press EDIT to view the complete number.

To plot a scatterplot of the data and the regression curve:

1. Perform the regression as described above.
2. Press ◀ STAT PLOT SCATR to plot the data using auto-scaling (see page 23-20 for details).
3. After the plot is drawn, press STATL to superimpose the most recent regression model on top of the data.

Calculating Summary Statistics

There are six summary statistics available that you may use to analyze statistical peculiarities within a data set or to compute statistics other than those built into the STAT application.



The SUMMARY STATISTICS Screen

The six summary statistics are:

ΣX	Sum of the data in the X-COL of ΣDAT.
ΣY	Sum of the data in the Y-COL of ΣDAT.
ΣX2	Sum of the squares of the data in the X-COL of ΣDAT.
ΣY2	Sum of the squares of the data in the Y-COL of ΣDAT.
ΣXY	Sum of the products of corresponding X-COL and Y-COL data. ΣDAT.
NΣ	Number of rows in ΣDAT.

To compute a summary statistic:

1. Press ▶ STAT ▲ OK to display the SUMMARY STATISTICS form.

2. Enter or choose the data matrix containing the data with which you are computing.
3. Enter the column numbers of the independent (X-COL) and dependent (Y-COL) variables.
4. Put a check-mark in each summary statistic that you wish to compute.
5. Press OK. Labeled results are placed on the stack.

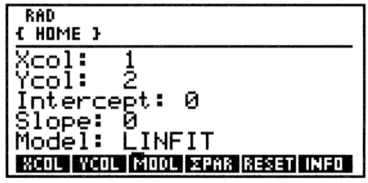
Using the ΣPAR Reserved Variable

The HP 48 uses a built-in statistics parameter variable named ΣPAR to store the statistics parameters. ΣPAR contains a list with the following objects:

{ independent-col dependent-col intercept slope model }

21 **To view the current settings in ΣPAR:**

- Do one of the following:
  - Press ◀ STAT ΣPAR INFO. The default settings are shown below.



  - Press ◀ STAT ΣPAR ▶ ΣPAR ON to show the list form. The default list is { 1 2 0 0 LINFIT }.

You normally control the parameters automatically using the STAT application. Because ΣPAR is a variable, you can have a different ΣPAR in every directory.

22

Plotting

Using the PLOT Application

The PLOT application lets you draw graphs of one or more functions in various formats, calculate roots and other parameters, plot statistical data in various formats, and embellish plots with additional elements.

The HP 48 can plot an equation, expression, or, for some plot types, a program:

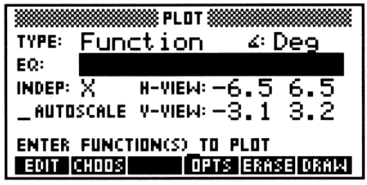
- **Equation.** An equation is an algebraic object containing = (for example, 'A+B=C').
- **Expression.** An expression is an algebraic object not containing = (for example, 'A+B').
- **Program.** A program to be plotted must return one real number (or a complex number for PARAMETRIC plots).

Throughout this chapter, unless otherwise stated, the term "equation" refers to all objects used to create plots: equations, expressions, programs, and lists of equations, expressions, and programs.

Plots are always drawn on the graphics object currently stored in the reserved variable *PICT*. To view the "picture" currently stored in *PICT*, you may press ◀ PICTURE.

To plot a single expression:

1. Press ▶ PLOT to open the PLOT application. You will the see the main PLOT screen showing the current plot type in TYPE: and the current equation in EQ: (if any). (The three statistical plot types (Scatter, Bar, and Histogram) use the ΣDAT: field instead of the EQ:.)



The Default PLOT Screen

2. If necessary, press ▲ and change the plot type, by doing any one of the following:
   - Press +/- repeatedly until the desired choice appears in the field.
   - Press CHOOS, highlight the desired choice in the choose list and press OK .
   - Press α followed by the first letter of the desired choice. You may need to repeat this step one or more times for plot types that share the same starting letter with others (e.g. Polar, Parametric, Pr-Surface, Ps-Contour).
3. Enter new values (or accept the current values) for the various plotting parameters available for the selected plot type. Chapter 23, "Plot Types," discusses each of the 15 available plot types and the plotting parameters and display options in greater detail. Most plot types have a second screen, accessed by pressing OPTS that contains display options for the plot type.
4. Once all values, parameters, and options have been set, do one of the following:
   - Press ERASE DRAW to "erase" *PICT* of any picture it contained previously and draw the plot according to the specifications you've set. You will see the plot being drawn and have access to it after the drawing is complete. Chapter 22, "Plotting" discusses the kinds of analysis and embellishments you can do to a plot after it has been drawn.
   - Press DRAW to draw the plot superimposed on the previous picture in *PICT*.
   - Press NXT OK to save the settings and options and return to the stack *without* drawing the plot.
   - Press NXT CANCL (or CANCL) to restore the settings and options in place before you made any changes and return to the stack *without* drawing the plot.

To plot a single equation:

1. Use the same general procedure as for plotting an expression, but make note of the following differences when the plot type is Function:
   - For equations whose left-hand side consists *only* of the name of the dependent variable (such as $y = 4x^2 - 7x + 29$), only the right-hand side expression is plotted.
   - For equations whose left-hand side consists of some expression other than the name of the dependent variable (such as $\sin x = \cos x$), the left-hand and right-hand expressions are *both* plotted.

To plot a group of expressions or equations:

1. Press (→)(PLOT) to open the PLOT application.
2. Do one of the following to enter a list of the expressions or equations into the EQ: field.
   - If all of the expressions or equations are stored in variables, press CHOOS, move the arrow keys to highlight each expression or equation in turn, and press ✓CHK to select it. Return a list of all checked equations to EQ: by pressing OK.
   - Type (←)({}) to begin a list and then type each expression or equation in as an element of the list. Press (ENTER) to enter the list into EQ:.
   - Combine the previous two approaches by selecting those expressions and equations that are stored in variables into a list using CHOOS, entering it into EQ:, and then editing the list using EDIT. You may add, insert or modify equations in the list.

   Note that each expression or equation in the group must be appropriate for the given plot type (see chapter 23 for details). Also, when equations (containing the = sign) are included in a list to be plotted using the Function plot type, only the *right-hand side expressions* of each equation is plotted. The left-hand side expressions are ignored. You may wish to rearrange some equations so that they are expressions (or equations whose left-hand side is zero).
3. Enter values for the plotting parameters and display options, if necessary.

4. For Function, Polar, and Parametric plot types, put a check-mark in the SIMULT field (on the PLOT OPTIONS screen) if you want the plots of all expressions and equations in the list to be drawn simultaneously. If this field is left unchecked, then the plots will be drawn sequentially (as they always are for other plot types).
5. Press ERASE DRAW (or just DRAW if you don't want to erase the previous plot or picture).

Cursor Coordinates: Standard and TRACE Modes

To display the current cursor coordinates:

- While viewing the plot, press (X,Y) to hide the menu and display the coordinate values (in user-units) of the current cursor position. Press NXT to redisplay the menu and cancel the coordinates display.

Whenever a plot is drawn—either originally or as a part of a zoom operation—the cursor begins in *standard* graphics mode. In standard mode, pressing ◀, ▶, ▲, or ▼ causes the cursor to move in the indicated direction without regard to the current plot. In standard mode, the horizontal and vertical "centers" of the pixel currently at the intersection of the crosshairs are reported as the current coordinates.

Some plot types also offer TRACE mode as an alternative mode of cursor movement. In TRACE mode, the cursor jumps from plotted point to plotted point along the function, instead of along pixel rows and columns. TRACE will appear in the menu if the current plot type uses TRACE mode.

To turn TRACE mode on and off:

- While viewing the plot, press TRACE to turn on TRACE mode. The ■ is displayed in the menu label whenever TRACE is on. Press TRAC■ to turn off TRACE mode. Note that executing a zoom or other function that redraws the graph will automatically turn off TRACE mode, as well.

For Function, Polar, and Parametric plot types, TRACE mode redefines the arrow keys. ◀ and ▶ move the cursor backwards and forwards along the plot of the current equation. If multiple functions are plotted, then ▲ and ▼ "jump" the cursor between the different functions. You may press (X,Y) while TRACE mode is on in order to display the coordinates of points on the graph.

Keyboard Operations in the PICTURE Environment

The PICTURE environment redefines the keyboard so that only a few keys are active. Those that are active are described in the table below:

Key	Description
Menu Keys	Behave as they always do—executing the operation indicated in the corresponding menu label.
NXT	Displays the next menu page.
◀ ▶ ▲ ▼	Move the cursor in the indicated direction. When TRACE mode is on, the cursor movement is constrained according to the content and type of plot (see chapter 23 for more details).
↰ PICTURE	Toggles the scrolling mode on and off. Scrolling mode hides the menu and the cursor and, if *PICT* is larger than the display allows you to scroll around using the cursor keys.
↰ CLEAR	Erase the picture. It is a shortcut for EDIT NXT ERASE.
↰ VIEW	Displays the current equation while the key is held down. If TRACE mode is on, it displays the function which is currently being traced.

Key	Description
DEL	Erases the rectangular region defined by the cursor and the mark. It is a shortcut for EDIT NXT DEL.
STO	Places a copy of the current picture on the stack as a graphics object. It is a shortcut for EDIT NXT NXT PICT→.
ENTER	Enters the current coordinates of the cursor onto the stack as a complex number. It is a shortcut for EDIT NXT NXT X,Y→.
CANCEL	Returns to the display from which the PICTURE environment was entered.
×	Sets the mark at the current location of the cursor. It is a shortcut for EDIT NXT MARK. The mark is used to delineate one endpoint of a range. After setting a mark, the cursor may then move and delineate the other endpoint.
−	Toggles the menu labels off and on, showing that portion of the plot that is hidden by the labels. It is a shortcut for EDIT NXT MENU.
+	Toggles the cursor coordinates display on and off. It is the same as (X,Y).
+/−	Switches the cursor appearance. The cursor is either always dark (default) or is dark against a light background and light against a dark background.

22

Using Zoom Operations

The zoom operations in the PICTURE environment let you look at a particular region of the plot in more detail (by zooming in) or look at more of the plot than is currently displayed (by zooming out).

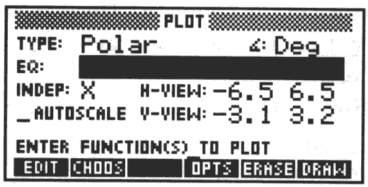
A *zoom*, or zoom operation, redraws the current plot by computing new display parameters. Zooms are a shortcut for the process of returning to the PLOT application, changing the values of the display parameters, and redrawing the plot. Note that not all plot types make use of zoom.

Setting Zoom Defaults

Several of the zoom operations make use of the current *zoom factor* and *recenter option* settings, which you can control.

To set the zoom factors:

1. Press ZOOM ZFACT to open the ZOOM FACTORS form.



The ZOOM FACTORS Screen

2. Enter the multiplicative factors for the horizontal and vertical axes that you want Zoom In and Zoom Out (and a few other zooms) to use. Note that Zoom Out multiplies the scale by the factor while Zoom In divides the scale by the factor.
3. Set the recenter option that you want zooms to use. Leaving the RECENTER AT CROSSHAIRS field unchecked causes the after-zoom display to be centered at the same point as the before-zoom display. Putting a check-mark in the field causes the after-zoom display to be centered around the point where the crosshairs cursor was located when the zoom was executed.
4. Press OK.

Selecting a Zoom

To perform a zoom:

1. While viewing the plot, move the cursor to the desired spot (if necessary for the zoom you wish to use), and press ZOOM.
2. Select the zoom you want (see below for details about each).

BOXZ — **Zoom-to-Box.** Allows you to draw a box around the region of interest and then zoom so that the boxed-in region fills the display. Move the cursor to one corner of the region before selecting this zoom.

ZIN	**Zoom-In.** Decreases both horizontal and vertical scales by the current zoom factors.
ZOUT	**Zoom-Out.** Increases both horizontal and vertical scales by the current zoom factors.
ZSQR	**Zoom-to-Square.** Changes the vertical scale so that it matches the horizontal scale.
ZDFLT	**Zoom-to-Default.** Redisplays the plot using the built-in default display ranges. Ignores the recenter option.
HZIN	**Horizontal Zoom-In.** Decreases the horizontal scale by the current zoom factor without affecting the vertical scale.
HZOUT	**Horizontal Zoom-Out.** Increases the horizontal scale by the current zoom factor without affecting the vertical scale.
VZIN	**Vertical Zoom-In.** Decreases the vertical scale by the current zoom factor without affecting the horizontal scale.
VZOUT	**Vertical Zoom-Out.** Increases the vertical scale by the current factor without affecting the horizontal scale.
CNTR	**Recenter-on-Cursor.** Redisplays the plot around the point where the crosshairs cursor was located when CNTR is pressed. Overrides default recenter option.
ZAUTO	**Autoscale Zoom.** Rescales the vertical axis using the built-in autoscaling computation without affecting the horizontal scale.
ZDECI	**Decimal Zoom.** Rescales the horizontal axis so that each pixel equals exactly 0.1 units. Does not affect the vertical axis.
ZINTG	**Integer Zoom.** Rescales the horizontal axis so that each pixel equals exactly 1 unit. Does not affect the vertical axis.
ZTRIG	**Trigonometric Zoom.** Rescales the horizontal axis so that 10 pixels equals $\frac{\pi}{2}$ units and rescales the vertical axis so that every 10 pixels equals one unit.
ZLAST	**Zoom-to-Last.** Redraws the display as it was prior to the most recent zoom. Ignores the recenter option.

Analyzing Functions

The PICTURE FCN menu lets you analyze the mathematical behavior of plotted functions. You use the graphics cursor to indicate the region or point of interest on the graph, then execute the desired calculation from the menu. You can calculate function values, slopes, areas under curves, roots, extrema and other critical points, and intersections of two curves. You can also plot derivatives of plotted functions.

To do function analysis, the current plot type must be `Function`. In addition, *EQ* must contain an equation, expression, or a list of equations or expressions—*it cannot contain a program.*

If *EQ* is a list of expressions, FCN menu operations use only the first (or first and second) element(s) of the list. The `NXEQ` operation is used to rotate the elements of the list so that different expressions are "first" and "second."

To analyze a plotted function:

1. While viewing the plot, press `FCN`.
2. Press ▲ ▼ ◀ ▶ to move the cursor to the point you want to analyze. (For certain operations, the cursor merely needs to be *near* the point.)
3. Press the menu key for the function analysis operation you want. See the table below.
4. Press `PICT` (in the second page of the FCN menu) to return to the main PICTURE menu.

When you perform a function analysis operation, the HP 48 does the following:

- Moves the cursor to the corresponding point on the function (if that point is in the display).
- Displays a message in the lower-left corner of the display showing the result.
- Returns the result to the stack as a tagged object.

The PICTURE FCN Menu

Key	Description
... FCN	(in the PICTURE menu):
ROOT	**Root.** Moves the cursor to a root (intersection of the function and the x-axis) and displays the value of the root. If the root is not in the display window, briefly displays the message OFF SCREEN before displaying the value of the root. If there are multiple roots, the root-finder usually finds the root closest to the current cursor location. For an equation, it searches for a root of the expression on the right side of the equation.
ISECT	**Intersection.** If only one function is plotted, moves the cursor to a root (same as ROOT). If two or more functions are plotted, moves the cursor to the closest intersection of two functions and displays the (x,y) coordinates. If the closest intersection is not in the display window, briefly displays the message OFF SCREEN before displaying the coordinates of the intersection.
SLOPE	**Slope.** Calculates and displays the slope of the function at the x-value of the cursor, and moves the cursor to the point on the function where the slope was calculated.
AREA	**Area.** Calculates and displays the area beneath the curve between two x-values defined by the mark and cursor. (Before you execute this operation, press ⊠ to mark one end of the x interval, then move the cursor to the other end.)
SHADE	**Shade.** If only one function is plotted, shades the region between the x-values defined by the mark and the cursor, between the function and the x-axis. If there are two functions plotted, shades the region between the two functions, and between the x-values defined by the mark and the cursor.

The PICTURE FCN Menu (continued)

Key	Description
EXTR	**Extremum.** Moves the cursor to an extremum (local minimum or maximum) or other critical point and displays the (x,y) coordinates. If the closest extremum or inflection point is not in the display window, briefly displays the message OFF SCREEN before displaying the value.
F(X)	**Function Value.** Displays the function value at the current x-value of the cursor, and moves the cursor to that point on the function curve.
F'	**Derivative Plot.** Plots the first derivative of the function and replots the original function. Also adds the symbolic expression for the first derivative to the contents of EQ. (If EQ is a list, F' adds the expression to the front of the list. If EQ is not a list, F' creates a list and inserts the expression to the front of the list.)
TANL	**Tangent Line.** Draws the line tangent to the current function at the x-value represented by the cursor. It returns the equation of the tangent line to the stack.
NXEQ	**Next Equation.** Rotates the list in EQ and displays the equation now at the beginning of the list. (The second equation is moved to the beginning of the list and the first equation is moved to the end.)
VIEW	**View Equation.** Displays currently selected equation from EQ or EQ list.
PICT	**Picture.** Displays the PICTURE menu.

Understanding the PLOT Reserved Variables

The PLOT application makes it easy to declare the plot and display ranges, the scale and resolution of the plots, and various other characteristics of the plot.

All of the information about a plot is automatically stored in a small set of *reserved variables* that you have direct access to if you wish. Because they are variables, stored in directories, you can have a different version of these reserved variables in every directory.

EQ

EQ contains the current equation or the name of the variable containing the current equation.

Specifically, the "equation" that *EQ* contains can be any of the following for the PLOT application:

- A single algebraic object or a name containing a single algebraic.
- A real number (or a complex number for the Parametric plot type) or a name containing a real number.
- A program that takes nothing from the stack and yields exactly one real (or complex for the Parametric plot type) result or a name containing such a program.
- A list containing any combination of the preceding three possibilities or the name of such a list. Although all elements are plotted, the first element of the list is always considered to be the "current" equation.

ΣDAT

ΣDAT contains the current statistical matrix or the name of the current statistical matrix. It is used instead of *EQ* by the three statistical plot types—Scatter, Bar, and Histogram.

ZPAR

ZPAR stores zoom information: the horizontal and vertical scale factors, a recenter flag, and (sometimes) a copy of *PPAR* to be used by the Zoom-to-Last (ZLAST) operation. *ZPAR* contains a list with the following objects:

{ *h-factor v-factor center-flag* { *previous PPAR (if any)* } }

PPAR

The HP 48 uses a built-in plot parameter variable named *PPAR* to store the plotting parameters. You normally control the plot parameters using commands in the PLOT and PLOT OPTIONS forms. *PPAR* contains a list with the following objects:

{ (x_{min}, y_{min}) (x_{max}, y_{max}) *indep res axes ptype depend* }

Contents of the PPAR List

Element	Description	Default
(x_{min}, y_{min})	A complex number representing the coordinates of the lower left corner of the display range.	(−6.5,−3.1)
(x_{max}, y_{max})	A complex number representing the coordinates of the upper right corner of the display range.	(6.5,3.2)
indep	Independent variable. The name of the variable, or a list containing the name and two real numbers (the horizontal plotting range).	*X*
res	Resolution. For equations, a real number or binary integer representing the interval between plotted points. For statistical data, the meaning varies.	0 (points plotted in every pixel column)
axes	A complex number representing the coordinates of the axes intersection, or a list containing the intersection and labels (strings) for both axes. The Diff Eq plot type uses this element in a special way (see page 23-11). This element can also contain information about the tick-spacing for each axis.	(0,0)
ptype	Command name specifying the plot type.	FUNCTION
depend	Dependent variable. The name of the variable, or a list containing the name and two real numbers (the vertical plotting range). The Diff Eq plot type uses this element in a special way (see chapter 23).	*Y*

To reset PPAR to its default:

- Press ◀ PLOT PPAR RESET. The RESET operation resets all parameters in *PPAR* to their default states—except the plot type—and erases *PICT* and restores it to its default size.

VPAR

VPAR contains the current settings that determine the View Volume, eyepoint, and plotting density for the six plot types for functions of two variables. See page 23-22 for a full explanation of how its parameters relate to the plot display.

VPAR is a list of real numbers:

$\{\ X_{\text{left}}\ X_{\text{right}}\ Y_{\text{near}}\ Y_{\text{far}}\ Z_{\text{low}}\ Z_{\text{high}}\ XX_{\text{left}}\ XX_{\text{right}}\ YY_{\text{left}}\ YY_{\text{right}}\ X_{\text{eyept}}\ Y_{\text{eyept}}\ Z_{\text{eyept}}\ N_{\text{X}}\ N_{\text{Y}}\ \}$

Contents of the VPAR List

Element	Description	Default
X_{left}	The smallest output (View Volume) value for the x-axis (width) to be plotted.	-1
X_{right}	The largest output (View Volume) value for the x-axis (width) to be plotted.	1
Y_{near}	The smallest output (View Volume) value for the y-axis (depth) to be plotted.	-1
Y_{far}	The largest output (View Volume) value for the y-axis (depth) to be plotted.	1
Z_{low}	The smallest output (View Volume) value for the z-axis (height) to be plotted.	-1
Z_{high}	The largest output (View Volume) value for the z-axis (height) to be plotted.	1

Contents of the VPAR List (continued)

Element	Description	Default
XX_{left}	The smallest value for the horizontal axis of the input plane.	-1
XX_{right}	The largest value for the horizontal axis of the input plane.	1
YY_{left}	The smallest value for the vertical axis of the input plane.	-1
YY_{right}	The largest value for the vertical axis of the input plane.	1
X_{eyept}	The x-axis coordinate for the eyepoint.	0
Y_{eyept}	The y-axis coordinate for the eyepoint. It must always be at least one less than the Y_{near} value.	-3
Z_{eyept}	The z-axis coordinate for the eyepoint.	0
N_{X}	The number of columns in the plotted lattice. Used instead of or in combination with the *res* element of PPAR.	10
N_{Y}	The number of rows in the plotted lattice. Used instead of or in combination with the *res* element of PPAR.	8

22

ΣPAR

ΣPAR is used in conjunction with ΣDAT by the statistics plot types. It contains either the current statistical parameter list or the name of the variable containing this list. See page 21-12 for an explanation of this reserved variable.

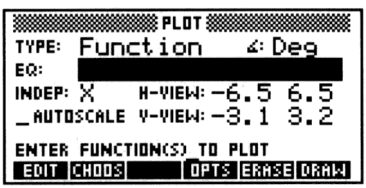
23

Plot Types

Function Plots

The Function plot type plots equations that return a unique $f(x)$ for each value of x. It is the default plot type and the only plot type that can use the PICTURE FCN analysis tools (see chapter 22).

The Default FUNCTION Plot Screens



The Function PLOT Form

∠:	Shows the current angle mode. Change it by pressing (+/−) one or more times or by using CHOOS.
EQ:	Enter the current expression, equation, or program to be plotted. It may contain a list of expressions, equations, or programs if you are plotting multiple functions. Names of variables containing expressions, equations, or programs (or lists of these) may be used in place of the objects themselves.
INDEP:	Enter the name of the independent variable.
H-VIEW:	Enter the horizontal display range in the two fields, low endpoint in the left and high endpoint in the right. To enter computed endpoints, use (NXT) CALC (see page 24-6).
V-VIEW:	Enter the vertical display range in the two fields, low endpoint in the left and high endpoint in the right.

To enter computed endpoints, use NXT CALC (see page 24-6).

AUTOSCALE — When checked the vertical display range is automatically scaled based on sampling 40 values equally spaced across the horizontal display range. When left unchecked the vertical display range is determined by the values entered in the two fields labeled V-VIEW.

OPTS — Opens the PLOT OPTIONS form.

ERASE — Erases the *PICT* screen (without displaying it).

DRAW — Stores all of the values in their appropriate places in the reserved variables, *EQ* and *PPAR*, and draws the plot accordingly, leaving you in the PICTURE environment when finished.

The Function PLOT OPTIONS Form

INDEP: — Enter the name of the independent variable, if necessary.

LO: — Enter the smallest value of the independent variable that you want *plotted*. Plotting range can be different than display range (see page 24-3). To use a computed endpoint, use NXT CALC (see page 24-6).

HI: — Enter the largest value of the independent variable that you want *plotted*. Plotting range can be different than display range (see page 24-3). To use a computed endpoint, use NXT CALC (see page 24-6).

AXES — When checked (default), the coordinate axes are drawn along with the plot. When unchecked, the axes are not drawn.

CONNECT — When checked (default), the plotted points are connected by short line segments. When unchecked, only the plotted points are displayed.

SIMULT — When checked, multiple functions are plotted simultaneously—a point is plotted for each function at a given sample value before moving to the next sample value. When unchecked (default), multiple functions are plotted sequentially—all points are plotted for the first function before the first point is plotted for the second function, and so forth.

STEP: This determines the *resolution* of the plot. It is the horizontal distance (in units or pixels—see next field) between two plotted points. Larger step-sizes provide speedier plots, but show less detail, while smaller step-sizes provide more detail, but take longer to draw. The default step-size for Function is 0.1 units.

PIXELS When checked, the step-size is interpreted to represent pixels. When unchecked (default) the step-size is interpreted to represent units.

H-TICK Enter the tick-spacing that you want displayed on the horizontal axis. It can be determined either in pixels or units depending on the state of its PIXELS field (see below). The default is one tick-mark every 10 pixels.

V-TICK Enter the tick-spacing that you want displayed on the vertical axis. It can be determined either in pixels or units depending on the state of its PIXELS field (see below). The default is one tick-mark every 10 pixels.

PIXELS When checked (default), the tick-spacing in H-TICK and V-TICK is interpreted to represent pixels. When unchecked, the tick-spacing is interpreted to represent units.

TRACE Mode

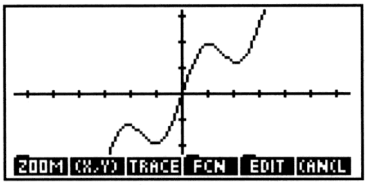
- ◀ and ▶ move the cursor along the plot of the current function.
- ▲ and ▼ jump the cursor between functions when multiple functions are plotted.

Special Notes

- Algebraic expressions in EQ: may contain any number of variables. However, all variables except the independent variable must evaluate to a real number in order for EQ: to be plotted. Otherwise, you will get the Undefined Name error message.

Example: Display the function example plot, XSIN: $x + \sin x$. If necessary, type in TEACH to install the EXAMPLES directory and then:

Press (VAR) EXAM PLOTS XSIN



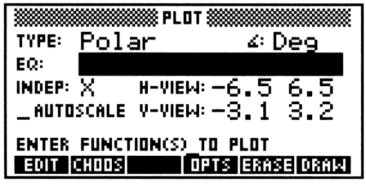
After plotting, press (CANCEL) (→)(PLOT) to review the PLOT forms that generated the plot. Experiment by changing values or settings and redrawing the plot.

Polar Plots

The Polar plot type plots functions described according to the polar coordinate system $f(\theta)$. The independent variable is the polar angle, θ.

23

The Default POLAR Plot Screens



The Polar PLOT Form

∠:	Shows the current angle mode. Change it by pressing (+/−) one or more times or by using CHOOS.
EQ:	Enter the expression, equation, or program to be plotted.
INDEP:	Enter the name of the independent variable. Note that the commonly used polar variable, θ, is entered by pressing (α)(→) F.
H-VIEW:	Enter the horizontal display range in the two fields, low endpoint in the left and high endpoint in the right.

V-VIEW: Enter the vertical display range in the two fields, low endpoint in the left and high endpoint in the right.

AUTOSCALE When checked the vertical display range is automatically scaled based on sampling 40 values equally spaced across the horizontal display range. Note that because the HP 48 computes an appropriate x- and y-axis display range based on the θ-range, the resulting x- and y-axis scales may differ from one another. When this field is left unchecked the vertical display range is determined by the values entered in the two fields labeled V-VIEW.

OPTS Opens the PLOT OPTIONS form.

ERASE Erases the *PICT* screen (without displaying it).

DRAW Stores all of the values in their appropriate places in the reserved variables, *EQ* and *PPAR*, and draws the plot accordingly, leaving you in the PICTURE environment when finished.

The Polar PLOT OPTIONS Form

INDEP: Enter the name of the independent variable.

LO: Enter the smallest value of the independent variable that you want *plotted*. For Polar plots, the plotting range is different than display range.

HI: Enter the largest value of the independent variable that you want *plotted*. Plotting range is always different than display range for polar plots because the independent variable is distinct from the horizontal-axis variable.

AXES See the Function plot type.

CONNECT See the Function plot type.

SIMULT See the Function plot type.

STEP: This determines the *resolution* of the plot. It is the interval between two plotted points. The default step-size for Polar is 2 degrees or $\pi/90$ radians.

PIXELS Leave unchecked for polar plots.

H-TICK See the Function plot type.

V-TICK See the Function plot type.

PIXELS See the Function plot type.

TRACE Mode

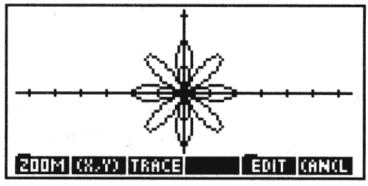
- ◀ and ▶ move the cursor along the plot of the current function. ◀ moves the cursor to the next lower value of the independent variable and ▶ moves the cursor to the next higher value of the independent variable. This may result in directional movement counter to the "direction" implied by the arrow keys. A polar plot can be traced through the range $\theta \geq 0$, so you will be able to press ▶ indefinitely in TRACE mode, even beyond the plotted interval.
- ▲ and ▼ jump the cursor between polar functions when multiple functions are plotted.

Special Notes

- Unless you specify otherwise, polar plots are drawn for a full circle of the independent variable θ (0 through 360 degrees, 2π radians, or 400 grads, according to the current angle mode).

Example: Display the polar example plot, ROSE: $r = 2\cos 4\theta$. If necessary, type in TEACH to install the EXAMPLES directory and then:

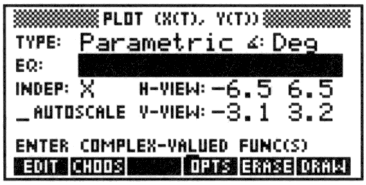
Press VAR EXAM PLOTS ROSE



After plotting, press CANCEL ↷ PLOT to review the PLOT forms that generated the plot. Experiment by changing values or settings and redrawing the plot.

Parametric Plots

The Default PARAMETRIC Plot Screens



The Parametric PLOT Form

∠: Shows the current angle mode. Change it by pressing (+/−) one or more times or by using CHOOS.

EQ: Enter the expression, equation, or program to be plotted. The Parametric plot type requires that a *complex number* result be returned when *EQ* is evaluated (see Special Notes below). *EQ* may contain a list of expressions, equations, or programs if you are plotting multiple functions. Names of variables containing expressions, equations, or programs (or lists of these) may be used in place of the objects themselves.

INDEP: Enter the name of the independent variable (commonly T).

H-VIEW: Enter the horizontal display range in the two fields, low endpoint in the left and high endpoint in the right.

V-VIEW: Enter the vertical display range in the two fields, low endpoint in the left and high endpoint in the right.

AUTOSCALE When checked the vertical display range is automatically scaled based on sampling 40 values equally spaced across the horizontal display range. When left unchecked the vertical display range is determined by the values entered in the two fields labeled V-VIEW.

OPTS Opens the PLOT OPTIONS form.

ERASE Erases the *PICT* screen (without displaying it).

DRAW Stores all of the values in their appropriate places in the reserved variables, *EQ* and *PPAR*, and draws

the plot accordingly, leaving you in the PICTURE environment when finished.

The Parametric PLOT OPTIONS Form

INDEP:	Enter the name of the independent variable.
LO:	Enter the smallest value of the independent variable that you want *plotted*. Plotting range for Parametric plots are usually different than display range (see page 24-3).
HI:	Enter the largest value of the independent variable that you want *plotted*. Plotting range can be different than display range (see page 24-3).
AXES	See the Function plot type.
CONNECT	See the Function plot type.
SIMULT	See the Function plot type.
STEP:	This determines the *resolution* of the plot. It is the horizontal distance (in units or pixels—see next field) between two plotted points. The default step-size for Parametric is an interval equal to $\frac{1}{130}$th of the difference between the LO and HIGH values in the plotting range (in units).
PIXELS	When checked, the step-size is interpreted to represent pixels. When unchecked (default) the step-size is interpreted to represent units.
H-TICK	See the Function plot type.
V-TICK	See the Function plot type.
PIXELS	See the Function plot type.

TRACE Mode

- ◀ and ▶ move the cursor along the plot of the current function. ◀ moves the cursor to the next lower value of the independent variable and ▶ moves the cursor to the next higher value of the independent variable. This may result in directional movement counter to the "direction" implied by the arrow keys. A parametric plot can be traced through an unlimited range of the independent variable so you will be able to press ▶ or ◀ indefinitely in TRACE mode, even beyond the plotted interval.
- ▲ and ▼ jump the cursor between functions when multiple functions are plotted.

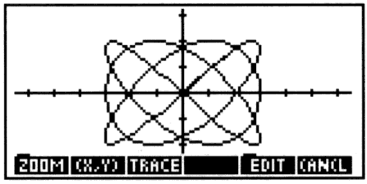
Special Notes

- Algebraic expressions should be entered in complex form, '(F,G)', where F and G are each expressions involving the independent variable.
- Programs should take nothing from the stack and return one complex number result.

Examples

Example 1: Display the parametric example plot, LISSA: $x(t) = 3\sin 3t, y(t) = 2\sin 4t$. If necessary, type in TEACH to install the EXAMPLES directory then:

VAR EXAM PLOTS LISSA



After plotting, press CANCEL ↷ PLOT to review the PLOT forms that generated the plot. Experiment by changing values or settings and redrawing the plot.

Example 2: Do the two particles described below parametrically actually collide between $t = 0$ and $t = 6.5$ or do their path traces merely cross?

Particle 1: $x(t) = \frac{16}{3} - \frac{8}{3}t$, $y(t) = 4t - 5$.
Particle 2: $x(t) = 2\sin\frac{\pi}{2}t$, $y(t) = -3\cos\frac{\pi}{2}t$.

Step 1: In the Parametric PLOT form, enter a list containing the two parametric expressions into EQ:

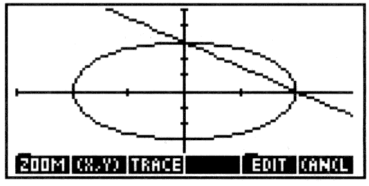
```
{ '(16/3-8/3*T,4*T-5)' '(2*SIN(π/2*T),-
3*COS(π/2*T))' }
```

Step 2: Set the independent variable, the display and plotting ranges, simultaneous plotting, and the tick-spacing as shown.

```
INDEP: T LO: 0 HI: 6.5
H-VIEW: -3  3
V-VIEW: -5  5 ✓SIMULT
H-TICK: 1 V-TICK: 1
_PIXELS
```

Step 3: Erase *PICT* and draw the plot. Watch to see if the two plots turn on the same pixel at the same time—a possible collision.

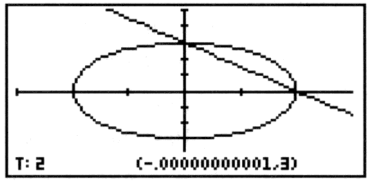
OK ERASE DRAW



23

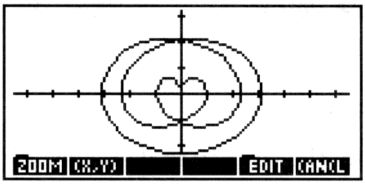
Step 4: After watching the plot, you suspect that the point (0,3) is a likely collision spot. Turn on TRACE and the coordinates viewer and move the cursor over to the suspected point. Note that it occurs at $t = 2$, which after substituting back into the original equations prove that there is a collision.

TRACE (X,Y) ▶ as necessary



Differential Equation Plots

Differential equation plotting is covered in detail in chapter 19. For an additional example of a differential equation plot, install TEACH (if necessary), and then press EXAM PLOTS DEQ :



Special Notes

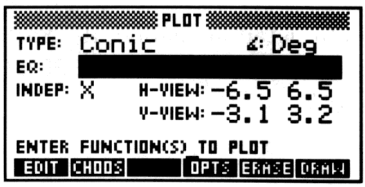
- The Diff Eq plot type uses the *axes* element of PPAR in a special way. It expects the two strings that normally contain axis-labels to each contain an integer. The integers indicate which solution component to plot on each axis ("0" indicates the independent variable, "1" indicates the first (or only) solution component, "2" indicates the second solution component (for a vector-valued solution), and so forth.

Conic Plots

The equation for a conic section is second degree or less in both x and y. For example, the following equations are all valid equations for plotting conic sections:

$$\begin{aligned} x^2 + y^2 + 4x + 2y - 5 &= 0 \quad \text{(circle)} \\ 5x^2 + 3y^2 - 18 &= 0 \quad \text{(ellipse)} \\ x^2 - 4x + 3y + 2 &= 0 \quad \text{(parabola)} \\ 2x^2 - 3y^2 + 3y - 5 &= 0 \quad \text{(hyperbola)} \end{aligned}$$

The Default CONIC Plot Screens



The Conic PLOT Form

∠:	Shows the current angle mode. Change it by pressing (+/-) one or more times or by using CHOOS.
EQ:	Enter the expression, equation, or program to be plotted.
INDEP:	Enter the name of the independent variable.
H-VIEW:	Enter the horizontal display range in the two fields, low endpoint in the left and high endpoint in the right.
V-VIEW:	Enter the vertical display range in the two fields, low endpoint in the left and high endpoint in the right.
OPTS	Opens the PLOT OPTIONS form.
ERASE	Erases the *PICT* screen (without displaying it).
DRAW	Stores all of the values in their appropriate places in the reserved variables, *EQ* and *PPAR*, and draws the plot accordingly, leaving you in the PICTURE environment when finished.

The Conic PLOT OPTIONS Form

INDEP:	Enter the name of the independent variable.
LO:	Enter the smallest value of the independent variable that you want *plotted*.
HI:	Enter the largest value of the independent variable that you want *plotted*. Plotting range can be different than display range (see page 24-3).
AXES	See the Function plot type.
CONNECT	See the Function plot type.
DEPND:	Enter the dependent (or second independent) variable.
STEP:	This determines the *resolution* of the plot. It is the horizontal distance (in units or pixels—see next field)

between two plotted points. The default step-size for Conic is an interval equal to 1 pixel.

PIXELS — When checked, the step-size is interpreted to represent pixels. When unchecked (default) the step-size is interpreted to represent units.

H-TICK — See the Function plot type.

V-TICK — See the Function plot type.

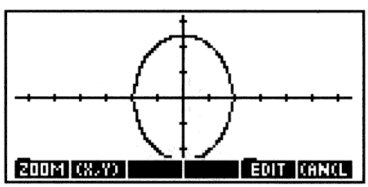
PIXELS — See the Function plot type.

Special Notes

- For conic plots, the HP 48 actually plots the two branches of the conic section separately. This may introduce one or two discontinuities in the connected graph. Specifying a smaller step size (decreasing the interval between plotted points) helps eliminate visual discontinuities.
- If flag -1 (Principal Values) is set, the Conic plot will only display its principal branch (half of the plot). Clear flag -1 and redraw to plot the entire conic section.
- Equations that are higher than second-order in the independent or dependent variable are converted to their second-order Taylor approximations before being plotted.
- Conic is a useful plot type for plotting systems of equations involving two variables where none of the equations are higher than second order in either variable (see Example 2 in "Truth Plots" for a sample application).

Example: Display the conic example plot, ELLIP: $5x^2 + 3y^2 - 18 = 0$. If necessary, type in TEACH to install the EXAMPLES directory then:

VAR EXAM PLOTS ELLIP

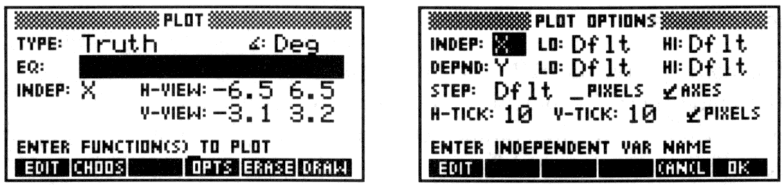


After plotting, press CANCEL → PLOT to review the PLOT forms that generated the plot. Experiment by changing values or settings and redrawing the plot.

Truth Plots

Truth plots evaluate expressions that return true (any nonzero real number) or false (0) results. At the coordinates for each pixel, the pixel is turned *on* if the expression is true—it's ***unchanged*** if the expression is false.

The Default TRUTH Plot Screens



The Truth PLOT Form

∠:	Shows the current angle mode. Change it by pressing (+/-) one or more times or by using CHOOS.
EQ:	Enter the truth-valued expression, inequality, or program to be plotted.
INDEP:	Enter the name of the independent variable. It will be plotted on the horizontal axis.
H-VIEW:	Enter the horizontal display range in the two fields, low endpoint in the left and high endpoint in the right.
V-VIEW:	Enter the vertical display range in the two fields, low endpoint in the left and high endpoint in the right.
OPTS	Opens the PLOT OPTIONS form.
ERASE	Erases the *PICT* screen (without displaying it).
DRAW	Stores all of the values in their appropriate places in the reserved variables, *EQ* and *PPAR*, and draws the plot accordingly, leaving you in the PICTURE environment when finished.

The Truth PLOT OPTIONS Form

INDEP:	Enter the name of the independent variable.
LO:	Enter the smallest value of the independent variable that you want *plotted*. Plotting range for Parametric

plots are usually different than display range (see page 24-3).

HI: Enter the largest value of the independent variable that you want *plotted*. Plotting range can be different than display range (see page 24-3).

DEPND: Enter the dependent (or second independent) variable. It will be plotted on the vertical axis.

LO: Enter the smallest value of the dependent variable that you want *plotted*.

HI: Enter the largest value of the dependent variable that you want *plotted*.

STEP: This determines the *resolution* of the plot. It is the horizontal distance (in units or pixels—see next field) between two plotted points. The default step-size for Truth is an interval equal to 1 pixel.

PIXELS When checked, the step-size is interpreted to represent pixels. When unchecked (default) the step-size is interpreted to represent units.

AXES See the Function plot type.

H-TICK See the Function plot type.

V-TICK See the Function plot type.

PIXELS See the Function plot type.

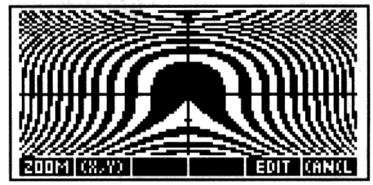
Special Notes

- Unless otherwise specified, every pixel in the display is evaluated. For a full-size display, this means *EQ* must be evaluated 8,384 times (compared to 131 times for the average function plot). You can speed up the plot by specifying a smaller x and y plotting range (see Example 2 below).

Examples

Example 1: Display the conic example plot, PTRN: $(x^2 + y^3)$ **mod** $2 < 4$. If necessary, type in TEACH to install the EXAMPLES directory then:

(VAR) EXAM PLOTS PTRN



After plotting, press (CANCEL) (↱)(PLOT) to review the PLOT forms that generated the plot. Experiment by changing values or settings and redrawing the plot.

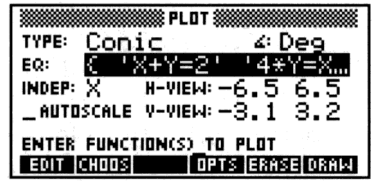
Example 2: Plot the solution set for the following system of inequalities: $x + y \geq 2$, $4y \leq x + 8$, $2y \geq 3x - 6$.

Step 1: Create a single truth expression: `'X+Y≥2 AND 4*Y≤X+8 AND 2*Y≥3*X-6'`. Store it in the variable *INEQ*.

Step 2: Create a list of the three inequalities with the inequality signs converted to equal signs (=): `{ 'X+Y=2' '4*Y=X+8' '2*Y=3*X-6' }`. Store the list in the variable *NEQL*.

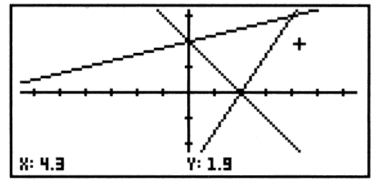
Step 3: Begin the PLOT application, change the plot type to `Conic`, reset the plotting defaults, and choose *NEQL* for the field `EQ:`.

(↱)(PLOT) (▲)(α)C
(DEL)(▼) OK
(▼) CHOOS OK



Step 4: Erase *PICT* and plot the three lines. (Note that these equations meet the requirements for Conic plots.) After drawing use `(X,Y)` to determine the region of interest for inequalities.

ERASE DRAW (X,Y)
(▲)(▼)(◀) or (▶) as necessary

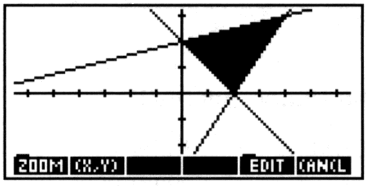


Step 5: Return to the PLOT form, change the plot type to Truth, choose the truth expression *INEQ* into the EQ: field, and reduce the plotting range to the region of interest that you determined from looking at the Conic plot.

NXT CANCEL ▲ α T ▼
CHOOS ▼ ▼ OK
OPTS ▶ 1 +/- ENTER 5
ENTER ▶ 1 +/- ENTER 4
ENTER

Step 6: Draw the truth plot, *without erasing PICT first*. The truth plot will be superimposed over the lines drawn previously.

OK DRAW



Statistical Plots

You can plot statistical data three ways:

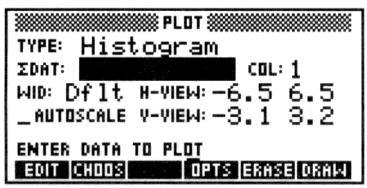
- **Scatter Plot.** For two variables, their values at each data point are depicted by a dot in the *x*-*y* plane.
- **Bar Chart.** For one variable, its value at each sequential data point is shown by a vertical bar.
- **Histogram.** For one variable, the number of times its value falls within certain ranges—called *bins*—is depicted by a vertical bar.

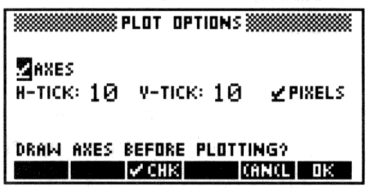
Statistical plots use data stored in the reserved matrix ΣDAT variable, which plays a role for statistics analogous to *EQ* for function plotting and solving.

Histogram Plots

A histogram divides the range of values of one variable into a number of *bins* and for each bin shows the number of data points for which the variable value falls within the bin. It shows the *relative frequency*—the maximum y value is the total number of data points.

The Default HISTOGRAM Plot Screens





The Histogram PLOT Form

ΣDAT:	Enter the data matrix or the name of the data matrix containing the data to be plotted.
COL:	Enter the column number in ΣDAT that contains the data to be plotted.
WID:	Enter the desired bar width. The default sets the width of each bar equal to one user-unit.
H-VIEW:	Enter the horizontal display range (in user-units) in the two fields, low endpoint in the left and high endpoint in the right.
V-VIEW:	Enter the vertical display range (in user-units) in the two fields, low endpoint in the left and high endpoint in the right.
AUTOSCALE:	When checked, the horizontal display range is set to match the range of the data in the selected ΣDAT column and the vertical display range is set so that all bars will fit vertically into the display, regardless of the actual distribution. When unchecked, the display uses the display ranges indicated by the H-VIEW and V-VIEW fields.
OPTS	Opens the PLOT OPTIONS form.
ERASE	Erases the *PICT* screen (without displaying it).
DRAW	Stores all of the values in their appropriate places in the reserved variables, ΣDAT, *PPAR*, and ΣPAR

and draws the plot accordingly, leaving you in the PICTURE environment when finished.

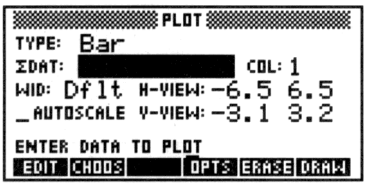
The Histogram PLOT OPTIONS Form

AXES	See the Function plot type.
H-TICK	See the Function plot type.
V-TICK	See the Function plot type.
PIXELS	See the Function plot type.

Bar Plots

A bar chart shows the values of one variable in the order they appear in the statistical matrix.

The Default BAR Plot Screens



The Bar PLOT Form

ΣDAT:	Enter the data matrix or the name of the data matrix containing the data to be plotted.
COL:	Enter the column number in ΣDAT that contains the data to be plotted.
WID:	Enter the desired bar width. The default sets the width of each bar equal to 1 user-unit.
H-VIEW:	Enter the horizontal display range (in user-units) in the two fields, low endpoint in the left and high endpoint in the right.
V-VIEW:	Enter the vertical display range (in user-units) in the two fields, low endpoint in the left and high endpoint in the right.

AUTOSCALE: When checked, the horizontal display range is set from 0 to n, where n is the number of data points in ΣDAT and the vertical display range is set so that all bars will fit vertically into the display and WID: set to the default (1 unit per bar). When unchecked, the display uses the display ranges indicated by the H-VIEW and V-VIEW fields.

OPTS Opens the PLOT OPTIONS form.

ERASE Erases the *PICT* screen (without displaying it).

DRAW Stores all of the values in their appropriate places in the reserved variables, ΣDAT, $PPAR$, and ΣPAR and draws the plot accordingly, leaving you in the PICTURE environment when finished.

The Bar PLOT OPTIONS Form

AXES See the Function plot type.

H-TICK See the Function plot type.

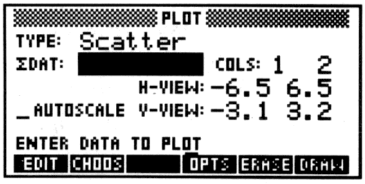
V-TICK See the Function plot type.

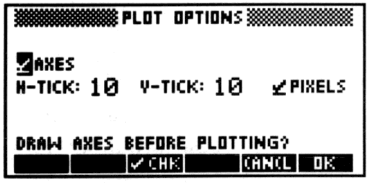
PIXELS See the Function plot type.

Scatter Plots

A scatter plot shows the relationship between two variables by plotting a point at each x-y coordinate pair. For variables that are statistically correlated, the points should cluster along a curve representing the statistical model.

The Default SCATTER Plot Screens





The Scatter PLOT Form

ΣDAT:	Enter the data matrix or the name of the data matrix containing the data to be plotted.
COLS:	Enter the column numbers in *ΣDAT* that contain the data to be plotted. The left field indicates the column to be plotted along the horizontal axis and the right field indicates the column to be plotted along the vertical axis.
H-VIEW:	Enter the horizontal display range (in user-units) in the two fields, low endpoint in the left and high endpoint in the right.
V-VIEW:	Enter the vertical display range (in user-units) in the two fields, low endpoint in the left and high endpoint in the right.
AUTOSCALE:	When checked, the horizontal and vertical display ranges are set so that all plotted points are displayed with as little extra space as possible. When unchecked, the display uses the display ranges indicated by the H-VIEW and V-VIEW fields.
OPTS	Opens the PLOT OPTIONS form.
ERASE	Erases the *PICT* screen (without displaying it).
DRAW	Stores all of the values in their appropriate places in the reserved variables, *ΣDAT*, *PPAR*, and *ΣPAR* and draws the plot accordingly, leaving you in the PICTURE environment when finished.

The Scatter PLOT OPTIONS Form

AXES	See the Function plot type.
H-TICK	See the Function plot type.
V-TICK	See the Function plot type.
PIXELS	See the Function plot type.

Special Notes

- After drawing the Scatter plot, press STATL to superimpose the a plot of the current regression model on the Scatter plot. This temporarily changes the plot type to Function so that executing a zoom will redraw the regression curve but not the scatter data.

Plotting Functions of Two Variables

There are six different plot types that can be used to help you visualize functions of two variables. Some of these display simulated three-dimensional surfaces; other provide distinct two-dimensional views of an underlying (but not displayed) "three-dimensional" function.

Sampling Grid

Functions of two independent variables need two inputs in order to generate an output. The HP 48 uses a two-dimensional *sampling grid* of points whose two coordinates provide the two inputs required.

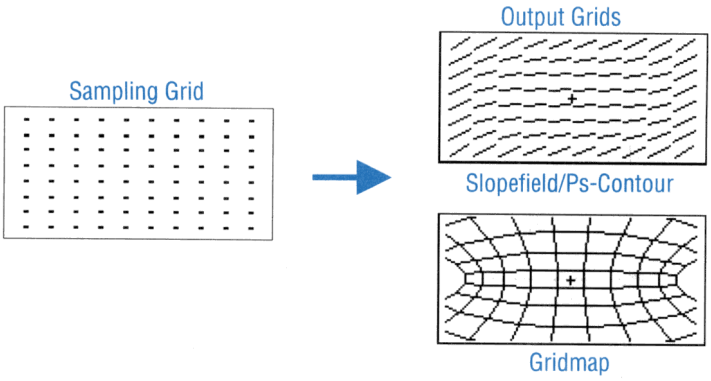
Each of the six plot types that use functions of two variables allow you to determine the size of the sampling grid. By default, it consists of 80 points—10 columns by 8 rows. Increasing the number of points in the sampling grid increases the time it takes to draw the plot—and the detail with which the function is plotted.

However, for plots of functions of two variables, more detail does not always translate into a more meaningful plot. Each combination of function and plot type has its own built-in optimum sampling grid size that is neither too small to properly reflect the function nor too large to obscure the important aspect. You will probably need to experiment a bit with the sampling grid dimensions when you first plot a function.

Output Grid

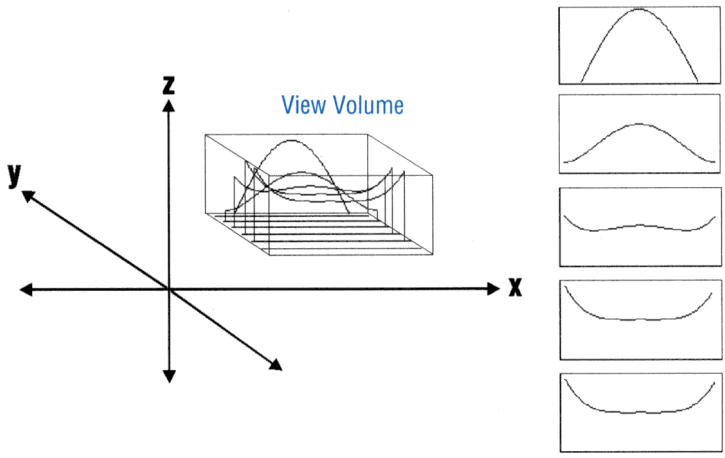
The six plot types each transform the sampling grid into an *output grid*, using the function to guide the transformation. However, each plot type makes use of the sampling grid in different ways.

Three of them—Slopefield, Ps-Contour, and Gridmap—take each set of sample coordinates and use the current equation to transform them into a new two-dimensional *output grid* that enables you to visualize the nature of the transforming equation. The plot you see is nothing more than the two-dimensional output grid.



Transforming Sampling Grid to Output Grid

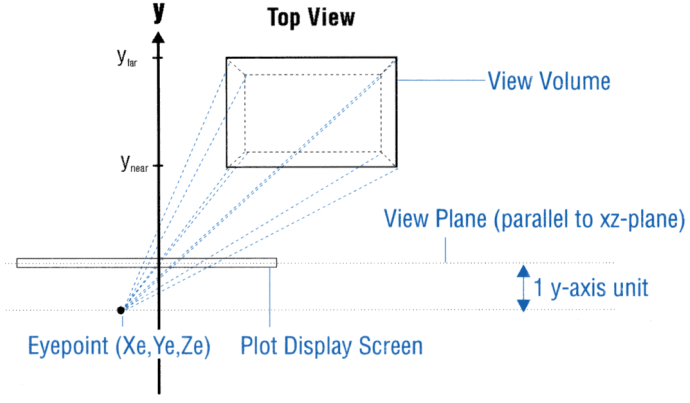
A fourth plot type, YSlice, performs the same transformation as Wireframe, but displays the output very differently. Instead of showing the entire output surface at once, YSlice plots, one after the other, two-dimensional cross-sections of the surface perpendicular to the y-axis. It draws one plot for each row in the sampling grid. Once it has completed drawing all of the "slices," it creates and runs an animation using each "slice" as one frame. This allows you to visualize a moving slice through the computed surface.



The YSLICE View

23

The final two, Wireframe and Pr-Surface, transform the 2-dimensional sample grid into a three-dimensional output *surface*. The plot that you see is the output surface as viewed from a specified vantage point—the *eyepoint*. The only part of the surface plotted will be that within the within a region in 3-dimensional space, called the *View Volume*, defined by ranges on each of the three coordinate axes.



The Relationship of the Eyepoint, View Volume, and Plot Display

Note that the three-dimensional coordinate system on the HP 48 is somewhat constrained compared to its abstract, mathematical counterpart. Specifically:

- The plot display screen does not rotate in space; it always stays parallel to the xz-plane and perpendicular to the y-axis. This means that, visually, "height" is always along the z-axis, "width" along the x-axis, and "depth" along the y-axis.
- The y-axis is always oriented so that negative y-values are "nearer" and positive y-values "farther" away from the plot display.
- The eyepoint must be at least one unit "nearer" than y_{near} ($y_e \leq y_{near} - 1$) and can never exist "within" the View Volume. Whenever you move the eyepoint, you also move the plot display screen so that it remains exactly one unit away in the direction of the y-axis.
- You cannot plot a "top" view of a function (looking down onto the xy-plane) simply by moving the eyepoint. (You may simulate this by transforming coordinates, however.)

Slopefield Plots

The Slopefield plot type draws a lattice of line segments whose slopes represent the value of the function, $f(x,y)$, at their centerpoint. Using Slopefield allows your eye to pick out integral curves of the differential equation $y' = F(x, y)$. It is quite useful in understanding the "arbitrary constant" in antiderivatives.

The Default SLOPEFIELD Plot Screens

The Slopefield PLOT Form

∠:	Shows the current angle mode. Change it by pressing +/- one or more times or by using CHOOS.
EQ:	Enter the current expression, equation, or user-defined function to be plotted. Names of variables containing expressions, equations, or UDFs may be used in place of the objects themselves.
INDEP:	Enter the name of one of the independent variables.
STEPS:	Enter the number of columns in the sampling grid.
DEPND:	Enter the name of the second independent variable.
STEPS:	Enter the number of rows in the sampling grid.
OPTS	Opens the PLOT OPTIONS form.
ERASE	Erases the *PICT* screen (without displaying it).
DRAW	Stores all of the values in their appropriate places in the reserved variables—*EQ*, *PPAR*, and *VPAR*—and draws the plot accordingly, leaving you in the PICTURE environment when finished.

23

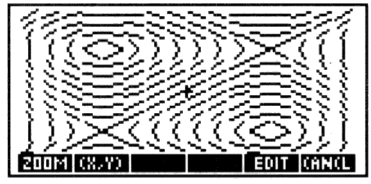
The Slopefield PLOT OPTIONS Form

X-LEFT: X-RIGHT:	Enter the horizontal display range, corresponding to the first independent variable (entered in INDEP).
Y-NEAR: Y-FAR:	Enter the vertical display range, corresponding to the second independent variable (entered in DEPND).

Examples

Example 1: Display the Slopefield example plot, SPFLD: $y' = \frac{(x^2-1)}{(y^2-1)}$. If necessary, type in TEACH to install the EXAMPLES directory and then:

Press VAR EXAM PLOTS
NXT SPFLD



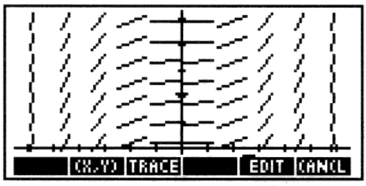
After plotting, press CANCEL → PLOT to review the PLOT forms that generated the plot. Experiment by changing values or settings and redrawing the plot.

Example 2: Plot the Slopefield of the differential equation, $y'(x) = x^2$. Then superimpose the solution of the equation with a particular initial condition.

Step 1: In the Slopefield PLOT form, enter the expression ('X^2') into EQ:, and set the display ranges to [-3 3] (horizontal) and [-1 5] (vertical). Set the independent step variable to 10 steps and the dependent step variable to 8 steps.

Step 2: Plot the slopefield.

ERASE DRAW



Step 3: Turn on TRACE mode, press ▼ seven times and ▶ one time to jump to a point in the lower left corner of the screen, and press ENTER to put the coordinates onto the stack.

Step 4: Return to the PLOT form and change the plot type to Diff Eq. Then highlight the solution variable's INIT: field and press NXT CALC DROP so that the coordinate point (a tagged list) is sitting on level 1 of the stack.

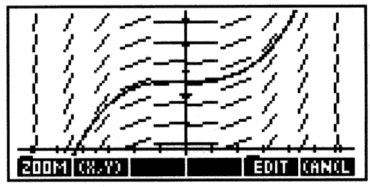
Step 5: Press PRG LIST OBJ→ DROP to remove the INPUT tag. Press OBJ→ DROP to separate the two coordinates, and then ↩ CONT OK to store the y-coordinate as the initial solution value.

Step 6: Highlight the independent variable's INIT: field and press CALC DROP OK to retrieve the x-coordinate as the initial independent value. Then set the FINAL: value to 3.

Step 7: Set the step-size to 0.1.

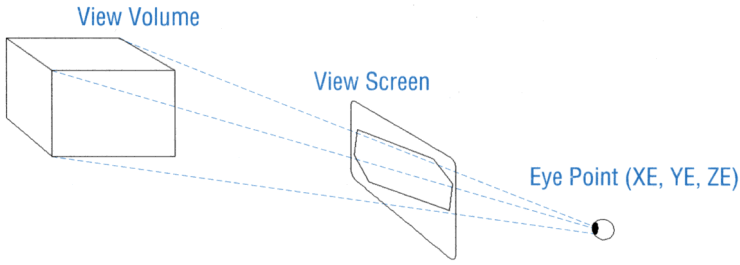
Step 8: Draw the plot *without erasing first* so as to superimpose the Diff Eq plot on the previous Slopefield plot.

OK DRAW



Wireframe Plots

The Wireframe plot type draws an oblique-view, perspective, 3D plot of a wire-frame model of the surface determined by $Z = F(x, y)$. Each point in the sampling grid is perspective-projected onto the display screen along the line connecting the sample and the eyepoint (X_e, Y_e, Z_e).



Perspective Projection

Neighboring samples are connected by straight lines. The sampling grid is determined by the "base" of the View Volume $(X_{left}, X_{right}, Y_{near}, Y_{far})$.

The Default WIREFRAME Plot Screens

The Wireframe PLOT Form

∠: Shows the current angle mode. Change it by pressing (+/-) one or more times or by using CHOOS.

EQ: Enter the current expression, equation, or user-defined functions to be plotted. Names of variables containing expressions, equations, or UDFs may be used in place of the objects themselves.

INDEP: Enter the name of one of the independent variables.

STEPS: Enter the number of columns in the sampling grid.

DEPND: Enter the name of the second independent variable.

STEPS: Enter the number of rows in the sampling grid.

OPTS Opens the PLOT OPTIONS form.

ERASE Erases the *PICT* screen (without displaying it).

DRAW Stores all of the values in their appropriate places in the reserved variables—*EQ*, *PPAR*, and *VPAR*—and draws the plot accordingly, leaving you in the PICTURE environment when finished.

The Wireframe PLOT OPTIONS Form

X-LEFT:
X-RIGHT: Enter the x-axis range ("width") of the View Volume.

Y-NEAR:
Y-FAR: Enter the y-axis range ("depth") of the View Volume.

Z-LOW:
Z-HIGH: Enter the z-axis range ("height") of the View Volume.

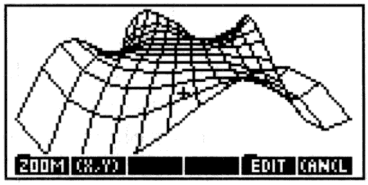
XE: Enter the x-coordinate of the eyepoint. Use the midpoint of the x-axis range for the View Volume if you want the plot "centered" horizontally in the display.

YE: Enter the y-coordinate of the eyepoint. It must be at least one unit smaller than the value for Y-NEAR: above. The greater the difference between YE: and Y-NEAR:, the "farther away" the plot will appear.

ZE: Enter the z-coordinate of the eyepoint. Use the midpoint of the z-axis range for the View Volume if you want the plot "centered" vertically in the display.

Example: Display the Wireframe example plot, WIRE : $z = x^3y - xy^3$. If necessary, type in TEACH to install the EXAMPLES directory and then:

Press (VAR) EXAM PLOTS (NXT) WIRE



After plotting, press (CANCEL) (→)(PLOT) to review the PLOT forms that generated the plot. Experiment by changing values or settings and redrawing the plot.

Pseudo-Contour Plots

The Ps-Contour plot type plots a lattice of line segments, each tangent to a contour of the function (a curve satisfying $F(x,y)$=constant). It computes a tangent for each point in the sampling grid. The Ps-Contour produces a "fast" contour plot, allowing your eye to pick out the integral curves (contours) without actually plotting them.

The Default PS-CONTOUR Plot Screens

The Ps-Contour PLOT Form

∠: Shows the current angle mode. Change it by pressing (+/−) one or more times or by using CHOOS.

EQ: Enter the current expression, equation, or user-defined function to be plotted. Names of variables containing expressions, equations, or UDFs may be used in place of the objects themselves.

INDEP: Enter the name of one of the independent variables.

STEPS: Enter the number of columns in the sampling grid.

DEPND: Enter the name of the second independent variable.

STEPS: Enter the number of rows in the sampling grid.

OPTS Opens the PLOT OPTIONS form.

ERASE Erases the *PICT* screen (without displaying it).

DRAW Stores all of the values in their appropriate places in the reserved variables—*EQ*, *PPAR*, and *VPAR*—and draws the plot accordingly, leaving you in the PICTURE environment when finished.

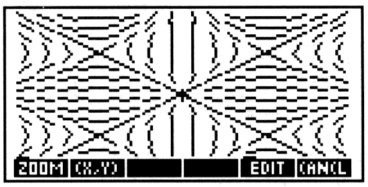
The Ps-Contour PLOT OPTIONS Form

X-LEFT: Enter the horizontal display range, corresponding to the
X-RIGHT: first independent variable (entered in INDEP).

Y-NEAR: Enter the vertical display range, corresponding to the
Y-FAR: second independent variable (entered in DEPND).

Example: Display the Ps-Contour example plot, PSCN: $z = \frac{(x^2-1)}{(y^2-1)}$. If necessary, type in TEACH to install the EXAMPLES directory and then:

Press (VAR) EXAM PLOTS (NXT) PSCN



After plotting, press (CANCEL) (→)(PLOT) to review the PLOT forms that generated the plot. Experiment by changing values or settings and redrawing the plot.

Y-Slice Plots

The YSLICE plot type draws a series of cross-sections (each perpendicular to the *y*-axis) of the surface determined by the current equation. It draws one plot for each row in the sampling grid. Once it has completed drawing all of the "slices," it creates and runs an animation using each "slice" as one frame. This allows you to visualize a moving slice through the computed surface.

The Default YSLICE Plot Screens

The Y-Slice PLOT Form

∠:	Shows the current angle mode. Change it by pressing (+/−) one or more times or by using CHOOS.
EQ:	Enter the current expression, equation, or user-defined function to be plotted. Names of variables containing expressions, equations, or UDFs may be used in place of the objects themselves.
INDEP:	Enter the name of one of the independent variables.

STEPS:	Enter the number of columns in the sampling grid.
DEPND:	Enter the name of the second independent variable.
STEPS:	Enter the number of rows in the sampling grid.
OPTS	Opens the PLOT OPTIONS form.
ERASE	Erases the *PICT* screen (without displaying it).
DRAW	Stores all of the values in their appropriate places in the reserved variables—*EQ*, *PPAR*, and *VPAR*—and draws the series of plots accordingly. Once it has drawn all of the slices, it animates them in a continuously repeating loop. Press CANCEL to stop the animation.

The YSlice PLOT OPTIONS Form

X-LEFT: X-RIGHT:	Enter the x-axis range ("width") of the View Volume.
Y-NEAR: Y-FAR:	Enter the y-axis range ("depth") of the View Volume.
Z-LOW: Z-HIGH:	Enter the z-axis range ("height") of the View Volume.
SAVE ANIMATION	When checked, the series of "slices" used in the animation are put onto the stack and the number of slices put onto level 1 of the stack. When unchecked, all "slices" except the current one are deleted once the PICTURE environment is exited.

Example: Display the YSlice example plot, SLICE: $z = x^3y - xy^3$. If necessary, type in TEACH to install the EXAMPLES directory and then:

Press VAR EXAM PLOTS NXT SLICE

Press CANCEL to halt animation.

After plotting, press CANCEL ↷ PLOT to review the PLOT forms that generated the plot. Experiment by changing values or settings and redrawing the plot.

Gridmap Plots

The GRIDMAP plot type transforms (maps) the specified rectilinear sampling grid via the current complex-valued function. The coordinates (a complex number) of each point in the sampling grid are the inputs for the function which then maps the coordinates into the output grid. You can control how much of the output grid is displayed by adjusting the x- and y-ranges for the View Volume.

The Default GRIDMAP Plot Screens

23

The Gridmap PLOT Form

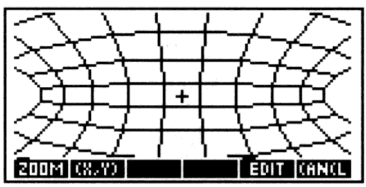
∠:	Shows the current angle mode. Change it by pressing (+/-) one or more times or by using CHOOS.
EQ:	Enter the current expression, equation, or user-defined function to be plotted. Names of variables containing expressions, equations, or UDFs may be used in place of the objects themselves. The function or expression in *EQ* should use a *complex number* argument.
INDEP:	Enter the name of one of the independent variables.
STEP:	Enter the number of columns in the sampling grid.
DEPND:	Enter the name of the second independent variable.
STEP:	Enter the number of rows in the sampling grid.
OPTS	Opens the PLOT OPTIONS form.
ERASE	Erases the *PICT* screen (without displaying it).
DRAW	Stores all of the values in their appropriate places in the reserved variables—*EQ*, *PPAR*, and *VPAR*—and draws the plot accordingly, leaving you in the PICTURE environment when finished.

The Gridmap PLOT OPTIONS Form

X-LEFT: X-RIGHT:	Enter the horizontal display range.
Y-NEAR: Y-FAR:	Enter the vertical display range.
XX-LEFT: XX-RIGHT:	Enter the horizontal range of the input sampling grid, corresponding to the first independent variable (entered in INDEP).
YY-NEAR: YY-FAR:	Enter the vertical range of the input sampling grid, corresponding to the second independent variable (entered in DEPND).

Example: Display the Gridmap example plot, GRID: $x + y\mathbf{i} \Rightarrow \sin(x + y\mathbf{i})$. If necessary, type in TEACH to install the EXAMPLES directory and then:

Press VAR EXAM PLOTS NXT GRID

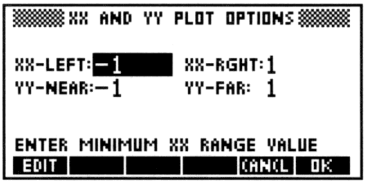


After plotting, press CANCEL ↷ PLOT to review the PLOT forms that generated the plot. Experiment by changing values or settings and redrawing the plot.

Parametric Surface Plots

The Pr-Surface plot type draws an oblique-view, perspective, 3D plot of a wire-frame model of the surface determined by $F(u, v) = x(u, v)\mathbf{i} + y(u, v)\mathbf{j} + z(u, v)\mathbf{k}$ where u and v are drawn from the sampling grid (XX- and YY- ranges). Pr-Surface combines the coordinate-mapping approach of Gridmap with the three-dimensional perspective plotting of Wireframe.

The Default PR-SURFACE Plot Screens



23

The Pr-Surface PLOT Form

∠:	Shows the current angle mode. Change it by pressing (+/−) one or more times or by using CHOOS.
EQ:	Enter the function as a list containing the three algebraic objects representing the parametric vector components.
INDEP:	Enter the name of one of the independent variables.
STEP:	Enter the number of columns in the sampling grid.
DEPND:	Enter the name of the second independent variable.
STEP:	Enter the number of rows in the sampling grid.
OPTS	Opens the PLOT OPTIONS form.
ERASE	Erases the *PICT* screen (without displaying it).
DRAW	Stores all of the values in their appropriate places in the reserved variables—*EQ*, *PPAR*, and *VPAR*—and draws the plot accordingly, leaving you in the PICTURE environment when finished.

The Pr-Surface PLOT OPTIONS Form

X-LEFT:
X-RIGHT:
: Enter the x-axis range ("width") of the View Volume.

Y-NEAR:
Y-FAR:
: Enter the y-axis range ("depth") of the View Volume.

Z-LOW:
Z-HIGH:
: Enter the z-axis range ("height") of the View Volume.

XE:
: Enter the x-coordinate of the eyepoint. Use the midpoint of the x-axis range for the View Volume if you want the plot "centered" horizontally in the display.

YE:
: Enter the y-coordinate of the eyepoint. It must be at least one unit smaller than the value for Y-NEAR: above. The greater the difference between YE: and Y-NEAR:, the "farther away" the plot will appear.

ZE:
: Enter the z-coordinate of the eyepoint. Use the midpoint of the z-axis range for the View Volume if you want the plot "centered" vertically in the display.

XX,YY
: Opens the XX AND YY PLOT OPTIONS form, in order to enter the sampling grid ranges.

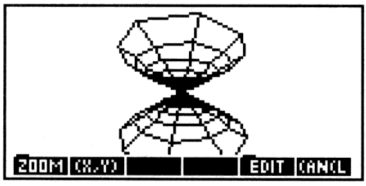
The Pr-Surface XX AND YY PLOT OPTIONS Form

XX-LEFT:
XX-RIGHT:
: Enter the horizontal range of the input sampling grid.

YY-NEAR:
YY-FAR:
: Enter the vertical range of the input sampling grid.

Example: Display the Pr-Surface example plot, PSUR: $F(u,v) = x(u,v)\mathbf{i} + y(u,v)\mathbf{j} + z(u,v)\mathbf{k}$ where $x(u,v) = u\cos v$, $y(u,v) = u\sin v$, and $z(u,v) = u$. If necessary, type in TEACH to install the EXAMPLES directory and then:

Press VAR EXAM PLOTS NXT PSUR



After plotting, press CANCEL ↱ PLOT to review the PLOT forms that generated the plot. Experiment by changing values or settings and redrawing the plot.

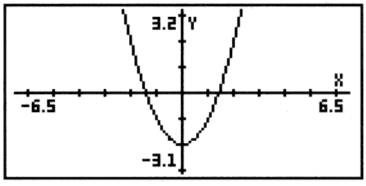
24

Advanced Plot Options

Labeling and Locating the Axes

To label the coordinate axes with the variable names:

- After the plot is drawn, press EDIT (NXT) LABEL. The names of the independent and dependent variables and the coordinates (in user-units) of the largest and smallest *displayed* values for each variable are added to the plot. The figure below shows labels added to the plot of $x^2 - 2$ using the default settings.



To label the axes with user-defined labels:

1. Press (CANCEL) to return to the stack, if necessary.
2. Enter a list containing the horizontal and vertical axis labels (as strings): { "*h-label*" "*v-label*" }
3. Press ◀ (PLOT) PPAR (NXT) AXES to store the labels.
4. Press ◀ (PICTURE) to display the plot again.
5. Press EDIT (NXT) LABEL.

To have the axes intersect at some point other than (0,0):

1. From the stack, press ◀ ({}).
2. Type in the complex number containing the desired point of intersection and press (ENTER).

3. Press ◀ PLOT PPAR NXT AXES to store the intersection point.
4. Press ▶ PLOT ERASE DRAW to redraw the plot using the new axes intersection point.

Plotting Programs and User-Defined Functions

You can plot more than just expressions and equations—you can also plot programs. And your expressions, equations, and programs can include user-defined functions.

You can plot a program if it takes nothing from the stack, uses the independent variable in the program, and returns exactly one untagged number to the stack:

- **Real result.** Equivalent to the expressions $f(x)$ (plot type Function) and $r(\theta)$ (plot type Polar). For example, the program

  ```
  « IF 'X<0' THEN '3*X^3-45*X^2+350' ELSE 1000 END»
  ```

  plots

  $$f(x) = \begin{cases} 3x^3 - 45x^2 + 350 & \text{if } x < 0 \\ 1000 & \text{if } x \geq 0 \end{cases}$$

  Store the program in *EQ*, select autoscaling, and draw the plot.

- **Complex result.** Equivalent to $(x(t), y(t))$ (plot type Parametric). For example, the program

  ```
  « 't^2-2' →NUM 't^3-2*t+1' →NUM R→C »
  ```

  plots the parametric equations

  $$x = t^2 - 2 \qquad \text{and} \qquad y = t^3 - 2t + 1$$

Store the program in *EQ*, make 'T' the independent variable, select autoscaling, and draw the plot.

Plotting Range vs. Display Range

The *plotting range* is the range of the independent variable (or variables) over which the current equation is evaluated. If you don't specify the plotting range, the HP 48 uses the x-axis display range (specified by XRNG in ◀ PLOT or by H-VIEW in ▶ PLOT) as the plotting range. However, you can specify a plotting range that's different from the x-axis display range:

- For Polar and Parametric plots, the independent variable isn't related to the x-axis variable—so you specify the plotting range to control the range of the independent variable.
- For Truth and Conic plots, you can shorten plotting time by specifying plotting ranges that are smaller than the x- and y-axis display ranges. These plot types require you to specify the *dependent* variable—you can specify its plotting range different from the y-axis display range.

You can make *PICT* larger than its default size (131 by 64 pixels)—and either keep the same x and y scale factors (which extends the display range), or keep the same display range (which extends the scale and appears to "stretch" the plot.

To check the current size of PICT:

- Press PRG PICT PICT ▶ RCL. You will see Graphic *width*× *height*—the current dimensions of PICT.

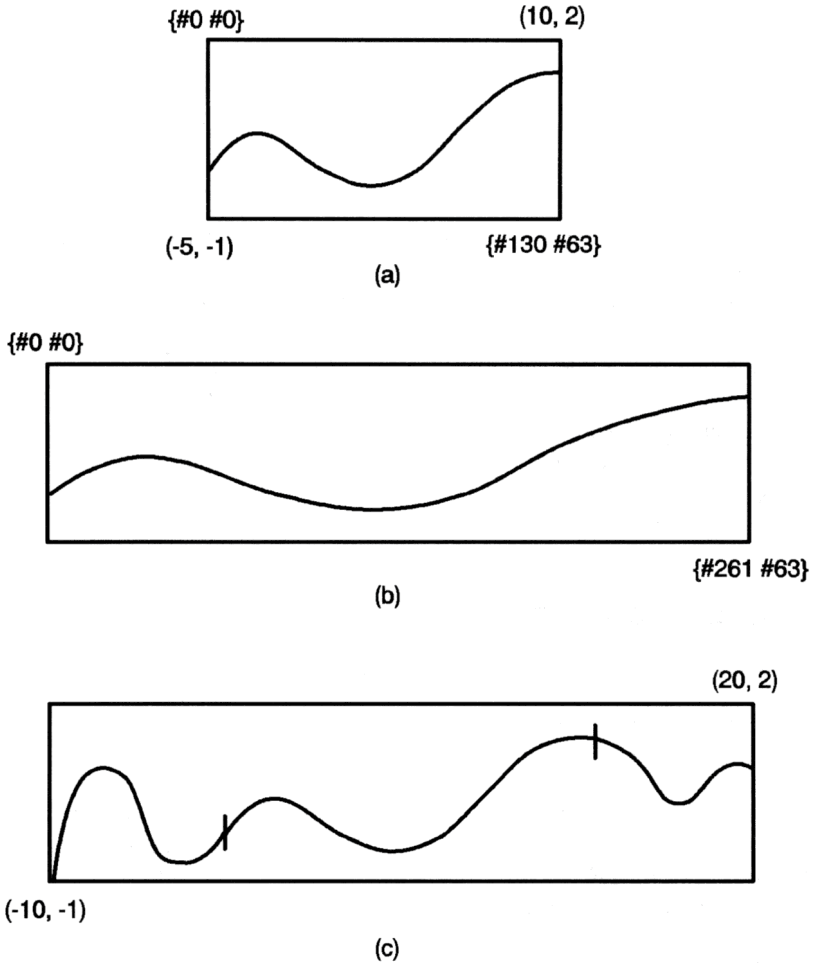
To change the size of PICT:

- To keep the same scaling, enter two complex numbers (with ( ) delimiters) specifying the coordinates of diagonally opposite corners in user-units, then press PRG PICT PDIM.
- To keep the same display ranges, enter two binary integers (with # delimiter) specifying the horizontal and vertical sizes in pixels, then press PRG PICT PDIM.

The result of the PDIM (*PICT dimension*) command depends on the type of coordinates—user-units or pixels—though both forms change the size of *PICT*.

Example: Assume *PICT* is its default size, shown in figure (a) below.

To double the x range of *PICT* in the horizontal direction and keep the same scales (units per pixel), enter `(-10,-1)` and `(20,2)` and press `PDIM`. (*PICT* becomes #261 wide by #64 high in pixel units.) If you redraw the graph, the effect is to add more points to the graph at both ends, shown in figure (c).



Changing the Size of *PICT*

To extend the plotting range beyond the limits of the display:

1. Change the the size of *PICT* so that it encompasses the plotting range. Enter two complex numbers (with ( ) delimiters) specifying the coordinates of diagonally opposite corners in user-units, then press PRG PICT PDIM.
2. Open the PLOT application and set the plotting ranges and display ranges. The display range may be smaller than the plotting range.

3. After all plot parameters are set, press ERASE DRAW to draw the plot. You will only see a portion of the plot in the display.
4. Press ◀ PICTURE and use the arrow keys to scroll around and view the larger plot. Press ◀ PICTURE again to exit scrolling mode.

To use computed values for plotting or display ranges:

1. In the PLOT or PLOT OPTIONS form, highlight the range field whose value you wish to compute.
2. Press NXT CALC to prepare for a side-calculation on the stack.
3. Perform the desired computation. For example, if you want to use $\frac{3\pi}{4}$ as one endpoint, you would press 3 ◀ π × 4 ÷.
4. If it is not already one, convert the result on level 1 to a real number by pressing ◀ →NUM.
5. Press OK to return the result to the original field.

Saving and Restoring Plots

A *plot* can consist of several components:

- The plot *picture*, a graphic object.
- The current equation or equations, stored in the reserved variable *EQ*.
- The current plot parameters, set in the PLOT forms, and stored in the reserved variable *PPAR* and, in the case of three-dimensional plot types, *VPAR*.
- Flag settings that determine plotting or display options.

You have the option to save any or all of these plot components in a variable so that you may retrieve them at a later time. Here are two useful approaches:

1. Save just the plot picture—the "result"—in a variable. This is a simple procedure (see below) but each plot picture uses about one Kilobyte of memory.
2. Save the current *EQ*, *PPAR*, *VPAR* (if necessary), and flag settings in a list. The plot can be reconstructed by restoring each of these to the values contained in the list.

The following set of procedures illustrate how to accomplish these approaches to saving and restoring plots.

To save the current plot picture in a variable:

1. After drawing the plot and while viewing it in PICTURE, press [STO] to send a copy of the plot to the stack as a graphics object. Press [CANCEL] to return to the stack.
2. Enter a name for the plot (`'P1'`, for example) and press [STO].

To view a plot picture that was previously stored in a variable:

1. Press [VAR] and then [→] and the menu key corresponding to the variable containing the plot picture (P1 , for example) to recall it to the stack.
2. Press [PRG] PICT PICT [STO] to store the plot picture in *PICT*.
3. Press [←][PICTURE] to view the plot picture.

To save a "reconstructable" version of the current plot:

1. After drawing the plot, press [CANCEL] to return to the stack.
2. Recall the current contents of *EQ*, *PPAR*, and *VPAR* (if the current plot is a three-dimensional plot) to the stack by pressing [VAR] and then EQ PPAR and VPAR (if necessary). You may need to use [NXT] to switch menu pages to find each of these variables.
3. Press [←][MODES] FLAG [NXT] RCLF to recall the current flag settings to the stack.
4. You should have three or, if you included VPAR, four new objects on the stack. Enter the number (3 or 4) and press [PRG] LIST →LIST to group these objects into a list.
5. Enter a name for the list and press [STO].

To reconstruct a plot from its stored version:

1. Press [VAR] and then the menu key associated with the variable containing the stored version (in list form) of the plot.
2. Press [PRG] TYPE OBJ→ [DROP] to disassemble the list and put the components onto the stack.
3. Press [←][MODES] FLAG [NXT] STOF to restore the flag settings. Note that your current flag settings will be lost.

4. If the plot is a three-dimensional plot, press ', type VPAR and press STO to restore *VPAR*.
5. Press ', type PPAR and press STO to restore *PPAR*.
6. Press ◀ PLOT ◀ EQ to restore *EQ*.
7. Press ▶ PLOT ERASE DRAW to redraw the plot.

25

The Equation Library

The Equation Library is a collection of equations and commands that enable you to solve common science and engineering problems. The library consists of more than 300 equations grouped into 15 technical subjects containing more than 100 problem titles. Each problem title contains one or more equations that help you solve that type of problem. Appendix G contains a table of the groups and problem titles available in the Equation Library. More detailed information about each equation set is given in the *HP 48G Series Advanced User's Reference*.

Solving a Problem with the Equation Library

Follow these steps for solving a problem using the Equation Library:

1. Press (→) EQLIB to start the Equation Library.
2. Set the unit options you want by pressing the SI, ENGL, and UNITS menu keys.
3. Highlight the subject you want, then press ENTER.
4. Highlight the title you want.
5. Optional—if you want to find out more about the equations in this set, press other keys as described in the next few sections.
6. For each known variable, type its value and press the corresponding menu key. Press NXT if necessary to access additional variables.
7. Press SOLV to start solving the problem.

8. For each known variable, type its value and press the corresponding menu key. Press NXT if necessary to access additional variables.

9. Optional: Supply a guess for the unknown variable(s). This can speed the solution process or help to focus on one of several solutions. Enter a guess just as you would the value of a known variable. If you are working with multiple equations, press ' and then the variable menu key MCAL after entering the guess (reconverting the menu label to white).

10. Press ← followed by the menu key of the variable for which you're solving. If you're solving a set of equations, you can press ← ALL to solve for all remaining unknown variables—all variables not previously defined by you.

Using the Solver

25 When you select a subject and a title in the Equation Library, you specify a set of one or more equations. Then, when you press SOLV, you leave the Equation Library catalogs and start solving the equations you've selected.

When you press SOLV in the Equation Library, the application does the following:

- The set of equations is stored in the appropriate variable: *EQ* for one equation, *EQ* and *Mpar* for more than one equation. (*Mpar* is a reserved variable name used by the Multiple-Equation Solver.)
- Each variable is created and set to zero *unless it already exists*. (If the variable name has been used by the Solver before, then it is a global variable and therefore already exists—until you purge it.)
- Each variable's units are set to the conditions you specified—SI or English units, and units used or not used—*unless the variable already exists* and has units dimensionally consistent with what you specified. (To change from English to SI units or vice-versa, you must first purge the existing variables *or* explicitly enter the units with the values.)
- The appropriate solver is started: the SOLVR application (see page 18-7) for one equation, the Multiple-Equation Solver for more than one equation.

Because *EQ* and *Mpar* are variables, you can have a different *EQ* and *Mpar* for each directory in memory.

Using the Menu Keys

The actions of the unshifted and shifted variable menu keys for *both* solvers are identical. Notice that the Multiple-Equation Solver uses two forms of menu labels: black and white. The NXT key shows additional menu labels, if required. In addition, each solver has special menu keys, which are described in the following table. You can tell which solver is started by looking at the special menu labels. (Or you can check the title—the title for a library equation in the HP Solve application starts with EQ:.)

Actions for Solver Menu Keys

Operation	SOLVE Application	Multiple-Equation Solver
Store value	X	X X
Solve for value	↰ X	↰ X ↰ X
Recall value	↱ X	↱ X ↱ X
Evaluate equation	EXPR=	
Next equation (if applicable)	NXEQ	
Undefine all		ALL
Solve for all		↰ ALL
Progress catalog		↱ ALL
Set states		MUSE MCAL

Browsing in the Equation Library

When you select a subject and title in the Equation Library, you specify a set of one or more equations. You can get the following information about the equation set from the Equation Library catalogs:

- The equations themselves and the number of equations.
- The variables used and their units—you can also change the units.
- A picture of the physical system (for most equation sets).

Viewing Equations

All equations have a *display form*—some equations also have a *calculation form*. The display form gives the equation in its basic form—the form you'd see in books. The calculation form includes computational refinements. If the equation has a calculation form, an * appears in the upper-left corner of the equation display.

Operations for Viewing Equations and Pictures

Key	Action	Example
EQN NXEQ	Shows display form of current or next equation in EquationWriter format.	$B = \frac{\mu 0 \cdot \mu r \cdot I}{2 \cdot \pi \cdot r}$
ENTER	Shows display form of current or next equation as an algebraic object. ENTER or ▼ shows the next equation, ▲ shows the previous.	'B=(μ0*μr*I)/(2*π*r)'
→STK	Shows calculation forms by putting a list containing the current set of equations on the stack.	{ 'B=IFTE(r<rw, CONST(μ0)*μr*I*r/(2*π*rw^2), CONST(μ0)*μr*I/(2*π*r))' }

Viewing Variables and Selecting Units

After you select a subject and title, you can view the catalog of names, descriptions, and units for the variables in the equation set by pressing VARS. The table below summarizes the operations available to you in the Variable catalogs.

Operations in Variable Catalogs

Key	Action
NXT	Toggles between catalog of descriptions and catalog of units.
SI ENG	Makes SI or English units active *unless* this conflicts with the units already defined for an existing (global) variable. Purge existing variables (or enter the specific units) to eliminate conflicts.
UNITS	Toggles between units used or units not used.
→VAR	Creates or changes all equation variables to have indicated unit type and usage.
PURG	Purges all equation variables for this title in the current directory. This also eliminates SI vs. English units conflicts.

Viewing the Picture

After you select a subject and title, you can view the picture of the problem—but only if the title has a picture.

To see the picture, press PIC. While the picture is displayed, you can do the following:

- Press →PICT to store the picture in *PICT*, the graphics memory—then you can use ◀ PICTURE to view the picture after leaving the Equation Library catalogs.
- Press the menu keys or ENTER to show other equation information.

For information about displaying and manipulating graphics objects, see chapter 9, "Graphic Objects."

Using the Multiple-Equation Solver

The Equation Library starts the Multiple-Equation Solver automatically if the equation set contains more than one equation. However, you can also start it up explicitly using your own set of equations (see "Defining a Set of Equations" on page 25-8).

When the Equation Library starts the Multiple-Equation Solver, it first stores the equation set in *EQ* and stores a copy of the equation set, the list of variables plus additional information in *Mpar*. *Mpar* is then used to set up the Solver menu for the current equation set. (Note that, although you can view and edit *EQ* directly like any other variable, *Mpar* can only be edited indirectly (executing commands that modify it) as it is structured as library data dedicated to the Multiple Equation Solver application.)

The following table summarizes the actions for the solver menu keys. The NXT key shows additional menu labels.

25

Solver Menu Keys

Operation	Key	Action
Store value	X X	Creates variable if necessary, stores value in variable, and makes variable user-defined. If the value has no units, the units of the previous value are appended, if any.
Solve for value	← X ← X	Creates variable if necessary, solves for value of variable, and makes variable not user-defined.
Recall value	→ X → X	Recalls value of variable to stack.
Undefine all	ALL	Makes all variables not user-defined, but does not modify their values.

Solver Menu Keys (continued)

Operation	Key	Action
Solve for all	◀ ALL	Creates variables if necessary and solves for all variables not user-defined (or as many as possible).
Progress catalog	▶ ALL	Shows information about last solution.
User-defined	MUSE	Sets states to user-defined for variable or list of variables on the stack.
Calculated	MCAL	Sets state to *not* user-defined (calculated result) for variable or list of variables on the stack.

The menu labels for the variable keys are white at first—they change during the solution process as described below.

Because a solution involves many equations and many variables, the Multiple-Equation Solver must keep track of variables that are user-defined and not defined—those it can't change and those it can. In addition, it keeps track of variables that it used or found during the last solution process.

The menu labels indicate the states of the variables. They're automatically adjusted as you store values and solve for variables. You can check that variables have proper states when you supply guesses and find solutions.

Notice that ▪ marks the variables that were used in the last solution—their values are compatible with each other. Other variables may *not* have compatible values because they weren't involved in the solution.

Meanings of Menu Labels

Label	Meaning
X0	Value *x0* not defined by you and not used in the last solution—it can change in the next solution.
X0■	Value *x0* not defined by you, but found in the last solution—it can change in the next solution.
X0	Value *x0* defined by you, but not used in the last solution—it can't change in the next solution (unless you solve for only this variable).
X0■	Value *x0* defined by you and used in the last solution—it can't change in the next solution (unless you solve for only this variable).

25

Defining a Set of Equations

When you design a set of equations, you should do it with an understanding of how the Multiple-Equation Solver uses the equations to solve problems.

The Multiple-Equation Solver uses the same process you'd use to solve for an unknown variable—assuming you weren't allowed to create additional equations. You'd look through the set of equations for one that has only one variable that you don't know. You'd use the HP 48 root-finder to find its value. Then you'd do this again until you've found the variable you want.

You should choose your equations to allow likely unknown variables to occur individually in equations. You must avoid having two or more unknown variables in all equations. You can also specify the equations in an order that's best for your problems.

For example, the following three equations define initial velocity and acceleration based on two observed distances and times. The first two equations alone are mathematically sufficient for solving the problem, but each equation contains two unknown variables. Adding the third equation allows a successful solution because it contains only one of the unknown variables.

$$x_1 = v_0 + a \cdot t_1$$
$$x_2 = v_0 + a \cdot t_2$$
$$(x_2 - x_1) = a \cdot (t_2 - t_1)$$

To create more robust equations, you can include functions that ensure proper and faster calculations—for example CONST and TDELTA, UBASE, EXP, and IFTE. See the *HP 48G Series Advanced User's Reference* for details and examples.

If your equations use any of the following functions, their variables won't necessarily be detected by the Multiple-Equation Solver: Σ, ∫, ∂, |, QUOTE, APPLY, TVMROOT, and CONST.

The list of equations in *EQ* may contain menu definitions, but those definitions are ignored by MINIT when it creates *Mpar*. However, you can reorder the menu labels using MITM, described under "Changing the Title and Menu" later in this chapter.

To create a set of equations for the Multiple Equation Solver:

1. Enter each equation in the set onto the stack.
2. Press ▲ to begin the Interactive Stack and then move the cursor up to the level containing the first equation you entered.
3. Press →LIST to combine them into a list.
4. Press ' α E α Q STO (or ← SOLVE ROOT ← EQ) to store the list into the *EQ* variable.
5. Press ← EQ LIB MES MINIT to create *Mpar* and prepare the equation set for use with the Multiple Equation Solver.
6. Press MSOL to launch the solver with the new set of equations.

To change the title and menu for a set of equations:

1. Make sure the set of equations is the current set (they are used when the Multiple Equation Solver is launched).
2. Enter a text string containing the new title onto the stack.
3. Enter a list containing the variable names in the order you want them to appear in the menu. Use a "" to insert a blank label. You must include *all* variables in the original menu and no others, and you must match uppercase and lowercase names properly.
4. Press ← EQ LIB MES MITM .

Interpreting Results from the Multiple Equation Solver

The Multiple-Equation Solver solves for variables by repeatedly looking through the set of equations for one that contains only one variable that's "unknown" (not user-defined and not found by the solver during this solution)—then it uses the HP 48 root-finder to find that value. It continues eliminating "unknown" variables until it solves for the variable you specified—or until it can't solve for any more variables. Each time the Multiple-Equation Solver starts solving for a variable, only the variables with black menu labels are "known."

During the solution process, the Multiple-Equation Solver shows the variable it's currently solving for. It also shows the type of root found by the HP 48 root-finder (zero, sign reversal, or extremum)—or the problem if no root is found (bad guesses or constant). (You can watch the iterations if you press any key except CANCEL during the root-finding process. For more information about the root-finder, see chapter 18.

The following messages indicate errors in the problem setup:

- `Bad Guess(es)`. Units may be missing or inconsistent for a variable. For a list of guesses, at least one of the list elements must have consistent units.
- `Too Many Unknowns`. The solver eventually encountered only equations having at least two unknowns. Either enter other known values, or change the set of equations—whichever is appropriate for your problem. See "Changing the Equations" later in this chapter.
- `Constant?` The initial value of a variable may be leading the root finder in the wrong direction. Supply a guess in the opposite direction from a critical value—if negative values are valid, try one.

Checking Solutions

The variables having a ▪ mark in their menu labels are related for the most-recent solution—they form a compatible set of values satisfying the equations used. The values of any variables *without* marks may not satisfy the equations because those variables weren't involved in the solution process.

If any solutions seem improper, check for the following problems:

- Wrong units. A known or found variable may have units different from those you assumed. These are global variables. If the variable existed before this calculation, then its unit system (SI or English) takes priority. To correct the units, either purge the variables before solving the equation, or enter the specific units you want.
- No units. If you're not using units, your implied units may not be compatible among your variables or with the implied units of constants or functions. The current angle mode sets the implied units for angles.
- Multiple roots. An equation may have multiple roots, and the solver may have found an inappropriate one. Supply a guess for the variable to focus the search in the appropriate range.
- Wrong variable states. A known or unknown variable may not have the appropriate state. A known variable should have a black menu label, and an unknown variable should have a white label.
- Inconsistent conditions. If you enter values that are mathematically inconsistent for the equations, the application may give results that satisfy *some* equations, but not *all*. This includes over-specifying the problem, for which you enter values for more variables than needed to define a physically realizable problem—the extra values may create an impossible or illogical problem. (The solution satisfies the equations the solver used, but the solver doesn't try to verify that the solution satisfies *all* of the equations.)
- Not related. A variable may not be involved in the solution (no ▪ in its menu label), so it's not compatible with the variables that *were* involved.
- Wrong direction. The initial value of a variable may be leading the root finder in the wrong direction. Supply a guess in the opposite direction from a critical value—if negative values are valid, try one.

Using the Constants Library

The Constants Library contains a collection of common physical constants and quantities. You can use them in equations and programs. (Several of these constants are used by the Equation Library.) The following table lists them in the order they appear in the Constants Library.

Constants Library

Name	Description	Value (SI)
NA	Avogadro's number	6.0221367E23 $gmol^{-1}$
k	Boltzmann constant	1.380658E−23 J/K
Vm	Molar volume	22.4141 l/gmol
R	Universal gas constant	8.31451 J/(gmol·K)
StdT	Standard temperature	273.15 K
StdP	Standard pressure	101.325 kPa
σ	Stefan-Boltzmann constant	5.67051E−8 W/(m^2·K^4)
c	Speed of light in vacuum	299792458 m/s
$\varepsilon 0$	Permittivity of vacuum	8.85418781761E−12 F/m
$\mu 0$	Permeability of vacuum	1.25663706144E−6 H/m
g	Acceleration of gravity	9.80665 m/s^2
G	Gravitational constant	6.67259E−11 m^3/(s^2·kg)
h	Planck's constant	6.6260755E−34 J·s
hbar	Dirac's constant	1.05457266E−34 J·s
q	Electron charge	1.60217733E−19 C
me	Electron rest mass	9.1093897E−31 kg
qme	q/me	175881962000 C/kg
mp	Proton rest mass	1.6726231E−27 kg
mpme	mp/me	1836.152701
α	Fine structure constant	0.00729735308
ϕ	Magnetic flux quantum	2.06783461E−15 Wb
F	Faraday constant	96485.309 C/gmol

Constants Library (continued)

Name	Description	Value (SI)
R∞	Rydberg constant	10973731.534 m^{-1}
a0	Bohr radius	0.0529177249 nm
μB	Bohr magneton	9.2740154E−24 J/T
μN	Nuclear magneton	5.0507866E−27 J/T
λ0	Photon wavelength (ch/e)	1239.8425 nm
f0	Photon frequency (e/h)	2.4179883E14 Hz
λc	Compton wavelength	0.00242631058 nm
rad	1 radian	1 radian
twoπ	2π radians	6.28318530718 radians
angl	∡ in trig mode	180°
c3	Wien displacement constant	0.002897756 m·K
kq	k/q	0.00008617386 J/(K·C)
ϵ0q	ϵ0/q	55263469.6 F/(m·C)
qϵ0	q*ϵ0	1.4185979E−30 F·C/m
ϵsi	Dielectric constant	11.9
ϵox	SiO_2 dielectric constant	3.9
I0	Reference intensity	0.000000000001 W/m^2

To view the Constants Library:

- Press ↰ EQ LIB COLIB CONLI.

To view the value of a particular constant in full:

1. While viewing the Constants Library, move the highlight to the desired constant. You may either use the ▲ and ▼ arrow keys or press α followed by the first character of the constant.
2. Press VALUE and UNITS (if necessary) so that the ▪ is displayed in the menu label in order to display the numerical value and units of the constant.
3. Press ENTER. If the constant value is too long to be fully displayed in a single line, then you will see it displayed more fully on a screen by itself.

To put a constant from the library on the stack:

1. Press ◀ EQ LIB COLIB CONLI to open up the Constants Library.
2. Move the highlight to the desired constant.
3. Optional: If you want the units included be sure that the ▪ is showing in the UNITS label.
4. Press →STK QUIT.

To include a constant in an algebraic expression:

1. Begin entering the algebraic into the command line.
2. Press ◀ EQ LIB COLIB CONST. You will see CONST( ) entered into the algebraic.
3. Type in the symbol for the constant. Note that the value returned by the CONST function may or may not include units depending on how you have currently set UNITS.

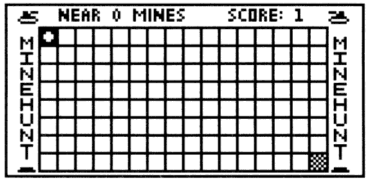
25

Minehunt Game

The Minehunt game is a battlefield adventure. You start at the upper-left corner of an 8 × 16 battlefield grid. Your mission is to travel safely to the lower-right corner, avoiding invisible mines along the way. The game tells you how many mines are under the eight squares adjacent to your position.

To play Minehunt:

- Press ◀ EQ LIB UTILS MINE.



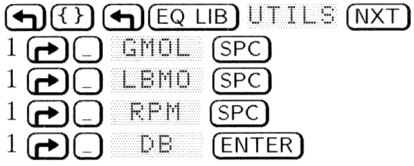
- Use the number keys or arrow keys to move. The "corner" number keys allow you to move diagonally. Exit at any time by pressing CANCEL.

User-Defined Units

The Equation Library provides four user-defined units: "gmol" (gram-moles, mol), "lbmol" (pound-moles, approximately 454 mol), "rpm" (revolutions per minute, 1/min), and "dB" (decibels, dimensionless). You can use their menu keys as typing aids. To fully use these units, add them to the custom menu. (User-defined units are described on page 10-14.)

Example: Put the Equation Library units in a custom menu.

Step 1: Enter this list of unit objects (one for each unit): `{1_gmol 1_lbmol 1_rpm 1_dB}`.



Step 2: Store this in the custom menu and display the menu. (Custom menus are described on page 30-1.)

You get the custom menu at any time by pressing CST. The following examples show how to use the user-defined units in the custom menu:

- Press GMOL to add units to the number you're entering or to append units to the unit numerator of the object in level 1.
- Press → GMOL to append units to the unit denominator of the object in level 1.
- Press ← GMOL to convert the unit object in level 1 to "gmol."

26

Time Management

Using the Clock (Date and Time)

When you display the clock, it appears in the upper-right corner of the display. It shows the current date and time in your choice of formats, shown in the table below. The formats also determine the way you enter dates and times in the command line. The following table illustrates how the clock shows 4:31:04 PM on February 21, 1994.

Clock Display	Format	Number Form
Date:		
02/21/1994	Month/day/year format	2.211994
21.02.1994	Day.month.year format	21.021994
Time:		
04:31:04P	12-hour format	16.3104
16:31:04	24-hour format	16.3104

To display the date and time:

1. Press ↗ (MODES).
2. Move the highlight to the CLOCK field and press ✓CHK to put a check-mark in the field.
3. Press OK.

To change the date or time:

1. Press ↱ TIME ▲ OK to open the SET TIME AND DATE dialog.

2. Highlight the hours field and type the hour, minutes and seconds, pressing ENTER after each.
3. Optional. If you want the clock to report the time using 24-hour format, press +/- until 24-hr is displayed.
4. Highlight the first date field and type the day, month and year in their respective fields, pressing ENTER after each. The year can be from 1991 through 2090.
5. Optional. If you want the clock to report the date using day-month-year format, press +/- until D.M.Y is displayed.
6. Press OK to confirm the changes and return to the stack.

Setting Alarms

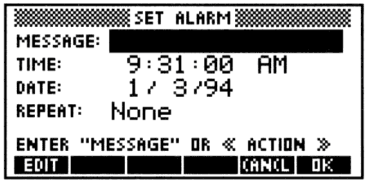
You can set two types of alarms, which perform different actions when they come due:

- **Appointment alarm.** This displays the message you specified when you set the alarm. It also sounds a sequence of beeps for about 15 seconds—or until you press a key. You're expected to acknowledge an appointment alarm after it comes due.
- **Control alarm.** This executes the program or other object you specified when you set the alarm—no other action occurs. You don't acknowledge a control alarm.

When you set an alarm, it's saved in the system alarm list, where it can be reviewed and edited ("browsed") later.

To set an appointment alarm:

1. Press ▶ TIME ▼ OK to open the SET ALARM dialog.



2. Press ▶ " ", type in the message you want displayed when the alarm comes due and press ENTER.
3. Enter the time (hours, minutes, seconds) and the format (AM, PM, or 24-hr) of the alarm.
4. Enter the date of the alarm. It is shown in the current date format (either D.M.Y or M/D/Y).
5. Highlight the REPEAT field and enter the number and period of the repeat interval. For example, press 15 ENTER α D to enter 15 Days as the repeat interval. Press 0 ENTER for a non-repeating alarm.
6. Press OK to set the alarm and return to the stack.

To set a control alarm:

1. Press ▶ TIME ▼ OK to open the SET ALARM dialog.
2. Enter the program or other object you wish to execute when the alarm comes due into the MESSAGE: field.
3. Enter the time and date for the alarm.
4. Highlight the REPEAT field and enter the number and period of the repeat interval. Press 0 ENTER for a non-repeating alarm.
5. Press OK to set the alarm.

Responding to Alarms

When an appointment alarm comes due, the ((•)) annunciator turns on, the beeper sounds at short intervals for about 15 seconds, and the alarm message is displayed. If you press a key during the beeps, the alarm is acknowledged, and the current appointment is deleted.

If you don't acknowledge an alarm during the beeps, the beeper stops and the message is cleared from the display. A repeating alarm is normally automatically rescheduled. A nonrepeating alarm becomes "past due," but not deleted—the ((•)) annunciator remains on to show you have a past-due alarm to respond to.

If you have several past-due alarms, you can view them by pressing (→)(TIME) OK . Press PURG to delete an alarm. Each time you press (←)(TIME) ALRM ACK , the oldest past-due alarm is deleted. The ((•)) annunciator turns off when no past-due alarms remain.

To respond to an appointment alarm:

- While the alarm is beeping, press any key, such as (CANCEL).
  or
- After the beeping stops, the message appears. Press (←)(TIME) ALRM ACK . (You can then press (CANCEL) to return to the stack.)

26

To respond to a control alarm:

- Do nothing. You don't acknowledge a control alarm when it comes due—it's automatically considered to be acknowledged. When a control alarm comes due, a copy of the *alarm index* is returned to level 1, then the specified object is executed. The alarm index is a real number that identifies the alarm based on its chronological order in the system alarm list—you can use it with programmable alarm commands (described in the *HP 48G Series Advanced User's Reference*).

To acknowledge all past-due alarms at once:

- Press (←)(TIME) ALAR ACKA .

It's possible for a repeating alarm to have a short enough repeat interval that it reschedules and executes faster than you can delete it from the alarm list. This may occur if you mistakenly set a repeating appointment alarm for a very short interval. It may also occur in the

case of a control alarm that executes a program to take measurements at short intervals.

To recover from a short-interval repeating alarm:

- Press the ON and 4 keys simultaneously, then release them. This keystroke sets a state in the calculator that cancels the rescheduling of the *next* due alarm (presumably the short-interval repeat alarm). When that alarm comes due—or when you press the next key—the special "no-reschedule" state of the calculator is canceled so future alarms aren't affected. Because pressing a key cancels the "no-reschedule" state, you should wait until the alarm comes due before pressing any keys.

To save or not save non-repeating alarms you acknowledge:

- To delete alarms when they're acknowledged, press 44 +/- ◀ MODES FLAG CF. This is the default setting. All control alarms (nonrepeating or repeating) that come due are always *saved* in the system alarm list, regardless of the setting of Flag −44.
- To save alarms when they're acknowledged, press 44 +/- ◀ MODES FLAG SF. Past-due occurrences of repeating alarms are not saved.

Viewing and Editing Alarms

To view, edit, or delete an alarm:

- Open the Browse alarms... dialog box. ▶ TIME OK All existing alarms are displayed.
- To edit an alarm, highlight the alarm you want to edit and press EDIT.
- To delete an alarm, highlight the alarm you want to delete and press PURG.

Use NEW to create multiple alarms (after you create an alarm you return to the ALARMS dialog).

To change the way repeating alarms work:

- To automatically delete and reschedule them, press 43 [+/−] [↰] [MODES] FLAG CF .
- To make them past-due and not reschedule them, press 43 [+/−] [↰] [MODES] FLAG SF .

To control the alarm beeper:

- To enable the alarm beeper, press 57 [+/−] [↰] [MODES] FLAG CF .
- To suppress the alarm beeper, press 57 [+/−] [↰] [MODES] FLAG SF .

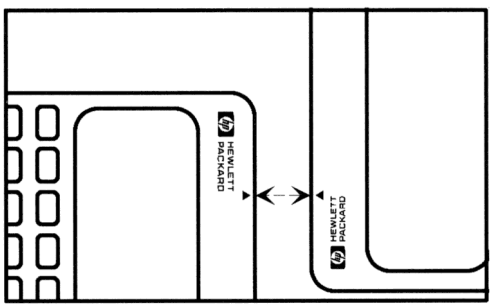
27

Transmitting and Printing Data

Transferring Data Between Two HP 48s

To transfer objects from one HP 48 to another:

1. Line up the infrared ports by lining up the ▲ marks (near the Hewlett-Packard logo just above the display). The calculators should be no farther apart than 2 inches.



2. **Receiver.**
   a. Change to the directory where the objects are to be stored.
   b. Press → I/O.
   c. Select `Get from HP 48` from the menu and press `OK`.
3. **Sender.**
   a. Press → I/O.
   b. Select `Send to HP 48...` from the menu and press `OK`.
   c. Press `CHOOS` and select the names of the objects to be transferred into the `NAME` field. Press `OK`.
   d. Press `SEND`.

Printing

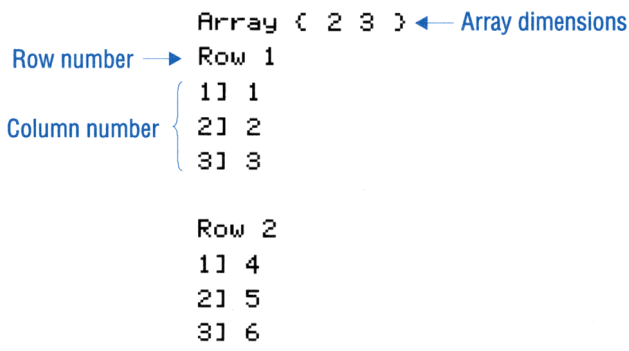
With certain exceptions, printing commands print objects according to these guidelines:

- An object is printed with its delimiters.
- An object that doesn't fit in one line of output continues on the following lines.
- An array object is printed in expanded form.

  When you print an array in expanded form, each row and column is labeled. For example, the 2 × 3 array

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

  is printed like this:



- A graphics object is printed in its stack form.

You can perform any printing operation with any compatible printer—*with these exceptions*:

- Special characters in the HP 48 character set may not print properly on a serial printer.
- You can't print a graphics object or the display on a serial printer.

Setting Up the Printer

To set up the HP 82240B infrared printer:

1. Place the HP 48 and the printer on a flat surface. Aim the ▲ (near the Hewlett-Packard logo just above the display) toward the window on the printer. Keep them within 18 inches (45 centimeters).
2. Press 34 [+/-] [←] [MODES] FLAG CF to make sure flag −34 is clear (its default state).
3. If you previously pressed OLDPR for any reason, reset the variable *PRTPAR*—press [←] [I/O] PRINT PRTPA RESET.

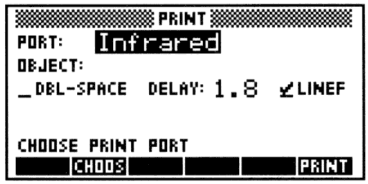
To set up a serial printer:

1. Connect the 9-pin end of an HP 48 serial cable to the serial printer. If necessary, use a 9-pin to 25-pin adapter.
2. Keep the HP logo on the 4-pin connector facing up, then plug the cable into the HP 48. You should feel it lightly snap into place.
3. If your printer uses XON/XOFF handshaking: Press [←] [I/O] [NXT] SERIA OPENI [NXT] I/O CLOSE to create *IOPAR*. Then press [VAR] [→] IOPAR [←] [EDIT] and change the fourth number to 1—for example, { 9600 0 0 1 3 1 }. Press [ENTER] [←] IOPAR.
4. If your printer requires an end-of-line sequence other than carriage-return/line-feed, press [←] [I/O] PR1 to create *PRTPAR*, then edit the *end-of-line* parameter (the fourth element in the *PRTPAR* list).

Printing Tasks

To print an object via infrared:

1. Be sure that the printer is setup properly and turned on.
2. Press [→] [I/O] [▼] [▼] [▼] OK to open the PRINT form.
3. If necessary, press [▲] [+/-] to change the communications port to Infrared.



The Infrared PRINT Screen

4. Press CHOOS, move the highlight to a variable you wish to print and press OK.
5. Optional: Adjust any of the printing parameters that you wish.

   DBL-SPACE Put a check-mark to produce double-spaced output.

   DELAY: Enter the number of seconds (cannot be more than 6.9) the HP 48 waits between sending lines of information to an infrared printer. To optimize printing efficiency, set the delay just longer than the time the printhead requires to print one line of information (default is 1.8 seconds).

   LINEF Normally (check-mark *on*), each print command completes data transmission by *automatically* executing the CR (*carriage right*) command, which tells the printer to print the data currently in its buffer and leave the print head at the right end of the print line. Alternatively (check-mark *off*), you can suppress the automatic CR command and accumulate several print commands in the print buffer, which are then printed only when you *manually* execute CR (◀ I/O PRINT CR).
6. Press PRINT.

To print a variable via the Serial Interface cable:

1. Be sure that the printer is set up properly and that the cable is properly connected.
2. Press ▶ I/O ▼ ▼ ▼ OK to open the PRINT form.

The Wire PRINT Screen

3. If necessary, press ▲ +/- to change the communications port to `Wire`.
4. Press `CHOOS`, move the highlight to a variable you wish to print and press `OK`.
5. Optional: Adjust any of the printing parameters that you wish.

`DBL-SPACE`	Put a check-mark to produce double-spaced output.
`XLAT:`	Choose which of four different character translation options you wish to use. See page 27-15 for more details about these options.
`LINEF`	Normally (check-mark *on*), each print command completes data transmission by *automatically* executing the CR (*carriage right*) command, which tells the printer to do a carriage-return/line-feed. Then the printer prints the data currently in its buffer. Alternatively (check-mark *off*), you can suppress the automatic CR command and accumulate several print commands in the print buffer, which are then printed only when you *manually* execute CR (↶ I/O `PRINT` `CR`).
`BAUD:`	Enter or choose the transfer speed. The printer and HP 48 must have the same setting.
`PARITY:`	Enter or choose the parity setting for the transfer. The printer and HP 48 must have the same setting.
`LEN:`	Enter the printer's line length (in characters).

6. Press `PRINT`.

To print the object in level 1:

1. Be sure that printer and HP 48 are properly set up for printing.
2. If the port and print parameters are set properly, press ◀ I/O PR1 .
3. If you need to change the port and print parameters, then:
   a. Press ▶ I/O ▼ ▼ ▼ OK .
   b. Press NXT CALC OK to enter the object on level 1 of the stack.
   c. Set the port and print parameters as necessary (see the previous two procedures for more details).
   d. Press PRINT.

To print the current display image:

1. Be sure that printer and HP 48 are properly set up for printing.
2. Press ▶ I/O ▼ ▼ OK . The display you see after pressing OK is the display that was printed. (If the printing does not occur, you may need to change the communications port or other print parameters and try again.)
3. If you cannot select Print display without changing or altering the display you wish to print, prepare the display the way you wish and press and hold down ON, press and release 1, and release ON.

To print all objects on the stack:

- Press ◀ I/O PRINT PRST .

To print a group of variables:

1. Set the port and print parameters as needed.
2. Put the list of variables on level 1 of the stack.
3. Press ◀ I/O PRINT PRVAR. Both the name and contents of each variable will be printed.

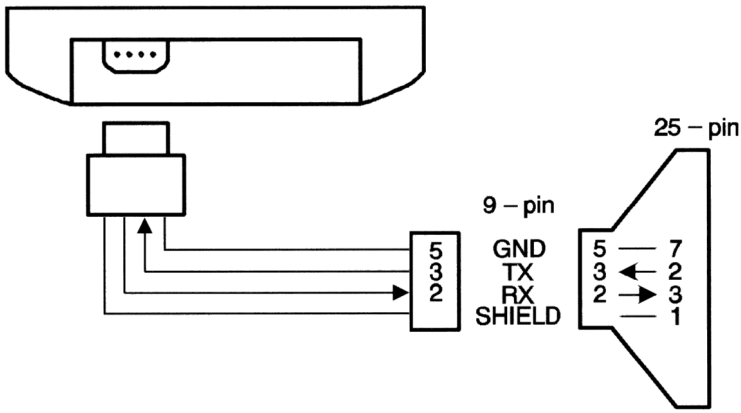
Transferring Data Between the HP 48 and a Computer

Preparing the Computer and the HP 48

Use a Serial Interface Cable to connect the HP 48 and the computer. The cable is included with the Serial Interface Kit, available from Hewlett-Packard. (For information about these products, see your HP dealer.)

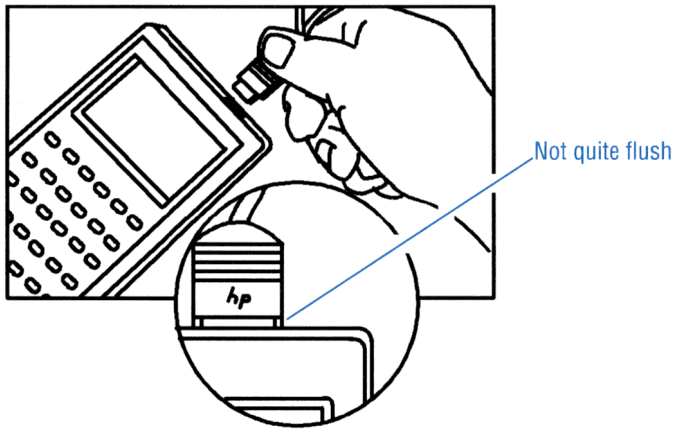
To connect a computer and HP 48:

1. Connect the computer end of the serial cable to the serial port on the computer. If necessary, use a connector adapter. (The following diagram shows the wiring used by the PC version of the Serial Interface Cable and its adapter. If you need more information, consult your computer documentation.)



Serial Interface Cable and Adapter (PC Version)

2. Keep the HP logo on the 4-pin connector facing up, then plug the cable into the HP 48. You should feel it lightly snap into place.



To set up for HP 48-to-computer transfers:

1. Make sure that the serial cable is properly connected to the computer and to the HP 48.
2. **HP 48**. Press → I/O ▲ ▲ OK to view the TRANSFER form.

27

The Wire TRANSFER Screen

3. **HP 48** Set the I/O parameters:

PORT:	Select the communications port (usually Wire for transfers between the HP 48 and a computer).
TYPE:	Select the transfer protocol, either Kermit or XModem.
FMT:	Select the transfer format, either ASCII or Binary (only available for Kermit).

XLAT:	Select one of four character translation options (only available for Kermit). See page 27-15 for more details.
CHK:	Select one of three error detection protocols (only available for Kermit).
BAUD:	Enter or choose the transfer speed. The setting must match that of the computer.
PARITY:	Enter or choose the transfer parity (only available for Kermit). The setting must match that of the computer.
OVRW	Put a check-mark here if you want received objects to overwrite objects with the same name. When unchecked, name conflicts are resolved by adding number extensions to the incoming object.

Using Kermit

By default, the HP 48 uses *Kermit* file transfer protocol to transfer data and to correct transmission errors. The HP 48 also provides commands for XMODEM and other non-Kermit serial data transfers, such as sending data to a serial printer or instrument. Kermit protocol was developed at the Columbia University Center for Computing Activities and is available for many computers.

Transferring Variables with Kermit

To transfer data to and from a computer using Kermit, the computer must be running a program that has Kermit protocol. If you want additional information on Kermit protocol, the following books are available or can be ordered in many bookstores: *Using MS-DOS Kermit* by Christine M. Gianone, Digital Press, 1990, and *KERMIT, A File Transfer Protocol* by Frank da Cruz, Digital Press, 1987.

To transfer variables from the HP 48 to a computer:

1. **Computer.** Change to the directory where the objects are to be stored.
2. **Computer.** Run the program that has Kermit. Set the transfer format to Binary or ASCII to match the current HP 48 setting.

Binary is much faster, but if you want to edit objects on the computer use ASCII.

3. **Computer.** Execute the Kermit command to make it the server, such as SERVER.
4. **HP 48.** Press → I/O ▲ ▲ OK .
5. **HP 48.** Enter or choose the *names* of the HP 48 variable or variables to be transferred. Press CHOOS OK to browse the current directory, put a check-mark next to each variable you wish to transfer to the computer. If necessary, move to a different directory to choose variables (although you may only transfer variables from a single directory at a time). Press OK to enter the list of names in the TRANSFER form.
6. **HP 48.** Make sure that the I/O parameters are properly set up for the transfer (see page 27-8 for details).
7. **HP 48.** Press SEND .
8. **HP 48.** Terminate server mode by pressing ← I/O SRVR FINIS.

To transfer files from a computer to the HP 48 using the HP 48:

1. **Computer.** Run the program that has Kermit. Set the transfer format to Binary or ASCII to match the current HP 48 setting.
2. **Computer.** Execute the Kermit command to make it the server, such as SERVER.
3. **HP 48.** Press → I/O ▲ ▲ OK .
4. **HP 48.** Press CHOOS ▼ OK to bring up a listing of the computer's current directory. (Note that this will only work for PC-compatible computers.) Select the files you wish to transfer by placing check-marks next to their names. You may switch directories by pressing CHOOS, just as with the Variable Browser, if the files are located in another directory. Press OK after all files are selected to return the list to the NAME: field in the TRANSFER form.
5. **HP 48.** Make sure that the I/O parameters are properly set up for the transfer (see page 27-8 for details).
6. **HP 48.** Press KGET .
7. **HP 48.** Terminate server mode by pressing ← I/O SRVR FINIS.

To transfer a file to the HP 48 from a computer using the computer:

1. **Computer.** Change to the directory where the files are to be stored.
2. **Computer.** Run the program that has Kermit.
3. **HP 48.** Press → I/O ▲ ▲ OK .
4. **HP 48.** Make sure that the I/O parameters are properly set up for the transfer (see page 27-8 for details).
5. **HP 48.** Press RECV .
6. **Computer.** Execute the Kermit command to send the file, such as SEND *file*.
7. Optional: To transfer additional files, repeat steps 5 and 6.
8. **Computer.** To end the session, execute the Kermit command to shut down the server, such as FINISH.

Choosing and Using File Names

The naming conventions for computer files are different from those for HP 48 variables.

When the HP 48 *receives* a file from a computer, certain difficulties may arise due to the computer file name.

- If the file name contains characters not allowed in a variable name (such as AB# or «ABC»), the HP 48 terminates the transfer and sends an error message to the computer.
- If the file name matches a built-in command (such as SIN or DUP), the HP 48 appends a number extension to the name (such as SIN.1).
- If the name matches a variable name in the current directory and flag −36 is clear (to protect existing variables), a number extension is added to the name (such as NAME.1).

When the HP 48 *sends* a variable to a computer, its name may be incompatible with the naming conventions of the computer software. Transferring such a file can result in a transfer error. (You can avoid this problem by renaming the variable before sending it.)

Backing Up HP 48 Memory

You can back up and restore the contents of the entire *HOME* directory in a file on your computer. The *HOME* directory includes all variables, user key assignments, and alarms. You can also include all flag settings if you want.

The following steps assume you've prepared the computer and HP 48 for data transfer—see "Preparing the Computer and HP 48" on page 27-7.

To back up all user memory to a computer file:

Caution

While backing up memory, make sure the ticking clock is not in the display. If the clock is in the display, it may corrupt the backup data.

1. **Computer.** Execute the Kermit command to set up binary transfer, if available.
2. **Computer.** Execute the Kermit command to make it the server, such as SERVER.
3. **HP 48.** Optional: To back up flag settings too, press ◀ MODES FLAG NXT RCLF, enter a flag-variable name (with ' delimiters), and press STO.
4. **HP 48.** Enter the tagged object :IO:*name* on the stack, where *name* is the name of the file to be created on the computer.
5. **HP 48.** Press ◀ MEMORY NXT ARCHI.
6. **HP 48.** To end the session, press ◀ I/O SRVR FINIS.
7. **HP 48.** Optional: To conserve battery power, press ◀ I/O NXT CLOSE.

ARCHIVE always uses binary transfer, regardless of the ASCII/Binary setting on the HP 48.

Caution

Use the RESTORE command with care; restoring backed up user memory completely erases current user memory and replaces it with the backup copy.

To restore HP 48 user memory from a computer file:

1. Transfer the computer file to an HP 48 variable using one of the data transfer methods discussed above. Make sure that the transfer mode is Binary.
2. **HP 48.** Enter the received variable name (with ' delimiters) onto the stack and press ▶ RCL to recall the backup object.
3. **HP 48.** Press ◀ MEMORY NXT RESTO.
4. **HP 48.** Optional: To restore flag settings previously saved, enter the flag-variable name (with ' delimiters), press ▶ RCL, and press ◀ MODES FLAG NXT STOF.

Example: To back up memory into a file named *AUG1*, enter the tagged object :IO:AUG1 as the backup name. Then, if you later retrieve this data to the HP 48, you can enter 'AUG1' and press ▶ RCL to get Backup HOMEDIR on the stack—ready for the RESTORE command.

Sending Kermit Commands

You can use an HP 48 to send Kermit commands to a Kermit server—another HP 48 or a computer. If the HP 48 is a *server*, you can send Kermit commands to it (although it only responds to GET (KGET) SEND, REMOTE DIR, REMOTE HOST, FINISH, and LOGOUT). The following steps assume the receiving device is already set up as a server.

To send a Kermit command from an HP 48:

1. Enter the command as a string (with " " delimiters).
2. Enter the packet type as a string (with " " delimiters).
3. Press ◀ I/O SRVR PKT.

The server sends one of the following responses to the PKT command:

- An acknowledging message. The reply to the packet is returned as a string to level 1—an empty string is returned if no response is appropriate.
- An error packet. The HP 48 briefly displays the contents of the error packet. To retrieve it, press ◀ I/O NXT KERR.

Example: To request a directory listing, enter "D" and "G" and press PKT. The directory is returned as a string.

Using XMODEM

The XMODEM protocol built into the HP 48 doesn't perform any CRC checking, but it will work with a computer-based XMODEM program that does. In this situation, you may have to wait a few moments before the computer's program gives up trying to perform CRC checking and reverts to plain XMODEM.

To transfer a variable to a computer using XMODEM:

1. **HP 48.** Press → I/O ▲ ▲ OK to open the TRANSFER form.
2. **HP 48.** Set the port to Wire, the type to XModem, and make sure the baud rate matches that of the computer.
3. **HP 48.** Highlight the NAME: field, press CHOOS to select a variable, and enter it.
4. **Computer.** If necessary, move to the directory where the variable is to be stored, start the XMODEM program and select Receive.
5. **Computer.** Enter the filename, and start Receive.
6. **HP 48.** Press SEND.

To transfer a variable from a computer using XMODEM:

1. **Computer.** Move to the directory where the variable is stored.
2. **Computer.** Start the XMODEM program.
3. **HP 48.** Move to the directory where you want to put the incoming variable and then press → I/O ▲ ▲ OK to open the TRANSFER form.
4. **HP 48.** Set the port to Wire, the type to XModem, and make sure the baud rate matches that of the computer.
5. **HP 48.** Type in a name for the variable to be received. Put a check-mark in the OVRW field if you want to overwrite a variable with the same name.
6. **HP 48.** Press RECV.
7. **Computer.** Start Send.

Using Other Serial Protocols

You can send and receive data and commands with serial devices that *don't* use Kermit protocol, such as serial printers and instruments. You do this using the general-purpose serial I/O commands.

To review the current HP 48 I/O parameters:

- Press ← (I/O) IOPAR. If the parameters aren't displayed, press (NXT) INFO.

To change HP 48 I/O parameters:

1. Type in -58 and press ← (MODES) FLAGS CF. This makes the current settings visible while you change them.
2. Press ← (I/O) IOPAR
3. Change the desired parameter or parameters as follows:
   - Press IR/W to select IR or Wire as the current communications port.
   - Type in either 1200, 2400, 4800, or 9600 and press BAUD to select the current transfer speed.
   - Type in either 1 (odd), 2 (even), 3 (mark), 4 (space), or 0 (none) and press PARIT to select the current parity setting. Optionally, you may enter the negative of any of these options if you wish to use the parity setting only to transmit and disable parity-checking on receive.
   - If you are using ASCII transfer or printing, type in the number of the translation option you desire (see table below) and press TRAN. In the following table, "10 → 10,13" would be read "character 10 is translated into characters 10 and 13." Selecting 0 means you do not want any translation.

Summary of ASCII Data Translation Options

Option 1	Option 2	Option 3
Data Sent by HP 48		
10 → 10,13	10 → 10,13 \ → \\ 128 → *trans* ⋮ 159 → *trans*	10 → 10,13 \ → \\ 128 → *trans* ⋮ 255 → *trans*
Data Received by HP 48		
10,13 → 10	10,13 → 10 \\ → \ *trans* → *char* \000 → *char* ⋮ \159 → *char*	10,13 → 10 \\ → \ *trans* → *char* \000 → *char* ⋮ \255 → *char*

ASCII Character Translations (Character Codes 128-255)

HP 48 Code	HP 48 Char	Trans	HP 48 Code	HP 48 Char	Trans	HP 48 Code	HP 48 Char	Trans
128	∡	\<)	142	←	\<-	156	∏	\PI
129	x̄	\x-	143	↓	\\|	157	Ω	\GW
130	▽	\.V	144	↑	\\|	158	▓	\[]
131	∫	\U/	145	γ	\Gg	159	∞	\oo
132	∫	\.S	146	δ	\Gd	171	«	\<<
133	Σ	\GS	147	∈	\Ge	176	°	\^o
134	▶	\\|	148	η	\Gn	181	μ	\Gm
135	π	\pi	149	θ	\Gh	187	»	\>>
136	∂	\.d	150	λ	\Gl	215	×	\.x
137	≤	\<=	151	ρ	\Gr	216	Ø	\O/
138	≥	\>=	152	σ	\Gs	223	β	\Gb
139	≠	\=/	153	τ	\Gt	247	÷	\:-
140	α	\Ga	154	ω	\Gw	*nnn*	other	*nnn*
141	→	\->	155	Δ	\GD			

To transfer serial data with a non-Kermit serial device:

1. Press ◀ I/O IOPAR and set up the I/O parameters to match the serial device. If necessary, press NXT INFO to see the current settings.
2. If the serial device uses receive or transmit pacing (XON/XOFF signals) during transfers, press IOPAR ◀ EDIT:
   - To receive data using pacing, change the third number to 1.
   - To send data using pacing, change the fourth number to 1—for example, { 9600 0 0 1 3 1 }. Press ENTER ' IOPAR STO.
3. Optional: Press I/O NXT SERIA OPENI to open the HP 48 serial port. (This step isn't necessary for most connections, but it prevents difficulties caused by the inability of certain devices to communicate with a closed port.)
4. To send or receive serial data or commands, use the I/O menu keys for the desired operations—see the table below.

The I/O Menu—Serial I/O Commands

Key	Programmable Command	Description
◀ I/O SERIA:		
XMIT	XMIT	Sends the string in level 1 without Kermit protocol. After the entire string is sent, 1 is returned to level 1. If the entire string failed to transmit, 0 is returned to level 1 and the unsent part of the input string is returned to level 2—execute ERRM to see the error message.
SRECV	SRECV	Receives the number of characters specified in level 1. For a successful transfer, the characters are returned to level 2 as a string, and 1 is returned to level 1. For an unsuccessful transfer, an empty or incomplete string is returned to level 2, and 0 is returned to level 1—execute ERRM to return the error message. (An unsuccessful transfer occurs if the characters contain a parity error, framing error, or overrun error, or if fewer than the specified number of characters are received before the timeout period expires, 10 seconds by default.) Characters are taken from the input buffer—no waiting occurs if you specify the number of characters in the buffer, which is returned by BUFLE.
STIME	STIME	Sets the serial transmit/receive timeout to the number of seconds specified in level 1. The timeout value can be from 0 to 25.4 seconds. If you specify 0, the HP 48 waits indefinitely, which could result in excessive battery drain.
SBRK	SBRK	Sends a serial BREAK signal.

The I/O Menu—Serial I/O Commands (continued)

Key	Programmable Command	Description
BUFLE	BUFLEN	Returns the *number* of characters in the input buffer to level 2, and the error status to level 1 (1=no framing error or UART overrun, or 0=framing error or UART overrun). If 0 is returned to level 1, the number of characters returned to level 2 represents the part of the data received *before* the error—you can use it to determine where the error occurred.

Note

Although XMIT, SRECV, and BUFLEN check the send and receive mechanisms, the integrity of the data isn't checked. One method to check the integrity of data transmission is for the sending device to append a checksum to the end of the data being sent, and for the receiving device to verify the checksum.

OPENIO, XMIT, SRECV, and SBRK automatically open the IR/serial port using the current values of the first four *IOPAR* parameters (baud, parity, receive pacing, and transmit pacing) and the current IR/wire setting (set using IR/W in the I/O SETUP menu). If you open the port, the input buffer can receive incoming data (up to 255 characters), even before you execute SRECV.

28

Libraries, Ports and Plug-In Cards

Port Memory and Plug-In Card Slots

Port memory, also referred to as *independent* memory, is structured differently than user memory:

- User memory can be subdivided into directories; port memory cannot.
- Variables (global) in user memory are active and can move around physically in memory. Variables (port) in port memory are inactive and maintain a consistent physical location in memory.

Port memory contains two kinds of objects:

- **Backup objects.** Backup objects are regular objects converted to an "inactive" form suitable for port memory.
- **Libraries.** Libraries are collections of named objects that act to extend the built-in command set. They must be stored in port memory and *attached* to a user directory in order to be usable. You can *execute* a named object from a library but not *view* or *edit* it, much as you can use a built-in command, but not edit it.

To display the menu of objects stored in a port:

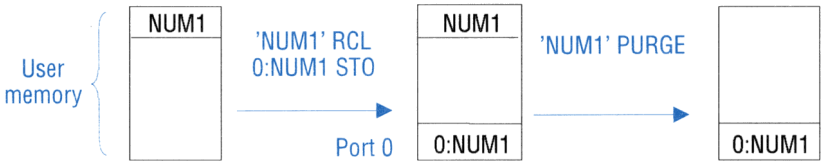
1. Press ← LIBRARY PORTS.
2. Press the menu key associated with the port you wish to view.

To display the menu of libraries accessible from the current directory:

- Press → LIBRARY.

Port 0

Port 0 is the only port memory available on all HP 48s. The memory for port 0 is taken out of user memory—so objects stored in port 0 decrease the amount of user memory available. The size of port 0 is dynamic—it grows and shrinks to accommodate its contents. This diagram shows port 0 taking memory as necessary from user memory.



If you don't have or don't want to use plug-in cards, you can use port 0 for storing backup objects and library objects.

Card Slot 1

The HP 48GX has two card slots. These card slots are *not* identical. Card Slot 1 can accept a plug-in card no larger than 128 KBytes. Any RAM memory on a card plugged into Card Slot 1 can be merged with built-in user memory to expand the amount of active memory available or it can remain as regular port memory. Card Slot 1 is identical to the card slots available on its predecessor, the HP 48SX. When port memory is used in Card Slot 1, it is designated as Port 1. The cards plugged into Card Slot 1 can be either RAM or ROM.

Card Slot 2

Card Slot 2 can contain a plug-in card up to 4 MBytes in capacity. (This translates into 4096 KBytes, of which only 3968 are accessible.) RAM memory on a card plugged into Card Slot 2 *cannot* be merged with built-in user memory. Instead, the port memory offered in Card Slot 2 is divided into distinct ports of 128 KBytes each. Thus, a 1-MByte plug-in card offers Ports 2 through 9, each containing up to 128 KBytes of backup objects and libraries. A 4-MByte plug-in card offers Ports 2 through 32. The cards plugged into Card Slot 2 can be either RAM or ROM.

Using Backup Objects

The HP 48 uses a special object type, the *backup object*, to store backup data. A backup object contains another object, its name, and its checksum.

Backup objects can exist only in *port memory*:

- Port 0.
- Port 1 if it contains RAM cards set up as port memory (that is, not merged). When you first install a card, it's set up as port memory. (Port 1 doesn't exist in the HP 48G.)
- Ports 2 through 32 if they exist. (They don't exist in the HP 48G).

To back up an object onto a card:

1. Put the object on the stack.
2. Enter a *backup identifier* for the backup object to create—see below.
3. Press STO.
4. Optional: Purge the original object in user memory.

The STO command creates the backup copy using the port and name specified by the *backup identifier*—it has the form

:*port*:*name*

where *port* is the port number (0 through 32), and *name* is the name where the backup copy is stored. If you use port 1, it must not be merged with user memory. The name of the backup object can be different from the original name.

You can back up an entire directory (and its subdirectories) in one backup object by putting the directory object on the stack and making a backup copy.

To recall a port object to the stack:

- Display the appropriate PORT menu, then press → and the menu key for the object.

  or
- Enter the backup identifier for the backup object and press → RCL.

To evaluate a backup object:

- Display the appropriate PORT menu, then press the menu key for the object.

  or

- Enter the backup identifier for the backup object and press EVAL.

To evaluate several backup objects in a row, enter a list (with ‹ › delimiters) containing the backup identifiers, then press EVAL for each backup object.

To delete a backup object:

- Enter the backup identifier for the backup object and press ← PURG. You can't delete a backup object that you recalled to the stack—you get the `Object in Use` message. If you delete the object from the stack or store the object in a variable, then you can delete the backup object.

To purge several backup objects simultaneously:

1. Enter a list (with ‹ › delimiters) containing the backup identifiers.
2. Press ← PURG.

To search all ports for a backup object:

28

1. Enter the backup identifier for the object—*except* use & for the port number. (Press α ← ENTER to type &.)
2. Execute RCL, EVAL, or PURGE. Whenever you use the & "wildcard" character for the port number, the HP 48 searches ports in reverse numerical order beginning with the highest available (32, 31, . . . , 2, 1, 0) and then main memory for the backup object—it uses the first occurrence of the name.

Example: If you enter `:&:BPG1` and press ← PURG, you delete the first occurrence of *BPG1* in port 32, 31, . . . , 2, 1, 0, or main memory.

To get a list of backup objects in a port:

- Enter the port number and press ← LIBRARY `PVARS`. The PVARS command actually returns two results. Level 1 indicates the type of memory contained in the port: `"ROM"` (application card), `"SYSRAM"` (merged memory), or a number (the number of available bytes in user memory for port 0, or in the port's independent

memory for another port). Level 2 contains a list of backup identifiers and library identifiers.

To copy backup objects from a card into another HP 48:

1. Turn off the HP 48 and install the card—see "Installing and Removing Plug-In Cards" on page 28-9.
2. Turn on the HP 48.
3. Recall the object to the stack—see "To recall a port object to the stack" on page 28-3.

You can also transfer objects between two HP 48s using their infrared ports—see "Transferring Data between Two HP 48s" on page 27-1.

Backing Up All Memory

You can back up and restore the contents of the entire *HOME* directory in a backup object. The *HOME* directory includes all variables, user key assignments, and alarms. You can also include all flag settings if you want.

You can also back up memory in a computer file. See "Backing Up HP 48 Memory" on page 27-12.

Caution

While backing up memory, make sure the ticking clock is not in the display. If the clock is in the display, it may corrupt the backup data.

To back up all of user memory in a backup object:

1. Optional: To back up flags settings too, press ◀ MODES FLAG NXT RCLF, enter a variable name (with ' delimiters), and press STO.
2. Enter a backup specifier for the backup object to create.
3. Press ◀ MEMORY NXT ARCHI.

ARCHIVE backs up only user memory—it does *not* back up independent memory.

Caution

Executing RESTORE *overwrites* the entire contents of user memory with the contents of the backup object. To save the stack, you can save it in another backup object.

To restore HP 48 user memory from a backup object:

1. Enter the backup object's identifier (with :: delimiters) onto the stack. (Remember the name includes a port number.)
2. Press ◀ MEMORY NXT RESTO.
3. Optional: To restore flag settings previously saved, recall the contents of the variable containing the flag data and press ◀ MODES FLAG NXT STOF.

Using Libraries

A *library* is an object that contains named objects that can act as extensions to the built-in command set. The primary use of a library is to serve as a vehicle for a ROM- or RAM-based application. A ROM-based library resides in a plug-in application card and is installed by inserting the card into one of the card slots. (The HP 48G has no plug-in card slots.) A RAM-based library can reside in a plug-in RAM card, or it can be transferred into user memory from the infrared or serial I/O port. (See the library's documentation for details).

Caution

Libraries originally designed for use with the older HP 48S and HP 48SX models may not be compatible with the HP 48G and HP 48GX. Loss of memory may occur. You should backup your user memory (see page 28-5) before attempting to use such libraries. Contact the vendor or author of the library for more details about compatibility.

Libraries offer several advantages over programs:

- Applications you write are protected from copying because the contents of a library can't be viewed, edited, or recalled to the stack.

- Libraries offer faster access to the variables used by applications.
- You can designate variables used in applications as "hidden" (unnamed) variables, which avoids cluttering the library's menu.

Each library is identified two ways:

- A *library identifier*, which has the form :*port*:*number*, where *number* is a unique number associated with the library. If you press ◀ LIBRARY PORTS and the menu key for the port where you stored the library, the library number appears in the menu.
- A *library name*, which is a sequence of characters. If you press ▶ LIBRARY in the directory where you attached the library or any of its subdirectories, the library name appears in the menu.

The ability to create libraries is not built into the HP 48. Normally, they are created on a computer and downloaded to the HP 48 via a cable or plug-in card. If you are interested in creating libraries, contact the HP Electronic Information Service described on the inside back cover of this manual.

To set up a library:

1. Install the library in a port:
   - For an application card library, *turn off the HP 48* and insert the card into port 1 or 2.
   - For a RAM-based library, store it in port memory.
2. Attach the library (see below). Some libraries are "auto-attaching" and some must be manually attached. You can attach only one library to each directory—*except* you can attach any number to the *HOME* directory. (See also the documentation that comes with the application card or RAM-based library for any other information about attaching the library.)

To use a library, it must be installed in a port and attached to a directory in user memory. The attachment may happen automatically when you install an application card—or you may have to do it yourself.

To store a RAM-based library in port memory:

1. Put the library object on the stack. (Notice its library number and name.)
2. Enter the port number for storing the library. If you use port 0, the library is always available, even if you remove plug-in cards. If

you use a port in one of the card slots, the appropriate slot must contain a RAM card set up as unmerged port memory.

3. Press [STO].
4. Optional: Purge the original library object in user memory, if you haven't already done so.

To attach an "auto-attaching" library to the HOME directory:

- Turn the HP 48 off and on. All auto-attaching libraries stored in port memory will attach themselves to the HOME directory (if not already attached).

To manually attach a library to a directory:

1. Change to the desired directory:
   - For access from all directories, change to the *HOME* directory.
   - For limited access, change to the desired directory. The library will be available only in this directory and its subdirectories.
2. Enter the *library identifier* for the library—it has the form :*port*:*number*.
3. Press [◀] [LIBRARY] [NXT] ATTAC.
4. Turn the HP 48 off and on.

To detach a library from a directory:

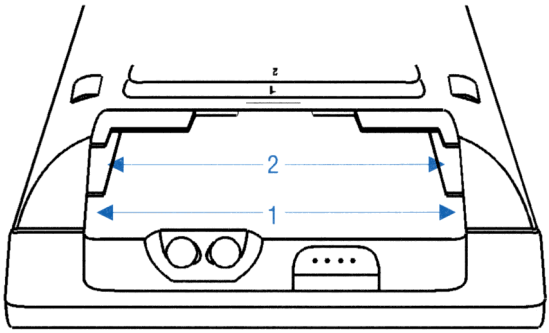
1. Change to the directory where the library is attached.
2. Enter the *library number* for the library you wish to detach.
3. Press [◀] [LIBRARY] DETAC to detach it from the directory.

To purge a library from memory:

1. Make sure that the library has been detached from every directory to which it had been attached.
2. Enter the *library identifier* for the library in independent memory—it has the form :*port*:*number*.
3. Press [◀] [PURG] to delete the library from independent memory. If you receive the `Object In Use` error, it indicates that the library is still attached to some directory.

Installing and Removing Plug-In Cards

The two slots for installing plug-in cards are designated Card Slot 1 and Card Slot 2. Card Slot 1 is closest to the front of the calculator—Card Slot 2 is closest to the back. These slots are *not identical*. See page 28-2 for details about the differences.

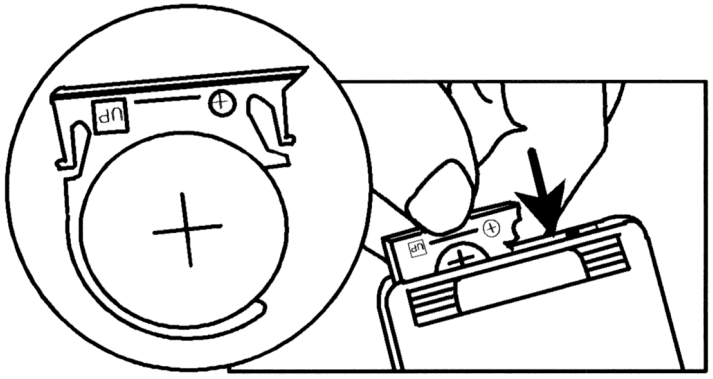


Caution

Nonapproved plug-in cards and accessories may cause damage to the HP 48. You can distinguish a potentially damaging card or plug-in accessory from an HP-approved card by looking at the back side of the card where it plugs into the HP 48. An approved card has a metal shutter to protect the HP 48 from static charges. The nonapproved cards and accessories examined to date by HP do not have this shutter, but have exposed gold contacts instead.

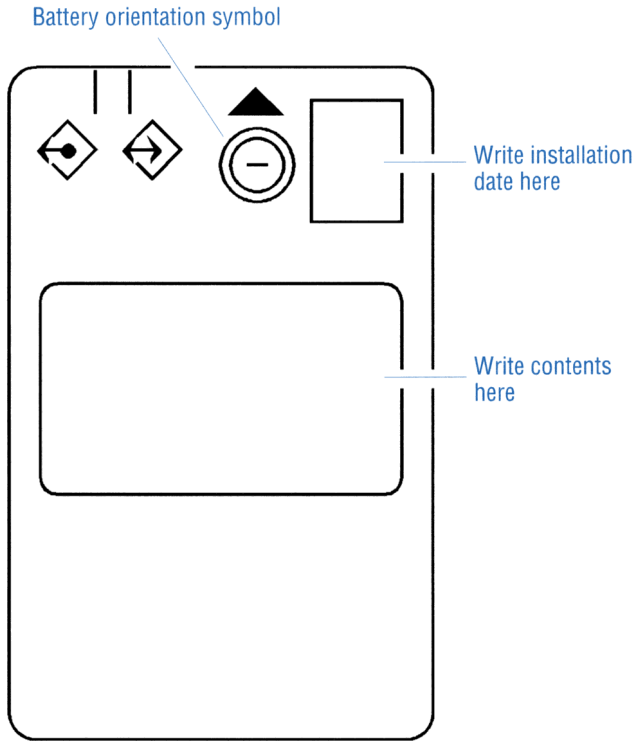
To install the battery in a new RAM card:

1. Do not use this procedure to *replace* the battery in a RAM card—it could cause loss of memory in the RAM card. To *replace* a battery, see "To change a RAM card battery" on page A-7.
2. Slide the battery holder from the card by inserting your thumbnail or a small screwdriver into the groove and pulling.
3. The grooved side of the battery holder is marked with the + symbol and the word UP. Insert the battery into the holder with its + side up, and then slide the holder into the card.



28

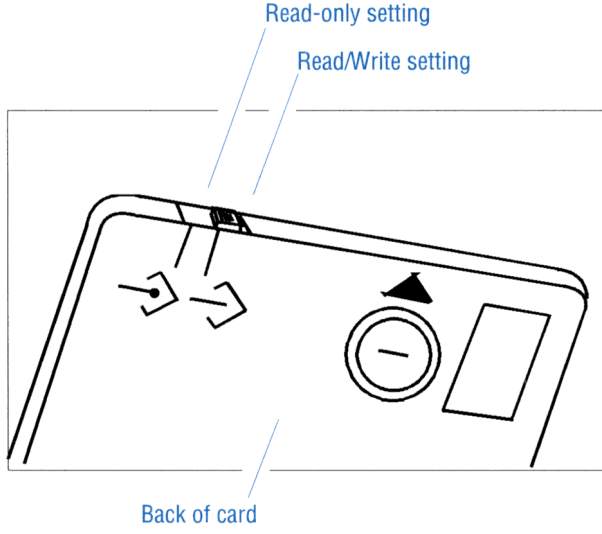
4. Write the date of installation on the card using a fine-point, permanent marker. The date is important for determining when to replace the battery.



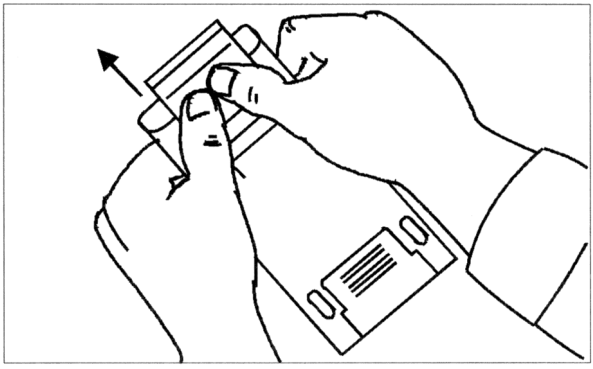
5. Set an alarm in the calculator for 1 year from the date of installation to remind you to replace the battery. (Depending on the use, the battery should last between 1 and 3 years. When the battery needs replacing, a display message will appear—*but only if the card is in the calculator.* You set this alarm to remind yourself in case the card isn't in the calculator when the battery gets low.) To set an alarm, see "Setting Alarms" on page 26-2. To replace a RAM-card battery, see "To change a RAM card battery" on page A-7.

To install a plug-in card:

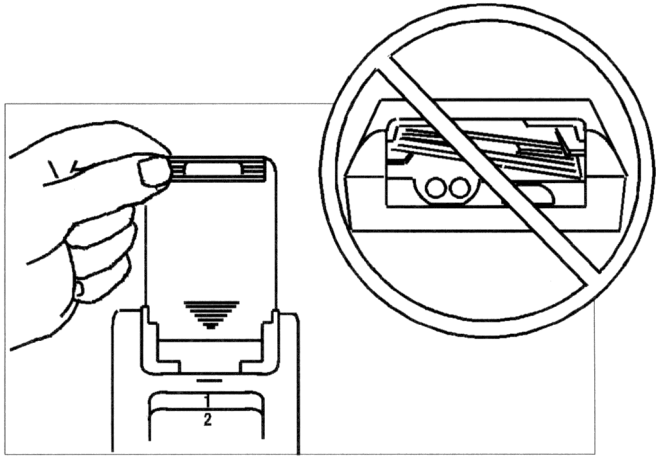
1. Store any objects currently on the stack that you want to save. (Installing or removing any plug-in card erases the stack.)
2. If the card you wish to install contains any libraries or applications designed for the older HP 48SX model, then backup all user memory as a precaution before installing the card (see page 28-5). Not all older libraries are compatible with the HP 48GX and may cause loss of user memory.
3. Turn off the calculator. Otherwise, all of user memory could be erased.
4. If the card is a *new* RAM card, install its battery (see above).
5. For a RAM card, check or set the write-protect switch. *For a new RAM card, set it to Read/Write.* (Always turn the calculator off before changing the write-protect switch.)
   - **Read Only.** You can read the contents of the card, but you can't change, erase, or store data. It protects the contents of the RAM card from being accidentally overwritten or erased. *Never* use this setting for a RAM card containing merged memory.
   - **Read/Write.** You can read, change, and erase the contents and store data, as you do with built-in user memory.



6. Remove the port cover at the top of the calculator by pressing down against the grip area and then pushing in the direction shown. Removing the cover exposes the two plug-in ports.



7. Select the empty slot for the card.
8. Position the plug-in card as shown. The triangular arrow on the card must point down, toward the calculator. Make sure the card is lined up properly with a slot opening and not positioned half in one slot and half in the other.



9. Slide the card firmly into the slot until it stops. When you first feel resistance, the card has about $\frac{1}{4}$ inch to go to be fully seated.
10. Replace the port cover by sliding it on until the latch engages.
11. Press ON to turn on the calculator.

Note

When you install a new RAM card (or one containing ports that you've never used) and turn on the calculator, you get the message `Invalid Card Data`. You may disregard the message—ports are automatically initialized the first time you use them. Or, if you prefer, press ← LIBRARY NXT `PINIT` to initialize all available RAM ports. The PINIT command will *not* affect any data currently stored in any port.

To remove a plug-in card:

Caution

Never remove a RAM card that contains *merged* memory—it will probably cause a loss of data stored in user memory. Before you remove the RAM card, you must *free* the merged memory. See page 28-15.

If you accidentally remove a card with merged memory and see the message `Replace RAM, Press ON`, you can minimize memory loss by *leaving the calculator on*, reinserting the card in the *same* port, and then pressing ON.

1. *If you're removing a RAM card from Card Slot 1, make sure it contains unmerged, free memory*—see the caution below and page 28-15.
2. Turn off the calculator. *Do not press* ON *until you've removed the card.*
3. Remove the port cover.
4. Press against the grip and slide the card out of the port.
5. Replace the port cover.

Expanding User Memory with Plug-In RAM Cards

You can extend built-in HP 48GX user memory by installing a RAM card into Card Slot 1 and merging its memory with user memory. (The HP 48G model has no plug-in card slots.)

Each RAM card contains a battery that preserves its contents while the calculator is turned off and after you've properly removed the card from the calculator. (The calculator batteries power the RAM card only while the calculator is turned on.)

You set up a RAM card as one of two types of memory—each with its own benefits. You can change between the two types—but you can't use one card as both types at the same time.

- **Merged user memory.** The part of user memory that's contained in a RAM card—the card's memory is *merged* with built-in user memory. This lets you expand the amount of user memory for creating variables and directories, and for putting objects on the stack.
- **Free port memory.** RAM memory that's *independent* of user memory—in built-in memory (in port 0) or in a RAM card (in ports 1 through 32). This lets you back up individual objects or entire directories, much as you'd back up computer files to a disk, then store it in a safe place. You can also use it to transfer data to another HP 48 by installing it and copying the objects there. See "Backing Up Data" on page 28-3.

To check the type of memory in a port:

- Enter the port number and press ◀ LIBRARY PVARS. The result in level 1 indicates the type of memory:

"ROM"	ROM in an application card.
"SYSRAM"	Merged user memory in a RAM card.
number	Free port memory in a RAM card.

To merge memory on the RAM card in Card Slot 1 with user memory:

1. Turn off the calculator and make sure the card is *not write-protected*.
2. Turn the calculator back on and press ◀ LIBRARY MERG. If the card previously contained any backup objects or libraries, the MERGE1 command automatically moves them to a special part of memory called port 0. See "Using Port 0" on page 28-2.

To free a RAM card in Card Slot 1 that's merged into user memory:

1. Press ◀ { } ENTER to enter an empty list.
2. Press ◀ LIBRARY FREE1. If the RAM card is already free (port memory), you'll get a `Port Not Available` error when you execute FREE1. If there isn't enough memory available to free the RAM card, you'll get a memory error when you execute FREE1 (see below).
3. Optional: Turn off the HP 48 and unplug the card—see "To remove a plug-in card" on page 28-14.

To check the amount of available user memory, press ◀ MEMORY MEM —the number returned is the amount of unused user memory in bytes. *To be able to free the RAM card, you must have an unused amount that's greater than or equal to the size of the RAM card*—otherwise, the HP 48 doesn't have enough unused memory to allocate to the card.

If there's insufficient user memory to free a RAM card:

- Purge unneeded variables from user memory.
- Back up data into another RAM card installed in the other card slot and then purge the original variables.
- Back up data into port 0, purge the originals, and then move the backup objects to the RAM card as it's being freed (see below).

To free a merged RAM card and move backup objects there:

1. Back up the desired objects into port 0—see "To back up an object" on page 28-3.
2. Enter a list (with ‹ › delimiters) containing the simple names of the backup objects in port 0.
3. Press ← MEMORY FREE1. The objects named in the list are removed from port 0 and stored in the newly freed RAM card (in port memory).
4. Optional: Turn off the HP 48 and unplug the card—see "To remove a plug-in card" on page 28-14.

To change the write-protect switch with the card installed:

1. *Make sure the card contains free, unmerged port memory*—see "To check the type of memory in a port" on page 28-15.
2. Turn off the HP 48.
3. Move the switch to the correct position:
   - For Read Only, the switch is *toward* the corner of the card.
   - For Read/Write, the switch is *away from* the corner of the card.

29

Programming the HP 48

This chapter is an introduction to some of the programming capabilites of the HP 48. For a comprehensive command listing and in-depth programming guide see the *HP 48G Series Advanced User's Reference* (part number 00048-90136).

Understanding Programming

An HP 48 program is an object with « » delimiters containing a sequence of numbers, commands, and other objects you want to execute automatically to perform a task.

For example, if you want to find the negative square root of a number that's in level 1, you might press [√x] [+/-]. The following program executes the same commands:


```
« √ NEG »
```

Without changing the program, we *could* show it with one command per line—similar to other programming languages:

```
«
 √
 NEG
»
```

The Contents of a Program

As mentioned above, a program contains a sequence of objects. As each object is processed in a program, the action that results depends on the type of object, as summarized below.

Actions for Certain Objects in Programs

Object	Action
Command	*Executed.*
Number	Put on the stack.
Algebraic	Put on the stack.
String	Put on the stack.
List	Put on the stack.
Program	Put on the stack.
Global name (quoted)	Put on the stack.
Global name (unquoted)	■ Program *executed.* ■ Name evaluated. ■ Directory becomes current. ■ Other object put on the stack.
Local name (quoted)	Put on the stack.
Local name (unquoted)	Contents put on the stack.

29

As you can see from this table, most types of objects are simply put on the stack—but built-in commands and programs called by name are *executed*. The following examples show the results of executing programs containing different sequences of objects.

Examples of Program Actions

Program	Results
« 1 2 »	2: 1 1: 2
« "Hello" { A B } »	2: "Hello" 1: { A B }
« '1+2' »	1: '1+2'
« '1+2' →NUM »	1: 3
« « 1 2 + » »	1: « 1 2 + »
« « 1 2 + » EVAL »	1: 3

Actually, programs can contain more than just objects—they can also contain *structures*. A structure is a program segment with a defined organization. Two basic kinds of structures are available:

- **Local variable structure.** The → command defines local variable names and a corresponding algebraic or program object that's evaluated using those variables.
- **Branching structures.** Structure words (like DO...UNTIL...END) define conditional or loop structures to control the order of execution within a program.

A *local variable structure* has one of the following organizations inside a program:

« → $name_1$ ... $name_n$ '*algebraic*' »
« → $name_1$ ... $name_n$ « *program* » »

The → command removes n objects from the stack and stores them in the named local variables. The algebraic or program object in the structure is *automatically evaluated* because it's an element of the structure—even though algebraic and program objects are put on the stack in other situations. Each time a local variable name appears in the algebraic or program object, the variable's contents are substituted.

So the following program takes two numbers from the stack and returns a numeric result:

```
« → a b 'ABS(a-b)' »
```

Calculations in a Program

Many calculations in programs take data from the stack—sometimes put there by the user or by another program. Here are two typical ways to manipulate that data:

- **Stack commands.** Operate directly on the objects on the stack.
- **Local variable structure.** Stores the stack objects in temporary local variables, then uses the variable names to represent the data in the following algebraic or program object.

Numeric calculations provide convenient examples of these methods. The following three programs use two numbers from the stack to calculate the hypotenuse of a right triangle using the formula $\sqrt{x^2+y^2}$.

```
« SQ SWAP SQ + √ »
« → x y « x SQ y SQ + √ » »
« → x y '√(x^2+y^2)' »
```

29

The first program uses stack commands to manipulate the numbers on the stack—the calculation uses stack syntax. The second program uses a local variable structure to store and retrieve the numbers—the calculation uses stack syntax. The third program also uses a local variable structure—the calculation uses algebraic syntax. Note that the underlying formula is most apparent in the third program.

Local variable structures with algebraic objects are favored by many programmers because they're easy to write, easy to read, and simple to debug.

Structured Programming

The HP 48 encourages *structured programming*. Every program has only one entrance point—the beginning of the program. It also has only one exit point—the end of the program. There are no labels inside a program to jump to—there are no GOTO commands to exit from. From an external point of view, program flow is extremely simple—start at the beginning, stop at the end. (Of course, *inside* the

program you can use branching structures to control the execution flow.)

You can take advantage of structured programming by creating "building-block" programs. Each building-block program can stand alone—and it can act like a subroutine in a larger program. For example, consider the following program:

```
« GETVALUE CALCULATE SHOWANSWER »
```

This program is separated into three main tasks, each with a subroutine. The flow is predictable. Only the input and output of each subroutine matter—the internal workings don't matter at this level.

Within each subroutine, its task can be simple—or it can be subdivided further into other subroutines that perform smaller tasks. This lets you have relatively simple subroutines—even if your main program is large.

So, programs become extensions to the set of built-in commands, as mentioned earlier. You execute them by name. They take certain inputs, and they produce certain results.

Entering and Executing Programs

A program is an object—it occupies one level on the stack, and you can store it in a variable.

To enter a program:

1. Press ◀ (« »). The PRG annunciator appears, indicating Program-entry mode is active.
2. Enter the commands and other objects (with appropriate delimiters) in the required order for the operations you want the program to execute.
   - Press SPC to separate consecutive numbers.
   - Press ▶ to move past closing delimiters.
3. Optional: Press ▶ ↵ (newline) to start a new line in the command line at any time.
4. Press ENTER to put the program on the stack.

In Program-entry mode (PRG annunciator on), command keys aren't executed—they're entered in the command line instead. Only nonprogrammable operations (such as ◀ and VAR) are executed.

Line breaks are discarded when you press ENTER.

To enter commands and other objects in a program:

- Press the keyboard or menu key for the command or object.
  or
- Type the characters using the alpha keyboard.

To store or name a program:

1. Enter the program on the stack.
2. Enter the variable name (with ' delimiters) and press STO.

To execute a program:

- Press VAR and then the menu key for the program name.
  or
- Enter the program name (with *no* delimiters) and press ENTER.
  or
- Put the program name in level 1 and press EVAL.
  or
- Put the program object in level 1 and press EVAL.

To stop an executing program:

- Press CANCEL.

Example: Enter a program that takes a radius value from the stack and calculates the volume of a sphere of radius r using

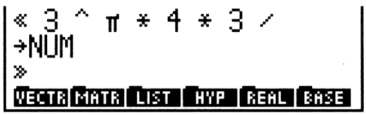
$$V = \frac{4}{3}\pi r^3$$

Step 1: If you were going to calculate the volume manually after entering the radius on the stack, you might press these keys:

3 y^x ◀ π × 4 × 3 ÷ ◀ →NUM

Enter the same keystrokes in a program. (▶ ↵ starts a new line.)

◀ «»
3 y^x ◀ π × 4 × 3 ÷
▶ ↵ ◀ →NUM



Step 2: Put the program on the stack.

ENTER

Step 3: Store the program in variable *VOL*. Then put a radius of 4 on the stack and run the *VOL* program.

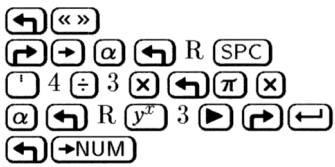
' α-(hold) VOL (release)
STO
4 VAR VOL

Example: Replace the program from the previous example with one that's easier to read. Enter a program that uses a local variable structure to calculate the volume of a sphere. The program is

```
« → r '4/3*π*r^3' →NUM »
```

(You need to include →NUM because π causes a symbolic result. Try the program with and without the →NUM command.)

Step 1: Enter the program.



Step 2: Put the program on the stack, store it in *VOL* and calculate the volume for a radius of 4 .

Viewing, Debugging, and Editing Programs

To view or edit a program:

1. View the program:
   - If the program is in level 1, press ◀ EDIT.
   - If the program is stored in a variable, put the variable name in level 1 and press ◀ EDIT.
2. Make any desired changes.
3. Press ENTER to save any changes (or press CANCEL to discard changes) and return to the stack.

It's easier to understand how a program works if you execute it step by step, observing the effect of each step. Doing this can help you "debug" your own programs or understand programs written by others.

To single-step from the start of a program:

1. Put any data required by the program on the stack on the appropriate levels.
2. Put the program or program name in level 1 (or the command line).
3. Press PRG NXT RUN DBUG to start and immediately suspend execution. The HALT annunciator is displayed in the status area.

4. Take any action:
   - To see the next program step displayed in the status area and then executed, press SST .
   - To display but *not* execute the next one or two program steps, press NEXT .
   - To continue with normal execution, press ↩ CONT .
   - To abandon further execution, press KILL .
5. Repeat step 4 as desired.

To single-step from the middle of a program:

1. Insert a HALT command in the program where you want to begin single-stepping.
2. Execute the program normally. The program stops when the HALT command is executed, and the HALT annunciator is displayed.
3. Take any action:
   - To see the next program step displayed in the status area and then executed, press SST .
   - To display but not execute the next one or two program steps, press NEXT .
   - To continue with normal execution, press ↩ CONT .
   - To abandon further execution, press KILL .
4. Repeat step 3 as desired.

When you want the program to run normally again, remove the HALT command from the program.

To single-step when the next step is a subroutine:

- To execute the subroutine in one step, press SST .
- To execute the subroutine step-by-step, press SST↓ .

SST executes the next step in a program—if the next step is a subroutine, SST executes the subroutine in one step. SST↓ works just like SST —except if the next program step is a subroutine, it single-steps to the first step in the subroutine.

To turn off the HALT annunciator at any time:

- Press PRG CTRL KILL .

Using Programming Structures

A programming structure allows a program to make a decision about how it should execute depending on given conditions or the values of particular arguments. Careful use of these structures make it possible to create programs of extraordinary flexibility.

Conditional Structures

Conditional structures let a program make a decision based on the result of one or more tests.

Here is a summary of the conditional structures available on the HP 48:

IF...THEN...END

Enter this structure into a program by pressing (PRG) BRCH (◀) IF. Its syntax is:

« ... IF *test-clause* THEN *true-clause* END ... »

IF...THEN...END executes the sequence of commands in the *true-clause* only if the *test-clause* evaluates to true. The test-clause can be a command sequence (for example, A B ≤) or an algebraic (for example, 'A≤B'). If the test-clause is an algebraic, it's *automatically evaluated* to a number—you don't need →NUM or EVAL.

IF begins the test-clause, which leaves a test result on the stack. THEN removes the test result from the stack. If the value is nonzero, the true-clause is executed—otherwise, program execution resumes following END.

IF...THEN...ELSE...END

Enter this structure into a program by pressing (PRG) BRCH (▶) IF. Its syntax is:

« ... IF *test-clause*
THEN *true-clause* ELSE *false-clause* END' ... »

IF...THEN...ELSE...END executes either the *true-clause* sequence of commands if the *test-clause* is true, or the *false-clause* sequence of commands if the *test-clause* is false. If the test-clause is an algebraic,

it's automatically evaluated to a number—you don't need →NUM or EVAL.

IF begins the test-clause, which leaves a test result on the stack. THEN removes the test result from the stack. If the value is nonzero, the true-clause is executed—otherwise, the false-clause is executed. After the appropriate clause is executed, execution resumes following END.

CASE...END

To enter CASE...END in a program:

1. Press PRG BRCH ◀ CASE to enter CASE...THEN...END...END.
2. For each additional test-clause, move the cursor after a test-clause END and press ▶ CASE to enter THEN...END.

The syntax for the CASE...END structure is:

```
CASE
   test-clause1 THEN true-clause1 END
   test-clause2 THEN true-clause2 END
   ⋮
   test-clausen THEN true-clausen END
   default-clause (optional)
END
```

This structure lets you execute a series of *test-clause* commands, then execute the appropriate *true-clause* sequence of commands. The first test that returns a true result causes execution of the corresponding true-clause, ending the CASE...END structure. Optionally, you can include after the last test a *default-clause* that's executed if all the tests evaluate to false. If a test-clause is an algebraic, it's automatically evaluated to a number—you don't need →NUM or EVAL.

When CASE is executed, test-clause$_1$ is evaluated. If the test is true, true-clause$_1$ is executed, and execution skips to END. If test-clause$_1$ is false, execution proceeds to test-clause$_2$. Execution within the CASE structure continues until a true-clause is executed, or until all the test-clauses evaluate to false. If a default clause is included and all the test-clauses evaluate to false, the default clause executes.

Loop Structures

Loop structures let a program execute a sequence of commands several times. To specify in advance how many times to repeat the loop, use a *definite loop*. To use a test to determine whether or not to repeat the loop, use an *indefinite loop*.

START...NEXT

Enter this structure into a program by pressing PRG BRCH ◀ START. Its syntax is:

« ... *start finish* START *loop-clause* NEXT ... »

START...NEXT executes the *loop-clause* sequence of commands one time for each number in the range *start* to *finish*. The loop-clause is always executed at least once.

START takes two numbers (*start* and *finish*) from the stack and stores them as the starting and ending values for a loop counter. Then, the loop-clause is executed. NEXT increments the counter by 1 and tests to see if its value is less than or equal to *finish*. If so, the loop-clause is executed again—otherwise, execution resumes following NEXT.

START...STEP

29

Enter this structure into a program by pressing PRG BRCH ▶ START. Its syntax is:

« ... *start finish* START *loop-clause*
increment STEP ... »

START...STEP executes the *loop-clause* sequence just like START...NEXT does—except that the program specifies the increment value for the counter, rather than incrementing by 1. The loop-clause is always executed at least once.

START takes two numbers (*start* and *finish*) from the stack and stores them as the starting and ending values of the loop counter. Then the loop-clause is executed. STEP takes the increment value from the stack and increments the counter by that value. If the argument of STEP is an algebraic or a name, it's automatically evaluated to a number.

The increment value can be positive or negative. If it's positive, the loop is executed again if the counter is less than or equal to *finish*. If

the increment value is negative, the loop is executed if the counter is greater than or equal to *finish*. Otherwise, execution resumes following STEP.

FOR...NEXT

Enter this structure into a program by pressing PRG BRCH ◀ FOR. Its syntax is

« ... *start finish* FOR *counter loop-clause* NEXT ... »

FOR...NEXT executes the *loop-clause* program segment one time for each number in the range *start* to *finish*, using local variable *counter* as the loop counter. You can use this variable in the loop-clause. The loop-clause is always executed at least once.

FOR takes *start* and *finish* from the stack as the beginning and ending values for the loop counter, then creates the local variable *counter* as a loop counter. Then the loop-clause is executed—*counter* can appear within the loop-clause. NEXT increments *counter-name* by one, and then tests whether its value is less than or equal to *finish*. If so, the *loop-clause* is repeated (with the new value of *counter*)—otherwise, execution resumes following NEXT. When the loop is exited, *counter* is purged.

FOR...STEP

Enter this structure by pressing PRG BRCH ▶ FOR. Its syntax is:

« ... *start finish* FOR *counter loop-clause increment* STEP ... »

FOR...STEP executes the *loop-clause* sequence just like FOR...NEXT does—except that the program specifies the increment value for *counter*, rather than incrementing by 1. The loop-clause is always executed at least once.

FOR takes *start* and *finish* from the stack as the beginning and ending values for the loop counter, then creates the local variable *counter* as a loop counter. Next, the loop-clause is executed—*counter* can appear within the loop-clause. STEP takes the increment value from the stack and increments *counter* by that value. If the argument of STEP is an algebraic or a name, it's automatically evaluated to a number.

The increment value can be positive or negative. If the increment is positive, the loop is executed again if *counter* is less than or equal to *finish*. If the increment is negative, the loop is executed if *counter* is greater than or equal to *finish*. Otherwise, *counter* is purged and execution resumes following STEP.

DO...UNTIL...END

Enter this structure into a program by pressing PRG BRCH ◀ DO. Its syntax is:

« ... DO *loop-clause* UNTIL *test-clause* END ... »

DO...UNTIL...END executes the *loop-clause* sequence repeatedly until *test-clause* returns a true (nonzero) result. Because the test-clause is executed *after* the loop-clause, the loop-clause is always executed at least once.

DO starts execution of the loop-clause. UNTIL marks the end of the loop-clause. The test-clause leaves a test result on the stack. END removes the test result from the stack. If its value is zero, the loop-clause is executed again—otherwise, execution resumes following END. If the argument of END is an algebraic or a name, it's automatically evaluated to a number.

29

WHILE...REPEAT...END

Enter this structure into a program by pressing PRG BRCH ◀ WHILE. Its syntax is:

« ... WHILE *test-clause* REPEAT *loop-clause* END ... »

WHILE...REPEAT...END repeatedly evaluates *test-clause* and executes the *loop-clause* sequence if the test is true. Because the test-clause is executed *before* the loop-clause, the loop-clause is not executed if the test is initially false.

WHILE starts execution of the test-clause, which returns a test result to the stack. REPEAT takes the value from the stack. If the value is nonzero, execution continues with the loop-clause—otherwise, execution resumes following END. If the argument of REPEAT is an algebraic or a name, it's automatically evaluated to a number.

Error Trapping Structures

Many conditions are automatically recognized by the HP 48 as error conditions—and they're automatically treated as errors in programs. A command with an improper argument or an improper number of arguments causes an error in a program. An out-of-range result can cause an error. An invalid calculator condition can cause an error.

Error trapping structures allow programs to intercept (or *trap*) error conditions that otherwise would cause the program to abort.

IFERR...THEN...END

Enter this structure into a program by pressing PRG NXT ERROR ◀ IFERR. Its syntax is:

« ... IFERR *trap-clause* THEN *error-clause* END ... »

The commands in the *error-clause* are executed only if an error is generated during execution of the *trap-clause*. If an error occurs in the trap-clause, the error is ignored, the remainder of the trap-clause is skipped, and program execution jumps to the error-clause. If *no* errors occur in the trap-clause, the error-clause is skipped and execution resumes after the END command.

IFERR...THEN...ELSE...END

Enter this structure into a program by pressing PRG NXT ERROR ▶ IFERR. Its syntax is:

« ... IFERR *trap-clause*
THEN *error-clause* ELSE *normal-clause* END ... »

The commands in the *error-clause* are executed only if an error is generated during execution of the *trap-clause*. If an error occurs in the trap-clause, the error is ignored, the remainder of the trap-clause is skipped, and program execution jumps to the error-clause. If *no* errors occur in the trap-clause, execution jumps to the *normal-clause* at the completion of the trap-clause.

Using Local Variables

There are disadvantages to using global variables in programs:

- After program execution, global variables that you no longer need to use must be purged if you want to clear the VAR menu and free user memory.
- You must explicitly store data in global variables prior to program execution, or have the program execute STO.

Local variables address the disadvantages of global variables in programs. Local variables are temporary variables *created by a program*. They exist only while the program is being executed and cannot be used outside the program. They never appear in the VAR menu. In addition, local variables are accessed faster than global variables. (By convention, this manual uses lowercase names for local variables.)

Creating Local Variables

In a program, a *local variable structure* creates local variables.

To enter a local variable structure in a program:

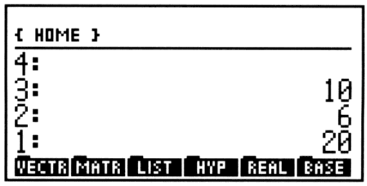
1. Enter the → command (press ▶ →).
2. Enter one or more variable names.
3. Enter a *defining procedure* (an algebraic or program object) that uses the names.

« → $name_1$ $name_2$ ... $name_n$ '*algebraic*' »

or

« → $name_1$ $name_2$ ... $name_n$ « *program* » »

When the → command is executed in a program, n values are taken from the stack and assigned to variables $name_1$, $name_2$, ... $name_n$. For example, if the stack looks like this:



then

- → a creates local variable $a = 20$.
- → a b creates local variables $a = 6$ and $b = 20$.
- → a b c creates local variables $a = 10$, $b = 6$, and $c = 20$.

The defining procedure then uses the local variables to do calculations.

Local variable structures have these advantages:

- The → command stores the values from the stack in the corresponding variables—you don't need to explicitly execute STO.
- Local variables automatically disappear when the defining procedure for which they are created has completed execution. Consequently, local variables don't appear in the VAR menu, and they occupy user memory only during program execution.
- Different local variable structures can use the same variable names without conflict.

Evaluating Local Names

Local names are evaluated differently from global names. When a global name is evaluated, the object stored in the corresponding variable is itself evaluated. (You've seen how programs stored in global variables are automatically evaluated when the name is evaluated.)

When a local name is evaluated, the object stored in the corresponding variable is returned to the stack but is *not* evaluated. When a local variable contains a number, the effect is identical to evaluation of a global name, since putting a number on the stack is equivalent to evaluating it. However, if a local variable contains a program, algebraic expression, or global variable name—and if you

want it evaluated—the program should execute EVAL after the object is put on the stack.

Using Local Variables within Subroutines

Because a program is itself an object, it can be used in another program as a subroutine. When program *B* is used by program *A*, program *A* *calls* program *B*, and program *B* is a *subroutine* in program *A*.

Normally, a local variable exists *only* inside its defining procedure (and *not* within any subroutines called by the defining procedure). Thus, normal local variables can only be used within a subroutine if the subroutine is embedded, or *nested* within the local variable's defining procedure.

However, the HP 48 offers a way to include local variables in subroutines that are not nested within the local variable's defining procedure.

To use a local variable in a subroutine called by the variable's defining procedure:

- When defining the local variable, name it using → (α → ◀) as the first character. This creates a *compiled* local variable.
- When calling the local variable within a subroutine, specify its name using → as the first character.

A compiled local variable is available to any subroutine called by the defining procedure for the local variable. However, compiled local variables are still *local* variables and are purged when the defining procedure is completed.

Local Variables and User-Defined Functions

The defining procedure for a local variable structure can be either an algebraic or program object.

A user-defined function is actually a *program* that consists solely of a local variable structure whose defining procedure is a an algebraic expression. The syntax is:

« → $name_1$ $name_2$ ... $name_n$ '*expression*' »

It takes an unlimited number of arguments (can use an unlimited number of local variables), but returns *one* result to the stack.

If a program begins with a local variable structure and has a program as the defining procedure that returns exactly one result, the complete program acts like a user-defined function in two ways:

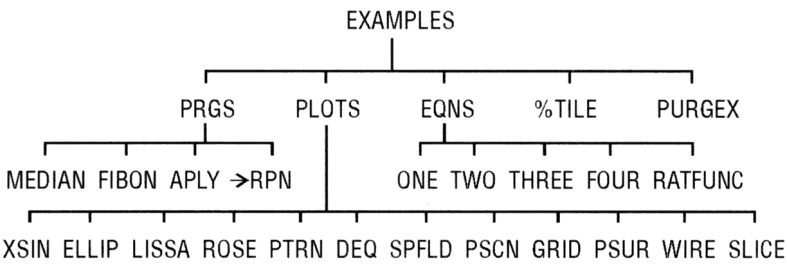
- It takes numeric or symbolic arguments.
- It takes its arguments either from the stack or in algebraic syntax.

However, although such a program can contain commands not allowed in algebraic expressions, it does *not* have a derivative.

Exploring the Programs in the EXAMPLES Directory

To use and explore the EXAMPLES directory:

1. Type TEACH into the command line and press ENTER. This loads the EXAMPLES directory from built-in memory into your HOME directory where you can access it.
2. Press VAR EXAM to open the EXAMPLES directory.



All of the objects contained in EXAMPLES (apart from the subdirectories PRGS, PLOTS, and EQNS) are either programs or algebraic objects. The algebraics in the EQNS subdirectory are used in examples in the *HP 48G Series Quick Start Guide*. The short

programs contained in PLOTS each plot an example of a different plot type. The remaining objects are example programs that accomplish a variety of different tasks.

MEDIAN	Returns a vector containing the median values for each column in the current statistics matrix.
FIBON	Using the contents of variable *n*, returns the *n*-1 Fibonacci number.
APLY	Applies a program to each element of an array. The program applied must take exactly one input and return exactly one output. If output is symbolic the result is returned as a "symbolic array" (that is, a list of "row" lists instead of an array of row vectors).
→RPN	Converts an algebraic object into a list of equivalent RPN commands. Evaluating the resulting list returns the original algebraic. Illustrates the equivalency between algebraic and RPN procedures.
%TILE	Takes a list of data on level 2 and a percentile number on level 1 and returns the value of the percentile for the list. For example, typing { *datalist* } 50 and pressing %TILE returns the median (50th percentile) of the list.

You may wish to work through these programs using single-stepping (see page 29-8).

29

Using HP 48S/SX programs with the HP 48G/GX

There are many programs distributed (both commercially and otherwise) today that were originally written for the HP 48S and HP 48SX, the predecessors of the HP 48G Series.

Caution

Before running a library developed for the HP 48S Series calculators on your HP 48G Series calculator, *back up* the contents of your memory to an external source (computer or plug-in card). Incompatibilities between the library and the HP 48G Series calculator may cause memory loss.

There are no guarantees that such programs will run without errors on the HP 48G Series calculators. However, most older programs that use only *User-RPL* commands—the set of commands that are recognized when you type in their names on the keyboard—will work on the newer HP 48G Series.

There are some differences between the older HP 48S Series and the newer HP 48G Series that may (or may not) affect older programs:

- HP 48S Series programs using the SYSEVAL command may cause memory loss when run on a HP 48G Series calculator due to changes in the internal memory mapping.
- HP 48S Series programs that use variable names that are identical to commands newly introduced on the HP 48G Series calculators will give unpredictable results due to the name conflict. Changing the names used in the older program will prevent this problem.
- HP 48S Series programs that use the MENU command to display a built-in menu may give unexpected results because the HP 48G Series calculators uses a different menu structure (see Appendix C).
- HP 48S Series programs that use flags -14, -28, -29, or -54 will conflict with the built-in meanings for these flags in the HP 48G Series calculators.

Some commercially-distributed *libraries* developed for the HP 48S Series may not work on the HP 48G Series and may, in fact, cause memory loss. In addition, some libraries on plug-in cards may only work when the card is stored in Card Slot 1; others may only work when the card is stored in Card Slot 2. Be sure to backup user memory before experimenting with an untested library.

Where to Find More Information

- The *HP 48G Series Advanced User's Reference* (part number 00048-90136) contains programming information, including syntax information for all HP 48G Series commands, in a reference format.
- The HP Calculator Bulletin Board System (see inside back cover) provides a forum for exchange of information about the availability and compatibility of software developed for either Series S or Series G calculators. It is also a good source of programming tips and interesting programs.

30

Customizing the HP 48

Customizing Menus

A custom menu is a menu that you create. It can contain menu labels for operations, commands, and other objects that you create or group together for your own convenience.

A custom menu is defined by the contents of a *reserved* variable named *CST*. So, the way to create a custom menu involves creating a variable *CST* that contains the objects you want in your menu.

To create and display a custom (CST) menu:

1. Enter a list containing the objects you want in the menu. (The different object types serve different purposes.)
2. Press ◀ (MODES) MENU MENU .

To display the current CST menu:

- Press (CST).

Objects in the CST menu usually have the same functionality they do in built-in menus:

- **Names.** Names behave like VAR menu keys. Thus, if *ABC* is a variable name, ABC evaluates *ABC*, ▶ ABC recalls its contents, and ◀ ABC stores new contents in *ABC*. Also, the menu label for the name of a directory has a bar over the left corner of the label—pressing the menu key switches to that directory.
- **Units.** Unit objects act like UNITS Catalog entries. For example, they have their left-shifted conversion capability.
- **Strings.** String objects echo the contents of the string, like a typing aid.

- **Commands.** Almost all command names behave like normal command keys.

You can include backup objects in the list defining a custom menu by tagging the name of the backup object with its port location. For example, if :2:TOM were included in the custom menu list, a menu label TOM would represent the backup object *TOM* in port 2.

If you want to create typing aids for certain commands that affect program flow (such as HALT, PROMPT, IF...THEN...END, and other program control structures), include them as string objects, not as command names.

Example: Create a custom menu containing the built-in command →TAG, the unit object 1_m^3, a string to serve as a typing aid for VOLUME, and the variable name *CST*.

Step 1: Enter the list of objects.

(←) {} PRG TYPE →TAG
1 (→) _ (α) (←) m y^x 3
(→) " " (α) (α) VOLUME (α)
(▶)
(α)-(hold) C S T (release)
ENTER

Step 2: Create and display the CST menu.

(←) MODES MENU MENU

Step 3: Convert 1075 cm^3 to m^3.

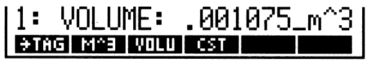
1075 (→) _ (α) (←) c (α) (←) m
y^x 3 ENTER
(←) M^3

Step 4: Enter the string "VOLUME".

(→) " " VOLU ENTER

Step 5: Create a tagged object from the contents of levels 2 and 1.

`→TAG`



Step 6: Display the current contents of *CST*.

`CST`

You can create a *CST* in each directory in memory, just as you can for other variables. This lets you have a different custom menu in each directory.

Also, instead of storing the list of objects itself in *CST*, you can optionally store the name of another variable that contains the list. This gives you the ability to have in one directory several variables that contain different custom-menu lists. That way, you can easily switch the CST menu from one custom menu to another by simply storing a new name in *CST*.

Enhancing Custom Menus

You can enhance the CST menu by creating special menu labels and by specifying different actions for unshifted and shifted keys.

To create a special menu label for an object:

- Inside the CST list, replace the object by an embedded list of the form « "*label*" *object* ».

The default label for an object in the CST menu is the underlying name, command, unit, or typing aid—as many characters as fit in the space available.

Example: Storing « →TAG 1_m^3 « "VOL" "VOLUME"» « "CUST" CST » » in *CST* gives the same CST menu operations as the previous example, but the labels are `→TAG`, `M^3`, `VOL`, and `CUST`.

To specify functionality for shifted keys:

- Inside the CST list, replace the object by an embedded list of objects: { $object_{\text{unshifted}}$ $object_{\text{left-shift}}$ $object_{\text{right-shift}}$ }. (You can omit the last one or two objects if you want.)

You must specify the unshifted action in order to have the shifted actions. In addition, you can combine the special-label enhancement and the shifted-functionality enhancement—see the following example.

Example: Suppose you want the CST menu key VOL to provide the following three actions:

- VOL evaluates a program that stores the value in level 1 in a variable named *VBOX*.
- ◀ VOL evaluates a program that computes the product of levels 1, 2, and 3.
- ▶ VOL types VOLUME.

The following CST list provides the desired custom menu. The menu contains only one label: VOL.

```
{ { "VOL" { « 'VBOX' STO » « * * » "VOLUME" } } }
```

To create a temporary menu:

30

1. Enter the menu list as with CST.
2. Press ◀ MODES MENU TMEN. This creates and displays a temporary menu without overwriting the contents of the variable *CST*.

Customizing the Keyboard

The HP 48 lets you assign alternate functionality to any key on the keyboard (including alpha and shifted keys), enabling you to customize the keyboard for your particular needs. Your customized keyboard is called the *user keyboard*, and it's active whenever the calculator is in *User mode*.

User Modes

To activate User mode:

- If you want to execute only one operation (1USR), press ◀ USER. (It turns off after the operation.)
- If you want to execute several operations (USER), press ◀ USER ◀ USER. (Press ◀ USER a third time to turn off User mode.)

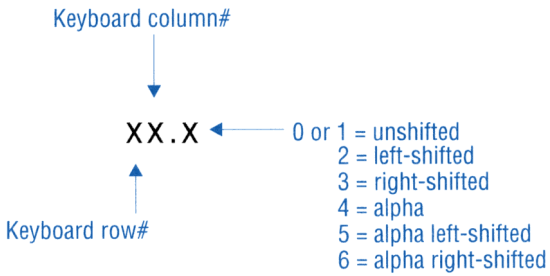
The ◀ USER key is a three-way switch, much like the α key. Pressing the key once activates the mode only for the next operation, while pressing it twice consecutively locks the mode on, requiring a third press to turn the mode off again. If you prefer, set Flag −61 to enable a single press of ◀ USER to lock User mode on.

Assigning and Unassigning User Keys

You can assign commands or other objects to any user keys (including shifted keys). The behaviors for different types of objects are the same as for custom menus—see "Customizing Menus" on page 30-1.

To assign an object to a user key:

1. Enter the object to be assigned to the key.
2. Enter the three-digit location number that specifies the key. (See the diagram below.)
3. Press ◀ MODES KEYS ASN .



To assign a built-in command to a user key:

1. Enter a list containing two parameters—the command to be assigned to the key followed by the three-digit key location number (see above).
2. Press ◀ MODES KEYS STOK .

To assign several user keys:

1. Enter a list containing two key-assignment parameters for each key—the object to be assigned to the key followed by the three-digit key location number (see above).
2. Press ◀ MODES KEYS STOK .

This is an example of a key-assignment list for STOKEYS:

```
{ SINH 41 "3.14" 94.2 ABC 11.4 }
```

You can use 'SKEY' as an assignment object. It means the "standard" (unassigned) key definition.

When you press a user key, its assigned object is executed—or, if the key is unassigned, the standard operation is performed. (You can also disable keys, as described in the next topic.)

After you've assigned a user key, the assignment remains in effect until you reassign the key using ASN or STOKEYS, or until you *unassign* the key. An unassigned user key reverts to its standard definition—the same as for the standard keyboard.

30

To unassign previously-assigned user keys:

- To unassign one user key, enter the three-digit key number, then press ◀ MODES KEYS DELK . An unassigned user key reverts to its standard definition—the same as for the standard keyboard.
- To unassign several user keys, enter a list containing the three-digit key numbers, then press ◀ MODES KEYS DELK .
- To unassign all user keys, press 0 ◀ MODES KEYS DELK . All user keys are unassigned and all disabled keys are enabled (see next topic).

Disabling User Keys

You can *disable* user keys that are unassigned—so they do nothing. This lets you control the user keys that are active, including assigned keys and standard (unassigned) keys.

If you assign a disabled user key, it becomes enabled.

To disable all unassigned user keys:

- Enter `'S'` and press ◀ (MODES) `KEYS` `DELK`.

To reenable unassigned, disabled user keys:

- To enable one unassigned key, enter `'SKEY'`, enter the three-digit key number, then press ◀ (MODES) `KEYS` `ASN`.
- To enable several unassigned keys, enter a list containing `'SKEY'` and the three-digit key number for each key, then press ◀ (MODES) `KEYS` `STOK`. (Include one `'SKEY'` for each key.)
- To enable and unassign *all* user keys, press 0 ◀ (MODES) `KEYS` `DELK`.

To reenable and assign disabled user keys:

- To enable and assign one user key, enter the object to be assigned to the key, enter the three-digit key number, then press ◀ (MODES) `KEYS` `ASN`.
- To enable all user keys and assign several keys, enter a list with `S` as the first object, followed by the assigned object and three-digit key number for each key assignment, then press ◀ (MODES) `KEYS` `STOK`.

Recalling and Editing User Key Assignments

To recall the current user key assignments:

- Press ◀ (MODES) `KEYS` `RCLK` (the RCLKEYS command).

The RCLKEYS command returns to level 1 a list of all the current user key assignments—pairs of assignment objects and three-digit key numbers. If the first item in the list is the letter `S`, then unassigned user keys are currently enabled—otherwise, unassigned keys are currently disabled.

To edit the user key assignments:

1. Press ◀ (MODES) KEYS RCLK (the RCLKEYS command).
2. Press ◀ (EDIT) and edit the key-assignment list.
3. Press ◀ (MODES) KEYS DELK STOK (the STOKEYS command) to delete the old assignments and activate the edited assignments.

Note

If you get stuck in User mode—probably with a "locked" keyboard—because you've reassigned or disabled the keys for canceling User mode, hold down the (ON) key and press the C key, then release the C key first.

Deleted user key assignments still take up from 2.5 to 15 bytes of memory each. You can free this memory by packing your user key assignments—press ◀ (MODES) KEYS RCLK 0 DELK STOK.

A

Support, Batteries, and Service

Answers to Common Questions

You can obtain answers to questions about using your calculator from our Calculator Support department. Our experience has shown that many customers have similar questions about our products, so we've provided this section to answer many of these questions. If you don't find the answer to your question here, contact us at the address or phone number on the inside back cover.

Q: *Sometimes my HP 48 will flash when I turn it on. Is this normal?*

A: This is normal for the HP 48.

Q: *I'm not sure whether the calculator is malfunctioning or if I'm doing something incorrectly. How can I verify that the calculator is operating properly?*

A: See "Testing Calculator Operation" on page A-9.

Q: *The (•)) annunciator stays on even when the calculator is turned off. Is anything wrong?*

A: This indicates a low-battery condition in the calculator or a RAM card, or an alarm that is past due. To determine what is causing the (•)) annunciator to stay on, turn the calculator off and then on. A message in the display will identify the problem. See "When to Replace Batteries" on page A-4 or "Responding to Alarms" on page 26-4.

Q: *How can I determine how much memory is left in the calculator?*

A: Press ← MEMORY MEM. The number of bytes of available memory will appear at the lower right corner of the display. For example, an empty memory for the HP 48GX should show approximately 127000 bytes of internal RAM (with no RAM cards installed).

Q: *What does an* E *in a number mean (for example,* 2.51E-13*)?*

A: Exponent of 10 (for example, 2.51×10^{-13}). See "Keying In Numbers" on page 2-1 and "Setting the Display Mode" on page 4-2.

Q: *Why do trig functions give me unexpected results?*

A: The angle mode may be wrong for your problem. Check the angle mode annunciator: RAD means radians, GRAD means grads, and none means degrees. Press ◀ RAD or use the ▶ MODES screen to change the angle mode.

Q: *When I take the sine of π in Degrees mode, why do I get* '$\sin(\pi)$' *instead of a number?*

A: The calculator is in Symbolic Result mode; '$\sin(\pi)$' is the symbolic answer. Press ◀ →NUM to convert '$\sin(\pi)$' to its numeric equivalent of .0548 ... up to 11 decimal places (sin 3.14°). You can also press ◀ MODES MISC SYM▪ to change to Numeric Results mode and prevent symbolic evaluation.

Q: *When I evaluate* '$\sin(\pi)$' *I don't get zero. Why not?*

A: The HP 48, like all calculators, can only compute using a finite number of decimal places. Because π contains an infinite number of decimal places, any results involving π must necessarily be rounded. Sometimes, as in this case, the rounded number differs from the theoretical answer by 10^{-12} (a millionth of a millionth) or so.

Q: *When I differentiate or integrate I get an* Undefined Name *error message. What is wrong?*

A: Symbolic solution mode is not set and the calculator is attempting (unsucessfully) to find a numerical answer using only symbolic variables. Press ◀ MODES MISC SYMB, or make sure that the RESULTS: field in the Integrate or Differentiate input forms shows Symbolic, and try again.

Q: *When I compute* $(-1)^{\frac{2}{3}}$ *I get a complex number instead of 1. What is wrong?*

A: The HP 48 is designed to return the complex *principal* solution for any fractional exponent. To get the real number root, use the $\sqrt[x]{y}$ operator (the $\sqrt[y]{y}$ key or XROOT command) instead.

Q: *What does "object" mean?*

A: "Object" is the general term for all elements of data the HP 48 works with. Numbers, expressions, arrays, programs, and so on, are all types of objects.

Q: *What do three dots (...) mean at either end of a display line?*

A: The three dots (called an *ellipsis*) indicate that the displayed object is too long to display on one line. To view undisplayed portions of the object, use the ◀ or ▶ cursor keys.

Q: *How do I turn off the* HALT *annunciator?*

A: Press PRG NXT RUN KILL.

Q: *The calculator beeps and displays* Bad Argument Type. *What's wrong?*

A: The objects on the stack aren't the correct type for the command you are attempting. For example, executing →UNIT (in the PRG TYPE menu) with a number in stack levels 1 and 2 causes this error.

Q: *The calculator beeps and displays* Too Few Arguments. *What's wrong?*

A: There are fewer arguments on the stack than required by the command you are attempting. For example, executing + with only one argument or number on the stack causes this error.

Q: *The calculator beeps and displays a message different from the two listed above. How do I find out what's wrong?*

A: Refer to appendix B, "Messages."

Q: *I can't find some variables that I used earlier. Where did they go?*

A: You may have been using the variables in a different directory. If you can't remember which directory you were using, you'll need to check all the directories in your calculator.

Q: *Sometimes my HP 48 seems to pause momentarily during a calculation. Is anything wrong?*

A: Nothing is wrong. The calculator does some system cleanup from time to time to eliminate temporary objects created from normal operation. This cleanup process frees memory for current operations. This happens less often if you make more memory available.

Q: *The Equation Library Solver gives me SI units even when I specify* ENG *(or vice-versa).*

A: The Solver uses and creates global variables. If the variables in question were created before, then they still exist (until you purge them). Their unit definitions also still exist. To override the old unit system, you must either purge the variables before solving, or you must enter the specific units you want (such as _ft).

Environmental Limits

To maintain product reliability, avoid getting the calculator and plug-in cards wet, and observe the following temperature and humidity limits:

Calculator:

- 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.

Plug-In Cards:

- Operating temperature: 0° to 45°C (32° to 113°F).
- Storage temperature: −20° to 60°C (−4° to 140°F).
- Storage temperature for RAM card data retention: 0° to 60°C (32° to 140°F).
- Operating and storage humidity: 90% relative humidity at 40°C (104°F) maximum.

When to Replace Batteries

When a low-battery condition exists, the ((•)) annunciator remains on, even when the calculator is turned off. When the calculator is turned on during a low-battery condition, `Warning: LowBat( )` is displayed for approximately 3 seconds:

A

`LowBat(P1)` refers to port 1.
`LowBat(P2)` refers to port 2.
`LowBat(S)` refers to the calculator (system) batteries.

Note

Replace the RAM card battery or the calculator batteries as soon as possible after the ((•)) low-battery annunciator and warning message appear. If you continue to use the calculator while the ((•)) annunciator is on, the display will eventually dim and you may lose calculator and RAM card data.

Under typical use, a RAM card's battery should last between 1 and 3 years. Be sure to mark the card with the battery-installation date, and, in case the RAM card is not in the calculator when the battery needs replacement, set an alarm for 1 year from that date to remind you to install a fresh battery. RAM cards do not come with a battery installed.

Changing Batteries

The HP 48 uses the following kinds of batteries:

- **Calculator Batteries.** Any brand of size AAA batteries. *Be sure that all three batteries are of the same brand and type.* (The use of rechargeable batteries is not recommended because of their lower capacity and short low-battery warning time.)
- **Plug-In RAM Card Batteries.** 3-volt 2016 coin cell. (Not used in the HP 48G.)

To replace calculator batteries, use the steps below. To replace RAM card batteries, see "To change a RAM card battery" on page A-7.

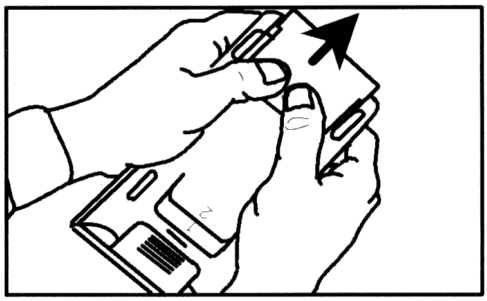
Caution

Whenever you remove batteries from the calculator, be sure the calculator is off, and *do not press the* ON *key until the new batteries are installed.* If you press ON when batteries are not in the calculator, you may lose all calculator memory.

To change calculator batteries:

1. Turn the calculator off. You may lose memory in the calculator and plug-in RAM cards if the calculator batteries are removed when the calculator is on.
2. Have three, fresh size AAA batteries (of the same brand and type) at hand. Wipe off both ends of each battery with a clean, dry cloth.

3. Remove the calculator battery-compartment door by pressing down and sliding it off away from the calculator. Be careful not to press the calculator's ON key. See the following illustration.



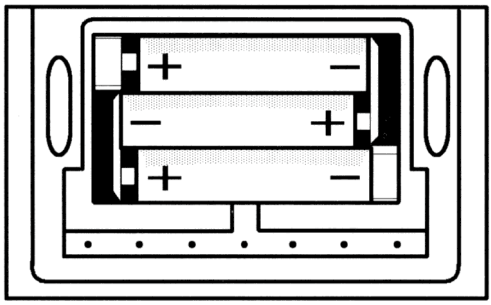
4. Turn the calculator over and shake the batteries out. After the batteries are out, you should replace them with fresh batteries within 2 minutes to protect against memory loss.

Warning

Do not mutilate, puncture, or dispose of batteries in fire. The batteries can burst or explode, releasing hazardous chemicals. Discard used batteries according to the manufacturer's instructions.

A

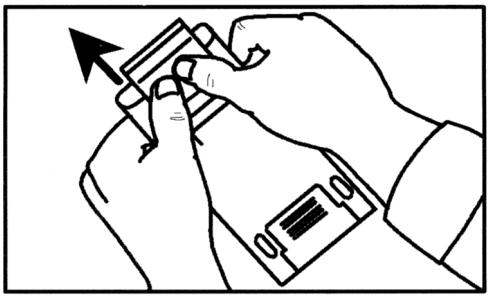
5. Position the batteries according to the outlines in the bottom of the battery compartment. *Avoid touching the battery terminals.* Batteries are easier to install if the negative (plain) ends are inserted first, and if the center battery is installed last. See the following illustration.



6. Replace the battery-compartment door by sliding the tabs on the door into the slots in the calculator case.
7. Press [ON] to turn the calculator on.

To change a RAM card battery:

1. Turn the calculator over and remove the plastic cover over the plug-in card ports (on the display-end of the calculator).

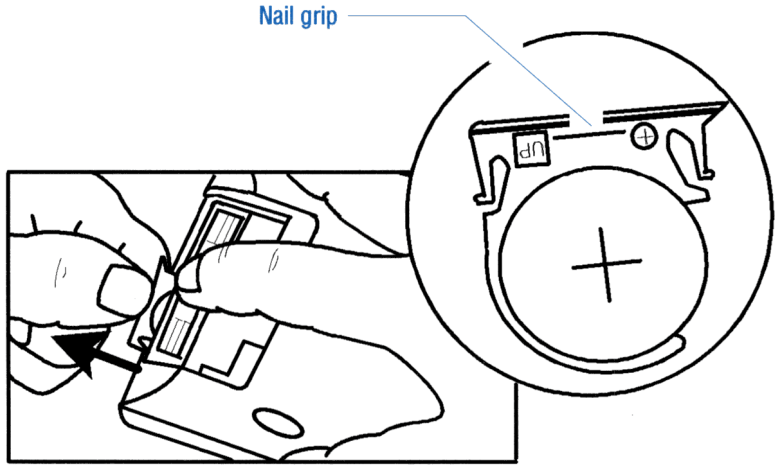


2. With the RAM card in port 1 or 2, *turn* **on** *the calculator.*

Caution

Make sure you *turn on the calculator* before you change a RAM card battery. RAM cards run off the calculator batteries only while the calculator is on. RAM memory may be lost if you remove a RAM card battery while the calculator is off, or while the card is not installed in the calculator.

3. Place your index finger in the recess near the exposed end of the RAM card—this prevents removal of the card from the calculator when you remove the card's battery holder. Now insert the thumbnail of your free hand into the nail grip in the black plastic at the left side of the end of the card and pull the battery holder out of the card.



A

4. Remove the old battery from the plastic battery holder.

Warning

Do not mutilate, puncture, or dispose of batteries in fire. The batteries can burst or explode, releasing hazardous chemicals. Discard used batteries according to the manufacturer's instructions.

5. Install a fresh, 3-volt 2016 coin cell in the plastic battery holder, and reinsert the holder (with battery) into the card. *Be sure to install the battery with the side marked "+" toward the front of the card.*
6. Mark the card with the battery-installation date, and set an alarm for 1 year from that date to remind you to change it. (If you unplug the card, the HP 48 can't check the card's battery level.)
7. Replace the plug-in port cover.

Testing Calculator Operation

Use the following guidelines to determine whether the calculator is functioning properly. Test the calculator after every step to see if operation has been restored. If your calculator requires service, see "If the Calculator Requires Service" on page A-16.

If the calculator won't turn on or doesn't respond when you press the keys:

1. Make sure that three fresh batteries are correctly installed in the calculator.
2. Press and release ON.
3. If the calculator comes on, but the display is blank, press and hold ON, press and release + several times until characters become visible, then release ON. If no characters appear in the display, the calculator requires service.
4. If a halted program won't respond when you press CANCEL, try pressing CANCEL again.
5. If the keyboard is "locked," perform a system halt:
   a. Press and hold ON.
   b. Press and release the "C" key (the key with C next to it).
   c. Release ON. The empty stack display should appear.
   d. If the problem still exists, perform a manual system halt (see page 5-17).

6. If the problem still exists, perform a memory reset. You may lose data during a memory reset, so do this only if you must:

   a. Press and hold (ON).
   b. Press and hold the "A" and "F" keys (the keys with A and F next to them).
   c. Release all three keys.

   The calculator will beep and display the message `Try To Recover Memory?` at the top of the display. Press `YES` to recover as much memory as possible.

If these steps fail to restore operation, the calculator requires service.

If the calculator responds to keystrokes, but you suspect it's malfunctioning:

1. Run the self-test described in the next section.
   - If the calculator fails the self-test, it requires service.
   - If the calculator passes the self-test, you may have made a mistake operating the calculator. Reread appropriate portions of the manual and check "Answers to Common Questions" on page A-1.
2. Contact the Calculator Support department. The address and phone number are listed on the inside back cover.

Self-Test

If the display turns on, but the calculator does not seem to be operating properly, run the diagnostic self-test.

To run the self-test:

1. Turn on the calculator.
2. Press and hold (ON).
3. Press and release the "E" key (the key with E next to it).
4. Release (ON).

The diagnostic self-test tests the internal ROM and RAM, and generates various patterns in the display. The test repeats continuously until you perform a system halt.

To halt the self-test (system halt):

1. Press and hold (ON).
2. Press and release the "C" key (the key with C next to it).
3. Release (ON). The empty stack display should appear.

If the self-test indicates an internal ROM or RAM failure (if IROM OK and IRAM OK are not displayed), the calculator requires service.

The diagnostic self-test should be successfully completed before running any of the tests described in the following sections.

Keyboard Test

This test checks all of the calculator's keys for proper operation.

To run the interactive keyboard test:

1. Turn on the calculator.
2. Press and hold (ON).
3. Press and release the "D" key (the key with D next to it).
4. Release (ON).
5. Press and release the "E" key (the key with E next to it). KBD1 will appear in the upper left corner of the display.
6. Starting at the upper left corner and moving left to right, press each of the 49 keys on the keyboard.

If you press the keys in the proper order and they're functioning properly, the calculator emits a high-pitch beep at each press of a key. When you press the 49th key, (+), the displayed message should change to KBD1 OK.

If you press a key out of sequence, a five-digit hexadecimal number will appear next to KBD1. Rerun the test by repeating steps 2 through 6 above.

If a key isn't functioning properly, the next keystroke displays the hex location of the expected and the received location. If you pressed the keys in order and got this message, the calculator requires service. Be sure to include a copy of the error message when you ship the calculator for service.

To exit the keyboard test (system halt):

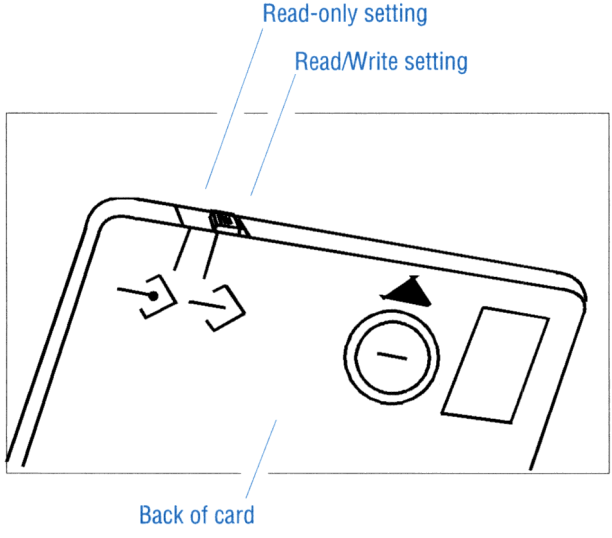
1. Press and hold ON.
2. Press and release the "C" key (the key with C next to it).
3. Release ON. The empty stack display should appear.

Port RAM Test

The port RAM test nondestructively tests the ports (for those calculators that have them) and the installed plug-in RAM cards. (Plug-in RAM-card memory is preserved.)

To run the port RAM test:

1. Check that a plug-in RAM card is properly installed in port 1 or port 2.
2. Verify that the switch on each card is set to the "Read/Write" position.



3. Turn on the calculator.
4. Press and hold ON.

5. Press and release the "D" key (the key with D next to it).
6. Release ON. A vertical line will appear at both sides and at the center of the display.
7. Press and release ▲.

RAM1 or RAM2 will appear at the top left corner of the display and the size of the corresponding plug-in RAM card (32K or 128K) will appear at the top right corner of the display. OK will appear to the right of RAM1 or RAM2 when the port RAM test has been successfully completed.

A failure message (for example, RAM1 00002) will be displayed for each port that does not contain a plug-in RAM card or if a card's read/write switch is in the "write-protect" position. This message should be ignored.

If OK doesn't appear for a RAM card set to read/write, the card should be moved to the other port and the test rerun. If OK still doesn't appear, the RAM card should be replaced with a new one.

To return to normal calculator operation (system halt):

1. Press and hold ON.
2. Press and release the "C" key (the key with C next to it).
3. Release ON. The empty stack display should appear.

IR Loop-Back Test

This test checks the operation of the send and receive infrared sensors and their associated circuits.

To run the IR loop-back test:

1. Turn on the calculator.
2. Press and hold ON.
3. Press and release the "D" key (the key with D next to it).
4. Release ON. A vertical line will appear at both sides and at the center of the display.
5. Be sure that the plastic plug-in card cover is in place and that it covers the clear lamp bulbs in the top end of the calculator.
6. Press EVAL.

IRLB will appear at the top left corner of the display. If OK appears to the right of IRLB, the calculator passes this test. If OK doesn't appear, the calculator requires service.

To return to normal calculator operation (system halt):

1. Press and hold (ON).
2. Press and release the "C" key (the key with C next to it).
3. Release (ON). The empty stack display should appear.

Serial Loop-Back Test

This test checks the operation of the send and receive circuits of the serial interface at the top of the calculator.

To run the serial loop-back test:

1. Turn on the calculator.
2. Press and hold (ON).
3. Press and release the "D" key (the key with D next to it).
4. Release (ON). A vertical line will appear at both sides and at the center of the display.
5. Temporarily connect (short) the middle two pins (pins 2 and 3) of the 4-pin serial connector at the top end of the calculator. Be careful not to bend or severely jar the pins. You must have this connection bridged during the entire duration of the test.
6. Press (PRG).

A

U_LB will appear at the top left corner of the display. If OK appears to the right of U_LB, the calculator passes this test. If OK doesn't appear, the calculator requires service.

Note

If you inadvertently short pins 1 and 2 or pins 3 and 4 of the serial connector, the loop-back test will return U_LB 00001 or U_LB 00002 (test-failed message), but you will not damage the calculator.

To return to normal calculator operation (system halt):

1. Press and hold ON.
2. Press and release the "C" key (the key with C next to it).
3. Release ON. The empty stack display should appear.

Limited One-Year Warranty

What Is Covered. *The calculator* (except for the batteries, or damage caused by the batteries) *and calculator accessories are 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. (Replacement may be made with a newer model of equal or better functionality.)

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.*

Damage caused to the HP 48 as the result of using nonapproved plug-in cards and plug-in accessories is not covered by the Hewlett-Packard warranty.

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 Australia and the United Kingdom. The above disclaimers and limitations shall not apply to consumer transactions in Australia and the United Kingdom and shall not affect the statutory rights of consumers.

If the Calculator Requires Service

Hewlett-Packard maintains service centers in many countries. These centers will repair a calculator, or replace it with the same model or one of equal or better functionality, whether it is under warranty or not. There is a service charge for service after the warranty period. Calculators normally are serviced and reshipped within 5 working days.

Note

If the contents of your calculator's memory are important, you should back up the memory on a plug-in RAM card, another HP 48, or a computer before sending in the calculator for repair.

A

- **In the United States:** Send the calculator to the Corvallis Service Center listed on the inside of the back cover.
- **In Europe:** Contact your Hewlett-Packard sales office or dealer, or Hewlett-Packard's European headquarters (address below) 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: 022 780.81.11

- **In other countries:** Contact your Hewlett-Packard sales office or dealer or write to the Corvallis Service Center (listed on the inside of the back cover) for the location of other service centers. If local service is unavailable, you can ship the calculator to the Corvallis Service Center for repair.

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

Service Charge. Contact the Corvallis Service Center (inside back cover) for the standard out-of-warranty repair charges. This 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 charges. These 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.

- Include your return address and a *detailed description* of the problem. Details should include ROM/RAM cards installed, error messages, and any peripherals connected at the time of malfunction.
- 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.
- Ship your calculator postage *prepaid* in adequate protective packaging to prevent damage. Shipping damage is not covered by the warranty, so we recommend that you insure the shipment.

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. For additional information, contact the Corvallis Service Center (see the inside of the back cover).

B

Error Messages

This appendix lists selected HP 48 messages, arranged alphabetically.

Messages Listed Alphabetically

Message	Meaning	# (hex)
Alarm	Alarm not acknowledged yet.	(none)
All Variables Known	There are no unknown variables to solve for.	E405
Bad Argument Type	One or more stack arguments were incorrect type for operation.	202
Bad Argument Value	Argument value out of operation's range.	203
Bad Guess(es)	Guess(es) supplied to HP Solve application or ROOT lie outside domain of equation.	A01
Bad Packet Block Check	Kermit error: Computed packet checksum doesn't match checksum in packet.	C01
Can't Edit Null Char	Attempted to edit a string containing character with code 0.	102
Circular Reference	Attempted to store a variable name into itself.	129

Messages Listed Alphabetically (continued)

Message	Meaning	# (hex)
Constant?	HP Solve application or ROOT returned same value at every sample point of current equation.	A02
Directory Not Allowed	Name of existing directory variable used as argument.	12A
Directory Recursion	Attempted to store a directory into itself.	002
EQ Invalid for MINIT	*EQ* must contain at least two equations (or programs) and two variables.	E403
Extremum	Result returned by HP Solve application or ROOT is an extremum rather than a root.	A06
HALT Not Allowed	A program containing HALT executed while MatrixWriter application, DRAW, CALC, or HP Solve application active.	126
Illegal During MROOT	Multiple-Equation Solver command attempted during MROOT execution.	E406
Inconsistent Units	Attempted unit conversion with incompatible units.	B02
Infinite Result	Math exception: Calculation such as 1/0 has infinite result.	305
Insufficient Memory	Not enough free memory to execute operation.	001

Messages Listed Alphabetically (continued)

Message	Meaning	# (hex)
`Insufficient Σ Data`	A Statistics command was executed when ΣDAT did not contain enough data points for calculation.	603
`Interrupted`	The HP Solve application or ROOT was interrupted by (CANCEL).	A03
`Invalid Array Element`	(ENTER) returned object of wrong type for current matrix.	502
`Invalid Card Data`	HP 48 does not recognize data on plug-in card, or at least one port on the card has never been used.	008
`Invalid Date`	Date argument not real number in correct format, or was out of range.	D01
`Invalid Definition`	Incorrect structure of equation argument for DEFINE.	12C
`Invalid Dimension`	Array argument had wrong dimensions.	501
`Invalid EQ`	Attempted operation from GRAPHICS FCN menu when *EQ* did not contain algebraic, or attempted DRAW with CONIC plot type when *EQ* did not contain algebraic.	607
`Invalid IOPAR`	*IOPAR* not a list, or one or more objects in list missing or invalid.	C12
`Invalid Mpar`	*Mpar* variable was not created by MINIT.	E401

Messages Listed Alphabetically (continued)

Message	Meaning	# (hex)
`Invalid Name`	Received illegal filename, or server asked to send illegal filename.	C17
`Invalid PPAR`	*PPAR* not a list, or one or more objects in list missing or invalid.	12E
`Invalid PRTPAR`	*PRTPAR* not a list, or one or more objects in list missing or invalid.	C13
`Invalid PTYPE`	Plot type invalid for current equation.	620
`Invalid Repeat`	Alarm repeat interval out of range.	D03
`Invalid Server Cmd.`	Invalid command received while in Server mode.	C08
`Invalid Syntax`	HP 48 unable to execute OBJ→, ENTER, or STR→ due to invalid object syntax.	106
`Invalid Time`	Time argument not real number in correct format, or out of range.	D02
`Invalid Unit`	Unit operation attempted with invalid or undefined user unit.	B01
`Invalid User Function`	Type or structure of object executed as user-defined function was incorrect.	103
`Invalid Σ Data`	Statistics command executed with invalid object stored in *ΣDAT*.	601

B

Messages Listed Alphabetically (continued)

Message	Meaning	# (hex)
Invalid Σ Data LN(Neg)	Nonlinear curve fit attempted when ΣDAT matrix contained a negative element.	605
Invalid Σ Data LN(0)	Nonlinear curve fit attempted when ΣDAT matrix contained a 0 element.	606
Invalid ΣPAR	ΣPAR not list, or one or more objects in list missing or invalid.	604
LAST STACK Disabled	↷ UNDO pressed while that recovery feature disabled.	124
LASTARG Disabled	LASTARG executed while that recovery feature disabled.	205
LowBat( )	Replace calculator batteries (S), or replace plug-in card batteries (P1) or (P2).	(none)
Low Battery	System batteries too low to safely print or perform I/O.	C14
Memory Clear	HP 48 memory was cleared.	005
Name Conflict	The | (where) function attempted to assign a value to the variable of integration or summation index.	13C
Negative Underflow	Math exception: Calculation returned negative result, between 0 and −MINR.	302
No Current Equation	SOLVR, DRAW, or RCEQ executed with nonexistent EQ.	104

Messages Listed Alphabetically (continued)

Message	Meaning	# (hex)
No current equation	Plot or HP Solve application executed with nonexistent *EQ*.	609
No Room in Port	Insufficient free memory in specified RAM port.	00B
No Room to Save Stack	Not enough free memory to save copy of the stack. LAST STACK is automatically disabled.	101
No Room to Show Stack	Stack objects displayed by type only due to low memory condition.	131
No stat data to plot	No data stored in ΣDAT.	60F
Non-Empty Directory	Attempted to purge nonempty directory.	12B
Non-Real Result	Execution of HP Solve application, ROOT, DRAW, or $\int$ returned result other than real number or unit.	12F
Nonexistent Alarm	Alarm list did not contain alarm specified by alarm command.	D04
Nonexistent ΣDAT	Statistics command executed when ΣDAT did not exist.	602
Object Discarded	Sender sent an EOF (Z) packet with a "D" in the data field.	C0F
Object In Use	Attempted PURGE or STO into a backup object when its stored object was in use.	009

Messages Listed Alphabetically (continued)

Message	Meaning	# (hex)
Object Not in Port	Attempted to access a nonexistent backup object or library.	00C
(OFF SCREEN)	Function value, root, extremum, or intersection was not visible in current display.	61F
Out of Memory	One or more objects must be purged to continue calculator operation.	135
Overflow	Math exception: Calculation returned result greater in absolute value than MAXR.	303
Parity Error	Received bytes' parity bit doesn't match current parity setting.	C05
Port Closed	Possible IR or serial hardware failure. Run self-test.	C09
Port Not Available	Used a port command on an empty or nonexistent port, or one containing ROM instead of RAM. (Ports 1 and 2 don't exist in the HP 48G.) Attempted to execute a server command that itself uses the I/O port.	00A
Positive Underflow	Math exception: Calculation returned positive result, between 0 and MINR.	301

Messages Listed Alphabetically (continued)

Message	Meaning	# (hex)
Power Lost	Calculator turned on following a power loss. Memory may have been corrupted.	006
Protocol Error	Received a packet whose length was shorter than a null packet. Maximum packet length parameter from other machine is illegal.	C07
Receive Buffer Overrun	Kermit: More than 255 bytes of retries sent before HP 48 received another packet. SRECV: Incoming data overflowed the buffer.	C04
Receive Error	UART overrun or framing error.	C03
Sign Reversal	HP Solve application or ROOT unable to find point at which current equation evaluates to zero, but did find two neighboring points at which equation changed sign.	A05
Single Equation	Only one equation has been supplied to the Multiple-Equation Solver.	E402

Messages Listed Alphabetically (continued)

Message	Meaning	# (hex)
Timeout	Printing to serial port: Received XOFF and timed out waiting for XON. Kermit: Timed out waiting for packet to arrive.	C02
Too Few Arguments	Command required more arguments than were available on stack.	201
Transfer Failed	Ten successive attempts to receive a good packet were unsuccessful.	C06
Unable to Isolate	ISOL failed because specified name absent or contained in argument of function with no inverse.	130
Undefined Constant	The name supplied to CONST is not in the Constants Library.	E129
Undefined Local Name	Executed or recalled local name for which corresponding local variable did not exist.	003
Undefined Name	Executed or recalled global name for which corresponding variable does not exist.	204
Undefined Result	Calculation such as 0/0 generated mathematically undefined result.	304
Undefined XLIB Name	Executed an XLIB name when specified library absent.	004

Messages Listed Alphabetically (continued)

Message	Meaning	# (hex)
Wrong Argument Count	User-defined function evaluated with an incorrect number of parenthetical arguments.	128
Zero	Result returned by the HP Solve application or ROOT is a root (a point at which current equation evaluates to zero).	A04

C

Menus

No.	Name
0	Last Menu
1	CST
2	VAR
3	MTH
4	MTH VECTR
5	MTH MATR
6	MTH MATR MAKE
7	MTH MATR NORM
8	MTH MATR FACTR
9	MTH MATR COL
10	MTH MATR ROW
11	MTH LIST
12	MTH HYP
13	MTH PROB
14	MTH REAL
15	MTH BASE
16	MTH BASE LOGIC
17	MTH BASE BIT
18	MTH BASE BYTE
19	MTH FFT
20	MTH CMPL
21	MTH CONS
22	PRG
23	PRG BRCH
24	PRG BRCH IF
25	PRG BRCH CASE
26	PRG BRCH START

No.	Name
27	PRG BRCH FOR
28	EDIT
29	PRG BRCH DO
30	SOLVE ROOT SOLVR
31	PRG BRCH WHILE
32	PRG TEST
33	PRG TYPE
34	PRG LIST
35	PRG LIST ELEM
36	PRG LIST PROC
37	PRG GROB
38	PRG PICT
39	PRG IN
40	PRG OUT
41	PRG RUN
42	UNITS Catalog
43	UNITS LENG
44	UNITS AREA
45	UNITS VOL
46	UNITS TIME
47	UNITS SPEED
48	UNITS MASS
49	UNITS FORCE
50	UNITS ENRG
51	UNITS POWR
52	UNITS PRESS

No.	Name
53	UNITS TEMP
54	UNITS ELEC
55	UNITS ANGL
56	UNITS LIGHT
57	UNITS RAD
58	UNITS VISC
59	UNITS Commands
60	PRG ERROR IFERR
61	PRG ERROR
62	CHAR
63	MODES
64	MODES FMT
65	MODES ANGL
66	MODES FLAG
67	MODES KEYS
68	MODES MENU
69	MODES MISC
70	MEMORY
71	MEM DIR
72	MEM ARITH
73	STACK
74	SOLVE
75	SOLVE ROOT
76	SOLVE DIFFE
77	SOLVE POLY
78	SOLVE SYS
79	SOLVE TVM
80	SOLVE TVM SOLVR
81	PLOT
82	PLOT PTYPE
83	PLOT PPAR
84	PLOT 3D
85	PLOT 3D PTYPE
86	PLOT 3D VPAR

No.	Name
87	PLOT STAT
88	PLOT STAT PTYPE
89	PLOT STAT ΣPAR
90	PLOT STAT ΣPAR MODL
91	PLOT STAT DATA
92	PLOT FLAG
93	SYMBOLIC
94	TIME
95	TIME ALRM
96	STAT
97	STAT DATA
98	STAT ΣPAR
99	STAT ΣPAR MODL
100	STAT 1VAR
101	STAT PLOT
102	STAT FIT
103	STAT SUMS
104	I/O
105	I/O SRVR
106	I/O IOPAR
107	I/O PRINT
108	I/O PRINT PRTPA
109	I/O SERIA
110	LIBRARY Commands
111	LIBRARY PORTS
112	LIBRARY Catalog
113	EQLIB
114	EQLIB EQLIB
115	EQLIB COLIB
116	EQLIB MES
117	EQLIB UTILS

D

System Flags

This appendix lists the HP 48 system flags in functional groups. You can set, clear, and test all flags. The default state of the flags is *clear*—except for the Binary Integer Wordsize flags (flags −5 through −10).

System Flags

Flag	Description
−1	Principal Solution. *Clear*: QUAD and ISOL return a result representing all possible solutions. *Set*: QUAD and ISOL return only the principal solution.
−2	Symbolic Constants. *Clear*: Symbolic constants (e, i, π, MAXR, and MINR) retain their symbolic form when evaluated, unless the Numerical Results flag −3 is set. *Set*: Symbolic constants evaluate to numbers, regardless of the state of the Numerical Results flag −3.
−3	Numerical Results. *Clear*: Functions with symbolic arguments, including symbolic constants, evaluate to symbolic results. *Set*: Functions with symbolic arguments, including symbolic constants, evaluate to numbers.
−4	Not used.
−5 thru −10	Binary Integer Wordsize. Combined states of flags −5 through −10 set the wordsize from 1 to 64 bits.

System Flags (continued)

Flag	Description
−11 and −12	Binary Integer Base. HEX: −11 *set*, −12 *set*. DEC: −11 *clear*, −12 *clear*. OCT: −11 *set*, −12 *clear*. BIN: −11 *clear*, −12 *set*.
−13	Not used.
−14	Financial Payment Mode. *Clear:* TVM calculations assume end-of-period payments. *Set:* TVM calculations assume beginning-of-period payments.
−15 and −16	Rectangular: −16 *clear*. Polar/Cylindrical: −15 *clear*, −16 *set*. Polar/Spherical: −15 *set*, −16 *set*.
−17 and −18	Degrees: −17 *clear*, −18 *clear*. Radians: −17 *set*. Grads: −17 *clear*, −18 *set*.
−19	*Clear*:→V2 and [→][2D] create a 2-dimensional vector from 2 real numbers. *Set*:→V2 and [→][2D] create a complex number from 2 real numbers.
−20	Underflow Exception. *Clear*: Underflow exception returns 0, sets flag −23 or −24. *Set*: Underflow exception treated as an error.
−21	Overflow Exception. *Clear*: Overflow exception returns ±9.99999999999E499 and sets flag −25. *Set*: Overflow exception treated as an error.
−22	Infinite Result Exception. *Clear*: Infinite result exception treated as an error. *Set*: Infinite result exception returns ±9.99999999999E499 and sets flag −26.
−23 −24 −25 −26	Negative Underflow Indicator. Positive Underflow Indicator. Overflow Indicator. Infinite Result Indicator. When an exception occurs, corresponding flag (−23 through −26) is set only if the exception is *not* treated as an error.

System Flags (continued)

Flag	Description
−27	Display of symbolic complex numbers. *Clear*: Displays symbolic complex numbers in coordinate form (i.e. '(x,y)'). *Set*: Displays symbolic complex numbers using 'i' (i.e. 'x+y*i').
−28	Simultaneous Plotting of Multiple Functions. *Clear*: Multiple equations are plotted serially. *Set*: Multiple equations are plotted simultaneously.
−29	Draw Axes. *Clear*: Axes are drawn for two-dimensional and statistical plots. *Set*: Axes are not drawn for two-dimensional and statistical plots.
−30	Not used.
−31	Curve Filling. *Clear*: Curve filling between plotted points enabled. *Set*: Curve filling between plotted points suppressed.
−32	Graphics Cursor. *Clear*: Graphics cursor always dark. *Set*: Graphics cursor dark on light background and light on dark background.
−33	I/O Device. *Clear*: I/O directed to serial port. *Set*: I/O directed to IR port.
−34	Printing Device. *Clear*: Printer output directed to IR printer. *Set*: Printer output directed to serial port if flag −33 is clear.
−35	I/O Data Format. *Clear*: Objects transmitted in ASCII form. *Set*: Objects transmitted in binary (memory image) form.

System Flags (continued)

Flag	Description
−36	I/O Receive Overwrite. *Clear*: If file name received by HP 48 matches existing HP 48 variable name, new variable name with number extension is created to prevent overwrite. *Set*: If file name received by HP 48 matches existing HP 48 variable name, existing variable is overwritten.
−37	Double-Spaced Printing. *Clear*: Single-spaced printing. *Set*: Double-spaced printing.
−38	Line Feed. *Clear*: Linefeed added at end of each print line. *Set*: No linefeed added at end of each print line.
−39	I/O Messages. *Clear*: I/O messages displayed. *Set*: I/O messages suppressed.
−40	Clock Display. *Clear*: Clock is not displayed. *Set*: Ticking clock displayed at all times.
−41	Clock Format. *Clear*: 12-hour clock. *Set*: 24-hour clock.
−42	Date Format. *Clear*: MM/DD/YY (month/day/year) format. *Set*: DD.MM.YY (day.month.year) format.
−43	Repeat Alarms Not Rescheduled. *Clear*: Unacknowledged repeat appointment alarms automatically rescheduled. *Set*: Unacknowledged repeat appointment alarms not rescheduled.
−44	Acknowledged Alarms Saved. *Clear*: Acknowledged appointment alarms deleted from alarm list. *Set*: Acknowledged appointment alarms saved in alarm list.

System Flags (continued)

Flag	Description
−45 thru −48	Number of Decimal Digits. Combined states of flags −45 through −48 sets number of decimal digits in Fix, Scientific, and Engineering modes.
−49 and −50	Number Display Format. Standard: −49 *clear*, −50 *clear*. Fix: −49 *set*, −50 *clear*. Scientific: −49 *clear*, −50 *set*. Engineering: −49 *set*, −50 *set*.
−51	Fraction Mark. *Clear*: Fraction mark is . (period). *Set*: Fraction mark is , (comma).
−52	Single-Line Display. *Clear*: Display gives preference to object in level 1, using up to four lines of stack display. *Set*: Display of object in level 1 restricted to one line.
−53	Precedence. *Clear*: Certain parentheses in algebraic expressions suppressed to improve legibility. *Set*: All parentheses in algebraic expressions displayed.
−54	Tiny Array Elements. *Clear*: Singular values computed by RANK (and other commands that compute the rank of a matrix) that are more than 1×10^{-14} times smaller than the largest computed singular value in the matrix are converted to zero. Automatic rounding for DET is enabled. *Set*: Small computed singular values (see above) not converted. Automatic rounding for DET is disabled.
−55	Last Arguments. *Clear*: Command arguments saved. *Set*: Command arguments not saved.
−56	Error Beep. *Clear*: Error and BEEP-command beeps enabled. *Set*: Error and BEEP-command beeps suppressed.

System Flags (continued)

Flag	Description
−57	Alarm Beep. *Clear*: Alarm beep enabled. *Set*: Alarm beep suppressed.
−58	Verbose Messages. *Clear*: Parameter variable data automatically displayed. *Set*: Automatic display of parameter variable data is suppressed.
−59	Fast Browser Display. *Clear*: Variable Browser shows variable names and contents. *Set*: Variable Browser shows variable names only.
−60	Alpha Lock. *Clear*: Single-Alpha activated by pressing [α] once. Alpha lock activated by pressing [α] twice. *Set*: Alpha lock activated by pressing [α] once. (Single-Alpha not available.)
−61	User-Mode Lock. *Clear*: 1-User mode activated by pressing [←][USER] once. User mode activated by pressing [←][USER] twice. *Set*: User mode activated by pressing [←][USER] once. (1-User mode not available.)
−62	User Mode. *Clear*: User mode not active. *Set*: User mode active.
−63	Vectored [ENTER]. *Clear*: [ENTER] evaluates command line. *Set*: User-defined [ENTER] activated.
−64	Index Wrap Indicator. *Clear*: Last execution of GETI or PUTI did not increment index to first element. *Set*: Last execution of GETI or PUTI did increment index to first element.

Table of Units

HP 48 Units

Unit (Full Name)	Value in SI Units
`a` (Are)	100 m^2
`A` (Ampere)	1 A
`acre` (Acre)	4046.87260987 m^2
`arcmin` (Minute of arc)	$2.90888208666 \times 10^{-4}$ r
`arcs` (Second of arc)	$4.8481368111 \times 10^{-6}$ r
`atm` (Atmosphere)	101325 $kg/m{\cdot}s^2$
`au` (Astronomical unit)	1.495979×10^{11} m
Å (Angstrom)	1×10^{-10} m
`b` (Barn)	1×10^{-28} m^2
`bar` (Bar)	100000 $kg/m{\cdot}s^2$
`bbl` (Barrel)	.158987294928 m^3
`Bq` (Becquerel)	1 1/s
`Btu` (International Table Btu)	1055.05585262 $kg{\cdot}m^2/s^2$
`bu` (Bushel)	.03523907 m^3
`°C` (Degree Celsius)	1 K or 274.15 K
`c` (Speed of light)	299792458 m/s
`C` (Coulomb)	1 A·s
`cal` (Calorie)	4.1868 $kg{\cdot}m^2/s^2$
`cd` (Candela)	1 cd
`chain` (Chain)	20.1168402337 m
`Ci` (Curie)	3.7×10^{10} 1/s
`ct` (Carat)	.0002 kg
`cu` (US cup)	$2.365882365 \times 10^{-4}$ m^3
`°` (Degree)	$1.74532925199 \times 10^{-2}$ r
`d` (Day)	86400 s
`dB` (Decibel)	1

HP 48 Units (continued)

Unit (Full Name)	Value in SI Units
`dyn` (Dyne)	.00001 $kg \cdot m/s^2$
`erg` (Erg)	.0000001 $kg \cdot m^2/s^2$
`eV` (Electron volt)	$1.60217733 \times 10^{-19}$ $kg \cdot m^2/s^2$
`F` (Farad)	1 $A^2 \cdot s^4/kg \cdot m^2$
`°F` (Degrees Fahrenheit)	0.555555555556 K or 255.927777778 K
`fath` (Fathom)	1.82880365761 m
`fbm` (Board foot)	.002359737216 m^3
`fc` (Footcandle)	10.7639104167 $cd \cdot sr/m^2$
`Fdy` (Faraday)	96487 A·s
`fermi` (Fermi)	1×10^{-15} m
`flam` (Footlambert)	3.42625909964 cd/m^2
`ft` (International foot)	.3048 m
`ftUS` (Survey foot)	.304800609601 m
`g` (Gram)	.001 kg
`ga` (Standard freefall)	9.80665 m/s^2
`gal` (US gallon)	.003785411784 m^3
`galC` (Canadian gallon)	.00454609m^3
`galUK` (UK gallon)	.004546092 m^3
`gf` (Gram-force)	.00980665 $kg \cdot m/s^2$
`gmol` (Gram-mole)	1 mol
`grad` (Grade)	$1.57079632679 \times 10^{-2}$ r
`grain` (Grain)	.00006479891 kg
`Gy` (Gray)	1 m^2/s^2
`H` (Henry)	1 $kg \cdot m^2/A^2 \cdot s^2$
`ha`(Hectare)	10000 m^2
`h` (Hour)	3600 s
`hp` (Horsepower)	745.699871582 $kg \cdot m^2/s^3$
`Hz` (Hertz)	1 1/s
`in` (Inch)	.0254 m
`inHg` (Inches of mercury, 0°C)	3386.38815789 $kg/m \cdot s^2$
`inH2O` (Inches of water, 60°F)	248.84 $kg/m \cdot s^2$
`J` (Joule)	1 $kg \cdot m^2/s^2$
`K` (Kelvins)	1 K
`kg` (Kilogram)	1 kg

HP 48 Units (continued)

Unit (Full Name)	Value in SI Units
kip (Kilopound-force)	4448.22161526 kg·m/s^2
knot (Nautical miles per hour)	.514444444444 m/s
kph (Kilometers per hour)	.277777777778 m/s
l (Liter)	.001 m^3
lam (Lambert)	3183.09886184 cd/m^2
lb (Avoirdupois pound)	.45359237 kg
lbf (Pound-force)	4.44822161526 kg·m/s^2
lbmol (Pound-mole)	453.59237 mol
lbt (Troy pound)	.3732417216 kg
lm (Lumen)	1 cd·sr
lx (Lux)	1 cd·sr/m^2
lyr (Light year)	9.46052840488 × 10^{15} m
m (Meter)	1 m
μ (Micron)	1 × 10^{-6} m
mho (Mho)	1 A^2·s^3/kg·m^2
mi (International mile)	1609.344 m
mil (Mil)	.0000254 m
min (Minute)	60 s
miUS (US statute mile)	1609.34721869 m
mmHg (Millimeter of mercury (torr), 0°C)	133.322368421 kg/m·s^2
mol (Mole)	1 mol
mph (Miles per hour)	.44704 m/s
N (Newton)	1 kg·m/s^2
nmi (Nautical mile)	1852 m
Ω (Ohm)	1 kg·m^2/A^2·s^3
oz (Ounce)	.028349523125 kg
ozfl (US fluid ounce)	2.95735295625 × 10^{-5} m^3
ozt (Troy ounce)	.0311034768 kg
ozUK (UK fluid ounce)	2.8413075 × 10^{-5} m^3
P (Poise)	.1 kg/m·s
Pa (Pascal)	1 kg/m·s^2
pc (Parsec)	3.08567818585 × 10^{16} m
pdl (Poundal)	.138254954376 kg·m/s^2
ph (Phot)	10000 cd·sr/m^2

HP 48 Units (continued)

Unit (Full Name)	Value in SI Units
pk (Peck)	.0088097675 m^3
psi (Pounds per square inch)	6894.75729317 $kg/m \cdot s^2$
pt (Pint)	.000473176473 m^3
qt (Quart)	.000946352946 m^3
r (Radian)	1 r
R (Roentgen)	.000258 $A \cdot s/kg$
°R (Degrees Rankine)	0.555555555556 K
rad (Rad)	.01 m^2/s^2
rd (Rod)	5.02921005842 m
rem (Rem)	.01 m^2/s^2
rpm (Revolutions per minute)	.0166666666667 1/s
s (Second)	1 s
S (Siemens)	1 $A^2 \cdot s^3/kg \cdot m^2$
sb (Stilb)	10000 cd/m^2
slug (Slug)	14.5939029372 kg
sr (Steradian)	1 sr
st (Stere)	1 m^3
St (Stokes)	.0001 m^2/s
Sv (Sievert)	1 m^2/s^2
t (Metric ton)	1000 kg
T (Tesla)	1 $kg/A \cdot s^2$
tbsp (Tablespoon)	$1.47867647813 \times 10^{-5}$ m^3
therm (EEC therm)	105506000 $kg \cdot m^2/s^2$
ton (Short ton)	907.18474 kg
tonUK (Long (UK))	1016.0469088 kg
torr (Torr (mmHg))	133.322368421 kg/ms^2
tsp (Teaspoon)	$4.92892159375 \times 10^{-6}$ m^3
u (Unified atomic mass)	$1.6605402 \times 10^{-27}$ kg
V (Volt)	1 $kg \cdot m^2/A \cdot s^3$
W (Watt)	1 $kg \cdot m^2/s^3$
Wb (Weber)	1 $kg \cdot m^2/A \cdot s^2$
yd (International yard)	.9144 m
yr (Year)	31556925.9747 s

Table of Built-In Equations

The Equation Library consists of 15 subjects (corresponding to the sections in the table below) and more than 100 titles. The numbers in parentheses below indicate (*number of equations in the set, number of variables in the set*). There are a total of 315 equations using 396 variables.

Subjects and Titles

1: Columns and Beams (14,20)

1: Elastic Buckling (4,8)
2: Eccentric Columns (2,11)
3: Simple Deflection (1,9)
4: Simple Slope (1,10)
5: Simple Moment (1,8)
6: Simple Shear (1,7)
7: Cantilever Deflection (1,10)
8: Cantilever Slope (1,10)
9: Cantilever Moment (1,8)
10: Cantilever Shear (1,6)

2: Electricity (42,56)

1: Coulomb's Law (1,5)
2: Ohm's Law and Power (4,4)
3: Voltage Divider (1,4)
4: Current Divider (1,4)
5: Wire Resistance (1,4)
6: Series and Parallel R (2,4)
7: Series and Parallel C (2,4)
8: Series and Parallel L (2,4)
9: Capacitive Energy (1,3)
10: Inductive Energy (1,3)
11: RLC Current Delay (5,9)
12: DC Capacitor Current (3,8)
13: Capacitor Charge (1,3)
14: DC Inductor Voltage (3,8)
15: RC Transient (1,6)
16: RL Transient (1,6)
17: Resonant Frequency (4,7)
18: Plate Capacitor (1,4)
19: Cylindrical Capacitor (1,5)
20: Solenoid Inductance (1,5)
21: Toroid Inductance (1,6)
22: Sinusoidal Voltage (2,6)
23: Sinusoidal Current (2,6)

Subjects and Titles (continued)

3: Fluids (29,29)

1: Pressure at Depth (1,4)
2: Bernoulli Equation (10,15)
3: Flow with Losses (10,17)
4: Flow in Full Pipes (8,19)

4: Forces and Energy (31,36)

1: Linear Mechanics (8,11)
2: Angular Mechanics (12,15)
3: Centripetal Force (4,7)
4: Hooke's Law (2,4)
5: 1D Elastic Collisions (2,5)
6: Drag Force (1,5)
7: Law of Gravitation (1,4)
8: Mass-Energy Relation (1,3)

5: Gases (18,26)

1: Ideal Gas Law (2,6)
2: Ideal Gas State Change (1,6)
3: Isothermal Expansion (2,7)
4: Polytropic Processes (2,7)
5: Isentropic Flow (4,10)
6: Real Gas Law (2,8)
7: Real Gas State Change (1,8)
8: Kinetic Theory (4,9)

6: Heat Transfer (17,31)

1: Heat Capacity (2,6)
2: Thermal Expansion (2,6)
3: Conduction (2,7)
4: Convection (2,6)
5: Conduction + Convection (4,14)
6: Black Body Radiation (5,9)

7: Magnetism (4,14)

1: Straight Wire (1,5)
2: Force between Wires (1,6)
3: B Field in Solenoid (1,4)
4: B Field in Toroid (1,6)

8: Motion (22,24)

1: Linear Motion (4,6)
2: Object in Free Fall (4,5)
3: Projectile Motion (5,10)
4: Angular Motion (4,6)
5: Circular Motion (3,5)
6: Terminal Velocity (1,5)
7: Escape Velocity (1,14)

Subjects and Titles (continued)

9: Optics (11,14)

1: Law of Refraction (1,4)
2: Critical Angle (1,3)
3: Brewster's Law (2,4)
4: Spherical Reflection (3,5)
5: Spherical Refraction (1,5)
6: Thin Lens (3,7)

10: Oscillations (17,17)

1: Mass-Spring System (3,5)
2: Simple Pendulum (3,4)
3: Conical Pendulum (4,6)
4: Torsional Pendulum (3,7)
5: Simple Harmonic (4,8)

11: Plane Geometry (31,21)

1: Circle (5,7)
2: Ellipse (5,8)
3: Rectangle (5,8)
4: Regular Polygon (6,8)
5: Circular Ring (4,7)
6: Triangle (6,10)

12: Solid Geometry (18,12)

1: Cone (5,9)
2: Cylinder (5,9)
3: Parallelepiped (4,9)
4: Sphere (4,7)

13: Solid State Devices (33,53)

1: PN Step Junctions (8,19)
2: NMOS Transistors (10,23)
3: Bipolar Transistors (8,14)
4: JFETs (7,15)

14: Stress Analysis (16,28)

1: Normal Stress (3,7)
2: Shear Stress (3,8)
3: Stress on an Element (3,7)
4: Mohr's Circle (7,10)

15: Waves (12,15)

1: Transverse Waves (4,9)
2: Longitudinal Waves (4,9)
3: Sound Waves (4,8)

G

Operation Index

This index contains reference information for all operations in the HP 48. For each operation, this index shows:

- **Name.** The name associated with the operation. Operations that can be included in programs (commands) are shown in all uppercase.
- **Description.** What the operation does. If the operation expects arguments from the stack, the description includes the variables which represent arguments in levels 1 (x), 2 (y), 3 (z), 4 (t) and 5 (v).
- **Type.** The type of operation is given by one of the following codes:

Code	Description
O	**Operation.** Any action built into the calculator that is represented by a name or a key.
C	**Command.** Any programmable operation.
F	**Function.** Any command that can be included in algebraic objects.
A	**Analytic Function.** A function for which the HP 48 provides an inverse and derivative.

- **Keys.** The keys to access the operation. Operations that aren't key-accessible are identified by "Must be typed in."
- **Page.** Where to find a description of the operation.

Operations whose names contain both alpha and special characters are listed alphabetically. Operations whose names contain special characters only are listed at the end of the index.

Name, Key, or Label	Description, Type, and Keys	Page
←A	Associate left. O ↰ EQUATION ◀ RULES ←A	20-23
↱ ←A	Executes ←A until no change. O ↰ EQUATION ◀ RULES ↱ ←A	20-26
A→	Associate right. O ↰ EQUATION ◀ RULES →A	20-23
↱ A→	Executes A→ until no change. O ↰ EQUATION ◀ RULES ↱ A→	20-26
ABS	Absolute value of an object (x). MTH REAL NXT ABS MTH MATR NORM ABS MTH NXT CMPL ABS F MTH VECTR ABS	12-9
ACK	Acknowledges displayed past due alarm. C ↰ TIME ALRM ACK	26-4
ACKALL	Acknowledges all past due alarms. C ↰ TIME ALRM ACKA	26-4
ACOS	Arc cosine of a number (x). A ↰ ACOS	12-2
ACOSH	Arc hyperbolic cosine of a number (x). A MTH HYP ACOSH	12-3
ADD	Adds two lists (x and y) element by element. C MTH LIST ADD	17-3
AF	Add fractions. O ↰ EQUATION ◀ RULES AF	20-25
ALOG	Common (base 10) antilogarithm of a number (x). A ↰ 10^x	12-2
AMORT	Calculates the amount of principal, interest, and balance for a number of payments for the financial scenario currently stored in the TVM variables (*%I/YR*, *PMT*, *FV*, and *PV*). O ↱ SOLVE **Solve finance...** AMOR AMOR	18-19

G

Name, Key, or Label	Description, Type, and Keys	Page
AMORT	Calculates the amount of principal, interest, and balance for a number (x) of payments for the financial scenario currently stored in the TVM variables (*%I/YR*, *PMT*, *FV*, and *PV*). C ◀ SOLVE TVM AMOR	H-1
AMRT	Calculates the amount of principal, interest, and balance for a number (x) of payments for the financial scenario entered in the interactive solver. O ◀ SOLVE TVM SOLVR AMRT	18-19
AND	Logical AND of two expressions (x and y) that evaluate to 1 or 0 or binary AND combining two integers (x and y) or two strings (x and y). MTH BASE NXT LOGIC AND F PRG TEST NXT AND	15-4 H-1
ANIMATE	Successively displays a specified number (x) of GROBs (y, z ...) that are on the stack. C PRG GROB NXT ANIM	9-10
APPLY	Creates an unevaluated expression combining an unevaluated function name (x) with a list (y) of evaluated arguments. F ◀ SYMBOLIC NXT APPLY	H-1
ARC	Draws arc in *PICT* counterclockwise from one angle (y) to another (x) along the circle with a center at z and radius t. C PRG PICT ARC	9-8
ARCHIVE	Makes backup copy of HOME directory. C ◀ MEMORY NXT ARCHI	28-5
AREA	Calculates and displays area under function graph between two x-values specified by the mark and cursor; returns area to stack. O ◀ PICTURE FCN AREA	22-10
ARG	Returns polar angle (θ) of a complex number (x). F MTH NXT CMPL ARG	12-13
ARRY→	Returns array elements to stack. C Must be typed in.	H-2

Name, Key, or Label	Description, Type, and Keys	Page
→ARRY	Combines numbers into array. C [PRG] TYPE →ARR	14-4
ASIN	Arc sine of a number (x). A [↩] [ASIN]	12-2
ASINH	Arc hyperbolic sine of number (x). A [MTH] HYP ASINH	12-3
ASN	Assigns an object (y) to a user-key (x). C [↩] [MODES] KEYS ASN	30-5
ASR	Shifts a binary integer (x) one bit to the right. C [MTH] BASE [NXT] BIT ASR	15-5
ATAN	Arc tangent of a number (x). A [↩] [ATAN]	12-2
ATANH	Arc hyperbolic tangent of a number (x). A [MTH] HYP ATAN	12-3
ATICK	Sets the axes tick mark annotation by a list (x) containing the axes tick mark interval in user-units or pixels. C [↩] [PLOT] PPAR [NXT] ATICK	H-2
ATTACH	Attaches a library (x) to current directory. C [↩] [LIBRARY] [NXT] ATTAC	28-8
AUTO	Auto-scales y-axis. C [↩] [PLOT] [NXT] AUTO	
AXES	Specifies the plot axes using a list (x) containing the coordinates of axes intersection, the tick mark interval, or the axes labels, or any combination of the above. C [↩] [PLOT] PPAR [NXT] AXES	24-1
BAR	Selects BAR plot type. C [↩] [PLOT] [NXT] STAT PTYPE BAR	23-19
BARPLOT	Draws bar plot of data in ΣDAT. C [↩] [STAT] PLOT BARPL	21-7
BAUD	Sets the baud rate to x. C [↩] [I/O] IOPAR BAUD	27-15

G

Name, Key, or Label	Description, Type, and Keys	Page
BEEP	Sounds beep at a frequency (y Hz) for x seconds. C [PRG] [NXT] OUT [NXT] BEEP	4-10
BEG	Toggles payment mode between Beginning-, or End-, of-Month. See TVMBEG and TVMEND for the command form of this operation. O [◀] [SOLVE] TVM BEG	
BESTFIT	Selects statistics model yielding largest correlation coefficient (absolute value) and executes LR. C [◀] [STAT] ΣPAR MODL BESTF	
BIN	Sets binary base. C [MTH] BASE BIN	15-1
BINS	Sorts elements in independent variable column of ΣDAT into a number ($x + 2$) of "bins" of a designated width (y), starting with a minimum data value (z). C [◀] [STAT] 1VAR BINS	H-2
BLANK	Creates blank graphics object, y pixels wide by x pixels high. C [PRG] GROB BLAN	9-9
BOX	Draws box with opposite corners defined by coordinates x and y. C [PRG] PICT BOX	9-8
BOX	Draws box with opposite corners defined by mark and cursor. O [◀] [PICTURE] EDIT BOX	9-3
BOXZ	Toggles expanding-box drawing mode on and off. O [◀] [PICTURE] ZOOM BOXZ	22-7
BUFLEN	Returns number of characters in serial buffer. C [◀] [I/O] [NXT] SERIA BUFLE	27-19
BYTES	Returns object size (in bytes) and checksum for an object (x). C [◀] [MEMORY] BYTES	H-2
B→PV	Transfers the remaining balance after an amortization to the PV variable in preparation for amortizing the next group of payments. O [▶] [SOLVE] Solve finance... AMOR B→PV	18-20

G

Name, Key, or Label	Description, Type, and Keys	Page
B→R	Converts a binary integer (x) to a real number. C (MTH) BASE B→R	15-3
CALC	Copies the contents of the current field to the stack and displays the stack. Used to make side-calculations or take side-trips to other parts of the calculator while working within an input form. O *[input form]* (NXT) CALC	6-5
CANCL	Erases the command line and cancels the entry in process or exits an input form without executing its main action. *[input form]* (CANCEL) O *[input form]* CANCL	6-7
CASE	Begins CASE structure. C (PRG) BRCH CASE CASE	29-11
◀CASE	Types CASE THEN END END. O (PRG) BRCH ◀ CASE	29-11
▶CASE	Types THEN END. O (PRG) BRCH ▶ CASE	29-11
CEIL	Returns next integer greater than x. F (MTH) REAL (NXT) (NXT) CEIL	12-9
CENTR	Sets center of plot display at specified (x, y) coordinates. C ◀ (PLOT) PPAR (NXT) CENT	H-3
CF	Clears flag x. (PRG) TEST (NXT) (NXT) CF C ◀ (MODES) FLAG CF	4-8
%CH	Returns % change from y to x. F (MTH) REAL %CH	12-9
✓CHK	Toggles the checkmark field on and off. O *[input form]* ✓CHK	6-4
CHOOS	Displays a choose box with additional possible entries relevant to the current field. O *[input form]* CHOOS	6-3

G

Name, Key, or Label	Description, Type, and Keys	Page
CHOOSE	Creates a user-defined choose box from a title string (z), a list of objects (y), and the number (x) of the default object to be highlighted. C [PRG] [NXT] IN CHOOS	H-3
CHR	Converts a character code (x) to one-character string. C [PRG] TYPE [NXT] CHR	H-3
CIRCL	Draws circle with center at the mark and radius equal to the distance from cursor to mark. O [←] [PICTURE] EDIT CIRCL	9-4
CKSM	Selects checksum error-detection scheme (x). C [←] [I/O] IOPAR CKSM	H-3
CLEAR	Clears stack. C [CLEAR] or [←] [CLEAR]	3-5
CLK	Turns the clock display on and off. O [←] [MODES] MISC CLK	4-10
CLKADJ	Adds x clock ticks (1 tick = $\frac{1}{8192}$ second) to system time. C [←] [TIME] [NXT] [NXT] CLKA	H-3
CLLCD	Blanks stack *display* (but doesn't clear the stack itself). C [PRG] [NXT] OUT CLLCD	
CLOSEIO	Closes I/O port. C [←] [I/O] [NXT] CLOSE	
CLΣ	Purges statistical data in ΣDAT. C [←] [STAT] DATA CLΣ	
CLUSR	Purges all user variables in current directory. C Must be typed in.	
CLVAR	Purges all user variables in current directory. C Must be typed in.	
CNCT	Toggles the flag controlling whether plotted points are connected with line segments. O [←] [PLOT] [NXT] FLAG CNCT	
CNRM	Calculates column norm of an array (x). C [MTH] MATR NORM CNRM	14-8

Name, Key, or Label	Description, Type, and Keys	Page
CNTR	Redraws graph with center at current cursor position. O ◀ PICTURE ZOOM NXT CNTR	22-8
→COL	Transforms a matrix (x) into a series of column vectors. C MTH MATR COL →COL	14-4
+COL	Inserts a row of zeros at current column in MatrixWriter application. O ▶ MATRIX NXT +COL	8-8
COL+	Inserts a column vector (y) into an array (z) as the column x. C MTH MATR COL COL+	14-5
−COL	Deletes current column in MatrixWriter application. O ▶ MATRIX NXT COL−	8-8
COL−	Deletes column x from array y. C MTH MATR COL −COL	14-6
COL→	Transforms a series of x column vectors (y, z, etc.) into a matrix containing those vectors as its columns. C MTH MATR COL COL→	14-3
COLΣ	Specifies dependent and independent columns in ΣDAT. C Must be typed in.	H-4
COLCT	Collects like terms in expression (x). C ◀ SYMBOLIC COLCT	20-18
COLCT	Collects like terms in specified subexpression. O ◀ EQUATION ◀ RULES NXT COLCT	20-21
COLCT	Collects like terms in the expression in the current field. O ▶ SYMBOLIC Manip expr... COLCT	20-28
COMB	Returns number of combinations of y items taken x at a time. F MTH NXT PROB COMB	12-4

G

Name, Key, or Label	Description, Type, and Keys	Page
CON	Creates a constant array from a list of dimensions (y) and the constant number (x). C (MTH) MATR MAKE CON	14-2
COND	Estimates the condition number of a square matrix (x). C (MTH) MATR NORM COND	14-9
CONIC	Selects CONIC plot type. C (←) (PLOT) PTYPE CONIC	23-11
CONJ	Returns complex conjugate of x. F (MTH) (NXT) CMPL (NXT) CONJ	12-13
CONLIB	Opens the Constants Library catalog. C (←) (EQ LIB) COLIB CONLI	25-13
CONST	Returns the value of the specified constant (x). F (←) (EQ LIB) COLIB CONS	25-14
CONT	Continues halted program. C (←) (CONT)	29-9
CONVERT	Converts one unit object (y) to dimensions of a different compatible unit (x). C (←) (UNITS) CONV	10-6
COPY	Copies the highlighted object to a new location. O (→) (MEMORY) COPY	5-9
CORR	Calculates correlation coefficient of statistical data in ΣDAT. C (←) (STAT) FIT CORR	H-4
COS	Cosine of an angle (x). A (COS)	12-2
COSH	Hyperbolic cosine of an angle (x). A (MTH) HYP COSH	12-3
COV	Calculates covariance of statistical data in ΣDAT. C (←) (STAT) FIT COV	H-4
CR	Causes printer to do carriage return/line feed. C (←) (I/O) PRINT CR	

G

Name, Key, or Label	Description, Type, and Keys	Page
CRDIR	Creates a directory named x. C ◀ MEMORY DIR CRDIR	H-4
CROSS	Cross product of two vectors ($y \times x$). C MTH VECTR CROSS	13-4
CST	Returns contents of *CST* variable. C ◀ MODES MENU CST	30-1
CSWP	Swaps column y with column x of a matrix (z). C MTH MATR COL CSWP	14-6
CYLIN	Selects cylindrical vector mode. C MTH VECTR NXT CYLIN	13-2
C→PX	Converts user-unit coordinates (x) to pixel coordinates. C PRG PICT NXT C→PX	9-9
C→R	Separates a complex number (x) into two real numbers. MTH NXT CMPL C→R C PRG TYPE NXT C→R	12-14
←D	Distribute left. O ◀ EQUATION ◀ RULES ←D	20-24
▶ ←D	Executes ←D until no change in subexpression. O ◀ EQUATION ◀ RULES ▶ ←D	20-26
D→	Distribute right. O ◀ EQUATION ◀ RULES D→	20-24
▶ D→	Executes D→ until no change in subexpression. O ◀ EQUATION ◀ RULES ▶ D→	20-26
DARCY	Calculates the Darcy friction factor for a fluid flow as a function of the relative roughness of the pipe (y) and the Reynolds number (x) of the flow. F ◀ EQLIB UTILS DARCY	H-4
ΣDAT	Returns the contents of ΣDAT reserve variable to the stack. C ◀ STAT DATA ΣDAT	21-1

Name, Key, or Label	Description, Type, and Keys	Page
DATE	Returns system date. C ◀ TIME DATE	16-2
DATE+	Adds or subtracts a number of days (x) from a date (y). C ◀ TIME NXT DATE+	16-2
→DATE	Sets system date to a specified date (x). C ◀ TIME →DAT	H-5
DBUG	Halts program (x) execution before first object. O PRG NXT RUN DBUG	29-8
DDAYS	Returns number of days between date$_1$ (y) and date$_2$ (x). C ◀ TIME NXT DDAYS	16-2
DEC	Sets decimal base. C MTH BASE DEC	15-1
DECR	Decrements value of a variable (x) by 1. C ◀ MEMORY ARITH DECR	H-5
DEFINE	Creates variable or user-defined function from equation (x). C ◀ DEF	5-13 11-7
→DEF	Expands trigonometric and hyperbolic functions in terms of EXP and LN. O ◀ EQUATION ◀ RULES →DEF	20-25
DEG	Sets Degrees mode. C ◀ MODES ANGL DEG	4-4
DEL	Erases area whose opposite corners are defined by mark and cursor. ◀ PICTURE DEL O ◀ PICTURE EDIT NXT DEL	9-4
←DEL	Deletes all characters from cursor to start of word. O ◀ EDIT ←DEL	2-12
▶ ←DEL	Deletes all characters from cursor to start of line. O ◀ EDIT ▶ ←DEL	2-12

G

Name, Key, or Label	Description, Type, and Keys	Page
DEL→	Deletes all characters from cursor to start of next word. ◀ EDIT DEL→ O ... EDIT DEL→	2-12
▶ DEL→	Deletes all characters from cursor to end of line. ◀ EDIT ▶ DEL→ O ... EDIT ▶ DEL→	2-12
DELALARM	Deletes alarm (x) from system alarm list. C ◀ TIME ALRM DELAL	H-5
DELAY	Sets delay time (x seconds) between printing lines. C ◀ I/O PRINT PRTPA DELAY	27-4
DELKEYS	Clears specified user-key assignment for one or more keys (x). C ◀ MODES KEYS DELK	30-6
DEPND	Specifies name (x) of dependent plot variable. C ◀ PLOT PPAR DEPN	H-5
DEPTH	Returns number of objects on stack. C ◀ STACK DEPTH	3-11
DET	Finds the determinant of a square matrix (x). C MTH MATR NORM NXT DET	14-9
DETACH	Detaches specified library (x) from current directory. C ◀ LIBRARY DETAC	28-8
→DIAG	Returns the major diagonal vector of a matrix (x). C MTH MATR NXT →DIAG	14-5
DIAG→	Creates a matrix from a vector of diagonal elements (y) and a list of dimensions (x). C MTH MATR NXT DIAG→	14-3
DIFFEQ	Selects the differential equation plot type. C ◀ PLOT PTYPE DIFFE	23-11
DINV	Double invert. O ◀ EQUATION ◀ RULES DINV	20-21
DISP	Displays an object (y) in the xth display line. C PRG NXT OUT DISP	H-5

G

Name, Key, or Label	Description, Type, and Keys	Page
DNEG	Double negate. O ◀ EQUATION ◀ RULES DNEG	20-21
DO	Begins indefinite loop. C PRG BRCH DO DO	29-14
◀ DO	Types DO UNTIL END. O PRG BRCH ◀ DO	29-14
DOERR	Aborts program execution and displays specified message (x). C PRG NXT ERROR DOERR	H-6
DOLIST	Executes a program or command (x) on a specified number of lists (y) that are on the stack. C PRG LIST PROC DOLIS	17-4
DOSUBS	Executes a program or command (x) on a specified number of elements at a time (y) within a list (z). C PRG LIST PROC DOSUB	17-5
DOT	Dot product ($y \cdot x$) of two vectors. C MTH VECTR DOT	13-4
DOT+	Turns on pixels as cursor moves. O ◀ PICTURE EDIT DOT+	9-3
DOT−	Turns off pixels as cursor moves. O ◀ PICTURE EDIT DOT−	9-3
DRAW	Plots equation without axes. C ◀ PLOT DRAW	
DRAW	Plots function as specified in current input form. O [*plot input form*] DRAW	22-1
DRAX	Draws axes. C ◀ PLOT DRAX	
DROP	Drops object (x) in level 1; moves all remaining objects down one level. C ◀ DROP	3-5
DROPN	Drops x objects from stack. C ◀ STACK NXT DRPN	3-11

G

Name, Key, or Label	Description, Type, and Keys	Page
DRPN	Drops all objects from stack at and below the current stack pointer. O ▶ STACK NXT DRPN	3-8
DROP2	Drops first two objects (y and x) from stack. C ◀ STACK NXT DROP2	3-11
DTAG	Removes all tags from object (x). C PRG TYPE NXT DTAG	H-6
DUP	Duplicates object (x). C ENTER (when no command line) or ◀ STACK NXT DUP	3-4
DUPN	Duplicates x objects on stack. C ◀ STACK NXT DUPN	3-11
DUPN	Duplicates all objects on stack from current stack pointer through level 1. O ▶ STACK NXT DUPN	3-8
DUP2	Duplicates objects in level 1 and level 2. C ◀ STACK NXT DUP2	3-11
D→R	Degrees-to-radians conversion. F MTH REAL NXT NXT D→R	12-7
e	Returns the symbolic constant e (or 2.71828182846 depending on flag −2). F α ◀ E MTH NXT CONS E	11-4
ECHO	Copies object in current stack level to command line. O ▶ STACK ECHO	3-7
EDIT	Copies selected object into command line and selects EDIT menu. O [*input form*] EDIT	6-4
EDIT	Copies subexpression into command line and selects EDIT menu. O ◀ EQUATION ◀ EDIT	7-11
EDIT	Edits current matrix cell. O ▶ MATRIX EDIT	8-8

Name, Key, or Label	Description, Type, and Keys	Page
EEX	Types E or moves cursor to existing exponent in command line. O (EEX)	2-2
EGV	Computes the right eigenvectors and the eigenvalues for a square matrix (x). C (MTH) MATR (NXT) EGV	14-21
EGVL	Computes the eigenvalues of a square matrix (x). C (MTH) MATR (NXT) EGVL	14-20
ELSE	Begins false clause. (PRG) BRCH IF ELSE C (PRG) (NXT) ERROR IFERR ELSE	29-10
END	Ends program structures. (PRG) BRCH IF END (PRG) BRCH CASE END (PRG) BRCH DO END (PRG) BRCH WHILE END C (PRG) (NXT) ERROR IFERR END	29-10
ENDSUB	The frame index counter for DOSUBS. C (PRG) LIST PROC ENDS	17-5
ENG	Sets display mode to Engineering, displaying $x + 1$ significant digits. C (←) (MODES) FMT ENG	4-2
EQ	Returns the contents of the reserved variable, EQ. (←) (PLOT) (NXT) 3D EQ C (←) (PLOT) EQ	22-12
EQ→	Separates equation (x) into left and right sides. C (PRG) TYPE (NXT) EQ→	H-6
EQNLIB	Starts the Equation Library. C (←) (EQ LIB) EQLIB EQNLI	
ERASE	Erases $PICT$. (←) (PICTURE) EDIT (NXT) ERASE (←) (PICTURE) (←) (CLEAR) C (←) (PLOT) ERASE	22-1

G

Name, Key, or Label	Description, Type, and Keys	Page
ERASE	Erases *PICT*. O → PLOT ERASE	22-1
ERRM	Returns last error message. C PRG NXT ERROR ERRM	H-6
ERRN	Returns last error number. C PRG NXT ERROR ERRN	H-7
ERR0	Clears last error number. C PRG NXT ERROR ERR0	
EVAL	Evaluates object (x). C EVAL	7-13
EXIT	Exits EquationWriter options environment. O ← EQUATION ◀ EXIT	20-20
EXP	Constant e raised to power of object (x). A ← e^x	12-2
EXPAN	Expands algebraic object (x). C ← SYMBOLIC EXPA	20-18
EXPND	Expands the algebraic object in the current field. O → SYMBOLIC Manip expr... EXPN	20-28
EXPFIT	Sets curve-fitting model to exponential. C ← STAT ΣPAR MODL EXPFI	
EXPM	Natural exponential minus 1 ($e^x - 1$). A MTH HYP NXT EXPM	12-3
EXPR	Highlights subexpression for which specified object is top level function. O ← EQUATION ◀ EXPR	7-12 20-19
EXTR	Moves graphics cursor to nearest extremum, displays coordinates, and returns them to stack. O ← PICTURE FCN EXTR	22-11
EYEPT	Specifies the x (z), y (y), and z (x) coordinate of the eye-point in a perspective plot. C ← PLOT NXT 3D VPAR NXT EYEPT	H-7

G

Name, Key, or Label	Description, Type, and Keys	Page
E^	Replace power-product with power-of-power. O ◀ EQUATION ◀ RULES E^	20-25
E()	Replace power-of-power with power-product. O ◀ EQUATION ◀ RULES E()	20-25
F0λ	Calculates the fraction of total black-body emissive power at a given temperature (x) between wavelengths 0 to λ (y). F ◀ EQ LIB UTILS F0λ	H-7
FACT	Finds the factorial of x. Same as !. C Must be typed in.	H-7
FANNING	Calculates the Fanning friction factor for a fluid flow as a function of the relative roughness of the pipe (y) and the Reynolds number (x) of the flow. F ◀ EQ LIB UTILS FANNI	H-8
FC?	Tests if specified flag (x) is clear. PRG TEST NXT NXT FC? C ◀ MODES FLAG FC?	4-8
FC?C	Tests if specified flag (x) is clear, then clears it. PRG TEST NXT NXT FC?C C ◀ MODES FLAG FC?C	4-8
FFT	Computes the Discrete Fourier Transform of an array (x). C MTH NXT FFT FFT	13-8
FINDALARM	Returns first alarm due after specified time (x). C ◀ TIME ALRM FINDA	H-8
FINISH	Terminates Kermit server mode. C ◀ I/O SRVR FINIS	27-10
FIX	Selects Fix display mode to x decimal places. C ◀ MODES FMT FIX	4-2
FLOOR	Returns the next integer less than x. F MTH REAL NXT NXT FLOOR	12-9
FM,	Selects comma as the decimal place marker. O ◀ MODES FMT FM,	4-10

Name, Key, or Label	Description, Type, and Keys	Page
FOR	Begins a definite loop with beginning (y) and ending (x) loop-counter values. C [PRG] BRCH FOR FOR	29-13
[←]FOR	Types FOR NEXT. O [PRG] BRCH [←] FOR	29-13
[→]FOR	Types FOR STEP. O [PRG] BRCH [→] FOR	29-13
FP	Returns fractional part of a number (x). F [MTH] REAL [NXT] FP	12-9
FREE1	Frees the previously merged RAM in port 1 and moves a list of objects (x) from port 0 to port 1. C [←][LIBRARY] FREE1	28-16
FREEZE	Freezes a display area (x) until a key is pressed. C [PRG][NXT] OUT FREEZ	H-8
FS?	Tests if specified flag (x) is set. [PRG] TEST [NXT][NXT] FS? C [←][MODES] FLAG FS?	4-8
FS?C	Tests if specified flag (x) is set, then clears it. [PRG] TEST [NXT][NXT] FS?C C [←][MODES] FLAG FS?C	4-8
FUNCTION	Selects FUNCTION plot type. C [←][PLOT] PTYPE FUNC	23-1
FV	Sets the future value amount for the amortization solver. C [←][SOLVE] TVM SOLVR FV	18-16
F(X)	Displays value of function at x-value specified by cursor. Returns function value to stack. O [←][PICTURE] FCN [NXT] F(X)	22-11
F'	Plots first derivative of function, replots function, and adds derivative to EQ. O [←][PICTURE] FCN [NXT] F'	22-11
GET	Gets element in specified position (x) from array or list (y). C [PRG] LIST ELEM GET	14-7

Name, Key, or Label	Description, Type, and Keys	Page
GETI	Gets element in specified position (x) from array or list (y) and increments index. C [PRG] LIST ELEM GETI	17-6
GOR	Superimposes one graphics object (x) onto another (z) at specified coordinates (y), using a logical OR to determine the state of the pixels. C [PRG] GROB GOR	9-9
GO↓	Sets top-to-bottom entry mode. O [→] [MATRIX] GO↓	8-8
GO→	Sets left-to-right entry mode. O [→] [MATRIX] GO→	8-8
GRAD	Selects Grads angle mode. C [←] [MODES] ANGL GRAD	4-4
GRAPH	Enters Graphics environment. This is for compatibility only. C Must be typed in.	
GRIDMAP	Selects the GRIDMAP plot type. C [←] [PLOT] [NXT] 3D PTYPE GRID	23-34
→GROB	Converts object (y) into graphics object of a specified size (x). C [PRG] GROB →GRO	9-9
GXOR	Superimposes one graphics object (x) onto another (z) at the specified location (y), using the logical XOR to determine the state of the pixels. C [PRG] GROB GXOR	9-9
*H	Multiplies the vertical plot scale by a factor (x). C [←] [PLOT] PPAR [NXT] *H	H-8
HALT	Halts program execution. C [PRG] [NXT] RUN HALT	29-9
HEAD	Gets the first element from a list (x). C [PRG] LIST ELEM [NXT] HEAD	H-9

Name, Key, or Label	Description, Type, and Keys	Page
HEX	Sets hexadecimal base. C MTH BASE HEX	15-1
HISTOGRAM	Selects HISTOGRAM plot type. C ◀ PLOT NXT STAT PTYPE HISTO	23-18
HISTPLOT	Draws histogram of data in ΣDAT. C ◀ STAT PLOT HISTP	
HMS+	Adds two times (y and x) in HMS format. C ◀ TIME NXT HMS+	12-7 16-4
HMS−	Subtracts one time (x) from another (y) in HMS format. C ◀ TIME NXT HMS−	12-7 16-4
HMS→	Converts a time (x)from HMS to decimal format. C ◀ TIME NXT HMS→	12-7 16-3
→HMS	Converts a time (x) from decimal to HMS format. C ◀ TIME NXT →HMS	12-7 16-3
HOME	Makes the *HOME* directory the current directory. C ▶ HOME	5-12
HZIN	Zooms in horizontally. O ◀ PICTURE ZOOM NXT HZIN	22-8
HZOUTN	Zooms out horizontally. O ◀ PICTURE ZOOM NXT HZOUT	22-8
i	Returns the symbolic constant i ($\sqrt{-1}$ or (0,1)). F α ◀ I or MTH NXT CONS I	11-4
IDN	Creates square identity matrix of specified size (x). C MTH MATR MAKE IDN	14-2
IF	Begins test clause. C PRG BRCH IF IF	29-10
◀ IF	Types IF THEN END. O PRG BRCH ◀ IF	29-10
▶ IF	Types IF THEN ELSE END. O PRG BRCH ▶ IF	29-10

G

Name, Key, or Label	Description, Type, and Keys	Page
IFERR	Begins test clause. C [PRG] [NXT] ERROR IFERR IFERR	29-15
◀IFERR	Types IFERR THEN END. O [PRG] [NXT] ERROR ◀IFERR	29-15
▶IFERR	Types IFERR THEN ELSE END. O [PRG] [NXT] ERROR ▶IFERR	29-15
IFFT	Computes the Inverse Discrete Fourier Transform of an array (x). C [MTH] [NXT] FFT IFFT	13-8
IFT	Evaluates an object (x) if a test value (y) is a non-zero real number. C [PRG] BRCH [NXT] IFT	H-9
IFTE	Evaluates one object (y) if a test value (z) is a non-zero real number or another object (x) if the test value is zero. F [PRG] BRCH [NXT] IFTE	H-9
IM	Returns imaginary part of complex number or array (x). F [MTH] [NXT] CMPL IM	12-14
INCR	Increments value of specified variable (x). C [◀] [MEMORY] ARITH INCR	H-9
INDEP	Specifies independent variable (x) in a plot. C [◀] [PLOT] PPAR INDEP	H-10
INFO	Displays information about reserved variables. O [◀] [PLOT] [NXT] INFO [◀] [PLOT] PPAR [NXT] [NXT] INFO [◀] [PLOT] [NXT] 3D VPAR INFO [◀] [PLOT] [NXT] 3D VPAR [NXT] INFO [◀] [PLOT] [NXT] STAT ΣPAR INFO [◀] [STAT] ΣPAR INFO [◀] [I/O] IOPAR [NXT] INFO [◀] [I/O] PRINT PRTPA INFO	27-15

Name, Key, or Label	Description, Type, and Keys	Page
INFO	Displays information about the most recent computation by the root-finder. O → SOLVE SOLVE INFO	18-3
INFO?	Toggles the automatic display of parameter variable information. O ← MODES MISC NXT INFO?	4-10
INFORM	Displays a user-defined input form. C PRG NXT IN INFOR	6-8
INIT+	Stores the values of a differential equation solution as the new initial values to prepare for another iteration. O → SOLVE Solve diff eq... INIT+	19-2
INPUT	Suspends program execution, displays message (y) above the stack, and prompts (x) for data input to the command line. C PRG NXT IN INPUT	H-10
INS	Switches between insert/replace character. O ← EDIT INS	2-12
INV	Reciprocal of a number or array (x). A $1/x$	12-1 14-10
IOPAR	Returns the contents of the *IOPAR* reserved variable. C ← I/O IOPAR NXT IOPAR	27-3
IP	Integer part of real number of x. F MTH REAL NXT IP	12-10
ISECT	Moves graphics cursor to closest intersection in two-function plot, displays intersection coordinates, and returns coordinates to stack. O ← PICTURE FCN ISECT	22-10
ISOL	Isolates variable (x) on one side of an equation (y). C ← SYMBOLIC ISOL	20-15
KEEP	Clears all levels above current level. O → STACK NXT KEEP	3-8

Name, Key, or Label	Description, Type, and Keys	Page
KERRM	Returns text of most recently-received KERMIT error packet. C ◀ I/O NXT KERR	H-10
KEY	Returns number indicating last key pressed. C PRG NXT IN KEY	H-10
KGET	Gets a list of objects (x) from another device. C ◀ I/O SRVR KGET ▶ I/O Transfer... KGET	27-10
KILL	Aborts all suspended programs. C PRG NXT RUN KILL	29-9
LABEL	Labels axes with variable names and ranges. C ◀ PLOT NXT LABEL	
LABEL	Labels axes with variable names and ranges. O ◀ PICTURE EDIT NXT LABEL	24-1
LAST	Returns previous argument(s) to stack. C Must be keyed in.	
LASTARG	Returns previous argument(s) to stack. C PRG NXT ERROR LASTA	3-5
LCD→	Returns graphics object to stack representing stack display. C PRG GROB NXT LCD→	9-10
→LCD	Displays specified graphics object (x) in stack display. C PRG GROB NXT →LCD	9-10
LEVEL	Enters current level number into level 1. O ▶ STACK NXT LEVEL	3-8
LIBEVAL	Evaluates a system library object (x). *Use only as specified by HP applications.* C Must be typed in.	H-11
LIBS	Lists all libraries attached to current directory. C ◀ LIBRARY LIBS	H-11
LINE	Draws line between two coordinates (x and y). C PRG PICT LINE	9-8

Name, Key, or Label	Description, Type, and Keys	Page
LINE	Draws line from mark to cursor. O ◀ PICTURE EDIT LINE	9-3
ΣLINE	Returns best-fit line for data in ΣDAT according to selected statistical model. C ◀ STAT FIT ΣLINE	H-11
LINFIT	Sets curve-fitting model to linear. C ◀ STAT ΣPAR MODL LINFI	
LININ	Tests whether an expression (x) is a linear function of a variable (y). F PRG TEST ◀ PREV LININ	H-11
LIST→	Disassembles a list (x) into its constituent elements. C Must be typed in.	H-11
→LIST	Combines x objects (y, z, etc.) into a list. PRG TYPE →LIST C PRG LIST →LIST	17-1
→LIST	Combines objects from level 1 to current level into a list. O ▶ STACK →LIST	3-8
ΣLIST	Adds together all of the elements in a list (x). C MTH LIST ΣLIST	17-8
ΠLIST	Multiplies together all of the elements in a list (x). C MTH LIST πLIST	17-8
ΔLIST	Finds the set of first differences of a finite sequence in a list (x). C MTH LIST ΔLIST	17-8
LN	Natural (base e) logarithm of x. A ▶ LN	12-2
LNP1	Natural logarithm of $(x + 1)$. A MTH HYP NXT LNP1	12-3
LOG	Common (base 10) logarithm of x. A ▶ LOG	12-2

Name, Key, or Label	Description, Type, and Keys	Page
LOGFIT	Set curve-fitting model to logarithmic. C ◀ STAT ΣPAR MODL LOGFI	
LQ	Returns the LQ factorization of a matrix (x). C MTH MATR FACTR LQ	14-21
LR	Calculates linear regression. C ◀ STAT FIT LR	H-12
LSQ	Computes the minimum norm least squares solution to an under- or over-determined system of linear equations $\mathbf{AX} = \mathbf{B}$, where $\mathbf{A}$ (y) is the matrix of coefficients and $\mathbf{B}$ (x) is the vector of right-hand constants. ◀ SOLVE SYS LSQ C MTH MATR LSQ	14-15
LU	Returns the Crout LU factorization of a square matrix (x). C MTH MATR FACTR LU	14-21
L*	Replace log-of-power with product-of-log. O ◀ EQUATION ◀ RULES L*	20-25
L()	Replace product-of-log with log-of-power. O ◀ EQUATION ◀ RULES L()	20-25
←M	Merge-factors-left. O ◀ EQUATION ◀ RULES ←M	20-24
▶ ←M	Executes ←M until no change in subexpression. O ◀ EQUATION ◀ RULES ▶ ←M	20-26
M→	Merge-factors-right. O ◀ EQUATION ◀ RULES M→	20-24
▶ M→	Executes M→ until no change in subexpression. O ◀ EQUATION ◀ RULES ▶ M→	20-26
MANT	Mantissa (decimal part) of number (x). F MTH REAL NXT MANT	12-10

Name, Key, or Label	Description, Type, and Keys	Page
MARK	Sets mark at cursor position. ◀ PICTURE × O ◀ PICTURE EDIT NXT MARK	9-3
MATCH	Accesses the pattern matching functions within the symbolic input form. O ▶ SYMBOLIC Manip expr... MATC	20-28
↑MATCH	Rewrites an expression (y) using a pattern-matching list (x) to replace particular subexpressions, searching the most deeply nested subexpressions first. C ◀ SYMBOLIC NXT ↑MAT	H-12
↓MATCH	Rewrites an expression (y) using a pattern-matching list (x) to replace particular subexpressions, searching the top-level expressions first. C ◀ SYMBOLIC NXT ↓MAT	H-12
MAX	Maximum of two real numbers (x and y). F MTH REAL MAX	12-10
MAXR	Returns the symbolic constant *MAXR*, the maximum machine-representable real number (9.99999999999E499). F MTH NXT CONS NXT MAXR	11-4
MAXΣ	Maximum column values in statistics matrix in ΣDAT. C ◀ STAT 1VAR MAXΣ	H-12
MCALC	Declares the specified variable or list of variables (x) to be "calculated-only." Used only in conjunction with MROOT. C ◀ EQ LIB MES MCAL	H-13
MEAN	Calculates mean of statistical data in ΣDAT. C ◀ STAT 1VAR MEAN	H-13
MEM	Bytes of available memory. C ◀ MEMORY MEM	H-13
MENU	Displays the specified built-in or custom menu (x). C ◀ MODES MENU MENU	30-1

G

Name, Key, or Label	Description, Type, and Keys	Page
MENU	Toggles the softkey menu on and off. ◀ PICTURE - O ◀ PICTURE EDIT NXT MENU	22-6
MERGE1	Merges plug-in RAM memory in port 1 with main memory. C ◀ LIBRARY MERG	28-16
MIN	Minimum of two real numbers (y and x). F MTH REAL MIN	12-10
MINEHUNT	Starts the Minehunt game. C ◀ EQ LIB UTILS MINE	25-14
MINIT	Creates a new *Mpar* from *EQ*. C ◀ EQ LIB MES MINIT	25-9
MINR	Returns the symbolic constant *MINR*, the minimum machine-representable real number (1.00000000000E−499). F MTH NXT CONS NXT MINR	11-4
MINΣ	Finds minimum column values in statistics matrix in ΣDAT. C ◀ STAT 1VAR MINΣ	H-13
MITM	Customizes the solver menu, using a title string (y) and a list of variables (x). C ◀ EQ LIB MES MITM	25-9
ML	Chooses multiple-line or single-line display for multiline results. O ◀ MODES FMT ML	4-10
MOD	Returns the modulus remainder of y divided by x. F MTH REAL MOD	12-10
MOVE	Moves the selected variable(s) to a new directory. O ▶ MEMORY MOVE	5-10
MSGBOX	Creates a user-defined message box from a string (x). C PRG NXT OUT MSGB	H-14

Name, Key, or Label	Description, Type, and Keys	Page
MROOT	Solves a set of equations for the specified variable (x) starting with only user-defined values (see MUSER and MCALC). C ◀ EQ LIB MES MROO	H-14
MSOLVR	Begins the solver using the current contents of the reserved variable EQ. C ◀ EQ LIB MES MSOL	25-9
MUSER	Makes the specified variable or list of variables (x) user-defined. Used only in conjunction with MROOT. C ◀ EQ LIB MES MUSE	H-14
NDIST	Returns the normal probability distribution (bell curve) at x based on the variance (y) and mean (z) of the normal distribution. C MTH NXT PROB NXT NDIST	12-5
NΣ	Returns number of rows in ΣDAT. C ◀ STAT SUMS NΣ	H-15
NEG	Negates x. A +/− or MTH NXT CMPL NXT NEG	12-1
NEW	Creates a new, named object. O ▶ MEMORY NEW	5-6
NEW	Creates a new alarm. O ▶ TIME Browse alarms... NEW	26-5
NEWOB	Converts object (x) taken from a composite object or variable into a new, independent object. C ◀ MEMORY NEWO	H-14
NEXT	Ends a definite-loop structure. PRG BRCH START NEXT C PRG BRCH FOR NEXT	29-12 29-13
NEXT	Displays but does not execute next one or two objects in suspended program. O PRG NXT RUN NEXT	29-9

Name, Key, or Label	Description, Type, and Keys	Page
NOT	Returns the binary or logical NOT of x. PRG ST NXT NOT F MTH BASE NXT LOGIC NOT	 H-15 15-4
NOVAL	Place holder for reset and initial values in user-defined dialog boxes. NOVAL is returned to the stack when a field is empty. C PRG NXT IN NOVA	
NSUB	Provides a way to access the current frame number during an iteration of a program or command applied using DOSUBS. C PRG LIST PROC NSUB	17-5
NUM	Returns character code of first character in string (x). C PRG TYPE NXT NUM	H-15
NUMX	Sets the number of x-steps for each y-step in 3D perspective plots. C ◀ PLOT NXT 3D VPAR NXT NUMX	H-15
NUMY	Sets the number of y-steps across the view-volume in 3D perspective plots. C ◀ PLOT NXT 3D VPAR NXT NUMY	H-15
→NUM	Converts a symbolic object ($\dot{x}$) to a number, as far as possible. C ◀ →NUM	11-5
NXEQ	Changes the current equation by rotating the elements in the list in EQ. O ◀ PICTURE FCN NXT NXEQ	22-11
OBJ→	Separates a composite object (x) into its components. PRG TYPE OBJ→ PRG LIST OBJ→ C ◀ CHARS NXT OBJ→	H-16 17-7
OCT	Sets octal base. C MTH BASE OCT	15-1
OFF	Turns calculator off. C PRG NXT RUN NXT OFF	

Name, Key, or Label	Description, Type, and Keys	Page
OK	Accepts the values of all the fields as they are currently displayed or performs the main action of the input form. [*input form*] ENTER O [*input form*] OK	6-7
OLDPRT	Remaps HP 48haracter set to match HP 82240A Infrared Printer. C ◀ I/O PRINT PRTPA OLDPR	
OPENIO	Opens serial port. C ◀ I/O NXT SERIA OPENI	27-3
OPTS	Selects plot options. O ▶ PLOT all plot types OPTS	22-2
OR	Logical OR of two expressions (x and y) that evaluate to 1 or 0 or binary OR combining two integers (x and y) or two strings (x and y). MTH BASE NXT LOGIC OR F PRG ST NXT OR	15-4 H-16
ORDER	Rearranges VAR menu in order specified in list (x). C ◀ MEMORY DIR ORDER	H-16
OVER	Duplicates the object in level 2 in level 1. C ◀ STACK OVER	3-12
ΣPAR	Reserved variable that stores statistical regression data. C ◀ STAT ΣPAR ΣPAR	21-12
PARAMETRIC	Selects PARAMETRIC plot type. C ◀ PLOT PTYPE PARA	23-7
PARITY	Sets the specified parity value (x). C ◀ I/O IOPAR PARIT	H-17
PARSURFACE	Selects PARSURFACE plot type. C ◀ PLOT NXT 3D PTYPE PARSU	23-35
PATH	Returns list containing path to current directory. C ◀ MEMORY DIR PATH	H-17

Name, Key, or Label	Description, Type, and Keys	Page
PCOEF	Finds the coefficients of the polynomial with the specified array of roots (x). C ◀ SOLVE POLY PCOEF	18-10
PCONTOUR	Selects the PCONTOUR plot type. C ◀ PLOT NXT 3D PTYPE PCON	23-30
PCOV	Calculates the population covariance. C ◀ STAT FIT NXT PCOV	H-17
PDIM	Replaces *PICT* with a blank *PICT* of specified dimensions (y) and (x). C PRG PICT PDIM	9-8 24-3
PERM	Returns the permutations of y items taken x at a time. F MTH NXT PROB PERM	12-4
PEVAL	Evaluates a polynomial with a specified array of coefficients (y) at a given value (x). C ◀ SOLVE POLY PEVAL	18-11
PGDIR	Purges specified directory (x) and all of its contents. C ◀ MEMORY DIR PGDIR	H-17
PICK	Copies object in level x to level 1. C ◀ STACK PICK	3-12
PICK	Copies object in current level to level 1. O ▶ STACK PICK	3-7
PICT	Puts the name *PICT* on the stack. C PRG PICT PICT	9-8
PICT→	Copies the current *PICT* as a graphic object and puts it on the stack. ◀ PICTURE STO O ◀ PICTURE EDIT NXT NXT PICT→	22-6
PICTURE	Enters Graphics environment. C ◀ PICTURE	9-2

Name, Key, or Label	Description, Type, and Keys	Page
PINIT	Initializes all ports containing RAM without erasing data. C ◀ LIBRARY NXT PINIT	28-14
PIXOFF	Turns off specified pixel (x) in *PICT*. C PRG PICT NXT PIXOF	9-8
PIXON	Turns on specified pixel (x) in *PICT*. C PRG PICT NXT PIXON	9-8
PIX?	Tests whether specified pixel (x) in *PICT* is on or off. C PRG PICT NXT PIX?	9-9
PKT	Used to send a command string (y) of a given type (x) to a Kermit server. C ◀ I/O SRVR PKT	27-13
PMAX	Sets upper-right plot coordinates (x). C Must be typed in.	H-17
PMIN	Sets lower-left plot coordinates (x). C Must be typed in.	H-18
PMT	The payment variable in the amortization solver. C ◀ SOLVE TVM SOLVR PMT	18-16
POLAR	Selects POLAR plot type. C ◀ PLOT PTYPE POLAR	23-4
POS	Returns the position of substring (x) in string (y) or object (x) in list (y). C PRG LIST ELEM POS	17-6
PRED	Using the current regression model, calculates the predicted value of one variable given the value of the other. O ▶ STAT Fit Data... PRED PRED	21-10
PREDV	Returns predicted value for dependent variable, given value of independent variable (x). C Must be typed in.	H-18
PREDX	Returns predicted value for independent variable, given value of dependent variable (x). C ◀ STAT FIT PREDX	H-18

G

Name, Key, or Label	Description, Type, and Keys	Page
PREDY	Returns predicted value for dependent variable, given value of independent variable (x). C ◀ STAT FIT PREDY	H-18
PRINT	Prints an object. O ▶ I/O **Print...** PRINT	27-4
PRLCD	Prints current display. C ◀ I/O PRINT PRLCD ▶ I/O **Print display** Simultaneously press ON and 1	
PROMPT	Displays prompt string (x) in status area and halts program execution. C PRG NXT IN NXT PROM	H-19
PROOT	Computes all roots of a polynomial with the specified array of coefficients (x). C ◀ SOLVE POLY PROOT	18-10
PRST	Prints all objects on stack. C ◀ I/O PRINT PRST	27-6
PRSTC	Prints all objects on stack in compact format. C ◀ I/O PRINT PRSTC	
PRTPAR	Reserved variable that contains the current printer settings. C ◀ I/O PRINT PRTPA PRTPA	27-3
PRVAR	Prints name and contents of one or more variables (x), including port names. C ◀ I/O PRINT PRVAR	27-6
PR1	Prints object in level 1. C ◀ I/O PR1	27-6
PSDEV	Calculates population standard deviation. C ◀ STAT 1VAR NXT PSDEV	H-19
PURGE	Purges one or more specified variables (x). C ◀ PURG	5-12

Name, Key, or Label	Description, Type, and Keys	Page
PURG	Purges selected objects or alarms. → MEMORY NXT PURG O → TIME Browse Alarms... PURG Purges all variables for current title. O → EQ LIB ... VARS NXT PURG	5-10 26-5 25-5
PUT	Replaces element in specified position (y) in array or list (z) with another element (x). C PRG LIST ELEM PUT	14-7
PUTI	Replaces element in specified position (y) in array or list (z) with another element (x) and increments index. C PRG LIST ELEM PUTI	17-6
PV	Present value of a loan in the amortization solver. C ← SOLVE TVM SOLVR PV	18-16
PVAR	Calculates population variance. C ← STAT 1VAR NXT PVAR	H-19
PVARS	Returns list of current backup objects and libraries within a port (x). C ← LIBRARY PVARS	28-4
PVIEW	Displays *PICT* with specified pixel coordinates (x) as the upper-left corner. C PRG NXT OUT PVIEW	H-19
PWRFIT	Set curve-fitting model to Power. C ← STAT ΣPAR MODL PWRFI	
PYR	Payments per year in the amortization solver. C ← SOLVE TVM SOLVR NXT PYR	18-16
PX→C	Converts pixel coordinates (x) to user-unit coordinates. C PRG PICT NXT PX→C	9-9
→Q	Converts number (x) to fractional equivalent. C ← SYMBOLIC NXT →Q	16-5
QR	Computes the QR factorization of a matrix (x). C MTH MATR FACTR QR	14-21

G

Name, Key, or Label	Description, Type, and Keys	Page
QUAD	Solves a first or second order equation (y) for a given variable (x). C ◀ SYMBOLIC QUAD	20-15
QUOTE	Returns argument expression (x) unevaluated. F ◀ SYMBOLIC NXT NXT QUOT	H-20
→Qπ	Calculates and compares quotients of number (x) and number/π, returning the one with the smaller denominator. C ◀ SYMBOLIC NXT →Qπ	16-6
RAD	Sets Radians mode. C ◀ MODES ANGL RAD ◀ RAD	4-4
RAND	Returns random number, and updates the random-number seed. C MTH NXT PROB RAND	12-4
RANK	Computes the rank of a rectangular matrix (x). C MTH MATR NORM NXT RANK	14-9
RANM	Creates a matrix with random elements from a list of specified dimensions (x). C MTH MATR MAKE RANM	14-3
RATIO	Prefix form of / used internally by EquationWriter application. F Must be typed in.	H-20
RCEQ	Returns equation(s) in EQ to level 1. ◀ PLOT EQ C ◀ PLOT NXT 3D EQ	H-20
RCI	Multiples the specified row (x) of an array (z) by a factor (y). C MTH MATR ROW RCI	14-19
RCIJ	Multiples a specified row (y) of an array (t) by a factor (z) and adds the result to another row (x). C MTH MATR ROW RCIJ	14-19
RCL	Recalls object stored in specified variable (x) to stack. C ▶ RCL	7-11

Name, Key, or Label	Description, Type, and Keys	Page
RCL	Recalls the selected object to the stack. O → MEMORY NXT RCL	5-9
RCLALARM	Recalls specified alarm (x) from system alarm list. C ← TIME ALRM RCLAL	H-21
RCLF	Returns binary integer representing states of system flags. C ← MODES FLAG NXT RCLF	24-7
RCLKEYS	Returns list of current user-key assignments. C ← MODES KEYS RCLK	30-7
RCLMENU	Returns menu number of current menu. C ← MODES MENU RCLM	H-21
RCLΣ	Recalls current statistical matrix in ΣDAT. ← PLOT NXT STAT DATA ΣDAT C ← STAT DATA ΣDAT	H-21
RCWS	Recalls binary integer wordsize. C MTH BASE NXT RCWS	15-2
RDM	Redimensions the elements of an array (y) according to the specified dimensions (x). C MTH MATR MAKE RDM	14-10
RDZ	Sets random number seed. C MTH NXT PROB RDZ	12-4
RE	Returns real part of complex number or array (x). F MTH NXT CMPL RE	12-14
RECN	Waits for specified data (x) from remote source running Kermit software. C ← I/O NXT RECN	H-21
RECT	Sets Rectangular mode. C MTH VECTR NXT RECT	13-2
RECV	Waits for sender-specified data from remote source running Kermit software. C ← I/O RECV	

G

Name, Key, or Label	Description, Type, and Keys	Page
RECV	Prepares the HP 48 to receive data. O → I/O Transfer... RECV	27-11
REPEAT	Begins loop clause if the result of the test-clause (x) is non-zero; otherwise execution resumes following the corresponding END. C PRG BRCH WHILE REPEA	29-14
REPL	Replaces portion of object (z) with another like object (x) starting at the specified position (y). PRG LIST REPL C PRG OB REPL	 17-7 H-22
REPL	Replaces portion of *PICT* with level-1 graphics object. O ← PICTURE EDIT NXT NXT REPL	9-4
REPL	Replaces one symbolic pattern with another in an expression. O ← EQUATION ◀ REPL	7-12
RES	Sets spacing (x) between plotted points. C ← PLOT PPAR RES	H-22
RESET	Resets current field value (or, at user's discretion, all field values in current input form) to its default setting. O → PLOT NXT RESET	6-6
RESET	Resets plotting parameters to their defaults. ← PLOT NXT 3D NXT VPAR NXT RESET O ← PLOT PPAR RESET	22-14
RESTORE	Replaces *HOME* directory with the specified backup copy (x). C ← MEMORY NXT RESTO	28-6
REVLIST	Reverses the elements in a list (x). C MTH LIST REVLI PRG LIST PROC REVLI	17-6

G

Name, Key, or Label	Description, Type, and Keys	Page
RKF	Uses a list (z)—containing the name of the time variable, the name of the solution variable, and the differential function—and the absolute error tolerance (y) to compute the solution of the initial value problem at a point (x) using the Runge-Kutta-Fehlberg (4,5) method. C ◀ (SOLVE) DIFFE RKF	H-22
RKFERR	Uses a list (y)—containing the name of the time variable, the name of the solution variable, and the differential function—and a possible step size (x) to compute the change in solution and an absolute error estimate for that step using the Runge-Kutta-Fehlberg (4,5) method. C ◀ (SOLVE) DIFFE RKFE	H-23
RKFSTEP	Uses a list (z)—containing the name of the time variable, the name of the solution variable, and the differential function—and a possible step size (x) to compute the next solution step of the initial value problem using the Runge-Kutta-Fehlberg (4,5) method such that the specified absolute error tolerance (y) is satisfied. C ◀ (SOLVE) DIFFE RKFS	H-23
RL	Rotates binary integer (x) left by one bit. C (MTH) BASE (NXT) BIT RL	15-5
RLB	Rotates binary integer (x) left by one byte. C (MTH) BASE (NXT) BYTE RLB	15-5
RND	Rounds numerical object (y) to the specified number (x) of decimal places or significant digits. F (MTH) REAL (NXT) (NXT) RND	12-10
RNRM	Calculates row norm of array (x). C (MTH) MATR NORM RNRM	14-8
ROLL	"Rolls up" x levels of the stack, so that level $x+1$ moves to level 1. C ◀ (STACK) ROLL	3-12
ROLL	"Rolls up" stack so that pointer level moves to level 1, level 1 moves to level 2, and so forth. O ▶ (STACK) ROLL	3-8

G

Name, Key, or Label	Description, Type, and Keys	Page
ROLLD	"Rolls down" x levels of the stack so that level 2 (y) moves to level x. C ◀ STACK ROLLD	3-12
ROLLD	"Rolls down" stack, so that level 1 moves to pointer level, level 2 moves to level 1, and so forth. O ▶ STACK ROLLD	3-8
ROOT	Solves for unknown variable (y) in equation (z), beginning the search from an inital guess (x). C ◀ SOLVE ROOT ROOT	H-23
ROOT	Moves graphics cursor to intersection of function plot and x-axis, displays value of root, returns value to stack. O ◀ PICTURE FCN ROOT	22-10
ROT	Moves object in level 3 to level 1, moves up objects in levels 1 and 2. C ◀ STACK ROT	3-12
ROW+	Expands an array (z) by inserting a row vector (y) at the specified row number (x). C MTH MATR ROW ROW+	14-5
+ROW	Inserts row of zeros at current row in Matrix Writer. O ▶ MATRIX NXT +ROW	8-8
ROW−	Deletes the specified row (x) of an array (y). C MTH MATR ROW ROW−	14-6
−ROW	Deletes current row in Matrix Writer. O ▶ MATRIX NXT −ROW	8-8
→ROW	Disassembles an array (x) into its component row vectors. C MTH MATR ROW →ROW	14-4
ROW→	Assembles a sequence of row vectors (... , z, y) into a matrix containing x rows. C MTH MATR ROW ROW→	14-3
RR	Rotates binary integer (x) right by one bit. C MTH BASE NXT BIT RR	15-5

Name, Key, or Label	Description, Type, and Keys	Page
RRB	Rotates binary integer (x) right by one byte. C (MTH) BASE (NXT) BYTE RRB	15-5
RREF	Computes the reduced row echelon form of a rectangular matrix (x). C (MTH) MATR FACTR RREF	14-20
RRK	Uses a list (z)—containing the name of the time variable, the name of the solution variable, the differential function and its first two derivatives—and the absolute error tolerance (y) to compute the solution of the initial value problem at a point (x) using the Rosenbrock and Runge-Kutta methods. C (←)(SOLVE) DIFFE RRK	H-23
RRKSTEP	Uses a list (t)—containing the name of the time variable, the name of the solution variable, the differential function and its first two derivatives—the absolute error tolerance (z), a possible step size (y), and a value (x) indicating the solution method used in the previous step. Computes the next solution step of the initial value problem using a combination of Rosenbrock and Runge-Kutta methods. C (←)(SOLVE) DIFFE RRKS	H-24
RSBERR	Uses a list (y)—containing the name of the time variable, the name of the solution variable, the differential function, and its first two derivatives—and a possible step size (x) to compute the change in solution and an absolute error estimate for that step using a combination of Rosenbrock and Runge-Kutta methods. C (←)(SOLVE) DIFFE RSBER	H-24
RSD	Computes the residual z-yx of three arrays. C (MTH) MATR (NXT) RSD	14-17
RSWP	Swaps two specified rows (y and x) of an array (z). C (MTH) MATR ROW (NXT) RSWP	14-6
R→B	Converts a positive real integer (x) to its binary integer equivalent. C (MTH) BASE R→B	15-3

Name, Key, or Label	Description, Type, and Keys	Page
R→C	Combines separate real (y) and imaginary (x) components into one complex number (or array). [PRG] TYPE [NXT] R→C C [MTH] [NXT] CMPL R→C	12-14
R→D	Converts a angle measure (x) from radians to degrees. F [MTH] REAL [NXT] [NXT] R→D	12-7
SAME	Tests two objects (y and x) for equality. C [PRG] TEST [NXT] SAME	H-24
SBRK	Sends serial break. C [←] [I/O] [NXT] SERIA SBRK	27-18
SCALE	Sets horizontal (y) and vertical (x) scale of PLOT axes. C [←] [PLOT] PPAR [NXT] SCALE	H-24
SCATRPLOT	Draws scatter plot of statistical data in ΣDAT. C [←] [STAT] PLOT SCATR	21-11
SCATTER	Selects SCATTER plot type. C [←] [PLOT] [NXT] STAT PTYPE SCATT	23-20
SCHUR	Computes the Schur decomposition of a square matrix (x) C [MTH] MATR FACTR SCHUR	14-22
SCI	Sets Scientific display mode with x decimal places. C [←] [MODES] FMT SCI	4-2
SCLΣ	Autoscales data in ΣDAT for scatter plot. C Must be typed in.	
SCONJ	Conjugates contents of variable (x). C [←] [MEMORY] ARITH [NXT] SCON	H-24
SDEV	Calculates standard deviation for each of the columns in ΣDAT. C [←] [STAT] 1VAR SDEV	H-25
SEND	Sends a copy of a variable (x) to a Kermit device. C [←] [I/O] SEND	H-25

Name, Key, or Label	Description, Type, and Keys	Page
SEND	Sends the specified object(s) in the manner designated in the input form. O → I/O [*various*] SEND	27-10
SEQ	Generates a sequence (a list) from an expression (v) involving a variable (t) whose value is incremented from z to y in steps of size x. C PRG LIST PROC NXT SEQ	17-7
SERVER	Puts HP 48nto Kermit Server mode. C ← I/O SRVR SERVE	
SF	Sets specified flag (x). PRG TEST NXT NXT SF C ← MODES FLAG SF	4-8
SHADE	Shades the area between the graph of a function and the x-axis or between two graphs, between the x-values defined by the mark and cursor. O ← PICTURE FCN SHADE	22-10
SHOW	Reconstructs expression (y) to make explicit all references to a variable (x). C ← SYMBOLIC SHOW	20-17
SIDENS	Calculates the intrinsic density of silicon as a function of temperature (x). F ← EQ LIB UTILS SIDEN	H-25
SIGN	Returns sign of a number (x). F MTH REAL NXT SIGN Returns unit vector of a complex number (x). F MTH NXT CMPL NXT SIGN	12-10 12-14
SIMU	Toggles between simultaneous and sequential plotting of multiple functions. O ← PLOT NXT FLAG SIMU	23-2
SIN	Sine of x. A SIN	12-2
SINH	Hyperbolic sine of x. A MTH HYP SINH	12-3
SINV	Replaces contents of variable (x) with its inverse. C ← MEMORY ARITH NXT SINV	H-25

G

Name, Key, or Label	Description, Type, and Keys	Page
SIZE	Finds dimensions of list, array, string, algebraic object, or graphics object (x). PRG LIST ELEM SIZE C PRG GROB NXT SIZE	 17-6 9-10
SIZE	Displays the size of the selected object in bytes and the amount of available memory. O ↷ MEMORY NXT SIZE	5-11
←SKIP	Moves cursor left to next logical break. ↶ EDIT ←SKIP O ... EDIT ←SKIP	2-12
SKIP→	Moves cursor right to next logical break. ↶ EDIT SKIP→ O ... EDIT SKIP→	2-12
SL	Shifts a binary integer (x) left by one bit. C MTH BASE NXT BIT SL	15-5
SLB	Shifts a binary integer (x) left by one byte. C MTH BASE NXT BYTE SLB	15-5
SLOPE	Calculates and displays slope of function at cursor position, returns slope to stack. O ↶ PICTURE FCN SLOPE	22-10
SLOPEFIELD	Selects the SLOPEFIELD plot type. C ↶ PLOT NXT 3D PTYPE SLOPE	23-25
SNEG	Negates contents of variable (x). C ↶ MEMORY ARITH NXT SNEG	H-26
SNRM	Computes the spectral norm of an array (x). C MTH MATR NORM SNRM	14-8
SOLVE	Initiates the solving procedure for the current problem. O ↷ SOLVE [*various*] SOLVE	18-1
SOLVEQN	Sets up the solver with a built-in equation set—designated by subject (z) and title (y)—and loads the accompanying diagram into *PICT* if specified (x). C ↶ EQ LIB EQLIB SOLVE	H-26

Name, Key, or Label	Description, Type, and Keys	Page
SORT	Sorts the elements in a list (x) in ascending order. C (MTH) LIST SORT	17-6
SPHERE	Sets spherical coordinate mode. C (MTH) VECTR (NXT) SPHER	13-2
SQ	Returns square of x. A (←)(x^2)	12-1
SR	Shifts a binary integer (x) right by one bit. C (MTH) BASE (NXT) BIT SR	15-5
SRAD	Computes the spectral radius of a square matrix (x). C (MTH) MATR NORM SRAD	14-9
SRB	Shifts a binary integer (x) right by one byte. C (MTH) BASE (NXT) BYTE SRB	15-5
SRECV	Reads specified number of characters (x) from serial port. C (←)(I/O) (NXT) SERIA SRECV	27-18
SST	Single-steps through suspended program. O (PRG) (NXT) RUN SST	29-8
SST↓	Single-steps through suspended program and its subroutines. O (PRG) (NXT) RUN SST↓	
START	Begins definite loop. C (PRG) BRCH START START	29-12
(←)START	Types START NEXT. O (PRG) BRCH (←)START	
(→)START	Types START STEP. O (PRG) BRCH (→)START	
STD	Sets Standard display mode. C (←)(MODES) FMT STD	4-2
STEP	Ends definite loop. (PRG) BRCH FOR STEP C (PRG) BRCH START STEP	29-12 29-13

Name, Key, or Label	Description, Type, and Keys	Page
STEP	Executes the next step in a step-wise differentiation. O → SYMBOLIC Differentiate... STEP	20-10
STEQ	Stores equation (x) in EQ. ← PLOT ← EQ C ← PLOT NXT 3D ← EQ	18-7
STIME	Sets serial transmit/receive timeout (x seconds). C ← I/O NXT SERIA STIME	27-18
↑STK	Selects Interactive Stack. ← EDIT ↑STK O → MATRIX NXT ↑STK	3-7 8-8
→STK	Copies the currently highlighted element in the MatrixWriter to the stack. O → MATRIX NXT →STK	8-8
→STK	Copies the equation set to the stack. O → EQ LIB ENTER →STK	25-4
STO	Stores object (y) in variable (x). C STO	5-11
STOALARM	Stores alarm (x) in system alarm list. C ← TIME ALRM STOAL	H-26
STOF	Uses a binary integer (x) to set state of system flags or a list of two binary integers (x) to set state of system flags and user flags. C ← MODES FLAG NXT STOF	24-7
STOKEYS	Uses a list (x) to make multiple user-key assignments. C ← MODES KEYS STOK	30-5
STO+	Adds number or other object to the contents of specified variable. C ← MEMORY ARITH STO+	H-27
STO−	Calculates difference between contents of specified variable (y). and specified number or other object, and stores result in specified variable. C ← MEMORY ARITH STO−	H-27

Name, Key, or Label	Description, Type, and Keys	Page
STO*	Multiplies contents of specified variable by specified number or other object. C ◀ MEMORY ARITH STO*	H-27
STO/	Calculates quotient of contents of specified variable and specified number or other object, and stores result in specified variable. C ◀ MEMORY ARITH STO/	H-27
STOΣ	Stores current statistics matrix (x) in ΣDAT. C ◀ STAT DATA ◀ ΣDAT	H-28
STR→	Evaluates a string (x) as if its text were entered from the command line. C Must be typed in.	
→STR	Converts an object (x) into a string. C PRG TYPE →STR	H-28
STREAM	Takes an object (x) (normally a program or command) and applies it to every element of a list (y). C PRG LIST PROC STREA	17-5
STS	Displays a status line showing the current directory, the modes and flags that are set, and the current date and time. O [*input form*] NXT CALC STS	6-5
STWS	Sets binary integer wordsize to x bits. C MTH BASE NXT STWS	15-2
SUB	Extracts the portion of list, string, array, or graphics object (z) specified by start (y) and end (x) positions. PRG LIST SUB ◀ CHAR SUB MTH MATR MAKE NXT SUB C PRG GROB SUB	17-7 9-10
SUB	Returns specified portion of *PICT* to stack. O ◀ PICTURE EDIT NXT NXT SUB	9-4
SUB	Returns specified subexpression to stack. O ◀ EQUATION ◀ SUB	20-20

Name, Key, or Label	Description, Type, and Keys	Page
SVD	Compute the singular value decomposition of a rectangular matrix (x). C MTH MATR FACTR SVD	14-22
SVL	Computes the singular values of a rectangular matrix (x). C MTH MATR FACTR NXT SVL	14-22
SWAP	Exchanges objects in levels 1 and 2. C ← SWAP	3-4
SYM	Selects whether or not symbolic constants evaluate to numbers. O ← MODES MISC SYM	4-10
SYSEVAL	Evaluates a system object (x). *Use only as specified by HP applications.* C Must be typed in.	H-28
←T	Move term left. O ← EQUATION ◀ RULES ←T	20-22
→←T	Executes ←T until no change in subexpression. O ← EQUATION ◀ RULES → ←T	20-26
T→	Move term right. O ← EQUATION ◀ RULES T→	20-22
→T→	Executes T→ until no change in subexpression. O ← EQUATION ◀ RULES → T→	20-26
%T	Returns percent fraction that x is of y. F MTH REAL %T	12-9
→TAG	Tags (labels) an object (y) with a name or descriptive string (x). C PRG TYPE →TAG	H-28
TAN	Tangent of x. A TAN	12-2
TAIL	Returns all but the first element of a list or all but the first character of a string. C PRG LIST ELEM NXT TAIL	17-6

Name, Key, or Label	Description, Type, and Keys	Page
TANH	Hyperbolic tangent of x. A (MTH) HYP TANH	12-3
TANL	Draws the line tangent to the current function at the x-value of the cursor and returns the equation of the tangent line to the stack. O (◀) (PICTURE) FCN (NXT) TANL	22-11
TAYLR	Calculates the symbolic xth order Taylor's polynomial approximation of an expression (z) in a specified variable (y). C (◀) (SYMBOLIC) TAYLR	20-12
TDELTA	Returns the increment between a final temperature (y) and an initial temperature (x). This is special temperature version of the normal subtraction function. F (◀) (EQ LIB) UTILS (NXT) TDELT	H-28
TEACH	Loads built-in examples. C Must be typed in.	29-19
TEXT	Displays stack when display is updated. C (PRG) (NXT) OUT TEXT	
THEN	Begins true clause. (PRG) (NXT) ERROR IFERR THEN (PRG) BRCH CASE THEN C (PRG) BRCH IF THEN	29-10
TICKS	Returns system time as binary integer in units of clock ticks (1 tick = $\frac{1}{8192}$ seconds). C (◀) (TIME) TICKS	16-4
TIME	Returns current time in 24-hour HH.MMSSs format. C (◀) (TIME) TIME	16-3
→TIME	Sets system time to a time (x) specified in 24-hour HH.MMSSs format. C (◀) (TIME) →TIM	H-29
TINC	Increases or decreases a given temperature (y) by a specified temperature increment (x). This is a special temperature version of the normal addition function. F (◀) (EQ LIB) UTILS (NXT) TINC	H-29

Name, Key, or Label	Description, Type, and Keys	Page
TLINE	In *PICT*, toggles pixels on the line defined by two coordinates (y and x). C [PRG] PICT TLINE	9-8
TLINE	Switches pixels on and off on line between mark and cursor. O [↰][PICTURE] EDIT TLINE	9-4
TMENU	Displays list-defined custom menu (x) but does not change contents of *CST*. C [↰][MODES] MENU TMEN	30-4
TOT	Sums each column of matrix in ΣDAT. C [↰][STAT] 1VAR TOT	H-29
TRACE	Computes the sum of the diagonal elements (trace) of a square matrix (x). C [MTH] MATR NORM [NXT] TRACE	14-9
TRACE	Toggles TRACE mode on and off. O [↰][PICTURE] TRACE	22-4
TRANSIO	Selects the specified character translation settings (x). C [↰][I/O] IOPAR TRAN	H-29
TRG*	Expands trigonometric and hyperbolic functions of sums and differences. O [↰][EQUATION][◀] RULES TRG*	20-25
→TRG	Replace exponential with trigonometric functions. O [↰][EQUATION][◀] RULES →TRG	20-25
TRN	Transposes matrix (x). C [MTH] MATR MAKE TRN	14-10
TRNC	Truncates a number (y) to the specified number (x) of decimal places or significant digits. F [MTH] REAL [NXT][NXT] TRNC	12-10
TRUTH	Selects TRUTH plot type. C [↰][PLOT] PTYPE TRUTH	23-14
TSTR	Converts date (y) and time (x) in number form to string form, including the calculated day of the week. C [↰][TIME] [NXT][NXT] TSTR	16-4

Name, Key, or Label	Description, Type, and Keys	Page
TVARS	Returns variables containing specified object type (x). C ◀ MEMORY DIR TVARS	H-30
TVM	Displays the TVM menu. C ◀ SOLVE TVM	18-13
TVMBEG	Sets payment mode to Beginning-of-Period. C Must be typed in.	
TVMEND	Sets payment mode to End-of-Period. C Must be typed in.	
TVMROOT	Solves for the specified TVM variable (x) using the values stored in the remaining TVM variables. C ◀ SOLVE TVM TVMR	H-30
TYPE	Returns the type number of an object (x). PRG TYPE NXT NXT TYPE C PRG TEST NXT TYPE	H-30
TYPES	Display a list of valid object types for the selected field. O [*input form*] NXT TYPES	6-6
UBASE	Converts unit object (x) to SI base units. F ◀ UNITS UBASE	10-6
UFACT	Factors a unit (x) from the unit expression of another unit object (y). C ◀ UNITS UFACT	10-9
→UNIT	Creates a unit object from a real number (y) and the unit part of a unit object (x). PRG TYPE →UNIT C ◀ UNITS →UNIT	10-14
UNTIL	Begins test clause. C PRG BRCH DO UNTIL	29-14
UPDIR	Makes parent directory the current directory. C ◀ UP	5-12
UTPC	Returns probability that chi-square random variable is greater than x, given the degrees of freedom (y) of the distribution. C MTH NXT PROB NXT UTPC	12-5

G

Name, Key, or Label	Description, Type, and Keys	Page
UTPF	Returns probability that Snedecor's F random variable is greater than x, given the numerator (z) and denominator (y) degrees of freedom of the distribution. C (MTH) (NXT) PROB (NXT) UTPF	12-5
UTPN	Returns probability that normal random variable is greater than x, given the mean (z) and variance (y) of the distribution. C (MTH) (NXT) PROB (NXT) UTPN	12-5
UTPT	Returns probability that Student's t random variable is greater than x, given the degrees of freedom (y) of the distribution. C (MTH) (NXT) PROB (NXT) UTPT	12-5
UVAL	Eliminates the unit part of specified unit object (x). (←) (UNITS) UVAL	10-14
VAR	Calculates variance of statistical data columns in ΣDAT. C (←) (STAT) 1VAR (NXT) VAR	H-31
VARS	Returns list of variables in current directory. C (←) (MEMORY) DIR VARS	H-31
VEC	Switches between vector and array modes. O (→) (MATRIX) VEC	8-8
VIEW	Copies object in current level into appropriate environment for viewing. O (←) (VIEW)	3-7
(→) VIEW	Displays the menu keywords for the current menu. If the keywords are variables, their values are shown. O (→) (VIEW)	3-7
VIEW	Shows the range and current equation while key is held. O (←) (PICTURE) FCN (NXT) VIEW	22-5
VPAR	Returns the *VPAR* reserved variable. C (←) (PLOT) (NXT) 3D VPAR (NXT) VPAR	22-15

G

Name, Key, or Label	Description, Type, and Keys	Page
VTYPE	Returns type number of object stored in local or global name (x). C [PRG] TYPE [NXT] [NXT] VTYPE	H-31
VZIN	Zooms in vertically. O [◀] [PICTURE] ZOOM [NXT] VZIN	22-8
VZOUT	Zooms out vertically. O [◀] [PICTURE] ZOOM [NXT] VZOUT	22-8
→V2	Combines two real numbers (y and x) into a 2-D vector or complex number. O [MTH] VECTR →V2	13-3
→V3	Combines three real numbers into 3-D vector according to the current Coordinate System mode. C [MTH] VECTR →V3	13-3
V→	Separates vector or complex number (x) into its component elements according to current angle mode. C [MTH] VECTR V→	13-4
*W	Multiplies the horizontal plot scale by a factor (x). C [◀] [PLOT] PPAR [NXT] *W	H-31
WAIT	Halts program execution for specified number of seconds (x) or until key pressed. C [PRG] [NXT] IN WAIT	H-32
WHILE	Begins indefinite loop. C [PRG] BRCH WHILE WHILE	29-14
G [◀] WHILE	Types WHILE REPEAT END O [PRG] BRCH [◀] WHILE	H-32
WID→	Increases column width and decrements number of columns. O [▶] [MATRIX] WID→	8-8
←WID	Decreases column width and increments number of columns. O [▶] [MATRIX] ←WID	8-8
WIREFRAME	Selects WIREFRAME plot type. C [◀] [PLOT] [NXT] 3D PTYPE WIREF	23-28

Name, Key, or Label	Description, Type, and Keys	Page
WSLOG	Returns a series of strings recording the date, time, and cause of each warmstart event. C Must be typed in.	H-32
ΣX	Returns sum of data in independent column in ΣDAT. C ◀ STAT SUMS ΣX	H-32
ΣX^2	Returns sum of squares of data in independent column in ΣDAT. C ◀ STAT SUMS ΣX^2	H-33
XCOL	Specifies independent-variable column (x) in matrix in ΣDAT. C ◀ STAT ΣPAR XCOL	H-33
XMIT	Sends the given string (x) through the serial port without using the Kermit protocol. C ◀ I/O NXT SERIA XMIT	27-18
XOR	Logical XOR of two expressions (x and y) that evaluate to `1` or `0` or binary XOR combining two integers (x and y) or two strings (x and y). MTH BASE NXT LOGIC XOR F PRG TEST NXT XOR	15-4
XPON	Returns exponent of number (x). F MTH REAL NXT XPON	12-10
XRECV	Receives an object via xmodem. C ◀ I/O NXT XRECV	H-33
XRNG	Specifies the horizontal plot axis display range (from y to x). C ◀ PLOT PPAR XRNG	H-33
XROOT	Returns the xth root of a real number y. A ▶ $\sqrt[x]{y}$	12-1
XSEND	Sends an object via xmodem. C ◀ I/O NXT XSEN	H-33
XVOL	Sets the coordinates, X_{left} (y) and X_{right} (x), establishing the width of the 3D plotting volume. C ◀ PLOT NXT 3D VPAR XVOL	H-34

Name, Key, or Label	Description, Type, and Keys	Page
XXRNG	Sets the coordinates XX_{left} (y) and XX_{right} (x) establishing the width of the 3D target mapping range (for GRIDMAP and PARSURFACE plots). C ◀ PLOT NXT 3D VPAR XXRN	H-34
ΣX*Y	Returns sum of products of data in independent and dependent columns in ΣDAT. C ◀ STAT SUMS ΣX*Y	H-34
(X,Y)	Toggles on and off the current cursor coordinates at bottom left of display. O ◀ PICTURE + O ◀ PICTURE X,Y	22-4
X,Y→	Enters the current cursor coordinates as a complex number onto the Stack. ◀ PICTURE ENTER O ◀ PICTURE EDIT NXT NXT X,Y→	22-6
ΣY	Returns sum of data in dependent column in ΣDAT. C ◀ STAT SUMS ΣY	H-34
ΣY^2	Returns sum of squares of data in dependent column in ΣDAT. C ◀ STAT SUMS ΣY^2	H-34
YCOL	Selects indicated column (x) of ΣDAT as dependent-variable column for two-variable statistics. C ◀ STAT ΣPAR YCOL	H-35
YRNG	Specifies the vertical plot axis display range (from y to x). C ◀ PLOT PPAR YRNG	H-35
YSLICE	Selects YSLICE plot type. C ◀ PLOT NXT 3D PTYPE YSLIC	23-32
YVOL	Sets the coordinates, Y_{far} (y), and Y_{near} (x) establishing the width of the 3D plotting volume. C ◀ PLOT NXT 3D VPAR YVOL	H-35

G

Name, Key, or Label	Description, Type, and Keys	Page
YYRNG	Sets the coordinates YY_{far} (y) and XX_{near} (x) establishing the width of the 3D target mapping range (for GRIDMAP and PARSURFACE plots). C ◀ PLOT NXT 3D VPAR YYRN	H-35
ZAUTO	Autoscales and redraws the graph. O ◀ PICTURE ZOOM NXT ZAUTO	22-8
ZDECI	Scale the horizontal axis so that every pixel represents 0.1. O ◀ PICTURE ZOOM NXT NXT ZDECI	22-8
ZDFLT	Resets PPAR to the current scale settings. O ◀ PICTURE ZOOM ZDFLT	22-8
ZFACT	Displays input form to set zoom default factors. O ◀ PICTURE ZOOM ZFACT	22-7
ZFACTOR	Calculates the gas compressibility factor Z using the temperature reduction ratio (y) and the pressure reduction ratio (x). F ◀ EQ LIB UTILS ZFACT	H-35
ZIN	Zooms in by a standard factor. O ◀ PICTURE ZOOM ZIN	22-8
ZINTG	Sets the horizontal scale so every pixel represents 1 unit. Does not affect vertical scale. O ◀ PICTURE ZOOM NXT NXT ZINTG	22-8
ZLAST	Returns you to the previous zoom factor. O ◀ PICTURE ZOOM NXT NXT ZLAST	22-8
ZOOM	Magnifies a rectangular area (that you have drawn) to fill the entire screen. O ◀ PICTURE ZOOM BOXZ [*draw a box*] ZOOM	22-7
ZOUT	Zooms out by a standard factor. O ◀ PICTURE ZOOM ZOUT	22-8
ZSQR	Resets the vertical scale to match the horizontal scale. O ◀ PICTURE ZOOM ZSQR	22-8

Name, Key, or Label	Description, Type, and Keys	Page
ZTRIG	Sets the horizontal scale so that every ten pixels represent $\pi/2$ and sets the vertical scale so that every ten pixels represent 1. O ◀ PICTURE ZOOM NXT NXT ZTRIG	22-8
ZVOL	Sets the coordinates, Z_{low} (y) and Z_{high} (x) establishing the height of the 3D plotting volume. C ◀ PLOT NXT 3D VPAR ZVOL	H-36
+	Adds two objects (y and x). A +	12-1
+/−	Switches cursor style between super-imposing and inverting cross. Also changes sign of number. ◀ PICTURE +/− O ◀ PICTURE EDIT NXT +/−	22-6
+1−1	Add and subtract 1. O ◀ EQUATION ◀ RULES +1−1	20-21
−	Subtracts one object (x) from another (y). A −	12-1
−()	Double negate and distribute. O ◀ EQUATION ◀ RULES NXT −()	20-24
*	Multiplies two objects (y and x). A ×	12-1
*1	Multiply by 1. O ◀ EQUATION ◀ RULES *1	20-21
/	Divides one object (y) by another (x). ◀ SOLVE SYS / A ÷	12-1
/1	Divide by 1. O ◀ EQUATION ◀ RULES /1	20-21
^	Raises a number (y) to specified power (x). A y^x	12-1
^1	Raise to power 1. O ◀ EQUATION ◀ RULES ^1	20-21

G

Name, Key, or Label	Description, Type, and Keys	Page
$<$	Tests whether $y < x$. F PRG TEST <	H-36
$\leq$	Tests whether $y \leq x$. F PRG TEST ≤	H-36
$>$	Tests whether $y > x$. F PRG TEST >	H-37
$\geq$	Tests whether $y \geq x$. F PRG TEST ≥	H-37
$=$	Returns a equation formed of two expressions (y and x). A ← =	11-4
==	Tests whether $y = x$. F PRG TEST ==	H-38
$\neq$	Tests whether $y \neq x$. F PRG TEST ≠	H-38
!	Factorial of x. F MTH NXT PROB !	12-4
$\int$	Integrates an expression (y) from t to z with respect to the specified variable of integration (x). A → ∫	H-38
∂	Takes the derivative of an expression (y) with respect to the specified variable of differentiation (x). A → ∂	H-38
%	Computes y percent of x. A MTH REAL %	12-9
π	Returns the symbolic constant π (or 3.14159265359 depending on flag −2). MTH NXT CONS π F ← π	11-4
Σ	Calculates the sum of an expression (x) evaluated a series of times as an index variable (t) moves from z to y. F → Σ	H-39 7-5

Name, Key, or Label	Description, Type, and Keys	Page
Σ+	Adds data point (x) to matrix in ΣDAT. C ◀ STAT DATA Σ+	H-39
Σ−	Subtracts data point (x) from matrix in ΣDAT. C ◀ STAT DATA Σ-	H-39
√	Returns square root of x. A √x	12-1
|	Uses a list of names and values (x) to substitutes values for names in an expression (y). F ◀ SYMBOLIC NXT |	20-17
1/()	Double-invert and distribute. O ◀ EQUATION ◀ RULES NXT NXT 1/()	20-24
(( ))	Parenthesize neighbors. O ◀ EQUATION ◀ RULES (())	20-23
(←	Expand-subexpression-left. O ◀ EQUATION ◀ RULES NXT (←	20-23
▶(←	Executes (← until no change in subexpression. O ◀ EQUATION ◀ RULES NXT ▶ (←	20-26
→()	Distribute prefix function. O ◀ EQUATION ◀ RULES NXT →()	20-23
→)	Expand-subexpression-right. O ◀ EQUATION ◀ RULES NXT →)	20-23
▶→)	Executes →) until no change in subexpression. O ◀ EQUATION ◀ RULES NXT ▶ →)	20-26
←→	Commute arguments. O ◀ EQUATION ◀ RULES ←→	20-23
→	Begins a local variable structure. C ▶ →	29-16

G

H

Stack Diagrams for Selected Commands

AMORT Command: Amortizes a loan or investment based upon the current amortization settings.

Level 1	→	Level 3	Level 2	Level 1
n	→	*principal*	*interest*	*balance*

AND Function: Returns the logical AND of two arguments.

Level 2	Level 1	→	Level 1
#n_1	#n_2	→	#n_3
"$string_1$"	"$string_2$"	→	"$string_3$"
T/F_1	T/F_2	→	0/1
T/F	'*symb*'	→	'*T/F* AND *symb*'
'*symb*'	*T/F*	→	'*symb* AND *T/F*'
'$symb_1$'	'$symb_2$'	→	'$symb_1$ AND $symb_2$'

APPLY Function: Creates an expression from the specified function name and arguments.

Level 2	Level 1	→	Level 1
{ $symb_1$ ... $symb_n$ }	'*name*'	→	'*name*($symb_1$ ... $symb_n$)'

ARRY→ Command: Takes an array and returns its elements as separate real or complex numbers. Also returns a list of the dimensions of the array.

Level 1	→	Level nm+1 ... Level 2	Level 1
[*vector*]	→	$z_1 \ \cdots \ z_n$	$\{ \ n_{element} \ \}$
[[*matrix*]]	→	$z_{11} \ \cdots \ z_{nm}$	$\{ \ n_{row} \ m_{col} \ \}$

ATICK Command: Sets the axes tick-mark annotation in the reserved variable *PPAR*.

Level 1	→	Level 1
x	→	
#*n*	→	
{ *x*,*y* }	→	
{ #*n* #*m* }	→	

BINS Command: Sorts the elements of the independent column (XCOL) of the current statistics matrix (the reserved variable ΣDAT) into $(n_{bins} + 2)$ bins, where the left edge of bin 1 starts at value x_{min} and each bin has width x_{width}.

Level 3	Level 2	Level 1	→	Level 2	Level 1
x_{min}	x_{width}	n_{bins}	→	[[$n_{bin1} \ \cdots \ n_{binn}$]]	[$n_{binL} \ n_{binR}$]

BYTES Command: Returns the number of bytes and the checksum for the given object.

Level 1	→	Level 2	Level 1
obj	→	$\#n_{checksum}$	x_{size}

CENTR Command: Adjusts the first two parameters in the reserved variable *PPAR*, $\langle x_{min}, y_{min} \rangle$ and $\langle x_{max}, y_{max} \rangle$, so that the point represented by the argument $\langle x, y \rangle$ is the plot center.

Level 1	→	Level 1
(*x*, *y*)	→	
x	→	

CHOOSE Command: Creates a user-defined choose box.

Level 3	Level 2	Level 1	→	Level 2	Level 1
"*prompt*"	{ c_1 ... c_n }	n_{pos}	→	*obj* or *result*	*1*
"*prompt*"	{ c_1 ... c_n }	n_{pos}	→		*0*

CHR Command: Returns a string representing the HP 48 character corresponding to the character code *n*.

Level 1	→	Level 1
n	→	"*string*"

CKSM Command: Specifies the error-detection scheme.

Level 1	→	Level 1
$n_{checksum}$	→	

CLKADJ Command: Adjusts the system time by *x* clock ticks, where 8192 clock ticks equal 1 second.

Level 1	→	Level 1
x	→	

COLΣ Command: Specifies the independent-variable and dependent-variable columns of the current statistics matrix (the reserved variable ΣDAT).

Level 2	Level 1	→	Level 1
x_{xcol}	x_{ycol}	→	

CORR Command: Returns the correlation coefficient of the independent and dependent data columns in the current statistics matrix (reserved variable ΣDAT).

Level 1	→	Level 1
	→	$x_{correlation}$

COV Command: Returns the sample covariance of the independent and dependent data columns in the current statistics matrix (reserved variable ΣDAT).

Level 1	→	Level 1
	→	$x_{covariance}$

CRDIR Command: Creates an empty subdirectory with the specified name within the current directory.

H

Level 1	→	Level 1
'global'	→	

DARCY Function: Calculates the Darcy friction factor of certain fluid flows.

Level 2	Level 1	→	Level 1
$x_{e/D}$	y_{Re}	→	x_{Darcy}

→DATE Command: Sets the system date to *date*.

Level 1	→	Level 1
date	→	

DECR Command: Takes a variable on level 1, subtracts 1, stores the new value back into the original variable, and returns the new value to level 1.

Level 1	→	Level 1
'*name*'	→	x_{new}

DELALARM Command: Deletes the alarm specified in level 1.

Level 1	→	Level 1
n_{index}	→	

DEPND Command: Specifies the dependent variable (and its plotting range for TRUTH plots).

Level 2	Level 1	→	Level 1
	'*global*'	→	
	{ *global* }	→	
	{ *global* y_{start} y_{end} }	→	
	{ y_{start} y_{end} }	→	
y_{start}	y_{end}	→	

DISP Command: Displays *obj* in the *n*th display line.

Level 2	Level 1	→	Level 1
obj	*n*	→	

DOERR Command: Executes a "user-specified" error, causing a program to behave exactly as if a normal error had occurred during program execution.

Level 1	→	Level 1
n_{error}	→	
$\#n_{error}$	→	
"*error*"	→	
0	→	

DTAG Command: Removes all tags (labels) from an object.

Level 1	→	Level 1
:tag:obj	→	*obj*

EQ→ Command: Separates an equation into its left and right sides.

Level 1	→	Level 2	Level 1
'$symb_1 = symb_2$'	→	'$symb_1$'	'$symb_2$'
z	→	*z*	0
'*name*'	→	'*name*'	0
x_unit	→	*x_unit*	0
'*symb*'	→	'*symb*'	0

H

ERRM Command: Returns a string containing the error message of the most recent calculator error.

Level 1	→	Level 1
	→	"*error message*"

ERRN Command: Returns the error number of the most recent calculator error.

Level 1	→	Level 1
	→	$\#n_{error}$

EYEPT Command: Specifies the coordinates of the eye point in a perspective plot.

Level 3	Level 2	Level 1	→	Level 1
x_{point}	y_{point}	z_{point}	→	

F0λ Function: Returns the fraction of total black-body emissive power.

Level 2	Level 1	→	Level 1
y_{lambda}	x_T	→	x_{power}
y_{lambda}	'*symb*'	→	'F0λ(y_{lambda},*symb*)'
'*symb*'	x_T	→	'F0λ(*symb*,x_T)'
'$symb_1$'	'$symb_2$'	→	'F0λ($symb_1$,$symb_2$)'

FACT Function: Provided for compatibility with the HP 28. FACT is the same as !.

Level 1	→	Level 1
n	→	*n*!
x	→	Γ(*x*+1)
'*symb*'	→	'(symb)!'

FANNING Function: Calculates the Fanning friction factor of certain fluid flows.

Level 2	Level 1	→	Level 1
$x_{x/D}$	y_{Re}	→	$x_{fanning}$
$x_{x/D}$	'symb'	→	'FANNING($x_{x/D}$,symb)'
'symb'	y_{Re}	→	'FANNING(symb,y_{Re})'
'$symb_1$'	'$symb_2$'	→	'FANNING($symb_1$,$symb_2$)'

FINDALARM Command: Returns the alarm index n_{index} of the first alarm due after the specified time.

Level 1	→	Level 1
date	→	n_{index}
{ *date time* }	→	n_{index}
0	→	n_{index}

FREEZE Command: Freezes the part of the display specified by $n_{display\ area}$, so that it is not updated until a key is pressed.

Level 1	→	Level 1
$n_{displayarea}$	→	

H

***H Command:** Multiplies the vertical plot scale by x_{factor}.

Level 1	→	Level 1
x_{factor}	→	

HEAD Command: Returns the first element of a list or string.

Level 1	→	Level 1
{ obj_1 ... obj_n }	→	obj_1
"*string*"	→	"$element_1$"

IFT Command: Executes *obj* if *T/F* is nonzero. Discards *obj* if *T/F* is zero.

Level 2	Level 1	→	Level 1
T/F	*obj*	→	*It depends!*

IFTE Function: Executes the *obj* on level 2 if *T/F* is nonzero. Executes the *obj* on level 1 if *T/F* is zero.

Level 3	Level 2	Level 1	→	Level 1
T/F	obj_{true}	obj_{false}	→	*It depends!*

INCR Command: Takes a variable on level 1, adds 1, stores the new value back into the original variable, and returns the new value to level 1.

Level 1	→	Level 1
'*name*'	→	$x_{increment}$

INDEP Command: Specifies the independent variable and its plotting range.

Level 2	Level 1	→	Level 1
	'global'	→	
	{ global }	→	
	{ global x_{start} x_{end} }	→	
	{ x_{start} x_{end} }	→	
x_{start}	x_{end}	→	

INPUT Command: Prompts for data input to the command line and prevents the user access to stack operations.

Level 2	Level 1	→	Level 1
"stack prompt"	"command-line prompt"	→	"result"
"stack prompt"	{ $list_{command-line}$ }	→	"result"

KERRM Command: Returns the text of the most recent Kermit error packet.

Level 1	→	Level 1
	→	"error-message"

H

KEY Command: Returns to level 1 a test result and, if a key is pressed, returns to level 2 the row-column location x_{nm} of that key.

Level 1	→	Level 2	Level 1
	→	x_{nm}	1
	→		0

LIBEVAL Command: Evaluates unnamed library functions.

Level 1	→	Level 1
$\#n_{function}$	→	

LIBS Command: Lists the title, number, and port of each library attached to the current directory.

Level 1	→	Level 1
	→	{ "*title*" n_{lib} n_{port} ... "*title*" n_{lib} n_{port} }

ΣLINE Command: Returns an expression representing the best fit line according to the current statistical model, using X as the independent variable name, and explicit values of the slope and intercept taken from the reserved variable ΣPAR.

Level 1	→	Level 1
	→	'$symb_{formula}$'

LININ Function: Tests whether an algebraic is structurally linear for a given variable.

Level 2	Level 1	→	Level 1
'*symb*'	'*name*'	→	0/1

LIST→ Command: Takes a list of n objects and returns them to separate levels, and returns the total number of objects to level 1.

Level 1	→	Level n+1 ...	Level 2	Level 1
{ obj_1 ... obj_n }	→	obj_1 ...	obj_n	n

LR Command: Uses the currently selected statistical model to calculate the linear regression coefficients (intercept and slope) for the selected dependent and independent variables in the current statistics matrix (reserved variable ΣDAT).

Level 1	→	Level 2	Level 1
	→	Intercept: x_1	Slope: x_2

↑MATCH Command: Rewrites an expression. ↑MATCH works from bottom up; that is, it checks the lowest level (most deeply nested) subexpressions first.

Level 2	Level 1	→	Level 2	Level 1
'$symb_1$'	{ '$symb_{pat}$' '$symb_{repl}$' }	→	'$symb_2$'	0/1
'$symb_1$'	{ '$symb_{pat}$' '$symb_{repl}$' '$symb_{cond}$' }	→	'$symb_2$'	0/1

↓MATCH Command: Rewrites an expression. ↓MATCH works from top down; that is, it checks the entire expression first.

Level 2	Level 1	→	Level 2	Level 1
'$symb_1$'	{ '$symb_{pat}$' '$symb_{repl}$' }	→	'$symb_2$'	0/1
'$symb_1$'	{ '$symb_{pat}$' '$symb_{repl}$' '$symb_{cond}$' }	→	'$symb_2$'	0/1

H **MAXΣ Command:** Finds the maximum coordinate value in each of the m columns of the current statistics matrix (reserved variable ΣDAT).

Level 1	→	Level 1
	→	x_{max}
	→	$[x_{max1}\ x_{max2}\ \cdots\ x_{maxm}]$

MCALC Command: Designates a variable as a calculated value (not user-defined) for the Multiple-Equation Solver.

Level 1	→	Level 1
'name'	→	
{ *list* }	→	
"ALL"	→	

MEAN Command: Returns the mean of each of the m columns of coordinate values in the current statistics matrix (reserved variable ΣDAT).

Level 1	→	Level 1
	→	x_{mean}
	→	$[\ x_{mean1}\ x_{mean2}\ \cdots\ x_{meanm}\ ]$

MEM Command: Returns the number of bytes of available RAM.

Level 1	→	Level 1
	→	x

MINΣ Command: Finds the minimum coordinate value in each of the m columns of the current statistics matrix (reserved variable ΣDAT).

Level 1	→	Level 1
	→	x_{min}
	→	$[\ x_{min1}\ x_{min2}\ \cdots\ x_{minm}\ ]$

MROOT Command: Uses the Multiple-Equation Solver to solve for one or more variables using the equation set in *Mpar*.

Level 1	→	Level 1
'name'	→	*x*
"ALL"	→	

MSGBOX Command: Creates a user-defined message box.

Level 1	→	Level 1
"message"	→	

MUSER Command: Designates a variable as user-defined for the Multiple-Equation Solver.

Level 1	→	Level 1
'name'	→	
{ list }	→	
"ALL"	→	

NEWOB Command: Creates a new copy of the specified object.

Level 1	→	Level 1
obj	→	*obj*

NOT Command: Returns the one's complement or logical inverse of the argument.

Level 1	→	Level 1
$\#n_1$	→	$\#n_2$
T/F	→	0/1
"*string*$_1$"	→	"*string*$_2$"
'*symb*'	→	'NOT *symb*'

NUM Command: Returns the character code n for the first character in the string.

Level 1	→	Level 1
"*string*"	→	*n*

NUMX Command: Sets the number of x-steps for each y-step in 3D perspective plots.

Level 1	→	Level 1
n_x	→	

NUMY Command: Sets the number of y-steps across the view volume in 3D perspective plots.

Level 1	→	Level 1
n_y	→	

NΣ Command: Returns the number of rows in the current statistical matrix (reserved variable ΣDAT).

Level 1	→	Level 1
	→	n_{rows}

OBJ→ Command: Separates an object into its components onto the stack. For some object types, the *number* of components is returned to level 1.

Level 1	→	Level n+1 ...	Level 2	Level 1
(*x*,*y*)	→		*x*	*y*
{ obj_1 ... obj_n }	→	obj_1	obj_n	*n*
[x_1 ... x_n]	→	x_1	x_n	{ *n* }
[[$x_{1\,1}$ ... $x_{m\,n}$]]	→	$x_{1\,1}$	$x_{m\,n}$	{ *m n* }
"*obj*"	→			*evaluated-object*
'*symb*'	→	arg_1 ... arg_n	*n*	'*function*'
x_unit	→		*x*	1_*unit*
:*tag*:*obj*	→		*obj*	"*tag*"

OR Function: Returns the logical OR of two arguments.

Level 2	Level 1	→	Level 1
$\#n_1$	$\#n_2$	→	$\#n_3$
"$string_1$"	"$string_2$"	→	"$string_3$"
T/F_1	T/F_2	→	0/1
T/F	'*symb*'	→	'*T/F* OR *symb*'
'*symb*'	*T/F*	→	'*symb* OR *T/F*'
'$symb_1$'	'$symb_2$'	→	'$symb_1$ OR $symb_2$'

H

ORDER Command: Reorders the variables in the current directory (shown in the VAR menu) to the order specified.

Level 1	→	Level 1
{ $global_1$ ... $global_n$ }	→	

PARITY Command: Sets the parity value in the reserved variable *IOPAR*.

Level 1	→	Level 1
n_{parity}	→	

PATH Command: Returns a list specifying the path to the current directory.

Level 1	→	Level 1
	→	{ HOME *directory-name*$_1$ ... *directory-name*$_n$ }

PCOV Command: Returns the population covariance of the independent and dependent data columns in the current statistics matrix (reserved variable ΣDAT).

Level 1	→	Level 1
	→	$x_{pcovariance}$

PGDIR Command: Purges the named directory (whether empty or not).

Level 1	→	Level 1
'*global*'	→	

PMAX Command: Specifies (x, y) as the coordinates at the upper right corner of the display.

Level 1	→	Level 1
(*x*, *y*)	→	

PMIN Command: Specifies (x, y) as the coordinates at the lower left corner of the display.

Level 1	→	Level 1
(x,y)	→	

PREDV Command: Returns the predicted dependent-variable value $y_{dependent}$, based on the independent-variable value $x_{independent}$, the currently selected statistical model, and the current regression coefficients in the reserved variable ΣPAR.

Level 1	→	Level 1
$x_{independent}$	→	$y_{dependent}$

PREDX Command: Returns the predicted independent-variable value $x_{independent}$, based on the dependent-variable value $y_{dependent}$, the currently selected statistical model, and the current regression coefficients in the reserved variable ΣPAR.

Level 1	→	Level 1
$y_{dependent}$	→	$x_{independent}$

PREDY Command: Returns the predicted dependent-variable value $y_{dependent}$, based on the independent-variable value $x_{independent}$, the currently selected statistical model, and the current regression coefficients in the reserved variable ΣPAR.

Level 1	→	Level 1
$x_{independent}$	→	$y_{dependent}$

PROMPT Command: Displays the contents of "*prompt*" in the status area, and halts program execution.

Level 1	→	Level 1
"*prompt*"	→	

PSDEV Command: Calculates the population standard deviation of each of the m columns of coordinate values in the current statistics matrix (reserved variable ΣDAT).

Level 1	→	Level 1
	→	x_{psdev}
	→	$[\ x_{psdev1}\ x_{psdev2}\ \cdots\ x_{psdevm}\ ]$

PVAR Command: Calculates the population variance of the coordinate values in each of the m columns in the current statistics matrix (ΣDAT).

Level 1	→	Level 1
	→	$x_{pvariance}$
	→	$[\ x_{pvariance1}\ \cdots\ x_{pvariancem}\ ]$

PVIEW Command: Displays *PICT* with the specified coordinate at the upper left corner of the graphics display.

Level 1	→	Level 1
(*x*, *y*)	→	
{ #*n* #*m* }	→	
{ }	→	

QUOTE Function: Returns its argument unevaluated.

Level 1	→	Level 1
'symb'	→	'symb'
obj	→	obj

RATIO Function: Prefix form of / (divide) generated by the EquationWriter application.

Level 2	Level 1	→	Level 1
z_1	z_2	→	z_1 / z_2
[array]	[[matrix]]	→	[[array × $matrix^{-1}$]]
[array]	z	→	[array/z]
z	'symb'	→	'z/symb'
'symb'	z	→	'symb/z'
'$symb_1$'	'$symb_2$'	→	'$symb_1$ / $symb_2$'
#n_1	n_2	→	#n_3
n_1	#n_2	→	#n_3
#n_1	#n_2	→	#n_3
x_unit_1	y_unit_2	→	$(x/y)_unit_1$ /$unit_2$
x	y_unit	→	(x/y)_1/unit
x_unit	y	→	(x/y)_unit
'symb'	x_unit	→	'symb/x_unit'
x_unit	'symb'	→	'x_unit/symb'

H

RCEQ Command: Returns the unevaluated contents of the reserved variable *EQ* from the current directory.

Level 1	→	Level 1
	→	obj_{EQ}

RCLALARM Command: Recalls a specified alarm.

Level 1	→	Level 1
n_{index}	→	{ *date time* obj_{action} x_{repeat} }

RCLMENU Command: Returns the menu number of the currently displayed menu.

Level 1	→	Level 1
	→	x_{menu}

RCLΣ Command: Returns the current statistics matrix (the contents of reserved variable ΣDAT) from the current directory.

Level 1	→	Level 1
	→	*obj*

RECN Command: Prepares the HP 48 to receive a file from another Kermit device, and to store the file in a specified variable.

Level 1	→	Level 1
'*name*'	→	
"*name*"	→	

REPL Command: Replaces a portion of the level 3 target object with the level 1 object, beginning at a position specified in level 2.

Level 3	Level 2	Level 1	→	Level 1
[[*matrix*]]$_1$	$n_{position}$	[[*matrix*]]$_2$	→	[[*matrix*]]$_3$
[[*matrix*]]$_1$	{ n_{row} n_{column} }	[[*matrix*]]$_2$	→	[[*matrix*]]$_3$
[*vector*]$_1$	$n_{position}$	[*vector*]$_2$	→	[*vector*]$_3$
{ $list_{target}$ }	$n_{position}$	{ $list_1$ }	→	{ $list_{result}$ }
"$string_{target}$"	$n_{position}$	"$string_1$"	→	"$string_{result}$"
$grob_{target}$	{#*n* #*m*}	$grob_1$	→	$grob_{result}$
$grob_{target}$	(*x*,*y*)	$grob_1$	→	$grob_{result}$
PICT	{#*n* #*m*}	$grob_1$	→	
PICT	(*x*,*y*)	$grob_1$	→	

RES Command: Specifies the resolution of mathematical and statistical plots, where the resolution is the interval between values of the independent variable used to generate the plot.

Level 1	→	Level 1
$n_{interval}$	→	
#$n_{interval}$	→	

RKF Command: Computes the solution to an initial value problem for a differential equation, using the Runge-Kutta-Fehlberg (4,5) method.

H

Level 3	Level 2	Level 1	→	Level 2	Level 1
{ *list* }	x_{tol}	x_{Tfinal}	→	{ *list* }	x_{tol}
{ *list* }	{ x_{tol} x_{hstep} }	x_{Tfinal}	→	{ *list* }	x_{tol}

RKFERR Command: Returns the absolute error estimate for a given step h when solving an initial value problem for a differential equation using the Runge-Kutta-Fehlberg method.

Level 2	Level 1	→	Level 4	Level 3	Level 2	Level 1
{ *list* }	*h*	→	{ *list* }	*h*	y_{delta}	*error*

RKFSTEP Command: Computes the next solution step (h_{next}) to an initial value problem for a differential equation.

Level 3	Level 2	Level 1	→	Level 3	Level 2	Level 1
{ *list* }	x_{tol}	*h*	→	{ *list* }	x_{tol}	h_{next}

ROOT Command: Returns a real number x_{root} that is a value of the specified variable *global* for which the specified program or algebraic object most nearly evaluates to zero or a local extremum.

Level 3	Level 2	Level 1	→	Level 1
≪ *program* ≫	'*global*'	*guess*	→	x_{root}
≪ *program* ≫	'*global*'	{ *guesses* }	→	x_{root}
'*symb*'	'*global*'	*guess*	→	x_{root}
'*symb*'	'*global*'	{ *guesses* }	→	x_{root}

RRK Command: Computes the solution to an intial value problem for a differential equation with known partial derivatives.

Level 3	Level 2	Level 1	→	Level 2	Level 1
{ *list* }	x_{tol}	x_{Tfinal}	→	{ *list* }	x_{tol}
{ *list* }	{ x_{tol} x_{hstep} }	x_{Tfinal}	→	{ *list* }	x_{tol}

RRKSTEP Command: Computes the next solution step (h_{next}) to an intial value problem for a differential equation, and displays the method used to arrive at that result.

Lvl 4	Lvl 3	Lvl 2	Lvl 1	→	Lvl 4	Lvl 3	Lvl 2	Lvl 1
{ *list* }	x_{tol}	*h*	*last*	→	{ *list* }	x_{tol}	h_{next}	*current*

RSBERR Command: Returns an error estimate for a given step *h* when solving an initial values problem for a differential equation using the Rosenbrock method.

Level 2	Level 1	→	Level 4	Level 3	Level 2	Level 1
{ *list* }	*h*	→	{ *list* }	*h*	y_{delta}	*error*

SAME Command: Compares two objects, and returns a true result (1) if they are identical, and a false result (0) if they are not.

Level 2	Level 1	→	Level 1
obj_1	obj_2	→	0/1

SCALE Command: Adjusts the first two parameters in *PPAR*, (x_{min}, y_{min}) and (x_{max}, y_{max}), so that x_{scale} and y_{scale} are the new plot horizontal and vertical scales, and the center point doesn't change.

Level 2	Level 1	→	Level 1
x_{scale}	y_{scale}	→	

SCONJ Command: Conjugates the contents of a named object.

Level 1	→	Level 1
'*name*'	→	

SDEV Command: Calculates the sample standard deviation of each of the m columns of coordinate values in the current statistics matrix (reserved variable ΣDAT).

Level 1	→	Level 1
	→	x_{sdev}
	→	[x_{sdev1} x_{sdev2} ··· x_{sdevm}]

SEND Command: Sends a copy of the named objects to a Kermit device.

Level 1	→	Level 1
'*name*'	→	
{ $name_1$ ... $name_n$ }	→	
{{ $name_{old}$ $name_{new}$ } *name* ... }	→	

SIDENS Command: Calculates the intrinsic density of silicon as a function of temperature, x_T.

Level 1	→	Level 1
x_T	→	$x_{density}$
x_unit	→	*x_1/cm*^{3}
'*symb*'	→	'SIDENS(*symb*)'

H

SINV Command: Replaces the contents of the named variable with its inverse.

Level 1	→	Level 1
'*name*'	→	

SNEG Command: Replaces the contents of a variable with its negative.

Level 1	→	Level 1
'name'	→	

SOLVEQN Command: Starts the multiple equation solver for a specified set of equations.

Level 3	Level 2	Level 1	→	Level 1
n	*m*	*0/1*	→	

START Command: Begins START ... NEXT and START ... STEP definite loop structures.

	Level 2	Level 1	→	Level 1
START	x_{start}	x_{finish}	→	
NEXT			→	
STEP		$x_{increment}$	→	
STEP		$'symb_{increment}'$	→	

STOALARM Command: Stores an alarm in the system alarm list and returns its alarm index number.

H

Level 1	→	Level 1
x_{time}	→	n_{index}
{ *date time* }	→	n_{index}
{ *date time* obj_{action} }	→	n_{index}
{ *date time* obj_{action} x_{repeat} }	→	n_{index}

STO+ Command: Adds a number or other object to the contents of a specified variable.

Level 2	Level 1	→	Level 1
obj	'*name*'	→	
'*name*'	*obj*	→	

STO− Command: Calculates the difference between a number (or other object) and the contents of a specified variable, and stores the new value to the specified variable.

Level 2	Level 1	→	Level 1
obj	'*name*'	→	
'*name*'	*obj*	→	

STO∗ Command: Multiplies the contents of a specified variable by a number or other object.

Level 2	Level 1	→	Level 1
obj	'*name*'	→	
'*name*'	*obj*	→	

STO/ Command: Calculates the quotient of a number (or other object) and the contents of a specified variable, and stores the new value to the specified variable.

Level 2	Level 1	→	Level 1
obj	'*name*'	→	
'*name*'	*obj*	→	

STOΣ Command: Stores *obj* in the reserved variable ΣDAT.

Level 1	→	Level 1
obj	→	

→STR Command: Converts any object to string form.

Level 1	→	Level 1
obj	→	"*obj*"

SYSEVAL Command: Evaluates unnamed operating system objects specified by their memory addresses.

Level 1	→	Level 1
$\#n_{address}$	→	

→Tag Command: Combines objects in levels 1 and 2 to create tagged (labeled) object.

Level 2	Level 1	→	Level 1
obj	"*tag*"	→	:*tag*:*obj*
obj	'*name*'	→	:*name*:*obj*
obj	*x*	→	:*x*:*obj*

H

TDELTA Function: Calculates a temperature change.

Level 2	Level 1	→	Level 1
x	*y*	→	x_{delta}
x_unit1	*y_unit2*	→	x_unit1_{delta}
x_unit	'*symb*'	→	'TDELTA(*x_unit*,*symb*)'
'*symb*'	*y_unit*	→	'TDELTA(*symb*,*y_unit*)'
'$symb_1$'	'$symb_2$'	→	'TDELTA($symb_1$,$symb_2$)'

TIME Command: Returns the system time in the form HH.MMSSs.

Level 1	→	Level 1
	→	*time*

TINC Command: Calculates a temperature increment.

Level 2	Level 1	→	Level 1
$x_{initial}$	y_{delta}	→	x_{final}
x_unit1	y_unit2_{delta}	→	x_unit1_{final}
x_unit	*'symb'*	→	'TINC(*x_unit,symb*)'
'symb'	y_unit_{delta}	→	'TINC(*symb,y_unit*$_{delta}$)'
'$symb_1$'	'$symb_2$'	→	'TINC($symb_1$,$symb_2$')'

TOT Command: Computes the sum of each of the m columns of coordinate values in the current statistics matrix (reserved variable ΣDAT).

Level 1	→	Level 1
	→	x_{sum}
	→	$[\ x_{sum1}\ x_{sum2}\ \cdots\ x_{summ}\ ]$

TRANSIO Command: Specifies the character translation option. These translations affect only ASCII Kermit transfers and files printed to the serial port.

Level 1	→	Level 1
n_{option}	→	

TVARS Command: Lists all global variables in the current directory that contain objects of the specified types.

Level 1	→	Level 1
n_{type}	→	{ *global* ... }
{ n_{type} ... }	→	{ *global* ... }

TVMROOT Command: Solves for the specified TVM variable using values from the remaining TVM variables.

Level 1	→	Level 1
'*TVM variable*'	→	$x_{TVMvariable}$

TYPE Command: Returns the type number of an object.

Level 1	→	Level 1
obj	→	n_{type}

Object Type Numbers

Object Type	Number
User Objects:	
Real number	0
Complex number	1
Character string	2
Real array	3
Complex array	4
List	5
Global name	6
Local name	7
Program	8
Algebraic object	9
Binary integer	10
Graphics object	11
Tagged object	12
Unit object	13
XLIB name	14

Object Type	Number
Directory	15
Library	16
Backup object	17
Built-in Commands:	
Built-in function	18
Built-in command	19
System Objects:	
System binary	20
Extended real	21
Extended complex	22
Linked array	23
Character	24
Code object	25
Library data	26
External object	26-31

VAR Command: Calculates the sample variance of the coordinate values in each of the m columns in the current statistics matrix (ΣDAT).

Level 1	→	Level 1
	→	$x_{variance}$
	→	$[\ x_{variance1}\ \cdots\ x_{variancem}\ ]$

VARS Command: Returns a list of all variables' names in the VAR menu (the current directory).

Level 1	→	Level 1
	→	$\{\ global_1\ \ldots\ global_n\ \}$

VTYPE Command: Returns the type number of the object contained in the named variable.

Level 1	→	Level 1
'*name*'	→	n_{type}
$:n_{port} : name_{backup}$	→	n_{type}
$:n_{port} : n_{library}$	→	n_{type}

***W Command:** Multiplies a plot's horizontal scale by x_{factor}.

Level 1	→	Level 1
x_{factor}	→	

WAIT Command: Suspends program execution for specified time, or until a key is pressed.

Level 1	→	Level 1
x	→	
0	→	x_{key}
−1	→	x_{key}

WHILE Command: Starts the WHILE ... REPEAT ... END indefinite loop structure.

Level 1		→	Level 1
WHILE		→	
REPEAT	*T/F*	→	
END		→	

WSLOG Command: Returns four strings recording the date, time, and cause of the four most recent warmstart events.

Level 1	→	Level 4 ... Level 1
	→	"log_4" ... "log_1"

ΣX Command: Sums the values in the independent-variable column of the current statistical matrix (reserved variable ΣDAT).

H

Level 1	→	Level 1
	→	x_{sum}

ΣX^2 Command: Sums the squares of the values in the independent-variable column of the current statistical matrix (reserved variable ΣDAT).

Level 1	→	Level 1
	→	x_{sum}

XCOL Command: Specifies the independent-variable column of the current statistics matrix (reserved variable ΣDAT).

Level 1	→	Level 1
n_{col}	→	

XRECV Command: Prepares the HP 48 to receive an object via XModem. The received object is stored in the given variable name.

Level 1	→	Level 1
'*name*'	→	

XRNG Command: Specifies the x-axis display range.

Level 2	Level 1	→	Level 1
x_{min}	x_{max}	→	

XSEND Command: Sends a copy of the named object via XModem.

Level 1	→	Level 1
'*name*'	→	

XVOL Command: Sets the width of the view volume in the reserved variable *VPAR*.

Level 2	Level 1	→	Level 1
x_{left}	x_{right}	→	

XXRNG Command: Specifies the x range of an input plane (domain) for GRIDMAP and PARSURFACE plots.

Level 2	Level 1	→	Level 1
x_{min}	x_{max}	→	

ΣX*Y Command: Sums the products of each of the corresponding values in the independent- and dependent-variable columns of the current statistical matrix (reserved variable *ΣDAT*).

Level 1	→	Level 1
	→	x_{sum}

ΣY Command: Sums the values in the dependent variable column of the current statistical matrix (reserved variable *ΣDAT*).

Level 1	→	Level 1
	→	x_{sum}

ΣY^2 Command: Sums the squares of the values in the dependent variable column of the current statistical matrix (reserved variable *ΣDAT*).

Level 1	→	Level 1
	→	x_{sum}

YCOL Command: Specifies the dependent variable column of the current statistics matrix (reserved variable $\Sigma;DAT$).

Level 1	→	Level 1
n_{col}	→	

YRNG Command: Specifies the y-axis display range.

Level 2	Level 1	→	Level 1
y_{min}	y_{max}	→	

YVOL Command: Sets the depth of the view volume in the reserved variable $VPAR$.

Level 2	Level 1	→	Level 1
y_{near}	y_{far}	→	

YYRNG Command: Specifies the y range of an input plane (domain) for GRIDMAP and PARSURFACE plots.

Level 2	Level 1	→	Level 1
y_{near}	y_{far}	→	

ZFACTOR Function: Calculates the gas compressibility correction factor for nonideal behavior of a hydrocarbon gas.

Level 2	Level 1	→	Level 1
x_{Tr}	y_{Pr}	→	$x_{Zfactor}$
x_{Tr}	'*symb*'	→	'ZFACTOR(x_{Tr},*symb*)'
'*symb*'	y_{Pr}	→	'ZFACTOR(*symb*,y_{Pr})'
'$symb_1$'	'$symb_2$'	→	'ZFACTOR($symb_1$,$symb_2$)'

ZVOL Command: Sets the height of the view volume in the reserved variable *VPAR*.

Level 2	Level 1	→	Level 1
x_{low}	x_{high}	→	

< Function: Tests whether one object is less than another object.

Level 2	Level 1	→	Level 1
x	*y*	→	0/1
$\#n_1$	$\#n_2$	→	0/1
"$string_1$"	"$string_2$"	→	0/1
x	'symb'	→	'x<symb'
'symb'	*x*	→	'symb<x'
'$symb_1$'	'$symb_2$'	→	'$symb_1$ < $symb_2$'
x_unit_1	y_unit_2	→	0/1
x_unit	'symb'	→	'x_unit<symb'
'symb'	*x_unit*	→	'symb<x_unit'

≤ Function: Tests whether one object is less than or equal to another object.

Level 2	Level 1	→	Level 1
x	*y*	→	0/1
$\#n_1$	$\#n_2$	→	0/1
"$string_1$"	"$string_2$"	→	0/1
x	'symb'	→	'x ≤ symb'
'symb'	*x*	→	'symb ≤ x'
'$symb_1$'	'$symb_2$'	→	'$symb_1$ ≤ $symb_2$'
x_unit_1	y_unit_2	→	0/1
x_unit	'symb'	→	'x_unit ≤ symb'
'symb'	*x_unit*	→	'symb ≤ x_unit'

> Function: Tests whether one object is greater than another object.

Level 2	Level 1	→	Level 1
x	*y*	→	0/1
#n_1	#n_2	→	0/1
"$string_1$"	"$string_2$"	→	0/1
x	'*symb*'	→	'*x*>*symb*'
'*symb*'	*x*	→	'*symb*>*x*'
'$symb_1$'	'$symb_2$'	→	'$symb_1$ >$symb_2$'
x_unit_1	y_unit_2	→	0/1
x_unit	'*symb*'	→	'*x_unit*>*symb*'
'*symb*'	*x_unit*	→	'*symb*>*x_unit*'

≥ Function: Tests whether one object is greater than or equal to another object.

Level 2	Level 1	→	Level 1
x	*y*	→	0/1
#n_1	#n_2	→	0/1
"$string_1$"	"$string_2$"	→	0/1
x	'*symb*'	→	'*x* ≥ *symb*'
'*symb*'	*x*	→	'*symb* ≥ *x*'
'$symb_1$'	'$symb_2$'	→	'$symb_1$ ≥ $symb_2$'
x_unit_1	y_unit_2	→	0/1
x_unit	'*symb*'	→	'*x_unit* ≥ *symb*'
'*symb*'	*x_unit*	→	'*symb* ≥ *x_unit*'

== Function: Tests if two objects are equal.

Level 2	Level 1	→	Level 1
obj_1	obj_2	→	0/1
(x,0)	x	→	0/1
x	(x,0)	→	0/1
z	'symb'	→	'z==symb'
'symb'	z	→	'symb==z'
'$symb_1$'	'$symb_2$'	→	'$symb_1$ ==$symb_2$'

≠ Function: Tests if two objects are not equal.

Level 2	Level 1	→	Level 1
obj_1	obj_2	→	0/1
(x,0)	x	→	0/1
x	(x,0)	→	0/1
z	'symb'	→	'z ≠ symb'
'symb'	z	→	'symb ≠ z'
'$symb_1$'	'$symb_2$'	→	'$symb_1$ ≠ $symb_2$'

∫ Function: Integrates an *integrand* from *lower limit* to *upper limit* with respect to a specified variable of integration.

H

Level 4	Level 3	Level 2	Level 1	→	Level 1
lower limit	upper limit	integrand	'name'	→	'$symb_{integral}$'

∂ Function: Takes the derivative of an expression, number, or unit object with respect to a specified variable of differentiation.

Level 2	Level 1	→	Level 1
'$symb_1$'	'name'	→	'$symb_2$'
z	'name'	→	0
x_unit	'name'	→	0

Σ Function: Calculates the value of a finite series.

Level 4	Level 3	Level 2	Level 1	→	Level 1
'indx'	x_{init}	x_{final}	smnd	→	x_{sum}
'indx'	'init'	x_{final}	smnd	→	'Σ(indx=init,x_{final},smnd)'
'indx'	x_{init}	'final'	smnd	→	'Σ(indx=x_{init},final,smnd)'
'indx'	'init'	'final'	smnd	→	'Σ(indx=init,final,smnd)'

Σ+ Command: Adds one or more data points to the current statistics matrix (reserved variable ΣDAT).

Level m ... Level 2	Level 1	→ Level 1
	x	→
	$[\ x_1\ \ x_2 \cdots\ x_m\ ]$	→
	$[[\ x_{11} \cdots\ x_{1m}\ ]\ [\ x_{n1} \cdots\ x_{nm}\ ]]$	→
$x_1 \cdots\ x_{m-1}$	x_m	→

Σ− Command: Returns a vector of m real numbers (or one number x if $m = 1$) corresponding to the coordinate values of the last data point entered by Σ+ into the current statistics matrix (reserved variable ΣDAT).

→	Level 1
→	x
→	$[\ x_1\ \ x_2\ \cdots\ x_m\ ]$

Index

Special characters

⧖ annunciator, 1-2
↰ ↱ annunciators, 1-2, 1-6
α annunciator, 1-2, 2-2
((•)) annunciator, 1-2, 26-4, A-1, A-4
≫ annunciator, 1-2
1USR annunciator, 1-3, 30-5
R∠Z annunciator, 1-3, 12-11, 13-2
R∠∠ annunciator, 1-3, 12-11, 13-2
▚ cursor, 2-12
▓ cursor, 2-12
ℙ character, 2-7
∠ character
  complex number separator, 12-12
  vector separator, 4-4, 13-3
= character, 11-4, 18-1, 22-1
& wildcard, 20-28, 28-4
... character, A-3
delimiter, 15-1
_ delimiter, 10-2
π
  in fraction conversions, 16-6
  numeric value, A-2
  symbolic constant, A-2
ΣDAT
  reserved variable, 5-6
ΣPAR
  plots and, 22-16
  reserved variable, 5-6
  statistical parameters, 21-12

A

absolute value, 12-9, 12-13
accuracy
  of integrals, 20-6
  of linear solutions, 14-17
acknowledging alarms, 26-4
alarms
  acknowledging, 26-4
  annunciator, 26-4
  appointment type, 26-2
  controlling beeper, 26-6
  control type, 26-2
  index number, 26-4
  no response needed, 26-4
  not responding, 26-4
  past due, 26-4
  repeating, 26-6
  responding to, 26-4
  saving, 26-5
  stopping repeating, 26-5
  types, 26-2
ALG annunciator, 1-3
Algebraic-entry mode, 2-8
Algebraic/Program-entry mode, 2-9

algebraics
action in programs, 29-2
array elements in, 14-12
as graphics objects, 7-13
as strings, 7-14
collecting terms, 20-18
converting to graphics objects, 7-13
converting to strings, 7-14
creating graphics objects from, 9-9
delimiters, 2-6
differentiating, 20-9
editing in command line, 7-10
editing in EquationWriter, 7-9
editing subexpressions, 7-10
entering, 2-6, 7-3
entry modes, 2-8, 2-9
evaluating, 11-3
evaluating selectively, 20-17
expanding terms, 20-18
general solutions, 20-16
in local variable structure, 29-3, 29-16
inserting stack objects, 7-11
integrating numerically, 20-1
integrating symbolically, 20-7
plotting, 22-1
precedence of operators, 11-3
principal solutions, 20-16
rearranging, 20-17, 20-27
replacing subexpressions, 7-12
showing hidden variables, 20-17
solving graphically, 22-10
solving numerically, 22-10
solving symbolically, 20-14, 20-15
subexpressions, 7-10, 7-12, 20-18, 20-19
types, 11-4, 18-1
unit objects in, 10-9
algebraic syntax, 11-2
in local variable structures, 29-4
user-defined functions, 11-8
algebraic transformations
built-in, 20-19
conditional, 20-28
user-defined, 20-27
wildcards in user-defined, 20-27, 20-28
Alpha-entry mode, 2-2
alpha keyboard
diagram, 2-3
locking lowercase, 2-4
locking on, 2-4
operation, 1-4, 2-2
setting automatic lock, 4-9
ALRMDAT
reserved variable, 5-6
amortization (TVM)
calculations, 18-13, 18-19
display mode, 18-19
payment modes, 18-14, 18-19
analytic functions, 11-1
angle modes, 4-3
affect complex numbers, 12-12
affect implied units, 25-11
affect trig functions, A-2
affect vectors, 4-4, 13-3
annunciators for, 1-3
changing, 4-4
Degrees, 4-3
Grads, 4-3
Radians, 4-3
angles
converting, 12-7

dimensionless units, 10-7, 10-8
HMS format, 12-7
angular units
converting, 10-7
animation
graphics objects, 9-10
YSLICE plots, 23-23, 23-33
annunciators
1USR, 1-3
alarm (•), 1-2, 26-4
ALG, 1-3
alpha α, 1-2
busy ⧗, 1-2
HALT, 1-3
indicate status, 1-1
I/O ≫, 1-2
listed, 1-2
Polar/Cylindrical mode R∠Z, 1-3
Polar/Spherical mode R∠∠, 1-3
PRG, 1-3
RAD, 1-3
shift keys ↰ ↱, 1-2, 1-6
USER, 1-3
user flags (1 2 3 4 5), 1-3
antiderivatives, 20-30
application cards
expand ROM, 5-1
installing, 28-9, 28-12
removing, 28-14
ROM-based libraries, 28-6
applications, 1-6
CHARS, 1-6
command menus and, 1-7
EQ LIB, 1-6
I/O, 1-6
LIBRARY, 1-6
MEMORY, 1-6
MODES, 1-6
PLOT, 1-6
SOLVE, 1-6
STACK, 1-6
STAT, 1-6
SYMBOLIC, 1-7
TIME, 1-7
UNITS, 10-1
appointment alarms, 26-2
arcs
drawing, 9-8
arguments, 3-1
bad, A-3
multiple, 3-2
on stack, 3-1
recalling last, 3-5
stack syntax, 3-1
too few, A-3
arithmetic
functions, 12-1
with arrays, 14-11
with temperatures, 10-11
with time, 16-4
with units, 10-7, 10-11
arrays
applying functions to, 14-13
arithmetic, 14-11
calculations, 14-11
column norm, 14-8
combining, 14-13
constant, 14-2
converting complex to real, 14-13
converting real to complex, 14-13
creating special, 14-2
dimensions, 14-8
disassembling, 14-13
editing, 8-5
entering, 2-6
entering with MatrixWriter, 8-2, 14-1

extracting columns, 14-6
extracting elements, 8-8, 14-7
extracting real part, 14-14
extracting rows, 14-6
Frobenius norm, 14-8
in algebraics, 14-12
inserting columns, 14-5
norms, 14-8
one-column, 8-1, 8-8
one-row, 8-1, 8-8
printing, 27-2
random, 14-3
redimensioning, 14-10
replacing elements, 14-7
row norm, 14-8
spectral norm, 14-8
swapping columns, 14-6
swapping rows, 14-6
vectors, 8-1

B

b (binary base marker), 15-1
backslash translations, 27-16
backspace
in command line, 2-8
in EquationWriter, 7-10
backup identifiers, 28-3
backup objects
all user memory, 28-5
creating, 28-3
directories, 28-3
evaluating, 28-4
identifiers, 28-3
in custom menus, 30-2
in independent memory, 28-3
in port 0, 28-3
listing, 28-4
moving to port 0, 28-16
moving to RAM card, 28-17
purging, 28-4
recalling, 28-3
restoring memory from, 28-6
wildcards, 28-4
BAR plots, 23-17
from Statistics, 23-19
base (binary)
affects display, 15-1
options, 15-1
setting, 15-1
typing, 15-2
BASE menu, 15-1, 15-4
batteries
calculator, A-4
changing (calculator), A-5
changing (RAM card), A-7
disposing, A-6, A-8
in new RAM cards, 28-10
low-battery warning, A-4
preserve RAM-card, 28-15
RAM card, A-4
types, A-5
when to replace, 28-11, A-4
beeper
controlling, 4-10
for alarms, 26-6
Begin mode (TVM), 18-14, 18-19
best editing environment, 2-11, 3-7
binary integers
as pixel coordinates, 9-7
bases, 15-1
bits displayed, 15-2
bits lost, 15-2
calculations, 15-2, 15-3
converting to real integers, 15-3
delimiters, 15-1
displaying, 15-1
entering, 15-2
internal representation, 15-2
logic operations, 15-4

rotating, 15-5
shifting, 15-5
boxes
drawing, 9-8
branching structures
conditional structures, 29-10
loop structures, 29-12
program element, 29-3
break (serial), 27-18
buffer (serial), 27-18, 27-19
"building-block" programs, 29-5
bytes
available memory, A-1
of built-in memory, 5-1

C

cable (serial), 27-7
calculation form (Equation Library)
equations, 25-4
calculator
battery type, A-5
environmental limits, A-4
fixing problems, A-1
halting, 5-16
questions about, A-1
repair service, A-16
testing, A-9, A-10, A-11, A-12, A-13, A-14
warranty, A-15
won't turn on, A-9
capital letters
keying in, 1-5
"case" branching, 29-11
catalogs
Equation Library, 25-1
chain calculations, 3-3
character codes
backslash translations, 27-16
characters
alpha keyboard diagram, 2-3
backslash translations, 27-16
keying in, 2-2
size in graphics objects, 9-9
uppercase and lowercase, 2-4
viewing keystrokes, 2-5
viewing numbers, 2-5
CHARS application, 1-6, 2-4
checksums
HP 48-to-HP 48, 27-1
verify backup objects, 28-3
chi-square distribution, 12-5
circles
drawing, 9-8
plotting, 23-11
clearing
flags, 4-8
memory, 5-17
stack, 3-5
user keys, 30-6
clock
displaying, 4-10
format options, 16-1, 26-1
ticks, 16-4
column vectors, 8-1, 8-8
combinations, 12-4
comma
complex number separator, 12-12
fraction mark, 4-6
command line
arguments go on stack, 3-1
comments in, 2-7
cursor keys, 2-8
deleting, 1-8
editing, 2-1, 2-8
editing environment, 2-12
entering objects, 2-5
entry modes, 2-8

insert and replace modes, 2-12
inserting stack object, 3-9
keying in special characters, 2-4
multiple arguments, 3-2
multiple objects, 2-7
operation, 1-4, 2-7
processing, 2-8
recalling previous, 2-10
stack and, 1-4
using in EquationWriter, 7-10
command menus
applications and, 1-7
commands
applying to lists, 17-2, 17-3
fraction conversion, 16-5
general math, 12-1
in custom menus, 30-2
in programs, 29-2
stack syntax, 3-1
subset of operations, 11-1
type of object, 11-1
comments in command line, 2-7
complex conjugates, 12-13
complex numbers
as plot coordinates, 9-7
assembling from real numbers, 12-14
calculations, 12-13
conjugates, 12-13
converting to real numbers, 12-13, 12-14
coordinate modes, 12-11
delimiters ( ), 12-12, 13-3
disassembling into real numbers, 12-14
displaying, 12-11
entering, 12-12
from real-number calculations, 12-13
internal representation, 12-12
normalized, 12-12
polar components, 12-11
rectangular components, 12-11
compound-interest calculations (TVM), 18-13
computer
connecting to HP 48, 27-7
file names, 27-11
restoring HP 48 memory, 27-13
conditional structures
"case" branching, 29-11
error branching, 29-15
"if" branching, 29-10, 29-15
program element, 29-3
test commands in, 29-10
CONIC plots, 23-11
default step size, 23-13
plotting ranges, 24-3
valid equations, 23-11
conjugate complex numbers, 12-13
conjugates (matrices), 14-14
constants
built-in, 11-4
in equations, 25-12
list of, 25-12
numeric, 11-4
symbolic, 11-4, 11-5
Constants Library, 25-12
continuing program execution, 29-9
control alarms, 26-2
converting
algebraics to graphics objects, 7-13
algebraics to strings, 7-14

angular units, 10-7
binary integers to real integers, 15-3
complex arrays to real arrays, 14-13
complex numbers to real numbers, 12-13, 12-14
dates to numbers, 16-1, 16-2
dates to strings, 16-4
decimal format to HMS, 12-7, 16-3
degrees to radians, 12-7
HMS format to decimal, 12-7, 16-3
numbers to fractions, 16-5
objects to graphics objects, 9-9
pixels to user-units, 9-9
radians to degrees, 12-7
real arrays to complex arrays, 14-13
real numbers to binary integers, 15-3
real numbers to complex numbers, 12-14
stack displays to graphics objects, 9-10
temperature units, 10-10
times to numbers, 16-1, 16-3
units, 10-6
user-units to pixels, 9-9
coordinate modes
affect complex numbers, 12-11
affect vectors, 4-4, 13-1, 13-3
annunciators for, 1-3
changing, 12-11, 13-2
Cylindrical, 4-5, 12-11, 13-1
Polar, 4-4, 12-11, 13-1
Rectangular, 4-4, 12-11, 13-1
Spherical, 4-5, 12-11, 13-1
coordinates
pixel, 9-7
user-unit, 9-7
coordinates (plot), 9-7, 9-9
counters
loop structures, 29-12, 29-13
negative steps, 29-13, 29-14
critical points
viewing in graph, 22-11
cross products, 13-4
CST
reserved variable, 5-6, 30-1
CST menu, 30-1
current directory
changing, 5-8
determines VAR menu, 5-4
displayed in status area, 1-1, 5-3
path of, 1-1, 5-3
variables created in, 5-6
current equation, 5-6
solving, 22-10
current statistical matrix, 21-1
cursor
command line, 2-12
coordinate modes, 22-4
displaying coordinates, 22-4
insert, 2-12
moving, 1-7
replace, 2-12
cursor keys
using, 1-7
custom menus
creating, 30-1
custom labels, 30-3
in each directory, 30-3
in SOLVR environment, 18-9
objects in, 30-1
shifted keys, 30-4
switching, 30-3
typing aids in, 30-1

user-defined units, 10-14, 25-15
Cylindrical coordinate mode, 4-5, 12-11, 13-1
Cylindrical mode, 4-10

D

d (decimal base marker), 15-1
date
converting to number, 16-1, 16-2
converting to string, 16-4
displaying, 4-10
format options, 16-1, 26-1
setting the current, 16-1
debugging programs, 29-8, 29-9
decimal format
converting to HMS, 12-7, 16-3
decimal places
number of displayed, 4-3
defining
user-defined functions from equations, 11-7
variables from equations, 5-13, 11-4
defining procedures
local variables in, 29-18
local variable structures, 29-16
degrees
converting to radians, 12-7
Degrees angle mode, 4-3
Degrees mode, 4-10
delimiters
entering, 2-5
' for algebraics, 2-6
[ ] for arrays, 2-6, 8-1
for binary integers, 15-1
( ) for complex numbers, 12-12
[ ] for matrices, 2-6, 8-1
« » for programs, 29-1
_ for unit objects, 10-2
[ ] for vectors, 2-7, 4-4, 8-1, 13-3
list of, 2-6
prevent evaluation of names, 5-14
dependent variable
plotting range, 24-3
derivatives
calculating numerically, 20-9
calculating symbolically, 20-10
"der" variables, 5-6, 20-11
in EquationWriter, 7-5
in PICTURE environment, 22-10, 22-11
user-defined, 20-11
"der" names, 20-11
variables, 5-6
DIFF EQ plots, 19-7, 23-11
initial values, 19-8
variables, 19-8
differential equations
initial value problems, 19-2
plotting second-order, 19-12
plotting solutions, 19-7
plotting stiff, 19-9
plotting vector-valued, 19-12
reducing order of, 19-5, 19-12
second-order, 19-5
solving, 19-1
stiff, 19-4
vector-valued, 19-5, 19-12
differentiation
implicit, 20-12
of algebraics, 20-9
dimensional consistency, 25-10
dimensionless units, 10-7

directories, 5-3
backing up, 28-3
changing current, 5-8
creating, 5-7
current directory, 5-3
custom menu in, 30-3
Equation Library, 25-3
evaluating variables containing, 5-13
in custom menus, 30-1
paths, 5-4
root directory, 5-3
stored in variables, 5-4
variables in, 5-4
display
annunciators, 1-1, 1-2
command line, 1-4
current path, 1-1
messages, 1-1
number format, 4-2
organization, 1-1
showing clock, 26-1
stack levels, 1-3
status area, 1-1
display form (Equation Library)
equations, 25-4
display modes
affect fraction conversions, 16-5
affect rounding, 12-9
affect truncating, 12-9
changing, 4-3
control number format, 4-2
Engineering, 4-2
Fix, 4-2
Scientific, 4-2
Standard, 4-2
display ranges
PARAMETRIC plots, 23-8
plotting ranges and, 24-3
POLAR plots, 23-5
TRUTH plots, 23-15
"do" looping, 29-14
dot products, 13-4
duplicating stack entries, 3-4

E

E (in numbers), 2-2, A-2
editing
algebraics, 7-9
arrays, 8-5
canceling changes, 2-10
in EquationWriter, 7-9
in MatrixWriter, 8-5
inserting objects into algebraics, 7-11
programs, 29-8
stack objects, 2-10
statistical data, 21-4
subexpressions, 7-10, 7-12, 20-20
user key assignments, 30-8
variables, 2-10
EDIT menu, 2-12
elapsed time
calculating, 16-4
ellipses
plotting, 23-11
ellipsis (...) in display, A-3
End mode (TVM), 18-14, 18-19
Engineering display mode, 4-2
entry modes
Algebraic, 2-8
Algebraic/Program, 2-9
Alpha, 2-2
annunciators for, 1-3
changing manually, 2-9
command line, 2-8
EquationWriter, 7-2
Immediate, 2-8
Program, 2-9
environmental limits, A-4

environments
best, 2-11, 3-7
editing, 2-10
EquationWriter, 7-2
exiting, 1-8
Interactive Stack, 3-6
MatrixWriter, 8-2
Selection, 7-10, 20-19
EQ
created by Equation Library, 25-2, 25-6
FUNCTION plots and, 23-1, 23-3
GRIDMAP plots and, 23-34
PARAMETRIC plots and, 23-7, 23-9
POLAR plots and, 23-4
PR-SURFACE plots and, 23-36
reserved variable, 5-6
SLOPEFIELD plots and, 23-26
TRUTH plots and, 23-14
EQ LIB application, 1-6
equal sign, 11-4, 18-1, 22-1
Equation Library, 25-1
catalogs, 25-1
chooses solver, 25-2
entering values for variables, 25-3
equation information, 25-4
interpreting results, 25-10
Minehunt game, 25-14
Multiple-Equation Solver, 25-2
pictures in, 25-5
purging variables, 25-5, A-3
reference, F-1
setting units options, 25-1, 25-5
SOLVE application and, 25-2
solving for variables, 25-3
solving problems, 25-1
starting solvers, 25-2
subjects, 25-4, F-1
titles, 25-4, F-1
unexpected results, 25-10
units catalog, 25-5
units options, 25-5
user-defined units, 25-15
uses Multiple-Equation Solver, 25-6
variable catalogs, 25-5
variable names, 25-5
viewing equations, 25-4
equations, 11-4, 18-1, 22-1
arguments to functions, 11-4
calculating slope of, 22-10
compared to expressions, 11-4, 18-1, 22-1
creating, 2-6
creating in EquationWriter, 7-3
creating user-defined functions from, 11-7
creating variables from, 5-13, 11-4
critical points, 22-11
general solutions, 20-16
plotting, 22-1
polynomial approximations, 20-12, 20-15
principal solutions, 20-16
rearranging, 20-17, 20-27
slope of, 22-11
solving, 18-2
solving for unknown variable, 18-1
solving graphically, 22-10
solving linear systems, 14-12, 14-14, 14-15, 14-17, 14-18
solving numerically, 22-10

solving quadratic, 20-15
solving symbolically, 20-14, 20-15
equations (Equation Library)
calculation form, 25-4
designing, 25-8
display form, 25-4
function limitations, 25-9
multiple roots, 25-11
pictures, 25-5
robust, 25-9
solving, 25-1
too many unknowns, 25-8, 25-10
unexpected solutions, 25-10
unused, 25-10, 25-11
variables in, 25-8
viewing, 25-4
EquationWriter, 7-2
creating equations in, 7-3
creating unit objects in, 7-6, 10-4
editing in command line, 7-10
editing subexpressions, 7-10
editing with backspace, 7-10
entering derivatives, 7-5
entering exponents, 7-4
entering fractions, 7-4
entering integrals, 7-5
entering math operators, 7-4
entering names, 7-3
entering numbers, 7-3
entering parentheses, 7-5
entering powers, 7-5
entering roots, 7-4
entering summations, 7-5
entering units, 7-6
entering variables, 7-3
entering where function, 7-6
entry mode, 7-2
examples, 7-8
exiting, 7-3
implicit parentheses, 7-6
inserting stack objects, 7-11
modes, 7-2
operations, 7-13
replacing subexpressions, 7-12
Rules transformations, 20-19
scrolling mode, 7-2, 7-10, 7-13
Selection environment, 7-2, 7-11, 7-12, 7-13, 20-19
starting, 7-3
subexpressions, 7-10
errors
causes, 29-15
conditional structures, 29-15
controlling beeper, 4-10
during equation solution, 25-10
messages listed, B-1
numbers for, B-1
serial I/O, 27-18, 27-19
trapping, 29-15
evaluation
of algebraics, 11-3
of backup objects, 28-4
of local variables, 29-17
of symbolic constants, 11-5
of test clauses, 29-10, 29-11, 29-14
of variable names, 5-13
preventing for names, 5-14
exponential functions, 12-2
exponents
display format of, 4-3
fractional, A-2
in EquationWriter, 7-4
expressions, 11-4, 18-1, 22-1
compared to equations, 11-4, 18-1, 22-1

differentiating, 20-9
integrating numerically, 20-1
integrating symbolically, 20-7
plotting, 22-1
rearranging, 20-17, 20-27
solving, 18-2
solving symbolically, 20-14, 20-15
extremum
in SOLVE application, 18-4
of graph, 22-11
eyepoint, 23-24
PR-SURFACE plots, 23-37
requirements, 23-25
WIREFRAME plots, 23-30

F

factorials, 12-4
f distribution, 12-5
fields (input forms), 6-1
check, 6-2, 6-4
data, 6-2, 6-3, 6-4
determining valid object types, 6-6
extended data, 6-2, 6-3
list, 6-2, 6-4
resetting, 6-6
selecting, 6-2, 6-3
types of, 6-2
files
backing up memory, 27-12
choosing names, 27-11
restoring memory, 27-13
Finance Solver application, 18-13
amortization, 18-19
payment modes, 18-14
Fix display mode, 4-2
fixing problems, A-1
flags
Acknowledged Alarms Saved (−44), 26-5
Alarm Beep (−57), 26-6
Alpha Lock (−60), 4-9
backing up in backup object, 28-5
backing up on computer, 27-12
clearing, 4-8
Complex Mode (−19), 12-14
control modes with, 4-7
default states, 4-9, D-1
Infinite Result Exception (−22), 14-16
Numeric Results (−3), 5-13, 11-5
Principal Solution (−1), 20-16
RECV Overwrite (−36), 27-11
Repeat Alarms Not Rescheduled (−43), 26-6
setting, 4-8
Symbolic Constants (−2), 11-5
system, D-1
testing, 4-8
user, 4-9
"for" looping, 29-13
fractional exponents, A-2
fractional part of real number, 12-9
fraction mark, 4-6
affects complex numbers, 12-12
comma, 4-6
period, 4-6
setting, 4-6

fractions
converting real numbers to, 16-5
in EquationWriter, 7-4
freeing merged memory, 28-16, 28-17
function analysis, 22-9
FUNCTION plots, 23-1
default step size, 23-3
EQ in, 23-1, 23-3
PICTURE FCN tools and, 23-1
TRACE mode, 23-3
valid equations, 23-1
functions
analyzing plots of, 22-9
angle conversion, 12-7
applying to arrays, 14-13
applying to lists, 17-5
arithmetic (summary), 11-5
equations as arguments for, 11-4
exponential, 12-2
fraction conversion, 16-5
general math, 12-1
hyperbolic, 12-3
logarithmic, 12-2
number parts, 12-9
percent, 12-9
scientific (summary), 11-5
subset of commands, 11-1
trigonometric, 10-8, 12-2, A-2
type of object, 11-1
user-defined, 11-6, 24-2
functions of two variables
plotting, 23-22
future value (TVM), 18-20

G

Gamma function, 12-4
Gaussian elimination, 14-18
general solutions
equations and algebraics, 20-16
ISOL and QUAD and, 20-16
n1 and, 20-16
principal solutions and, 20-16
s1 and, 20-16
specifying, 20-16
global variables, 25-2, 25-5, 25-11, A-3
action in programs, 29-2
disadvantages in programs, 29-16
VAR menu, 5-11
GRAD annunciator, 1-3, 4-3
Grads angle mode, 4-3
Grads mode, 4-10
Graphics environment
adding elements, 9-3
pixel operations, 9-8
graphics objects
character size in, 9-9
creating from algebraics, 7-13
creating from objects, 9-9
extracting images, 9-10
printing, 27-2
size of, 9-10
superimposing, 9-9
viewing in stack display, 9-10
Greek letters
keying in, 1-5
translations, 27-16
GRIDMAP plots, 23-34
controlling output display, 23-34
EQ in, 23-34
output grid, 23-22, 23-34
guarantee, A-15

guessed solutions
  help find solution, 25-10, 25-11
  SOLVE application, 18-1, 18-3, 18-5
guesses solutions
  help find solution, 25-11

H

h (hex base marker), 15-1
HALT annunciator, 1-3, 29-9, A-3
halting programs, 29-9
halting the calculator, 5-16
hidden variables
  showing, 20-17
HISTOGRAM plots, 23-17
  from Statistics, 23-18
HMS format
  converting to decimal, 12-7, 16-3
  for angles, 12-7
  for time, 16-3
HOME directory
  backing up, 27-12, 28-5
  restoring, 27-13, 28-6
  switching to, 5-12
humidity limits, A-4
hyperbolas
  plotting, 23-11
hyperbolic functions, 12-3
HYP menu, 12-3

I

IERR (integration uncertainty), 20-7
"if" branching, 29-10, 29-15
imaginary part
  of complex matrices, 14-14
  of complex numbers, 12-13
Immediate-entry mode, 2-8
implicit parentheses, 7-6
independent memory
  backup objects in, 28-3
  expanding, 28-15
  libraries in, 28-8
  port 0, 28-2, 28-3
independent variable
  plotting range, 24-3
index number (alarm), 26-4
infrared port
  testing, A-13
infrared printers
  printing, 27-2
  setting up, 27-3
initial value differential equations
  solving, 19-2
  stiff, 19-4
input forms, 6-1
  commands, 6-7
  creating, 6-7
  determining valid object types, 6-6
  editing data, 6-4
  entering data from keyboard, 6-3
  entering lists, 6-3
  entering stored objects, 6-3
  executing, 6-7
  exiting, 6-7
  fields, 6-1
  global changes, 6-7
  labels, 6-1
  navigating in, 6-2
  prompt line, 6-1
  resetting fields, 6-6
  selecting fields, 6-2
  selecting options, 6-4
  SOLVE and PLOT, 6-7
  stack calculations and, 6-5
  titles, 6-1
  using a second, 6-5

insert cursor, 2-12
integer part of real number, 12-9
integrals
  accuracy, 20-6
  and Taylor polynomials, 20-9
  definite, 20-1, 20-7
  IERR contains uncertainty, 20-7
  improper, 20-2
  indefinite, 20-9
  in EquationWriter, 7-5
  in PICTURE environment, 22-10
  limiting accuracy, 20-6
  multiple, 20-5
  of non-integrable expressions, 20-9
  solving numerically, 20-1
  solving symbolically, 20-7
  uncertainty, 20-6
integration
  legal symbolic expressions, 20-7
  numeric, 20-1
  symbolic, 20-7
integration patterns
  symbolic, 20-30
Interactive Stack
  editing environment, 3-6
  keyboard, 3-9
  menu, 3-7
  operation, 3-7
  pointer, 3-6
interest rate (TVM), 18-20
intermediate results
  using on stack, 3-3
International System of Units, 10-2, 10-6
intersections, 22-10
inverse functions, 20-15
inverting
  matrices, 14-10
I/O
  backing up memory, 27-12
  computer connection, 27-7
  errors, 27-18, 27-19
  file-naming, 27-11
  HP 48-PC, 27-9
  HP 48-to-computer, 27-7, 27-12
  HP 48-to-HP 48, 27-1
  input buffer, 27-18, 27-19
  Kermit commands, 27-13
  Kermit protocol, 27-9
  non-Kermit commands, 27-15
  parameters for serial printer, 27-3
  protecting variables, 27-11
  restoring memory, 27-13
  serial commands, 27-15
  serial wiring, 27-7
  testing ports, A-13, A-14
  XON/XOFF pacing, 27-3, 27-17
I/O application, 1-6
I/O menu, 27-18
IOPAR
  reserved variable, 5-6

K

Kermit
  file transfer protocol, 27-9
  packets, 27-13
  sending commands, 27-13
keyboard
  alpha, 1-5, 2-2
  alpha diagram, 2-3
  alpha left-shifted, 1-5
  alpha right-shifted, 1-5
  assigning user keys, 30-5
  backspacing, 2-1

disabling user keys, 30-7
entering delimiters, 2-5
entering objects, 2-5
entering special characters, 2-4
EquationWriter, 7-2
in PICTURE environment, 22-5
Interactive Stack, 3-9
keying in characters, 2-2
keying in numbers, 2-1
keystrokes queued, 1-2
left-shifted, 1-4
locked up, 30-8, A-9
math functions, 12-1
menu keys, 1-8
multiple functions of, 1-4
organization, 1-4
primary, 1-4
right-shifted, 1-5
shift keys, 1-4, 1-6
testing operation, A-11
unassigning user keys, 30-6
user keys, 30-4
key location numbers, 30-5
keystrokes
queued, 1-2
viewing, 2-5
killing programs, 29-9

L

last arguments
recalling, 3-5
saving, 4-10
last command line
recalling, 2-10
saving, 4-10
last stack
restoring, 3-6
saving, 4-10
letters
keying in, 2-2
uppercase and lowercase, 2-4
LIBRARY application, 1-6
library objects
attaching, 28-7, 28-8
compared to programs, 28-6
contain objects, 28-6
detaching, 28-8
extend command set, 28-6
identifiers, 28-7
in independent memory, 28-8
limiting access, 28-8
moving to port 0, 28-16
names, 28-7
purging, 28-8
RAM- or ROM-based, 28-6
setting up, 28-7
linear equations, 14-12, 14-15, 14-17, 14-18
finding "best" solution, 14-15
matrix equations and, 18-11
solvable systems, 18-12
solving systems of, 14-14, 18-11, 18-12
Linear System Solver
allowable systems, 18-12
interpreting results, 18-12
testing solution, 18-12
using, 18-11
lines
drawing, 9-8
lists, 17-1
action in programs, 29-2
adding elements in two, 17-3
appending, 17-1, 17-2
applying commands to, 17-2
applying functions to, 17-4, 17-5
applying procedures to, 17-5
applying programs to, 17-4

concatenating, 17-3, 17-6
counting elements in, 17-7
creating from keyboard, 17-1
creating from stack, 17-1
dividing two, 17-4
entering, 2-5
finding elements, 17-7
manipulating, 17-6
multiple-argument commands and, 17-3
multiplying two, 17-4
placing elements on stack, 17-6, 17-7
placing first element on stack, 17-7
processing, 17-2
replacing elements, 17-7
reversing, 17-6
sorting, 17-6
subtracting two, 17-4
using with input forms, 6-3
local variables, 29-4
action in programs, 29-2
compiled, 29-18
creating, 29-3, 29-16
evaluating, 29-17
exist temporarily, 29-16, 29-17, 29-18
in subroutines, 29-18
naming, 29-16
using outside of defining procedure, 29-18
local variable structures
advantages, 29-17
as user-defined functions, 29-19
calculations with, 29-4
create local variables, 29-16
defining procedure, 29-16, 29-18
entering, 29-16
in user-defined functions, 29-18
operation, 29-3, 29-16
program element, 29-3
syntax, 29-3, 29-16
logarithmic functions, 12-2
logical functions, 15-4
loop structures
counters, 29-12, 29-13
definite, 29-12
"do" looping, 29-14
"for" looping, 29-13
indefinite, 29-12
negative steps, 29-13, 29-14
program element, 29-3
"start" looping, 29-12
test commands in, 29-14
"while" looping, 29-14
low-battery warning, A-4
lowercase letters
in units, 10-3
keying in, 1-5, 2-4
low-memory conditions, 5-18

M

mantissas, 4-3
math operations, 12-1
matrices, 8-1
assembling from sequences, 14-4
assembling from vectors, 14-3, 14-4
augmented, 14-18
calculations, 14-11
characterizing, 14-7
condition number, 14-9
conjugating, 14-14
decomposing, 14-21
determinants, 14-9
disassembling into elements, 14-4

disassembling into vectors, 14-4, 14-5
eigenvalues, 14-20
eigenvectors, 14-21
entering, 2-6
entering with MatrixWriter, 8-2, 14-1
extracting diagonal elements, 14-5
factoring, 14-21
Gaussian elimination, 14-18
identity, 14-2
ill-conditioned, 14-16
inserting rows, 14-5
inverses, 14-16
inverting, 14-10, 14-18
rank, 14-9
reconstructing from singular values, 14-22
row operations, 14-19
row-reduced echelon, 14-19, 14-20
singular, 14-16
singular values, 14-22
solving linear equations, 14-14
spectral radius, 14-9
statistical data, 21-1
trace, 14-9
transforming, 14-18
transposing, 14-10
MATRIX menu, 8-8
MatrixWriter
cell entry order, 8-5, 8-8
deleting columns, 8-7, 8-8
deleting rows, 8-7, 8-8
editing arrays, 8-5
entering arrays, 8-2, 14-1
entering vectors, 8-8
inserting columns, 8-6, 8-8
inserting rows, 8-7, 8-8
MATRIX menu, 8-8
setting cell width, 8-5, 8-8
statistical data, 21-4
statistical data and, 21-2
using, 8-2
maximum
in SOLVE application, 18-4
of graph, 22-11
memory
amount available, A-1
automatic cleanup, 5-1, A-3
backing up in backup object, 28-5
backing up to computer, 27-12
backup objects in, 28-3
clearing all, 5-17
expanding, 5-1, 28-15, 28-16
low-memory conditions, 5-18
out-of-memory condition, 5-20
plug-in cards, 5-1
RAM defined, 5-1
recovering, 5-17, 28-14
restoring from backup object, 28-6
restoring from computer, 27-13
ROM defined, 5-1
user memory defined, 5-1
MEMORY application, 1-6
menu descriptions
CST, 30-1
EDIT, 2-12
Interactive Stack, 3-7
I/O, 27-18
MATRIX, 8-8
MTH BASE, 15-1, 15-4
MTH HYP, 12-3
MTH PARTS, 12-9
MTH PROB, 12-4, 12-5
MTH REAL, 12-7, 12-9

MTH VECTR, 13-2, 13-4
PICTURE FCN, 22-10
PRG STK, 3-11
RULES, 20-21
UNITS Catalog, 10-1, 10-2, 10-6
UNITS Command, 10-1
VAR, 5-11
menu keys
labels, 1-4
Multiple-Equation Solver, 25-3, 25-7
SOLVE application, 25-3
using, 1-8
menu labels, 1-4
bar indicates submenu, 5-4
black and white, 25-8
bottom of display, 1-8
custom, 30-3
indicate related variables, 25-8, 25-10, 25-11
indicate variable states, 25-8
indicating submenu, 1-9
in Equation Library, 25-3
wrong colors, 25-11
menus
custom, 30-1
displaying, 1-9
labels in display, 1-4
list of, C-1
numbers for, C-1
pages in, 1-9
previous menu, 1-10
selecting functions from, 1-10
using, 1-10
messages
displayed in status area, 1-1
in SOLVE application, 18-3
list of, B-1
low-memory, 5-18
numbers for, B-1
Minehunt game, 25-14
minimum
in SOLVE application, 18-4
of graph, 22-11
mode names
1-User, 30-5
Algebraic-entry, 2-8
Algebraic/Program-entry, 2-9
Alpha-entry, 4-9
Cylindrical, 4-5, 12-11, 13-1
Degrees, 4-3
Engineering, 4-2
entry (EquationWriter), 7-2
Fix, 4-2
Grads, 4-3
Immediate-entry, 2-8
Polar, 4-4, 12-11, 13-1
Program-entry, 2-9, 29-5
Radians, 4-3
Rectangular, 4-4, 12-11, 13-1
Scientific, 4-2
scrolling (EquationWriter), 7-2
selection (EquationWriter), 7-2
Spherical, 4-5, 12-11, 13-1
Standard, 4-2
User, 30-5
modes
angle, 4-3
command line entry, 2-8
coordinate, 4-4, 12-11, 13-1
display format, 4-2
in EquationWriter, 7-2
program entry, 29-5
resetting all, 4-9
setting, 4-7
MODES application, 1-6

Mpar
created by Equation Library, 25-2, 25-6
MTH BASE menu, 15-1, 15-4
MTH HYP menu, 12-3
MTH PARTS menu, 12-9
MTH PROB menu, 12-4, 12-5
MTH REAL menu, 12-7, 12-9
MTH VECTR menu, 13-2, 13-4
Multiple-Equation Solver
can't find solution, 25-10
compared to SOLVE application, 25-3
function limitations, 25-9
internal process, 25-8, 25-10
interpreting results, 25-10
menu actions, 25-3, 25-7
menu label colors, 25-3, 25-8
messages, 25-10
units usage, 25-7
used by Equation Library, 25-2, 25-6
uses root-finder, 25-8

N

n1
general solutions (integer), 20-16
reserved variable, 5-6
names
action in programs, 29-2
duplicate, 5-4
entering, 5-14
evaluating, 5-13, 5-14
evaluating variables containing, 5-13
finding, 5-4
in custom menus, 30-1
in EquationWriter, 7-3
menu of, 5-11
preventing evaluation, 5-14
restrictions, 5-5
negative
of arrays, 14-11
of numbers, 12-13
newlines, 29-5
normal distribution, 12-5
norms (arrays), 14-8
numbers
action in programs, 29-2
appearance, 4-2
assembling complex from real, 12-14
converting complex to real, 12-14
converting real to complex, 12-14
converting to fractions, 16-5
disassembling complex into real, 12-14
exponential form, 2-2
in EquationWriter, 7-3
internal representation, 4-2
keying in, 2-1
random, 12-4
rounding, 12-9
truncating, 12-9
with units, 10-2

O

o (octal base marker), 15-1
objects, 2-1, A-2
actions in programs, 29-2
converting to graphics objects, 9-9
created from command line, 2-8
deleting from stack, 3-5
delimiters for, 2-6
determing valid types for input forms, 6-6
editing, 2-10

entering, 2-5
entering in programs, 29-5, 29-6
HP 48-HP 48 I/O, 27-1
HP 48-PC I/O, 27-9
in custom menus, 30-1
printing, 27-2
storing in variables, 5-6, 5-11
type numbers of, H-30
using in input forms, 6-3
viewing, 2-10
operations
categories of, 11-1
out-of-memory condition, 5-20
output grid
GRIDMAP plots, 23-34
output "slices"
YSLICE plots, 23-23
output surfaces
PR-SURFACE plots, 23-24
WIREFRAME plots, 23-24
over-determined systems, 14-14, 14-17
estimating solution, 14-15

P

packets (Kermit), 27-13
pages (menus)
displaying, 1-9
parabolas
plotting, 23-11
PARAMETRIC plots, 23-7
default step size, 23-8
display ranges, 23-8
EQ in, 23-7, 23-9
plotting ranges, 23-8, 24-3
TRACE mode, 23-8
parentheses
implicit, 7-6
in algebraics, 11-3
in complex numbers, 12-12
in EquationWriter, 7-5, 7-6
PARTS menu, 12-9
past-due alarms, 26-4
paths, 5-4
payment modes (TVM), 18-14, 18-19
payments (TVM)
amount of, 18-20
number of, 18-19
percent functions, 12-9
period
fraction mark, 4-6
permutations, 12-4
PICT, 9-6
copying to stack, 9-5
resetting, 22-14
storing picture in, 25-5
PICTURE environment
derivatives in, 22-10
function analysis, 22-9
integrals in, 22-10
keyboard in, 22-5
solving the current equation, 22-10
zooming, 22-6
PICTURE FCN menu, 22-10
pixels, 9-9
converting to user-units, 9-9
coordinates, 9-7
turning on and off, 9-8
placeholder variables
in SOLVE application, 18-3
PLOT application, 1-6
differential equation option, 19-7
PLOT input forms, 6-7
plot parameters
resetting, 22-14
setting, 22-13

plots
ΣDAT and, 22-1, 22-12
ΣPAR and, 22-16
adding graphical elements, 9-3
analyzing, 22-9
converting coordinates, 9-9
coordinate types, 9-7
equations, 22-1
expressions, 22-1
function analysis, 22-9
functions of two variables, 23-22
labeling coordinate axes, 24-1
output of two-variable plots, 23-22
pixel coordinates, 9-7
pixel operations, 9-8
plot types, 24-3
PPAR and, 22-13, 22-14
programs, 22-1, 24-2
ranges, 24-3
reconstructing, 24-7
resetting plot parameters, 22-14
restoring, 24-6
sampling grid, 23-22
saving, 24-6
saving in variables, 24-7
saving "reconstructable" plots, 24-7
setting plot parameters, 22-13
solving the current equation, 22-10
user-defined functions, 24-2
user-unit coordinates, 9-7
viewing plots stored in variables, 24-7
VPAR and, 22-15
zooming, 22-6
plot step size
CONIC plots, 23-13
FUNCTION plots, 23-3
PARAMETRIC plots, 23-8
POLAR plots, 23-5
TRUTH plots, 23-15
plotted functions
analyzing, 22-9
plotting ranges
display ranges and, 24-3
PARAMETRIC plots, 23-8
POLAR plots, 23-5, 23-6
TRUTH plots, 23-15
plotting solutions
differential equations, 19-7
stiff differential equations, 19-9
plot types
BAR, 23-19
CONIC, 23-11, 24-3
DIFF EQ, 19-7, 23-11
FUNCTION, 23-1
GRIDMAP, 23-22, 23-34
HISTOGRAM, 23-18
PARAMETRIC, 23-7, 24-3
POLAR, 23-4, 24-3
PR-SURFACE, 23-24, 23-35
PS-CONTOUR, 23-22, 23-30
SCATTER, 23-20
SLOPEFIELD, 23-22, 23-25
TRUTH, 23-14, 24-3
WIREFRAME, 23-24, 23-28
YSLICE, 23-23, 23-32
plug-in cards
application, 5-1
environmental limits, A-4
expanding RAM, 5-1
expanding ROM, 5-1
installing, 28-9, 28-12
new RAM cards, 28-10
nonapproved, 28-9

removing, 28-14
plug-in ports
installing cards, 28-9
list of backup objects, 28-4
removing cards, 28-14
searching, 28-4
testing, A-12
type of memory in, 28-5, 28-15
wildcards, 28-4
pointer (Interactive Stack), 3-6
Polar coordinate mode, 4-4, 12-11, 13-1
POLAR plots, 23-4
default step size, 23-5
display ranges, 23-5
EQ in, 23-4
plotting ranges, 23-5, 23-6, 24-3
TRACE mode, 23-6
polynomials
as approximations, 20-12, 20-15
converting to algebraic form, 18-11
evaluating, 18-11
finding from roots, 18-10
finding roots, 18-10
in EquationWriter, 7-6
Maclaurin series, 20-12
Taylor's, 20-12
using root-finder, 18-10
population statistics, 21-6
port 0
backing up memory into, 28-5
built-in independent memory, 28-2, 28-3
libraries in, 28-8
moving objects into, 28-16
restoring memory from, 28-6
port memory
moving objects into, 28-17
PPAR
plot parameters, 22-13
reserved variable, 5-6
resetting, 22-14
precedence
symbolic operators, 11-3
top-level function and, 7-10
unit operators, 10-2, 10-9
precision, 4-2
present value (TVM), 18-20
previous menu
displaying, 1-10
PRG annunciator, 1-3, 29-5
PRG STK menu, 3-11
principal solutions
equations and algebraics, 20-16
general solutions and, 20-16
specifying, 20-16
printers
infrared, 27-2, 27-3
printing objects, 27-2
printing
objects, 27-2
to serial port, 27-3
probability commands, 12-4, 12-5
problems, A-1
PROB menu, 12-4, 12-5
procedures
applying to lists, 17-5
defining, 29-16, 29-18
Program Development Link, 27-7
Program-entry mode, 2-9, 29-5
programs, 22-1, 29-1
actions for object types, 29-2
applying to lists, 17-4

are sequences of objects, 29-1, 29-2
"building-block", 29-5
calculation styles, 29-4
compared to libraries, 28-6
conditional structures, 29-10, 29-15
debugging, 29-8
editing, 29-8
elapsed time, 16-4
entering, 29-5
entering algebraics in, 2-9
entry modes, 2-9, 29-5
evaluating local variables, 29-17
evaluating variables containing, 5-13
executing, 29-6
flow of, 29-4
HALT annunciator, 29-9
halting, 29-9
in local variable structure, 29-3, 29-16
killing, 29-9
local variable structures, 29-3, 29-16
loop structures, 29-12
naming, 29-6
newlines in, 29-5
not evaluating local variables, 29-17
objects in, 29-2
on the stack, 29-5
plotting, 22-1, 24-2
resuming, 29-9
scope of local variables in, 29-18
single-step execution, 29-8, 29-9
solving, 18-2
stopping, 1-8, 29-7
storing, 29-6
structured, 29-4
structures in, 29-3
subroutines, 29-5
trapping errors, 29-15
user-defined functions, 29-19
viewing, 29-8
PR-SURFACE plots, 23-35
EQ in, 23-36
eyepoint, 23-37
output surfaces, 23-24
view volume, 23-37
PRTPAR
reserved variable, 5-6
PS-CONTOUR plots, 23-30
output grid, 23-22
purging
backup objects, 28-4
Equation Library variables, 25-5, A-3
memory, 5-17
variables, 5-10, 25-5, A-3

Q

quadratic equations
solving, 20-15

R

R∠Z annunciator, 12-11, 13-2
R∠∠ annunciator, 12-11, 13-2
RAD annunciator, 1-3, 4-3
radians
converting to degrees, 12-7
Radians angle mode, 4-3
Radians mode, 4-10
RAM cards
as independent memory, 28-15
as merged memory, 28-15
backing up memory, 28-5
battery (initial), 28-10

battery preserves memory, 28-15
battery type, A-5
expanding user memory, 5-1, 28-16
for backup objects, 28-3
free before removing, 28-14
freeing, 28-16, 28-17
initializing, 28-14
installing, 28-9, 28-12
memory types, 28-15
moving objects into, 28-17
new, 28-10
removing, 28-14
replacing battery, A-7
restoring memory, 28-6
testing, A-12
type of memory in, 28-5, 28-15
write-protect switch, 28-12, 28-17
random numbers, 12-4
REAL menu, 12-7, 12-9
real numbers
complex results, 12-13
converting to binary integers, 15-3
converting to complex, 12-14
converting to complex numbers, 12-14
converting to fractions, 16-5
recalling
backup objects, 28-3
HP 48 memory from computer, 27-13
last arguments, 3-5
last command lines, 2-10
memory from backup object, 28-6
user key assignments, 30-7
rectangles
drawing, 9-8
Rectangular coordinate mode, 4-4, 12-11, 13-1
Rectangular mode, 4-10
repair service, A-16
repeating alarms, 26-5
replace cursor, 2-12
reserved variables, 5-6
resetting
flags, 4-9
input form fields, 6-6
memory, 5-17
PICT, 22-14
plot parameters, 22-14
PPAR, 22-14
restoring
last stack, 3-6
root-finder
displaying, 18-5
halting and restarting, 18-5
used by Multiple-Equation Solver, 25-8, 25-10
roots, 18-2
in EquationWriter, 7-4
in PICTURE environment, 22-10
multiple, 25-11
rotate (binary integers), 15-5
rounding numbers, 12-9
row vectors, 8-1, 8-8
RULES menu, 20-21
Rules transformations, 20-27

S

s1
general solutions (+ or −), 20-16
reserved variable, 5-6
sample statistics, 21-6
sampling grids, 23-22

SCATTER plots, 23-17
  comparing to regression, 23-21
  from Statistics, 23-20
Scientific display mode, 4-2
scrolling in EquationWriter, 7-2, 7-10, 7-13
Selection environment
  editing subexpressions, 7-11, 7-12
  EquationWriter mode, 7-2, 7-13
  Rules transformations, 20-19
self-test, A-10
semicolon
  complex number separator, 12-12
sequences, 17-1
  finding first differences, 17-8
  finding product of elements, 17-8
  generating, 17-7
serial cable, 27-7
serial port
  connecting printer, 27-3
  for printing, 27-3
  testing, A-14
  wiring, 27-7
serial printers, 27-2, 27-3
service repair, A-16
setting flags, 4-8
shift (binary integers), 15-5
shift keys
  annunciators, 1-6
  canceling, 1-6
  in custom menus, 30-4
  keyboard and, 1-4
  operation, 1-4
significant digits
  displayed, 4-2
  rounding to, 12-9
sign reversal
  in SOLVE application, 18-4
single-step execution of programs, 29-8, 29-9
single-variable statistics, 21-6
SI units
  base units, 10-2
  converting to, 10-6
size
  of built-in ROM, 5-1
  of memory, 5-1, A-1
  of stack, 3-11
  of variables, 5-11
slope
  calculating, 22-10
SLOPEFIELD plots, 23-25
  EQ in, 23-26
  output grid, 23-22
Snedecor's F distribution, 12-5
SOLVE application, 1-6, 18-1
  bad guesses, 18-5
  compared to Multiple-Equation Solver, 25-3
  constant-valued equations, 18-5
  displaying root-finder, 18-5
  getting solutions with units, 18-6
  halting and restarting root-finder, 18-5
  interpreting intermediate guesses, 18-5
  interpreting results, 18-3, 18-4
  menu actions, 25-3
  messages, 18-3
  rearranging variables, 18-6
  sign reversal, 18-4
  SOLVR environment, 18-6
  used by Equation Library, 25-2

using units with, 18-6
SOLVE differential equation solver
accuracy of results, 19-5
STIFF solver, 19-4
SOLVE input forms, 6-7
SOLVEQN command, F-1
solver menu (Equation Library)
actions, 25-3, 25-7
solvers (Equation Library)
choosing, 25-2
compared, 25-3
starting, 25-2
SOLVR environment
compared with SOLVE application, 18-6
creating custom menus, 18-9
Multiple Equation Solver and, 18-8
options not in SOLVE, 18-8
solving series of equations, 18-8
using, 18-6, 18-7
special characters
keying in, 2-4
viewing, 2-4
Spherical coordinate mode, 4-5, 12-11, 13-1
Spherical mode, 4-10
stack
calculations on, 3-1, 29-4
chain calculations, 3-3
command line and, 1-4
deleting objects, 3-5, 3-7, 3-11
diagrams, 29-4
displaying, 1-8
duplicating entries, 3-4
dynamic size, 1-3
input forms and, 6-5
Interactive Stack, 3-7
last, 3-6
manipulating, 3-7, 3-11
moving objects, 3-7, 3-11
operation, 1-3, 3-1
pointer, 3-6
putting objects into algebraics, 7-11, 7-12, 20-20
recalling last arguments, 3-5
restoring last, 3-6
rolling objects, 3-7
saving as graphics object, 9-10
size of, 3-11
swapping levels, 3-4
viewing, 3-7
STACK application, 1-6
stack display
organization, 1-1
returning to, 1-8
viewing graphics objects, 9-10
stack displays
converting to graphics objects, 9-10
stack syntax, 3-1
in local variable structures, 29-4
user-defined functions, 11-8
Standard display mode, 4-2
"start" looping, 29-12
STAT application, 1-6
statistical data
editing, 21-4
entering, 21-1, 21-2
in ΣDAT, 21-1
plotting, 21-7, 21-11, 23-17
plot types, 23-17
population data, 21-6
probabilities, 12-5
sample data, 21-6
test statistics, 12-4

upper-tail probabilities, 12-5
Statistics
ΣDAT data, 21-1
ΣPAR parameters, 21-12
BAR plots, 23-19
current matrix, 21-1, 21-2
data structure, 21-1
editing data, 21-4
entering data, 21-1, 21-2
HISTOGRAM plots, 23-18
plotting data, 23-17
plot types, 23-17
population, 21-6
probabilities, 12-5
sample, 21-6
SCATTER plots, 23-20
single-variable, 21-6
statistical data, 21-1
test statistics, 12-4
upper-tail probabilities, 12-5
status area, 1-1, 5-3
stiff initial value differential equations
solving, 19-4
STK menu, 3-11
storing
memory in backup object, 28-5
memory on computer, 27-12
objects in variables, 5-6, 5-11
programs, 29-6
user key assignments, 30-6
strings
action in programs, 29-2
creating from algebraics, 7-14
in custom menus, 30-1
sending to serial port, 27-18
structured programming, 29-4
Student's t distribution, 12-5
subdirectories
in custom menus, 30-1, 30-3
subexpressions, 7-10, 20-18, 20-19
editing, 7-10, 20-20
putting on stack, 20-20
rearranging, 20-19
replacing, 7-12, 20-20
submenus
indicated by label bar, 5-4
selecting, 1-9
subroutines
debugging, 29-9
in programs, 29-5
single-step execution, 29-9
summations
in EquationWriter, 7-5
swapping stack levels, 3-4
SYMBOLIC application, 1-7
symbolic constants
evaluating, 11-5
flags affect, 11-5
symbolic integration patterns, 20-30
symbols (alpha keyboard), 2-3
syntax
algebraic, 11-2
stack, 11-3
system flags
checking for math errors, 18-5
system halt, 5-16
systems of equations
"best" solution, 14-15
Gaussian elimination, 14-18
over-determined, 14-14, 14-15, 14-17
solution accuracy, 14-17
solving, 14-12, 14-14, 14-15, 14-18

under-determined, 14-14, 14-15, 14-17

T

Taylor's polynomials
  and derivatives, 20-13
  calculating, 20-12
t distribution, 12-5
temperatures
  calculations, 10-11
  calculator limits, A-4
  converting, 10-10
  differences, 10-10, 10-11
  levels, 10-10, 10-11
  plug-in card limits, A-4
  units of measure, 10-10
test commands
  in conditional structures, 29-10
  in loop structures, 29-14
testing
  calculator, A-9
  flag states, 4-8
  pixels, 9-9
test statistics, 12-4
three-dimensional plots
  constraints, 23-25
  coordinates relative to display, 23-25
  GRIDMAP, 23-34
  PR-SURFACE, 23-35
  PS-CONTOUR, 23-30
  SLOPEFIELD, 23-25
  WIREFRAME, 23-28
  YSLICE, 23-32
ticks (system clock), 16-4
time
  arithmetic with, 16-4
  as ticks, 16-4
  calculations, 16-3
  converting formats, 16-3
  converting to number, 16-1, 16-3
  displaying, 4-10
  elapsed time, 16-4
  format options, 16-1, 26-1
  HMS format, 16-3
  setting the current, 16-3
TIME application, 1-7
timeout (serial), 27-18
too few arguments, A-3
top-level function, 7-10, 20-18
TRACE mode
  cursor coordinates and, 22-4
  FUNCTION plots, 23-3
  PARAMETRIC plots, 23-8
  POLAR plots, 23-6
transforming columns, 21-5
transforming rows, 21-5
transmit modes
  HP 48-to-HP 48, 27-1
trigonometric functions, 10-8, 12-2, A-2
truncating numbers, 12-9
TRUTH plots, 23-14
  default step size, 23-15
  display ranges, 23-15
  EQ in, 23-14
  plotting ranges, 23-15, 24-3
TVM calculations, 18-13
  payment modes, 18-14
  performing, 18-16
TVM menu, 18-20
two's complement, 15-2, 15-3
typing aids, 30-1

U

under-determined systems, 14-14, 14-17
  estimating solutions, 14-15
unit objects
  calculations with, 10-7

calculations with temperatures, 10-11
consistent units, 10-8
converting angular units, 10-7
converting units, 10-6
converting units of temperature, 10-10
creating, 10-3, 10-4, 11-1
creating graphics objects from, 9-9
creating in EquationWriter, 7-6
delimiters, 10-2
factoring units, 10-9
in algebraics, 10-9
in custom menus, 30-1
inverse units, 7-6, 10-3
numeric part, 11-1
precedence of delimiter, 10-9
precedence of unit operators, 10-2
prefixes for units, 10-4
units
affect results, 25-10
consistent, 10-8, 25-10
converting, 10-6
dimensionless, 10-7
Equation Library, 25-15
implied, 25-11
initializing in Equation Library, 25-2
SI vs. English, 25-2, 25-5, 25-11, A-3
solving for unknown variables with, 18-6
unexpected, 25-11
user-defined, 10-14, 25-15
wrong, A-3
UNITS application, 10-1
units catalog
Equation Library, 25-5
UNITS Catalog menu, 10-1, 10-2, 10-6
UNITS Command menu, 10-1
units of measure
based on SI units, 10-2
case-sensitive names, 10-3
converting, 10-6
converting angles, 10-7
converting temperatures, 10-10
deleting, 11-1
dimensionally consistent, 10-8
entering in EquationWriter, 7-6
factoring, 10-9
in calculations, 10-7, 10-11
in custom menus, 30-1
inverse, 7-6, 10-3
operators, 10-2
prefixes, 10-4
temperature differences, 10-11
temperature units, 10-10, 10-11
units options (Equation Library), 25-1
affect equation solution, 25-10
effect of no units, 25-11
uppercase letters
in units, 10-3
keying in, 2-4
upper-tail probabilities, 12-5
USER annunciator, 1-3, 30-5
user-defined derivatives, 20-11
user-defined functions, 11-6
arguments, 11-7, 11-8
creating, 11-7
differentiating, 20-11
executing, 11-7

internal structure, 29-18, 29-19
nesting, 11-8
plotting, 24-2
user-defined transformations, 20-27
user-defined units, 10-14, 25-15
user flags, 1-3, 4-9
user keys
activating, 30-5
assigning, 30-5
disabling, 30-7
editing assignments, 30-8
operation, 30-4
packing assignments, 30-8
recalling assignments, 30-7
unassigning, 30-6
user memory, 5-1
expanding, 28-15, 28-16
user modes
activating, 30-5
annunciators for, 1-3
assigning keys, 30-5
disabling keys, 30-7
getting unstuck, 30-8
operation, 30-5
unassigning keys, 30-6
user-unit coordinates, 9-7
user-units
converting to pixels, 9-9
ΣDAT
plots and, 22-1, 22-12
1USR annunciator, 1-3, 30-5

V

variable catalogs
Equation Library, 25-5
variables
action in programs, 29-2
creating, 5-6, 5-13, 11-4
directories in, 5-4
duplicate names, 5-4
editing, 2-10
entering names, 5-14
evaluating, 5-11, 5-13, 5-14
evaluating selectively, 20-17
evaluating variables containing, 5-13
finding, 5-4, A-3
HP 48-HP 48 I/O, 27-1
HP 48-PC I/O, 27-9
in custom menus, 30-1
in EquationWriter, 7-3
in other directories, 5-4
isolating in an algebraic, 20-15
menu of, 5-11
naming, 5-5
placeholder, 18-3
preventing evaluation, 5-14
protecting during I/O, 27-11
purging, 5-12, 25-5, A-3
quoted names, 5-14
recalling contents, 5-11
reserved names, 5-6
saving plots in, 24-7
separating into directories, 5-3
showing hidden, 20-17
solving for values, 22-10
solving symbolically, 20-14, 20-15
storing objects in, 5-6, 5-11
unquoted names, 5-14
viewing, 2-10
viewing plots stored in, 24-7
variables (Equation Library)
equation design, 25-8
initializing, 25-2
involved in solution, 25-8, 25-10, 25-11
not detected, 25-9

states, 25-8
too many known, 25-11
too many unknown, 25-10
unexpected solutions, 25-10
wrong states, 25-11
variables (Solver)
global, 25-2, 25-5, A-3
VAR menu, 5-4, 5-11
displays directories, 5-4
vectors, 8-1
angle between, 13-5
assembling, 13-3
calculations, 13-4
column, 8-1
coordinate modes, 4-4, 13-1
delimiters, 13-3
disassembling, 13-4
displayed, 4-4
displaying, 13-1
entering, 2-7, 8-4, 8-8, 13-3
internal representation, 4-4, 13-3
normalized, 13-3
row, 8-1
unit vector, 13-5
VECTR menu, 13-2, 13-4
view volume, 23-24
PR-SURFACE plots, 23-37
WIREFRAME plots, 23-28, 23-30
YSLICE plots, 23-33
VPAR
and plots, 22-15

W

warranty, A-15
where function
in EquationWriter, 7-6
"while" looping, 29-14
wildcards
backup objects, 28-4
in user-defined transformations, 20-27
WIREFRAME plots, 23-28
eyepoint, 23-30
output surfaces, 23-24
view volume, 23-28, 23-30
wordsize (binary)
bits lost, 15-2
recalling, 15-2
setting, 15-2
write-protect switch, 28-12, 28-17

X

XON/XOFF handshaking, 27-3, 27-17

Y

YSLICE plots, 23-32
animating, 23-33
output, 23-23
view volume, 23-33

Z

zero value
in SOLVE application, 18-4
zoom defaults
ZPAR and, 22-13
zooming, 22-6
selecting a zoom, 22-7
setting defaults, 22-7
ZPAR
zoom parameters, 22-13

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 of the this manual. If you can't find an answer, you can contact the Calculator Support Department:

Hewlett-Packard
Calculator Support
1000 N.E. Circle Blvd.
Corvallis, OR 97330, U.S.A.
(503) 715-2004 (Mon.–Fri., 8:00am–3:00pm Pacific time)
(503) 715-3628 FAX

For Hardware Service. See appendix A of this manual for diagnostic instructions and information on obtaining service. *But, before you send your unit for service, please call HP Calculator Support at the number listed below.*

Hewlett-Packard
Corvallis Service Center
1030 N.E. Circle Blvd., Bldg. 11
Corvallis, OR 97330, U.S.A.
(503) 715-2004 (HP Calculator Support)

If you are outside the United States, see appendix A for information on locating the nearest service center.

HP Electronic Information Service. This service provides information on HP 48 calculators and is free of charge—you pay only for the phone call or Internet service. There are two ways to connect:

- **via modem:** (503)715-4448. It operates at 2400/9600/14400 baud, full duplex, no parity, 8 bits, 1 stop bit.
- **via the Internet:** Access as hpcvbbs.external.hp.com (192.6.221.13) using telnet, ftp, or World-Wide-Web browser (http://hpcvbbs.external.hp.com/hp48g.html).

Part 1: Fundamentals

1: The Keyboard and Display
2: Entering and Editing Objects
3: Stack
4: Modes
5: Memory

Part 2: Special Entry Environments

6: Input Forms and Choose Lists
7: The EquationWriter
8: The MatrixWriter
9: Graphic Objects
10: Unit Objects

Part 3: Mathematical Commands

11: Using Mathematical Functions
12: Functions of Real and Complex Numbers
13: Vectors and Transforms
14: Matrices and Linear Algebra
15: Binary Arithmetic and Number Bases
16: Date, Time, and Fraction Arithmetic
17: Lists and Sequences

Part 4: Interactive Mathematical Applications

18: Solving Equations
19: Differential Equations
20: Calculus and Symbolic Manipulation
21: Statistics and Data Analysis
22: Plotting
23: Plot Types
24: Advanced Plot Options
25: The Equation Library

Part 5: Extending and Customizing the HP 48

26: Time Management
27: Transmitting and Printing Data
28: Libraries, Ports and Plug-In Cards
29: Programming the HP 48
30: Customizing the HP 48

Part 6: Appendixes

A: Support, Batteries, and Service
B: Error Messages
C: Menus
D: System Flags
E: Table of Units
F: Table of Built-In Equations
G: Operation Index
H: Stack Diagrams for Selected Commands

HEWLETT PACKARD

Part Number 00048-90126 Edition 8

For HP internal use only:
00048-90104 (English)
Printed in Singapore 12/94